2012 IEEE 23rd International Symposium on Software Reliability Engineering

Using Multiclass Machine Learning Methods to Classify Malicious Behaviors Aimed at Web Systems

Katerina Goseva-Popstojanova and Goce Anastasovski
Lane Department of Computer Science and Electrical Engineering
West Virginia University, Morgantown, WV, USA
Katerina.Goseva@mail.wvu.edu; ganasta2@mix.wvu.edu

Risto Pantev*
Microsoft
Redmond, WA, USA
rpantev@mix.wvu.edu

***Abstract*—The number of vulnerabilities and attacks on Web systems show an increasing trend and tend to dominate on the Internet. Furthermore, due to their popularity and users ability to create content, Web 2.0 applications have become particularly attractive targets. These trends clearly illustrate the need for better understanding of malicious cyber activities based on both qualitative and quantitative analysis. This paper is focused on multiclass classification of malicious Web activities using three supervised machine learning methods: J48, PART, and Support Vector Machines (SVM). The empirical analysis is based on data collected in duration of nine months by a high-interaction honeypot consisting of a three-tier Web system, which included Web 2.0 applications (i.e., a blog and wiki). Our results show that supervised learning methods can be used to efficiently distinguish among multiple vulnerability scan and attack classes, with high recall and precision values for all but several very small classes. For our dataset, decision tree based methods J48 and PART perform slightly better than SVM in terms of overall accuracy and weighted recall. Additionally, J48 and PART require less than half of the features (i.e., session attributes) used by SVM, as well as they execute much faster. Therefore, they seem to be clear methods of choice.**

***Keywords*-Web 2.0 security; empirical study; vulnerability scans; attacks; multiclass classification**

I. Introduction

The widespread use of Web technologies have entered almost every aspect of our lives, from business and education to shopping, social life and entertainment. 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 [29]. In the last several years Web 2.0 technologies have enhanced content creating and sharing, collaboration, and the overall functionality of the Web. However, due to users ability to dynamically interact with the applications and 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.

Until recently not much research work was focused on characterization and quantification of malicious attacker behaviors. One obvious reason for this is the lack of publicly available, good quality, recent data on cyber security threats and malicious attacker activities. Furthermore, the significant amount of existing work on intrusion detection was mostly focused on data mining techniques aimed at constructing a "black-box" that classifies the network traffic on malicious and non-malicious, without addressing the discovery of the nature of malicious activities [13]. In addition, intrusion detection research work was, to a large extent, based on using outdated datasets 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, we developed and deployed a high-interaction honeypot as a means to collect such data [12]. The data used in this paper consists of malicious Web sessions extracted from application level logs of a three-tier Web system running on the Internet. Therefore, our dataset represents dynamic information on attacker activities, unlike data extracted from vulnerability databases, such as [6], [19], [30], that are focused on static information related to description of known vulnerabilities and the ways they may be exploited. Part of the dataset considered in this paper was used in our previous work [12] which was focused on descriptive analysis of different types of vulnerability scans and attacks, and also included inferential statistical analysis of malicious HTTP traffic. After the publication of [12] we collected data during additional five months, which led to a dataset in total duration of nine months. It should be noted that [12] did not include classification of malicious traffic using machine learning methods.

In this paper we use several multiclass supervised machine learning methods with a goal to automatically classify malicious Web sessions to multiple vulnerability scan classes and attack classes. 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 a *vulnerability scan*. Out of the 5,902 malicious Web sessions, around 49% belong to vulnerability scan classes and 51% belong to attack classes. The vulnerability scan and attack sessions are further divided

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

1071-9458/12 $26.00 © 2012 IEEE
DOI 10.1109/ISSRE.2012.30

into eleven and nine unique classes, respectively.

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. The fact that our dataset consists 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.

Multiclass classification is typically harder problem than two class classification. As the number of distinct classes increases, so does the difficulty of the classification, as well as the size of the training dataset needed. Typically, in any multiclass classification task, some classes are more difficult to classify than others.

In the dataset considered in this paper, the distribution of the number of malicious sessions across the twenty classes is rather imbalanced, with only five large classes. The remaining fifteen classes together have slightly over 9% of all malicious sessions, most of them contributing less than 1% of malicious sessions. This heavily imbalanced distribution of malicious sessions over different vulnerability scan and attack classes, which we believe is an intrinsic characteristic of attacker activities observed at servers on the Internet, poses a significant challenge to machine learning methods used for classification.

We explore this challenge by addressing the following specific research questions:

A) Can supervised machine learning methods be used to automatically classify malicious activities? Are some classes harder to predict than others?

B) Can malicious cyber activities be distinguished using a small number of features (i.e., session attributes)? Do some learners perform better than others?

The main observations are as follows:

- Our results show that supervised machine learning methods can be used to separate malicious traffic to multiple classes with high recall and precision for all but several very small classes.
- Decision tree based algorithms J48 and PART have several advantages over SVM (1) they have slightly better overall accuracy and weighted recall, (2) need less than half of the features to successfully classify the malicious classes, (3) execute much faster, and (4) provide better interpretability of the results which helps understanding different classes of malicious behaviors.

