Polymorphic Worms Detection Using A Supervised Machine Learning Technique

Mohssen M. Z. E. Mohammed, H. Anthony Chan, Neco Ventura
Dept. of Electrical Engineering, University of Cape Town
Rondebosch, South Africa
m_zin44@hotmail.com; h.a.chan@ieee.org; neco@crg.ee.uct.ac.za

Mohsin Hashim, Eihab Bashier
Faculty of Mathematical Science, University of Khartoum
Khartoum, Sudan
mohsinhashim@yahoo.com; eihabbashier@gmail.com

Abstract— Polymorphic worms are considered as the most dangerous threats to the Internet security, and the danger lies in changing their payloads in every infection attempt to avoid the security systems. We have designed a novel double-honeynet system, which is able to detect new worms that have not been seen before. To generate signatures for polymorphic worms we have two steps. The first step is the polymorphic worms sample collection which is done by a Double-honeynet system. The second step is the signature generation for the collected samples which is done by using a Support Vector Machines (SVMs) technique. The system is able to generate accurate signatures for single and multiple worms.

Keywords— honeynet; worms; machine learning algorithm.

1. Introduction

An Internet worm is a self-propagated program that automatically replicates itself to vulnerable systems and spreads across the Internet. Worms take the attack process one step further by self-replicating. Once a worm has compromised and taken over a system, it begins scanning again, looking for new victims. Therefore a single infected system can compromise one hundred systems, each of which can compromise another one hundred more systems, and so on. The worm continues to attack systems this way and grows exponentially. This propagation method can spread extremely fast, giving administrators little time to react and ravaging entire organizations. Although only a small percentage of individuals can identify and develop code for worms, but once the code of a worm is accessible on the Internet, anyone can apply it. The very randomness of these tools is what makes them so dangerous. A polymorphic worm is a worm that changes its appearance with every instance [1].

It has been shown that multiple invariant substrings must often be present in all variants of worm payload. These substrings typically correspond to protocol framing, return addresses, and in some cases, poorly obfuscated code [8].

Intrusion detection systems serve three essential security functions: they monitor, detect, and respond to unauthorized activities. There are two basic types of intrusion detection: host-based and network-based. Host-based IDSs examine data held on individual computers that serve as hosts, while network-based IDSs examine data exchanged between computers [15, 16].

Our research is based on Honeypot technique. Developed in recent years, honeypot is a monitored system on the Internet serving the purpose of attracting and trapping attackers who attempt to penetrate the protected servers on a network. Honeypots fall into two categories. A high-interaction honeypot such as (Honeynet) operates a real operating system and one or multiple applications. A low-interaction honeypot such as (Honeyed) simulates one or multiple real systems. In general, any network activities observed at honeypots are considered suspicious [1, 9].

Supervised machine learning is the search for algorithms that reason from externally supplied instances to produce general hypotheses, which then make predictions about future instances. In other words, the goal of supervised learning is to build a concise model of the distribution of class labels in terms of predictor features. There are several applications for Machine Learning (ML), the most significant of which is data mining. People are often prone to making mistakes during analyses or, possibly, when trying to establish relationships between multiple features. This makes it difficult for them to find solutions to certain problems. Machine learning can often be successfully applied to these problems, improving the efficiency of systems and the designs of machines. Every instance in any dataset used by machine learning algorithms is represented using the same set of features. The features may be continuous, categorical or binary. If instances are given with known labels (the corresponding correct outputs) then the learning is called supervised, in contrast to unsupervised learning, where instances are unlabeled. By applying these unsupervised (clustering) algorithms, researchers hope to discover unknown, but useful, classes of items. Another kind of machine learning is reinforcement learning. The training information provided to the learning system by the
environment (external trainer) is in the form of a scalar reinforcement signal that constitutes a measure of how well the system operates. The learner is not told which actions to take, but rather must discover which actions yield the best reward, by trying each action in turn [17].

This paper is organized as follows: Section 2 reviews General issues of supervised learning algorithms. Section 3 discusses the related work regarding automated signature generation systems. Section 4 introduces the proposed system architecture to address the problems faced by current automated signature systems. Signature generation algorithm for Polymorphic Worm will be discussed in section 5. Section 6 concludes the paper.

2. General issues of supervised learning algorithms

Inductive machine learning is the process of learning a set of rules from instances (examples in a training set), or more generally speaking, creating a classifier that can be used to generalize from new instances. The process of applying supervised ML to a real-world problem is described in Figure 1.

