Cross-domain Collaborative Anomaly Detection: So Far Yet So Close [⋆]

Nathaniel Boggs[1], Sharath Hiremagalore[2], Angelos Stavrou[2], and Salvatore J. Stolfo[1]

[1] Department of Computer Science, Columbia University
`{boggs,sal}@cs.columbia.edu`
[2] Department of Computer Science, George Mason University
`{shiremag,astavrou}@gmu.edu`

Abstract. Web applications have emerged as the primary means of access to vital and sensitive services such as online payment systems and databases storing personally identifiable information. Unfortunately, the need for ubiquitous and often anonymous access exposes web servers to adversaries. Indeed, network-borne zero-day attacks pose a critical and widespread threat to web servers that cannot be mitigated by the use of signature-based intrusion detection systems. To detect previously unseen attacks, we correlate web requests containing user submitted content across multiple web servers that is deemed abnormal by local Content Anomaly Detection (CAD) sensors. The cross-site information exchange happens in real-time leveraging privacy preserving data structures. We filter out high entropy and rarely seen legitimate requests reducing the amount of data and time an operator has to spend sifting through alerts. Our results come from a fully working prototype using eleven weeks of real-world data from production web servers. During that period, we identify at least three application-specific attacks not belonging to an existing class of web attacks as well as a wide-range of traditional classes of attacks including SQL injection, directory traversal, and code inclusion without using human specified knowledge or input.

Keywords: Intrusion Detection, Web Security, Anomaly Detection, Attacks, Defenses, Collaborative Security

1 Introduction

Web applications are the primary means of access to the majority of popular Internet services including commerce, search, and information retrieval. Indeed,

[⋆] This work is sponsored in part by US National Science Foundation (NSF) grant CNS-TC 0915291 and AFOSR MURI grant 107151AA "MURI: Autonomic Recovery of Enterprise-wide Systems After Attack or Failure with Forward Correction." The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government.

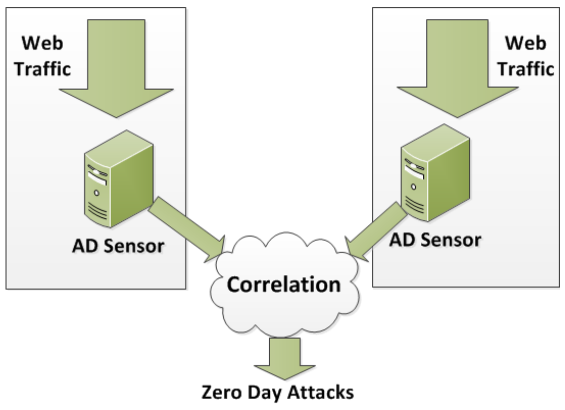


Fig. 1. Architecture

online web portals have become a crucial part of our everyday activities with usage ranging from bank transactions and access to web email to social networking, entertainment, and news. However, this reliance on ubiquitous and, in most cases, anonymous access has turned web services into prime targets for attacks of different levels of sophistication. Newly crafted attacks, often termed "zero-day," pose a hard to address challenge compromising thousands of web servers before signature-based defenses are able to recognize them [31]. Although recent research indicates that Anomaly Detection (AD) sensors can detect a class of zero-day attacks, currently, AD systems experience limitations which prevent them from becoming a practical intrusion detection tool.

In this paper, we propose a new defense framework where Content Anomaly Detection (CAD) sensors, rather than traditional IDS systems, share content alerts with the aim of detecting wide-spread, zero-day attacks. Contrary to pure alert correlation and fusion [29], we exchange abnormal content across sites as a means to reduce the inherent high false positive rate of local CAD systems. We leverage local CAD sensors to generate an accurate, reliable alert stream where false positives are consumed through a process of alert validation; false positives rarely make their way in front of a human operator. We implement information exchange mechanisms enabling the collaborative detection of attacks across administrative domains. We believe such collaboration, if done in a controlled and privacy preserving manner, will significantly elevate costs for attackers at a low cost for defenders. Our system has a number of core capabilities: high-quality, verified alert streams that focus on detecting the presence of and learn from zero-day attacks and previously unseen attack instances; scalable alert processing; and modular multi-stage correlation. Figure 1 illustrates the overall architecture.

Intuitively, inbound web requests fall into three categories: legitimate low entropy requests, legitimate high entropy or rarely seen requests, and malicious requests. Legitimate low entropy requests are the most accurately modeled by CAD systems. Therefore, each individual CAD sensor will label previously seen, low entropy requests as normal and will not exchange them with other CAD sensors. Legitimate high entropy or rare requests will often show up as abnormal to the local CAD sensor and will therefore be exchanged. Since remote sites do not have similar content due to the high entropy nature or rarity of these requests, no matches will be identified, and thus no alerts will be raised. On the other hand, malicious requests will appear as abnormal in **many** local CAD models. Therefore, when exchanged, they will match other sites and alerts will be raised. The more sites participating the better the coverage and the faster the response to wide-spread web attacks. Space and structural constraints due to HTTP protocol and specific web application parsing limit the ability for an attacker to fully exploit polymorphism techniques, analyzed in [22], so each zero-day attack should exhibit similar content across the attacked web services.

In our experimental evaluation, we use eleven weeks of traffic captured from real-world, production web servers located in different physical and network locations. We do not inject any artificial or additional data. All attacks and statistics described are observed on live networks. We measured the detection and false positive changes from adding an additional server in the sharing system. Most interestingly, we confirm the theory presented by [4] that false positives tend to repeat across sites. Additionally, as most of the false positives occur early and often, we show that CAD systems can benefit greatly from a reasonable cross-site training period. This reduces the number of the false positives to 0.03% of all the normalized web requests. Furthermore, we quantify the similarity of the produced CAD models from each site over long periods of time. Using these models we provide an analysis of how aggregate normal and abnormal data flows compare between sites and change over time. Moreover, we furnish results regarding the threshold of the matching content and the effects of increasing the set of participating collaborating sites. Finally, we are the first to present a real-world study of the average number of alerts a human operator has to process per day. Moreover, we show that the alert sharing and correlation of alerts reduces the human workload by at least an order of magnitude.

