A Neural Network Approach To Intrusion Detection

Mary Krajnak
Contents

Introduction
   1. Problem Statement & Motivation
   2. Other Approaches
   3. This Projects Approach

Work Performed
   1. Data Retrieval
   2. Data Analysis & Encoding
   3. Preprocessing
   4. Network Configuration

Results

Discussion

References
Introduction

Motivation

Network security has become increasingly important as computers become increasing integrated parts of our lives. Many companies, governments, and private homes rely on secure connections over web services. Computer hackers pose a significant threat to our daily lives.

Problem Statement

In order to protect networks from misuse an intrusion detection system (IDS) is often put in place. Methods requiring hand-coded rules to determine which packets are legitimate and which packets get dropped are difficult to create. They are also unreliable as the methods of attack change with time. A system which can identify network attacks without having to hard-code rules is desirable.

Other Approaches

The basic schemes for flexibly detecting malicious network traffic is known as "anomaly intrusion detection". It classifies attacks by setting some baseline for normal behavior and classifying all other abnormal traffic as malicious. There have been many different approaches to implementing this type of IDS.

Some systems implement a packet analyzer which inspects each packet, rejecting ones which do not comply with some predetermined standard. Kuo-Chen Lee and his associates implemented an packet analyzer IDS which inspected packet headers with a Bayesian approach. Their model led to over 99% accuracy in detection, as well as a .03% false positive rate [1].

Other systems use statistical analysis to attempt to model "normal" behavior of a network. Bahrololum and Khaleghi of the Iran Telecommunication Research Center modeled behavior using Hierarchical Gaussian Mixture Model, which accurately classified about 95% of attacks [2].

This Projects Approach

This paper presents a Neural Network approach to classifying attacks. It analyzes tcpdump data from network traffic to determine trends. The aim is to determine if a neural network would be an appropriate tool for detecting intrusions. If successful, it would provide anomaly detection without complex means for determining a behavior model.
**Work Performed**

**Data Retrieval**

The US Defense Advanced Research Projects Agency (DARPA) simulated two months of network traffic and attacks. DARPA then released the data set to the public to encourage performance improvements of the then current IDS's. This paper uses the 1998 DARPA Intrusion Detection Evaluation Data Sets.

**Data Analysis & Encoding**

<table>
<thead>
<tr>
<th>Samples</th>
<th>&gt; 1,000,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Features</td>
<td>8</td>
</tr>
<tr>
<td>Feature Types</td>
<td>session date, start time, duration of connection, service used by session, source port, destination port, source IP, destination IP</td>
</tr>
</tbody>
</table>

Along with the eight features of each connection, a score is assigned to each session to indicate if it is an attack and if so the name of the attack.

Once the data was retrieved it had to be encoded into numeric data for use in the neural network. The table below describes how each feature was encoded.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Original format</th>
<th>Encoded format</th>
</tr>
</thead>
<tbody>
<tr>
<td>session date</td>
<td>m:int + '/' + d:int + '/' + y:int</td>
<td>mdy:int</td>
</tr>
<tr>
<td>start time, duration</td>
<td>h:int + ':' + m:int + ':' + s:int</td>
<td>i = h<em>3600 + m</em>60 + s</td>
</tr>
<tr>
<td>service name</td>
<td>array of characters</td>
<td>int of each character's ascii value added together</td>
</tr>
<tr>
<td>source port, destination port</td>
<td>x:int</td>
<td>x:int</td>
</tr>
<tr>
<td>source IP, destination IP</td>
<td>w:int + '.' x:int + '.' + y:int + '.' + z:int</td>
<td>wxyz:int</td>
</tr>
</tbody>
</table>

**Data Preprocessing**

Since the features are encoded using different methods the input data was preprocessed to prevent any large feature value from dominating. The data was scaled into range [-1, 1]. Of the eight features available this project used duration, name, source port, destination port, source IP, and destination IP. The choices of the features used was determined empirically. Initially the IP addresses were not included, but they produced better classification.

**Testing and Training Data**

Since there is an abundance of data samples it was not necessary to use techniques to artificially increase the amount of data. Instead, 70% of the collected data was used for training and 30% was used for testing.

**Network Configuration**

This project used a back propagation neural network, as the tcp data is most likely not linearly separable. Initially, six different network configurations were tested: 3-layer network with 2, 4, or 5 hidden neurons and a 4-layer network with 2, 4, or 5 hidden neurons. There was no significant
difference found in any of the test results so a 3-layer, 4 hidden neuron configuration was selected for the final tests.

Results

The neural network was very unsuccessful at classifying all types of attacks at the same time. The data is composed of several different categories of attacks: Denial of Service, Root Attacks, Remote to User Attacks, and Probing. When attempting to identify all types of attacks with one network it achieved a 39.3% classification rate, with a 65.5% false positive rate.

However, when each attack was considered individually the classification rate greatly improved to around 99% for others. Table 1 contains the classification rate for some of the tests.

<table>
<thead>
<tr>
<th>Attack Description</th>
<th>Classification Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>all attacks</td>
<td>39.30%</td>
</tr>
<tr>
<td>smurf attacks (DoS)</td>
<td>99.90%</td>
</tr>
<tr>
<td>back (DoS)</td>
<td>95.50%</td>
</tr>
<tr>
<td>satan (probe)</td>
<td>99.30%</td>
</tr>
</tbody>
</table>

Graph 1

Classification of 'smurf' attacks.
Discussion

Using neural networks for intrusion detection seems to have promise, at least when each attack is considered individually. One could build a clustering algorithm and use the results of the backpropagation neural network to create a machine to classify all types of attacks. The classification rate of the tests compares well to the other methods of intrusion detection.

One item of concern is the false positives. Many of the individual tests produced 3-5% false positive rates. As one could imagine, an ISP provider would not be keen on using a system which slows the flow of legitimate traffic. However, a company with a great need for security might be willing to be more conservative with their traffic flow.

References