The first step is collecting the dataset. If a requisite expert is available, then s/he could suggest which fields (attributes, features) are the most informative. If not, then the simplest method is that of “brute-force,” which means measuring everything available in the hope that the right (informative, relevant) features can be isolated. However, a dataset collected by the “brute-force” method is not directly suitable for induction. It contains in most cases noise and missing feature values, and therefore requires significant pre-processing.

The second step is the data preparation and data preprocessing. Depending on the circumstances, researchers have a number of methods to choose from to handle missing data. Hodge & Austin [22] have recently introduced a survey of contemporary techniques for outlier (noise) detection. These researchers have identified the techniques’ advantages and disadvantages. Instance selection is not only used to handle noise but to cope with the infeasibility of learning from very large datasets. Instance selection in these datasets is an optimization problem that attempts to maintain the mining quality while minimizing the sample size. It reduces data and enables a data mining algorithm to function and work effectively with very large datasets. There is a variety of procedures for sampling instances from a large dataset.

Feature subset selection is the process of identifying and removing as many irrelevant and redundant features as possible. This reduces the dimensionality of the data and enables data mining algorithms to operate faster and more effectively. The fact that many features depend on one another often unduly influences the accuracy of supervised ML classification models. This problem can be addressed by constructing new features from the basic feature set. This technique is called feature construction/transformation. These newly generated features may lead to the creation of more concise and accurate classifiers. In addition, the discovery of meaningful features contributes to better comprehensibility of the produced classifier, and a better understanding of the learned concept [17].

3. Related Work

Honeypots are an excellent source of data for intrusion and attack analysis. Levin et al. described how honeypot extracts details of worm exploits that can be analyzed to generate detection signatures [4]. The signatures are generated manually.

One of the first systems proposed was Honeycomb developed by Kreibich and Crowcroft. Honeycomb generates signatures from traffic observed at a honeypot via its implementation as a Honeyd [5] plugin. The longest common substring (LCS) algorithm, which looks for the longest shared byte sequences across pairs of connections, is at the heart of Honeycomb. Honeycomb generates signatures consisting of a single, contiguous substring of a worm’s payload to match all worm instances. These signatures, however, fail to match all polymorphic worm instances with low false positives and low false negatives.

Kim and Karp [6] described the Autograph system for automated generation of signatures to detect worms. Unlike Honeycomb, Autograph’s inputs are packet traces from a DMZ that includes benign traffic. Content blocks that match “enough” suspicious flows are used as input to COPP, an
algorithm based on Rabin fingerprints that searches for repeated byte sequences by partitioning the payload into content blocks. Similar to Honeycomb, Auto-graph generates signatures consisting of a single, contiguous substring of a worm’s payload to match all worm instances. These signatures, unfortunately, fail to match all polymorphic worm instances with low false positives and low false negatives.

S. Singh, C. Estan, G. Varghese, and S. Savage [7] described the Earlybird system for generating signatures to detect worms. This system measures packet-content prevalence at a single monitoring point such as a network DMZ. By counting the number of distinct sources and destinations associated with strings that repeat often in the payload, Earlybird distinguishes benign repetitions from epidemic content. Earlybird, also like Honeycomb and Autograph, generates signatures consisting of a single, contiguous substring of a worm’s payload to match all worm instances. These signatures, however, fail to match all polymorphic worm instances with low false positives and low false negatives.

New content-based systems like Polygraph, Hamsa and LISABETH [8, 10 and 11] have been deployed. All these systems, similar to our system, generate automated signatures for polymorphic worms based on the following fact: there are multiple invariant substrings that must often be present in all variants of polymorphic worm payloads even if the payload changes in every infection. All these systems capture the packet payloads from a router, so in the worst case, these systems may find multiple polymorphic worms but each of them exploits a different vulnerability from each other. So, in this case, it may be difficult for the above systems to find invariant contents shared between these polymorphic worms because they exploit different vulnerabilities. The attacker sends one instance of a polymorphic worm to a network, and this worm in every infection automatically attempts to change its payload to generate other instances. So, if we need to capture all polymorphic worm instances, we need to give a polymorphic worm chance to interact with hosts without affecting their performance. So, we propose new detection method “Double-honeynet” to interact with polymorphic worms and collect all their instances. The proposed method makes it possible to capture all worm instances and then forward these instances to the Signature Generator which generates signatures, using a particular algorithm.

An Automated Signature-Based Approach against Polymorphic Internet Worms by Yong Tang and Shigang Chen[9] described a system to detect new worms and generate signatures automatically. This system implemented a double-honeypots (inbound honeypot and outbound honeypot) to capture worms payloads. The inbound honeypot is implemented as a high-interaction honeypot, whereas the outbound honeypot is implemented as a low-interaction honeypot. This system has limitation. The outbound honeypot is not able to make outbound connections because it is implemented as low-interaction honeypot which is not able to capture all polymorphic worm instances. Our system overcomes this disadvantage by using double-honeynet (high-interaction honeypot), which enables us to make unlimited outbound connections between them, so we can capture all polymorphic worm instances.

4. Double- Honeynet System

We propose a double-honeynet system to detect new worms automatically. A key contribution of this system is the ability to distinguish worm activities from normal activities without the involvement of experts.

Figure 2 shows the main components of the double-honeynet system. Firstly, the incoming traffic goes through the Gate Translator which samples the unwanted inbound connections and redirects the samples connections to Honeynet 1.

The gate translator is configured with publicly-accessible addresses, which represent wanted services. Connections made to other addresses are considered unwanted and redirected to Honeynet 1 by the Gate Translator.

Secondly, Once Honeynet 1 is compromised, the worm will attempt to make outbound connections. Each honeynet is associated with an Internal Translator implemented in router that separates the honeynet from the rest of the network. The Internal Translator 1 intercepts all outbound connections from honeynet 1 and redirects them to honeynet 2 which does the same forming a loop.

Only packets that make outbound connections are considered malicious, and hence the Double-honeynet forwards only packets that make outbound connections. This policy is due to the fact that benign users do not try to make outbound connections if they are faced with non-existing addresses.

Lastly, when enough instances of worm payloads are collected by Honeynet 1 and Honeynet 2, they are forwarded to the Signature Generator component which generates signatures automatically using specific algorithms that will be discussed in the next section. Afterwards, the Signature Generator component updates the IDS database automatically by using a module that converts the signatures into Bro or pseudo-Snort format. The above proposed system implemented by using
VHVMware Server 2. The implementation results are out of the scope of this paper.

For further details on the double-honeynet architecture the reader is advised to refer to our published works [13].

5. Signature Generation Algorithms

In this section, we describe the Support Vector Machines technique which we use it to generates signatures for polymorphic worms.

Support Vector Machines (SVMs) are the newest supervised machine learning technique [17]. SVMs revolve around the notion of a “margin”—either side of a hyperplane that separates two data classes. Maximizing the margin and thereby creating the largest possible distance between the separating hyperplane and the instances on either side of it has been proven to reduce an upper bound on the expected generalization error.

If the training data is linearly separable, then a pair \((w, b)\) exists such that

\[
W^T X_i + b \geq 1, \text{ for all } X_i \in P
\]

\[
W^T X_i + b \leq -1, \text{ for all } X_i \in N
\]

With the decision rule given by \(f_{w,b}(X) = \text{sgn}(W^T X + b)\), where \(w\) is termed the weight vector and \(b\) the bias (or \(-b\) is termed the threshold).

It is easy to show that, when it is possible to linearly separate two classes, an optimum separating hyperplane can be found by minimizing the squared norm of the separating hyperplane. The minimization can be set up as a convex quadratic programming (QP) problem:

\[
\text{Minimize}_{w,b} \Phi(w) = \frac{1}{2} \|w\|^2
\]

Subject to \(y_i (W^T X_i + b) \geq 1, i=1,\ldots, l\).

In the case of linearly separable data, once the optimum separating hyperplane is found, data points that lie on its margin are known as support vector points and the solution is represented as a linear combination of only these points (see Figure 3). Other data points are ignored.

Therefore, the model complexity of an SVM is unaffected by the number of features encountered in the training data (the number of support vectors selected by the SVM learning algorithm is usually small). For this reason, SVMs are well suited to deal with learning tasks where the number of features is large with respect to the number of training instances.

A general pseudo-code for SVMs is illustrated in Figure 4.

1) Introduce positive Lagrange multipliers, one for each of the inequality constraints (1). This gives Lagrangian:

\[
L_p = \frac{1}{2} \|w\|^2 - \sum_{i=1}^{N} \alpha_i y_i (w^T x_i - b) + \sum_{i=1}^{N} \alpha_i
\]

2) Minimize \(L_p\) with respect to \(w, b\). This is a convex quadratic programming problem.

