Distinguishing between Web Attacks and Vulnerability Scans based on Behavioral Characteristics

Katerina Goseva-Popstojanova[1] and Ana Dimitrijevikj[*]

[1]Lane Dept. of Computer Science and Electrical Engineering, West Virginia University, Morgantown, WV

***Abstract*—The number of vulnerabilities and reported attacks on Web systems are showing increasing trends, which clearly illustrate the need for better understanding of malicious cyber activities. In this paper we use clustering to classify attacker activities aimed at Web systems. The empirical analysis is based on four datasets, each in duration of several months, collected by high-interaction honeypots. The results show that behavioral clustering analysis can be used to distinguish between attack sessions and vulnerability scan sessions. However, the performance heavily depends on the dataset. Furthermore, the results show that attacks differ from vulnerability scans in a small number of features (i.e., session characteristics). Specifically, for each dataset, the best feature selection method (in terms of the high probability of detection and low probability of false alarm) selects only three features and results into three to four clusters, significantly improving the performance of clustering compared to the case when all features are used. The best subset of features and the extent of the improvement, however, also depend on the dataset.**

***Keywords*-classification of malicious cyber activities; attacks; vulnerability scans; Web applications; honeypots.**

I. Introduction

Internet and Web technologies are widely used in almost every aspect of the modern society. Web applications, however, have many vulnerabilities that can be exploited by attackers. SANS [15] reported that 60% of the total attack attempts observed on Internet were against Web applications. New emerging Web 2.0 technologies enhance information sharing, collaboration, and functionality of the Web, but due to users ability to create content they also provide attackers with a broad range of new vulnerabilities to exploit. These trends clearly illustrate the need for better understanding of malicious cyber activities based on both qualitative and quantitative analysis, which will allow better protection, detection, and service recovery. However, the research work on intrusion detection in the past was mainly focused on development of data mining techniques aimed at constructing a "black-box" that classifies the network traffic on malicious and non-malicious, rather than on discovery of the nature of malicious activities [9]. Even more, significant amount of intrusion detection research work was based on outdated data sets (i.e., the DARPA Intrusion Detection Data Set and its derivative KDD 1999.) Analysis and classification of malicious activities have practical values only if they accounts for emerging technologies and new types of vulnerabilities.

[*]This work was done while Ana Dimitrijevikj was affiliated with West Virginia University.

In this paper we use behavioral clustering with a goal to distinguish malicious attack sessions from vulnerability scan sessions. In this context, a session is considered as an *attack* session if the attacker attempts to exploit a vulnerability in at least one request in that session. If all requests in the session were used to check for vulnerabilities then the session is considered as *vulnerability scan*. Although both attacks and vulnerability scans are malicious activities, it is important to be able to classify them automatically because attacks are much more critical events than vulnerability scans. Specifically, we explore the following research questions:

1) Can clustering be used to automatically distinguish between Web attacks and vulnerability scans?
2) Do attacks and vulnerability scans differ in a small number of features?

The main contributions of this paper are as follows:

(1) We use behavioral clustering analysis of malicious Web traffic in order to separate attack sessions from vulnerability scan sessions. Behavioral clustering has an advantage over previous efforts to automatically classify and analyze malicious behaviors, such as anti-virus software and intrusion detection systems, that were primarily focused on content-based signatures. These signatures are inherently susceptible to inaccuracies due to polymorphic and metamorphic nature of malicious code.

(2) We use four data sets, each in duration of 4-5 months, which allows us to compare the performance of clustering achieved in classifying malicious activities aimed at different system configurations and/or different time periods, and thus, to some extent, to generalize our observations.

(3) We use several feature selection methods in combination with K-means clustering to explore whether attacks differ from vulnerability scans only in a small subset of features. The results show that, depending on the feature selection algorithm and the dataset, anywhere from two to ten features can be used to perform clustering, typically with better performance than when all 43 features are used. The results further show that the best features differ across data sets collected by different system configurations.

(4) Overall, our results show that clustering, especially when the selected feature subsets are used, can be used to distinguish attack sessions from vulnerability scan sessions. The performance of the classification done by clustering, however, significantly differs across different datasets. Specifically, when the best feature selection methods in terms of high probability of detection and low probability of false alarm (i.e., high balance value) are chosen for each dataset, the probability of detection ranged from around 44% to close to 93% and the probability of false alarm was in the range from 0% to 39%.

