Effective Anomaly Detection with Scarce Training Data

William Robertson*
wkr@eecs.berkeley.edu
UC Berkeley

Federico Maggi*
fmaggi@elet.polimi.it
Politecnico di Milano

Christopher Kruegel Giovanni Vigna
{chris,vigna}@cs.ucsb.edu
Computer Security Group
UC Santa Barbara

Abstract

Learning-based anomaly detection has proven to be an effective black-box technique for detecting unknown attacks. However, the effectiveness of this technique crucially depends upon both the quality and the completeness of the training data. Unfortunately, in most cases, the traffic to the system (e.g., a web application or daemon process) protected by an anomaly detector is not uniformly distributed. Therefore, some components (e.g., authentication, payments, or content publishing) might not be exercised enough to train an anomaly detection system in a reasonable time frame. This is of particular importance in real-world settings, where anomaly detection systems are deployed with little or no manual configuration, and they are expected to automatically learn the normal behavior of a system to detect or block attacks.

In this work, we first demonstrate that the features utilized to train a learning-based detector can be semantically grouped, and that features of the same group tend to induce similar models. Therefore, we propose addressing local training data deficiencies by exploiting clustering techniques to construct a knowledge base of well-trained models that can be utilized in case of undertraining. Our approach, which is independent of the particular type of anomaly detector employed, is validated using the realistic case of a learning-based system protecting a pool of web servers running several web applications such as blogs, forums, or Web services. We run our experiments on a real-world data set containing over 58 million HTTP requests to more than 36,000 distinct web application components. The results show that by using the proposed solution, it is possible to achieve effective attack detection even with scarce training data.

Keywords: *Anomaly detection, training data, web application.*

1 Introduction

The Internet has evolved from its humble beginnings at CERN in 1991 into a massive network of ubiquitous services that spans the globe and reaches an estimated 1.4 billion people [27]. The World Wide Web contains more than 100 million sites [46] and around 1 trillion unique URLs as indexed by Google [1]. Due to its pervasive nature, the Internet – and in particular the Web – has become a predominant medium for disseminating and collecting information. In fact, web applications have enjoyed immense popularity as an efficient means for providing services to users. For instance, Facebook has more than 250 million active users, uploading more than 1 billion photos *each month*, and Twitter distributed more than 3 million messages *per day* in March 2008 [34].

Unfortunately, applications have been found to contain many security vulnerabilities, due to a combination of unsafe development tools and a historical lack of security awareness among developers. In addition, the risks are magnified when vulnerable software is deployed in the context of the Web, since applications are typically widely accessible and often have access to sensitive information. These factors have naturally resulted in web-related vulnerabilities receiving substantial attention from the criminal underground [40]. As a consequence, the incidence of data breaches, online fraud, and other crimes resulting from the exploitation of web application vulnerabilities continues to rise [29,33], and, therefore, it is essential to protect applications and systems connected to the Internet against such attacks.

Anomaly detection has received much attention from the research community as an approach to detecting and preventing unknown attacks by monitoring a network's traffic [20,25,26,32,43,44,47] or a host's operating system [3,14,21,24,28,31,38,42]. Recently, anomaly-based techniques have also been shown to be effective against web-based threats [5, 15, 19, 30]. Effective anomaly detection systems are attractive because they consider the protected system as a black box. As a result, they can

be deployed in live environments without any *a priori* knowledge about the application.

Anomaly detection systems contain specifications, or models, of the normal behavior of the protected system, and consider deviations from the specifications to be evidence of malicious behavior. In contrast to signature-based systems, anomaly detectors have the desirable property that previously unknown attacks can be identified automatically. Though anomaly detection models can be manually specified by domain experts, this is a tedious, labor-intensive, and error-prone process. Therefore, most research has instead focused on applying machine learning techniques to automatically derive models of normal behavior from unlabeled training data. The term *normal behavior* generally refers to a set of characteristics (e.g., the distribution of the symbols of strings, or the mean and standard deviation of the values of numerical variables) extracted from data observed during a system's normal operation. For instance, such data could be the payloads of network packets, or HTTP requests and responses exchanged between a web server and clients. Those characteristics are used to build *models* of normal behavior. Learning-based anomaly detectors obviate the tedious and error-prone task of creating specifications, and, additionally, are able to adapt to the particular characteristics of the local environment. Therefore, anomaly detectors typically require only a modest initial configuration effort to provide effective attack detection.

In an ideal case, a learning-based anomaly detection system is deployed in front of a system and, in a completely automated fashion, learns the normal interaction between the system and its users. Once enough training data has been analyzed and the profiles for the monitored systems have been established, the anomaly detector switches to detection mode; it is then able to detect attacks that represent anomalies with respect to normal usage. These types of anomaly detection systems are extremely attractive to security officers and site administrators, who have neither the resources nor the skills to manually analyze applications composed of hundreds of components. Because of this, several commercial web application firewalls implement some form of machine learning to support anomaly detection [4, 6, 11].

Learning-based anomaly detectors are not without their drawbacks, however. In fact, these systems are known for their tendency to produce a non-negligible amount of false positives due to the difficulty of accurately modeling non-trivial domains. This is a limiting factor for the effectiveness of such systems [2, 13]. An additional limitation is that anomaly detection systems critically rely upon the quality of the training data used to construct their models. In particular, training sets must be free from attacks. Otherwise, the resulting models will be prone to false negatives, as attack manifestations will have been learned as normal behavior [8, 10, 17, 39].

Another limitation that is well-known in the research community is the difficulty of obtaining enough high-quality training data. Unfortunately, to our knowledge, no proposals exist that satisfactorily address the problem. In particular, our experiments suggest that web application component invocations are non-uniformly distributed. That is, relatively few components dominate, and the remaining components are accessed relatively infrequently. Thus, for those components, it is often impossible to gather enough training data to accurately model their normal behavior. We informally refer to the components that receive insufficient accesses as the "long tail" of a web application. Note that, however, this does not necessarily imply a power law distribution of these accesses. Nevertheless, components that are infrequently accessed lead to poor detection capabilities due to undertrained models (i.e., models possessing knowledge limited to the low number of samples they have been trained on).

This work addresses the problem of undertrained models by exploiting natural similarities among the modeled features. We demonstrate this hypothesis using the web applications context as a real-world example. In particular, we show that the values of the parameters extracted from HTTP requests can generally be categorized according to their type, such as an integer, date, or string. Moreover, our experiments demonstrate that parameters of similar type induce similar models of normal behavior. Taken together, these results can be leveraged to effectively supplement a lack of training data for one web application component with similar data from another component that has received more requests.

In this paper, we make the following contributions:

- We introduce the problem that arises from the fact that traffic is distributed in a non-uniform fashion, and we provide evidence that this occurs in the real world in the case of web applications.
- We propose an approach to address the problem of undertraining by using *global knowledge* built by exploiting similarities between web application parameters of similar type.
- We evaluate our approach on a large data set of real-world traffic from many web applications, and demonstrate how anomaly detectors can accurately model those components that would otherwise be associated with undertrained models.

The results of our experiments show that by using our approach, it is possible to improve the detection rate of undertrained models. In particular, web application

$$R_i = \left\{ \begin{array}{l} r_{i,1} = \texttt{/article}, \\ r_{i,2} = \texttt{/comments}, \\ r_{i,3} = \texttt{/comments/edit}, \\ r_{i,4} = \texttt{/account}, \\ r_{i,5} = \texttt{/account/password} \end{array} \right\}$$

Figure 1: Example resources comprising a web application.

components that have received *only a few dozen requests* can be protected almost as effectively as those that components have received thousands of requests.

2 Training data scarcity

To understand why the problem of undertrained models exists, we first present a general description of an anomaly detection system designed to protect web applications. While this description is based on the system described in [19], we believe it represents an accurate abstraction of web application anomaly detection. We then use these concepts to introduce the long-tail problem and its ramifications on the feasibility of constructing accurate models.

It is important to note that although there is no established architecture for these systems, a large portion of learning-based anomaly detectors are designed in a similar manner. In particular, they extract some significant features from the captured traffic, estimate the parameters of a set of pre-existing models (learning phase), and then use these models to analyze live traffic and recognize unexpected values of the selected features. Thus, the description that follows can easily be generalized. In addition, we remind the reader that a low detection accuracy due to undertraining may affect any type of learning-based protection system, rather than those detectors specifically designed for the web domain.

2.1 Web application anomaly detection

Without loss of generality, a set of web applications A can be organized into a set of *resource paths* or *components* R, and named *parameters* P. For example, a web application $a_i = \texttt{blog.example.com}$ might be composed of the resources shown in Figure 1.

In this example, resource path $r_{i,5}$ might take a set of parameters as part of the HTTP request: $P_{i,5} = \{p_{i,5,1} = \texttt{id},\ p_{i,5,2} = \texttt{oldpw},\ p_{i,5,3} = \texttt{newpw}\}$.

A generic learning-based application intrusion detection system operates by observing a sequence of *requests* $Q = \{q_1, q_2, \ldots\}$ issued by clients to the set of monitored applications. Each request $q \in Q$ is represented by the tuple $\langle a_i, r_{i,j}, P_q \rangle$, where P_q is a set of parameter name-value pairs such that $P_q \subseteq P_{i,j}$.

