A Hierarchical Anomaly Network Intrusion Detection System using Neural Network Classification

ZHENG ZHANG, JUN LI, C. N. MANIKOPOULOS, JAY JORGENSON and JOSE UCLES
ECE Department, New Jersey Inst. of Tech., University Heights, Newark, NJ 07102, USA
Department of Mathematics, CUNY, Convent Ave. at 138 ST., New York, NY 100031, USA
Network Security Solutions, 15 Independence Blvd. 3rd FL., Warren, NJ 07059, USA

Abstract: In this paper, we introduce a hierarchical anomaly network intrusion detection system, which is capable of detecting network-based attacks using statistical preprocessing models and neural network classification. The sample network used has a three-tier hierarchy, where the lower tier detectors report to the higher tiers. The statistical preprocessor converts network traffic sample information into a PDF that is compared to a historically developed PDF for corresponding normal network traffic, thus deriving a statistical similarity decision vector that the neural network classifies into anomalous (attack) or normal instance. Several simulation experiments have been carried out focusing on the Denial of Service attack. We used the Perceptron-Backpropagation-Hybrid (PBH) as the neural net classifier, which showed fast convergence (only a few epochs needed) and a small number of weights. The classification results are characterized by low misclassification error rates, for both false positives and false negatives.

Key-Words: Security, Intrusion Detection, Statistical Preprocessing, Neural Network Classification, Perceptron-Backpropagation-Hybrid, PBH, Anomaly Detection

1 Introduction

The basic assumption of intrusion detection is that an intruder’s behavior will be noticeably different from that of legitimate users. Most intrusion detection systems are developed along two complementary trends: misuse detection, and anomaly detection. Misuse detection systems, such as [1][2], search evidence of attacks based on the knowledge accumulated from known attacks and security gaps. Anomaly detection systems, such as [3][4][8], identify intrusions by observing a deviation from normal or expected behavior of the systems or users.

Many technologies have been developed to detect possible attacks. For example, NIDES [3] represents user or system behaviors by a set of statistical variables and detects the deviation between the observed and the standard activities. JAM [2] uses data mining approaches to extract features of attackers and normal users. A system, which identifies intrusions using packet filtering and neural networks, is introduced in [5].

This paper presents the prototype of a hierarchical anomaly network intrusion detection system that uses statistical models and neural networks to detect attacks. Section 2 describes the details of the system architecture, the statistical models and the neural networks used in the system. Section 3 introduces the test bed and the attack schemes we simulated. Some experimental results are also reported in that section. Section 4 draws some conclusions and outlines future work.

2 System Architecture

Our system is a distributed hierarchical application, which consists of several tiers while each tier is composed by several agents. Agents are IDS components that monitor the activities of a host or a network. Different tiers correspond to different network scopes that their agents protect.
For a sample network given in Fig. 1, the intrusion detection system can be divided into 3 tiers. Tier 1 agents monitor system activities of the servers and bridges within a department and periodically generate reports for Tier 2 agents. Tier 2 agents detect the network status of a departmental LAN based on the network traffic that they observe as well as the reports for the Tier 1 agents within the LAN. Tier 3 agents collect data from the Tier 1 agents at the firewall and the router as well as data of Tier 2 agents at the departmental LANs. A system hierarchy is shown in Fig. 2.

Subsequent subsections are organized as follows: subsection 2.1 introduces the structure of Intrusion Detection Agents (IDA); subsection 2.2 describes the statistical model of IDA; and section 2.3 discusses the neural networks used in this system.

### 2.1 Intrusion Detection Agent (IDA)

Because this system is distributed and hierarchical, the IDAs of all tiers have the same structure. A diagram of IDA is illustrated in Fig. 3.

An IDA consists of following components: the probe, the event preprocessor, the statistical model, the neural networks and the post processor. The functionalities of these components are described as below:

**Probe:** collects the network traffic of a host or a network, abstracts the traffic into a set of statistical variables to reflect the network status, and periodically generates reports to the event preprocessor.