Examples of areas that can benefit from methods for automatic classification of malicious behaviors include an automatic labeling of malicious traffic, generating signatures of new attacks, developing attack patterns for testing intrusion detection systems, and developing models for attack injections that can be used for testing services and systems.

II. Related Work

In the past only several papers used machine learning techniques for classification of some aspects of the malicious traffic. In [4] data collected by two high-interaction honeypots were used to analyze malicious attacks to port 445. Three classes of attacks were classified using the K-means algorithm. Two papers were focused on clustering system events collected during execution of sample malware programs, with a goal to automatically categorize the malware into groups that reflect similar classes of behaviors [2], [3]. Another recent work presented in [12] was focused on finding similarities among different samples of malicious HTTP traffic. [2], [3], [12] used anti-virus scanners to label the collected samples and applied single-linkage hierarchial clustering to group the malicious samples in classes with similar behaviors.

Two other papers used machine learning to analyze malicious behaviors. A decision tree was used in [10] to classify port scans observed by a Darknet (i.e., a set of passive sensors). [1] used Principal Component Analysis for characterizing the traffic collected by the Leurre.com project and presented the characteristics of the attacker activities for the first seven principle components.

In our recent work we used three supervised machine learning methods (i.e., J48, PART, and SVM) to first distinguish between attacks and vulnerability scans [7] and then to perform multiclass classification (i.e., distinguish among eleven vulnerability scan and nine attack classes) [8]. In both cases J48 and PART outperformed SVM and were able to classify the malicious activities with high probability of detection. However, supervised machine learning methods require data to be labeled, which is always time consuming, costly activity and in some cases is not feasible.

In this paper we use behavioral clustering with a goal to distinguish between attack and vulnerability scan sessions. Since clustering is unsupervised learning method it does not require the data to be labeled. Related work that used clustering distinguished among three types of attacks on port 445 [4] or aimed at grouping collected malware samples into malware families [2], [3], [12]. None of these related works was based on data collected by advertised, fully functional, three-tier honeypot systems, which allow collecting samples of typical attacks aimed at these systems. Even more, we identify the best subsets of the available features that are most useful for classification of malicious activities, thus identifying the simplest, most efficient model for each dataset. Related work papers either build behavioral profiles based on observing a small number of system events [2], [3], or used small number of features (i.e., four features in [4] and seven features in [12]) and did not use feature selection methods. Finally, we validate the clustering results using the four datasets labeled by a semiautomatic process. We assess the clustering performance in terms of several metrics: probability of detection, probability of false alarm, precision, and accuracy. This is important because the results show that when the target class (i.e., attacks) is in a minority, accuracy alone is a poor measure of learner's performance.

III. Data Collection and Extraction

Because data on recent malicious attacker activities were not publicly available, we developed and deployed high-interaction honeypots as a means to collect such data [5], [6]. Instead of a set of independent applications typical for honeypots in related work, our honeypots had meaningful functionality and followed a three-tier architecture consisting of a front-end Web server, application server, and a back-end database. Furthermore, they ran off-the-shelf operating systems and applications that followed typical security guidelines and did not include user accounts with weak or nil passwords. The honeypots were advertised using a technique called 'transparent linking' which involves placing hyperlinks pointing to the honeypots on public Web pages. This way the honeypots were indexed by search engines and Web crawlers, but could not be accessed directly by humans. Advertising honeypts that ran Web systems allowed us to observe typical malicious activities aimed at these systems, including attacks based on search engines. Overall, we ran honeypots with three different configurations and collected four datasets whose details are given next.

The first configuration ran Ubuntu 7.04, with Apache Web Server 2.2.3-3, PHP (version 5.2.1) as an application server, and MySQL database (version 5.0.38-0). As a Web application, we installed phpMyAdmin (version 2.9.1.1), which is a popular open source application widely used to handle database administration over the Web. *WebDBAdmin I* is the data set collected from this configuration.