As highlighted in Figure 5, the initial *training* is performed offline. During this phase, the anomaly detector learns the behavior of the monitored web applications in terms of *models*. As new web application, resource path, and parameter instances are observed, the sets A, R, and P are updated. For each unique parameter $p_{i,j,k}$ observed in association with a particular application a_i and path $r_{i,j}$, a set of models that characterizes the normal behavior of various features of the parameter is constructed. The set of models associated with each unique parameter instance can be represented as a tuple $c_{(.)} = \langle m_1, m_2, \ldots, m_u, \ldots, m_U \rangle$, referred to as a *profile* or *model composition*. Therefore, for each application a_i and resource path $r_{i,j}$, a set $\mathcal{C}_{i,j}$ of model compositions is constructed, one for each parameter $p_{i,j,k} \in P_{i,j}$. The *knowledge base* of an anomaly detection system trained on web application a_i is denoted by $\mathcal{C}_{a_i} = \bigcup_j \mathcal{C}_{i,j}$. A graphical representation of how a knowledge base is modeled for multiple web applications is depicted in Figure 2.

To introduce and address the problem of undertraining, we leverage the set of models described in [19] as a real-world case. According to the system described in [19], a profile for a given parameter $p_{i,j,k}$ is the tuple $c_{i,j,k} = \langle m^{(\text{tok})}, m^{(\text{int})}, m^{(\text{len})}, m^{(\text{char})}, m^{(\text{struct})} \rangle$. $m^{(\text{tok})}$ models parameter values as a set of legal tokens (e.g., the set of of possible values for the parameter `gender` observed during training). $m^{(\text{int})}$ and $m^{(\text{len})}$ describe normal intervals for integers and string lengths, respectively, in a distribution-independent fashion using the Chebyshev inequality. $m^{(\text{char})}$ models character strings as a ranked frequency histogram, or *Idealized Character Distribution* (ICD), that are compared using the χ^2 or G tests. $m^{(\text{struct})}$ models sets of character strings by inducing a *Hidden Markov Model* (HMM). The HMM encodes a probabilistic grammar that can produce a superset of strings observed in a training set. Aside from the addition of $m^{(\text{int})}$, which is a straightforward generalization of $m^{(\text{len})}$ to numerical ranges, the interested reader may refer to [19] for further details.

After training, learning-based systems are typically switched to *detection mode*, which is performed online. The models trained in the previous phase are queried to determine whether or not the new parameters observed are anomalous. Without going into the details of a particular implementation, each parameter is compared to all the applicable models (e.g., an integer parameter is compared to $m^{(\text{int})}$, while a string parameter is compared to $m^{(\text{len})}$, $m^{(\text{char})}$, and $m^{(\text{struct})}$) and an aggregated anomaly score on the interval $[0, 1]$ is calculated by composing the values returned by the individual models. If the anomaly score is above a certain threshold, an alert is generated.

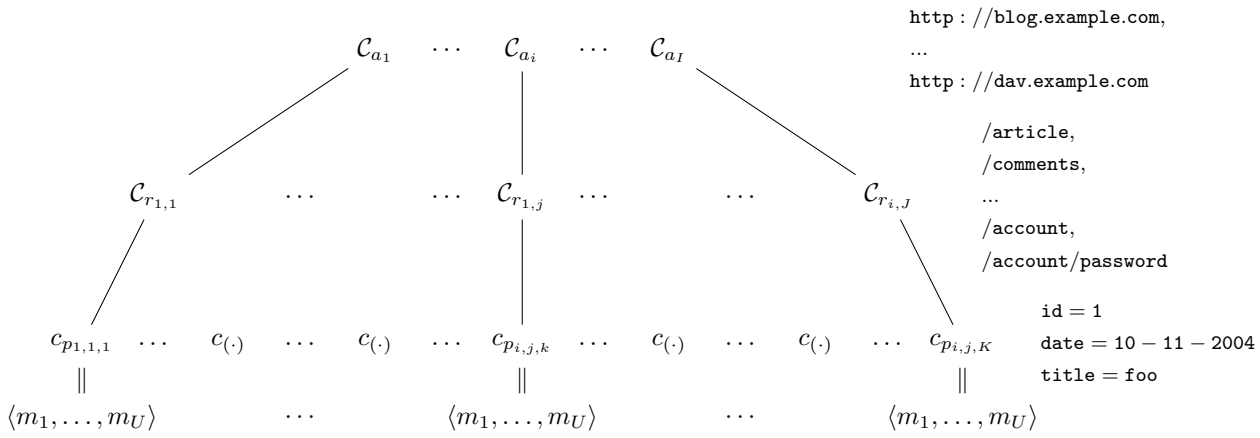


Figure 2: Overview of web application model construction.

2.2 The problem of non-uniform training data

Because learning-based detectors dynamically build specifications of normal behavior from training data, it is clear that the quality of the detection critically relies upon the quality of the training data. For instance, one requirement typically imposed upon a training set is that it should be *attack-free* – that is, it should not contain traces of malicious activity that would induce the resulting models to consider malicious behavior as normal. One solution to this issue is described in [8]. Another requirement that a training set should satisfy is that it should accurately represent the normal behavior of the modeled features. In some ways, this requirement is the dual of the previous one: training data should completely cover all aspects of normal behavior.

The difficulty of obtaining sufficient training data to accurately model application behavior is intuitively clear. We are, however, not aware of any solutions that can address this issue when insufficient training data is available. In a sense, this issue is similar to those addressed by statistical analysis methods with missing data [22]. Although a training procedure would benefit from such mechanisms, they require a complete redesign of the training algorithm specific to each model. Instead, a non-obtrusive approach that can improve an existing system without modifying the undertrained models is more desirable. Typically, anomaly-based detectors cannot assume the presence of a testing environment that can be leveraged to generate realistic training data that exercises the web application in a safe, attack-free environment. Instead, the anomaly detection system is deployed in front of live web applications with no *a priori* knowledge of the applications' components and their behavior. If anomaly-based detectors required manual or semi-automatic testing to be effective, their maintenance would be as tedious as that of misuse-based systems.

$$\begin{aligned} /\texttt{article} &= 475,000 \\ /\texttt{comments} &= 15,000 \\ /\texttt{comments/edit} &= 9,000 \\ /\texttt{account} &= 900 \\ /\texttt{account/password} &= 100 \end{aligned}$$

Figure 4: Example non-uniform web application request distribution.

In the case of low-traffic applications, problems arise if the rate of client requests is inadequate to allow models to train in a timely manner. However, even in the case of high-traffic applications, a large subset of resource paths might fail to receive enough requests to adequately train[1] the associated models. This phenomenon is a direct consequence of the fact that requests issued by clients often follow a non-uniform distribution. To illustrate this point, Figure 3 plots the normalized cumulative distribution function of web client resource path invocations for a variety of real-world, high-traffic web applications (details on this data are provided in Section 4). Although several applications have an approximately uniform client access distribution, a clear majority exhibit skewed distributions. Indeed, in many cases, a large percentage of resource paths receive a comparatively minuscule number of requests. Returning to the example resources shown in Figure 1, assuming an overall request volume of 500,000 requests per day, the example resource path set might result in the client access distribution shown in Figure 4.

Clearly, profiles for parameters to resource paths such as `/article` will likely receive sufficient training data.

[1] A more formal definition of the minimum amount of samples required for a complete training is provided in Section 3.1.

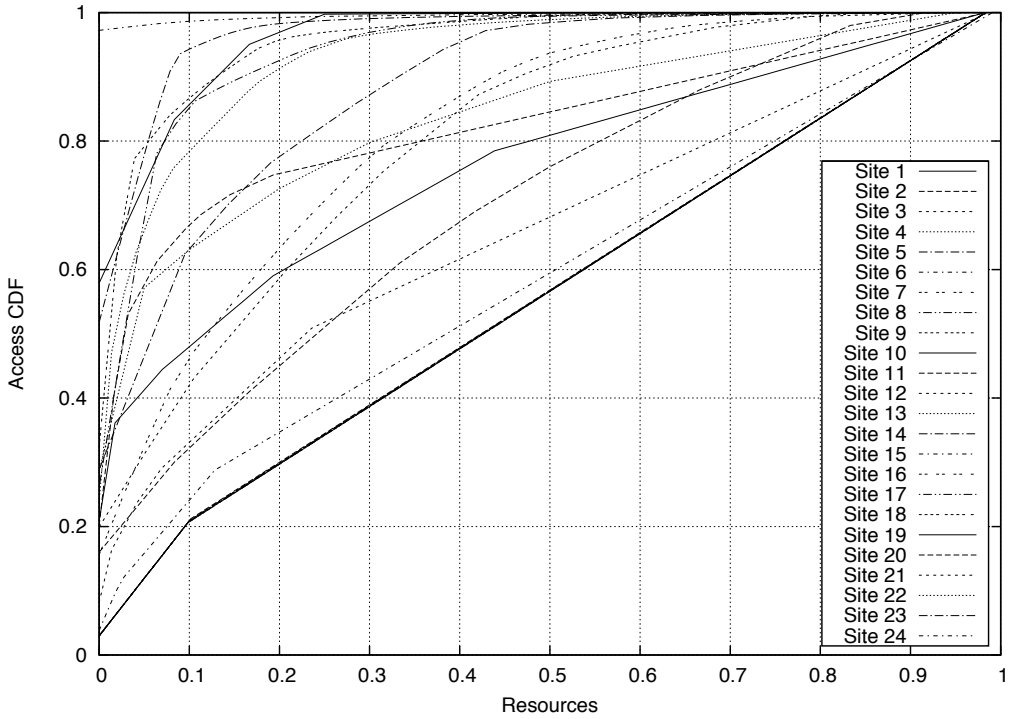


Figure 3: Web client resource path invocation distributions from a selection of real-world web applications.

