# Model Aggregation for Distributed Content Anomaly Detection

Sean Whalen
shwhalen@gmail.com

Nathaniel Boggs
boggs@cs.columbia.edu

Salvatore J. Stolfo
sal@cs.columbia.edu

Columbia University
New York NY 10027, USA

## ABSTRACT

Cloud computing offers a scalable, low-cost, and resilient platform for critical applications. Securing these applications against attacks targeting unknown vulnerabilities is an unsolved challenge. Network anomaly detection addresses such zero-day attacks by modeling attributes of attack-free application traffic and raising alerts when new traffic deviates from this model. Content anomaly detection (CAD) is a variant of this approach that models the payloads of such traffic instead of higher level attributes. Zero-day attacks then appear as outliers to properly trained CAD sensors. In the past, CAD was unsuited to cloud environments due to the relative overhead of content inspection and the dynamic routing of content paths to geographically diverse sites. We challenge this notion and introduce new methods for efficiently aggregating content models to enable scalable CAD in dynamically-pathed environments such as the cloud. These methods eliminate the need to exchange raw content, drastically reduce network and CPU overhead, and offer varying levels of content privacy. We perform a comparative analysis of our methods using Random Forest, Logistic Regression, and Bloom Filter-based classifiers for operation in the cloud or other distributed settings such as wireless sensor networks. We find that content model aggregation offers statistically significant improvements over non-aggregate models with minimal overhead, and that distributed and non-distributed CAD have statistically indistinguishable performance. Thus, these methods enable the practical deployment of accurate CAD sensors in a distributed attack detection infrastructure.

## Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and Protection*

## Keywords

Model Aggregation, Anomaly Detection, Machine Learning

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

*AISec'14,* November 7, 2014, Scottsdale, Arizona, USA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2953-1/14/11 ...$15.00.
http://dx.doi.org/10.1145/2666652.2666660.

## 1. INTRODUCTION

More applications each year are migrated to, or developed for, the cloud [1]. The economy of scale, on-demand computation, centralized analytics, and numerous other advantages have resulted in mandates for government cloud deployments [2] in addition to the momentum of the private sector. Growing with this demand is the need for robust zero-day attack detection as data centers become more lucrative targets. Current signature-based intrusion detection fails to identify such novel attacks, while traditional anomaly detection may be too slow to react due to its dependence on sufficient traffic statistics.

In recent years, content anomaly detection (CAD) has proven effective for this task while being resistant to mimicry and poisoning attacks [3,4]. While CAD has been shown effective for sites with static request paths, load balanced distributed environments can violate the traditional assumptions of anomaly detection by spreading requests (and thus content) over multiple geographically diverse sites due to a lack of session affinity or other factors. Even at a single geographic location, load balancers may not be suitable hosts for heavyweight CAD sensors due to impacts on performance, resilience, and security. For these reasons we wish to spread the overhead of CAD sensors across nodes in the system, but each distributed sensor has an incomplete view of an application's content in this environment. Since new content is much more likely due to a routing change than an actual attack, false positive rates are prohibitively high. We term this the *dynamic pathing problem* of distributed anomaly detection.

In this paper, we introduce methods for aggregating the local CAD models of Random Forest (supervised), Logistic Regression (supervised), and Bloom Filter (unsupervised) classifiers trained on $n$-gram payloads of normal application content. We then present an extensive comparative analysis of CAD using our aggregation methods with a publicly available IDS dataset containing labeled content-based HTTP attacks, as well as larger private labeled dataset collected at Columbia University. Summarizing our primary findings:

- The area under the ROC curve for aggregated Bloom Filter, Logistic Regression, and Random Forest content models have 95% confidence intervals of 97.3-97.7% and 98.9-99.9% for ISCX and Columbia datasets respectively, using an $n$-gram length of 5 and a 25-node distributed web application simulated via cross validation. This is a statistically significant performance improvement over non-aggregate models.

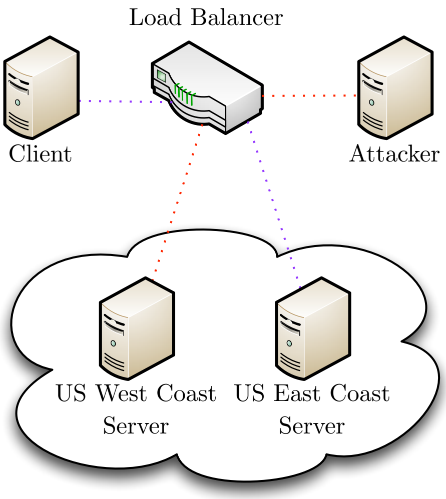

**Figure 1: A load balancer routes a normal request and an SQL injection attack to application servers on different coasts of the US. If the west coast CAD model detects the injection but the attacker migrates to the east coast server, the injection may be undetected there. The attack would be detected if the individual CAD models were first aggregated and re-distributed.**

- Aggregate, unsupervised Bloom Filters have *performance comparable to supervised models*, with the additional benefit of better generalization to zero-day attacks since an explicit representation of the attack class is not necessary.
- Aggregate content model performance is *statistically indistinguishable from non-distributed CAD* where all traffic is observed by the application server.
- Our aggregation methods *do not require the exchange of content* between nodes, dramatically reducing overhead and offering varying degrees of content privacy.

As a result, we find that unsupervised Bloom Filter models combined with our aggregation methods enable practical deployment of distributed content-based anomaly detectors in the cloud, wireless sensor networks, or other similar environments.

## 2. BACKGROUND

### 2.1 Load Balancing

One simple method of load balancing creates multiple IPs for a given DNS record. To illustrate, consider a subset of IPs returned from querying Google:

```
$ host -t a google.com
google.com has address 74.125.239.105
google.com has address 74.125.239.97
google.com has address 74.125.239.102
```

DNS clients receiving such a record will permute this address list in a client-determined way such that different connection attempts are likely to use a different IP. This permutation is often round robin and thus termed Round Robin DNS (RRDNS). If the servers at each IP provide identical services, this offers a basic method of load balancing and scales horizontally. The simplicity of this method is also its weakness, as clients will attempt to use servers that malfunction until the client's DNS entries expire and an updated record (lacking the offending server) is fetched.

