Applying Machine Learning Techniques for Detection of Malicious Code in Network Traffic

Yuval Elovici, Asaf Shabtai, Robert Moskovitch, Gil Tahan, and Chanan Glezer

Deutsche Telekom Laboratories at Ben-Gurion University,
Be'er Sheva, 84105 Israel
{elovici,shabtaia,robertmo,gilta,chanan}@bgu.ac.il

Abstract. The Early Detection, Alert and Response (eDare) system is aimed at purifying Web traffic propagating via the premises of Network Service Providers (NSP) from malicious code. To achieve this goal, the system employs powerful network traffic scanners capable of cleaning traffic from known malicious code. The remaining traffic is monitored and Machine Learning (ML) algorithms are invoked in an attempt to pinpoint unknown malicious code exhibiting suspicious morphological patterns. Decision trees, Neural Networks and Bayesian Networks are used for static code analysis in order to determine whether a suspicious executable file actually inhabits malicious code. These algorithms are being evaluated and preliminary results are encouraging.

Keywords: Malicious Code, Machine Learning, Network Service Provider (NSP), Feature Selection.

1 Introduction

In a recent online safety survey conducted by America Online and the National Cyber Security Alliance (NCSA), 81% of the respondents were found to be lacking recently-updated anti-virus software, a properly-configured firewall, and/or spyware protection. Nevertheless, 74% of the respondents use the Internet for "sensitive" transactions from their home computers, including among others banking, stock trading, and reviewing personal medical information [1-4]. One way to prevent users from being infected by malicious code is to purify the network traffic by the Network Service Provider (NSP). The Early Detection, Alert and Response (eDare) system was designed to accomplish this task [5]. The proposed system employs powerful network traffic scanners to remove known malicious code from network traffic. The remaining suspicious traffic is monitored and ML techniques, such as classification algorithms, are invoked for identifying unknown malicious code. Each ML technique is implemented as a modular plug-in appended to the core system. *Decision Trees, Bayesian Networks* and *Artificial Neural Network* plug-ins are all used in this study for static code analysis in order to determine whether a file contains malicious code.

Detection of malicious employing content-based features operationalized by ML learning algorithms is not a new concept. In [6], ML techniques were applied for detection of unknown malicious code based on binary code content. They used three

J. Hertzberg, M. Beetz, and R. Englert (Eds.): KI 2007, LNAI 4667, pp. 44–50, 2007.

different feature extraction methods: Program Header, Strings features and Byte Sequences features, on which they applied four classifiers: Signature-based method (anti-virus), Ripper – a rule based learner, Naïve Bayes and Multi Naïve Bayes. Their study showed that all ML methods had better accuracy than the signature-based algorithm. Other studies used various features extracted from the suspected binary code, feature selection methods and ML techniques in order to detect unknown malicious code [6-8].

The major advantages of eDare when compared to the aforementioned approaches are: first, its flexibility in using plug-ins (without recompiling the system when adding or updating plug-ins); and second, its ability to weigh and integrate the results of the multiple plug-ins into one final detection decision with superior accuracy than each of the individual plug-ins. In this paper we present a short review on eDare and its plug-ins and will show some preliminary evaluation results that demonstrate the system ability to detect unknown malicious code.

The rest of the paper is structures as follows: Section 2 describes the architecture of the eDare system; section 3 presents the ML algorithms employed for detection of unknown malicious code; section 4 illustrates the preliminary results emanating from an empirical evaluation of eDare. Finally, section 6 concludes the paper with a summary and an overview of future research avenues.

2 The eDare Framework

This research adopts a distributed, network-based approach, in which NSPs constantly monitor traffic flowing via their infrastructure in an attempt to detect and remove malicious code (hereafter: eThreats). eDare is designed to provide maximum automation in the cycle of intercepting, analyzing, alerting and overcoming instances of eThreats. The system aims to provide very low false positive by integrating multiple sources of information and multiple eThreat detection techniques. Finally, the system easily accommodates external plug-ins, expert consultation and risk assessment.

The conceptual architecture of eDare is depicted in Figure 1. The group of eThreats faced by eDare can be classified into the following two types:

- **Known** eThreats for which eDare has already generated a distinct signature and are intercepted by the **Known eThreat handling Module**; and
- **Unknown** (New) eThreats which eDare is yet to encounter and classify, and for which eDare needs to generate a distinct signature via the **New eThreat handling Module.** In order to enable flexibility in employing various new eThreat detection algorithms, modular plug-ins are used. All plug-ins have a similar interface, that is, a suspicious file to be examined as input, and a threat rank as output. Examples of plug-ins incorporated by eDare include: a statistical plug-in, static analysis (morphological) plug-in, ML plug-in and a collaborative plug-in. Finally, a **Risk-weighing** function collects all ranks from the relevant plug-ins and calculates an integrated rank according to some weighing scheme. In case the final rank of a file is above a devisable threshold, the file will be transferred to the **Signature builder Module** which will construct a unique signature.

When encountering new eThreats or suspicious behaviors, the response time and effectiveness of the system is substantially expedited by sharing observations and warnings between users and the system via a **Collaborative Module**, which further propagates relevant alerts across the protected network.

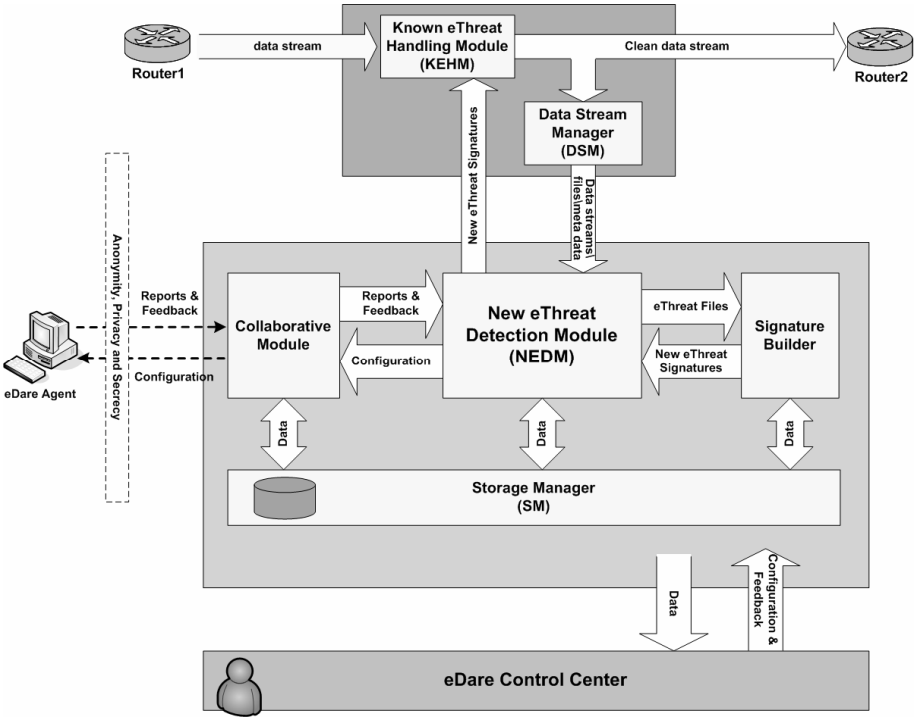

Fig. 1. Architecture of the eDare System

The role of the **Data Stream Manager Module** is to obtain a "clean" network data stream from the Known eThreat Handler Module**.** It then extracts and assembles files from the continuous network data stream and forwards them to the New eThreat Detection Module. The overall configuration and operation of the system is controlled by a human administrator via a console termed **eDare Control Center.**

3 Detecting Malicious Code Using ML Techniques

In this paper we present how eDare performed when using five plug-ins. The plug-ins were based on three ML techniques (described in subsections 3.1-3.3) and on two types of inputs (described in subsection 3.4).

3.1 Decision Trees

Decision tree learners [9] are a well-established family of ML algorithms. Classifiers are represented as trees whose internal nodes are tests on individual features and

leaves are classification decisions. Typically, a greedy heuristic search method is used to find a compact decision tree that correctly classifies the training data. The decision tree is induced from the dataset by splitting the variables based on the *expected information gain*. Modern implementations include pruning which avoids over-fitting. In this study we evaluated J48, the Weka [10] version of the commonly used C4.5 algorithm [9]. An important characteristic of Decision Trees is the explicit form of their knowledge which can be easily represented as a set of rules.

3.2 Bayesian Networks

Bayesian networks are a form of probabilistic graphical modeling [11]. Specifically, a Bayesian network is a directed acyclic graph of nodes with variables and arcs representing dependencies among the variables. Bayesian networks are based on Bayes' theorem, and they are known for their ability to represent conditional probabilities which capture the internal relationships between the studied variables. A Bayesian network can thus be considered a mechanism for automatically constructing extensions of Bayes' theorem to more complex problems. We evaluated the Bayesian Network standard version which comes with WEKA [10].

3.3 Artificial Neural Network

An Artificial Neural Network (ANN) [12] is an information processing paradigm inspired by information processing mechanisms of biological nervous systems (i.e., the brain). The key element of this paradigm is the structure of the information processing system modeled as a network comprising many highly interconnected Processing Elements (PEs) (also termed Neurons). Neurons work together in order to approximate a specific transformation function. An ANN is configured for a specific application, such as pattern recognition or data classification, through a *learning process* during which the weights of the inputs in each neuron are updated. The weights are updated by a *training algorithm*, such as back-propagation, according to examples the network receives in order to reduce the value of an *error function*. The power and usefulness of ANN have been demonstrated in numerous applications including speech-synthesis, medicine, finance and many other pattern-recognition problems., Neural models show more promise in achieving human-like performance when dealing with unstructured application domains where traditional AI knowledge representation techniques (i.e., rule/frame bases) are cumbersome. All ANN manipulations in this study have been performed within the MATLAB(r) environment using the Neural Network Toolbox [13].

3.4 Feature Selection

In order to determine whether a suspected file is malicious or not, using ML techniques, we extracted two types of static features: n-grams and Win32 executables Portable Executable header. We implemented a tool that extracts 5-grams from the binary representation of a file. Out of all 5-grams that were extracted we chose the top 5,500 which are the most frequent in the file set. Next, for each 5-gram we calculated its *tf-idf* score. The *tf-idf* score is mainly used as a text retrieval method. tf_i is the term frequency (i.e., n-gram) in the file *i* and *idf* is the log value of the number of files in

whole repository, divided by the number of files that include that term. We used Fisher's Score [14] for feature selection in order to choose the top 300 5-grams out of the 5,500 5-grams, to be used for the evaluation. Next, PE features were extracted from Win32 executables using the PE Feature Extractor tool which parses an EXE\DLL file according to PE Format rules. We statically extracted different PE format features representing information contained within each Win32 PE binary (EXE or DLL). Examples of the information extracted are: creation\modification time, machine type, file size, linker version, section alignment, code size, imported DLLs, exported functions, etc.

4 Preliminary Evaluation Results

eDare was deployed and tested in a network-security lab with distinct "clean" and "infected" environments. We collected a repository of 7694 malicious files representing a variety of eThreats and 22736 benign files. This collection was used to train and test the effectiveness of each of eDare's plug-ins and their effect on the overall system's performance. After conducting a rigorous research, we chose to use the following plug-ins for classification of suspected files:

- ANN (5grams; top300; Fisher): Artificial Neural Network (ANN) classifier trained on the top 300 5-grams, selected using Fisher score.
- BN (5grams; top300; Fisher): Bayesian Network (BN) classifier trained on the top 300 5-grams, selected using Fisher score
- DT (5grams; top300; Fisher): Decision Tree classifier trained on the top 300 5-grams, selected using Fisher score.
- BN (PE): Bayesian Network (BN) classifier trained on the PE (Portable Executable) features.
- DT (PE): Decision Tree (DT) classifier trained on the PE (Portable Executable) features.

A simple voting scheme [15] was applied as an ensemble algorithm in order to generate a final recommendation from the classifications of the individual plug-ins.
In order to compare the various plug-ins we used the following common evaluation measures: *True Positive Rate* (*TPR*), *False Positive Rate* (*FPR*) and *Accuracy*. TPR is the proportion of malware files classified correctly; FPR is the proportion of benign files misclassified; and Accuracy is the number of correctly classified instances (either malware or benign), divided by the entire number of instances. We also used *Receiver Operating Characteristics* (*ROC*) curves. A ROC curve is a graphical representation of the trade-off between the true positive (TPR) and false positive rates (FPR) for every possible cut-off.

Each plug-in was trained using 30% of the dataset and tested using the rest of the dataset (70%). Table 1 describes the evaluation results when the detection threshold was set to 0.5, where FPR stands for the false positive rate, TPR stands for the true positive rate and the accuracy stands for the detection accuracy. It is easy to see in Table 1 that, among individual plug-ins, DT (PE) had the highest accuracy with lowest false positive. The results also show that simple voting improves the overall accuracy, reduces the false positive rate and improves the true positive rate compared to all of the individual plug-ins except the BN (PE).

The graph in Fig. 2 presents the ROCs of all individual plug-ins and their combination using a standard weighing mechanism. The weighing mechanism combines the individual plug-in decision into a final single decision based on the simple voting algorithm. It is easy to see in Fig. 2 that the Risk Weighting line provides the best true positive rate with a minimum false positive rate and thus should be preferred over each individual ML technique. This observation is also captured quantitatively by the Area Under Curve (AUC) measure which reaches its peak (0.983) in the case of the weighted line.

Table 1. Evaluation results for detection threshold = 0.5

	FPR	**TPR**	**Accuracy**
ANN (5grams; top300; Fisher)	0.038	0.893	0.945
BN (5grams; top300; Fisher)	0.206	0.885	0.816
BN (PE)	0.058	**0.933**	0.94
DT (5grams; top300; Fisher)	0.039	0.87	0.938
DT (PE)	**0.035**	0.925	**0.955**
Risk Weighing(Voting)	**0.032**	**0.928**	**0.958**

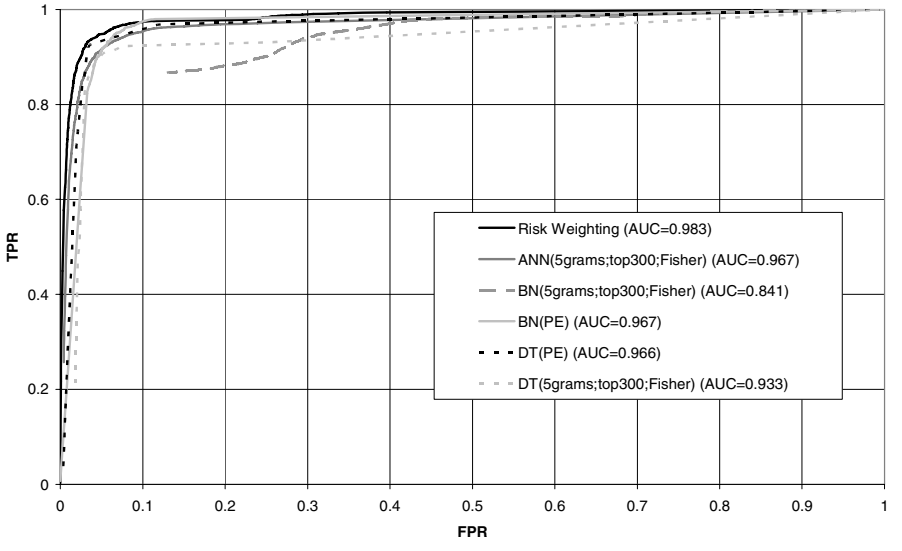

Fig. 2. Detection Plug-ins ROC

5 Summary

In this paper we presented a new system aimed at protecting users from various eThreats. eDare protects end-users by purifying NSPs' traffic from known eThreats without a mandatory requirement to install any software on end-users' computers. Sanitation of Web traffic from malicious code is performed by dedicated commercial

cleaning devices capable of removing threats based on prespecified signatures that eDare generates.

eDare can also detect unknown threats by employing, among others, ML algorithms as plug-ins in a flexible, open manner. Once a new threat is detected, eDare generates a signature and update the above cleaning devices deployed over NSPs' network.

Our empirical experimental findings suggest that a new eThreat risk-weighing scheme, weighing several ML algorithms (i.e., ANN, BN, DT) in various configurations (n-grams, PE) outperforms each of individual algorithms in terms of prediction accuracy. This important finding suggests that eDare's new eThreat detection module should be designed in a flexible, plug-in mode accommodating a heterogeneous collection of ML algorithms in order to facilitate better prediction accuracy of new eThreats.

References

1. NCSA Study, http://www.staysafeonline.info/pdf/safety_study_2005.pdf
2. Symantec Internet Security Threat Report (January-June 2004), www.symantec.com
3. The Danger of Spyware, Symantec Security Response (June 2003), http://www.symantec.com
4. Symantec 2006 Security Report, http://www.symantec.com/specprog/threatreport/entwhitepaper_symantec_internet_security_threat_report_x_09_2006.en-us pdf
5. Tahan, G., Glezer, C., Elovici, Y.: eDare- Early Detection Alert and Response to Electronic Threats, Working Paper, Deutsche Telekom Labs at Ben Gurion University
6. Schultz, M., Eskin, E., Zadok, E., Stolfo, S.: Data Mining Methods for Detection of New Malicious Executables. In: Proc. of the IEEE Symposium on Security and Privacy, pp. 178–184 (2001)
7. Abou-Assaleh, T., Cercone, N., Keselj, V., Sweidan, R.: N-gram based Detection of New Malicious Code. In: Proc. of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04) (2004)
8. Kolter, J.Z., Maloof, M.A.: Learning to detect malicious executables in the wild. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 470–478. ACM Press, New York, NY (2004)
9. Quinlan, J.R.: C4.5: Programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco (1993)
10. Weka software, http://www.cs.waikato.ac.nz/ml/weka/
11. Pearl, J.: Fusion, propagation, and structuring in belief networks. Artificial Intelligence 29(3), 241–288 (1986)
12. Bishop, C.: Neural Networks for Pattern Recognition. Clarendon Press, Oxford (1995)
13. Demuth, H., Beale, M.: Neural Network toolbox for use with Matlab. The Mathworks Inc., Natick, MA (1998)
14. Golub, T., Slonim, D., Tamaya, P., Huard, C., Gaasenbeek, M., Mesirov, J., Coller, H., Loh, M., Downing, J., Caligiuri, M., Bloomfield, C., Lander, E.: Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. Science 286, 531–537 (1999)
15. Bauer, E., Kohavi, R.: An empirical comparison of voting classification Algorithms. Bagging, Boosting, and Variants. Machine Learning 35, 1–38 (1999)