This is not true, however, for profiles associated with paths such as `/account/password`. Further exacerbating the situation is the fact that a client request does not necessarily include all possible parameters.

The infeasibility for an anomaly detection system to accurately model a large subset of a web application is problematic in itself. We argue, however, that the impact of the problem is magnified by the fact that components of a web application that are infrequently exercised are also likely to contain a disproportionately large share of security vulnerabilities. This is a consequence of the reduced amount of testing that developers invariably perform on less prominent components of a web application, resulting in a higher rate of software defects. In addition, the relatively low request rate from users of the web application results in a reduced exposure rate for these defects. Finally, when flaws are exposed and reported, correcting the flaws may be given a lower priority than those in higher traffic components of a web application. An example is the authentication manager of a blog, which receives a low share of the traffic if compared with the component that handles the comments or the archive.

Therefore, we conclude that a mechanism to address the problem of model undertraining caused by the non-uniform distribution of training data is necessary for a web application anomaly detection system to provide an acceptable level of security.

3 Exploiting global knowledge

The lack of available training data is a fundamental obstacle when constructing accurate profiles for many parameters of a web application. Without a minimum number of requests to a given parameter, it is infeasible to construct models that encode a reasonably precise approximation of normal behavior.

We observe, however, that parameters associated with the invocation of components belonging to different web applications often exhibit a marked similarity to each other. Referring again to the example shown in Figure 1, many web applications take an integer value as a unique identifier for a class of objects such as a blog article or comment, as in the case of the `id` parameter. Many web applications also accept date ranges similar to the `date` parameter as identifiers or as constraints upon a search request. Similarly, as in the case of the `title` parameter, web applications often expect a short phrase of text as an input, or perhaps a longer block of text in the form of a comment body. One can consider each of these groupings of similar parameters as distinct *parameter types*, though this need not necessarily correspond to the concept of types as understood in the programming languages context.

The key insight behind our approach is that parameters of the same type tend to induce model compositions that are similar to each other in many respects. Consequently, if the lack of training data for a subset of the

components of a web application prevents an anomaly detection system from constructing accurate profiles for the parameters of those components, it is possible to *substitute* profiles for similar parameters of the same type that were learned when enough training data was available. It must be underlined that the substitution operates at the granularity of *parameters* rather than *requests* (which may contain more than one parameter). This increases the likelihood of finding applicable profile similarities, and allows for the substitution of models taken from radically different components. However, although the experiments we run on real-world data confirm that the aforementioned insight is realistic, our hypothesis might not hold in some very specific settings. Thus, to minimize the risks brought by migrating global knowledge across different deployments, we interpreted this result only as an insight and developed a robust criterion able to find similar profiles independently from the actual types of the modeled parameters.

Our approach is composed of three phases. The *first phase*, shown in Figures 5 and 6b, is an extension of the training procedure originally implemented in [19], where undertrained versions of profiles are recorded in addition to their final states. In the *second phase*, a global knowledge base of profiles $\mathcal{C} = \bigcup_{a_i} \mathcal{C}_{a_i}$ is constructed offline, where $\mathcal{C}_{a_i}$ are knowledge bases containing only well-trained, stable profiles (right side of Figure 6a) from anomaly detection systems previously deployed on a set of web applications $\bigcup_i a_i$. The left side of Figure 6a depicts the progressive construction of the knowledge base $\mathcal{C}^I = \bigcup_{a_i} \mathcal{C}^I_{a_i}$ of undertrained profiles(i.e., an index into $\mathcal{C}$, where $\mathcal{C}^I_{a_i}$ is a knowledge base of undertrained profiles from the web application a_i). Additionally, we define a mapping $f : \left\{\mathcal{C}^I\right\} \times \mathcal{C}_{a_i} \mapsto \mathcal{C}$ (shown as a dotted, directed arrow) between undertrained and well-trained profiles.

The *third phase* is performed online. For any new web application where insufficient training data is available for a component's parameter, the anomaly detector first extracts the undertrained profile c'. Then, the global knowledge base $\mathcal{C}$ is queried to find a similar, previously constructed profile $f\left(\mathcal{C}^I, c'\right) = c$. The well-trained profile c is then substituted for the undertrained profile c' in the detection process.

3.1 Phase I: Enhanced training

As a first step, we extended the anomaly detector described in [19] with a mechanism to generate undertrained profiles from a data set. These undertrained profiles are generated using the following procedure. Let $Q^{(p)}_{a_i} = \{q^{(p)}_1, q^{(p)}_2, \ldots\}$ denote a sequence of client requests containing parameter p for a given web application. Over $Q^{(p)}_{a_i}$, profiles are deliberately undertrained on randomly sampled κ-sequences, where κ can take values in $\bigcup_{i=0}^{8} 2^i$ (a discussion of appropriate values for κ is deferred until Section 3.4). Each of the resulting profiles is then added to a knowledge base $\mathcal{C}^I_{a_i}$. Note that the random sub-sampling is performed with the goal of inducing undertraining to show that clustering is feasible and leads to the desired grouping even – and especially – in the presence of undertraining.

In general, κ corresponds to a number of training samples that is considered insufficient to accurately characterize a feature. This, however, warrants a discussion of what is considered sufficient. An obvious choice is to fix a large, constant training phase length (e.g., 1000 requests). Unfortunately, an appropriate training phase length is dependent upon the complexity of modeling a given set of features. Therefore, we have developed an automated method that leverages the notion of *model stability* to determine when a model has observed enough training samples to accurately approximate the normal behavior of a parameter.

As new training samples are observed early in the training phase, the state of a model typically exhibits frequent and significant change as its approximation of the normal behavior of a parameter is updated. Informally, in an information-theoretic sense, the average information gain of each new training sample is high. As a model's state converges to a more precise approximation of normal behavior, its state gradually exhibits infrequent and incremental changes. In other words, the information gain of new training samples approaches zero, and the model stabilizes.

Each model monitors its stability during the training phase by maintaining a history of snapshots of its internal state. Periodically, a model checks if the sequence of deltas between each successive historical state is monotonically decreasing and whether the degree of change drops below a certain threshold. If both conditions are satisfied, then the model is considered stable. Let $\kappa^{(u)}_{\text{stable}}$ denote the number of training samples required for a model to achieve stability. A profile is considered stable when all of its constituent models are stable. Thus, the number of training samples required for a profile to achieve stability is given by

$$\kappa_{\text{stable}} = \max_{u \in U} \kappa^{(u)}_{\text{stable}}. \tag{3.1}$$

At the end of this phase, the final state of each well-trained, or stable, profile is stored in a knowledge base $\mathcal{C}_{a_i}$. Both $\mathcal{C}_{a_i}$ and $\mathcal{C}^I_{a_i}$ are collected from each web application, and serve as input to the next phase.

Instead of describing the internal stop criterion specific to each model, if any, we developed a model-agnostic minimization algorithm detailed in Section 3.4 (and evaluated in Section 4) that allows one to trade off

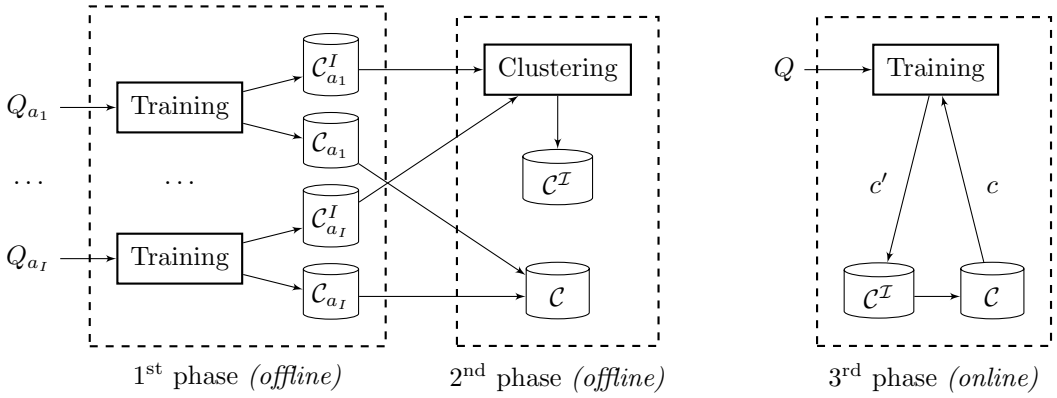


Figure 5: Overall procedure. Profiles, both undertrained and well-trained, are collected from a set of web applications. These profiles are processed offline to generate the global knowledge base $\mathcal{C}$ and index $\mathcal{C}^I$. At another web application, given an undertrained profile c', $\mathcal{C}$ can then be queried to find a suitable replacement profile c.

detection accuracy against the number of training samples available.