The second configuration ran Windows XP Service Pack 2, with Microsoft IIS 5.1 Web server, PHP 5.0.2 server, and MySQL database (version 4.1). This configuration also included phpMyAdmin (version 2.9.1.1) as a Web application.

Table I. Breakdown of malicious Web Sessions for all datasets

	WebDBAdmin I sessions		WebDBAdmin II sessions		Web 2.0 I sessions		Web 2.0 II sessions	
Vulnerability scans: Total	**185**	**86.45%**	**513**	**93.44%**	**824**	**73.77%**	**2059**	**43.03%**
DFind	17	7.94%	19	3.46%	24	2.15%	20	0.42%
Other fingerprint	14	6.54%	3	0.55%			2	0.04%
Static	26	12.15%	305	55.56%	181	16.20%	327	6.83%
Blog					107	9.58%	690	14.42%
Wiki			1	0.18%	385	34.47%	922	19.27%
Blog & Wiki					73	6.54%	77	1.61%
Static & Blog					10	0.90%	1	0.02%
Static & Wiki					19	1.70%	3	0.06%
Static & Blog & Wiki					25	2.24%	3	0.06%
phpMyAdmin	77	35.98%	155	28.23%			11	0.23%
Static & phpMyAdmin	51	23.83%	30	5.46%			3	0.06%
Attacks: Total	**29**	**13.55%**	**36**	**6.56%**	**293**	**26.23%**	**2726**	**56.97%**
DoS					4	0.36%		
Password cracking phpMyAdmin user accounts	18	8.41%						
Password cracking Blog user accounts					9	0.81%	1	0.02%
Password cracking Wiki user accounts							71	1.48%
E-mail harvesting	5	2.34%						
Spam on Blog					23	2.06%	1411	29.49%
Spam on Wiki					249	22.29%	1055	22.05%
RFI			1	0.18%	4	0.36%	5	0.10%
SQL injection	1	0.47%			2	0.18%		
XSS					2	0.18%	11	0.23%
Other Attacks	5	2.34%	35	6.38%			172	3.59%
Total	**214**	**100%**	**549**	**100%**	**1,117**	**100%**	**4,785**	**100%**

WebDBAdmin II is the data set collected from this honeypot.

The third configuration ran the same operating systems and servers as the second. Instead of phpMyAdmin, two Web 2.0 applications were installed: the most widely used wiki software MediaWiki (version 1.9.0), which is used as an application base for Wikipedia, and the most downloaded open source blogging software Wordpress (version 2.1.1). From the honeypot with this configuration, we collected two data sets: *Web 2.0 I* and *Web 2.0 II*.

Web sessions, each defined as a sequence of requests from the same source IP address to port 80, with a time between two successive request not exceeding a threshold of thirty minutes [5], [6], were extracted automatically from the logs of the front-end Web servers (i.e., Apache and ISS). Since honeypots could not be accessed directly by human users because of the 'transparent linking' approach used for advertising, the only non-malicious sessions in the logs consisted of system management traffic generated by our team and legitimate Web crawlers such as Google and MSNbot. Removing the system management traffic was a trivial task. The crawlers were removed based on the IP addresses listed in iplists.com and other similar sites and based on manual inspection of the remaining traffic.

It should be noted that we use clustering in this paper, which is unsupervised method that do not use the labels in the learning process. However, the data have to be labeled in order to evaluate the performance of clustering in distinguishing between attacks and vulnerability scans. We used a semiautomated process based on identification of patterns in the HTTP application level logs, which allowed us to assure the accuracy and identify different classes of vulnerability scans and attacks. (The reader is referred to our previous work [6] for the details on labeling Web sessions.)

The breakdown of malicious Web sessions to different vulnerability scan and attack classes is shown in Table I. A brief overview across four data sets is as follows.

