Global Information Assurance Certification Paper

Using Decision Tree Analysis for Intrusion Detection: A How-To Guide

GIAC (GCIA) Gold Certification

Author: Jeff Markey, Markey.Jeff@gmail.com
Advisor: Dr. Antonios Atlasis

Accepted: June 5, 2011

Abstract

As cyber threats grow in sophistication, network defenders need to use every tool in the defensive arsenal to protect their networks. Data mining techniques, such as decision tree analysis, offer a semi-automated approach to detect adversary threats. This paper presents a repeatable process to implement the decision tree technique on a small set of network data. Using this process, a security team can gather data, build a decision tree model, and incorporate the model's logic into Snort signatures, firewall rules, and custom-built detection scripts. The process presented in this paper can serve as a preliminary test to determine the value of data mining techniques before deciding whether or not to incorporate the techniques across the enterprise. Alternatively, the proposed methodology can be used to implement ad-hoc decision tree analysis as the security data is available. Either approach allows corporations or security teams to quickly, easily, and inexpensively implement decision tree analysis and gain unique security insights based on the corporation's network data.

1. Introduction

As the volume and sophistication of computer network attacks increase, it becomes increasingly difficult to detect and counter intrusions into a network of interest. Advanced Persistent Threats (APTs) have become a larger threat to both corporations and nation-states (Rachwald, 2010), cyber criminals continue to use more sophisticated techniques to illegally gain access to systems (TrendMicro Threat Report, 2010), and companies and employees are exponentially adopting new and more sophisticated, networked technologies in the workplace. All of these factors significantly hamper the task of defending a network and the trends indicate that these difficulties are likely to increase over time.

Combating these challenges requires a variety of tools and techniques to detect and defend against attacks. Decision trees, a technique from the field of data mining, can assist with this task. Decision trees provide unique insights into the problem of identifying malicious activity and can assist in the creation of technology-specific techniques to defend against attacks.

Historically, data mining techniques have not been adopted in the IT security community. This lack of adoption has been for a number of reasons including the difficulty obtaining the necessary data and a lack of awareness about the techniques. As network security experts work with network traffic, the barriers to gathering the data have been gradually minimized over time. New tools exist that support the functionality necessary to process network data for the purposes of data mining, such as free and open source software packages, commercial tools, and easy-to-use scripting languages.

In academia, many papers have been written on the subject of data mining for intrusion detection; from building decision trees with honeypot data (Grégio, Santos, & Montes, 2007) to classifying threats in real-time (Sangkatsanee, Wattanapongsakorn, & Charnsripinyo, 2009), these documents have provided important, rigorous results regarding data mining for intrusion detection. However, academic research is tailored to

an audience that is highly trained in data mining algorithms, techniques, and terminology. Academic documents can gloss over important details for readers unfamiliar with the techniques. This paper intends to provide a detailed explanation of the required inputs, outputs, and steps necessary to implement the techniques and apply them to real-world security challenges facing corporations.

One challenge to the development of end-to-end instructions for data mining is the problem of scale and scope of the how-to guide. Instructions can be written at a management level – for a large company to create and fund robust data mining support to intrusion detection. Alternatively, the instructions can be written at a technical level – for experienced network experts to implement smaller-scale data mining techniques for the purposes of intrusion detection.

This paper is scoped to address the technical challenges of implementing smaller scale data mining analysis. Specifically, it focuses on applying decision tree algorithms to intrusion detection.

2. Decision Tree Background

Understanding the basics of decision tree analysis provides the foundation necessary to apply this technique to intrusion detection. When looking at applying decision tree analysis to intrusion detection, a number of questions come to mind, namely: What are decision trees, how can they help in intrusion detection, and how is this technique different from existing techniques?

2.1. What are Decision Trees?

A decision tree is defined as "a predictive modeling technique from the fields of machine learning and statistics that builds a simple tree-like structure to model the underlying pattern [of data]" (PredictionWorks, 2011). Decision trees are one example of a classification algorithm. Classification is a data mining technique that assigns objects to one of several predefined categories (Tan, Steinbach, & Kumar, 2005).

Classification algorithms (also called classifiers) have helped solve problems ranging from credit card theft detection (Chan, Fan, Prodromidis, & Stolfo, 1999) to diagnosing patients with heart problems (Ganesan, 2005) by recognizing distinctive patterns in a dataset and classifying activity based on this information. From an intrusion detection perspective, classification algorithms can characterize network data as malicious, benign, scanning, or any other category of interest using information like source/destination ports, IP addresses, and the number of bytes sent during a connection.

Figure 1 presents a simple example decision tree based on a single decision rule for a known malicious port 8787:

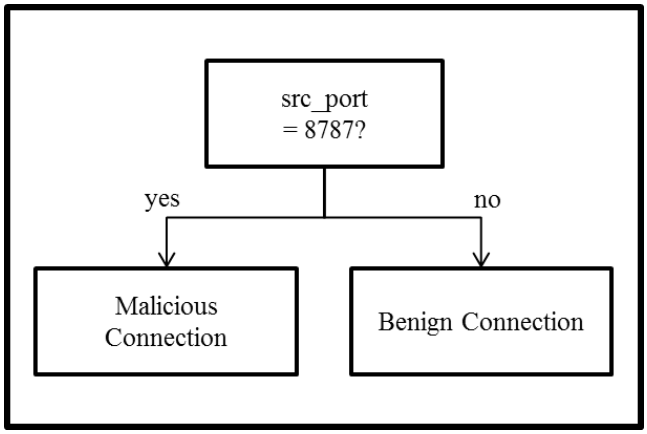

Figure 1: Decision Tree Example

Classification algorithms create a decision tree like the one presented in Figure 1, by identifying patterns in an existing dataset and using that information to create the tree. The algorithms take pre-classified data as input. They learn the patterns in the data and create simple rules to differentiate between the various types of data in the pre-classified data set.

