Classification of malicious Web sessions

Katerina Goseva-Popstojanova[1], Goce Anastasovski[1], and Risto Pantev[2,*]

[1]Lane Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown, WV
[2]Microsoft, Redmond, WA

***Abstract*—The ever increasing number of vulnerabilities and reported attacks on Web systems clearly illustrate the need for better understanding of malicious cyber activities, which will allow better protection, detection, and service recovery in the cyberspace. In this paper we use three supervised machine learning methods, Support Vector Machines (SVM), and decision trees based J48 and PART, to classify attacker activities aimed at Web systems. The empirical analysis is based on four datasets, each in duration of four to five months, collected by high-interaction honeypots. Malicious Web sessions are characterized with forty three different features (i.e., session attributes) extracted from Web server logs. Our results show that the supervised learning methods can be used to efficiently distinguish attack sessions from vulnerability scan sessions, with very high probability of detection and very low probability of false alarms. Furthermore, we follow the principle of Occam's razor, that is, we seek for the simplest possible model that can successfully classify malicious Web sessions. Our results show that attacks differ from vulnerability scans only in a small number of features (i.e., session attributes). In particular, depending on the data set, classification of malicious activities can be performed using from four to six features without significantly affecting learners' performance compared to when all 43 features are used. Decision tree based methods J48 and PART perform better than SVM across all datasets.**

Keywords: classification of malicious cyber activities, attacks, vulnerability scans, Web applications, honeypots.

I. Introduction

Many businesses are using Internet and Web technologies to build new communication channels with customers around the globe. Web applications, however, have many vulnerabilities that can be exploited by attackers. SANS reported that 60% of the total attack attempts observed on Internet were against Web applications [25]. 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. The increasing popularity of Web 2.0 based applications, such as blogs, wikis, and social sites, makes them even more attractive targets for attackers. 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.

Unlike the long tradition and great success in characterization of the regular network traffic and server workloads, until recently not much research was focused on characterization and quantification of malicious attacker behaviors. One evident reason for this is the lack of publicly available, good quality, recent data on cyber security threats and malicious attacker activities. Although a significant amount of research work on intrusion detection exist, it 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 [13]. Last but not least, significant amount of intrusion detection research work was based on outdated data sets such as the DARPA Intrusion Detection Data Set [8] and its derivative KDD 1999 [14]. To be of practical value, the analysis of malicious activities have to account for emerging technologies that typically introduce new types of vulnerabilities.

Motivated by the lack of publicly available datasets that incorporate attacker activities related to recent technologies and new types of vulnerabilities, we developed and deployed high-interaction honeypots as a means to collect such data. Our honeypots were configured in a three-tier architecture (consisting of a front-end Web server, application server, and a back-end database) and had meaningful functionality [10], [11]. Furthermore, they ran standard off-the-shelf operating system and applications which followed typical security guidelines and did not include user accounts with nil or weak passwords. This approach allowed us to collect datasets of malicious activities aimed at realistic systems with different configurations. The data collected by our honeypots are grouped into four datasets, each in duration of four to five months. Each dataset consists of malicious Web sessions extracted from application level logs of systems running on the Internet. Therefore, our datasets represent dynamic information on attacker activities, unlike data extracted from vulnerability databases, such as [6], [17], [26], that are focused on static information related to description of known vulnerabilities and the ways they may be exploited.

Two of the four datasets considered in this paper were used in our previous work [10], [11] which was focused on descriptive analysis of different types of attacks and vulnerability scans and inferential statistical analysis of characteristics of malicious TCP [10] and HTTP traffic [11]. Neither [10] nor [11] included classification of malicious traffic using machine learning methods. The two additional datasets were collected since the publication of [10], [11].

In this paper we use supervised machine learning methods

*The work was done while Risto Pantev was affiliated with West Virginia University.

978-1-4673-1544-9/12/$31.00 ©2012 IEEE

with a goal to automatically classify malicious Web sessions on attacks and vulnerability scans. In this context, a Web 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*. We characterize each Web session with 43 features which reflect different session characteristics such as the number of requests in a session, number of requests with a specific method type (e.g., GET, POST, OPTIONS), number of requests to dynamic application files, length of requests substrings within a session, and so on.

Both attacks and vulnerability scans are malicious activities. However, being able to automatically classify them is very important because actual attacks are much more critical events than vulnerability scans. It should be noted that our goal is not to identify whether attacks are preceded by vulnerability scans. Rather, our goal is to distinguish between them, regardless of their temporal order and origin. The fact that our datasets consist only of malicious Web sessions allows us to classify and study the characteristics of malicious Web traffic without the "noise" of regular, non-malicious traffic. Furthermore, although we use machine learning for classification, our goals are very different from standard classification to malicious and non-malicious traffic employed in intrusion detection systems.

In general, our work is based on the hypotheses that different malicious activities exhibit different behavioral patterns, which provides bases for using machine learning methods for their classification. Specifically, we explore the following research questions:

A) Can supervised machine learning methods be used to automatically distinguish between Web attacks and vulnerability scans?
B) Do attacks and vulnerability scans differ in a small number of features? If yes, are these subsets of "best" features consistent across different data sets?
C) Do some learners perform consistently better than other across multiple datasets?

The main contributions of this paper are as follows:

- Our results show that supervised machine learning methods can be used to separate attack sessions from vulnerability scan sessions with very high probability of detection and very low probability of false alarm. The analysis is based on four data sets which allows us to compare the performance of machine learning methods in classifying malicious activities aimed at different system configurations and/or different time periods, and thus, to some extent, to generalize our observations.
- We seek for the simplest possible model that can distinguish attacks from vulnerability scans. The results based on using information gain as a feature selection method show that a small subset of four to six features can be used to classify malicious activities without significantly worsening learners' performance compared to when all 43 features are used, which indicates that attacks differ from vulnerability scans only in a small number of features (i.e., session characteristics).
- Although there is some consistency across datasets regarding which features are the best predictors, we also observe differences, especially for systems running different applications. Hence, instead of advocating a particular subset of features as the best predictors for all datasets, our study suggests that classification of malicious activities should include feature selection method that helps identifying the best subset for a particular system.
- Decision tree algorithms J48 and PART perform better than SVM across all datasets considered in this paper. The pruned versions of J48 and PART have close, and in some cases even better performance than the unpruned tree versions.

Automatic classification and analysis of malicious behaviors can be used for example to support the generation of attack signatures or to develop attack patterns for testing system resilience to attacks. With the increase in the number and diversity of malicious behaviors on Internet, automatic classification holds enormous potential for improving the protection and resiliency of services and systems.

The paper is organized as follows. The related work is presented in Section II. Section III presents the data collection and extraction process. Our data mining approach and the main results are presented in Sections IV and V, respectively. The concluding remarks are given in Section VI.

II. Related Work

Significant amount of work in the past was focused on using different data mining methods for intrusion detection, that is, for classification of network traffic to malicious and non-malicious. These methods typically are classified either as misuse detection (which searches for known patterns of attacks) or anomaly detection (which starts with defining the expected normal behavior, and then reports as possible attacks any significant deviations from the normal profile). Examples of methods for both approaches can be found in [13], [18].

Data mining and entropy-based techniques were used in [28] to build behavior profiles of Internet backbone traffic. The authors reported that a large majority of the clusters fell into three profiles: typical server/service behavior (mostly providing well-known services), typical heavy-hitter host behavior (predominantly associated with well-known services), and typical scan/exploit behavior (frequently manifested by hosts infected with known worms).

Using data mining techniques for classification of some aspects of malicious traffic is an emerging recent trend. In [7] data collected by two high-interaction honeypots were used to analyze malicious attacks to port 445. That work was focused on distinguishing among three types of attacks using the K-means clustering algorithm. A decision tree was used in [15] to classify port scans observed by a Darknet (i.e., passive sensors) to three classes: random, periodic, and intensive. Two recent 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], [4]. Another recent work presented in [21] was focused on finding similarities among different samples of malicious HTTP traffic. [2], [4], [21] 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.

Our work differs from the related work in several ways. First, none of the 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. Second, while we use machine learning techniques as several recent papers that dealt with studying malicious behaviors, we classify malicious activities with a goal to distinguish between attack and vulnerability scan sessions, unlike [7] which distinguished among three types of attacks on port 445 or [2], [4], [21] which grouped malware programs with similar behaviors. Even more, we identify the best subsets consisting of a small number of features that are most useful for classification of malicious activities, thus identifying the simplest, most efficient model for each data set. Related work papers did not use feature selection methods; they either built behavioral profiles based on observing a small number of system events [2], [4], or used small number of features (i.e., four features in [7] and seven features in [21]). Finally, unlike related work, we use multiple learners and compare their performance on four datasets, which is important for generalizability of the results.