3.2 Phase II: Building profile knowledge bases

The second phase consists of processing the output of the first phase, namely the sets of knowledge bases of both undertrained and well-trained profiles learned from a variety of web applications. The goal is to create $\mathcal{C}$ and $\mathcal{C}^I$, global knowledge bases of well-trained and undertrained profiles, respectively, and a mapping between the two, allowing $\mathcal{C}^I$ to serve as an index to $\mathcal{C}$.

3.2.1 Constructing global knowledge base indices

The construction of the undertrained profile database $\mathcal{C}^I$ begins by merging a set of knowledge bases $\{\mathcal{C}^I_{a_1}, \mathcal{C}^I_{a_2}, \ldots, \mathcal{C}^I_{a_I}\}$ that have previously been built by a web application anomaly detector over a set of web applications $\bigcup_i a_i$ during the first phase. The profiles in $\mathcal{C}^I$ are then clustered to group profiles that are semantically similar to each other. Profile clustering is performed in order to time-optimize query execution when using $\mathcal{C}^I$ as an index into $\mathcal{C}$. The resulting clusters of profiles in $\mathcal{C}^I$ are denoted by $H^I = \bigcup_i h^I_i$. In this work, an agglomerative hierarchical clustering algorithm using group average linkage was applied, although the clustering stage is agnostic as to the specific algorithm. In principle, other mechanisms (e.g., Support Vector Machines, Locality-Sensitive Hashing) to find groups or classes of similar models based on their features can be used. We opted for a clustering algorithm as a proof-of-concept implementation of Phase II primarily due to its simplicity. For an in-depth discussion of clustering algorithms and techniques, we refer the reader to [45].

Central to any clustering algorithm is the distance function, which defines how similarities between the objects to be clustered are calculated. A suitable distance function must reflect the semantics of the objects under consideration, and should satisfy two conditions: 1) the overall similarity between elements within the same cluster is maximized, and 2) the similarity between elements within different clusters is minimized.

We define the distance between two profiles to be the sum of the distances between the models comprising each profile. More formally, the distance between the profiles c_i and c_j is defined as:

$$d(c_i, c_j) = \frac{1}{|c_i \bigcap c_j|} \sum_{m_i^{(u)}, m_j^{(u)} \in c_i \bigcap c_j} \delta_u \left(m_i^{(u)}, m_j^{(u)} \right), \tag{3.2}$$

where $\delta_u : \mathcal{M}_u \times \mathcal{M}_u \mapsto [0,1]$ is the distance function defined between models of type $u \in U = \{\text{tok}, \text{int}, \text{len}, \text{char}, \text{struct}\}$.

The token model $m^{(\text{tok})}$ is represented as a set of unique tokens observed during the training phase. Therefore, two token models $m_i^{(\text{tok})}$ and $m_j^{(\text{tok})}$ are considered similar if they contain similar sets of tokens. Accordingly, the distance function for token models is defined as the Jaccard distance [7]:

$$\delta_{\text{tok}} \left(m_i^{(\text{tok})}, m_j^{(\text{tok})} \right) = 1 - \frac{\left| m_i^{(\text{tok})} \bigcap m_j^{(\text{tok})} \right|}{\left| m_i^{(\text{tok})} \bigcup m_j^{(\text{tok})} \right|}. \tag{3.3}$$

The integer model $m^{(\text{int})}$ is parametrized by the sample mean μ and variance σ^2 of observed integers. Two integer models $m_i^{(\text{int})}$ and $m_j^{(\text{int})}$ are similar if these parameters are also similar. Consequently, the distance function for integer models is defined as:

$$\delta_{\text{int}} \left(m_i^{(\text{int})}, m_j^{(\text{int})} \right) = \frac{\left\| \frac{\sigma_i^2}{\mu_i^2} - \frac{\sigma_j^2}{\mu_j^2} \right\|}{\frac{\sigma_i^2}{\mu_i^2} + \frac{\sigma_j^2}{\mu_j^2}}. \tag{3.4}$$

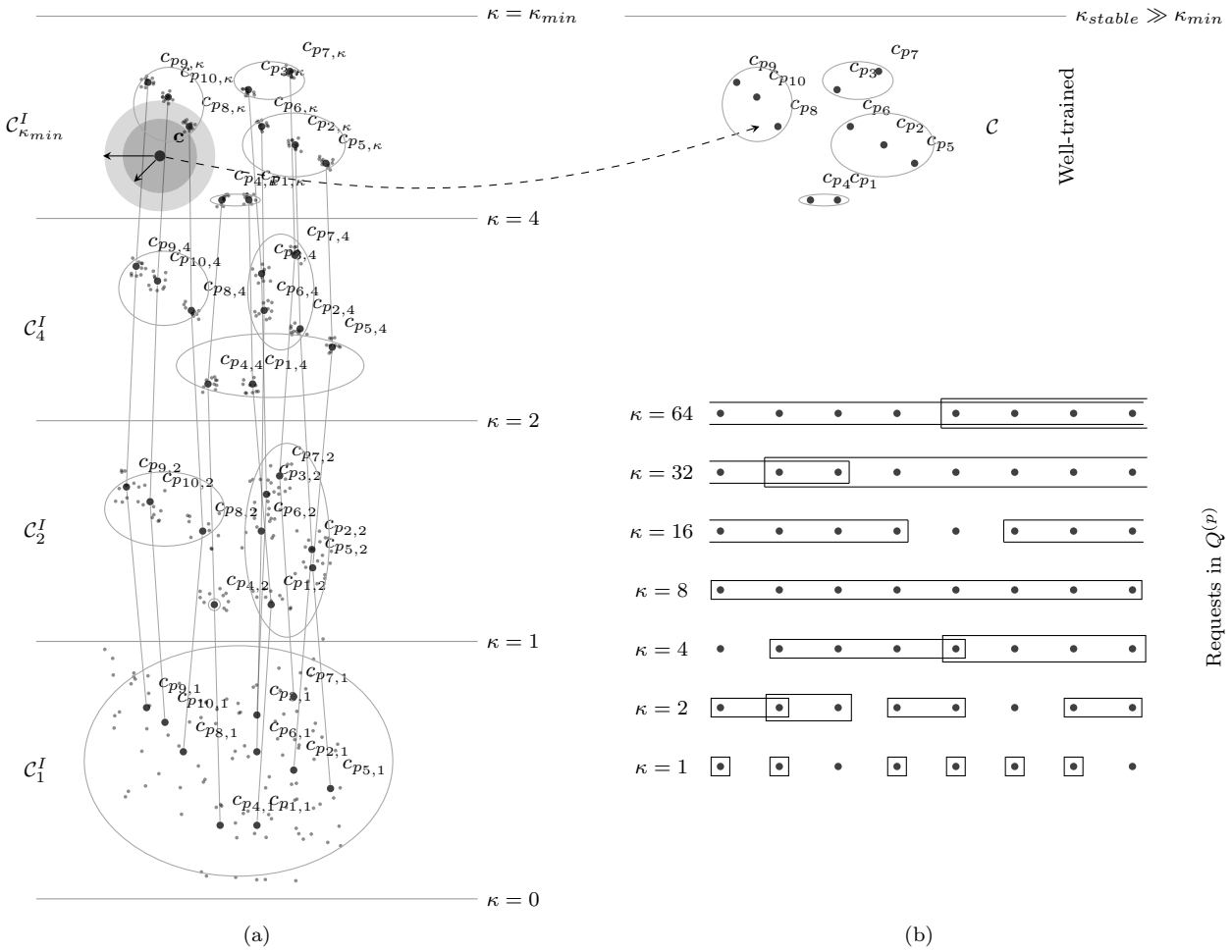


Figure 6: Global knowledge base indices (a) constructed by (b) sub-sampling the training set Q into small chunks of samples of progressively larger size (as detailed in Section 3.1). In (b), a dot indicates a sample for a certain parameter, and a full training set for one parameter corresponds to one horizontal row of dots. For instance, at $\kappa = 2$, the profiles for the same parameter are trained several times on 2-sized training subsets. This leads to the profiles indicated as small gray dots in (a) $\mathcal{C}^I_2$ and that, in this example, can be grouped into three clusters according to their similarity. As explained in Section 3.2.1, as κ increases, more accurate clusterings are achieved, and the undertrained knowledge base better resembles the well-trained one shown on the top-right.

As the length model is internally identical to the integer model, its distance function δ_{len} is defined similarly.

Recall that the character distribution model $m^{(\text{char})}$ learns the frequencies of individual characters comprising strings observed during the training phase. These frequencies are then ranked and coalesced into n bins to create an ICD. Two character distribution models $m_i^{(\text{char})}$ and $m_j^{(\text{char})}$ are considered similar if each model's ICDs are similar. Therefore, the distance function for character distribution models is defined as

$$\delta_{\text{char}}\left(m_i^{(\text{char})}, m_j^{(\text{char})}\right) = \sum_{l=0}^{n} \frac{\|b_i\,(l) - b_j\,(l)\|}{\max_{k=i,j}\, b_k\,(l)}, \quad (3.5)$$

where $b_i\,(k)$ is the value of bin k for $m_i^{(\text{char})}$.