To better illustrate the classification process, consider our previous decision tree example as a model. Figure 2 presents a simplistic example of the process necessary to build this tree:

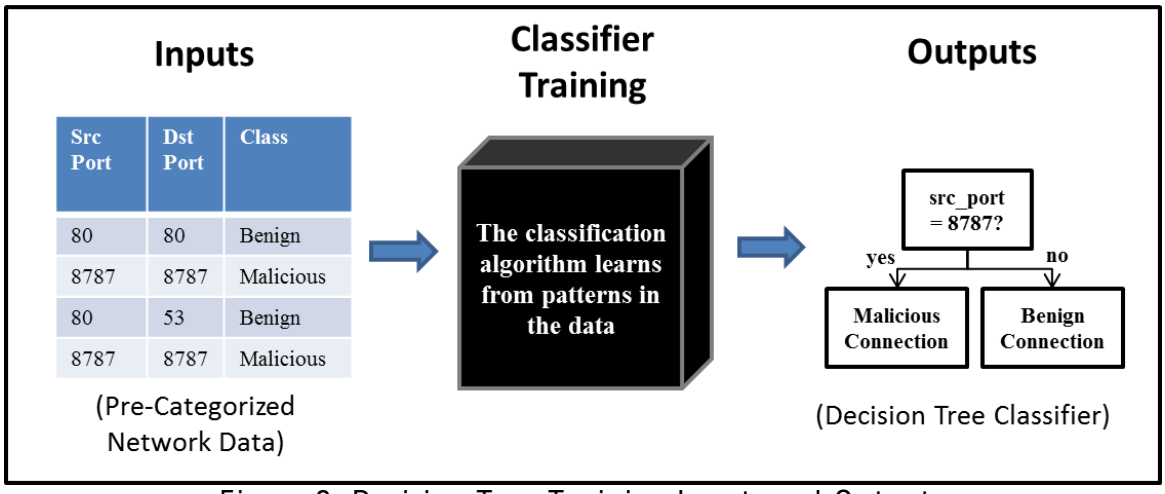

Figure 2: Decision Tree Training Inputs and Outputs

The inputs in Figure 2 are often called features of the data. These features are usually important, quantifiable characteristics of the data (e.g. source ports from a set of network connections). The features can be discrete (e.g. descriptive, non-numerical data elements) or continuous (e.g. numerical data elements). Typically, these features are stored in some form of table, such as a spreadsheet, a database, or another tabular file format (e.g. ARFF (Attribute-Relation File Format) files for the popular data mining software package Weka).

Decision trees are created using algorithms to build the tree iteratively in a short period of time. Creating an optimal tree is usually computationally infeasible, as the number of possible trees grows exponentially with the set of features. Many greedy algorithms, such as Hunt's algorithm, are based on minimizing the entropy associated with different rule sets recursively to build a quick, effective, but sub-optimal tree (Tan et al, 2005). Hunt's algorithm can be combined with other techniques to create different decision tree classifiers. Table 1 presents a list of 3 decision tree classifiers and their pros and cons:

Decision Tree Classifiers	Description	Advantages	Disadvantages
Best Frist Tree	Creates Haijian Shi's best-first decision tree (Shi, 2006).	Creates simpler, less complicated trees for some datasets	A newer technique evaluated for a subset of decision tree situations
C4.5 Tree	Implements Hunt's algorithm to create a decision tree. This classifier was developed by Ross Quinlan (Quinlan, 1993)	Classifies data with missing attributes	Slower to classify than other techniques
ID3	A precursor to the C4.5 tree. (Breiman, 1993)	Extensively tested and used across many different applications	Unable to handle both discrete and continuous variables

Table 1: Decision Tree Algorithm Comparison

In academia, there is still some debate about which decision tree algorithm is the best. The consensus seems to be that there is no one right answer to this question. Different decision trees perform better in different situations and their relative performance can be unpredictable across multiple datasets with different features. Most data mining packages automatically implement a variety of decision tree algorithms. The optimized configuration and a comparison of the effectiveness of these algorithms is beyond the scopes of the paper and most experiments will focus on how useful the resulting model is in practice.

It is important to differentiate decision trees from other real-time tools that protect networks of interest. Decision trees are tools for analyzing data and identifying significant characteristics in network data that indicate malicious activities. Decision trees can help teams determine which IDS signatures to write, which firewall rules to implement, and what type of network activity to flag for further analysis. However, decision trees alone do not take action to stop threat, like firewalls and Intrusion Prevention Systems (IPS). Their decision logic can be used in conjunction with other real-time tools to take corrective action against cyber threats by highlighting what the malicious activity looks like.

Figure 3 provides a depiction of the classification logic and real-time tools categorizing new data – using the rules generated from the learning algorithm. As an input, new network data, with no known classification is provided. Using the decision rules identified, the classification logic is run against this new set of data. As an output, each data element is categorized and corrective actions are taken based on the category.