3) In the solution, those points for which \(\alpha_i > 0\) are called “support vectors.”

Even though the maximum margin allows the SVM to select among multiple candidate hyperplanes, for many datasets, the SVM may not be able to find any separating hyperplane at all because the data contains misclassified instances. The problem can be addressed by using a soft margin that accepts some misclassifications of the training instances. This can be done by introducing positive slack variables \(\xi_i, i =1,\ldots, N\) in the constraints, which then become:
Thus, for an error to occur the corresponding \( \xi_i \) must exceed unity, so \( \sum_i \xi_i \) is an upper bound on the number of training errors. In this case the Lagrangian is:

\[
I_p = \frac{1}{2} \| w \|^2 + C \sum_i \xi_i - \sum_i \alpha_i [y_i (x_i \cdot w - b) - 1 + \xi_i] - \sum_i \mu_i \xi_i
\]

Where the \( \mu_i \) are the Lagrange multipliers introduced to enforce positivity of the \( \xi_i \).

Nevertheless, most real-world problems involve non-separable data for which no hyperplane exists that successfully separates the positive from negative instances in the training set. One solution to the inseparability problem is to map the data onto a higher dimensional space and define a separating hyperplane there. This higher-dimensional space is called the transformed feature space, as opposed to the input space occupied by the training instances.

With an appropriately chosen transformed feature space of sufficient dimensionality, any consistent training set can be made separable. A linear separation in a transformed feature space corresponds to a non-linear separation in the original input space. Mapping the data to some other (possibly infinite dimensional) Hilbert space \( H \) as \( \Phi : \mathbb{R}^d \rightarrow H \). Then the training algorithm would only depend on the data through dot products in \( H \), i.e. on functions of the form \( \Phi \circ (x_j) \). If there were a “kernel function” \( K \) such that \( K(x_i, x_j) = \Phi(x_j) \cdot \Phi(x_j) \), we would only need to use \( K \) in the training algorithm, and would never need to explicitly determine \( \Phi \). Thus, kernels are a special class of function that allows inner products to be calculated directly in feature space, without performing the mapping described above. Once a hyperplane has been created, the kernel function is used to map new points into the feature space for classification.

The selection of an appropriate kernel function is important, since the kernel function defines the transformed feature space in which the training set instances will be classified. Genton [21] described several classes of kernels, however, he did not address the question of which class is best suited to a given problem. It is common practice to estimate a range of potential settings and use cross-validation over the training set to find the best one. For this reason a limitation of SVMs is the slow speed of the training. Selecting kernel settings can be regarded in a similar way to choosing the number of hidden nodes in a neural network. As long as the kernel function is legitimate, a SVM will operate correctly even if the designer does not know exactly what features of the training data are being used in the kernel-induced transformed feature space.

Some popular kernels are the following:

1. \( K(x, y) = (x \cdot y + 1)^p \)
2. \( K(x, y) = e^{-\| x - y \|^2 / 2 \alpha n} \)
3. \( K(x, y) = \tanh (K(x, y) - \delta)^p \)

Training the SVM is done by solving \( N \)th dimensional QP problem, where \( N \) is the number of samples in the training dataset. Solving this problem in standard QP methods involves large matrix operations, as well as time-consuming numerical computations, and is mostly very slow and impractical for large problems. Sequential Minimal Optimization (SMO) is a simple algorithm that can, relatively quickly, solve the SVM QP problem without any extra matrix storage and without using numerical QP optimization steps at all. SMO decomposes the overall QP problem into QP sub-problems. Keerthi and Gilbert [20] suggested two modified versions of SMO that are significantly faster than the original SMO in most situations.

Finally, the training optimization problem of the SVM necessarily reaches a global minimum, and avoids ending in a local minimum, which may happen in other search algorithms such as neural networks. However, the SVM methods are binary, thus in the case of multi-class problem one must reduce the problem to a set of multiple binary classification problems. Discrete data presents another problem, although with suitable rescaling good results can be obtained.

6. Conclusion

We have proposed automated detection for Zero day polymorphic worms using double-honeynet. We have proposed new detection method “Double-honeynet” to detect new worms that have not been seen before. The system is based on the Support Vector Machines technique that used to generate signatures for polymorphic worms. The main objectives of this research are to reduce false alarm rates and generate high quality signatures for polymorphic worms.

7. References