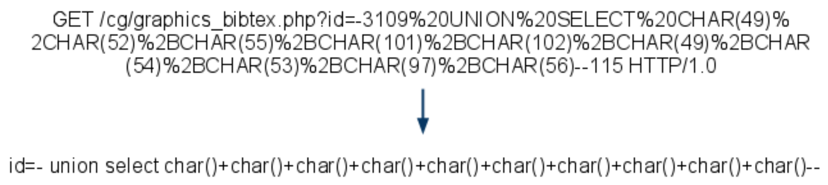
2 Related Work

Anomaly Detection techniques have been employed in the past with promising results. Alexsander Lazarevic *et al.* compares several AD systems in Network Intrusion Detection [12]. For our analysis, we use the STAND [5] method and Anagram [30] CAD sensor as our base CAD system. The STAND process shows improved results for CAD sensors by introducing a sanitization phase to scrub training data. Automated sensor parameter tuning has been shown to work well with STAND in [6]. Furthermore, the authors in [24] observe that replacing outdated CAD models with newer models helps improve the performance of the sensor as the newer models accurately represent the changes in network usage over time. Similarly, in [30] the authors proposed a local shadow server where the AD was used as a fiter to perform dynamic execution of suspicious data. In all of the above works, due to limited resources within a single domain, a global picture of the network attack is never examined. Furthermore, Intrusion Detection Systems that leverage machine learning techniques suffer from well-known limitations [20]. In the past, there has been a lot of criticism for Anomaly Detection techniques [25] especially focusing on the high volume of the false positives they generate. With our work we dispel some of this criticism and we show that we can improve the performance of CAD systems by sharing content information across sites and correlating the content alerts.

Initially, Distributed Intrusion Detection Systems (DIDS) dealt with data aggregated across several systems and analyzed them at a central location within a single organization. EMERALD [23] and GrIDS [18] are examples of these early scalable DIDS. Recent DIDS systems dealt with collaborative intrusion detection systems across organizations. Krügel *et al.* developed a scalable peer-to-peer DIDS, Quicksand [9, 10] and showed that no more messages than twice the number of events are generated to detect an attack in progress. DShield [27] is a collaborative alert log correlation system. Volunteers provide DShield with their logs where they are centrally correlated and an early warning system provides "top 10"-style reports and blacklists to the public gratis. Our work differs in that we rely on the actual user submitted content of the web request rather than on source IP. More general mechanisms for node "cooperation" during attacks are described in [2, 1].

DOMINO [33], a closely related DIDS, is an overlay network that distributes alert information based on hash of the source IP address. DShield logs are used to measure the information gain. DOMINO differs from our technique as it does not use AD to generate alerts. DaCID [7] is another collaborative intrusion detection system based on the Dempster Shafer theory of evidence of fusing data. Another DIDS with a decentralized analyzer is described by authors in [34].

Centralized and decentralized alert correlation techniques have been studied in the past. The authors in [26] introduce a hierarchical alert correlation architecture. In addition to scalability in a DIDS, privacy preservation of data send across organizations is a concern. Privacy preservation techniques that do not affect the correlation results have been studied. A privacy preserving alert correlation technique, also based on the hierarchical architecture [32] scrubs the alert strings based on entropy. We expand Worminator [15] a privacy preserving alert exchange mechanism based on Bloom filters, which had previously been used for IP alerts. Furthermore, Carrie Gates et al. [8] used a distributed sensor system to detect network scans albeit showing limited success. Finally, there has been extensive work in signature-based intrusion detection schemes [19] [16]. These systems make use of packet payload identification techniques that are based on string and regular expression matching for NIDS [28] [11] [14]. This type of matching is only useful against attacks for which some pattern is already known.



```
GET /cg/graphics_bibtex.php?id=-3109%20UNION%20SELECT%20CHAR(49)%
2CHAR(52)%2BCHAR(55)%2BCHAR(101)%2BCHAR(102)%2BCHAR(49)%2BCHAR
(54)%2BCHAR(53)%2BCHAR(97)%2BCHAR(56)--115 HTTP/1.0
```

```
id=- union select char()+char()+char()+char()+char()+char()+char()+char()+char()+char()--
```

Fig. 2. A normalization example from a confirmed attack. The first line of the original GET request is shown. We use the output of the normalization function for all future operations.

3 System Evaluation

3.1 Data Sets

We collected contiguous eight weeks of traffic between October and November 2010 of all incoming HTTP requests to two popular university web servers: *www.cs.columbia.edu* and *www.gmu.edu*. To measure the effects of scaling to multiple sites, we added a third collaborating server, *www.cs.gmu.edu*. This resulted in an additional three weeks in December 2010 of data from all three servers. The second data set allows us to analyze the effects of an additional web site to the overall detection rate and network load. To that end, we are able to show the change in the amount of alert parsing a human operator would have to deal with in a real-world setting and analyze models of web server request content. All attacks detected are actual attacks coming from the internet to our web servers and are confirmed independently using either IDS signatures[17, 16] developed weeks after the actual attacks occurred and manual inspection when such signatures were not available. However, that does not preclude false negatives that could have been missed by both signature-based IDS and our approach. The number of processed packets across all of our datasets are over 180 million incoming HTTP packets. Only 4 million of them are deemed as suspicious because our normalization process drops simple web requests with no user submitted variables.

3.2 Normalized Content

Our system inspects normalized content rather than packet header attributes such as frequency or source IP address. We process all HTTP GET requests and we extract all user-defined content (*i.e. user specified parameters*) from the URI across all request packets. Putting aside serious HTTP protocol or server flaws, the user specified argument string appears to be primary source of web attacks. We use these user-specified argument strings to derive requests that are deemed abnormal and can be used for correlating data across servers serving different pages. Additionally, we normalize these strings in order to more accurately compare them [4, 21]. We also decode any hex-encoded characters to identify potential encoding and polymorphic attacks. Any numeric characters are inspected and but not retained in the normality model to prevent overtraining from legitimate but high entropy requests. Also, we convert all the letters to lowercase to allow accurate comparisons and drop content less than five characters long to avoid modeling issues. Figure 2 illustrates this process.