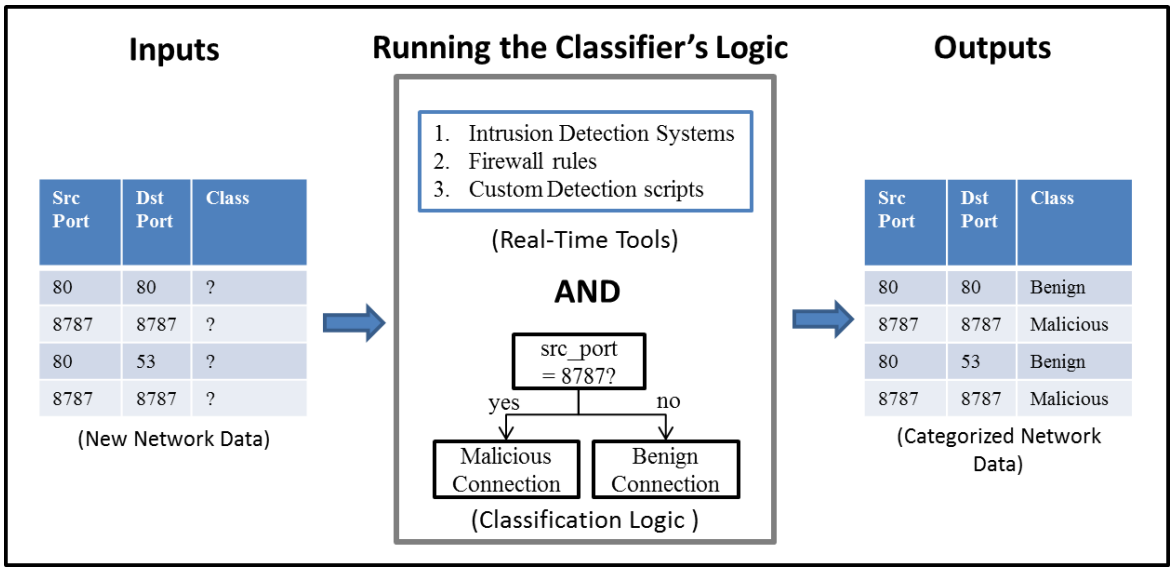

Figure 3: Decision Tree Classification Inputs and Outputs

In summary, decision trees provide a simple set of rules that can categorize new data. Creating decision trees requires a pre-classified dataset in order for the algorithms to learn patterns in the data. This training dataset is made up of features which are quantifiable characteristics of the data. When the decision tree is built from these features, the rules for characterizing information can be used to identify and classify new data of interest by incorporating the logic into existing defenses like IDS, firewalls, custom-built detection scripts, or classification software.

2.2. Why Use Decision Trees?

When considering decision trees, a natural question to ask is, "how can this technique support the day-to-day tasks involved in securing a network?" Many tools already exist to support IT security. Is another technique really needed to aid in intrusion

detection? The case for implementing the decision tree technique in an organization is not an obvious one.

In order for decision trees to be a viable tool in the intrusion detection toolkit, the technique needs to satisfy a minimum set of requirements. The technique needs to be beneficial to the intrusion analysis mission and produce real results for an organization. In addition, decision trees must be unique among existing tools. If other tools exist with the same functionality provided by decision trees, then decision trees may be redundant and unnecessary.

2.2.1. How can Decision Trees aid in Intrusion Detection?

Decision trees and other classification algorithms have been tested and applied to many practical intrusion detection challenges in academia. Some researchers have successfully created decision tree classifiers using data from honeypots (i.e. decoy systems put on a network as bait for attackers) and data from normal network activities. This analysis leads to the creation of decision rules to identify malicious activity (Grégio et al., 2007). Other classification techniques have also been used to classify and characterize scanning activity as well as to detect novel attacks from cyber adversaries (Simon, Xiong, Eilertson, & Kumar, 2006). In the corporate world, researchers have also used these techniques to prioritize IDS alarms and determine their root causes (Julisch, 2003). For more information on academic research in the field of data mining, Terry Bruger's paper "Data Mining Methods for Network Intrusion Detection" is an excellent source to read about existing data mining research efforts and their application to intrusion detection (Brugger, 2004).

Summarizing the benefits of decision tree analysis to cyber security, decision trees can help perform the following functions for an organization:

- Supplement honeypot analysis by learning from adversary trends and creating rules to detect malicious activity.
- Supplement penetration testing efforts by learning from the pen tester's

actions and creating rules to detect their tactics, techniques, and procedures.

- Identify and highlight malicious traffic (see example in Appendix A).
- Prioritize alerting by identifying and tagging low-priority alerts.
- Identify and characterize known scanning activity (see example in Appendix B).
- Supplement an incident response team's recovery and monitoring process by flagging a repeated intrusion attempt early.
- Detect previously unknown network anomalies.
- Support IDS signature development by providing a set of rules to identify malicious activity (see example in Appendix A).

After decision trees are built, they have the potential to reduce the amount of data required for analysis, help identify anomalous malicious activity, and provide analytic insight into the differences between malicious and benign network traffic. In addition, as the amount of data handled by IT security experts increases, flagging the priority of malicious activity in an automated fashion becomes more and more beneficial for the mission.

2.2.2. How are Decision Trees Unique?

Of all the tools in the intrusion detection toolkit, decision trees share the most similarities with an Intrusion Detection System (IDS). As listed in the previous section, a common goal for a decision tree is to identify the characteristics of malicious activity. This goal is very similar to that of an IDS. If the end goal of decision trees is identical to a tool that is already in use, why should decision trees be considered?

Despite the similarities between the tools, decision trees provide a set of unique methods that can add value many different real-time systems, including, but not limited

to an IDS. The technique does not replace an IDS, but can supplement the system. Decision trees are an automated technique that can assist with the analysis of large sets of intrusion detection data. The technique is designed to answer questions like: "what rules can we use to distinguish normal from malicious traffic?" and "what are the common characteristics of scanning activity when compared to other traffic?"

In short, the technique can help a team come up with intrusion detection solutions by providing a model that defines and differentiates malicious traffic from normal traffic. Decision trees can help a team by identifying some of the differences in an automated or semi-automated fashion. While decision trees cannot do the thinking for an experienced team of security experts, the technique can identify trends and patterns that may warrant further investigation, signature development, and other monitoring actions. After the decision tree identifies the rules to differentiate malicious from benign traffic, other systems like IDS, firewalls, and other detection technologies can flag information identified by the decision tree. If firewalls, IDS, and IPS are the guns of an organization's defenses that take action and actively stop network threats, then decision trees are the scopes on the guns that help point them in the right direction.

This analytic support can be especially important as the complexity of attacks increases. As the numbers of variables involved in attacks grow and the volume of this data expands, automated techniques, like decision trees, are better able to identify patterns and the techniques become more beneficial. Conversely, as the analysis of the data becomes easier for the machines, it can become more time-intensive for people. This makes decision trees a potential time saver for IT security teams.

Another advantage of decision trees is that IT team members with less experience handling and analyzing network traffic can still implement the decision tree technique and gain insight from the analysis. In fact, for organizations and teams considering the adoption of decision trees, one litmus test for the methodology may be for a newer team member or an intern to spend a month using the technique on available data to determine the benefit of decision trees to the security posture of the organization.

Decision trees are only one example of a classification algorithm. Many other classification algorithms exist and have been applied to intrusion detection. Other classification algorithms, such as Bayesian-based classifiers, Artificial Neural Networks, and rule-based classifiers can also be used to accomplish the same goals as decision trees. The models and their decision rules look different, but they share the same inputs and outputs.