Hardware load balancers (HLBs, also called network-based load balancers) improve on the weaknesses of DNS-based load balancing by adding server awareness and offering more advanced balancing strategies. An HLB might monitor server health via echo requests and route only to responsive servers based on (for example) average response time, fewest connections, or hard-coded weights representing hardware capacity. However, HLBs are not as scalable and as a result are often used in combination with RRDNS for extremely high volume sites.

### 2.2 Geo-Distribution

Servers may reside in geographically diverse locations in distributed environments, particularly in the cloud. To illustrate, Amazon provides servers in multiple geographic areas called *regions* that are further divided into isolated locations called *availability zones* (ex: us-east-1, us-west-1, and us-west-2). Applications are commonly deployed using HLBs across multiple regions in combination with DNS-based load balancing to improve not only fault tolerance but application latency. Companies such as Amazon, Akamai, Dynect, and UltraDNS all have such offerings utilizing IP-based geolocation for routing requests to servers "closest" to the client, often at the global scale (termed Global Server Load Balancing).

### 2.3 The Dynamic Pathing Problem

Load balancing offers increased fault tolerance and lower latency but may impact application usability. For example, a user authenticated to server $A$ in region us-west-1 might be routed to us-west-2 due to a hardware failure and thus have to re-login since server $B$ is unaware of the user's session and its associated cookie on server $A$. *Session stickyness* or *affinity* is an option provided by some HLBs to ensure all requests during a user's session are routed to the same application server. It does this by creating independent load balancer cookies that are optionally tied to the duration of the application session cookie. For the duration of the load balancer cookie, the same user will be routed to the same server.

These optimizations paint a complex picture of the path application data takes through the network. By design, each server in the network will see only a fraction of the overall application traffic. To secure these servers we may wish to deploy different types of anomaly detection sensors that allow local detection of attacks. Anomaly detection traditionally builds models over an attack-free dataset and flags outliers as potential attacks. Without seeing the complete picture of the network, however, a sensor deployed at a geographically diverse application server is likely to have an extremely high false positive rate since what it considers "normal" will vary, even from collaborating servers of the same application.

In the following example, Amazon's Route 53 DNS load balancer routes client requests to an application server in Amazon zones us-west-1 and us-east-1 (see Figure 1). An attacker sends an HTTP request containing an SQL injection to the application, and this request is routed to us-west-1 (path in red) by the load balancer based on their geo-located IP. Within a similar time frame, a non-attacker sends a normal request that is routed instead to us-east-1 (path in purple). For illustration, let only the model at us-west-1 be capable of detecting this attack. Without some method of model aggregation, if the attacker is ever routed to us-east-1 due to non-sticky sessions, hardware failures, or attacker migration, the attack will not be detected (resulting in a false negative). This is the *dynamic pathing problem* for anomaly detection: Normal models of partial traffic are rarely trained on enough data to reliably detect attacks or distinguish real attacks from normal traffic. We aim to solve this problem through the introduction of low-overhead content model aggregation schemes suitable for distributed environments.

# 3. DATASET AND METHODS

## 3.1 Public Dataset

Publicly available IDS datasets containing both packet payloads and ground truth “normal" or “attack" labels are extremely rare, but are essential for fostering reproducible research. Towards this end, we selected the Information Security Centre of Excellence (ISCX) dataset [5] to evaluate our methods in a reproducible manner. We briefly summarize the dataset here and refer to the Shiravi et al. paper [5] for details.

The dataset captures packets as network flows (see RFC 3917) such that content sent with the same protocol, source IP, and source port within some time period is grouped into a single field, along with its ground truth label and other TCP/IP fields. The captures contain 84 gigabytes of traffic generated over 7 days; 31,414 out of 1,827,231 flows contain attacks. Most of these attacks represent denial-of-service, so the data is filtered down to 105 content-based attacks with unique HTTP request payloads. These attacks are combined with an equal number of randomly sampled non-attack HTTP request payloads to create a balanced dataset. This is essential for unbiased evaluation of the supervised models since normal traffic is orders of magnitude more prominent and would cause the classifier to overfit to the majority (normal) class, though does not affect unsupervised Bloom Filters.

## 3.2 Private Dataset

We augment our reproducible evaluation with a larger, private dataset collected at Columbia University since the ISCX dataset contains a smaller number of content-based HTTP attacks. This dataset consists of 800,000 packets collected from Computer Science departmental webservers in 2014, containing 34,207 normalized HTTP requests. These requests are clustered and manually labeled as malicious or benign. Malicious requests include SQL injection, file upload, and other widespread attacks.

For this dataset, the $n$-gram feature matrix used by the supervised models (discussed next) is more densely populated due to the larger number of samples. We downsampled the dataset to $\approx$ 5,000 normal and 5,000 attack payloads for computational efficiency when $n = 5$. This downsampling results in a conservative performance estimate for unsupervised Bloom Filters that intrinsically handle, and benefit greatly from, a larger number of samples.

## 3.3 Feature Representation

$n$-Grams are a standard representation of strings as consecutive sequences of $n$ bytes. They are alternately called unigrams when individual bytes are used, digrams for two bytes, and so on. This representation converts strings into a feature vector suitable for machine learning algorithms. The vector can be binary, representing the presence or absence of each $n$-gram, or alternately a multinomial distribution over $n$-gram counts.

To illustrate using payloads from our dataset, consider the following example. Below is the HTTP URI of a normal request (displayed across multiple lines but not containing actual line breaks), representing the relevant application content of the packet:

```
/catalog/admin/file_manager.php/login.php
    ?action=save
```

and its 5 most frequent digrams:

```
       count
ngram
.p         2
hp         2
ph         2
in         2
og         2
```

Now consider the HTTP URI of a content-based attack:

```
/cgi-bin/;echo${IFS}-ne${IFS}"\\x6e\\x63\\x20\\x2d
    \\x6c\\x70\\x20\\x
```

and similarly the top 5 digrams:

```
       count
ngram
\x         8
0\         3
x6         3
x2         3
IF         2
```

Given sufficient data and an appropriate $n$-gram length, these distributions can distinguish normal requests from attacks. Previous work found empirically that $n = 5$ is suitable for CAD [3]; we demonstrate the performance of additional values in Section 4.