The structural model $m^{(\text{struct})}$ builds an HMM by observing a sequence of character strings. The resulting HMM encodes a probabilistic grammar that can produce a superset of the strings observed during the training phase. The HMM is specified by the tuple $\langle \mathbb{S}, \mathbb{O}, M_{\mathbb{S}\times\mathbb{S}}, P\,(\mathbb{S}, \mathbb{O}), P\,(\mathbb{S})\rangle$. Several distance metrics have been proposed to evaluate the similarity between HMMs [16, 23, 36, 37]. Their time complexity, however, is non-negligible. Therefore, we adopt a less precise, but considerably more efficient, distance metric between two structural models $m_i^{(\text{struct})}$ and $m_j^{(\text{struct})}$ as the Jaccard distance between their respective emission sets

$$\delta_{\text{struct}}\left(m_i^{(\text{struct})}, m_j^{(\text{struct})}\right) = 1 - \frac{|\mathbb{O}_i \bigcap \mathbb{O}_j|}{|\mathbb{O}_i \bigcup \mathbb{O}_j|}. \quad (3.6)$$

3.2.2 Constructing a global knowledge base

Once a knowledge base of undertrained models $\mathcal{C}^I$ has been built, the next step is to construct a global knowledge base $\mathcal{C}$. This knowledge base is composed of the individual, well-trained knowledge bases from each web application as recorded during the first phase – that is, $\mathcal{C} = \bigcup_i \mathcal{C}_{a_i}$. Because undertrained profiles are built for each well-trained profile in $\mathcal{C}$, a well-defined mapping $f' : \mathcal{C}^I \mapsto \mathcal{C}$ (i.e., independent from the particular undertrained model in $\mathcal{C}_{a_i}$, as defined by f) exists between $\mathcal{C}^I$ and $\mathcal{C}$. Therefore, when a web application parameter is identified as likely to be undertrained, the corresponding undertrained profile c' can be compared to a similar undertrained profile in $\mathcal{C}^I$, that is then used to select a corresponding stable profile from $\mathcal{C}$.[2]

3.3 Phase III: Mapping undertrained profiles to well-trained profiles

With the construction of a global knowledge base $\mathcal{C}$ and an undertrained knowledge base $\mathcal{C}^I$, we can perform online querying of $\mathcal{C}$. That is, given an undertrained profile from an anomaly detector deployed over a web application a_i, the mapping $f : \left\{\mathcal{C}^I\right\} \times \mathcal{C}_{a_i} \mapsto \mathcal{C}$ is implemented as follows. A nearest-neighbor match is performed between $c' \in \mathcal{C}_{a_i}$ and the previously constructed clusters H^I from $\mathcal{C}^I$ to discover the most similar cluster of undertrained profiles. This is done to avoid a full scan of the entire knowledge base, which would be prohibitively expensive due to the cardinality of $\mathcal{C}^I$.

Then, using the same distance metric defined in Equation (3.2), a nearest-neighbor match is performed between c' and the members of H^I to discover the undertrained profile c^I at minimum distance from c'. If multiple nearest-neighbors are found, then one is chosen at random. Finally, the global, well-trained profile $f'\left(c^I\right) = c$ is substituted for c' for the web application a_i.

To make explicit how global profiles can be used to address a scarcity of training data, consider the example of Figure 4. Since the resource path `/account/password` has received only 100 requests, the profiles for each of its parameters $\{\texttt{id}, \texttt{oldpw}, \texttt{newpw}\}$ are undertrained. According to our confidence metric defined in Section 3.1, models that have received less than 1000 samples are likely undertrained (i.e., with a confidence close to zero). In the absence of a global knowledge base, the anomaly detector would provide no protection against attacks manifesting themselves in the values passed to any of these parameters. Also, the system may report too many false positives due to a lack of model generalization. If, however, a global knowledge base and index are available, the situation is considerably improved. Given $\mathcal{C}$ and $\mathcal{C}^I$, the anomaly detector can simply apply f to each of the undertrained parameters to find a well-trained profile from the global knowledge base that accurately models a parameter with similar semantics, even when the model is for *another web application*. Then, these profiles can be substituted for the undertrained profiles for each of $\{\texttt{id}, \texttt{oldpw}, \texttt{newpw}\}$. As will be demonstrated in the following section, the substitution of global profiles provides an acceptable detection accuracy for what would otherwise be a completely unprotected component (i.e., without a profile, none of the attacks against that component would be detected). If no matching well-trained profiles can be found, then the undertrained profile is used until the knowledge base is updated at a later time. This ensures that our approach can be deployed transparently and is adopted only when applicable, incurring no additional risk or overhead to the existing detection procedure.

3.4 Mapping quality

The selection of an appropriate value for κ is central to both the efficiency and the accuracy of querying $\mathcal{C}$. Clearly, it is desirable to minimize κ in order to be able to index into $\mathcal{C}$ as quickly as possible once a parameter has been identified to be subject to undertraining at run-time. On the other hand, setting κ too low is problematic. In fact, as Figure 6a indicates, for low values of κ, profiles are distributed with relatively high uniformity within $\mathcal{C}^I$, such that clusters in $\mathcal{C}^I$ are significantly different than clusters of well-trained profiles in $\mathcal{C}$. Therefore, slight differences in the state of the individual models can cause profiles that are close in $\mathcal{C}^I$ to map to radically different profiles in $\mathcal{C}$. As $\kappa \to \kappa_{\text{stable}}$, however, profiles tend to form semantically-meaningful clusters, and tend to approximate those found in $\mathcal{C}$. Therefore, as κ increases, profiles that are close in $\mathcal{C}^I$ become close in $\mathcal{C}$ according to f – in other words, f becomes *robust* with respect to model semantics.[3]

A principled criterion is required for balancing quick indexing against a robust profile mapping. To this end, we first construct a candidate knowledge base $\mathcal{C}^I_\kappa$ for a given κ. Additionally, we cluster the profiles in $\mathcal{C}$ as in the case of the undertrained knowledge base. Then, we define a *robustness metric* as follows. Recall that $H^I = \bigcup_i h_i^I$ is the set of clusters in $\mathcal{C}^I$, and let $H = \bigcup_i h_i$ be the set of clusters in $\mathcal{C}$. Let $g : H^I \mapsto \mathbb{Z}^+$ be a mapping from an undertrained cluster to the maximum number of

[2]Note that f' is a generalization of f.

[3]Our use of the term "robustness" is related, but not necessarily equivalent, to the definition of robustness in statistics (i.e., the property of a model to perform well even in the presence of small changes in the underlying assumptions.)

elements in that cluster that map to the same cluster in $\mathcal{C}$. The robustness metric ρ is then defined as

$$\rho\left(\mathcal{C}^I\right) = \frac{1}{|\mathcal{C}^I|}\sum_i g\left(h_i^I\right). \tag{3.7}$$

With this metric, an appropriate value for κ can now be chosen as

$$\kappa_{\text{min}} = \min_{\kappa}\left(\rho\left(\mathcal{C}_\kappa^I\right) \geq \rho_{\text{min}}\right), \tag{3.8}$$

where ρ_{min} is a minimum robustness threshold.

4 Evaluation

The goal of this evaluation is three-fold. First, we investigate the effects of profile clustering, and support the notion of parameter types by examining global knowledge base clusters. Then, we study how the quality of the mapping between undertrained profiles and well-trained profiles improves as the training slice length κ is increased. Finally, we present results regarding the accuracy of a web application anomaly detection system incorporating the application of a global knowledge base to address training data scarcity.

The experiments that follow were conducted using a data set drawn from real-world web applications deployed on both academic and industry web servers. Examples of representative applications include payroll processors, client management, and online commerce sites. For each application, the full content of each HTTP connection observed over a period of approximately three months was recorded. A portion of the resulting flows were then filtered using Snort to remove known attacks. In total, the data set contains 823 distinct web applications, 36,392 unique components, 16,671 unique parameters, and 58,734,624 HTTP requests.[4]

4.1 Profile clustering quality

To evaluate the accuracy of the clustering phase, we first built a global knowledge base $\mathcal{C}$ from a collection of well-trained profiles. The profiles were trained on a subset of the aforementioned data, containing traffic involving a wide variety of web applications. This subset was composed of 603 web applications, 27,990 unique resource paths, 9,023 unique parameters, and 3,444,092 HTTP requests. The clustering algorithm described in Section 3.2.2 was then applied to group profiles. Sample results from this clustering are shown in Figure 7b. Each leaf represents a profile. The name of the parameter and samples values observed during training are included to give an idea of the parameter's "type." As κ increases, profiles are clustered more accurately.

As the figure indicates, the resulting clusters in $\mathcal{C}$ are accurately grouped by parameter type. For instance, date parameters from different web applications belong a single hierarchy, while unstructured text strings are grouped into a separate hierarchy.