With the increase in the number and diversity of malicious behaviors on the 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 machine learning methods for intrusion detection, that is, for classification of network traffic to two classes: malicious and non-malicious. Examples of these methods can be found in [13], [20].

Only a few papers used multiclass machine learning methods to distinguish not only between non-malicious and malicious traffic, but also among different classes of malicious traffic. As most of the intrusion detection work, these papers used the outdated DARPA [8] and KDD 1999 datasets [14], which in addition to the normal traffic include four main classes of simulated attacks. Next, we summarize these papers. The work presented in [16] used a combination of several methods to distinguish between the normal class and four malicious classes of the DARPA dataset [8]. In [18] the authors compared three variants of the multiclass SVM algorithms (i.e., one-versus-one, one-versus-all, and decision tree based multiclass SVM) to the two-class SVM. The intrusion detection system proposed in [34] combined the principal component analysis (PCA) for feature reduction, multiclass SVM for classification in network based IDSs, and Markov models for detecting anomalous behavior in host based IDSs. Another work presented in [15] tried to speed up the training of SVM and improve its accuracy by applying reduction techniques that use clustering analysis to approximate support vectors.

It appears that not many datasets other than the KDD 1999 have been used in a multiclass setting. In [5] two classes of normal and six classes of bot traffic from the query stream of a large search engine provider were investigated. The authors used PCA to reduce the dimensionality of the data and compared the accuracy of seven different machine learning algorithms. Data mining and entropy-based techniques were used in [33] to cluster the Internet backbone traffic into three classes: typical server/service behavior, typical heavy-hitter host behavior, and typical scan/exploit behavior. A recent paper [9] was focused on improving the performance of Web vulnerability scanners in detecting existing vulnerabilities. In that work, the requests to the Web server were generated by the researchers (similarly as in penetration testing) to produce both rejection and execution pages, and then clustering was used to group together similar response pages with a goal to detect existing vulnerabilities in Web applications.

Classification of some aspects of malicious traffic, in an absence of normal traffic, is an emerging research direction. 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. Two recent papers

[2], [4] 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. Another recent work presented in [23] was focused on finding similarities among different samples of malicious HTTP traffic. [2], [4], [23] applied single-linkage hierarchial clustering to group the malicious samples in classes with similar behaviors. In a similar work [26] malware binaries were executed in a sandbox environment and SVM was used to identify the shared behavior of fourteen malware families.

Our work differs from the related work in several ways. First, none of the related work was based on data collected by advertised, fully functional, three-tier honeypot system, with standard security settings. Although based on honeypots, our data collection approach and goals are complementary to other existing approaches based on honeypots. These include honeypots deployed for the purpose of being compromised (typically throughout SSH authentication) in order to analyze the behavior of the adversaries following SSH compromises [1], [25], [27], [28]. Another complementary approach to ours is to use passive monitoring of the unused address space or active responders with a goal to cover a large range of IP addresses in order to collect information on malicious activities such as worm outbreaks and botnet sweeps [3], [31]. Our goal is to study attacks that spread along application-specific topologies (i.e., Web) which typically carefully select their victims and therefore are unlikely to be observed in the unused address space or a honeyfarm of active responders [31].

Second, while we use machine learning techniques as several recent papers which dealt with studying malicious behaviors, we classify malicious activities with a goal to distinguish among twenty malicious classes (eleven vulnerability scan classes and nine attack classes). Other works distinguished (1) among one normal and four malicious classes [15], [18], [34], (2) among two normal and six bot classes [5], (3) among three types of attacks on port 445 [7] or (4) grouped malware programs with similar behaviors [2], [4], [23], [26].

Third, we identify the best subset consisting of a small number of features that are most useful for classification of malicious activities, thus identifying the simplest, most efficient model for our dataset. Some of the 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 [23]). Other related work papers used feature ranking [26], dimensionality reduction based on PCA [5], [34] or reduction of support vectors in SVM [18], but not feature selection methods.

Finally, unlike most of the related work which used only one learner, we use multiple learners and compare their performance and effectiveness using several metrics.

III. Data Collection and Extraction

Facing the lack of publicly available, recent data on malicious attacker activities, we developed and deployed a high-interaction honeypot as a means to collect such data [12]. The honeypot ran off-the shelf operating system and applications, which 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. In particular, the honeypot ran the Windows XP Service Pack 2 operating system, with Microsoft IIS 5.1 Web server, and PHP 5.0.2 server. It also included two Web 2.0 applications: 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). Both Web 2.0 applications used the MySQL database (version 4.1) as a back-end tier. From the honeypot with this configuration, we collected data in a duration of nine months (i.e., 273 days).

The honeypot was 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 honeypots that ran Web systems is of crucial importance for collecting realistic data because it allowed us to observe typical malicious activities aimed at these systems, including attacks based on search engines.

Our data is organized in Web sessions, each defined as a sequence of requests from the same source IP address to port 80, with a time between two successive requests not exceeding a threshold of thirty minutes [12]. The malicious sessions were extracted automatically from the logs of the front-end Web server (i.e., IIS). Since the honeypot 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.