## 3.4 Bloom Filters

Bloom Filters are extremely compact, probabilistic set representations with a tunable false positive rate [7]. Items are inserted using multiple hash functions to set bits in a vector and filters can be combined or intersected using fast and efficient logical bitwise operations. They have previously been used as micromodels for CAD [8] where collaborating sites exchange models of *abnormal* instead of normal content to efficiently share attack patterns. This is not suitable for dynamically-pathed environments where most traffic may look like attacks to non-aggregate models, and so our aggregation operates on models of normal content.

To operate as a classifier, a filter is first trained by inserting all $n$-grams for each application payload in the training

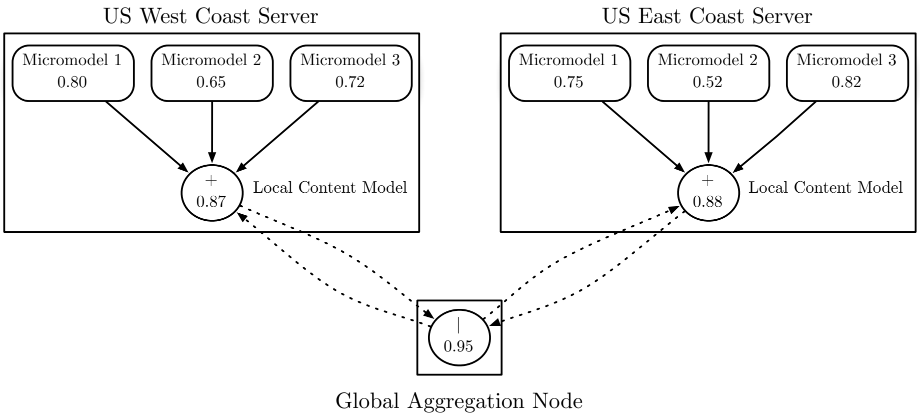

**Figure 2: Geographically diverse sites construct multiple micromodels across time with limited attack detection accuracy. These micromodels are first locally aggregated, resulting in improved detection. Local aggregate models are then sent to a central node, aggregated into a single model with a global representation of normal content and high detection accuracy, and finally re-distributed to application nodes. Given a suitable model, these operations have low network and CPU overhead and can be de-centralized using other algorithms such as peer-to-peer or gossip protocols [6].**

set. The filter then contains a compact representation of normal content $n$-grams. Predictions are made by calculating the percent of unobserved $n$-grams for each application payload in the test set. This produces a vector of probabilities for the test set, representing the model's confidence that each payload is an attack. Choosing a suitable value of $n$ is essential for this model as hash collisions are likely occur for unigrams and other small values of $n$.

Due to the simplicity of the model, aggregation is simple: Take the logical OR of each model sent to a designated aggregation node. This process may optionally incorporate a threshold by counting the number of times a bit is set at each position in the model and setting a bit in the final model only if this count is above a certain threshold. Effectively, this allows each local model to cast a vote for each bit position in the final filter. Note that the aggregating node needs only the space-efficient bit vector of each Bloom Filter rather than the original content to perform aggregation. The model also provides basic privacy as the content stored in the filter cannot be extracted—one could test for the presence of a particular $n$-gram, but the bits could have been set by some other combination of $n$-grams since the model achieves its compactness by allowing for false positives.

## 3.5 Logistic Regression

Logistic regression is the adaptation of linear regression to classification problems with a discrete instead of real-valued dependent variable. For binary classification, the model assumes the probability that the dependent variable $Y_i$ belongs to the positive class is a weighted combination of $m$ predictor variables $X_i = x_1 \ldots x_m$. The weights $\beta = \beta_0 \ldots \beta_m$, or *regression coefficients*, can be learned by various optimization procedures to maximize the likelihood of the training data. The $\beta_0$ term serves as the intercept. The probability of observation $i$ is then computed as:

$$\mathbb{E}[Y_i|X_i] = \text{logistic}(\beta \cdot X_i) = \frac{1}{1 + e^{-\beta \cdot X_i}}$$

where $\beta \cdot X_i$ is the dot product between the regression coefficients and the predictor variables, and the logistic function converts the outcome to a probability in the range 0..1. The term $x_0 = 1$ is included with the predictors to match up with the intercept $\beta_0$.

If each node uses the same feature space, aggregation is again simple: the vector of regression coefficients for each node are averaged together and re-distributed. This works surprisingly well and also provides a basic level of content privacy since the mapping between regression coefficients and $n$-grams is not directly revealed, though strong guarantees on privacy require more sophisticated approaches (see Chaudhuri et al. [9] and related work in Section 5). Coefficient vectors are relatively space-efficient, particularly if sparsity is enforced via L1-norm or LASSO regularization, though typically not as compact as Bloom Filters (see Section 4).

## 3.6 Random Forests

Decision trees map predictor variables to a target variable outcome by using a tree-based representation that recursively splits on individual feature values at each node (e.g. "Go to left child node if $x_2 > 3.3$"), starting at the root until a leaf containing a prediction is reached. They are extremely popular across many disciplines for their visual interpretability but are notorious for their tendency to overfit training data. Learning decision trees from data is a major topic in data mining and machine learning; we refer the reader to introductory texts such as Mitchell [10] for further detail.

Random forests overcome the performance limitations of decision trees by creating an ensemble (collection) of trees where each tree is trained on a random subset of predictors with a resampled (with replacement) version of the training data [11]. The predictions over all trees are averaged together to create a final prediction which is more accurate than any individual tree, and indeed random forests offer state-of-the-art performance for a wide variety of classification and regression problems.