4.2 Profile mapping robustness

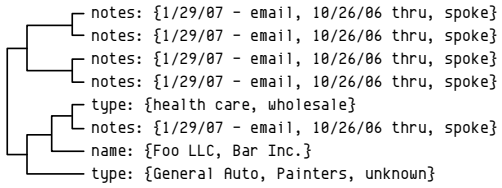
Recall that in order to balance the robustness of the mapping f between undertrained profiles and global profiles against the speed with which undertraining can be addressed, it is necessary to select an appropriate value for κ. To this end, we generated undertrained knowledge bases for increasing values of $\kappa = 1, 2, 4, 8, 16, 32, 64$ from the same data set used to generate $\mathcal{C}$, following the procedure outlined in Section 3.2.1. The resulting hierarchical clusters for various κ are presented in Figure 7c, 7d, 7a.

At low values of κ (e.g., Figure 7c), the clustering process exhibits non-negligible systemic errors. For instance, the parameter `stat` should be clustered as a token set of states, but instead is grouped with unstructured strings such as cities and addresses. A more accurate clustering would have dissociated the token and string profiles into well-separated sub-hierarchies.

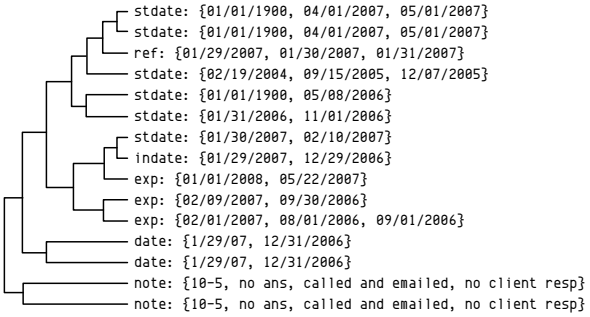
As shown in Figure 7d, larger values of κ lead to more semantically meaningful groupings. Some inaccuracies are still noticeable, but the clustering is significantly better that the one obtained at $\kappa = 8$. A further improvement in the clusters is shown in Figure 7a. At $\kappa = 64$, the separation between dates and unstructured strings is sharper; except for one outlier, the two types are recognized as similar and grouped together in the early stages of the clustering process.

Figure 8 plots the profile mapping robustness ρ against κ for different cuts of the clustering hierarchy, indicated by D_{max}. D_{max} is a threshold representing the maximum distance between two clusters. For low D_{max}, the "cut" will generate many clusters with a few elements. On the other hand, for high values of D_{max}, the algorithm will tend to form less clusters, each having a larger number of elements. In general, D_{max} can influence the result of a clustering algorithm significantly. Figure 8, however, shows two important properties of our technique. First, it demonstrates that the robustness is fairly insensitive to D_{max}. Second, the robustness of the mapping increases with κ until saturation at $32 \leq \kappa \leq 64$. This not only confirms the soundness of the mapping function, but it also provides insights on the appropriate choice of κ_{min} to minimize the delay to global profile lookup while maximizing the robustness

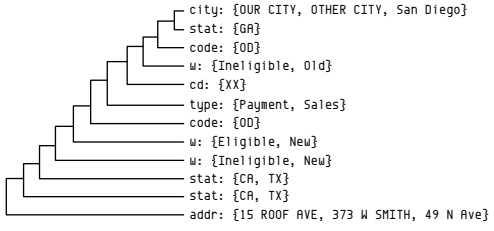
[4] Unfortunately, due to contractual agreements, we are unable to disclose specific information identifying the web applications themselves.



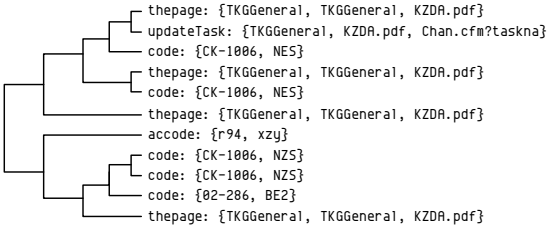
(a) $\kappa = 64$



(b) $\kappa_{\text{stable}} \simeq 10^3$



(c) $\kappa = 8$



(d) $\kappa = 32$

Figure 7: Graphical representation of the various steps of the hierarchical clustering of $\mathcal{C}$, (a-b), and $\mathcal{C}^I$, (c-d), at various κ. Each leaf represents a profile. The name of the parameter and samples values observed during training are included to give an idea of the parameter's "type." We note that these sample values are shown for visualization purposes only, and are never taken into account by the clustering algorithm. As κ increases, profiles are clustered more accurately.

of the mapping.

4.3 Detection accuracy

Having studied the effects of profile clustering and varying κ upon the robustness of the profile mapping f, a separate experiment was conducted to evaluate the detection accuracy of an anomaly detector incorporating the knowledge bases $\mathcal{C}$ and $\mathcal{C}^I$ constructed in the previous experiments. In particular, the goal of this experiment is to demonstrate that an anomaly detector equipped with a global knowledge base exhibits an improved detection accuracy in the presence of training data scarcity. As detailed in the following, we measured the baseline accuracy, on the same system and under the same conditions, with all the models undertrained.

The data used in this experiment was a subset of the full data set described above, containing traffic from one related set of web applications implementing online commerce sites. This data set was completely disjoint from the one used to construct the global knowledge base and its indices, to prevent any potential for the substitution of profiles from the same application. Additionally, the use of a global knowledge base generated from many types of web applications to address a lack of training data for a specific web application mirrors the intended usage of the technique. In total, this data set consisted of 220 unique real-world web applications, 8,402 unique resource paths, 7,648 distinct parameters, and 55,290,532 HTTP requests.

The threat model we assume is that of an attacker attempting to compromise the confidentiality or integrity of data exposed by a web application by tampering with request parameters.[5] Therefore, a set of 100,000 attacks was introduced into the data set. These attacks were real-world examples and variations upon cross-site scripting (XSS), SQL injection, command execution exploits, and other attacks that manifest themselves in request parameter values.[6] Examples of these attacks include:

- malicious code inclusion: `<script src="http://example.com/malware.js"></script>`;
- bypassing login authentication: `' OR 'x'='x'--`;

[5] Although the anomaly detector used in this study is capable of detecting more complex session-level anomalies, we restrict the threat model to request parameter manipulation because we do not address session profile clustering in this work.

[6] These attacks remain the most common attacks against web applications. However, both the anomaly detector and the improvement we designed apply to any malicious activity caused by modifications to HTTP requests, and therefore we by no means limit our scope to these classes of attacks.

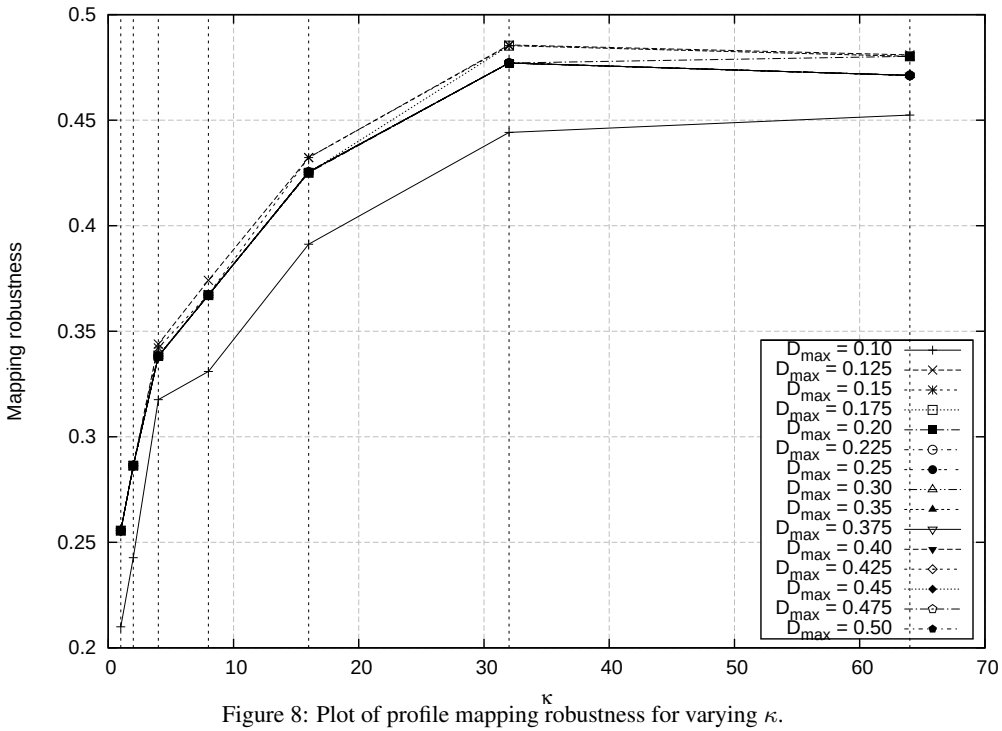


Figure 8: Plot of profile mapping robustness for varying κ.

- command injection: `; cat /etc/passwd | mail attacker@gmail.com #`.

To establish a worst-case bound on the detection accuracy of the system, profiles for each observed request parameter were deliberately undertrained to artificially induce a scarcity of training data for *all* parameters. That is, for each value of $\kappa = 1, 2, 4, 8, 16, 32, 64$, the anomaly detector prematurely terminated profile training after κ samples, and then used the undertrained profiles to query $\mathcal{C}$. The resulting global profiles were then substituted for the undertrained profiles and evaluated against the rest of the data set. The sensitivity of the system was varied over the interval $[0, 1]$, and the resulting ROC curves for each κ are plotted in Figure 9.