III. Data Collection and Extraction

Facing the lack of publicly available, recent data on malicious attacker activities, we developed and deployed high-interaction honeypots as a means to collect such data [10], [11]. These honeypots ran off-the shelf operating systems and applications that followed typical security guidelines and did not include user accounts with nil or weak passwords. Furthermore, instead of a set of independent applications typically installed on honeypots, 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. The honeypots were advertised using a technique called 'transparent linking' which involves placing hyperlinks pointing to the honeypot on public Web pages, so that it is indexed by search engines and Web crawlers, but cannot 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.

It should be noted that our data collection approach, although based on honeypots, is complementary to other existing approaches based on honeypots which were deployed either for the sole purpose of being compromised in order to assess the adversaries [1], [24] or to cover a large range of IP addresses (by passive monitoring of the unused address space or by using active responders) in order to collect information on malicious activities such as worm outbreaks and botnet sweeps [3], [27]. Many non-random attacks that spread along application-specific topologies (e.g., Web, peer-to-peer, instant messenger, e-mail) and carefully select their victims likely will never be observed in the unused address space or a honeyfarm of active responders [27].

Overall, we ran honeypots with three different configurations and collected four data sets whose details are given next. The first configuration ran a default installation of 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 June 2 to September 28, 2008 on the honeypot with 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. *WebDBAdmin II* is the data set collected from this honeypot from August 17, 2009 to January 17, 2010.

The third configuration was identical to the second with respect to the operating systems and servers, including the static Web content. 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* from March 30 to July 26, 2009 and *Web 2.0 II* from August 17, 2009 to January 17, 2010.

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 [10], [11], 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.

To identify different classes of vulnerability scans and attacks we first automatically identified all unique malicious requests in the HTTP application level logs. Then, we examined different fields in these requests and manually assigned the specific classes of attacker activities. This process included using the descriptions provided by the Open Web Application Security Project (OWASP) [19] and searching public databases such as [17] and [26]. (Further details on Web sessions labeling can be found in our previous work [11].) The breakdown of malicious Web sessions to different vulnerability scan and attack classes is shown in Table I. It should be noted that, for

the brevity of the presentation, the classes shown in Table I were obtained by grouping much finer grain classes used during the labeling process.

We characterize each Web session with a vector of 43 different features (i.e., session characteristics), which were extracted automatically for each dataset. These 43 features extend the feature sets used in [7] and [21] (consisting of four and seven features, respectively) by considering features similar to those used in articles on network and Web server intrusion detection and Web crawlers identification. (Further details and references are given in [20].) The complete list of 43 features is as follows: (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 requests' 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 any request of that session; (36) it was a night session (between 12 am to 8 am local time); (37) there was a remote site injection in at least one request; (38) a semicolon was used to divide multiple attributes passed 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. Data Mining Approach

Since the ranges of the 43 features differ significantly we first apply Min Max Normalization, resulting in a new range $[0, 1]$ for each feature.

To classify the observed malicious traffic to attacks and vulnerability scans we use three different supervised machine learning methods: Support Vector Machines (SVM), decision tree J48, and a rule induction method based on partial decision trees (PART). SVM implicitly maps input feature vectors to a higher dimensional space by using a kernel function [5]. In the transformed space, a maximal separating hyperplane is built considering a two class problem. The choice of the kernel function is crucial for creating good class separating hyperplanes. Based on [12] we use the Radial Basis Function (RBF) as a kernel function. J48 is a Java implementation of the C4.5 decision tree algorithm [22], which divides the feature space successively by choosing primarily features with the highest information gain. Instances are classified by sorting them from the root to some leaf node, which specifies the class of the instance. Each node of the tree specifies a test of some feature and each branch from that node corresponds to one of the possible values for that feature. PART [9] infers rules by repeatedly generating partial decision trees by adopting the divide-and-conquer strategy of RIPPER and combining it with the decision tree approach of C4.5. After generating each rule, the partial decision tree is discarded which avoids early generalization. For the tree based methods, J48 and PART, we also build the pruned trees, which is a fundamental step in optimizing the computational efficiency and classification accuracy of a tree model. When pruning methods are applied the resulting tree is usually reduced in size or number of nodes in order to avoid unnecessary complexity and over-fitting of the data. For both J48 and PART we used Reduced Error Pruning (REP) [23].

To evaluate learners' performance, we first compute the confusion matrix

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

where TN, FN, FP, and TP denote true negatives, false negatives, false positives, and true positives, respectively. Then, we compute the following metrics which assess different aspects of the classification:

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

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

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

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

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

The *accuracy*, given with (1), provides the percentage of sessions that are detected correctly. Since attacks are more critical events than vulnerability scans *probability of detection* defined by (2), which sometimes 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*, defined 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. It appears that a rather useful metric of performance is so called *balance*, which 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 $\sqrt{2}$

TABLE I. BREAKDOWN OF MALICIOUS WEB SESSIONS FOR ALL DATA SETS

	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%**

and then subtracted from 1 (see (5)). It follows that higher balance is better since (pf, pd) point falls closer to the ideal point $(0, 1)$.

Besides applying the learners to all 43 features we also employ a feature selection method. The motivation for using feature selection is to explore whether a small subset of session characteristics can be used to efficiently separate attack sessions from vulnerability scan sessions. In addition to learning about characteristics of malicious activities, reducing the number of features by removing the irrelevant and noisy features speeds up the machine learning algorithms and hopefully improves their performance [16]. In this paper we use information gain feature selection method which ranks the features from the most informative to least informative using the information gain as a measure [16]. This is a filter selection method because it uses the characteristics of the data to evaluate the features (i.e., does not use any learning algorithm).

Since our goal is to identify the smallest number of features sufficient to accurately distinguish attacks from vulnerability scans, we use the following procedure for each individual dataset. For each learner, we start with the highest ranked feature and include one feature at a time until reaching less than or equal to 1% difference of the probability of detection (pd) compared to the case when all 43 features are used. Then, we choose the smallest set of features among all learners for that dataset and use it to test the performance of the remaining learners.

In this paper we use 10-fold cross-validation which involves partitioning the dataset into ten complementary subsets, and then using nine subsets as training data and validating the results on the remaining subset (called validation data or testing data). This process is repeated ten times, with each of the ten subsets used exactly once as validation data. The advantage of this method over repeated random sub-sampling is that all observations are used for both training and validation. The results presented in this paper are the averages of the ten results from the folds.

V. MAIN OBSERVATIONS

In this section we discuss the main observation as they pertain to our research questions.

A. Can supervised machine learning methods be used to automatically distinguish between Web attacks and vulnerability scans?

In order to answer this question we present Table II where the method that gives the best (worst) probability of detection is highlighted in *italic* (**bold**). It is clear that for three out of four datasets the learners have very high probability of detection and low probability of false alarm. More specifically, the ranges for the probability of detection, when all 43 features are used, for WebDBAdmin I, Web2.0 I, and Web2.0 II datasets are 82.76% – 96.55%, 96.92% – 98.63%, and 92.52% – 97.36%, respectively. Furthermore, the probability of false alarm in these three datasets in many cases is less than 1% and never goes above 3.69%.

As we can see in Table II, WebDBAdmin II dataset has significantly lower probability of detection (with a range of 41.67% – 75.00%) when compared to the other three datasets. Interestingly, the probability of false alarm is very low for this dataset as well (less than 1.75% for all learners). The lower probability of detection in the case of WebDBAdmin II dataset is not unexpected, having in mind that this dataset has the smallest percentage of attacks (only 6.65%). Even

TABLE II. SUMMARY OF CLASSIFICATION RESULTS FOR ALL DATA SETS

Dataset	Features used	Learner	Accuracy	Prob. of detection	Prob. of false alarm	Precision	Balance
WebDBAdmin I	All 43 features	SVM	97.66%	**82.76**%	0.00%	100.00%	87.81%
		J48	95.79%	**82.76**%	2.16%	85.71%	84.17%
		J48 pruned	96.26%	93.10%	3.24%	81.82%	86.25%
		PART	98.13%	*96.55%*	1.62%	90.32%	97.30%
		PART pruned	98.13%	*96.55%*	1.62%	90.32%	97.30%
	9, 1, 17, 2	SVM	95.79%	**68.97**%	0.00%	100.00%	78.06%
		J48	95.79%	86.21%	2.70%	83.33%	90.06%
		J48 pruned	96.73%	89.66%	2.16%	86.67%	92.53%
		PART	96.26%	86.21%	2.16%	86.21%	90.13%
		PART pruned	97.66%	*93.10%*	1.62%	90.00%	94.99%
WebDBAdmin II	All 43 features	SVM	95.81%	**41.67**%	0.39%	88.24%	57.92%
		J48	96.72%	*75.00%*	1.75%	75.00%	75.00%
		J48 pruned	97.09%	72.22%	1.17%	81.25%	76.30%
		PART	96.54%	63.89%	1.17%	79.31%	70.57%
		PART pruned	96.53%	63.88%	1.16%	79.31%	74.45%
	24, 28, 26, 17, 25, 2	SVM	94.54%	**25.00**%	0.58%	75.00%	46.97%
		J48	97.09%	*75.00%*	1.36%	79.41%	82.30%
		J48 pruned	97.27%	*75.00%*	1.17%	81.82%	82.23%
		PART	96.90%	69.44%	1.17%	80.65%	78.38%
		PART pruned	95.81%	69.44%	2.34%	67.57%	78.33%
Web 2.0 I	All 43 features	SVM	99.37%	*98.63%*	0.36%	98.97%	98.79%
		J48	99.55%	*98.63%*	0.12%	99.66%	99.00%
		J48 pruned	99.19%	97.61%	0.24%	99.31%	98.24%
		PART	98.93%	96.93%	0.36%	98.95%	97.71%
		PART pruned	98.93%	**96.92**%	0.36%	98.95%	97.81%
	10, 28, 26, 25	SVM	98.75%	**95.90**%	0.24%	99.29%	97.10%
		J48	99.19%	*97.61%*	0.24%	99.30%	98.30%
		J48 pruned	98.83%	96.24%	0.24%	99.30%	97.34%
		PART	99.19%	*97.61%*	0.24%	99.30%	98.30%
		PART pruned	98.83%	96.24%	0.24%	99.30%	97.34%
Web 2.0 II	All 43 features	SVM	94.19%	**92.52**%	3.59%	97.15%	94.34%
		J48	94.19%	**92.52**%	3.59%	97.15%	94.34%
		J48 pruned	96.80%	96.88%	3.30%	97.49%	97.17%
		PART	96.91%	*97.36%*	3.69%	97.22%	97.29%
		PART pruned	96.90%	97.35%	3.69%	97.22%	96.79%
	28, 25, 10, 26, 29, 2	SVM	91.58%	**85.66**%	5.80%	99.49%	89.85%
		J48	95.71%	95.26%	3.69%	97.16%	95.76%
		J48 pruned	95.63%	*95.56%*	4.27%	96.73%	95.64%
		PART	94.96%	94.79%	4.81%	96.31%	94.99%
		PART pruned	94.65%	94.83%	5.59%	95.74%	94.62%

more, 35 out of total 36 observed attacks were classified as 'Other attacks' (see Table I) because they differed among themselves. Therefore, whenever any of these attack sessions appeared in the validation (i.e., testing) data it did not have corresponding instances in the training data. In this context, we can say that the best performing learner J48 is able to detect 75% of the new, previously unseen attacks. Note that in case of WebDBAdmin I dataset, even though it has only 29 attacks (i.e., 13.55% of all sessions), because multiple attack sessions belong to same attack classes (see Table I) all learner performed significantly better than in case of WebDBAdmin II dataset.

Another important observation made based on the data presented in Table II is that accuracy on its own can be a misleading measure of learner's performance. Specifically, when all 43 features are used for learning the accuracy is very high (i.e., above 94%) for all datasets and all learners, even in cases when the probability of detection is low to moderate (i.e., at most 75%).

Overall, as indicated by the high probability of detection in three datasets and low probability of false alarm in all four datasets, we can conclude that supervised machine learning methods can be used to successfully distinguish between attack and vulnerability scan sessions.

B. Do attacks and vulnerability scans differ in a small number of features? Are the subsets of "best" features consistent across different data sets?

When distinguishing between attacks and vulnerability scans it is very important to choose the simplest possible model because it leads to better efficiency and performance of the machine learning algorithms. Typically, some features have more predicting power than others and by studying these "best" features in more details we can develop better understanding how attacks and vulnerability scans differ. The results of using the information gain as a feature selection

method and the procedure described in section IV[1] show that, depending on the dataset, from four to six features can be used to classify malicious activities without significantly worsening learners performance compared to when all 43 features are used. (In Table II the selected features are ordered from most to least informative.) These results indicate that attacks differ from vulnerability scans only in a small number of features (i.e., session characteristics). Specifically, out of 43 features only 10 unique features are selected across all four datasets (i.e., 1, 2, 9, 10, 17, 24, 25, 26, 28, and 29), which means that less than one quarter of the features are important for classifying malicious traffic.

When comparing these small subsets of best predicting features across datasets, we observe some consistency (e.g., Web 2.0 I, Web 2.0 II, and WebDBAdmin II), as well as some differences, especially for systems running different applications (e.g., WebDBAdmin I and Web 2.0 I). To further explore this issue, we considered the top ten features for each data set ordered from the most to least informative based on the information gain shown in Table III. We also provide several examples of PART rules in Table IV as an illustration.

Two features – (2) Bytes transferred and (10) Number of requests with POST method – appear among top ten features in all datasets. When considered on its own, there is no clear difference between the values of bytes transferred (i.e., feature (2)) in attack sessions and vulnerability scan sessions. We noticed, however, that in PART rules this feature appeared most often in combination with features related to the length of the request substrings (i.e., (25) - (29)) or in combination with number of requests (i.e., feature (1)). As expected, the number of requests with POST method (i.e., feature (10)) played important role in distinguishing attacks from vulnerability scans. Thus, while all vulnerability scans have zero requests with POST method, attacks have zero or more requests with POST method. For example, both password cracking attacks in WebDBAdmin I dataset and posting spam on wiki or blog in Web 2.0 I and Web 2.0 II datasets include at least one request with POST method.

Other features that appear to play significant role in classifying malicious sessions are related to the length of the request substrings within a session (i.e., features (25) through (29)). We looked carefully into the sessions and noticed that attacks tend to have longer request substrings than vulnerability scans. Features (30), (33), and (34) which deal with the number of parameters in requests within a session were selected in the top ten features only for the Web 2.0 I and Web 2.0 II datasets. This is because Web 2.0 honeypots contained interactive content (i.e., ran MediaWiki and Wordpress). By studying these three features we noticed that attacks tend to pass more parameters in a session than vulnerability scans.

The fact that there is some consistency, but also differences in the subsets of features whcih are best predictors is likely an intrinsic characteristic that will hold true for other datasets of malicious activities as well. We believe that this phenomenon is due to the fact that Web systems are reached by attackers mainly using a search-based strategy and thus systems with different configurations are exposed to different types of malicious activities, with different behaviors. Hence, instead of advocating a particular subset of features as the best predictors for all systems, our study suggests that classification of malicious activities should include feature selection method that helps identifying the best subset for a particular system.

[1]The difference between probability of detection when using all 43 features and reduced subset of features for WebDBAdmin I dataset cannot be made less that 1% because there are only 29 attacks out of 214 sessions. 3% is the smallest difference that could be achieved in this case.

TABLE III. TOP TEN FEATURES FOR EACH DATASET ORDERED FROM THE MOST TO LEAST INFORMATIVE

Dataset	(ID) Feature name
WebDB-Admin I	(9) Number of requests with GET method (1) Number of requests (17) Number of requests to static files (2) Bytes transferred (21) Number of requests with 'Success' status code (3) Session duration (in seconds) (8) Std deviation of time between requests (10) Number of requests with POST method (25) Mean length of request substrings (29) Std deviation of length of request substrings
WebDB-Admin II	(24) Number of requests with 'Server' errors (28) Max length of request substrings (26) Median length request substrings (17) Number of requests to static files (25) Mean length of request substrings (2) Bytes transferred (10) Number of requests with POST method (27) Min length of request substrings (21) Number of requests with 'Success' status code (29) Standard deviation of length of request substrings
Web 2.0 I	(10) Number of requests with POST method (28) Max length of request substrings (26) Median length of request substrings (25) Mean length of request substrings (27) Min length of request substrings (29) Standard deviation of length of request substrings (30) Mean number of parameters in requests (33) Max number of parameters in requests (2) Bytes transferred (34) Standard deviation of number of parameters in requests
Web 2.0 II	(28) Max length of request substring (25) Mean length of request substring (10) Number of requests with POST method (26) Median length of request substring (29) Standard deviation of length of request substring (2) Bytes transferred (30) Mean number of parameters in requests (27) Standard deviation of length of request substring (17) Number of requests to static files (34) Standard deviation of number of parameters in requests

C. Do some learners perform consistently better than other across multiple datasets?

In this subsection we discuss the performance of the machine learning methods used in this paper: SVM, J48, and PART. Figure 1 shows the ROC squares for all four dataset. In each of these ROC plots, the x-axis is the probability of false alarm and the y-axis is probability of detection. The data point (0,1) in the upper left corner corresponds to the optimal performance, i.e., high probability of detection with a low probability of false alarm. Based on the results presented in Table II and Figure 1 we conclude that the tree based learners

TABLE IV. EXAMPLE PART RULES FOR ATTACKS AND VULNERABILITY SCANS FOR MODELS WITH SELECTED FEATURES

PART rule	Type of malicious activity
Bytes transferred $>$ 0.00072 AND Max length of request substrings $>$ 0.216327 AND Max length of request substrings $<=$ 0.832653	Attack (Web 2.0 II)
Bytes transferred $<=$ 0.000515 AND Median length of request substrings $>$ 0.118367 AND Median length of request substrings $<=$ 0.146939 AND Max length of request substrings $<=$ 0.702041	Vulnerability scan (Web 2.0 II)
Number of requests $>$ 0.1875 AND Number of requests $>$ 0.208333 AND Bytes transferred $<=$ 0.018642	Attack (WebDBAdmin I)
POST $=$ 0 AND Median length of request substrings $<=$ 0.070857 AND Median length of request substrings $>$ 0.005714	Vulnerability scan (Web 2.0 I)

J48 and PART, both when used with all features and with selected features, outperform the SVM. Only in one out of eight cases (for Web 2.0 I dataset with all 43 features) SVM has the best probability of detection value which is equal to J48. However, even in that case J48 has slightly lower probability of false alarm, and thus slightly better balance, than SVM. Furthermore, SVM requires more features than the tree-based methods to achieve the desired less than 1% difference when selected features are used compared to using all 43 features. Specifically, instead of four, six, four, and six features for datasets WebDBAdmin I, WebDBAdmin II, Web 2.0 I and Web 2.0 II shown in Table II which were all selected based on one of the tree-based methods, SVM requires ten, ten, twelve, and sixteen features, respectively. Combining these observations with the fact that SVM takes the longest execution time of the three learners, clearly indicates that in case of our datasets tree-based learners are preferred for classification of malicious traffic.

It seems that there is no clear winner between J48 and PART (see Figure 1). Thus, out of the eight cases shown in Table II, J48 has the highest probability of detection in four cases, PART in three cases, and there is one tie. Regarding the pruned trees, it can be noted that they result in as good as, or better probability of detection than the unpruned trees, both for J48 and PART.

With respect to the tree size, the number of leaves in J48 trees is comparable to the number of rules in PART, for all datasets. Pruning the trees reduces these numbers approximately in half. For example, unpruned J48 tree has eight leaves, while pruned J48 tree has five leaves. Similarly, unpruned PART has eight rules and pruned PART has four rules. Interestingly, the size of the trees and the number of rules for the Web 2.0 II dataset is significantly bigger than those generated for the other datasets. For example, unpruned J48 tree for Web 2.0 II has 59 leaves and 117 nodes, while pruned J48 tree has 33 leaves and 65 nodes. This means that Web2.0 II dataset contains more similar instances of attacks and vulnerability scans, and therefore more complex rules are required to divide them into two classes.

VI. CONCLUDING REMARKS

In this paper we used supervised machine learning methods to classify malicious Web sessions. In particular, our goal was to distinguish between attack sessions and vulnerability scan sessions. The results presented in this paper are based on using multiple learners on four datasets, which to some extent allows generalization of the observations.

Our results show that supervised machine learning methods can distinguish successfully between Web attacks and vulnerability scans. It appears that attacks differ from vulnerability scans only in a small number of session characteristics, from four to six depending on the dataset. More importantly, predictions based on these best predictors have performance very close to predictions based on all 43 features. When exploring the consistency of the best predictive features across datasets, we noticed that some features appear among best predictors in multiple datasets. However, we also observed differences, especially for systems running different applications. We believe that this phenomenon is likely to hold for other datasets as well because attackers target Web systems using predominately a search-based strategy and thus systems with different configurations are exposed to different types of malicious activities, with different behaviors. Hence, instead of advocating a particular subset of features as the best predictors for all systems, we recommend using feature selection method to identify the best subset for a particular system.

Our second set of observations are related to the learners' performance. While there is no clear winner between decision tree algorithms J48 and PART, both of them performed better than SVM consistently across all four datasets. In addition, J48 and PART needed significantly less features than SVM to achieve probability of detection close to the case when all features were used. The pruned versions of J48 and PART were approximately half the size of the unpruned versions and yet produced close, and in some cases even better performance than the unpruned tree versions. Finally, for realistic assessment of learners' performance metrics such as probability of detection and probability of false alarm have to be used because accuracy values tend to be very high even in cases when the attack detection was low or moderate.

The results presented in this paper enrich the empirical evidence on malicious cyber activities and can support areas such as generation of attack signatures and developing models for attack injection that can be used for testing the resilience of services and systems.

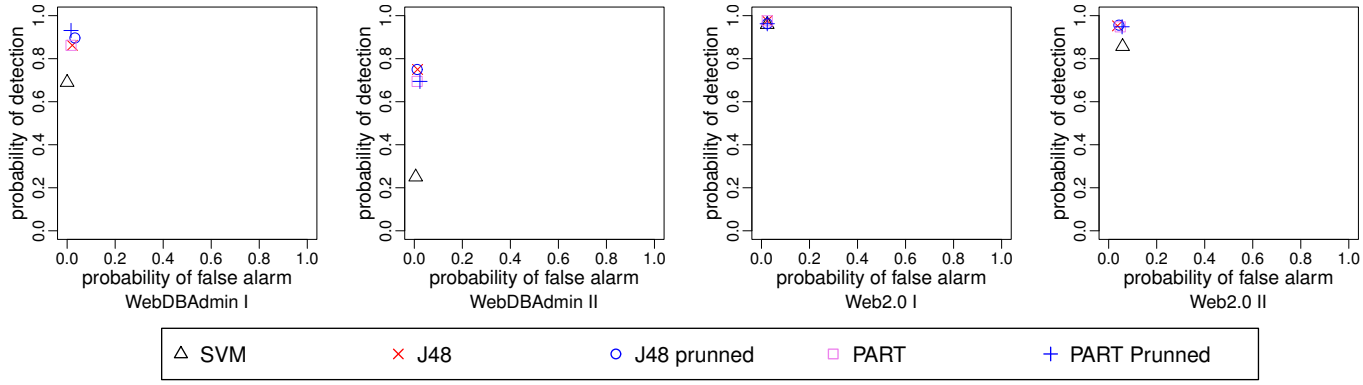

Fig. 1. ROC squares for learners applied on the selected features, for each data set

ACKNOWLEDGMENTS

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

REFERENCES

[1] E. Alata, V. Nicomette, M. Kaaniche, M. Dacier, and M. Herrb, "Lessons learned from the deployment of a high-interaction honeypot," *6th European Dependable Computering Conf.*, 2006, pp. 39-46.

[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] P. Barford, Y. Chen, A. Goyal, Z. Li, V. Paxon and V. Yegneswaran, "Employing honeynets for network situational awareness", *Cyber Situational Awareness*, in Advances in Information Security, S. Jajodia et al. (Editors), Springer, 2010, pp. 71-102.

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

[5] B. E. Boser, I. Guyon and V. Vapnik, "A training algorithm for optimal margin classifiers", *5th Annual Workshop on Computational Learning Theory*, 1992, pp. 144-152.

[6] Computer Emergency Response Team, http://www.cert.org/

[7] 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.

[8] DARPA Intrusion Detection Evaluation Project http://www.ll.mit.edu/mission/communications/ist/CST/index.html

[9] E. Frank and I. H. Witten, "Generating accurate rule sets without global optimization", *15th Int'l Conf. Machine Learning*, 1998, pp. 144-151.

[10] 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.

[11] 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.

[12] C.-W. Hsu, C.-C. Chang and C.-J. Lin, "A practical guide to support vector classification." *Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei*, 2003.

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

[14] KDD Cup 1999 data, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

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

[16] 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.

[17] National Vulnerability Database, http://nvd.nist.gov/

[18] S. Noel, D. Wijesekera and C. Youman, "Modern intrusion detection, data mining, and degress of attack guilt", *Applications of Data Mining in Computer Security*, in Advances in Information Security, D. Barbara and S. Jajodia (Editors), Kluwer Academic Publishers, 2002, pp. 1-31.

[19] Open Web Application Security Project (OWASP), http://www.owasp.org/

[20] R. Pantev, *Analysis and Classification of Current Trends in Malicious HTTP Traffic*, Master Thesis. West Virginia University, Morgantown, WV, 2011.

[21] 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.

[22] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, 1993.

[23] J.R. Quinlan, "Simplifying decision trees," *Intl J. Man-Machine Studies*, Vol.27, 1987, pp. 221-234.

[24] J. Riden, R. McGeehan, B. Engert and M. Mueter, "Using honeypots to learn about HTTP-based attacks", *Honeynet Project*, 2008, http://www.honeynet.org/papers/webapp/

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

[26] Security Focus, http://www.securityfocus.com/

[27] M. Vrable, J. Ma, J. Chen, D. Moore, E. Vandekieft, A. C. Snoeren, G. M. Voelker and S. Savage, "Scalability, fidelity, and containment in the Potemkin virtual honeyfarm", *SIGOPS Oper. Syst. Rev.* Vol. 39, No. 5, Oct. 2005, pp. 148-162.

[28] K. Xu, Z-L Zhang and S. Bhattacharyya, "Internet traffic behavior profiling for network security monitoring", *IEEE/ACM Trans. Networking*, Vol.16, No.6, Dec. 2008, pp. 1241-1251.