A simple method for aggregating multiple ensembles of trees takes the union of the trees contained in each ensemble. For example, aggregating 2 forests with 100 trees each results in a single forest of 200 trees. This is neither space nor time efficient and is included here for comparative analysis to the space-efficient models above in order to determine if extra predictive performance can be gained by using a more complex, non-linear model. More elaborate methods might prune the aggregate forest using some critera to maintain a fixed ensemble size and thus reduce network and CPU overhead.

## 3.7 Cross-Validation

We estimate the generalization performance of Logistic Regression, Random Forest, and Bloom Filter-based classifiers using a standard $K$-fold cross-validation scheme where $K = 20$. This divides the dataset into 20 folds, where 95% of the samples in each fold are used for training and the rest for testing the model's performance on data that has not been observed during training. The test set of each fold is non-overlapping so that eventually all samples are used independently for both testing and training. The performance over all folds is averaged to produce a final estimate of generalized predictive performance that is less biased than using a single training/test split.

We further divide the training set from each fold into $L$ subsets to simulate distributed environments with $L$ nodes as outlined in Section 2. Each subset is used to train a separate model, termed a *micromodel* [8], and represents the dynamically-pathed network traffic observed by distributed nodes such as geographically diverse application servers in the cloud or motes in a wireless sensor network.

## 3.8 Aggregation

To detect attacks observed at other nodes, micromodels must be aggregated and re-distributed so that false positives and false negatives can be reduced to levels that make CAD deployment practical. Each node may train a single micromodel or multiple micromodels using content collected at different time intervals. Consider the simplified example of US east and west coast regional application servers behind a geo-locating load balancer. Each server trains three Bloom Filter micromodels at different time intervals using requests routed via the load balancer. The accuracy of each micromodel is annotated in Figure 2. These three micromodels are aggregated locally into a single micromodel using the voting method described earlier, resulting in improved local accuracy:

US West Coast Server

| Model | Bits | | |
|---|---|---|---|
| Micromodel $A$ | 0 | 1 | 0 |
| Micromodel $B$ | 1 | 1 | 0 |
| Micromodel $C$ | 0 | 0 | 1 |
| Apply Sum | 1 | 2 | 1 |
| Apply Threshold | 0 | 1 | 0 |

US East Coast Server

| Model | Bits | | |
|---|---|---|---|
| Micromodel $A$ | 1 | 1 | 0 |
| Micromodel $B$ | 1 | 0 | 0 |
| Micromodel $C$ | 1 | 1 | 0 |
| Apply Sum | 3 | 2 | 0 |
| Apply Threshold | 1 | 1 | 0 |

Next, the bit vector of the locally aggregated east and west coast micromodels are transmitted to a global aggregation node that performs a logical OR operation and re-distributes the aggregate bit vector to each node:

Global Aggregation Node

| Model | Bits | | |
|---|---|---|---|
| Local Content Model $A$ | 0 | 1 | 0 |
| Local Content Model $B$ | 1 | 1 | 0 |
| Apply Logical OR | 1 | 1 | 0 |

Both east and west coast servers now have models that perform substantially better than their local versions, using extremely low network and CPU overhead. Other methods for model dissemination are also possible, such as peer-to-peer or gossip protocols [6], to reduce dependency on a single-point-of-failure aggregator.

The two-phase aggregation scheme outlined above is useful when each model votes during the first phase; if a simple OR is used without voting, the local aggregate is the same as having a single local micromodel. Since this concept cannot be implemented in the same way for Logistic Regression and Random Forests, we restrict the comparative analysis in Section 4 to one-phase aggregation by a single node and present our analysis of two-phase aggregation elsewhere.

## 3.9 Metrics: AUC

Presented with a test set, a model produces a vector of probabilities corresponding to its confidence that each test sample belongs to the positive (attack) class. To convert this probability into a label, a threshold would be set for the sensor and samples with a probability above this threshold will be flagged as an attack; otherwise it passes as normal. Setting this threshold involves a trade-off between acceptable false positive and false negative rates where normal content is flagged as an attack or an attack passes as normal, respectively. To evaluate performance in a threshold-independent manner using the known labels for each test set, we use the standard area under the Receiver Operating Characteristic curve (AUC) metric. This sums the area under the curve formed by all possible thresholds for the true positive rate *tpr* and false positive rate *fpr*:

$$\mathrm{tpr} = \frac{\mathrm{tp}}{\mathrm{tp} + \mathrm{fn}}$$

$$\mathrm{fpr} = \frac{\mathrm{fp}}{\mathrm{fp} + \mathrm{tn}}$$

where $tp$, $tn$, $fp$, and $fn$ are counts of true positives, true negatives, false positives, and false negatives. AUC is commonly approximated by the trapezoidal rule for integrating the area under the curve formed by the set of $(tpr, fpr)$ pairs resulting from thresholding.

## 3.10 Metrics: $F_{\max}$

No single metric can emphasize all possible errors equally; due to its use of the false positive rate, AUC emphasizes true negatives more heavily (see the denominator of $fpr$ in the above definition). Alternatives such as the $F$-measure combine different metrics, precision and recall, into a single number:

$$\text{precision} = \frac{\text{tp}}{\text{tp} + \text{fp}}$$

$$\text{recall} = \text{tpr}$$

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

For a more comprehensive evaluation, we include the $F_{\max}$ score (with $\beta = 1$) as it places emphasis on false positives over false negatives. Similar to AUC, this score computes the $F$-measure for every possible threshold, affecting the values of precision and recall, and takes the maximum of the $F$-measures over these thresholds. This metric was recently introduced as a more robust, interpretable alternative to measures such as area under the Precision-Recall curve (auPR).

## 3.11 Significance Testing

Hypothesis testing is used to evaluate if the result of an experiment occurred by random chance. A null model representing the distribution of statistics for random experiments is rejected if the test statistic computed for the experiment of interest falls within the extremes of the null distribution. The p-value is the probability of observing a value at least as extreme as the test statistic, assuming the null hypothesis is true. The null hypothesis is rejected if the p-value falls below some significance level $\alpha$. The significance level represents a calculated risk of falsely rejecting the null hypothesis when it is in fact true, and so is often set to values of 0.05 or lower to reduce the chance of false positives. In turn, lower $\alpha$ values increase the risk of false negatives (accepting the null hypothesis when it is false). Here we use the standard $\alpha = 0.05$ significance level.