**Event Preprocessor:** receives reports from both the probe and IDAs of lower tiers, and converts them into the format required by the statistical model.

**Statistical Model:** maintains a reference model of the normal network activities, compares the reference model with the reports from the event preprocessor, and forms a stimulus vector to feed into the neural networks. We will further discuss the statistical algorithms in subsection 2.2.

**Neural Networks:** analyzes the stimulus vector from the statistical model to decide whether the network traffic is normal or not. Subsection 2.3 will introduce the neural networks used in the system in detail.

**Post Processor:** generates reports for the agents at higher tiers. At the same time, it will display the detected results through a user interface.

### 2.2 Statistical Model

Statistics have been used in anomaly intrusion detection systems [3][4], however most of these systems simply measure the means and the variances of some variables and detect whether certain thresholds are exceeded. SRI’s NIDES [6][3] developed a more sophisticated statistical algorithm by using $\chi^2$-like test to measure the similarity between short-term and long-term profiles. Our current
statistical model uses a similar algorithm as NIDES but with major modifications. Therefore, we will first briefly introduce some basic information of the NIDES statistical algorithm.

In NIDES, user profiles are represented by a number of probability density functions. Let $S$ be the sample space of a random variable and events $E_1, E_2, ..., E_k$ a mutually exclusive partition of $S$. Assume that $p_i$ is the expected probabilities of the occurrence of events $E_i$, and that $p_i'$ represents the actual probability of the occurrences of $E_i$ during a time interval, and that $N$ is the total number of occurrences. NIDES statistical algorithm used $\chi^2$-like test to determine the similarity between the expected and actual distributions with equation as below:

$$Q = N \sum_{i=1}^{k} \frac{(p_i' - p_i)^2}{p_i}$$

When $N$ is large and the events $E_1, E_2, ..., E_k$ are independently, $Q$ approximately follows the $\chi^2$ distribution with $(k - 1)$ degrees of freedom. However in real applications the above two assumptions generally cannot be guaranteed, thus empirically $Q$ may not follows $\chi^2$-distributions. NIDES solved this problem by building an empirical probability distribution for $Q$ which is updated daily in real-time operation.

In our system, since we are using neural networks to identify possible intrusions, we are not so concerned with the actual distribution of $Q$. However, because network traffic is not stationary and network-based attacks may have different time durations, varying from a couple of seconds to several hours, we need an algorithm, which is capable of efficiently monitoring network traffic with different time windows. Based on the above observations, we used a layer-window statistical model, Fig. 4, with each layer-window corresponding to one different detection time slice. The newly arrived events will first be stored in the event buffer of layer 1. The stored events will be compared with the reference model of that layer and the results are fed into neural networks to detect the network status during that time window. The event buffer will be emptied once it becomes full, and the stored events will be averaged and forwarded to the event buffer of layer 2. The same process will be repeated recursively until it arrives at the top level where the events will simply be dropped after processing.

![Fig. 4 Statistical Model](image)

The similarity-measuring algorithm that we are using is shown below:

$$Q = f(N)(\sum_{i=1}^{k} \left| p_i - p_i' \right| + \max_{i=1}^{k} \left| p_i' - p_i \right|)$$

where $f(N)$ is a function that takes into account the total number of occurrences during a time window.

Besides similarity measurements, we also designed an algorithm for the real-time updating of the reference model. Let $\overline{p}_{old}$ be the reference model before updating, $\overline{p}_{new}$ be the reference model after updating, and $\overline{p}_{obs}$ be the observed user activity with a time window. The formula to update the reference model is

$$\overline{p}_{new} = s \times \overline{p}_{obs} + \alpha \times \overline{p}_{old}$$

in which $\alpha$ is the predefined adaptation rate and $s$ is the value generated by the output of the neural network. Assume that the output of the neural network $t$ is a continuous variable between –1 and 1, where –1 means intrusion with absolute certainty and 1 means no intrusion again with complete confidence. In between, the values of $t$ indicate proportional levels of certainty. The function of calculating $s$ is

$$s = \begin{cases} t, & \text{if } t \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