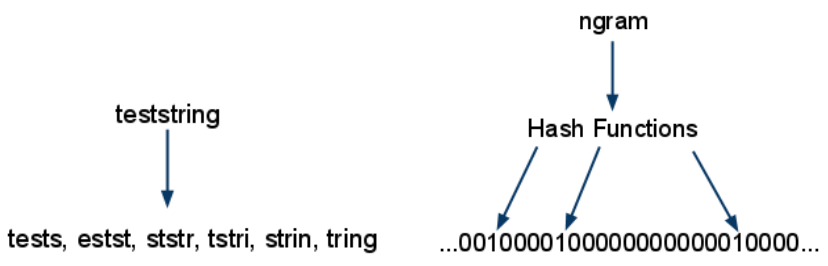
Moreover, we perform tests analyzing POST request data as well. POST requests are approximately 0.34% of the total requests. However, our experiments show that the current CAD sensor does not accurately train with data that exhibits large entropy typical in most POST requests. We leave the development of a new CAD sensor that can accurately model POST requests for future work and we focus on analyzing GET requests, which dominate the web traffic we observe (99.7%).

3.3 Content Anomaly Detector and Models

In cross-site content correlation, each site builds a local model of its incoming requests using a Content Anomaly Detection (CAD) sensor. In our experiments, we leverage the STAND [5] optimizations of the Anagram [30] CAD sensor although any CAD sensor with a high detection rate could be used with our approach. However, we apply the CAD sensors on normalized input instead of full packet content as they originally operated on in order to obtain more accurate results. Moreover, we fully utilize all of the automatic calibration described in [6] including the abnormal model exchange to exclude repeated attacks from poisoning the training data. The Anagram normal models are, as described in [30], Bloom filters [3] containing the n-gram representation of packets voted as normal by the STAND micro-models. A Bloom filter is a one-way data structure where an item is added by taking multiple hashes and setting those indices of a bit array to one. This provides space efficiency and incredible speed suitable for high speed networks since adding an element or checking if one is already present are constant time operations. Each normalized content is spilt into 5-gram sections as in [5] using a sliding window of five characters. See Figure 3(a) for an example. Requests can then be easily tested as to how many of the n-grams from their argument string are present in the model. N-grams give us a granular view of content allowing partial matches as opposed to hashing full content while maintaining enough structure of the content to be much more accurate than character frequency models. Previous work [15] calibrated Bloom filters to have an almost non-existent false positive rate and shows that extracting the content is infeasible, which allows for the preservation of privacy. The models we use are 2^{28} bits long and compress to about 10-80KB, a size that is easily exchanged as needed. The Anagram models test weather new content is similar to previous content by comparing how many of the n-grams exist in the model already.

3.4 Alert Exchange

We leverage the initial work of Worminator [15], an alert exchange system that we heavily extend to meet the needs of our system. A content exchange client



(a) A string broken into n-grams. (b) N-grams added to the Bloom Filter.

Fig. 3. n-grams and Bloom Filter

instance runs at each site and receives content alerts and abnormal models. We use a common format so that any CAD sensor can easily be adapted to work with the exchange system. In our case, each site's local STAND/Anagram sensor sends the content alert packet to the Worminator client as soon as it tests a packet and finds it abnormal. The Worminator client then encodes the content alerts as Bloom filters if at a remote site and then sends the content alerts and any abnormal models through a secure channel over the internet to the Worminator server. The bandwidth usage with this alert exchange turns out to be minimal since we only look at GET requests with argument strings and then further narrow down content by only exchanging the abnormal content alerts. It turns out that each alert encoded in a Bloom filter takes around 2KB to transmit on average. For our eight week experiment this translates into an average of 0.9Kb/sec bandwidth needed per site for a real time system, leaving plenty of room to scale up to a large set of collaborators before running into bandwidth constraints. A back-end process on the server performs the correlation of content alerts by comparing the local unencoded alerts to the Bloom filter representation of alerts from remote sites. We perform all our experiments faster than real time while exchanging encoded content alerts securely over the internet.

By exchanging content alerts from remote sites only in their Bloom filter form our system can protect the privacy of legitimate web requests. During the Bloom filter correlation process only the fact that a match occurs can be determined not any specific content. If a match occurs then this is a piece of content that a site has already seen coming to their server, so the only new information revealed is that the other site also had this content incoming. In this way we can gain information about the content we have in common, which most likely represents attacks while keeping the remaining content private in case there is sensitive information in the web requests.

3.5 Scaling to Multiple Sites

Our initial system deployment consists of three web servers. However, to be even more effective at quickly detecting widespread attacks, we envision a larger

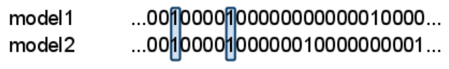

model1 ...00100001000000000000010000...
model2 ...00100001000000100000000001...

Fig. 4. Each model is stored in a Bloom filter, we count the number of bits set in common and then divide by the total number of bits set.

scale system deployment consisting of many collaborating sensors monitoring servers positioned in different locations on the Internet. For the system to scale up to include more sites, the correlation and alert comparison process has to scale accordingly. If we consider the pair-wise comparison of local alerts with each remote alert, it appears to grow asymptotically: O(n^2). This could turn can quickly become a problem; however, we can bound this total computation under a constant K by varying the amount of time duplicate alerts are stored in the system. In practice, we did not observe problems during our experiments even keeping eight weeks of data for correlation because indexing of alerts can be done using existing computationally efficient algorithm. Moreover, we only have to operate on unique alerts which are much smaller in size. Additionally, if a longer time frame is desirable, we can employ compression to the remote site alerts into a small number of Bloom filters by trading-off some accuracy and turn the scaling into order O(n) allowing many more alerts to be stored before running into any scaling issues. In that case, each time a new site joins the collaboration our local site must compare its alerts to the Bloom filters of those from the new site. Therefore, the overall computational complexity scales linearly with the number of remote sites participating. Since we can bound the local comparison with a remote site under K, the total computational cost scales linearly as well, and each site has optional tradeoffs in time alerts are kept and Bloom filter aggregation if local resources are limited. In practice, based on our numbers even with an unoptimized prototype we could scale to around 100 similar servers operating in real time and comparing all alerts over a few weeks' time. If additional utility is derived from having even more participating servers, then optimizing the code, time alerts are kept, and trading off accuracy in Bloom filter aggregation should easily allow additional magnitudes of scaling.