The main advantage of decision trees over many other classification techniques is that they produce a set of rules that are transparent, easy to understand, and easily incorporated into real-time technologies like IDS and firewalls. The process to implement decision trees discussed in this paper is general enough to apply to any classification technique. An organization may want to experiment with other classification techniques to determine their accuracy and the value of their resulting models; specifically the rule-based classification algorithms are another option that is pertinent to intrusion detection analysis. Rule-based classification algorithms share many of the advantages of decision trees in that their models are easily understood and can be incorporated into other real-time technologies. Like decision trees, researchers have also used this technique to identify intrusions in network traffic (Lee & Stolfo, 1998).

Appendix B provides instructions to implement the rule-based classifier for intrusion detection to differentiate scanning, malicious, and benign traffic.

3. Using Decision Trees for Intrusion Detection

Creating and using decision trees to gain insight for intrusion detection can be accomplished using a straightforward process. This process can take as little as hours or as long as months of labor, depending on the clarity of the goals, the scope of the effort, the availability of the data, and the pre-processing challenges associated with the data. Two prerequisites for the analysis are data collection (i.e. identifying and collecting data of interest) and tool acquisition and selection (i.e. identifying and deploying data mining tools). The gathered data requires a pre-processing phase to move it into the form

necessary for decision tree algorithms. After the data is processed, decision trees can be trained using the processed data and tools. Running and analyzing the result of this data is an important next step to understand the resulting model and its rule sets. The final step is using the results of the analysis to run the decision rules in real-time.

Figure 4 presents a summary of the process to implement decision trees for intrusion detection.

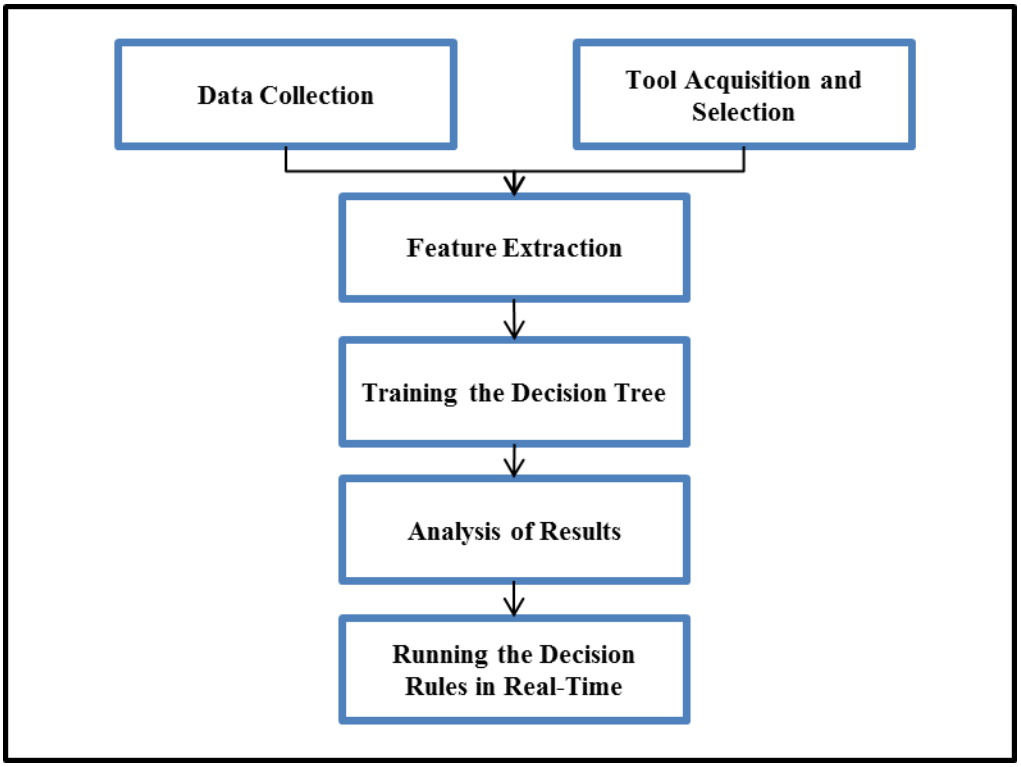

Figure 4: Process to implement decision trees for intrusion detection

3.1. Data Collection

Choosing an appropriate dataset for performing decision tree analysis can be a difficult task. Data collection and pre-processing often consume the majority of the time spent during most data mining efforts. The approach to collecting data should be feasible by a small team, permissible based on the IT policy of the organization, and compliant with all applicable laws. The data collected should also be relevant to challenges faced

by an intrusion detection team. When implementing decision trees, capturing datasets for each category the decision tree needs to learn is critical. For example, a decision tree approach that differentiates between benign data and malicious data requires both a known malicious dataset and a known benign dataset to train the algorithm. Finding and capturing this pre-classified data is not trivial; the ability to collect and store this information may be the biggest hurdle to implementing this technique.

Given this challenge, an important consideration is where to collect the data necessary to train the decision tree.

- Honeypots serve as one potential source of malicious data, while captures from normal network activity represent a source of primarily known benign data. Using both of these sets of data, an organization might be able to gain more insight from an active honeypot.
- Flat files from known historical intrusion attempts may also serve as an example of malicious data when compared to a normal network dataset.
- During the incident handling process, active exploits against a network could also be gathered as data for a decision tree model. Identifying and gathering this information could support the recovery phase of the incident handling process to ensure that relevant signatures can capture and alert on additional intrusion attempts.
- During a penetration test, the penetration tester's activity on the traffic can be captured and compared against normal network traffic to find anomalies and build decision rules based on the approaches used by the tester.
- Lastly, publicly available packet captures from sites such as www.openpacket.org can also provide known malicious training data for a model.

The type of data is also important to consider. When working to gain insight for the purposes of intrusion detection, many different data sources can be analyzed

including: pcap files, network alerts, or log files. Table 2 provides a summary of some of the advantages and disadvantages of using specific types of data.

