CPSC097 Project Proposal: Network Intrusion Detection Using Random Forests And Expectation Maximization Preprocessing

Chris Magnano
`cmagnan1` and Chris Lekas
`clekas1`

1 Abstract

Despite recent advanced in network intrusion detection algorithms, most network intrusion detection systems still struggle to detect novel attack types. We propose improving upon an already high-performing machine learning algorithm, random forests, with preprocessing step that uses expectation maximization. We will test our hybrid algorithm on a field standard dataset in cases with both known and unknown attacks.

2 Motivation

Network attacks are increasingly common. Although current network intrusion detection algorithms are quite successful in defending against known attacks, current methods struggle to identify new forms of attacks. We hope to use preprocessing combined with current high-performing machine learning methods to improve detectiion of both known and unknown attacks.

2.1 Problem Statment

Input: Training packet data labeled normal or with some type of network attack and unlabeled test packet data.

Output: Labels for unlabeled test packets as an attack or normal traffic.

3 Background

Network intrusion detection is based on the assumption that there is enough data availalbe in packet metadata and network traffic patterns to determine whether a given packet it part of an attack. There are three main types of algorithms used to implement network intrusion detection: statistical-based, knowledge-based, and machine learning-based. (García-Teodoro et al., 2009)

Machine learning is a natural method for detecting new patterns of attacks. Random forests are among the most successful machine learning algorithms and have proven effective in a variety of situations in which this type of pattern detection is important. Several characteristics of network packets involve continuous variables, which are best classified based on ranges instead of exact values. Expectation maximization(EM) is a good method for determining important ranges to use as features in a machine learning algorithm.

4 Project

4.1 Overview

We hope to contruct a pipeline which will be able to, given incoming network traffic, alert the user if that incoming traffic is some kind of attempted malicious action. We are planning on performing this prediction of malicous intent using random forests with an expectaion maximizatiom(EM) preprocessing step. We plan for our pipeline to be able to dectect known malicous attacks as well as novel attacks.

4.2 Algorithm

Our pipline will begin with love, as all good piplines should. Following love, the pipline will be trained using labeled packet data from the sources outlined in the data section of this paper. The raw data will be preprocessed using EM. EM will, for each feature, be used to determine relevant ranges for different types of traffic. These ranges will be used as bin features for a typical random forests implementation. After training, we will test our data on unlabeled data, including types of attacks not in the training data set.

4.2.1 Random Forests

We will begin by using a machine learning library, likely either a Java or Python library, for our initial tests. If we find off the shelf library implementations to be too slow or otherwise inadaquate, we will implement random forests outselves.We may have to also using a tuning set to determine a maximum tree depth in order to avoid overfitting.

4.2.2 Expectation Maximization

We are planning to implement EM from stratch. We believe that preprocessing with EM will allow the pipline to establish ”normal” ranges for features of normal traffic. For continuous features, attacks of similar nature will likely have similar, but slightly different, values. Establishing these ranges will aid in detecting novel attack types.

4.3 Data

We will use the KDD-99 dataset. These data are old and may have some problems as a result of their age, but they are still commonly used (KDD-99 is probably still the most used dataset for network intrusion detection). We will limit ourselves to ten percent of the KDD-99 dataset (approximately 75MB) or less for most of our training and testing, but we might train and/or test on a larger portion near the end of our project if we can find the necessary time.

There are several reasons for using KDD-99 instead of another dataset, which are discussed in the following subsubsections.

4.3.1 Feasibility Of Data Obtainment

Obtaining network intrusion data on our own would require isolating a computer and carefully controlling its connections, while feeding in a large quantity of dangerous packets. This would require preparing the attacks on our own, which would have the dual disadvantages of (1) being extremely time consuming without being extremely beneficial and (2) being likely to not include crucial attacks.

4.3.2 Minimal Parsing Required

Parsing packets of many different protocols in order to obtain the necessary data in order to perform network intrusion detection would be a very time consuming process. KDD-99 provides data in a very easy format to parse, allowing us to spend more time on the core of our project.

4.3.3 Comparability With Previous Results

Since many papers on network intrusion detection have used KDD-99, using KDD-99 allows our results to be compared with those of the other papers. Although we may not be using exactly the same pieces of the dataset, at least we can count on the data being similar. This is important because otherwise we will not be able to tell whether our random forest with EM support algorithm is more effective than, less effective than, or just as effective as previously implemented algorithms.

5 Milestone Goals

5.1 Milestone 1

By milestone 1, we will have chosen the random forest library implementation that we intend to use. Additionally, the EM algorithm will have been implemented. This means that we will be able to give it a list of observations for a given KDD-99 feature and it will return a list of relevant ranges for the observations given.

Also, code will be in place to parse the KDD-99 input data and call the EM function on relevant sections of the data. Finally, we will have set up version control for our project.

5.2 Milestone 2

By milestone 2, our implementation will be able to take in raw KDD-99 data, parse it into features usin EM, train using the chosen random forests implementation, and test on unlabeled data. We will also have code written to evaluate our results based on metrics used in other papers.

5.3 Milestone 3

The goals for milestone 3 depend on the results obtained from the implementation completed by milestone 2.

If it is determined that the library random forest algorithm is inadequate in some way (e.g. is not fast enough to run in real-time on incoming network traffic), we may implement our own random forests algorithm.

Alternatively, we might optimize our features and/or library random forest algorithm usage, try expectation maximization with a genetic algorithm, and/or hybridize our random forests algorithm with a genetic algorithm in order to boost the accuracy of our pipeline.

References

P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez. 2009. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers And Security*, 28:18–28.