4 Model Comparison

Each normal model is a Bloom filter with all the n-grams of all normalized requests. By comparing Bloom filters as bit-arrays, we are able to estimate how much content models share. We test how many set bits each pair of models have in common and divide by the total number of set bits to get a percentage of set bits in common. The generated Bloom filters are quite sparse; therefore, the overlap of bits between content should be small as observed in Figure 4. We used this model comparison metric to compute the server distinctness and change in normal flows over time, whether servers in the same domain share additional commonality, and how much abnormal data we see in common across servers.

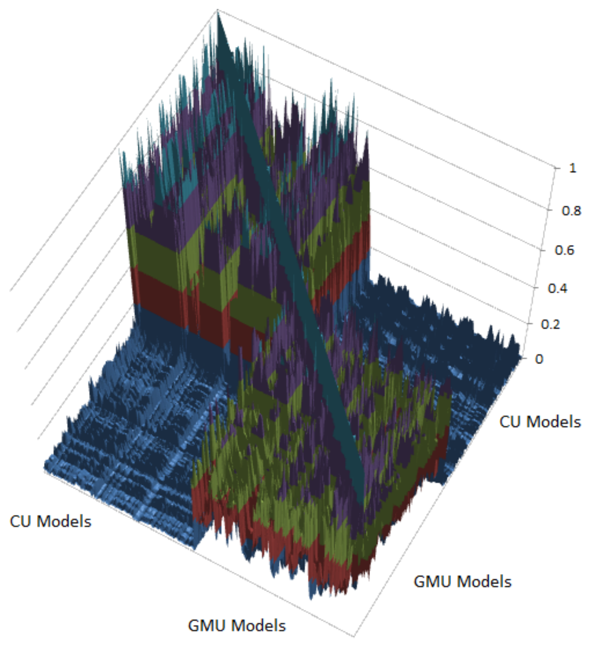


Fig. 5. Model comparison: the higher the figure the higher percentage of set bits the models had in common. The top and bottom quadrant are intra-site comparisons. The sides represent the comparison across-sites which, as expected, appear to have differences.

We first use this comparison to observe the differences in models from distinct sites with each other. We took every fifth model from our runs and compared the ones from the same runs to their counter parts at the other location. For normal models in our eight week run, we see on average 3.00% of bits set in common. We compare this to the over 40% of bits in common on average comparing models at the same site (Table 1). There is some overlap indicating that not filtering out normal content before performing the content alert correlation could lead to increased false positives. While we do not have enough sites to calculate how important this distinctness is to the accuracy achieved via correlation of alerts, we do confirm that at least for the distinct web servers our correlation process achieves effective results. See Figure 5 for a plot of the model comparison results. Models across long periods seem to keep a core of commonality but differ more than models close together in time. A product of this gradual change appears even with only five weeks difference in our datasets. Averaged over eight weeks both sites keep over 40% of bits in common while in the three week run this is closer to 50%. This reinforces existing work [5] showing that traffic patterns do evolve over time indicating that updating normal models periodically should increase effectiveness.

Comparison	**Normal Models Oct.-Nov.**	**Abnormal Models Oct.-Nov.**
Columbia CS	41.45%	52.69%
GMU Main	41.82%	38.51%
Cross site	3.00%	10.14%

Table 1. Commonality of normal and abnormal models.

Comparison - Normal Models	**Columbia CS**	**GMU Main**	**GMU CS**
Columbia CS	44.89%	3.89%	4.08%
GMU Main		48.39%	2.41%
GMU CS			56.80%
Comparison - Abnormal Models			
Columbia CS	53.05%	9.46%	9.32%
GMU Main		48.39%	8.55%
GMU CS			70.77%

Table 2. Comparison of abnormal and normal models between three sites. (Percentages of set bits in common shown.)

With our three week data set, we also have an additional web server from one administrative domain. With two web servers from the same Autonomous System we compare them to each other to see if our work has the potential to help a large organization that may have many separate web servers. See Table 2 for empirical details. Interestingly, we find no more commonality among normal models in the same domain than across domains. The fact that abnormal models at the these web servers share about as much in common with the server from another domain as each other suggests that attackers likely do not specifically target similar IP ranges with the same attacks. This suggests that web server administration and location may not play a factor in the effectiveness of using a particular web server for collaboration. An organization with sufficiently distinct web servers might be able to achieve good results without having to overcome the obstacles related to exchanging data between organizations.

The abnormal models from different sites show some similarity with close to 10% set bits matching, while models from the same site show more similarity. The high amount of common abnormal data between models at the same site may be influenced by legitimate requests classified as abnormal. More interesting is the commonality across sites. These shared bits most likely represent the attack data that we have found in common. There is an irregularity where some of the abnormal models are empty and could not be compared. We remove these empty models before computing the averages to avoid divide by zero issues. Multiple runs with the data confirm the strange behavior which can be contributed to a

cx=:cjygsheid&cof=forid:&ie=utf-&q=machine+learning+seminar&sa=go
o=-&id=&s=uhkf&k=hsbihtpzbrxvgi&c=kg

Table 3. Normalized examples of legitimate abnormal data seen at only one site.

faq=' and char()+user+char()= and "='
id=' and char()+user+char()= and "='

Table 4. Normalized example of abnormal data one from each site that match.

convergence of the STAND micromodels voted to include all the data from that time period into the normal models leaving the abnormal models empty.

Overall, our model comparisons provide quite interesting results. We find that each site has normal traffic flows that are distinct although changing somewhat over long periods of time. We see no major distinctions in comparison of same domain servers versus servers on separate domains, which indicates that our system could be deployed by a sufficiently large organization to protect itself without having to rely on outside collaborators. Finally, our measurements of abnormal data validate the idea that separate servers will receive similar attacks.