In order to be able to evaluate the supervised machine learning techniques the malicious Web sessions have to be labeled (i.e., assigned to classes), which is then used as a ground truth. 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) [21] and searching public databases such as [19] and [30]. (Further details on Web sessions labeling can be found in our previous work [12].) The breakdown of malicious Web sessions to different vulnerability scan and attack classes is shown in Table I. Overall, out of twenty classes only five have high frequency. The remaining fifteen classes account for a little over 9% collectively, with most of them contributing individually less than 1%.

It should be noted that the configuration of our honeypot and the malicious attacker activities collected by it are representative of Web systems that run Web 2.0 applications such as wiki and blog. Honeypots with different configurations will, at least partially, experience other types of vulnerability scan and attack classes.

In this paper each Web session is characterized with a vector of 43 different features (i.e., session characteristics). These 43 features extend the feature sets used in [7] and [23] (consisting of four and seven features, respectively) by considering features similar to those used in articles on network intrusion detection, host intrusion detection, and Web crawlers identification. (Further details and references are given in [22].)

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.

Table I. BREAKDOWN OF MALICIOUS WEB SESSIONS

Malicious activity	sessions	%
Vulnerability scans: Total	**2,883**	**48.85%**
DFind	44	0.75%
Other fingerprinting	2	0.03%
Static	508	8.61%
Blog	797	13.50%
Wiki	1,307	22.15%
Blog & Wiki	150	2.54%
Static & Blog	11	0.19%
Static & Wiki	22	0.37%
Static & Blog & Wiki	28	0.47%
phpMyAdmin	11	0.19%
Static & phpMyAdmin	3	0.05%
Attacks: Total	**3,019**	**51.15%**
Denial of Service (DoS)	4	0.07%
Password cracking Blog user accounts	10	0.17%
Password cracking Wiki user accounts	71	1.20%
Spam on Blog	1,434	24.30%
Spam on Wiki	1,304	22.09%
Remote File Inclusion (RFI)	9	0.15%
SQL injection	2	0.03%
Cross-site Scripting (XSS)	13	0.22%
Other Attacks	172	2.91%
Total	**5,902**	**100%**

IV. DATA MINING APPROACH

To classify the observed malicious traffic to eleven vulnerability scans classes and nine attack classes we use three different supervised machine learning methods: decision tree J48 which is a Java implementation of the C4.5 decision tree algorithm [24], a rule induction method based on partial decision trees (PART) [10] and multiclass one-vs-all SVM [32]. For pruning both J48 and PART we used the Reduced Error Pruning (REP). For SVM we used the Radial Basis Function (RBF) as a kernel function.

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 distinguish among classes of malicious sessions. In particular, 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 [17]. Our goal is to find a subset of features that works well for all classes, which is commonly referred to as simultaneous multiclass feature selection.

A. Classification process

Preprocessing. 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.

Model tuning and feature selection. We select one seventh of the dataset (i.e., 844 out of 5,902 sessions) using random stratified sampling and use is to tune the parameters of the three machine learning algorithms. We also run the feature selection method on this tuning subset, which avoids

overfitting the model. The feature selection is done based on stratified 10 cross validation within the tuning subset and produces a list of features ranked by information gain from highest to lowest. Since our goal is to identify the smallest number of features sufficient to accurately distinguish among different malicious classes, we use the following procedure for each learner. We start with the highest ranked feature and include one feature at a time until reaching less than or equal to 2% difference of the overall accuracy compared to the case when all 43 features are used.

Training the classifiers and making predictions. The remaining six sevenths of the data (i.e., 5,058 malicious Web sessions) are used to train and test the tuned models using stratified 10 cross validation. The results presented in section V are the averages over the ten folds.

B. Evaluating learners' effectiveness

To evaluate the performance of each learner, we compute the confusion matrix given with equation (1). For each class i out of total m classes, the diagonal element E_{ii} is equal to the number of sessions that actually belong to i-th class and are correctly assigned to this class, which is referred to as true positives TP_i. Each off-diagonal element $E_{ij}, i \neq j$ gives the number of sessions that actually belong to the i-th class that have been incorrectly assigned to class j.

$$\begin{bmatrix} E_{11} & E_{12} & \dots & E_{1j} & \dots & E_{1m} \\ E_{21} & E_{22} & \dots & E_{2j} & \dots & E_{2m} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ E_{i1} & E_{i2} & \dots & E_{ij} & \dots & E_{im} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ E_{m1} & E_{m2} & \dots & E_{mj} & \dots & E_{mm} \end{bmatrix} \quad (1)$$

Then, we use the confusion matrix to compute metrics for each class individually and for all classes together.

Per class metrics. For each individual class i, let TP_i denote true positives (the number of the sessions that in fact belong to i-the class and are correctly assigned to this class); FP_i denote false positives (the number of the sessions that in fact do not belong to i-th class, but are falsely assigned to this class); FN_i denote false negatives (the number of the sessions that in fact belong to i-th class, but are falsely not assigned to this class); and TN_i denote true negatives (the number of the sessions that in fact do not belong to i-th class and are correctly not assigned to this class).