- WebDBAdmin I data set contains 214 Web sessions, 185 (86.45%) of which were labeled as vulnerability scans and 29 (13.55%) as attacks. The most dominant types of vulnerability scans were sessions labeled as phpMyAdmin and Static & phpMyadmin. Among 29 attack sessions, 18 were Password cracking of phpMyAdmin user accounts.
- WebDBAdmin II data set contains 549 sessions, 513 (93.44%) labeled as vulnerability scans and only 36 (6.56%) as attacks. The most frequent vulnerability scans classes were Static and phpMyAdmin. Out of 36 attack sessions, 35 were labeled as Other attacks.
- Web 2.0 I data set contains 1117 Web Sessions, 824 (73.77%) labeled as vulnerability scans and 293 (26.23%) as attacks. The most dominant vulnerability scans were fingerprinting the Wiki, Blog and Static content. The most frequent type of attack was Spam on Wiki with 249 sessions.
- Web 2.0 II data set contains 4785 Web sessions, out of which 2059 (43.03%) were vulnerability scans and 2726 (56.97%) were attacks. Vulnerability scans that browsed Static content and accessed the Blog or Wiki were dominant. Among attacks, 1411 posted Spam on Blog and 1055 posted Spam on Wiki.

We characterize each Web session with a vector of the following 43 different features (i.e., session characteristics): (1) number of requests; (2) bytes transferred; (3) duration (in seconds); (4)-(8) mean, median, minimum, maximum, and standard deviation of the time between requests; (9)-(14) number of requests with a specific method type (i.e., GET, POST, OPTIONS, HEAD, PROPFIND, and other); (15) number of requests to picture files (e.g., .jpeg, .jpg, .gif, .ico, .png); (16) number of requests to video files (e.g., .avi, .mpg, .wmv); (17) number of requests to static application files (e.g., .html, .htm); (18) number of requests to dynamic application files (e.g., .php, .asp); (19) number of requests to text files (e.g., .txt, .ini, .css); (20)-(24) number of requests with specific status code (i.e., Informational (1xx), Success (2xx), Redirect (3xx), Client error (4xx), and Server error(5xx)); (25)-(29) mean, median, minimum, maximum, and standard deviation of the length of all request substrings within a session; (30)-(34) mean, median, minimum, maximum, and standard deviation of the number of parameters passed to application within a session; boolean indications of whether: (35) robots.txt file was accessed in that session; (36) it was a night session; (37) there was a remote site injection in at least one request; (38) a semicolon was used to divide multiple passed attributes to an application in at least one request; (39) a string containing suspicious encoding in any of the requests; (40) a reserved character was used in any of the requests; (41) an ASCII control character was used in any of the requests; (42) a non-ASCII control character was used in any of the requests; and (43) an invalid encoding was used in any of the requests.

IV. Our Behavioral Clustering Approach

Since the ranges of the 43 features differed significantly we first applied Min Max Normalization, resulting in a new range $[0, 1]$ for each feature. We used K-means clustering with Euclidian distance. The number of clusters K was determined using a method proposed in [13] which is based on minimizing the ratio of intra-cluster and inter-cluster distance.

We performed K-means clustering first using all 43 features and then on subsets of features selected by several feature selections methods. The motivation for using feature selection was to explore whether a small subset of session characteristics can be used to efficiently separate attack from vulnerability scan sessions. In addition, reducing the number of features by removing the irrelevant and noisy features has a potential to speed up the machine learning algorithms and improve their performance [11]. We used the following feature selection methods:

- *Information gain.* First, features were ranked from the most informative to least informative using the information gain as a measure [11], and then the three features with the highest information gain were selected.
- *Sequential Forward Selection (SFS).* This method starts with an empty set and adds one feature at a time, which with previously added feature(s) gives the best performance. We used two variants of SFS, one using Support Vector Machine (SVM) and another using J48 algorithm for evaluation.
- *Feature selection methods for clustering* perform the selection without the class information. In this paper we used one such selection method which results in a method for sparse K-means clustering [14].

To evaluate the performance of clustering, we counted how many attacks and vulnerability scans were in each cluster. If attack sessions dominated, the cluster was labeled as attack cluster, whereas if vulnerability scan sessions were the majority, we assigned a vulnerability scan label to that cluster. After all clusters were labeled, we computed the values of true negatives (TN), false negatives (FN), false positives (FP), and true positives (TP) in the confusion matrix.