The Wilcoxon signed-rank test is a non-parametric statistical hypothesis test for determining if there is a significant difference between the mean ranks of samples under two different conditions. In this paper, the conditions are the performance of local and global content models. The term non-parametric indicates there are no underlying assumptions made about the distribution of samples; this distinguishes the Wilcoxon test from the better-known Student's $t$-test that performs the same task but for normally distributed values.

The Friedman test is a non-parametric alternative to analysis of variance (ANOVA) for measuring differences in conditions using the same subjects. Here, the conditions are the model types and the subjects are CAD systems of different size. A single node system represents a non-distributed environment. In contrast to the better-known ANOVA test, the Friedman test uses rank transformations analyze non-normally distributed data and is considered the most appropriate method for comparing performance of machine learning classifiers [12].

# 4. RESULTS

We focus the presentation of our results on the public ISCX dataset due to its reproducibility, as well as space limitations. However, the figures and discussion below also apply to the Columbia dataset, with the primary difference being tighter error bounds due to a larger number of samples and a smaller difference between aggregate and local supervised content models.

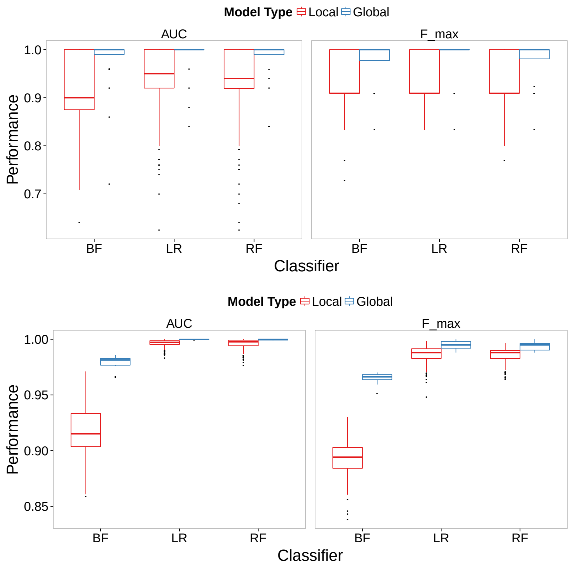

**Figure 3: Performance of local and global content models on ISCX (top) and Columbia (bottom) datasets as measured by 20-fold cross validation with AUC and $F_{\max}$ metrics for Logistic Regression (LR), Random Forest (RF), and Bloom Filter (BF) classifiers. HTTP request payloads are represented as 5-grams and subdivided into 25 equally-sized training sets to simulate cloud application servers behind a load balancer.**

To begin, Figure 3 (top) presents the performance of CAD for a 25 node system using 5-grams. The AUC and $F_{\max}$ of Logistic Regression (LR), Bloom Filter (BF), and Random Forest classifiers are shown for both the local and global (aggregate) models. Observe that for both metrics the median performance of local models is markedly lower, resulting in impractical false positive rates, while the median performance of the global models is substantially higher.

Next, Figure 4 is a facet diagram presenting the same data as Figure 3 (top) where $n$-gram length varies across columns and the number of nodes varies across rows. To simplify this complex figure, $F_{\max}$ numbers are omitted but the trends seen in Figure 3 still hold. A broader picture of performance emerges from the facets, such as the poor performance of BFs in combination with unigrams (left column) where global performance actually decreases due to collisions, while LR and RF are more robust. As one would expect, local and global performance are identical in a non-distributed setting (top row). Also observe that as the number of nodes increase, the performance gap between local and global models increases since each local model is trained on less data. The curse of dimensionality's effect can also be seen where the jump from trigrams to 5-grams reduces local performance for BFs with 10 and 25 nodes, suggesting BFs benefit from more densely populated bit vectors. Overall, median global performance remains similar for all three classifiers for trigrams and 5-grams using 10 or 25 nodes, but with slightly more variance than the non-distributed setting.

Figure 5 shows the size of local and global models under varying numbers of nodes and $n$-gram lengths. The y-axis is the square root of the actual model size (in KB) to prevent

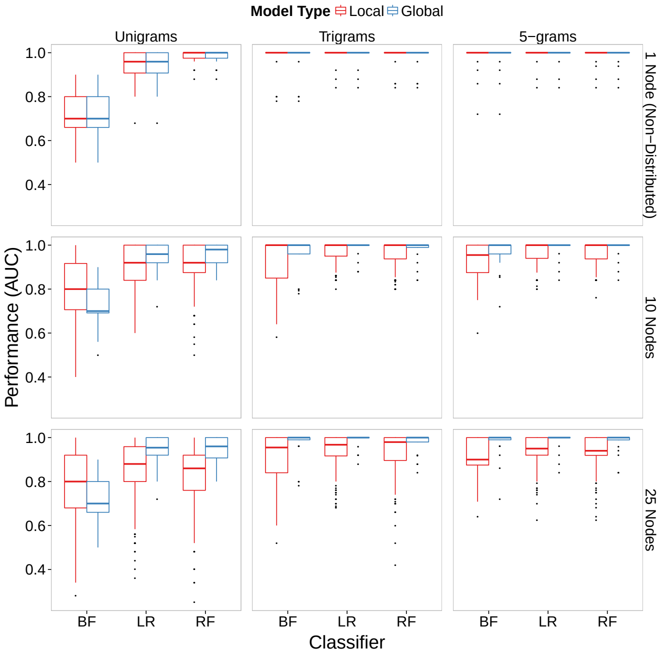

**Figure 4: Classifier performance as a function of $n$-gram length, node count, and model type. See Section 4 for detailed discussion.**

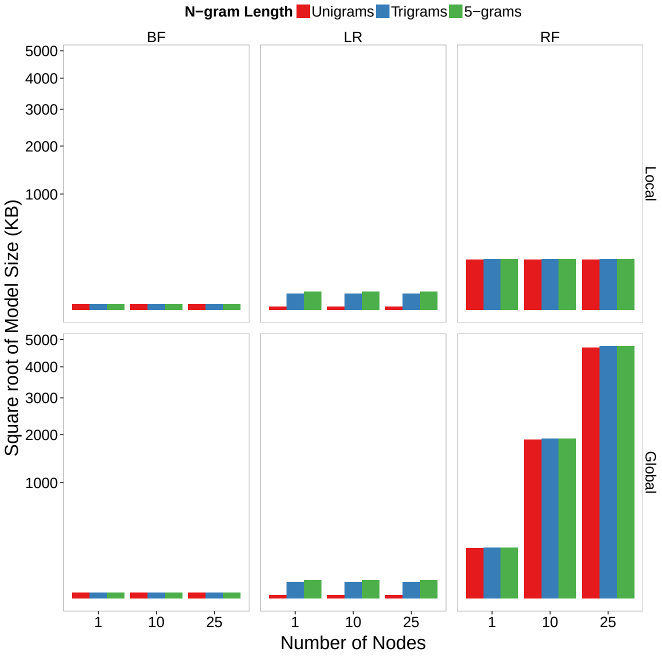

**Figure 5: Square root of content model size (in KB) as a function of classifier, node count, and model type. See Section 4 for detailed discussion.**

the large size of the RF from concealing other patterns. First observe BFs are approximately the same size over all $n$-gram lengths, though this size would need to increase to support even larger $n$-gram lengths while maintaining an acceptable false positive rate. LR coefficient vectors grow only slightly with $n$-gram length and taper off at $n = 5$ due to the smaller size of our dataset, while remaining independent of the number of nodes. The standout pattern is RF, dominating both BFs and LR for local and global models. This is unfair to RF as our simple aggregation method could be further refined to prune redundant or non-predictive trees to retain a fixed ensemble size independent of node count. However, given its large overhead for local models and without a significant performance gain over simpler classifiers, we feel this result (as well as its supervised nature) rules out RF as a candidate for practical CAD implementations. LR has comparable size and slightly improved performance over BFs, but this minor advantage is outweighed by its supervised nature as well.

While performance is similar across classifier types, we wish to evaluate if global aggregation offers statistically significant gains over local models for each combination of $n$-gram length and node count, excluding the non-distributed setting where performance is equivalent. This is performed with the Wilcoxon signed-rank test, a non-parametric hypothesis test for comparing the mean of non-normally distributed samples between two groups (see Section 3). The p-value of the test for each model, $n$-gram length, and node count combination is given in Table 1, rounded to three digits. There is indeed a statistically significant difference between local and global model performance at the $\alpha = 0.05$ level for every set of parameters.

We next use the Friedman test for repeated measures (see Section 3) to determine if aggregated CAD models offer similar performance to content models in a non-distributed setting under either performance metric. The p-values for AUC and $F_{\max}$ are 0.71 and 0.09, respectively, meaning we cannot reject the null hypothesis that performance is the same at the standard $\alpha = 0.05$ cutoff for statistical significance. Therefore, though one might expect some degradation due to aggregation, *model aggregation effectively closes the performance gap between a 10 or 25 node CAD system and a single non-distributed CAD node.* Observe also that the lower (but still insignificant) p-value under $F_{\max}$ picks up on performance differences better than AUC, as we would expect from the larger error bars of $F_{\max}$ in Figure 3.

The above analysis focuses on the public ISCX dataset. Similar performance is seen with the Columbia dataset where aggregate models achieve 98.9-99.9% AUC and 97.9-99.4% $F_{\max}$. Figure 3 (bottom) is representative of the primary differences between the two datasets. First, error bars are in much tighter as there is significantly more data for the models to train on, and performance estimates vary less between folds as a result. Second, both local and global Bloom Filter performance are lower in the Columbia dataset. This is due to the subsampling necessary for comparison to the supervised methods; there is roughly 7 times more data that the Bloom Filters could easily be trained on, but the resulting $n$-gram feature matrix for the supervised methods is prohibitively large. Due to the increased diversity of $n$-grams in this dataset, subsampling puts BFs at an unfortunate disadvantage. Finally, the gap between local and global performance for supervised content models is much smaller. Given this much data, supervised models construct a highly accurate representation of attacks in the training data. However, it is unlikely their representation will generalize to zero-day attacks. Bloom Filters have a distinct advantage in this regard (see related work), and the performance gap between the two approaches is small: 97.8% AUC versus 99.9% between BFs and LR/RF. Thus, aggregation greatly closes the gap between supervised and unsupervised content models in this dataset as well.

## 5. RELATED WORK

These methods offer basic privacy protection during aggregation, but this is not the primary focus of our work and more formal approaches exist. Chaudhuri et al. [9] present bounded, privacy-preserving versions of Logistic Regression and Support Vector Machines under the $\epsilon$-differential privacy model. In the distributed setting, privacy-preserving modeling of federated datasets is an active area of research in the biomedical community where patient data stored at different institutions cannot be exchanged due to federal regulations. Hall et al. [13] use homomorphic encryption to apply linear and ridge regression in such settings without sharing intermediate values. Wu et al. [14] propose an alternate version of the Logistic Regression aggregation scheme given here with the purpose of learning separate models at each institution and combining coefficients into a single model to improve pattern discovery across patients without needing direct access to the underlying data. Their work further reinforces the effectiveness of our aggregation method for linear models.

In the field of anomaly detection, “PAYLoad anomaly detection” (PAYL) is one of the earliest unsupervised learning algorithms applied to application content [15]. It first estimates 1-gram payload byte distributions conditioned on flow direction, port number, and payload length from training data. New data is labeled anomalous if the distance of its byte distributions to those of the training data is beyond some threshold. The “Payload Content based Network Anomaly Detection” algorithm (PCNAD) is a modification of PAYL to accommodate high speed links by training with subsets of the payload [16]. Both algorithms are fast and effective for detecting some abnormal content such as machine code in the payload of email traffic, but cannot consider broader context such as byte orderings and correlations between bytes. As a result, they are susceptible to simple mimicry attacks [4] and have high false positive rates—problems addressed by the higher order $n$-grams employed by the Bloom Filter-based Anagram [3].

Cretu et al. introduced a new approach to training and sanitizing anomaly detection models called “Sanitizing Training Data for Anomaly Sensors” (STAND) [8]. Their method applies the notion of ensemble learning to train multiple normal models and combine their predictions, an extension of earlier work by Stolfo et al. [17] in financial fraud detection. Instead of constructing a single model from the complete training set, the data is split into disjoint sets and used to train individual *micromodels.* Each packet is tested against these micromodels and is labeled normal or abnormal based on a weighted voting scheme. If some threshold of the micromodels has seen the packet then it is used to train a sanitized model free of attacks and outliers. Otherwise, the packet is added to a separate anomalous model. These anomalous models can be exchanged between collaborating sites to reduce poisoning attacks via a process called

| Classifier | N-gram Length | Node Count | P-value |
|---|---|---|---|
| Bloom Filter | 1 | 10 | 0.019 |
| Bloom Filter | 1 | 25 | 0.041 |
| Bloom Filter | 3 | 10 | 0.037 |
| Bloom Filter | 3 | 25 | 0.001 |
| Bloom Filter | 5 | 10 | 0.008 |
| Bloom Filter | 5 | 25 | 0.000 |
| Logistic Regression | 1 | 10 | 0.009 |
| Logistic Regression | 1 | 25 | 0.000 |
| Logistic Regression | 3 | 10 | 0.019 |
| Logistic Regression | 3 | 25 | 0.001 |
| Logistic Regression | 5 | 10 | 0.002 |
| Logistic Regression | 5 | 25 | 0.001 |
| Random Forest | 1 | 10 | 0.003 |
| Random Forest | 1 | 25 | 0.000 |
| Random Forest | 3 | 10 | 0.017 |
| Random Forest | 3 | 25 | 0.001 |
| Random Forest | 5 | 10 | 0.002 |
| Random Forest | 5 | 25 | 0.002 |

**Table 1: P-values for a Wilcoxon signed-rank test of performance differences between global and local models over each combination of classifier, $n$-gram length, and node count. The results show statistically significant performance improvements for global models over all parameter combinations at the $\alpha = 0.05$ level.**

*cross-sanitization.* Also employing ensemble learning techniques, Multiple-Classifier Payload-based Anomaly Detector (McPAD) [18] combines multiple one-class Support Vector Machines with diverse feature spaces to detect polymorphic shellcode contained within packet payloads.

Spectrogram employs a mixture of Markov Chains for content anomaly detection of web requests [19], directly modeling the distribution of overlapping request $n$-grams. This approach is typically exponential ($256^n$) in the $n$-gram size. However, this is reduced to linear complexity ($M*(256^2*(n-1)+M$ for $M$ chains) using an approximate factorization. In addition to directly modeling $n$-gram distributions, Spectrogram differs from Anagram in that it focuses on web request strings instead of the entire payload packet – an approach later combined with Anagram by Autosense [20] (reviewed below). In the distributed setting, aggregate Spectrogram models are not well-defined. In addition, Spectogram does not permit incremental training and has high computational overhead.

Each of these content-based sensors were originally designed for single sites. However, several systems aggregate or correlate alerts from distributed sites. DShield is a centralized repository of shared alert information. Another system called Worminator creates distributed watch lists using Bloom that are exchanged at a centralized location or by an overlay network called Whirlpool [21,22]. This enables sites to detect anomalous packets if a novel attack is widespread, but in isolation these sites cannot be sure if the resulting anomalies are false positives.

Though normal traffic is often site-specific, abnormal packets common across sites are reliable indicators of widespread attacks. Autosense [20] uses STAND with Anagram to detect such widespread zero-day attacks across collaborating sites with extremely low false positive rates.

Hadžiosmanović et al. [23] analyze the effectiveness of $n$-gram models including PAYL, Anagram, and McPAD with binary protocols such as SMB and Modbus. They find that while $n$-gram models can achieve high detection rates with binary protocols, this often comes at the cost of prohibitively high false positive rates. They attribute this weakness to the high variability of binary protocol payloads and suggest protocol-specific $n$-gram analysis to target relevant payload regions. We take such an approach by examining normalized rather than complete HTTP requests, and anticipate we would encounter similar difficulties with arbitrary binary protocols.

Other work has more formally examined methods for combining alerts of multiple intrusion detectors without the dynamic pathing problem. Gu et al. [24] propose a decision-theoretic technique for combining alerts in an IDS ensemble that may offer better performance than some of the simple aggregation methods here, though we chose such methods for their low CPU and network overhead.

# 6. CONCLUSION

Content anomaly detection has recently proven effective for zero-day attack detection, especially for remotely collaboration sites, while being resistant to mimicry and poisoning attacks [3]. However, CAD in its current state is less than ideal for clouds or other distributed systems where each node observes only a fraction of global traffic due to load balancing, or environmental constraints such as found in wireless sensor networks. Content models from each node must be combined into a single global model and re-distributed in order to accurately detect application-level attacks present in packet payloads.

In this paper we demonstrate how various content models can accurately detect such content-based attacks against distributed application servers by introducing new model aggregation techniques. Our methods eliminate the need for content exchange during aggregation, increasing network and CPU efficiency while providing varying levels of content privacy To foster reproducibility, our analysis uses a publicly available IDS dataset containing both application content and ground truth labels [5], as well as a larger private dataset collected from Columbia University departmental webservers.

Using these datasets we evaluate the performance of both supervised and unsupervised content models including Logistic Regression, Random Forests, and Bloom Filters. We show that unsupervised CAD models achieve 97.8% AUC, approaching the 99.9% AUC of supervised methods while generalizing to zero-day attacks without needing labeled data. Using the ISCX dataset we find global aggregation offers statistically significant performance improvements over local models, while aggregation primarily benefits unsupervised Bloom Filters using the larger Columbia dataset. In addition, due to the small size and comparable performance of Bloom Filters, we find the potential performance improvements offered by Random Forest models are further outweighed by their network and CPU overhead. Bloom Filters trail behind Logistic Regression slightly in terms of accuracy but offer modest savings in network overhead, though this gap may be further closed by sparse representations of coefficient vectors in combination with L1-norm or LASSO regularization.

Most importantly, using our aggregation methods there is no statistically significant performance difference (AUC or $F_{\max}$) between non-distributed and distributed CAD environments, nor a significant difference between supervised and unsupervised aggregate models. Thus, *aggregated unsupervised CAD models are as effective as either supervised or unsupervised models in both distributed and non-distributed settings*, while maintaining the advantages of unsupervised models for deployment in real-world networks.

Motivated by, but not limited to, the increasing risk posed by high-value assets in the cloud, these results demonstrate that CAD sensors offer efficient and accurate attack detection in distributed environments and may complement other lightweight sensors to form a distributed attack detection infrastructure resilient against a broad class of zero-day application attacks.

## 7. ACKNOWLEDGMENTS

This work was supported by DARPA through Contract FA8650-11-C-7190. Any opinions, findings, conclusions or recommendations expressed herein are those of the authors, and do not necessarily reflect those of the U.S. Government or DARPA.

## 8. REFERENCES

[1] H. Kisker, P. Matzke, S. Ried, and M. Lisserman, "Forrsights: The Software Market In Transformation, 2011 And Beyond," tech. rep., Forrester Research, 2011.

[2] DARPA, "I2O Mission-Oriented Resilient Clouds," tech. rep., 2011.

[3] K. Wang, J. J. Parekh, and S. J. Stolfo, "Anagram: A Content Anomaly Detector Resistant to Mimicry Attack," in *Proceedings of the 9th Symposium on Recent Advances in Intrusion Detection*, pp. 226–248, 2006.

[4] P. Fogla and W. Lee, "Evading Network Anomaly Detection Systems: Formal Reasoning and Practical Techniques," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pp. 59–68, 2006.

[5] A. Shiravi, H. Shiravi, M. Tavallee, and A. A. Ghorbani, "Towards Developing a Systematic Approach To Generate Benchmark Datasets for Intrusion Detection," *Computers & Security*, vol. 31, no. 3, pp. 357–374, 2012.

[6] M.-J. Lin and K. Marzullo, "Directional Gossip: Gossip in a Wide Area Network," in *Proceedings of the 3rd European Dependable Computing Conference*, pp. 364–379, 1999.

[7] B. H. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[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 2008 IEEE Symposium on Security and Privacy*, pp. 81–95, 2008.

[9] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially Private Empirical Risk Minimization," *Journal of Machine Learning Research*, vol. 12, pp. 1069–1109, 2011.

[10] T. M. Mitchell, *Machine Learning*. McGraw-Hill, 1997.

[11] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[12] J. Demšar, "Statistical Comparisons of Classifiers over Multiple Data Sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[13] R. Hall, S. E. Fienberg, and Y. Nardi, "Secure Multiple Linear Regression Based on Homomorphic Encryption," *Journal of Official Statistics*, vol. 27, no. 4, pp. 669–691, 2011.

[14] Y. Wu, X. Jiang, J. Kim, and L. Ohno-Machado, "Grid Binary LOgistic REgression (GLORE): Building Shared Models Without Sharing Data," *Journal of the American Medical Informatics Association*, vol. 19, no. 5, pp. 758–764, 2012.

[15] K. Wang and S. J. Stolfo, "Anomalous Payload-Based Network Intrusion Detection," in *Proceedings of the 7th Symposium on Recent Advances in Intrusion Detection*, pp. 203–222, 2004.

[16] S. A. Thorat, A. K. Khandelwal, B. Bruhadeshwar, and K. Kishore, "Payload Content Based Network Anomaly Detection," in *Proceedings of the 1st International Conference on Applications of Digital Information and Web Technologies*, pp. 127–132, 2008.

[17] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan, "Cost-based Modeling for Fraud and Intrusion Detection: Results from the JAM Project," in *Proceedings of the DARPA Information Survivability Conference and Exposition*, vol. 2, pp. 130–144, 2000.

[18] R. Perdisci, D. Ariu, P. Fogla, G. Giacinto, and W. Lee, "McPAD : A Multiple Classifier System for Accurate Payload-based Anomaly Detection," *Computer Networks*, vol. 53, no. 6, pp. 864–881, 2009.

[19] 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 16th Annual Network and Distributed System Security Symposium*, 2009.

[20] N. Boggs, S. Hiremagalore, A. Stavrou, and S. J. Stolfo, “Cross-Domain Collaborative Anomaly Detection: So Far Yet So Close,” in *Proceedings of the 14th Symposium on Recent Advances in Intrusion Detection*, pp. 142–160, 2011.

[21] M. E. Locasto, J. J. Parekh, S. J. Stolfo, A. D. Keromytis, T. Malkin, and V. Misra, “Collaborative Distributed Intrusion Detection,” tech. rep., Columbia University, 2004.

[22] M. E. Locasto, J. J. Parekh, A. D. Keromytis, and S. J. Stolfo, “Towards Collaborative Security and P2P Intrusion Detection,” in *Proceedings of the 6th Annual Information Assurance Workshop*, pp. 333–339, 2005.

[23] D. Hadžiosmanović, L. Simionato, D. Bolzoni, E. Zambon, and S. Etalle, “N-Gram against the Machine: On the Feasibility of the N-Gram Network Analysis for Binary Protocols,” in *Proceedings of the 15th International Conference on Research in Attacks, Intrusions, and Defenses*, pp. 354–373, 2012.

[24] G. Gu, A. A. Cárdenas, and W. Lee, “Principled Reasoning and Practical Applications of Alert Fusion in Intrusion Detection Systems,” in *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security*, pp. 136–147, 2008.