5 Correlation Results

The correlation process compares each unique content alert from the local sensors against the Bloom filter representation of each unique content alert from other sites. If at least 80% of the n-grams match the Bloom filter and the length of content before encoding, which is also exchanged, is within 80% of the raw content then we note it as a match. These matches are what the system identifies as attacks. Once these attacks are identified, the Bloom filter representation could be sent out to any additional participating servers and future occurrences could be blocked. In order to confirm our correlation results with the Bloom filters, we also perform an offline correlation of results using string edit distance [13] with a threshold of two changes per ten characters. We cluster together any pair of alerts from either site with less than this threshold. If a cluster contains alerts from more than one site, then it represents a common content alert. With only minor differences, these methods give us similar performance confirming that using privacy preserving Bloom filters provides an accurate and computationally efficient correlation. To simulate a production deployment, we use the Bloom filter comparison as our default correlation technique and use the string edit distance clustering only to facilitate manual inspection as needed, especially at a single site. See Table 3 for examples of true negatives where legitimate requests are not alerted on since each is seen at just one site. Table 4 shows an example of the same attack with slight variation being matched between two sites.

We run our experiments correlating the abnormal traffic between sites from our October-November eight week dataset and our December three week dataset

	Oct-Nov	**Oct-Nov with training**	**Dec.**	**Dec. with training**	**Gained by adding third server**	**Dec. Common to Three Sites**
Duration of testing[3]	54 days	47 days	19 days	12 days		
Total false positives	46653	362	40364	1031	1006	0
Unique false positives	64	13	48	5	3	0
Total true positives	19478	7599	7404	2805	186	322
Unique true positives	351	263	186	89	9	8

Table 5. Main experiment results considering a match to be at least 80% of n-grams being in a Bloom filter. Note that the 5th column results are included in column 4. Also, note that due to self training time for the CAD sensor actual time spent testing data is about two days less.

and then manually classify the results since ground truth is not known. We depict the system's alerts in Table 5. As we predicted in [4], most of the false positives repeat themselves early and often so we also show the results assuming a naïve one week training period which labels everything seen in that week and then ignores it. While this training technique could certainly be improved upon, we choose to show this example in order to better show the effectiveness of the approach as a whole and to preclude any optimizations that might turn out to be dataset specific. Such a training period provides a key service in that most false positives are either due to a client adding additional parameters regardless of web server, such as with certain browser add-ons, or servers both hosting the same application with low enough traffic throughput that it fails to be included in a normal model. Many of these cases tend to be rare enough to not be modeled but repeat often enough that a training period will identify them and prevent an operator from having to deal with large volumes of false positives. Certainly with such a naïve automated approach, attacks will not be detected during this training period, but after this period we end up with a large benefit in terms of few false positives with little negative beyond the single week of vulnerability. Any attacks seen during training that are then ignored in the future would have already compromised the system so we do not give an attacker an advantage going forward. In fact this training period serves an operator well in that many of the high volume attacks that are left over "background radiation" will be seen in this training period and thus not have to be categorized in the future. Adding an additional web server in our last experiment provides a glimpse at how broadening the scope of collaboration to a larger network of web servers can help us realize a high detection rate.

Let us now analyze how accurate our system is. The false positive rate is relatively easy to compute. We manually classify the unique alerts and then count the total occurrences of each. With regard to the number of requests that pass through the normalization process the false positive rate is 0.03%. If you calculate it based on the total incoming requests then it is much less. The true positive rate or detection rate is much harder to accurately measure since we have no ground truth. Recall, we are trying to detect widespread attacks and leave the

mosconfig_absolute_path=http://phamsight.com/docs/images/head??
config[ppa_root_path]=http://phamsight.com/docs/images/head??
option=com_gcalendar&controller=../../../../../../../../../../../../../../proc/self/environ%
id=' and user=–
id=-.+union+select+–
command=createfolder&type=image&currentfolder=/fck.asp&newfoldername=test&uuid=
option=com_user&view=reset&layout=confirm

Table 6. Normalized examples of actual attacks seen at multiple sites.

goal of detecting attacks targeted at a single site to other security methods in order to better leverage collaboration. With this in mind, there exists two places where a widespread attack could be missed. An attack could arrive at multiple sites but not be detected as abnormal by the one of the local CAD sensors and therefore, never be exchanged with other sites. The other possibility is that an attack could be abnormal at both sites but different enough that the correlation method fails to properly match it.

In the first case where a local CAD sensor fails to identify the attack as abnormal, we have a better chance to estimate our accuracy. Most CAD sensors are vulnerable to mimicry attacks where an attacker makes the attack seem like normal data by padding the malicious data in such a way as to fool the sensor. We can mitigate this by deploying very different sensors to each site, which while individually vulnerable to a specific padding method as a whole are very difficult to bypass. In this way an attacker might bypass some sites, but as the attack is widespread eventually two of the CAD sensors that the attacker is not prepared for can detect the attack and broadcast a signature out to the rest of the sites.

In the latter scenario, we have to rely heavily on the vulnerable web applications having some structure to what input they accept so that attacks exploiting the same vulnerability will be forced to appear similar. We can certainly loosen correlation thresholds as seen in Table 8 as well as come up with more correlation methods in the future. In practice, this is where the lack of ground truth hinders a comprehensive review of our performance. As far as we can tell, between the structure imposed by having to exploit a vulnerability with HTTP parameters, lower correlation thresholds, and finding additional attributes for correlation we should have a good head start on attackers in this arms race. At the very least, our layer of security will make it a race instead of just forfeiting to attackers immediately once a vulnerability is found. Without ground truth, we cannot be sure that we detect all widespread attacks. We have seen no indication in our data that attackers are using any of the above evasion techniques yet, so we believe that our system will provide an effective barrier, one which we can continue to strengthen using the above approaches.