It must be noted that, as shown in Section 4.2, with $\kappa = 1$ our system cannot reliably find semantically similar well-trained models. As κ increases, so does the quality of the global profiles returned by the querying process. In particular, this increase in quality closely follows the mapping robustness plot presented in Figure 8. As predicted, setting $\kappa = 32, 64$ leads to fairly accurate global profile selection, with the resulting ROC curves approaching that of fully-trained profiles. This means that even if the component or, in general, a parameter of a web application has received only a few requests (i.e., 64), by leveraging a global knowledge base, it is possible to achieve effective attack detection. As a consequence, our approach can improve the effectiveness of real-world web application anomaly detection systems. Clearly, the detection accuracy will improve as more training samples (e.g., 128, 256) become available. However, the goal of this experiment was to evaluate such an improvement with a very limited training set, rather than showing the detection maximum accuracy achievable.

One concern regarding the substitution of global profiles for local request parameters is that a global profile that was trained on another web application might not detect valid attacks against the undertrained parameter. Without this technique, however, recall that a learning-based web application anomaly detector would otherwise have no effective model whatsoever, and, therefore, the undertrained parameter would be unprotected by the detection system. Furthermore, the ROC curves demonstrate that while global profiles are in general not as precise as locally-trained models, they do provide a significant level of detection accuracy.[7] More precisely, with $\kappa = 1$, undertraining condition and system off, only 67.5% of the attacks are detected, overall, with around 5% of false positives. On the other hand, with $\kappa = 64$ (undertraining and system on), more than 91% of the attacks are detected with less than 0.2% of false positives (*vs.*, 0.1% of false positives in the case of no undertraining and system off). Therefore, we conclude that, assuming no mistrust among the parties that share the

[7]Note that if global profiles were found to be as accurate as local profiles, this would constitute an argument against site-specific learning of models, since in that case, models could be trained for one web application and applied directly to other web applications.

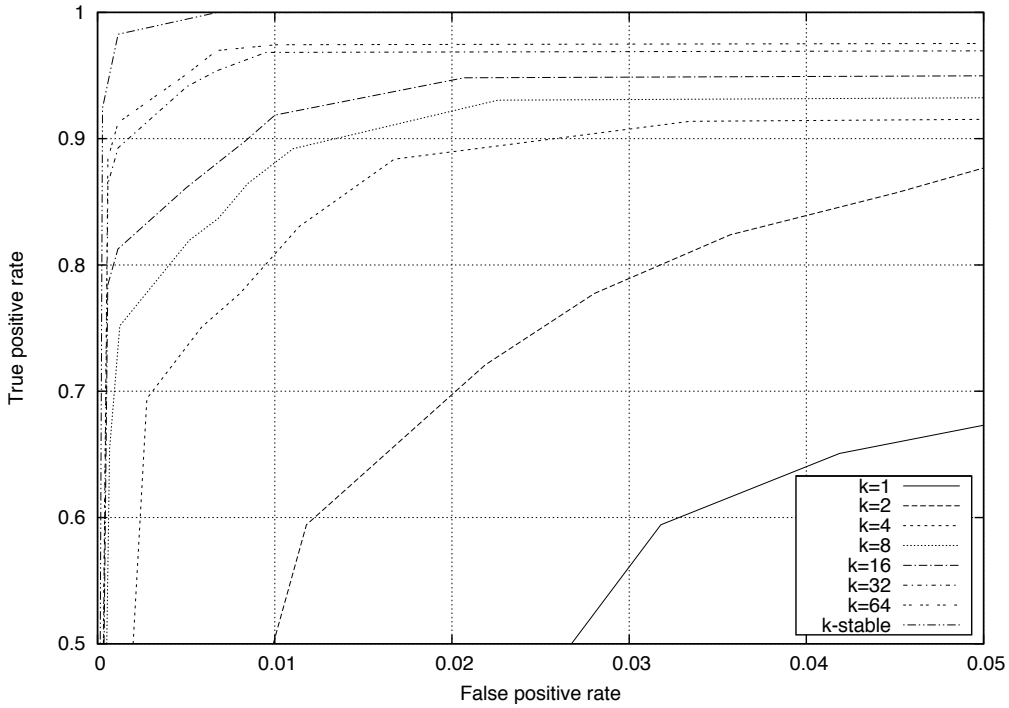


Figure 9: Global profile ROC curves for varying κ. In the presence of severe undertraining ($\kappa \ll \kappa_{\text{stable}}$), the system is not able to recognize most attacks and also reports several false positives. However, as κ increases, detection accuracy improves, and approaches that of the well-trained case ($\kappa = \kappa_{\text{stable}}$).

global knowledge base, our approach is a useful technique to apply in the presence of undertrained models and, in general, in the case of training data scarcity.

5 Related work

Anomaly detection has a rich history that dates back to Denning's seminal paper on intrusion detection [9]. It has been extensively applied to modeling legal sequences of system calls [12, 42] as well as other system call features [28]. Network-based anomaly detection has also been extensively studied [21, 41].

In [24], the authors further exploit the use of statistical learning procedures to build models of normal sequences of system calls in the Linux kernel. Here, ad-hoc distances are proposed to perform clustering of system calls in order to identify types of similar calls. Thus, the system calls are "compacted" and the reduced size of the input makes it feasible to train Markov chains that attempt to capture the behavior of each host application. Probabilistic thresholds are then used to detect deviating behaviors.

Anomaly detection techniques have also been applied at the application level. In [44], Wang and Stolfo describe PAYL, a service-agnostic anomaly detection system that models normal behavior by recording byte frequencies of network streams. An extension of the technique to higher-order n-grams is introduced in [43]. This system is also notable for its inclusion of a stop criterion for ending model training by estimating the likelihood of observing new n-grams; this approach is related to the notion of model stability described in Section 3.1.

One of the first applications of anomaly detection techniques to the web domain is described in [19], in which a multi-model approach to characterizing the normal behavior of web application parameters is shown to reliably detect web attacks while maintaining a low false positive rate. In [18], the use of Bayesian networks is proposed to compose models and express inter-model dependencies in order to further reduce false positive rates. A system for clustering anomalies and classifying clusters according to the type of attack is proposed in [30] to improve the explanatory power of web application anomaly detection as well as further reduce their false positive rate.

A recent effort on addressing training set deficiencies has been proposed in [8]. In this work, a sanitization phase is first performed to remove suspected attacks and other abnormalities from the data. Instead of creating one model instance, a set of "micro-models" is trained against disjoint subsets of the training data. These micro-models are then subject to one of several voting schemes to recognize and cull outliers that may represent attacks. While this work is somewhat related to

ours in that slices of training data are considered, the application of micro-models is intended to ensure that attacks are not present in training data. In contrast, our approach is complementary in the sense that it is focused on addressing the lack of training data.

In [13], the authors propose a cluster-based anomaly detection system as a means of reducing false positives. The system accomplishes this by clustering similar behavioral profiles for individual hosts using the k-means algorithm, although the exact distance metric used was not explicitly given. Then, alerts are generated according to a voting scheme, where the causal event for an alert is evaluated against behavior profiles from other members of that host's cluster. If the event is deemed anomalous by all members of the cluster, an alert is generated. Though this system shares the element of profile clustering with our work, the scope of the clustering is limited to end hosts connected to a single switch, while our work clusters across a large population of web application profiles. Also, the models operate over coarse network statistics such as the average number of hosts contacted per hour, the average number of packets exchanged per hour, and the average length of packets exchanged per hour. Our system, on the other hand, considers fine-grained features specific to web applications. Finally, our system does not require the use of a distributed voting scheme, which incurs additional overhead.

A recent proposal for the anomaly-based detection of web-based attacks is presented in [35]. In this work, a mixture of Markov chains incorporating n-gram transitions is used to model the normal behavior of HTTP request parameters. The resulting system attains a high detection accuracy for a variety of web-based attacks. In contrast to our work, however, a single mixture is learned for an entire web application. Additionally, the proposed system utilizes a supervised learning algorithm (i.e., attacks must be labeled as such in the training set), whereas ours operates on unlabeled training data.

6 Conclusions

In this work, we have identified that non-uniform web client access distributions cause model undertraining. This is an issue that must be addressed by web application anomaly detection systems in order to provide a reasonable level of security. The impact of this issue is particularly relevant for commercial web-based anomaly detection systems and web application firewall, which have to operate in real-world environments where sufficient training data might be unavailable within a reasonable time frame.

We propose the use of global knowledge bases of well-trained, stable profiles to remediate a local scarcity of training data by exploiting global similarities in web application parameters. We have evaluated the efficacy of this approach over an extensive data set collected from real-world web applications. We found that although using global profiles does result in a small reduction in detection accuracy, the resulting system, when given appropriate parameters, does provide reasonably precise modeling of otherwise unprotected web application parameters.

As future work, we plan to investigate the extension of global model clustering to other types of models, particularly HTTP response and session models. An additional line of future work is to investigate the application of different clustering methods in order to improve the efficiency of the querying procedure for high-cardinality knowledge bases.

Acknowledgments

The authors wish to thank the anonymous reviewers and our shepherd, Angelos Stavrou, for their insightful comments. This work has been supported by the National Science Foundation, under grants CCR-0238492, CCR-0524853, and CCR-0716095, by the European Union though the grant FP7-ICT-216026-WOMBAT, and by Secure Business Austria (SBA).