For each class i we compute the recall (R_i) and precision (P_i) as follows:

$$R_i = \text{TP}_i/(\text{TP}_i + \text{FN}_i) \quad (2)$$

$$P_i = \text{TP}_i/(\text{TP}_i + \text{FP}_i) \quad (3)$$

Recall (R_i) defined by (2), which sometimes is also called *probability of detection*, accounts for the probability of correctly classifying the sessions of each class (i.e., the ratio of detected sessions of class i to all sessions in class i). *Precision* (P_i), defined by (3), determines the fraction of sessions correctly classified to belong to class i out of all sessions classified as class i. The *F-score* (F_i) of each class i is then computed as a harmonic mean of the R_i and P_i as:

$$\text{F}_i = 2R_iP_i/(R_i + P_i) \quad (4)$$

R_i, P_i, and F_i values are in the interval $[0, 1]$; larger values correspond to better classification. Ideally, we want both recall and precision to be 1, which leads to F-score equal to 1. If either one is 0, F-score is 0 by definition.

Metrics for all classes. To evaluate the learners' performance for all m classes together, that is, for the entire classification problem, we use the *overall accuracy* given by

$$\text{accuracy} = \text{TP}/(\text{TP}+\text{FN}) = \sum_{i=1}^{m} \text{TP}_i / \sum_{i=1}^{m} (\text{TP}_i + \text{FN}_i) \quad (5)$$

The overall accuracy gives equal weight to each malicious session, regardless of the class distribution, and therefore it tends to be dominated by the classifier's performance on common (i.e., high frequency) classes.

In order to account for the classes and the distribution of malicious sessions among classes, we introduce a weighted averaging approach, which takes into account the frequency of each class in computing the average metrics for all classes.[1] Let m_i denote the number of instances in the i-th class. Then the weight of the i-th class is the fraction of sessions that belong to class i and it is given by $w_i = m_i / \sum_{i=1}^{m} m_i$. The weighted-recall is then defined as

$$\text{weighted-}R = \sum_{i=1}^{m} (w_i \text{R}_i) \quad (6)$$

The weighted precision is defined similarly and the weighted-F score is computed as a harmonic mean of the weighted-R and weighted-P.

V. Main Observations

In this section we discuss the main observation as they pertain to our research questions. The results are based on the random stratified sample of 5,058 malicious Web sessions, which do not include the one sevenths of the instances used for tuning the models.

A. Can supervised machine learning methods be used to automatically classify malicious activities? Are some classes harder to predict than others?

We start the evaluation with exploring the metrics for individual classes. For this purpose in Figure 1 we present the performance of J48 in terms of per class recall R_i,

[1]The weighted recall and precision reduce to arithmetic average metrics if all classes are given the same weight. Arithmetic averages typically are not used in multiclass classification because they are heavily influenced by the low frequency classes, and thus provide very pessimistic estimates.

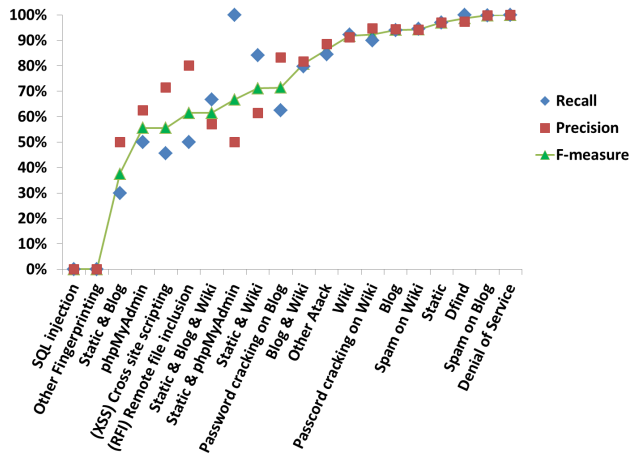


Figure 1. Per class recall (R_i), precision (P_i), and F-Score (F_i) for J48

precision P_i, and F-score F_i, with classes ordered on X-axes from the lowest to the highest values of their F-scores. From Figure 1 it is clear that most of the malicious classes are classified well. Based on the F-score values we identify four groups of classes. The first group, which consists of ten classes that make up 98.12% of all malicious sessions, have F-score above 80%. Out of these ten classes eight have F-score above 90%. The five largest classes (i.e., vulnerability scans classes *Static*, *Blog*, *Wiki* and attack classes *Spam on Blog* and *Spam on Wiki*) all belong to this group of best classified classes. Interestingly, three very small classes – *DoS*, *Dfind*, and *Password cracking on Wiki*, with 0.08%, 0.73% and 1.19% of malicious sessions respectively, are in the best classified group with all three metrics above 90%. Two other small classes (i.e., *Blog & Wiki* and *Other Attacks* also belong to the first group, each with less than 3% of malicious sessions and F-score above 80%.

The second group consists of five classes, which together account for only 1.21% of the total malicious sessions and have F-scores in the range 60–80%. These five classes are classified well, even though each of them has below 0.5% of sessions.

The third group consists of only two classes with F-scores in the range of 50–60%. The fairly low recall, precision and F-score are expected for classes like these which have only 0.20% and 0.22% of malicious sessions, respectively.

The last, fourth group of classes consists of only three classes which have F-scores below 50% and together account for only 0.26% of the malicious sessions. Two of these three classes, *SQL injection* and *Other Fingerprinting*, are not classified correctly at all, which is not surprising having in mind that they consists of only one or two sessions.