	Actual: Vulnerability Scan	Actual: Attack
Predicted: Vulnerability Scan	TN	FN
Predicted: Attack	FP	TP

Then, we computed the following metrics to assess the performance of clustering in detecting attacks:

$$\text{accuracy (acc)} = (TN{+}TP)/(TN{+}FN{+}FP{+}TP) \quad (1)$$

$$\text{probability of detection (pd)} = TP/(FN + TP) \quad (2)$$

$$\text{probability of false alarms (pf)} = FP/(TN + FP) \quad (3)$$

$$\text{precision (prec)} = TP/(TP + FP) \quad (4)$$

$$\text{balance (bal)} = 1 - \sqrt{(0 - pf)^2 + (1 - pd)^2}/\sqrt{2} \quad (5)$$

The *accuracy*, given with (1), provides the percentage of sessions that are detected correctly. *Probability of detection*, defined by (2), which is also called *recall*, accounts for the probability of detecting an attack (i.e., the ratio of detected attacks to all attacks). *Probability of false alarm*, given by (3), is the the ratio of vulnerability scans misclassified as attacks to all vulnerability scans. *Precision*, defined by (4), determines the fraction of sessions correctly classified as attacks out of all sessions classified as attacks. Ideally, we want probability of detection to be 1 and probability of false alarm to be 0. *Balance* is defined as the Euclidian distance from this ideal point of $pf = 0, pd = 1$ to a pair of (pf, pd). For convenience, the balance is normalized by the maximum possible distance across the ROC square $\sqrt{2}$ and then subtracted from 1 (see (5)). It follows that higher balance is better since (pf, pd) point falls closer to the ideal point $(0, 1)$.

Table II. Summary of clustering results for all data sets

		WebDBAdmin I	WebDBAdmin II	Web 2.0 I	Web 2.0 II
All features	K	3	8	10	8
	pd	17.00%	19.00%	63.00%	91.00%
	pf	0.00%	0.30%	0.20%	64.00%
	prec	100.00%	77.77%	98.93%	65.31%
	bal	*41.00%*	42.00%	*73.00%*	54.00%
	acc	89.00%	94.00%	90.00%	67.00%
	# of features	43	43	43	43
Information gain	K	4	4	4	4
	pd	72.41%	0.00%	87.71%	92.99%
	pf	0.54%	0.00%	0.48%	39.46%
	prec	95.45 %	0.00%	98.46%	75.70%
	bal	**80.48%**	*29.28%*	**91.30%**	**71.65%**
	acc	95.79%	93.44%	96.41%	79.01%
	# of features	3	3	3	3
	selected features	9, 1, 18	24, 28, 26	10, 28, 26	28, 25, 10
SFS with SVM	K	5	3	3	6
	pd	65.52%	44.44%	78.16%	90.35%
	pf	0.00%	0.00%	0.36%	69.47%
	prec	100.00%	100.00%	98.70%	76.78%
	bal	75.62%	**60.72%**	84.55%	*50.41%*
	acc	95.33%	96.36%	94.00%	64.60%
	# of features	5	3	5	7
	selected features	10, 18, 23, 30, 34	10, 37, 39	10, 14, 18, 38, 40	2, 6, 9, 10, 15, 27, 33
SFS with J48	K	5	4	2	6
	pd	58.62%	8.33%	87.71%	48.15%
	pf	1.08%	0.58%	4.98%	29.17%
	prec	89.47%	50.00%	86.24%	68.58%
	bal	70.73%	35.18%	90.63%	57.93%
	acc	93.46%	93.44%	93.11%	57.91%
	# of features	2	6	4	10
	selected features	2, 19	2, 9, 24, 27, 28, 38	10, 28, 31, 33	2, 8, 9, 21, 27, 28, 30, 31, 36, 38
Sparse K-means clustering	K	5	3	6	5
	pd	17.24%	0.00%	62.12%	92.29%
	pf	0.00%	0.00%	0.00%	64.76%
	prec	100.00%	0.00%	100.00%	65.34%
	bal	41.48%	29.29%	73.21%	53.89%
	acc	88.79%	93.44%	90.06%	67.73%
	# of features	43	43	43	43

V. Main Observations

Table II summarizes the behavioral clustering results for all four datasets, including clustering based on all 43 features and clustering with each of the four feature selection methods. The method that gave the best (worst) balance value for each data set is highlighted in **bold** (*italic*). It should be noted that the feature selection method that resulted in the best balance value also led to the best probability of detecting attacks and close to or smaller probability of false alarms than when all features were used. Figure 1 presents, for each data set, the performance of different variants in terms of the probability of false alarm versus probability of detection (i.e., in the ROC square).

Based on these results, it is evident that information gain, SFS with SVM, and SFS with J48 significantly improved the performance of clustering for at least two datasets compared to the case when all features were used. On the other side, the sparse K-means clustering method performed similarly or even worse than when all features were used and it did not reduce the feature set.

The best feature selection methods in terms of the highest balance value are: information gain for WebDBAdmin I, Web 2.0 I and Web 2.0 II data sets, and SFS with SVM for WebDBAdmin II data set. Each of the best feature selection methods selected three features and resulted in three to four clusters. Table III presents the distribution of attack and vulnerability scan sessions across clusters produced by the best selection method for each data set.

Next, we summarize the results per dataset and discuss the reasons behind the achieved clustering performance. Both WebDBAdmin I and II datasets had very low probability of false alarm (close or equal to 0%). When using the best feature selection method (i.e., information gain) WebDBAdmin I data set had relatively high probability of detection and thus balance. This method led to two clusters with attack labels C_3 and C_4 (see Table III) and only one vulnerability scan session was misclassified as attack (see C_4). The two vulnerably scan clusters C_1 and C_2 contained very few attacks misclassified as vulnerability scans.

On the contrary, WebDBAdmin II dataset had rather low probability of detection and correspondingly the lowest balance. This is due to the fact that WebDBAdmin II dataset

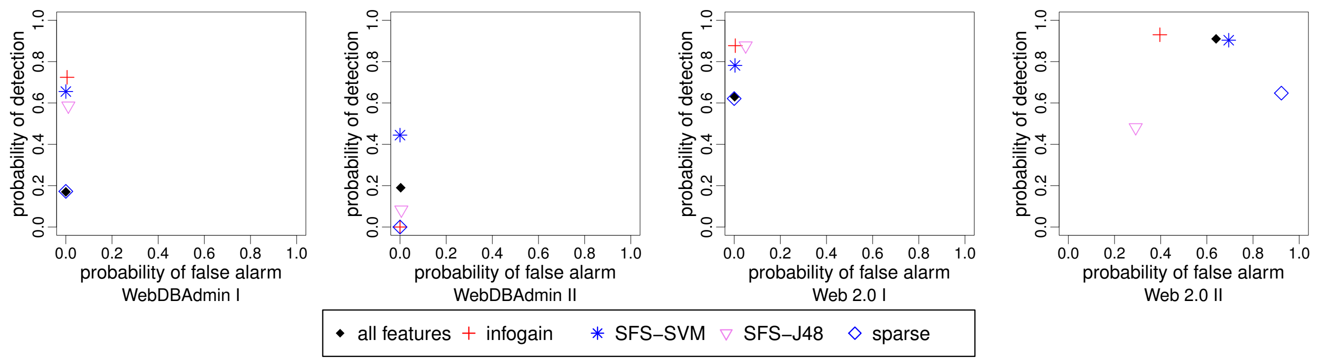

Figure 1. Comparison in ROC space for WebDBAdmin I, WebDBAdmin II, Web 2.0 I, and Web 2.0 II data sets

Table III. Clustering results using the best feature selection method for each dataset

Data set	Feature selection method	Selected features	Session type	C_1	C_2	C_3	C_4	Total
WebDBAdmin I	Information gain	9, 1, 18	Vulnerability scans	**32**	**152**	0	1	185
			Attacks	8	0	**5**	**16**	29
WebDBAdmin II	SFS with SVM	10, 37, 39	Vulnerability scans	**513**	0	0		513
			Attacks	20	**13**	**3**		36
Web 2.0 I	Information gain	10, 28, 26	Vulnerability scans	**178**	4	**642**	0	824
			Attacks	5	**171**	31	**86**	293
Web 2.0 II	Information gain	28, 25, 10	Vulnerability scans	498	315	**1033**	**214**	2060
			Attacks	**1448**	**1086**	1	190	2725

had the smallest percentage of attacks (only 6.65%), with many different types that could not be separated well by clustering. In particular, 16 attacks were divided between the two attack clusters C_2 and C_3, but the remaining 20 attacks were misclassified as vulnerability scans and assigned to the vulnerability scan cluster C_1, which led to low probability of detection. WebDBAdmin II dataset clearly illustrates that accuracy can be a misleading measure of learner's performance. Thus, for this data set, in spite of the poor probability of detection (at most 44%) the accuracy was very high (over 93% in all cases).