Data Source	Advantages	Disadvantages
Pcap files	Ubiquitous	Large, requires feature extraction tools (e.g. tcptrace) for classification
Connection log files	There are many academic examples to draw from for this data	Not always available
IDS alerts	Well-structured data	May capture relatively few features for algorithms to learn from
Firewall log files	Well-structured data	May require some parsing for intrusion detection

Table 2: Possible Data Sources for data mining purposes

3.2. Tool Acquisition and Selection

Implementing decision trees can require a variety of different tools. Tool categories may include feature extraction tools, data mining analysis tools, and databases. Feature extraction tools are used during the data pre-processing phase to collect and structure the features from a dataset in a format that can be used for training the decision tree. Appendix A presents detailed instructions about how to use the free and open source tcptrace tool to extract features from pcap files. Wireshark's command line utility, tshark, is another tool that can perform feature extraction from pcap files.

Data mining analysis tools are also necessary to implement a decision tree project. Choosing a data mining analysis tool can be a challenge as tools differ based on their ease of use, cost, and the number of data mining algorithms supported. Popular open source data mining packages include Weka, R, Tanagra, YALE, and KNIME. Blaz Zupan and Janez Demsar's survey "Open-Source Tools for Data Mining" (2008) provides more information comparing the different open-source options available. Popular commercial tools for data mining include SAS, SPSS, and Matlab.

Of all the open-source tools, Weka has been described as "perhaps the best-known open-source machine learning and data mining environment" (Zupan & Demsar, 2008). The free and open source software package Weka has been used for many data

mining efforts and can serve as a good tool for a preliminary test of the decision tree technique on network data. Weka is easy to use, provides an extensive list of machine learning algorithms, and is extensible through the Java programming language. Appendix A and B present detailed instructions about how-to use Weka, and Figure 5 presents a screenshot of the tool.

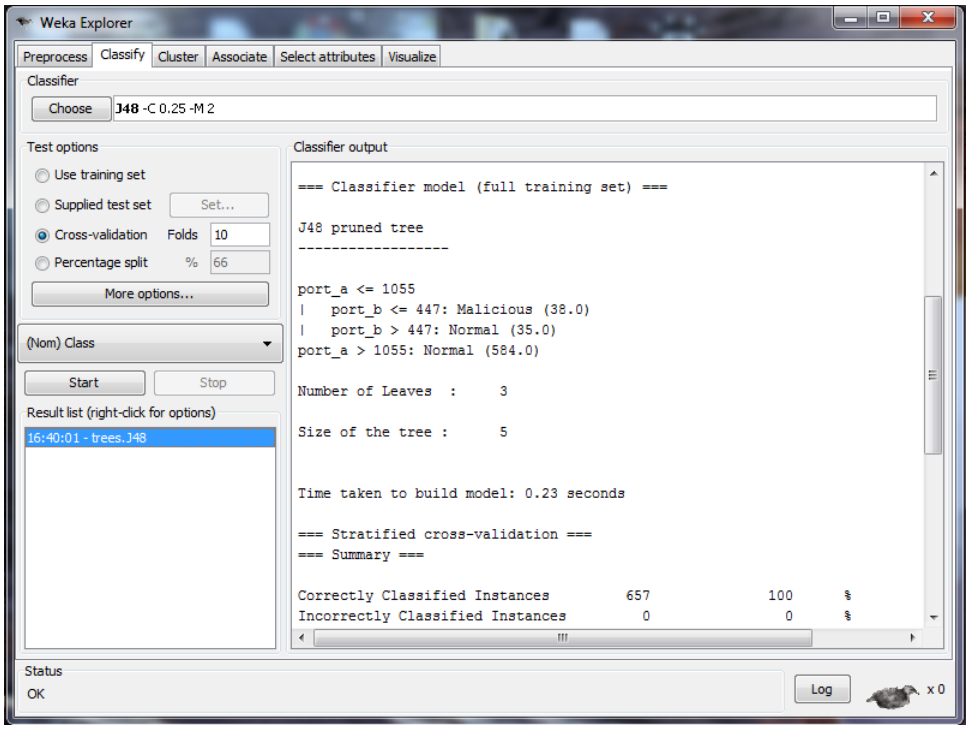

Figure 5: Weka screenshot

For larger efforts, databases to store data may be required to support decision tree analysis. These databases include options like SQL, Oracle, and MySQL. For smaller efforts, tools like Microsoft Excel or OpenOffice's Calc package are sufficient.

3.3. Feature Extraction

Extracting features from a given dataset presents many challenges for intrusion detection. Recall that features are important, quantifiable characteristics of the data used by classification algorithms to identify patterns. One challenge is to determine which features should be extracted from network data; the next challenge is to extract and store

the features.

Of the academic studies performed on applying classification techniques to intrusion detection, there are a number of common features that are used across different sources. The following table is a compilation derived from Terry Brugger's survey paper titled "Data Mining Methods for Network Intrusion Detection" about the features used in 22 academic research papers applying data mining techniques to intrusion detection (Brugger, 2004). Table 3 presents a summary of his survey regarding the use of different network features by researchers.

Feature	Number of uses in 22 academic papers
Destination Port	20
TCP Flags	17
Source IP	17
Destination IP	17
Source Port	16
Duration	14
Source Bytes	13
Timestamp	12
Protocol	11
Destination Bytes	11

Table 3: Possible Data Sources for data mining purposes

For more of the details regarding which academic paper used which features, please refer to Terry Brugger's document (Brugger, 2004).

The features used for intrusion detection analysis may depend on the end-goal of the organization implementing the study. For example, if honeypot data is the underlying source of data, the IP address features should probably be removed from the dataset. Decision trees may determine that the honeypot's IP address is a key indicator of known malicious activity, but this finding will not help an organization understand the root cause of the adversary's tactics against this IP address. Also, if the analysis is intended to aid

in the creation of new signatures for an intrusion detection system, then relying on features that can be mitigated with these tools is a sensible approach.

After selecting the features to gather, the next task is to extract the features into a format that is compatible with the selected data mining tools. Depending on the source data for the analysis – this extraction will likely require specialized tools. These tools may be custom-built for the extraction, free and open source tools, or commercial extractors. After moving the data into a compatible format, the next step is to build a decision tree from the data.