Through the above equations, we ensured that the reference model would be updated actively for normal traffic while kept unchanged when attacks occurred. The attack events will be diverted and stored, for us as attack scripts, in neural network learning.
2.3 Neural Networks

The neural networks are widely considered as an efficient approach to adaptively classify patterns, but the high computation intensity and the long training cycles greatly hindered their applications. In [5][8], BP neural networks were used to detect anomalous user activities. BP networks are excellent in finding out the nonlinear correlations between inputs and outputs, but the large number of hidden neurons makes the architecture computationally inefficient. In our application, we believe both linear and nonlinear correlations exist between the stimulus vectors and the output, therefore we employed a hybrid neural network paradigm [7], called perceptron-backpropagation-hybrid (or PBH) network, which is a superposition of a perceptron and a small backpropagation network. A diagram of the PBH architecture is illustrated in Fig. 5.

![Fig. 5 PBH architecture](image)

In our experiments, we used PBH networks with 4 hidden neurons. As we will see in the next section, the performance of these neural networks was performed very efficiently.

3 Experimental Results

In this section, we will present our simulation approach and the results in applying our statistical models and the PBH neural network to detect network-based attacks. First the testbed configuration and the simulation specifications will be introduced in subsection 3.1, and then subsection 3.2 reports the testing results.

3.1 Testbed

We used a virtual network using simulation tools to generate attack scenarios. The experimental testbed that we built using OPNET, a powerful network simulation facility, is shown in Fig. 6. The testbed consists of 3 10BaseX LANs, interconnected by 2 routers.

![Fig. 6 Simulation Testbed](image)

We simulated the udp flooding attack within the testbed. To extensively test our system, we ran two independent scenarios with different traffic loads and characteristics. Table 1 listed the traffic loads of the two simulation scenarios.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Scenario 1</th>
<th>Scenario 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP background traffic (Mbps)</td>
<td>1.05</td>
<td>1.05</td>
</tr>
<tr>
<td>UDP background traffic (Mbps)</td>
<td>1.08</td>
<td>6.82</td>
</tr>
<tr>
<td>Attack traffic (Mbps)</td>
<td>1.8</td>
<td>1.8</td>
</tr>
</tbody>
</table>

Table 1 Traffic Loads of Tow Simulation Scenarios

3.2 Results

For each simulation scenario, we collected 10,000 records of networks traffic. We evenly divided these data into two separate sets, one for training and the other for testing. In each scenario, the system was trained for 150 epochs.

The mean squared root errors of the outputs of the two scenarios are shown in Fig. 7 and Fig. 8. From the graphs, we can see that the MSR errors of both
scenarios decrease very fast after only the first few epochs, reaching satisfactory convergence levels within the first ten epochs or so. As the training continues, the MSR errors of Scenario 1 and Scenario 2 approach to 0.005 and 0.015 respectively.

The misclassification probabilities of the outputs of the two scenarios are shown in Fig. 9 and Fig. 10, in which we calculated the false-positive possibilities, i.e., the probabilities of classifying normal traffic as intrusion, and the false-negative probabilities, that is the probabilities of failing to identify intrusion, as well as the overall misclassification probabilities, which are the sum of both false-positive and false-negative probabilities. These graphs show similar trends as Figs. 7 and 8 for MSR.

From the simulation results, we can see that, in both scenarios, the system converged very fast, within several epochs, with high accuracy. Theses features are especially desirable for network intrusion detection systems, which need real-timely monitoring and online training.

4 Conclusions

In this paper, we described our design of a hierarchical network intrusion detection system. We discussed the system hierarchy, the statistical preprocessing modules and the neural network classifier. We then discussed the simulation experiments that we carried out for the Denial of Service attack script. The results showed fast convergence for the neural net classifier and low misclassification rates. Thus, these experiments showed that the proposed approach is effective and bears much promise.
Acknowledgements
Our research was partially supported by a Phase I SBIR contract with US Army.

We would also like to thank OPNET Technologies, Inc.™, for providing the OPNET simulation software.

Reference: