Classification of HTTP Attacks: A Study on the ECML/PKDD 2007 Discovery Challenge

Brian Gallagher and Tina Eliassi-Rad
Lawrence Livermore National Laboratory
P.O. Box 808, L-560, Livermore, CA 94551
{bgallagher, eliassi}@llnl.gov

Abstract

As the world becomes more reliant on Web applications for commercial, financial, and medical transactions, cyber attacks on the World Wide Web are increasing in frequency and severity. Web applications provide an attractive alternative to traditional desktop applications due to their accessibility and ease of deployment. However, the accessibility of Web applications also makes them extremely vulnerable to attack. This inherent vulnerability is intensified by the distributed nature of Web applications and the complexity of configuring application servers. These factors have led to a proliferation of Web-based attacks, in which attackers surreptitiously inject code into HTTP requests, allowing them to execute arbitrary commands on remote systems and perform malicious activities such as reading, altering, or destroying sensitive data.

One approach for dealing with HTTP-based attacks is to identify malicious code in incoming HTTP requests and eliminate bad requests before they are processed. Using machine learning techniques, we can build a classifier to automatically label requests as "Valid" or "Attack." For this study, we develop a simple, but effective HTTP attack classifier, based on the vector space model used commonly for Information Retrieval. Our classifier not only separates attacks from valid requests, but can also identify specific attack types (e.g., "SQL Injection" or "Path Traversal").

We demonstrate the effectiveness of our approach through experiments on the ECML/PKDD 2007 Discovery Challenge data set. Specifically, we show that our approach achieves higher precision and recall than previous methods. In addition, our approach has a number of desirable characteristics, including robustness to missing contextual information, interpretability of models, and scalability.

Keywords: Cybersecurity, pattern classification, machine learning.

1. Introduction

From a security standpoint, the World Wide Web is both critically important and critically vulnerable. The Web is critically important because it provides access to (1) crucial services (e.g., e-commerce, banking, taxes) and (2) sensitive data (e.g., financial, medical). The past decade has seen incredible growth in the availability of online services and information. According to recent estimates, 80% or more of U.S. Internet users (i.e., hundreds of millions) make purchases online [1, 3]. The Web is critically vulnerable because it is (1) inherently public and accessible and (2) inherently distributed. Recent studies have found that over 90% of Web hosts have serious vulnerabilities [9, 7] and 60 of the 100 most popular Web sites hosted or were involved in malicious activity during a six month period in 2008 [8].

The types of attacks made possible by the Web are numerous and varied, but generally involve attackers injecting code into an HTTP request that can execute on the Web server during the processing of the request. Such code can allow an attacker to execute arbitrary commands on remote systems. The havoc that an attacker can wreak on a vulnerable system is essentially unlimited. To give the reader a feel for what is possible, Figure 1 describes a *SQL Injection*, an example of an attack that can be embedded in an HTTP request.

Our approach for dealing with HTTP-based attacks is to identify malicious code in incoming HTTP requests and eliminate bad requests before they are processed. In particular, we utilize the vector space model [6] used commonly for Information Retrieval to build an effective classifier that automatically labels requests as "Valid" or "Attack." In case of an "Attack," our classifier also identifies the type of attack in the HTTP request.

Section 2 describes the ECML/PKDD 2007 Discovery Challenge, a data mining competition which we leverage to ground our study with a specific data set, prediction task, and previous classification approaches. Section 3 describes our approach to HTTP attack classification, based

Figure 1. SQL Injection. The "Cookie" field of this HTTP request contains an embedded SQL statement. This SQL statement, when executed, reads data from a database and writes it to a file that is publicly available on the Web and can later be retrieved by the attacker.

on the vector space model from Information Retrieval. Section 4 describes an evaluation of our method based on the ECML/PKDD Discovery Challenge. Finally, Section 5 presents our conclusions.

2. The ECML/PKDD 07 Discovery Challenge

The ECML/PKDD 2007 Discovery Challenge was a data mining competition held in conjunction with the 18th European Conference on Machine Learning (ECML) and the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD).

2.1 Objectives

The Discovery Challenge outlined two tasks related to HTTP attack detection:

1. **Classification of HTTP requests** into attack types. Submissions were to be judged on runtime performance as well as classification performance.

2. **Isolation of attack patterns**. The goal of this task was to identify (within an attack) the shortest string that conveys the attack.

For our study, we focus on the first task: classification of requests into attack types.

2.2 Data and Classification Task

The Discovery Challenge data was provided by challenge co-organizers LIRMM (Laboratoire dInformatique, de Robotique et de Microelectronique de Montpellier, FRANCE) and the LGI2P (Ecole des Mines dAlcs, FRANCE); and was based on a dataset of real-world web traffic in conjunction with Bee Ware. The data was divided into a training set and a test set. The training set was made available to challenge participants for experimentation prior to submission. The test set was released only once the Discovery Challenge was complete.

Each training and test instance in the data set contained the full text of the http request, divided into the following components: *method*, *protocol*, *uri*, *query*, *headers*, and *body*. In addition, each HTTP request included the following contextual attributes:

- Operating system running on the Web Server (UNIX, WINDOWS, UNKNOWN)

- HTTP Server targeted by the request (APACHE, MIIS, UNKNOWN)

- Is the XPATH technology understood by the server? (TRUE, FALSE, UNKNOWN)

- LDAP database on the Web Server? (TRUE, FALSE, UNKNOWN)

- SQL database on the Web Server? (TRUE, FALSE, UNKNOWN)

Finally, each request was assigned one or more of 8 possible class labels (i.e., "Valid" + 7 attack types). Table 1 shows the sizes and class distributions of both training and test sets for the Discovery Challenge.

	Training Set	Test Set
Total Requests	50,116	70,143
Valid Requests	35,006 (70%)	42,006 (60%)
Attacks	15,110 (30%)	28,137 (40%)
Cross-Site Scripting	12%	11%
SQL Injection	17%	18%
LDAP Injection	15%	16%
XPATH Injecction	15%	16%
Path traversal	20%	18%
Command execution	23%	23%
SSI attacks	13%	12%

Table 1. Characteristics of Discovery Challenge training and test sets. Note that percentages of attack types do not sum to 100% since some request instances are examples of more than one attack type.

2.3 Submissions

Over 25 groups registered for the Discovery Challenge, but only two submitted final results. According to Raïssi et al. [5], most researchers failed to submit results because they found that traditional data mining approaches were unsuccessful and felt that a specialized knowledge of attack detection was required to adequately address the problem. The two groups that submitted results took very different approaches to the problem.

Pachopoulos et al. [4] tried two different approaches. Approach #1: Extract binary features from the HTTP request data, perform feature selection, and then apply C4, a standard supervised-learning decision-tree algorithm, to build a classifier. Approach #2: Use the string representations of the various HTTP fields directly as features for input to a Support Vector Machine using a String Kernel. The authors abandoned the SVM approach in favor of the C4 approach after the former failed to deliver satisfactory performance.[1] The binary features used as input to C4 are based on the presence or absence of a number of attack indicators, derived manually by the authors.

The approach of Exbrayant [2] is based on constructing a language model that is used to define HTTP attack patterns. The author notes that attack patterns consist of sequences of keywords, variables, and symbols. Thus, the approach derives rules based on such sequences that can be used to identify the beginning and end of attack strings in HTTP requests. The core of this approach involves extracting and evaluating potential rules. Once this is done, it is straightforward to classify a candidate request based on whether it matches a given rule.

[1] Pachopoulos et al. [4] used WEKA's implementation of C4 and SVM [10].

3. A Vector Space Model For Traffic Classification

The vector space model was proposed by Salton et al. [6] as a way to efficiently index and compare text documents for information retrieval applications. In this model, a text document is represented by a weighted term vector, where terms with higher weight are considered to be more important or representative.

Although an HTTP request is semi-structured (i.e., information is divided into specific fields), the idea behind our approach is to ignore this structure and simply treat a request as unstructured text. This allows us to treat HTTP request classification as a document retrieval problem.

3.1 Learning

We train our classifier as follows:

- Merge all requests of a particular attack type into a single text document. This gives us a corpus of 8 documents (i.e., "Valid" + 7 attack types).
- Documents are tokenized by whitespace, '+' characters, and URL encoded characters (e.g., "%20").
- Tokenized documents are then represented as term vectors, where weights for each term are calculated using tf-idf [6]. A tf-idf weight consists of two components: *tf*, the term frequency and *idf*, the inverse document frequency. The tf component rewards terms that occur frequently in a document, whereas the idf component penalizes terms that also occur in many other documents.

  For term t and document d in corpus D, tf-idf is defined as:

$$tfidf(t,d) = tf(t,d) \cdot idf(t) \tag{1}$$

$$tf(t,d) = \frac{count(t,d)}{\sum_{v \in d} count(v,d)} \tag{2}$$

$$idf(t) = log \frac{|D|}{|\{d \in D : t \in d\}|} \tag{3}$$

  where $count(t,d)$ returns the number of occurrences of term t in document d.

3.2 Inference

Once trained, we apply our classifier to a new incoming HTTP request as follows. Incoming requests are treated as queries, which are converted to term vectors as described above and then compared against each of the 8 attack type term vectors. The request is then assigned the attack type that provides the best match (i.e., the class of the "document" most relevant to the "query").

More specifically, for each incoming request, we calculate a probability distribution over the possible classes as follows:

$$P(A = a|R) = \alpha \cdot sim_{cos}(a, R) \tag{4}$$

where A is a random variable representing attack type (i.e., class), a is specific assignment of A (i.e., a specific attack type), α is a normalizing constant to ensure $\sum_a P(A = a|R) = 1$, R is the incoming HTTP request, and sim_{cos} is the cosine similarity function, defined as follows:

$$sim_{cos}(a, R) = \frac{\vec{a} \cdot \vec{R}}{\|\vec{a}\| \cdot \|\vec{R}\|}$$

$$= \frac{\sum_{t \in a \cap R} tfidf(t, a) \cdot tfidf(t, R)}{\sqrt{\sum_{t \in a} tfidf(t, a)^2} \cdot \sqrt{\sum_{t \in R} tfidf(t, R)^2}} \tag{5}$$

Based on the probability distribution calculated in Equation (4) above, a class is assigned according to the following decision rule:

$$class(R) = \begin{cases} Valid & \text{if } P(A = Valid) > T \\ \underset{a}{\text{argmax}}\, P(A = a|R) & \text{otherwise} \end{cases}$$

where the decision threshold T is set based on 10-fold cross-validation performance on the training data set.

3.3 Time Complexity

Our approach to HTTP attack classification is quite efficient. For completeness, we discuss the complexity of both learning (i.e., training the model) and inference (i.e., applying the model). However, complexity of inference is of primary importance since learning can occur offline and infrequently, whereas inference must occur in real-time as requests are processed.

3.3.1 Complexity of Learning

Training our model involves calculating a tf-idf weight for each term in each request in our training data set. In a single pass through the training set, we can calculate $count(t, d)$, $\sum_{v \in d} count(v, d)$, and $idf(t)$. Then, using these components, the complete tf-idf weights can be calculated on a second pass.

Therefore, the complexity of the learning algorithm is $O(K)$ where K is the total number of tokens in the training set. In other words, learning is $O(|D| \cdot L_d)$ where L_d is the average length of documents in D.

3.3.2 Complexity of Inference

Inference is similar to learning except that: (1) we have only a single request q instead of a full training set and (2) in addition to calculating tf-idf weights, we must also calculate sim_{cos}. Since sim_{cos} depends only on the term weights, which can be calculated in $O(|q|)$ time, the full inference algorithm for 8 attack classes runs in $O(8 \cdot |q|) = O(|q|)$ time. Note that since every token in the request must be read by the server anyway in order to process the request, we cannot expect to process requests in less than linear time.

4. Evaluation

This section describes our evaluation of the tf-idf attack classifier with respect to the ECML/PKDD 2007 Discovery Challenge. We perform experiments to evaluate the classifier's performance on the Discovery Challenge training set as well as its ability to generalize to the separate Discovery Challenge test set.

4.1. Experimental Methodology

In order to evaluate the performance of our attack classifier on the Discovery Challenge training set, we divide the set into 10 disjoint folds and perform 10-fold cross-validation. To evaluate the generalization performance of our classifier, we replicate the setup of the Discovery Challenge, training on the full training set and then evaluating on the full test set.

We compare our results to those reported in the individual studies of the Discovery Challenge participants [2, 4] as well as the organizers' report on the results of the Discovery Challenge competition [5]. The official performance measures used in the Discovery Challenge are the standard information retrieval metrics of precision, recall, and F1 score. In addition, we report accuracy (for attack vs. non-attack) and area under the ROC curve (AUC). For the full 8-class task, we calculate AUC as a weighted average of the 1-vs-rest AUC scores over all classes.

	Classifier		
	tf-idf	**LM**	**C4**
Training Set Only			
Accuracy	> 0.99	-	0.77
AUC	> 0.99	-	-
Precision	> 0.99	0.98	-
Recall	> 0.99	0.93	-
F1	> 0.99	0.96	-
Discovery Challenge (Train/Test)			
Accuracy	0.94	-	-
AUC	0.97	-	-
Precision	0.98	0.82	0.50
Recall	0.88	0.78	0.47
F1	0.93	0.80	0.48

Table 2. Summary of results for all classifiers and experiments. Our approach is identified as *tf-idf*, *LM* is the language modelling approach of Exbrayat [2] and *C4* is the decision tree based approach of Pachopoulos et al. [4]

4.2. Results

Table 2 summarizes the results for all experiments and classifiers. We see that our tf-idf based approach delivers high performance across tasks and consistently outperforms the competing approaches.

Figure 2 shows the results of our tf-idf attack classifier using 10-fold cross-validation on the full training set. We produce a precision-recall curve by varying the decision threshold T. We see that the tf-idf classifier produces a near-perfect precision recall curve.

Figure 3 shows the results of several classifiers on the full Discovery Challenge problem (i.e., train on the full training set and test on the full test set). For, the tf-idf attack classifier, we show the full precision/recall curve. The circle represents the precision and recall obtained using the decision threshold T derived from our cross-validation experiments on the training set. We see that the tf-idf classifier outperforms both competing approaches in terms of both precision and recall.

For all of the results presented so far, the classifiers had access to the contextual information described in Section 2.2 (e.g., operating system, web server, etc). Figure 4 shows the results of an experiment where this contextual information is unavailable. Since the tf-idf classifier does not rely on contextual information, its performance is unchanged when this information is unavailable. However, without context, the performance of the other approaches degrades dramatically.

Figure 5 shows classifier performance on individual attack types. The tf-idf classifier demonstrates consistently high performance (i.e., F > 0.75) over the range of attack

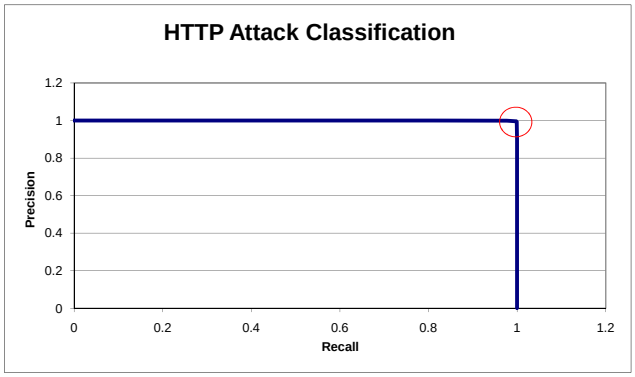

Figure 2. Training Data Results. 10-fold CV on training data. Circle represents decision threshold T for reported results.

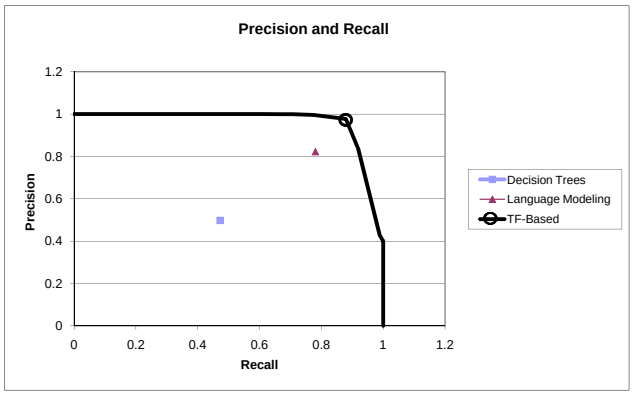

Figure 3. Discovery Challenge Results. Train on training set, evaluate on test set. Circle represents decision threshold T for reported results, derived from training data.

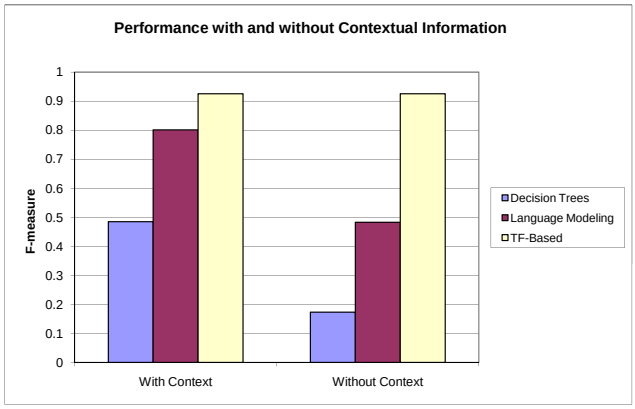

Figure 4. Effect of Context. F1 performance of classifiers with and without contextual information.

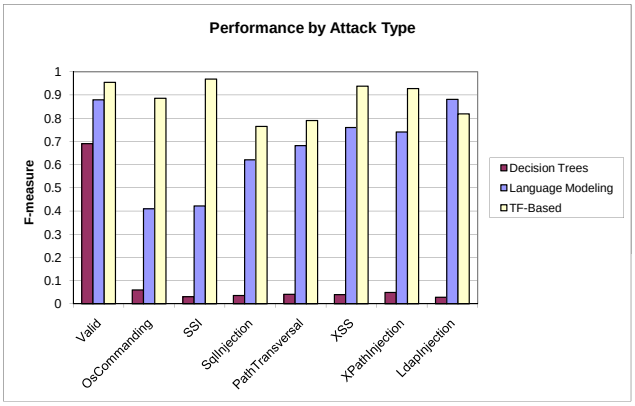

Figure 5. Attack-type Performance. F1 performance of classifiers for specific attack types. The performance of TF-based approach is more consistent across attack types.

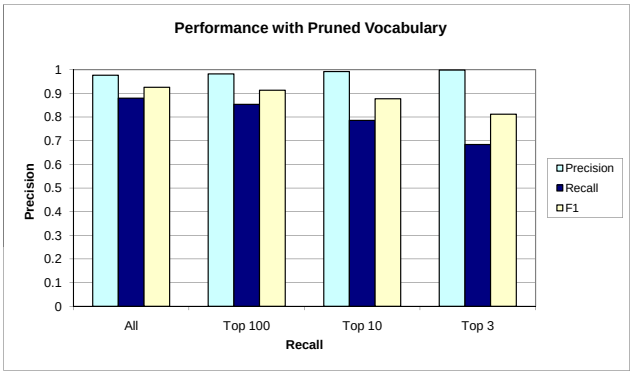

Figure 6. Performance with Pruned Vocabulary. Precision and recall of TF-based classifier as we reduce the size of the vocabulary we consider (top-k terms for each class). TF-based classifier is robust small vocabulary sizes.

types. Other methods perform poorly (i.e., F < 0.5) for at least some attack types.

Another advantage of the tf-idf based classifier is that it produces models that can be read and interpreted by a human analyst. More specifically, the model consists of a list of "keywords" for each attack type, ranked in order of relevance (i.e., tf-idf weight). Table 3 shows the top five keywords in the training and test sets for each of the 8 attack classes. Those familiar with the specific HTTP attacks will recognize many of the top terms for the 7 attack classes. Note that the terms associated with the *Valid* (i.e., non-attack) class are less meaningful and have correspondingly lower tf-idf weights. This is because *Valid* is basically a catch-all category that combines all requests that do not contain one of the 7 attack types. Therefore, we would expect little consistency among the set of *Valid* requests.

Finally, Figure 6 demonstrates the effect on the tf-idf classifier of pruning the number of terms in our vocabulary. Since classifier runtime scales with the length of the term vectors, we can considerably improve runtime performance by limiting the number of terms we consider. Figure 6 shows that we can achieve good classification performance with a very small term vocabulary (e.g., only the top 3 tf-idf weight terms for each attack class). Note that as we reduce the vocabulary size, we observe an increase in precision and a decrease in recall. We expect this intuitively because: (1) the presence of the very top terms is a strong indicator of class membership, so false positives decrease and (2) we are removing a number good terms, so false negatives increase. Note also that we can always adjust our decision threshold T to re-balance precision and recall as required by our application.

5. Conclusions

We propose a novel approach to HTTP attack classification based on the vector space model from information retrieval and demonstrate the effectiveness of our approach by applying it to the ECML/PKDD 2007 Discovery Challenge problem.

Our term-frequency based classifier has a number of desirable properties.

- It performs favorably in comparison with existing methods.
- It is efficient, running in time proportional to the size of an incoming request. Increased efficiency can be achieved with minimal classification-performance impact by pruning the term vocabulary.
- It produces understandable models.
- It is impervious to missing contextual information.
- It is simpler and more general than existing approaches (i.e., it does not rely on a domain-specific language model or a feature-set).

6. Acknowledgements

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344 (LLNL-TR-414570).

	Training Data		Test Data	
LDAP Injection	had*	0.005844	had*	0.004065
	objectclass	0.003944	objectclass	0.003861
	*o	0.003872	*o	0.002828
	brien*	0.003872	brien*	0.002828
	netscaperoot	0.001978	displayname	0.002616
Command Execution	..	0.003871	..	0.003558
	dir	0.003546	/c	0.003229
	/c	0.003328	dir	0.002507
	–	0.001650	–	0.001733
	../winnt/system32/cmd.exe	0.001612	../winnt/system32/cmd.exe	0.001678
Path Traversal	..	0.016041	..	0.016415
	.	0.005513	.	0.008600
	virtual	0.002526	virtual	0.001427
	–	0.001713	–	0.000968
	include	0.001263	file	0.000927
SSI Attack	–	0.006241	–	0.006003
	virtual	0.003719	statement	0.002167
	include	0.001859	odbc	0.002167
	statement	0.001275	virtual	0.001967
	odbc	0.001275	progra	0.000810
SQL Injection	**	0.003874	**	0.005199
	select	0.000883	statement	0.001163
	statement	0.000832	odbc	0.001163
	odbc	0.000832	–	0.000807
	union	0.000805	union	0.000674
XPATH Injection	path	0.005394	path	0.005449
	count	0.005108	count	0.005072
	child	0.003756	text	0.002616
	text	0.002421	comment	0.002093
	position	0.002200	child	0.001065
Cross-Site Scripting	document.cookie	0.006498	document.cookie	0.006504
	alert	0.004298	alert	0.004287
	javascript	0.003463	javascript	0.003449
	document.location.replace	0.003209	document.location.replace	0.003208
	url	0.001644	http	0.002411
Valid (No Attack)	13224	0.000061	dddddd	0.002054
	213.191.153.150	0.000057	lkl	0.000969
	9055,045,32	0.000055	large–majorite*des_membres	0.000751
	27260320301	0.000054	ministre-de-l-enseignement-superieur	0.000265
	13.228.134.190	0.000054	tehgghgjty	0.000259

Table 3. Top terms for each attack type, sorted by tf-idf weight.

References

[1] eMarketer. Where are all the online shoppers going? http://www.emarketer.com/Article.aspx?R=1004909, 2007.

[2] M. Exbrayat. Analyzing web traffic: A boundaries signature approach. In *Proceedings of the ECML/PKDD'2007 Discovery Challenge*, pages 53–64, 2007.

[3] iCrossing. How america searches: Online retail. http://www.icrossing.com/research/how-america-searches-online-shopping-2007.php, 2007.

[4] K. Pachopoulos, D. Valsamou, D. Mavroeidis, and M. Vazirgiannis. Feature extraction from web traffic data for the application of data mining algorithms in attack identification. In *Proceedings of the ECML/PKDD'2007 Discovery Challenge*, pages 65–70, 2007.

[5] C. Raïssi, J. Brissaud, G. Dray, P. Poncelet, M. Roche, and M. Teisseire. Web analyzing traffic challenge: Description and results. In *Proceedings of the ECML/PKDD'2007 Discovery Challenge*, pages 47–52, 2007.

[6] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

[7] Web Application Security Consortium. Web application security statistics project 2007. http://www.webappsec.org/projects/statistics/wasc_wass_2007.pdf, 2008.

[8] Websense Security Labs. New research: Web 2.0 and "legit" web sites are latest playground for information-stealing computer criminals. http://investor.websense.com/releasedetail.cfm?ReleaseID=324871, 2008.

[9] WhiteHat Security. Whitehat website security statistics report. http://www.whitehatsec.com/home/assets/WPStatsreport_100107.pdf, 2007.

[10] I. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.