References

[1] J. Alpert and N. Hajaj. We knew the web was big... `http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html`, July 2008.

[2] S. Axelsson. The Base-Rate Fallacy and the Difficulty of Intrusion Detection. *ACM Transactions on Information and System Security*, 3(3):186–205, August 2000.

[3] S. Bhatkar, A. Chaturvedi, and R. Sekar. Dataflow Anomaly Detection. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P 2006)*, Oakland, CA, USA, May 2006. IEEE Computer Society.

[4] Breach Security, Inc. Breach WebDefend. `http://www.breach.com/products/webdefend.html`, January 2009.

[5] S. Cho and S. Cha. SAD: Web Session Anomaly Detection Based on Parameter Estimation. *Computers & Security*, 23(4):312–319, 2004.

[6] Citrix Systems, Inc. Citrix Application Firewall. `http://www.citrix.com/English/PS2/products/product.asp?contentID=25636`, January 2009.

[7] W. Cohen, P. Ravikumar, and S. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03)*, 2003.

[8] G. F. Cretu, A. Stavrou, M. E. Locasto, S. J. Stolfo, and A. D. Keromytis. Casting out Demons: Sanitizing Training Data for Anomaly Sensors. In *Proceedings*

of the IEEE Symposium on Security and Privacy (S&P 2008), pages 81–95, Oakland, CA, USA, May 2008. IEEE Computer Society.

[9] D. E. Denning. An Intrusion-Detection Model. *IEEE Transactions on Software Engineering*, 13(2):222–232, 1987.

[10] H. J. Escalante and O. Fuentes. Kernel Methods for Anomaly Detection and Noise Elimination. In *Proceedings of the International Conference on Computing (CORE 2006)*, pages 69–80, Mexico City, Mexico, 2006.

[11] F5 Networks, Inc. BIG-IP Application Security Manager. `http://www.f5.com/products/big-ip/product-modules/application-security-manager.html`, January 2009.

[12] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff. A Sense of Self for Unix Processes. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P 1996)*, pages 120–128, Oakland, CA, USA, May 1996. IEEE Computer Society.

[13] V. Frias-Martinez, S. J. Stolfo, and A. D. Keromytis. Behavior-Profile Clustering for False Alert Reduction in Anomaly Detection Sensors. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC 2008)*, Anaheim, CA, USA, December 2008.

[14] J. T. Giffin, D. Dagon, S. Jha, W. Lee, and B. P. Miller. Environment-Sensitive Intrusion Detection. In *Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 185–206. Springer-Verlag, 2005.

[15] L. Guangmin. Modeling Unknown Web Attacks in Network Anomaly Detection. In *Proceedings of the* 3^{rd} *International Conference on Convergence and Hybrid Information Technology (ICCIT 2008)*, pages 112–116, Washington, DC, USA, 2008. IEEE Computer Society.

[16] B. Juang and L. Rabiner. A probabilistic distance measure for hidden Markov models. *AT&T Bell Laboratories Technical Journal*, 64(2):391–408, 1985.

[17] S.-i. Kim and N. Nwanze. Noise-Resistant Payload Anomaly Detection for Network Intrusion Detection Systems. In *Proceedings of the Performance, Computing and Communications Conference (IPCCC 2008)*, pages 517–523, Austin, TX, USA, December 2008. IEEE Computer Society.

[18] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur. Bayesian Event Classification for Intrusion Detection. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC 2003)*, Las Vegas, NV, USA, December 2003.

[19] C. Kruegel, W. Robertson, and G. Vigna. A Multi-model Approach to the Detection of Web-based Attacks. *Journal of Computer Networks*, 48(5):717–738, July 2005.

[20] C. Kruegel, T. Toth, and E. Kirda. Service-Specific Anomaly Detection for Network Intrusion Detection. In *Proceedings of the Symposium on Applied Computing (SAC 2002)*, Spain, March 2002.

[21] W. Lee and S. J. Stolfo. A Framework for Constructing Features and Models for Intrusion Detection Systems. *ACM Transactions on Information and System Security*, 3(4):227–261, 2000.

[22] R. Little and D. Rubin. *Statistical analysis with missing data*. Wiley New York, 1987.

[23] R. B. Lyngsø, C. N. S. Pedersen, and H. Nielsen. Metrics and similarity measures for hidden markov models. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 178–186. AAAI Press, 1999.

[24] F. Maggi, M. Matteucci, and S. Zanero. Detecting intrusions through system call sequence and argument analysis (preprint). *IEEE Transactions on Dependable and Secure Computing*, 99(1), 2009.

[25] M. Mahoney and P. Chan. Detecting novel attacks by identifying anomalous network packet headers. Technical Report CS-2001-2, Florida Institute of Technology, 2001.

[26] M. V. Mahoney and P. K. Chan. Learning Rules for Anomaly Detection of Hostile Network Traffic. In *Proceedings of the IEEE International Conference on Data Mining*, 2003.

[27] Miniwatts Marketing Group. World Internet Usage Statistics. `http://www.internetworldstats.com/stats.htm`, January 2009.

[28] D. Mutz, F. Valeur, C. Kruegel, and G. Vigna. Anomalous System Call Detection. *ACM Transactions on Information and System Security*, 9(1):61–93, February 2006.

[29] Open Security Foundation. DLDOS: Data Loss Database – Open Source. `http://datalossdb.org/`, January 2009.

[30] W. Robertson, G. Vigna, C. Kruegel, and R. A. Kemmerer. Using Generalization and Characterization Techniques in the Anomaly-based Detection of Web Attacks. In *Proceedings of the Network and Distributed System Security Symposium (NDSS 2006)*, San Diego, CA, USA, February 2006.

[31] R. Sekar, M. Bendre, D. Dhurjati, and P. Bollineni. A Fast Automaton-Based Method for Detecting Anomalous Program Behaviors. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P 2001)*, pages 144–155, Oakland, CA, USA, May 2001. IEEE Computer Society.

[32] R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang, and S. Zhou. Specification-based Anomaly Detection: A New Approach for Detecting Network Intrusions. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS 2002)*, pages 265–274, New York, NY, USA, 2002. ACM Press.

[33] O. Shezaf, J. Grossman, and R. Auger. Web Hacking Incidents Database. `http://www.xiom.com/whid-about`, January 2009.

[34] A. Singer. Social Media, Web 2.0 And Internet Stats. `http://thefuturebuzz.com/2009/01/12/social-media-web-20-internet-numbers-stats/`, January 2009.

[35] Y. Song, A. D. Keromytis, and S. J. Stolfo. Spectrogram: A Mixture-of-Markov-Chains Model for Anomaly Detection in Web Traffic. In *Proceedings of the Network and Distributed System Security Symposium (NDSS 2009)*, San Diego, CA, USA, February 2009.

[36] A. Stolcke and S. Omohundro. Hidden Markov Model Induction by Bayesian Model Merging. *Advances in Neural Information Processing Systems*, pages 11–11, 1993.

[37] A. Stolcke and S. Omohundro. Inducing Probabilistic Grammars by Bayesian Model Merging. *Lecture Notes in Computer Science*, pages 106–106, 1994.

[38] G. Tandon and P. Chan. Learning Rules from System Call Arguments and Sequences for Anomaly Detection. In *ICDM Workshop on Data Mining for Computer Security (DMSEC)*, pages 20–29, 2003.

[39] G. Tandon, P. Chan, and D. Mitra. MORPHEUS: Motif Oriented Representations to Purge Hostile Events from Unlabeled Sequences. In *Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, pages 16–25, Washington DC, USA, 2004. ACM.

[40] D. Turner, M. Fossi, E. Johnson, T. Mark, J. Blackbird, S. Entwise, M. K. Low, D. McKinney, and C. Wueest. Symantec Global Internet Security Threat Report – Trends for 2008. Technical Report XIV, Symantec Corporation, April 2009.

[41] A. Valdes and K. Skinner. Adaptive, Model-based Monitoring for Cyber Attack Detection. In H. Debar, L. Me, and F. Wu, editors, *Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID 2000)*, number 1907 in Lecture Notes in Computer Science, pages 80–92, Toulouse, France, October 2000. Springer-Verlag.

[42] D. Wagner and D. Dean. Intrusion Detection via Static Analysis. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P 2001)*, pages 156–168, Oakland, CA, USA, 2001. IEEE Computer Society.

[43] K. Wang, J. J. Parekh, and S. J. Stolfo. Anagram: A Content Anomaly Detector Resistant to Mimicry Attack. In *Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID 2006)*, Hamburg, GR, September 2006. Springer-Verlag.

[44] K. Wang and S. J. Stolfo. Anomalous Payload-based Network Intrusion Detection. In *Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID 2004)*. Springer-Verlag, September 2004.

[45] R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[46] R. H. Zakon. Hobbes' Internet Timeline v8.2. `http://www.zakon.org/robert/internet/timeline/`, November 2006.

[47] S. Zanero and S. M. Savaresi. Unsupervised learning techniques for an intrusion detection system. In *Proceedings of the 2004 ACM Symposium on Applied Computing*, pages 412–419. ACM Press, 2004.