It follows that, J48 can classify malicious Web sessions successfully, with fifteen out of twenty classes having F-score over 60%, eight of which have F-score over 90%. Our results also show that some classes are harder to classify than others. For instance, *SQL Injection* attacks, and *Other fingerprinting* and *Static & Blog* vulnerability scans are not classified well. Conversely, other comparably small classes are much better classified, such as for example *Password cracking to Blog accounts* with F-scores around 70% or *DoS* with a perfect F-score of 100%.

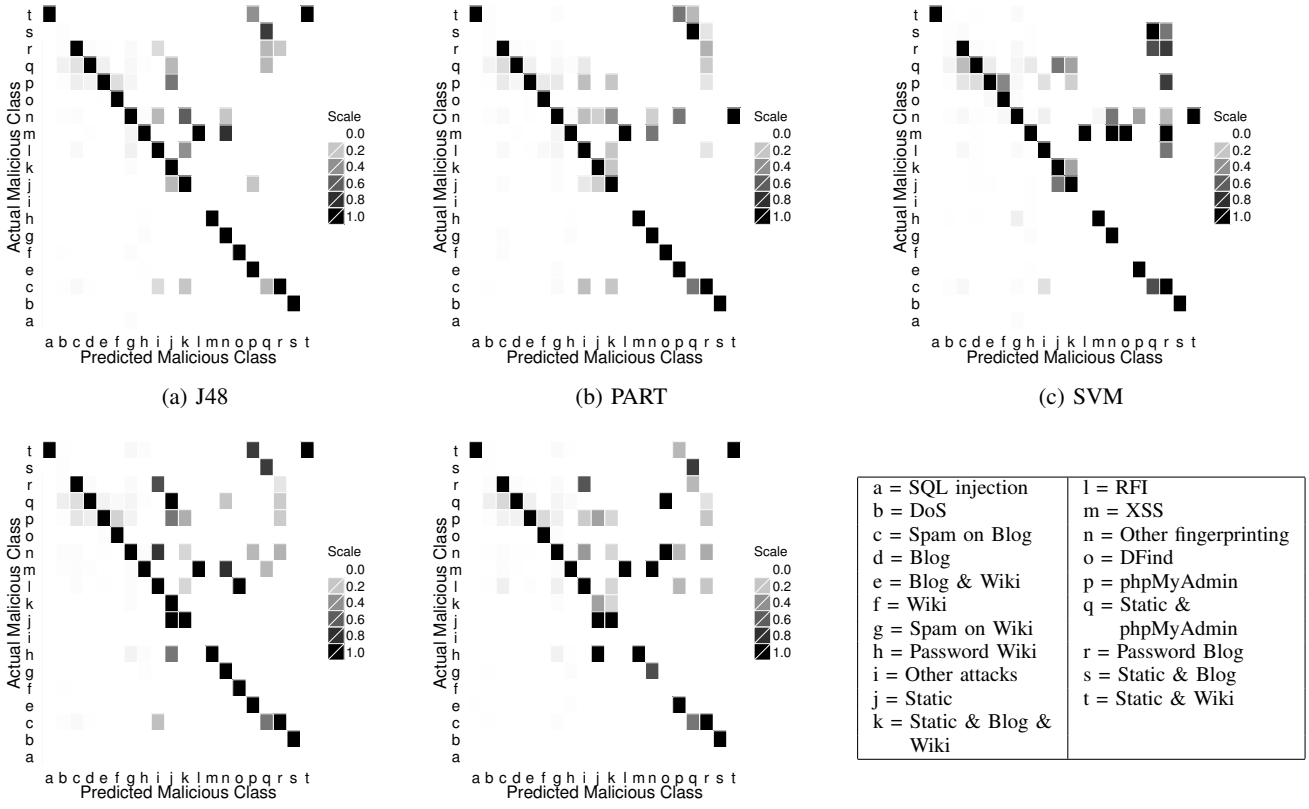
In a multiclass setting there are two reasons why some classes receive substantially lower recall and/or precision than others. The first reasons for poor classification is due to the fact that for some classes there are very few positive training examples. Although in some domains this problem may be resolved by obtaining more training cases for that class, we believe that may not be always possible in case of malicious activities. For instance, even though we collected data over nine months period, we observed only a few *SQL Injection* and *Other Fingerprinting* sessions out of total 5,902 malicious Web sessions.

The second reason why some small classes are classified worse than others is that they may have less predictive features than other classes. For example, in the case of J48, the *RFI* class with eight sessions has recall of only 50% and F-score of 61.50%. On the other side, the *DoS* class, which consists of only four sessions, is detected with 100% recall and precision due to the fact that it has distinctive features.

In order to address the class imbalance problem, we used the AdaBoost algorithm [11], which is a meta-algorithm used along with other learning algorithms. In particular, AdaBoost runs on the dataset several times and in each run it updates the distribution of weights for all classes. In every round the weights of poorly classified classes are increased and the weights of well classified classes are decreased, thus the classifier in the next run focuses on the classes which have so far eluded correct classification. Therefore, we can say that this algorithm is adaptive in the sense that subsequently built classifiers are tweaked in favor of those instances misclassified by previous classifiers. Somewhat surprisingly, AdaBoost did not improve learners' performance on the small misclassified classes for our dataset. We suspect that this result is due to the fact that our dataset is highly imbalanced, with the smallest classes consisting only of handful of instances.

The other learners considered in this paper have per class performance similar to J48, as it can be observed from Figures 2 (a) – (c), which show the confusion matrices of malicious classes for J48, PART, and SVM, respectively. (The density of each cell, represented by shades of grey, gives the percentage of a true malicious class assigned to a predicted class by the classifier. The main diagonal corresponds to correct classification assignments. For a good classifier, the cells on the main diagonal are darkest among all cells.) Detailed discussions for PART and SVM are omitted due to space limitations.

In summary, we conclude that supervised machine learning methods successfully classify malicious Web activities, both in terms of classes and total number of instances. Thus, in the case of J48, only three very small classes which



(a) J48 (b) PART (c) SVM

(d) J48 pruned (e) PART pruned

Figure 2. Confusion matrices for each learner trained on all 43 features

together account for only 0.26% of all sessions, are not classified well (i.e., have recall, precision, and F-score less than 50%). Twelve out of twenty classes, which contain close to 99% of all instances, have F-scores above 70%. The top eight best classified classes have recall, precision, and F-score above 90%. These eight classes contain more than 92% of all instances, and in addition to the five largest classes include some very small classes.

B. Can malicious cyber activities be distinguished using a small number of features (i.e., session attributes)? Do some learners perform better than others?

When one tries to distinguish among various classes of malicious sessions 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 of malicious activities.

As discussed in Sections IV, for each learner we use information gain to select the smallest subset of features that works well for all classes (i.e., we perform simultaneous multiclass feature selection), without significantly worsening the learner's performance. Table II shows the top ranked features ordered from the most to least informative. To achieve overall accuracy within 2% of the accuracy when all 43 features are used J48, J48 pruned, PART, and PART pruned require the top seven features, while SVM requires the top seventeen features. Obviously, multiclass machine learning methods can separate malicious Web sessions successfully using small number of features, from 16% of all features in the best case (J48 and PART) to around 40% in the worst case (SVM). Before continuing the comparison of the learners, we discuss the top ranked features which have the highest predictive power for the classification of malicious Web traffic.

The features that appear to play the most significant role in classifying malicious sessions in our dataset 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 attack classes tend to have longer request substrings than vulnerability scan classes. Furthermore, out of all attack sessions, the sessions that posted spam on the Wiki and Blog had the longest length of the request substring. Vulnerability scan sessions on the Wiki and the Blog also had among the longest length of request substring.

Table II. TOP FEATURES BASED ON INFORMATION GAIN

Rank	(ID) Feature name
1	(28) Max length of request substrings
2	(26) Median length of request substrings
3	(25) Mean length of request substrings
4	(27) Min length of request substrings
5	(30) Mean number of parameters
6	(2) Bytes transferred
7	(29) Std deviation of length of request substrings
8	(33) Max number of paramters
9	(10) Number of requests with POST method
10	(18) Number of requests to dynamic application files
11	(31) Median number of parameters
12	(34) Std deviation of number of parameters
13	(32) Min number of parameters
14	(21) Number of requests with Success status code
15	(3) Duration
16	(4) Average time between requests
17	(9) Number of requests with GET method

Features (30) – (34), which deal with the number of parameters passed in requests substring within a session, are also among the top seventeen features based on the information gain. This is due to the fact that our honeypots contained interactive content (i.e., ran MediaWiki and Wordpress). In particular, exploring the values of these features in the raw data showed that attacks which posted spam to the Wiki and Blog passed more parameters than any other attack class. Overall, attack sessions typically passed more parameters than vulnerability scan sessions.

Among the top seven features, which are included in the selected feature subset for all learners, is the feature (2) Bytes transferred, which does not appear to show a particular trend across different classes when considered on its own.

As expected, the number of requests with POST method (i.e., feature (10)) also appears among the top ranked features. A close inspection of the data showed that all vulnerability scan classes have zero requests with POST method, while attack classes have zero or more requests with POST method. For example, attacks such as password cracking or posting spam on Wiki or Blog include at least one request with POST method.

Following down the list of the top ranked features is the feature (18) Number of requests to dynamic application files (e.g. .php and .asp), which is expected having in mind that our servers ran dynamic content. Thus, we noticed that both vulnerability scan and attack classes related to spam have higher number of requests to the dynamic application files than any other class. This is because the malicious activities related to spam only work when there is an interactive component present in the attacked system. Feature (21) Number of requests with Success status code is also included among the top features. In our dataset, attacks tend to have greater number of requests with Success status code than vulnerability scans, which is due to two reasons. First, as discussed in our previous work [12], most of the attacks on Web systems are search-based (i.e., use search engines or malicious crawlers to locate the application before launching the attack), that is, attempt to exploit an existing application

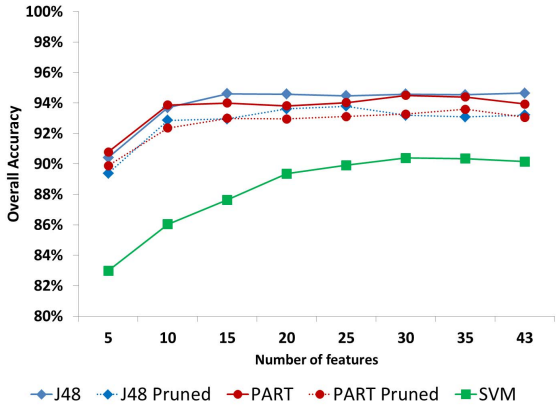


Figure 3. Performance of all learners by varying the number of features

on our system, which results in Success status code of the requests. On the other side, vulnerability scans often check for particular file and/or application that are not present on the system, which results in an error status code (e.g., 401).

The last three remaining features among the top seventeen features are (3) Duration, (4) Average time between requests and (9) Number of requests with GET method. Exploring the raw data led to the following interesting observations which apply to some instances of particular classes. The longest sessions in duration belong to the Spam on wiki class. However, there are also other much shorter Spam on wiki sessions. Sessions with the shortest average time between requests belong to different types of vulnerability scan sessions. Around 95% of all sessions have at least one requests (and at most 491 requests) with GET method, which is used for retrieving data.

We now return to the comparison of the learners performance. Figure 3 shows how the overall accuracy is affected by changing the number of features from five to 43, in increments of five, for all the learners. Note that the features are included using the ranked list by the information gain. By examining Figure 3, it can be noticed that J48 is slightly better than PART, with no more than 1% difference in the overall accuracy. The pruned J48 and PART (shown with dotted lines in Figure 3) do worse than unpruned J48 and PART by only 0.69% to 1.64%, depending on the number of features used. The same observation can be made comparing Figures 2 (a) and (d) for J48 unpruned and J48 pruned, and Figures 2 (b) and (e) for PART unpruned and PART pruned. Obviously, the pruned versions of the learners have slightly darker shades in some off-diagonal cells of the confusion matrix when compared to the unpruned versions.

Furthermore, both pruned and unpruned versions of J48 and PART outperform SVM, with a difference that decreases with increasing the number of selected features. We already mentioned that SVM needs seventeen features, i.e., more than twice as many features as J48 and PART to achieve accuracy within 2% of the case when all features are used. If we use the top seven features the overall accuracy of SVM

is around 84%, much lower than the accuracy of J48 and PART with the same number of features, which is in the range of 91% to 93%.

Table III summarizes the overall accuracy given by equation (5) and the time it takes to build the models, for all learners, first with all 43 features and then with the selected features. We also computed the values of the weighted recall using equation (6), which differed less than 1% from the corresponding values of the overall accuracy, and therefore are not reported in Table III. This phenomenon is due to the fact that classes with low recall are extremely small (i.e., less than 1%) and thus have very small effect on the weighted recall.

From Table III we observe that the pruned versions of J48 and PART produce the classification results much faster than the unpruned versions, both when all 43 features are used and when the selected seven features are used. This is an expected result because the pruning reduces the number of J48 leaves and PART rules significantly. For example, when all 43 features are used the unpruned J48 tree has 150 leaves, while the pruned J48 tree has 94 leaves. Similarly, unpruned PART has 152 rules and pruned PART has 67 rules. J48 and PART (both unpruned and pruned versions) require significantly less processing time than SVM. When feature selection is employed, the time taken to build the tree based learners decreases significantly. However, that is not the case with SVM, which required more processing time with the selected seventeen features than with all 43 features.

We conclude this section by providing several examples of PART rules in Table IV. These rules illustrate the usefulness of the results for intrusion protection and detection systems, such as firewalls and intrusion detection tools.

For example, out of total 1229 Spam on blog sessions 1218 were successfully classified using the rule in Table IV which includes at least one request with POST method, the maximum number of parameters passed in a request is at most one, and the median length of request substring is between 18 and 27. Note that the examples in Table IV, in addition to large classes such as *Spam on blog* and *Spam on wiki*, include classes such as *DFind* and *DoS*, which in spite of the small size are among the classes with the highest recall due to the fact that they have distinctive features.

VI. Concluding Remarks

In this paper we addressed the problem of distinguishing among multiple classes of malicious behaviors aimed at Web systems. The dataset consisting of over 5,900 malicious sessions was collected by a high-interaction honeypot which ran a fully functional three-tier Web system (including Web 2.0 applications such as blog and wiki) in duration of nine months. The configuration was based on off-the-shelf operating system and applications, with standard security settings and strong passwords.

Table III. Comparison of model performance

	Feature rank	**Learner**	**Overall accuracy**	**Time to build the model**
All features	1 - 43	J48	94.64%	1.58 sec
	1 - 43	J48 Pruned	93.21%	1.07 sec
	1 - 43	PART	93.93%	4.12 sec
	1 - 43	PART Pruned	93.06%	2.99 sec
	1 - 43	SVM	90.15%	10.78 sec
Selected features	1 - 7	J48	92.30%	0.35 sec
	1 - 7	J48 Pruned	91.63%	0.12 sec
	1 - 7	PART	92.36%	0.95 sec
	1 - 7	PART Pruned	91.38%	0.77 sec
	1 - 17	SVM	88.73%	16.34 sec

Our results show that, even in the presence of extreme skewness, the multiclass supervised machine learning methods can distinguish successfully among twenty malicious behavior classes (i.e., eleven vulnerability scan classes and nine attack classes), with high recall and precision for all but three small classes, together contributing to only 0.26% of the data. It appears that different classes of malicious behavior differ in small number of features (i.e., session characteristics). Specifically, using only seven out of 43 featured, J48 and PART successfully distinguished among malicious activities without significant loss of accuracy compared to when all features were used.

Our results further show that tree based algorithms J48 and PART perform slightly better than SVM with respect to the overall accuracy and weighted recall. Even more, J48 and PART require less than half of the features used by SVM to perform the classification task close to the case when all features were used. Considering that SVM executes much slower than J48 and PART, and provides almost no insight into characteristics of malicious activities, J48 and PART appear to be clear methods of choice for multiclass classification of malicious activities.

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.

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.

Table IV. EXAMPLES OF PART RULES BASED ON USING ALL FEATURES

PART rule	Class	Class size
Number of requests with Success status code = 0 AND Median length of requests substring <= 64 AND Bytes transferred > 2138 AND Min number of parameters passed = 0 AND Number of requests with POST method = 0 AND Min length of requests substring > 26 AND Min length of requests substring <= 32	DFind	37
Number of requests to dynamic application files = 0 AND Max length of requests substring <= 33 AND Number of requests to text files <= 8	Static	436
Number of requests with Success status code > 0 AND Number of requests to static application files = 0 AND Max number of parameters passed = 1 AND Number of requests with OPTIONS method > 0	DoS	4
Number of requests with POST method > 0 AND Max number of parameters passed <= 1 AND Median length of requests substring > 18 AND Median length of requests substring <= 27	Spam on blog	1,229
Median number of parameters passed > 1 AND Number of requests with POST method > 0 AND Standard deviation of number of parameters passed <= 3.07 AND Standard deviation of time between requests <= 396	Spam on wiki	1,118

[3] P. Barford, Y. Chen, A. Goyal, Z. Li, V. Paxon and V. Yegneswaran, "Employing honeynets for network situational awareness", Cyber Situational Awareness, 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] G. Buehrer, J. W. Stokes, K. Chellapilla and J. C. Platt, "Classification of automated Web traffic", in Weaving Services and People on the World Wide Web, I. King and R. Baeza-Yates (Editors), Springer, 2009, pp. 3-26.

[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, 2006, pp. 383-392.

[8] http://www.ll.mit.edu/mission/communications/ist/CST/index.html

[9] A. Dessiatnikoff, R. Akrout, E. Alata, M. Kaaniche and V. Nicomette, "A clustering approach for Web vulnerabilities detection", 17th Pacific Rim Int'l Symp. Dependable Computing, 2011, pp. 194-203.

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

[11] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm", 13th Int'l Conf. Machine Learning, 1996, pp. 148-156.

[12] 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, 2010, pp. 108-116.

[13] K. Julisch, "Data mining for intrusion detection – A critical review", Applications of Data Mining in Computer 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] L. Khan, M. Awadand and B. Thuraisingham, "A new intrusion detection system using support vector machines and hierarchical clustering", VLDB J, Vol. 16, No. 4, 2006, pp. 507-521.

[16] W. Lee, S. J. Stolfo and K. W. Mok, "A data mining framework for building intrusion detection models', IEEE Symp. Security and Privacy, 1999, pp. 120-132.

[17] 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.

[18] Z. Ma, L. Zhen and X. Liao, "On the efficiency of support vector classifiers for intrusion detection", Int'l Conf. Neural Networks and Brain, 2005, pp. 935-940.

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

[20] 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.

[21] OWASP http://www.owasp.org/

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

[23] 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, 2010, pp. 26-26.

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

[25] D. Ramsbrock, R. Berthier and M. Cukier, "Profiling attacker behavior following SSH compromises", 37th Int'l Conf. Dependable Systems and Networks, 2007, pp. 119-124.

[26] K. Rieck, T. Holz, C. Willems, P. Dussel and P. Laskov, "Learning and classification of malware behavior", 5th Int'l Conf. Detection of Intrusions and Malware, 2008, pp. 108-125.

[27] 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/

[28] G. Salles-Loustau, R. Berthier, E. Collange, B. Sobesto and M. Cukier, "Characterizing attackers and attacks: An empirical study", 17th Pacific Rim Int'l Symp. Dependable Computing, 2011, pp. 174-183.

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

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

[31] M. Vrable, et al. "Scalability, fidelity, and containment in the Potemkin virtual honeyfarm", SIGOPS Oper. Syst. Rev. Vol. 39, No. 5, Oct. 2005, pp. 148-162.

[32] I. H. Witten and E. Frank, "Data mining: Practical machine learning tools with Java implementations", 2000. Morgan Kaufmann, San Francisco.

[33] 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.

[34] X. Xu, "Adaptive intrusion detection based on machine learning: Feature extraction, classifier construction and sequential pattern prediction", Int'l Journal of Web Services Practices Vol.2, No.1-2, 2006, pp. 49-58.