3.4. Training the Decision Tree

The data mining tools identified in the tool selection phase are critical to creating a decision tree. The software packages offer either GUIs or command-line interfaces to perform the analysis. Recall that though there are many different algorithms that can create a decision tree, all of them share the same inputs and outputs. For the purposes of applying these techniques to intrusion detection, the algorithms themselves can be seen as a black box that provides a technique to differentiate and categorize a set of data.

With many available data mining software packages, implementing different algorithms is only a couple of lines of code or mouse-clicks away. This allows for quick easy testing of many different algorithms. The important thing for an organization is to ensure the effectiveness of the resulting model built from the algorithms. Appendix A presents an end-to-end example of the training process using the Weka package.

3.5. Analysis of Results

After the decision tree is created, the resulting model must be analyzed. The accuracy of the model and the insights gained from the resulting tree are important to consider. Model accuracy is usually straightforward to measure; techniques such as k-fold cross validation can test the model's accuracy in a meaningful way (Tan et al., 2005).

Cross validation calculates the accuracy of the model by separating the data into two different populations: a training set and a testing set. The decision tree model is created from the training set and its accuracy is measured based on how well it classifies the testing set. This testing process is continued k times to complete the k-fold cross validation procedure.

As an example of k-fold cross validation, consider that k=2, meaning that we have 2-fold cross-validation. The 2-fold cross validation starts its first iteration. The entire dataset is divided in two, with half of the elements belonging to a training set and half belonging to a testing set. A decision tree is created from the training set and it attempts to correctly identify elements from the testing set, a set of data the tree has not seen up to this point. The decision tree correctly guesses a percentage of the testing set's elements and this accuracy is recorded. The first iteration of the validation is complete. With the second iteration, the training set and the testing set are switched. A decision tree is built from the new training set (the testing set in iteration 1) and attempts to correctly classify elements in the new testing set (the training set in iteration 1). The accuracy of this classification is captured and the second iteration is completed. To assess the accuracy of the entire decision tree model, an average of the first accuracy score and the second accuracy score is presented.

The k-fold cross validation accuracy measure provides a meaningful estimation of the overall accuracy of the classifier. When model performance is poor, this may imply that there are no meaningful patterns in the feature data extracted for the experiment or that a different classification algorithm should be used.

The model itself often provides insight into the problem of intrusion detection. For example, a decision tree presents a set of rules that differentiate between malicious and benign traffic. The rules may highlight specific ports, IP addresses, or other features extracted from data of interest. These insights can help an intrusion detection team work towards follow-on actions based on the model's results.

3.6. Running the Decision Rules in Real-Time

After performing the decision tree analysis, we have a predictive model that can differentiate different types of traffic. For a model trained to identify benign and malicious traffic, we could perform a number of follow-on tasks:

- Build additional firewall rules (e.g. iptables) to drop offending traffic based on the decision tree results
- Generate IDS rules to detect malicious traffic based on the decision tree results
- Periodically scan through logs or traffic captures and report identified high-priority malicious traffic
- Implement data mining support to aid in intrusion detection for the organization – MITRE's "Data Mining for Network Intrusion Detection: How to Get Started" technical paper provides focused information about how-to implement a data mining capability (Bloedorn, Christiansen, Hill, Skorupka, Talbot, & Tivel, n.d.).

Although the benign traffic versus malicious traffic scenario is a common example discussed, a number of other models can also be built and used. For example, a team struggling with false positive signature hits might build a decision tree to differentiate low-priority scanning activity from high-priority exploitation activity. The technique can be used for a variety of purposes, with the goal of improving the overall IT security posture of the organization.

Appendix A provides step-by-step for how-to implement these techniques with freely available software and what the resulting analysis will provide for an organization.

4. Conclusion

Decision trees are a technique from data mining that categorize new pieces of information into a number of predefined categories. Decision trees use a pre-classified dataset to learn to categorize data based on existing trends and patterns. After the tree is created, the logic from the decision tree can be incorporated into a number of different intrusion detection technologies including firewalls and IDS signatures.

Decision tree analysis has the potential to support an intrusion detection team with the many challenges of defending a network. Due to the ever-increasing volume of data decision trees have the potential to save time for security experts and assist in the analysis of malicious data. An organization can try implementing decision trees with existing network data. When performing this analysis, the decision tree algorithm learns the idiosyncrasies of the network and provides tailored feedback to support intrusion detection.

Additional research in data mining can improve, automate, and simplify this technique for use in industry. As Weka is an extendable Java tool, it is feasible to write software to create a decision tree that incorporates data and learns in near-real time. A real-time data mining tool would make the technique more feasible for a wider audience. In addition, conversion rules can be developed to automatically create Snort signatures based on the resulting decision tree.

On a broader scale, if organizations pool their knowledge, a comprehensive decision tree can be built based on information from multiple sources. The knowledge base for a comprehensive decision tree could integrate information from industry, government, anti-virus vendors, and academia to increase awareness of APT tactics, and improve the overall security of the community.

Appendix A: Decision Tree Botnet Analysis Scenario

This appendix presents step-by-step instructions on how to apply data mining techniques to a sample set of data to create rules to find malicious traffic. The experiment implements a classification algorithm that differentiates between normal traffic from malicious traffic in the form of a decision tree. If certain conditions are met, the algorithm will classify traffic as either normal or malicious.

These instructions were built on a Windows 7 machine; however, the instructions and the tools used should also be compatible with older versions of Windows, Linux-based machines, and Macs.

Setup (Data Collection, Tool Acquisition)

1. Download the following files from http://www.openpacket.org/
    - example.com-1.pcap
    - example.com-3.pcap
    - example.com-4.pcap
    - example.com-5.pcap
    - example.com-6.pcap
    - example.com-7.pcap
    - zeus-sample-2.pcap (botnet traffic)
    - zeus-sample-3.pcap (botnet traffic)
    - 12b0c78f05f33fe25e08addc60bd9b7c.pcap (Kraken bot traffic)
2. Download and install tcptrace from http://www.tcptrace.org/
3. Download and install Weka from http://www.cs.waikato.ac.nz/ml/weka/

Feature Extraction

1. For each pcap file, run the command:

    tcptrace –csv -l filename1.pcap > filename1.csv

    (where filename is the name of the pcap file)

2. From each csv file, remove rows 1-8 (the row before conn #)
3. From each csv file, delete the following columns EXCEPT
    - port_a
    - port_b
    - total_packets_a2b
    - total_packets_b2a
    - unique_bytes_sent_a2b
    - unique_bytes_sent_b2a
    - Session Duration
4. Add a new column called "Class" to each spreadsheet. Fill in each cell of the new column with either "NORMAL" or "MALICIOUS," based on the filename. If the filename starts with "example," fill in all cells with the value "NORMAL". Otherwise, fill in each cell of the spreadsheet with the value "MALICIOUS"
5. Copy and paste all cells from the spreadsheets into a single csv file called "traffic_analysis.csv"

Model Creation

1. Run Weka
2. From the Weka GUI Chooser, click on the Explorer button
3. From the Weka Explorer GUI, click on Open File...
4. Using the explorer, open the traffic_analysis.csv file
5. Click on the Classify tab at the top of the Weka Explorer GUI
6. Click on the "Choose" button to select a classifier
   From the menu, expand the trees icon
   Click on the J48 Tree classifier
7. On the Classify GUI, click on the Start button to start the classifier

Weka Output

Weka produces the following output based on the input data and the J48 classifier:

```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     Weka1
Instances:    657
Attributes:   8
              port_a
              port_b
              total_packets_a2b
              total_packets_b2a
              unique_bytes_sent_a2b
              unique_bytes_sent_b2a
              Session Duration
              Class
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree
------------------

port_a <= 1055
|   port_b <= 447: Malicious (38.0)
|   port_b > 447: Normal (35.0)
port_a > 1055: Normal (584.0)

Number of Leaves  :   3
```

Size of the tree : 5

Time taken to build model: 0.06 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances	657	100%
Incorrectly Classified Instances	0	0%
Kappa statistic	1	
Mean absolute error	0	
Root mean squared error	0	
Relative absolute error	0%	
Root relative squared error	0%	
Total Number of Instances	657	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	0	1	1	1	1	Normal
	1	0	1	1	1	1	Malicious
Weighted Avg.	1	0	1	1	1	1	

=== Confusion Matrix ===

a	b	<-- classified as

```
619   0 |   a = Normal
  0  38 |   b = Malicious
```

Analysis of Weka Results

The Weka output is broken down into run information, classifier model, and stratified cross-validation results. The run information provides general information about the model including the algorithm used, the data file examined, the number of instances in the file, the features used in the experiment, and the testing method.

The next section of the Weka output is the classifier model. The classifier model section is the most pertinent for intrusion detection. When using the Weka decision tree algorithms, the section shows a set of rules identified to determine whether or not the connection is malicious. This decision tree in the output listed above states that connections with port_a <= 1055 and port_b <= 447 are malicious, otherwise the connection is normal. For reference, tcptrace defines port_a as the port of the machine initiating the connection and port_b as the port of the machine receiving the connection (Ramadas, 2003).

The last section of the Weka output is the stratified cross-validation results. Essentially, this section answers the question: how well did this model perform? For this experiment, 10-fold cross-validation was used as the technique used to evaluate the accuracy of the model. Essentially this technique builds 10 versions of the model using 9/10 of the data. Accuracy is estimated based on how well the model works on the remaining 1/10 of the data averaged across all 10 runs.

For this data set, the summary shows that the model had 100% accuracy in differentiating malicious traffic from normal traffic as well as some other error statistics. The detailed accuracy by class section presents a number of statistics for use in data mining, however these statistics can be difficult to interpret. The confusion matrix presents how the decision tree algorithm classified the data as compared to the actual category of the data.

These rules correctly classify 100% of the network data examined. After building

this model, the rules can be incorporated into tools like Snort, firewalls, or detection scripts to identify malicious activity in real-time.

Appendix B: Rule-Based Classification Example

The exercise in Appendix A can be expanded to include an additional category of traffic – scanning activity. The following presents a list of instructions to implement a rule-based classifier using these three categorizations of network traffic. Additionally, this appendix presents an example of the classification algorithm detecting activity based on data elements that are not captured by IDS like Snort in real-time, using features like the total packet counts sent from one host to another.

Setup (Data Collection, Tool Acquisition)

1. Download all of the tools and data files in Appendix A
2. Download the following additional file from http://www.openpacket.org/:
    - tcp-scan.pcap

Feature Extraction

1. Follow all instructions in Appendix A to extract all features from the malicious and benign traffic captures
2. For the tcp-scan.pcap file:
   For each pcap file, run the command:
   tcptrace –csv -l filename1.pcap > filename1.csv
   (where filename is the name of the pcap file)
3. Remove rows 1-8 (the row before conn #)
4. For the file, delete the following columns EXCEPT
    - port_a
    - port_b
    - total_packets_a2b
    - total_packets_b2a
    - unique_bytes_sent_a2b

- unique_bytes_sent_b2a
- Session Duration

5. Add a new column called "Class" to the resulting spreadsheet. Fill in each cell of the new column with "SCANNING"

6. Copy and paste all cells from the spreadsheets into a single csv file called "traffic_analysis2.csv"

Model Creation

1. Run Weka
2. From the Weka GUI Chooser, click on the Explorer button
3. From the Weka Explorer GUI, click on Open File...
4. Using the explorer, open the traffic_analysis2.csv file
5. Click on the Classify tab at the top of the Weka Explorer GUI
6. Click on the "Choose" button to select a classifier
   From the menu, expand the rules icon
   Click on the JRIP classifier
7. On the Classify GUI, click on the Start button to start the classifier

Weka Output

Listed below is a trimmed version of the output from Weka from the rule-based classifier:

JRIP rules:

==========

(port_a <= 1055) and (port_b <= 447) => Class=Malicious (38.0/0.0)

(total_packets_b2a <= 0) and (port_a >= 1721) and (port_a <= 4868) => Class=Scanning (420.0/2.0)

(total_packets_b2a <= 0) and (port_a <= 1559) and (port_a >= 1064) and (port_b >= 1093) => Class=Scanning (148.0/0.0)

(port_a <= 1089) and (total_packets_b2a <= 0) and (port_a >= 1053) and (port_b <= 2459) => Class=Scanning (7.0/0.0)

=> Class=Normal (619.0/2.0)

Number of Rules : 5

Time taken to build model: 0.22 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	1214	98.539%
Incorrectly Classified Instances	18	1.461%
Kappa statistic	0.9724	
Mean absolute error	0.0113	
Root mean squared error	0.0972	
Relative absolute error	3.2145%	
Root relative squared error	23.1572%	
Total Number of Instances	1232	

Analysis of Weka Results

For this experiment, the rule-based classifier JRIP provided 5 rules to characterize network traffic. These rules correctly classify over 98% of the data and can be incorporated into tools like Snort, firewalls, or detection scripts to identify malicious activity.