[3] Due to equipment outages approximately three hours of data is missing from the Oct.-Nov. www.cs.columbia.edu dataset and less than 0.5% of the Dec. dataset abnormal data totals are missing.

ul=&act=&build=&strmver=&capreq=
c=&load=hoverintentcommonjquery-color&ver=ddabcfcccfadf
jax=dashboard_secondary
feed=comments-rss

Table 7. Normalized examples of false positives seen at multiple sites.

We detect a broad range of widespread attacks, with some examples shown in Table 6. Common classes of attacks show up such as code inclusion, directory traversal, and SQL injection. Our system faithfully detects any wide spread variants of these attacks, some of which might evade certain signature systems; however, the novel attack detection our system provides lies with the last two examples shown. These two attacks are attempting to exploit application specific vulnerabilities, one attacking an in-browser text editor and the other a forum system. Since attacks such as these resemble the format of legitimate requests and lack any distinct attribute that must be present to be effective, existing defenses cannot defend against zero-day attacks of this class. The fact that our system caught these in the wild bodes well for its performance when encountering new widespread zero-day attacks.

An examination of the false positives explains the repeated nature and sporadic occurrences of new false positives. See Table 7 for some examples of normalized false positives. All the false positives fall into one of two broad categories: rare browser specific requests or rarely used web applications installed on two or more collaborating servers. For example the most common false positive we see is an Internet Explorer browser plug-in for Microsoft Office which sends a GET request to the web server regardless of user intent. The use of this plug-in is rare enough that the request shows up as abnormal at all sites. As for server side applications, we see most of the unique false positives relating to the administrative functions of isolated Word Press blogs which see so little use that the requests stand out as abnormal. New false positives will continue to occur in small numbers as web servers and browsers evolve over time (less than one per three days on average during our eight week run). We believe that identifying these few rare occurrences is quite manageable for operators. This task gets easier since as the number of collaborators grow so do the resources for the minimal manual inspection needed to identify these isolated occurrences.

Adding a third web server, www.cs.gmu.edu, to the collaboration shows that additional web servers help us to identify more attacks and allows some basic insight into what types of web servers might be best grouped together for collaboration. Assuming our training method, adding this third server as a collaborating server exchanging data with www.cs.columbia.edu allows us to detect 11.25% more unique attacks than just correlating alerts between www.cs.columbia.edu and www.gmu.edu. This increase over the 80 unique attacks we detect without it, supports the need for adding substantial numbers of collaborators to increase the detection rate. Unfortunately this new collaborating server also introduces false positives that we do not see in previous experiments. We expect as with

	Oct-Nov	Oct-Nov with training	Dec.	Dec. with training	Gained by adding third server	Dec. Common to Three Sites
Total false positives	47605	439	41845	1017	4	0
Unique false positives	77	23	55	5	1	0
Total true positives	25042	10168	9594	3272	254	293
Unique true positives	488	362	221	109	10	8

Table 8. Experimental results considering a match to be at least 60% of n-grams to be in a Bloom filter.

previous false positives that future experiments will most likely repeat these with few new additions. An offline correlation using edit distance shows both GMU web servers having a number of attacks in common as well. This supports an idea that collaborating with distinct web servers could be as useful as collaborating across sites. False positives seem to be a function of rarely used web services located at each server, so servers hosting only a few clearly defined and well used services may give substantially better results.

This additional web server also provides the opportunity to require alerts to be seen by at least three sites before reporting them as attacks. While this proposition is hard to accurately evaluate with only one data set and just three servers, of which www.cs.gmu.edu experiences much lower traffic volumes, a couple interesting results stand out. As expected, both false positives and true positives drop off significantly. We see no false positives after the training period. This shows that for at least our data sets all of the server-side services that cause false positives drop out once we require three web servers to have the data in common. If this continues to be the case as more servers are added, then only reporting attacks that target three or more servers could solve most of the false positive issues. While requiring three servers to confirm an attack does yield less true positives, the ones it does detect are quite widespread and if the collaboration is expanded, the detection should increase greatly. This method, while scaling in detection rate more slowly than only requiring two servers to confirm attacks, could be a much more effective option to keep false positives low once enough servers collaborate.

We calculate the implications of changing the threshold for matching two alerts. Increasing the threshold past 80% to require perfect or almost perfect matches fails to help in reducing the false positives, since at this threshold almost all of the false positives are exact matches so even requiring all n-grams to match a Bloom filter exactly does not help. Reducing the threshold to allow more loose matches does show a trade off in increased detection of attacks at the expense of additional false positives. By only requiring 60% of n-grams from one alert to match the Bloom filter representation of another site's alert, we can expect to capture attacks with significantly more variance such as similar payloads targeting different web applications. See experiment details in Table 8. While at first, the results from a lower threshold appear quite good in terms of raw numbers of alerts, looking at only the new unique alerts which human operators

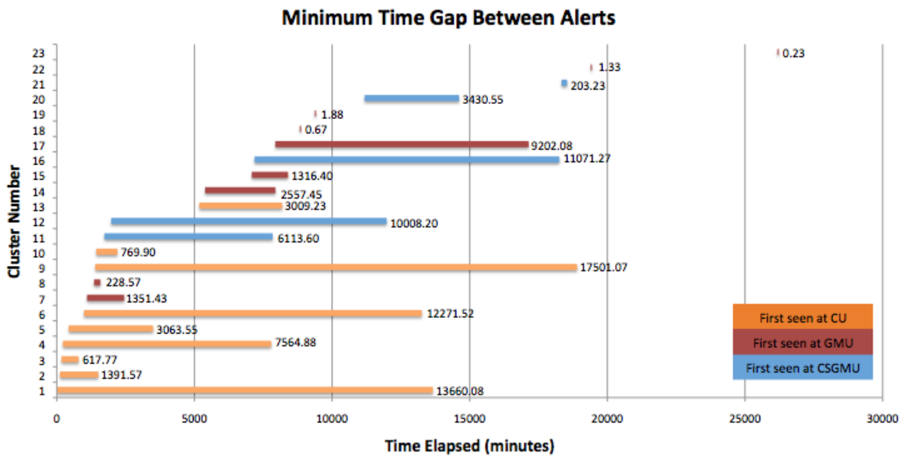


Fig. 6. Time gap between alerts at collaborating sites.

Time Gap in Minutes	Across Site CU, GMU and GMU CS	Across Site CU and GMU	Across Site CU and GMU CS	Across Site GMU and GMU CS
Min	0.23	1.48	7.52	0.23
Max	17501.07	25911.00	20589.02	24262.13
Average	4579.85	5477.35	7048.07	6489.08
Std. Dev.	5250.04	6173.61	7038.27	7634.13

Table 9. Time gap statistics across three sites

have to classify tells a more balanced story. Going from an 80% threshold to 60% for our eight week run with a training period increases the detection of new unique attacks by 37.6%, while increasing the newly seen unique false positives by 76.9%. In the three week run, the lower threshold adds no new unique false positives pointing to the need for threshold optimization once the system scales up. In fact, it lowers the utility of adding a new server since the existing ones detect additional attacks without it. However, as the number of web servers collaborating increases, this matching threshold along with the number of servers required to share an alert before reporting it as an attack should be key settings in order to optimize the system as they both key methods in this approach for controlling the false positive rate.

From the offline generated alert clusters, we conduct a temporal study of the alerts seen across the three servers. Firstly, we look at the time gap between alerts across sites. We compute the pairwise time gap of common alert clusters across the three servers. Additionally, we calculate the minimum time gap between alert clusters common to all of the three servers. Table 9 summarizes the minimum, maximum, average and standard deviations of the time gaps for the above cases. A better visual representation of the common alert clusters across all of the

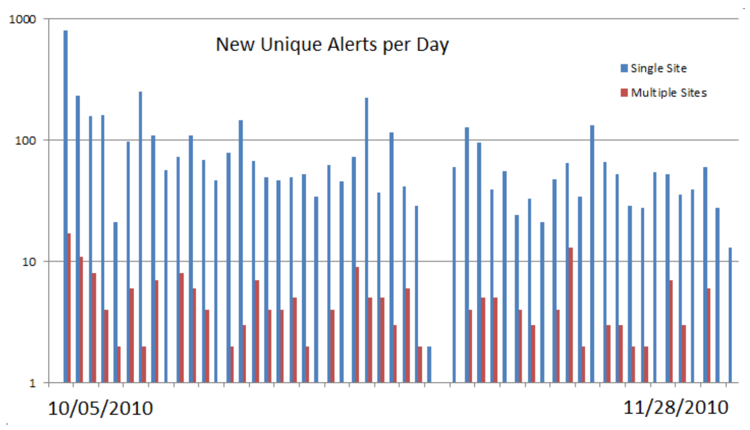


Fig. 7. Number of new unlabeled unique alerts per day that a human operator would have to parse. The large number of false positives from AD system is reduced by almost a magnitude difference when correlated to other sensors.

three servers is represented in Figure 6. The graph shows the minimum time gap between alerts observed at one server and the same alert being observed at the other two servers. The horizontal axis denotes the relative time elapsed since the start of observing the first alert. The vertical axis denotes the cluster. Each of the bars in the graph start at the time when an alert is observed at a site and ends at a time when it is seen first among the other two sites. The bar graphs are color coded to represent where the attack was first seen. From the statistics it can be seen that the average time gap between alerts could be used to our advantage. The results from the time gap analysis from the October-November run computed across CU and GMU shows a similar large average value (Min: 1.57min, Max: 71022.07min, Average: 15172.53min, Std. Dev.: 18504.44min). This gives us sufficient time to take preventive action at the collaborating sites by exchanging a small blacklist. Furthermore, we analyze the number of unclassified unique alerts that an operator has to manually classify every day. Figure 7 depicts the number of unique alerts generated daily. The graph shows both true positive and false positives observed using our collaborative approach alongside a stand alone approach. The horizontal axis denotes time in one day bins and the vertical axis denotes the frequency of alerts observed on a log scale. For the stand alone CAD sensor, a unique alert is included in the frequency when it is first observed at a site. However, for multiple sites collaborating, an alert is included in the frequency count at the time when it is confirmed to be seen at all sites. On average the number of unique alerts observed every day using a stand alone CAD sensor at CU is 82.84 compared to 3.87 alerts when a collaborative approach, over an order of magnitude in difference. Therefore, a collaborative approach clearly reduces the load on the operator monitoring alerts to an easily managed amount.

6 Conclusions

Web services and applications provide vital functionality but are often susceptible to remote zero-day attacks. Current defenses require manually crafted signatures which take time to deploy leaving the system open to attacks. Contrary, we can identify zero-day attacks by correlating Content Anomaly Detection (CAD) alerts from multiple sites while decreasing false positives at every collaborating site. Indeed, with a false positive rate of 0.03% the system could be entirely automated or operators could manually inspect the less than four new alerts per day on average that we observe in our eight week experiment. We demonstrate that collaborative detection of attacks across administrative domains, if done in a controlled and privacy preserving manner, can significantly elevate resources available to the defenders exposing previously unseen attacks.

References

1. Anagnostakis, K.G., Greenwald, M.B., Ioannidis, S., Keromytis, A.D.: Robust Reactions to Potential Day-Zero Worms through Cooperation and Validation. In: Proceedings of the 9th Information Security Conference (ISC). pp. 427–442 (August/September 2006)
2. Anagnostakis, K.G., Greenwald, M.B., Ioannidis, S., Keromytis, A.D., Li, D.: A Cooperative Immunization System for an Untrusting Internet. In: IEEE International Conference on Networks (2003)
3. Bloom, B.H.: Space/time trade-offs in Hash Coding with Allowable Errors. Communications of the ACM 13(7), 422–426 (1970)
4. Boggs, N., Hiremagalore, S., Stavrou, A., Stolfo, S.J.: Experimental results of cross-site exchange of web content anomaly detector alerts. In: Technologies for Homeland Security, 2010. HST '10. IEEE Conference on. pp. 8 –14 (Nov 2010)
5. Cretu, G., Stavrou, A., Locasto, M., Stolfo, S., Keromytis, A.: Casting out demons: Sanitizing training data for anomaly sensors. In: Security and Privacy, 2008. SP 2008. IEEE Symposium. pp. 81 –95 (may 2008)
6. Cretu-Ciocarlie, G., Stavrou, A., Locasto, M., Stolfo, S.: Adaptive Anomaly Detection via Self-Calibration and Dynamic Updating. In: Recent Advances in Intrusion Detection. pp. 41–60. Springer (2009)
7. Farroukh, A., Mukadam, N., Bassil, E., Elhajj, I.: Distributed and collaborative intrusion detection systems. In: Communications Workshop, 2008. LCW 2008. IEEE Lebanon. pp. 41 –45 (may 2008)
8. Gates, C.: Coordinated scan detection. In: Proceedings of the 16th Annual Network and Distributed System Security Symposium (NDSS 09) (2009)
9. Kruegel, C., Toth, T.: Distributed Pattern for Intrusion Detection. In: Network and Distributed System Security (NDSS) (2002)
10. Kruegel, C., Toth, T., Kerer, C.: Decentralized Event Correlation for Intrusion Detection. In: International Conference on Information Security and Cryptology (2002)
11. Kumar, S., Dharmapurikar, S., Yu, F., Crowley, P., Turner, J.: Algorithms to accelerate multiple regular expressions matching for deep packet inspection. In: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications. pp. 339–350. ACM (2006)

12. Lazarevic, A., Ozgur, A., Ertoz, L., Srivastava, J., Kumar, V.: A comparative study of anomaly detection schemes in network intrusion detection. In: In Proceedings of the Third SIAM International Conference on Data Mining (2003)
13. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics Doklady 10(8), 707–710 (1966), doklady Akademii Nauk SSSR, V163 No4 845-848 1965
14. Lin, P., Lin, Y., Lee, T., Lai, Y.: Using string matching for deep packet inspection. Computer 41(4), 23–28 (2008)
15. Locasto, M.E., Parekh, J.J., Keromytis, A.D., Stolfo, S.J.: Towards Collaborative Security and P2P Intrusion Detection. In: IEEE Information Assurance Workshop. West Point, NY (2005)
16. Norton, M., Roelker, D., Inc, D.R.S.: Snort 2.0: High performance multi-rule inspection engine
17. Paxson, V.: Bro: a system for detecting network intruders in real-time. In: SSYM'98: Proceedings of the 7th conference on USENIX Security Symposium. pp. 3–3. USENIX Association, Berkeley, CA, USA (1998)
18. Porras, P., Neumann, P.G.: EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. In: National Information Systems Security Conference (1997)
19. Sommer, R., Paxson, V.: Enhancing byte-level network intrusion detection signatures with context. In: CCS '03: Proceedings of the 10th ACM conference on Computer and communications security. pp. 262–271. ACM, New York, NY, USA (2003)
20. Sommer, R., Paxson, V.: Outside the closed world: On using machine learning for network intrusion detection. Security and Privacy, IEEE Symposium on 0, 305–316 (2010)
21. Song, Y., Keromytis, A.D., Stolfo, S.J.: Spectrogram: A mixture-of-markov-chains model for anomaly detection in web traffic. In: NDSS '09: Proceedings of the 16th Annual Network and Distributed System Security Symposium (2009)
22. Song, Y., Locasto, M.E., Stavrou, A., Keromytis, A.D., Stolfo, S.J.: On the infeasibility of modeling polymorphic shellcode. In: Proceedings of the 14th ACM conference on Computer and communications security. pp. 541–551. CCS '07, ACM, New York, NY, USA (2007), http://doi.acm.org/10.1145/1315245.1315312
23. Staniford-Chen, S., Cheung, S., Crawford, R., Dilger, M.: GrIDS - A Graph Based Intrusion Detection System for Large Networks. In: National Information Computer Security Conference. Baltimore, MD (1996)
24. Stavrou, A., Cretu-Ciocarlie, G.F., Locasto, M.E., Stolfo, S.J.: Keep your friends close: the necessity for updating an anomaly sensor with legitimate environment changes. In: AISec '09: Proceedings of the 2nd ACM workshop on Security and artificial intelligence. pp. 39–46. ACM, New York, NY, USA (2009)
25. Taylor, C., Gates, C.: Challenging the Anomaly Detection Paradigm: A Provocative Discussion. In: Proceedings of the 15^{th} New Security Paradigms Workshop (NSPW). pp. xx–yy (September 2006)
26. Tian, D., Changzhen, H., Qi, Y., Jianqiao, W.: Hierarchical distributed alert correlation model. In: IAS '09: Proceedings of the 2009 Fifth International Conference on Information Assurance and Security. pp. 765–768. IEEE Computer Society, Washington, DC, USA (2009)
27. Ullrich, J.: DShield home page (2005), http://www.dshield.org
28. Vasiliadis, G., Polychronakis, M., Antonatos, S., Markatos, E., Ioannidis, S.: Regular expression matching on graphics hardware for intrusion detection. In: Recent Advances in Intrusion Detection. pp. 265–283. Springer (2009)

29. Vigna, G., Gwalani, S., Srinivasan, K., Belding-Royer, E.M., Kemmerer, R.A.: An Intrusion Detection Tool for AODV-based Ad hoc Wireless Networks. In: Computer Security Applications Conference (2004)
30. Wang, K., Parekh, J.J., Stolfo, S.J.: Anagram: A Content Anomaly Detector Resistant to Mimicry Attack. In: Symposium on Recent Advances in Intrusion Detection. Hamburg, Germany (2006)
31. Websense: LizaMoon. `http://community.websense.com/blogs/securitylabs/archive/2011/03/31/update-on-lizamoon-mass-injection.aspx`
32. Xu, D., Ning, P.: Privacy-preserving alert correlation: a concept hierarchy based approach. In: Computer Security Applications Conference, 21st Annual. pp. 10 pp. –546 (dec 2005)
33. Yegneswaran, V., Barford, P., Jha, S.: Global Intrusion Detection in the DOMINO Overlay System. In: NDSS (2004)
34. Zaman, S., Karray, F.: Collaborative architecture for distributed intrusion detection system. In: Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on. pp. 1 –7 (july 2009)