Web 2.0 I data set had the highest balance value compared to the other datasets, regardless of the method used (i.e., the points in the ROC space (see Fig. 1) are close to the ideal point $pf = 0, pd = 1$). Apparently, attack sessions and vulnerability scans sessions in Web 2.0 I data set can be separated well. In case of the best feature selection method (i.e., information gain), as it can be seen from Table III, C_2 and C_4 were labeled as attack clusters, with only 4 vulnerability scan sessions in C_2 misclassified as attacks (i.e., low probability of false alarm). *Spam on wiki* attack sessions dominated both C_2 and C_4, which indicated that from behavioral perspective there were two different types of sessions which posted spam on wiki.

Web 2.0 II data set had very high probability of detection (over 90% for all methods except SFS with J48), but also significantly higher false alarm rate (29-69%) than any other dataset (which had false alarm rates close to 0%). Thus, K-means clustering with feature selection based on information gain (see Table III) successfully separated the attacks sessions, with cluster C_1 containing close to 100% of *spam on blog* sessions and cluster C_2 containing around 91% of all *spam on wiki* sessions, which led to high probability of attack detection. However, these two clusters also contained relatively large number of vulnerability scan sessions (mostly scans to blog and/or wiki) misclassified as attacks, which led to the moderately high probability of false alarm. In addition, we observed some attack sessions misclassified as vulnerability scans in cluster C_4. Having in mind the last two observations, it is not surprising that Web 2.0 II data set had the lowest accuracy.

It should be noted that the classification based on supervised learning methods presented in our earlier work [7] had better performance than the classification based on clustering presented in this paper. However, supervised learning methods require datasets to be labeled, which is not the case with unsupervised learning such as clustering.

VI. Concluding Remarks

In this paper we used behavioral clustering with a goal to distinguish between attack sessions and vulnerability scan sessions aimed at Web systems. In particular, we used K-means clustering in combination with several feature selection methods on four data sets collected by three different honeypot configurations.

Our results showed that clustering analysis can be used to separate attacks from vulnerability scans in malicious Web traffic, with a small number of clusters (i.e., from two to ten clusters depending on the method and dataset). Feature selection methods typically improved the performance of clustering, resulting in better probability of detecting attacks and accuracy, and similar or smaller probability of false alarm compared to the case when all features were used. It follows that attacks and vulnerability scans differ in a small number of features. Therefore, when redundant and noisy features are removed the K-means clustering not only works faster, but it also gives better results in terms of separating attacks from vulnerability scan sessions.

Typical to machine learning techniques, the success of clustering in separating attacks from vulnerability scans differed across datasets. The best feature selection method resulted in probability of detection ranging from 44% in the case of the data set with a very small percentage of attacks (i.e., only 6.56%) to over 80% or even 90% in data sets with more evenly distributed classes.

Another conclusion is that using only the overall accuracy to measure the performance of clustering may be misleading, especially if the data set has uneven class distributions. Using additional metrics such as probability of detection, probability of false alarm, precision, and balance provides much better assessment of the performance of clustering. For example, although the overall accuracy in the case of the WebDBAdmin II dataset was very high, the probability of detecting attacks was low since the attack class consisted of small number, very diverse attacks.

The best feature selection methods in terms of balance (which in our case also had the highest probability of detection and similar or lower probability of false alarm) selected only three features for each data set. The sets of best features, however, differed across datasets (see Table III 'Selected features' column). Thus, the number of requests with POST method feature (10) was common across three out of four datasets. The other two features in the Web 2.0 I and Web 2.0 II datasets were related to the length of requests substrings (i.e., (25), (26) and (28)). On the other side, features such as number of requests with GET method (9), number of requests (1), number of requests to dynamic application files (18), presence of remote site injection (37) and string containing suspicious encoding (39) were good predictors for the WebDBAdmin I and WebDBAdmin II datasets. This observation can be explained by the fact that attackers often use search-based strategy to identify their targets and therefore different systems are exposed to different malicious activities. Rather than advocating a particular subset of features as being the best subset across all data sets, we suggest running combinations of a clustering algorithm and feature selection methods to find the most appropriate subset of features for a particular system configuration and/or time period.

The presented results enrich the empirical evidence on malicious cyber activities and can help areas such as automatic labeling of malicious activities, developing attack patterns for testing intrusion detection systems, and developing models for attack injection that can be used for testing services and systems.

ACKNOWLEDGMENTS

This work is funded in part by the National Science Foundation under the grants CNS-0447715 and CCF-0916284. The authors thank David Krovich, Jonathan Lynch, J. Alex Baker, Brandon Miller, and Risto Pantev for their support with the experimental setup and data collection.

REFERENCES

[1] S. Almotairi, A. Clark, G. Mogay, J. Zimmermann, "Characterization of attackers' activities in honeypot traffic using Principal Component Analysis," *IFIP Int'l Conf. Network and Parallel Computing*, 2008, pp. 147-154

[2] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, and J. Nazario, "Automated classification and analysis of Internet malware", *Recent Advances in Intrusion Detection (RAID), LNCS 4637*, 2007, pp. 178-197

[3] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel and E. Kirda, "Scalable, behavior-based malware clustering", *Network and Distributed System Security Symp.* 2009

[4] M. Cukier, R. Berthier, S. Panjwani and S. Tan, "A statistical analysis of attack data to separate attacks", *36th Int'l Conf. Dependable Systems & Networks (DSN)*, 2006, pp. 383-392

[5] K. Goseva-Popstojanova, B. Miller, R. Pantev and A. Dimitrijevikj, "Empirical analysis of attackers' activity on multi-tier Web systems", *24th IEEE Int'l Conf. Advanced Information Networking and Applications (AINA)*, 2010, pp. 781-788

[6] K. Goseva-Popstojanova, R. Pantev, A. Dimitrijevikj and B. Miller, "Quantification of attackers activities on servers running Web 2.0 applications", *9th IEEE Int'l Symp. Network Computing and Applications (NCA)*, 2010, pp. 108-116

[7] K. Goseva-Popstojanova, G. Anastasovski and R. Pantev, "Classification of malicious Web sessions", *21st International Conference on Computer Communication Networks (ICCCN 2012)*, 2012, pp.1-9

[8] K. Goseva-Popstojanova, G. Anastasovski and R. Pantev, "Using multiclass machine learning methods to classify malicious behaviors aimed at Web systems", *23rd IEEE International Symposium on Software Reliability Engineering (ISSRE 2012)*, 2012, pp. 81-90

[9] K. Julisch, "Data mining for intrusion detection – A critical review" *Applications of Data Mining in Computer Security, Advances in Information Security*, D. Barbara and S. Jajodia (Editors), Springer, 2002, pp. 33-62

[10] H. Kikuchi, M. Terada, T. Pikulkaew, "Automated classification of port scans from distributed sensors," *22nd Int'l Conf. Advanced Information Networking and Application (AINA)*, 2008, pp. 771-778

[11] H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering", *IEEE Trans. Knowl. Data Eng*, vol.17, no.4, 2005, pp. 491-502

[12] R. Perdisci, W. Lee and N. Feamster, "Behavioral clustering of HTTP-based malware and signature generation using malicious network traces", *7th USENIX Symp. Networked Systems Design and Implementation (NSDI)*, 2010, pp. 26-26

[13] S. Ray and R. H. Turi, "Determination of number of clusters in K-means clustering and application in colour image segmentation", *Conference on Advances in Pattern Recognition and Digital Techniques (ICAPRDT)*, 1999, pp. 137-143

[14] D. Witten, R. Tibshirani, "A framework for feature selection in clustering", *J Am Stat Assoc*, vol. 105, no. 490, 2010, pp. 713-726

[15] SANS Top Security Risks, http://www.sans.org/top-cyber-security-risks/summary.php