References

Bloedorn, E., Christiansen, A. D., Hill, W., Skorupka, C., Talbot, L. M., & Tivel, J. (n.d.). Data Mining for Network Intrusion Detection: How to Get Started. MITRE Technical Report. Retrieved January 15, 2011, from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.102.8556&rep=rep1&type=pdf

Breiman, L. (1993). Classification and regression trees. New York, N.Y.: Chapman & Hall.

Brugger, S. (2004, June 9). Data Mining Methods for Network Intrusion Detection. Master's Thesis University of California, Davis. Retrieved December 30, 2010, from citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.88.3127&rep=rep1&type=pdf

Chan, P., Fan, W., Prodromidis, A., & Stolfo, S. (1999). Distributed data mining in credit card fraud detection. Intelligent Systems and Their Applications, IEEE, 14, 67-74.

Ganesan, V. (2005, February 27). VenChar: February 2005. VenChar. Retrieved April 1, 2011, from http://www.venchar.com/2005/02/index.html

Grégio, A., Santos, R., & Montes, A. (2007). Evaluation of data mining techniques for suspicious network activity classification using honeypots data. Proc. of SPIE, 6570, 1-10.

Lee,W. and Stolfo, S. J. (1998). Data mining approaches for intrusion detection. In Proceedings of the 7th Symposium on USENIX Security (San Antonio, TX, Jan.).

Julisch, K. (2003). Clustering intrusion detection alarms to support root cause analysis. ACM Transactions on Information and System Security (TISSEC), 6, 443-471.

PredictionWorks: Data Mining Glossary. (n.d.). PredictionWorks. Retrieved February 11, 2011, from http://www.predictionworks.com/glossary/index.html

Quinlan, J. R. (1993). C4.5: programs for machine learning. San Mateo, Calif.: Morgan Kaufmann Publishers.

Rachwald, R. (2010, November 12). Trend #1: Advanced Persistent Threat (APT) Meets Industrialization - Imperva Data Security Blog. Imperva Data Security Blog. Retrieved March 15, 2011, from http://blog.imperva.com/2010/11/trend-1-advanced-persistent-threat-apt-meets-industrialization.html

Ramadas, M. (2003, August 24). TCPTRACE Manual. tcptrace - Official Homepage. Retrieved December 15, 2010, from http://www.tcptrace.org/manual/index.html

Sangkatsanee, P., Wattanapongsakorn N., & Charnsripinyo C. (2009). Real-time

Intrusion Detection and Classification, IEEE network, 1-5.

Shi, H. (2006). Best-first Decision Tree Learning. Master's Thesis, University of Waikato. Retrieved April 26, 2011, from

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.149.2862&rep=rep1&type=pdf

Simon, G. J., Xiong, H., Eilertson, E., & Kumar, V. (2006). Scan detection - a data mining approach. SIAM International Conference on Data Mining, 6, 118-129.

Tan, P., Steinbach, M., & Kumar, V. (2005). Introduction to data mining . Boston: Pearson Addison Wesley.

TrendMicro. (2010). TrendLabs 2010 Annual Report. TrendMicro Threat Report. Retrieved April 29, 2011, from http://us.trendmicro.com/us/trendwatch/research-and-analysis/threat-reports/

Zupan, B., & Demsar, J. (2008). Open-Source Tools for Data Mining. Clinics in Laboratory Medicine, 28, 37-54.

Last Updated: September 19th, 2017

Upcoming Training

SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
Baltimore Fall 2017 - SEC503: Intrusion Detection In-Depth	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	vLive
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS October Singapore 2017	Singapore, Singapore	Oct 09, 2017 - Oct 28, 2017	Live Event
Community SANS Ottawa SEC503	Ottawa, ON	Oct 16, 2017 - Oct 21, 2017	Community SANS
SANS Berlin 2017	Berlin, Germany	Oct 23, 2017 - Oct 28, 2017	Live Event
SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
San Diego Fall 2017 - SEC503: Intrusion Detection In-Depth	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	vLive
SANS San Diego 2017	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Paris November 2017	Paris, France	Nov 13, 2017 - Nov 18, 2017	Live Event
Community SANS Pensacola SEC503	Pensacola, FL	Nov 27, 2017 - Dec 02, 2017	Community SANS
SIEM & Tactical Analytics Summit & Training	Scottsdale, AZ	Nov 28, 2017 - Dec 05, 2017	Live Event
SANS Munich December 2017	Munich, Germany	Dec 04, 2017 - Dec 09, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DC	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS Security East 2018	New Orleans, LA	Jan 08, 2018 - Jan 13, 2018	Live Event
SANS Las Vegas 2018	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	Live Event
SANS London February 2018	London, United Kingdom	Feb 05, 2018 - Feb 10, 2018	Live Event
SANS Dallas 2018	Dallas, TX	Feb 19, 2018 - Feb 24, 2018	Live Event
SANS Northern VA Spring - Tysons 2018	Tysons, VA	Mar 17, 2018 - Mar 24, 2018	Live Event
SANS Secure Canberra 2018	Canberra, Australia	Mar 19, 2018 - Mar 24, 2018	Live Event
SANS 2018	Orlando, FL	Apr 03, 2018 - Apr 10, 2018	Live Event
SANS Baltimore Spring 2018	Baltimore, MD	Apr 21, 2018 - Apr 28, 2018	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced