International Journal of Reliability, Quality and Safety Engineering
Vol. 23, No. 1 (2016) 1650005 (19 pages)
© World Scientific Publishing Company
DOI: 10.1142/S0218539316500054

Detecting Network Attacks Based on Behavioral Commonalities

Maryam M. Najafabadi*, Taghi M. Khoshgoftaar† and Amri Napolitano‡
Florida Atlantic University, Boca Raton, Florida, USA
**mmousaarabna2013@fau.edu*
†khoshgof@fau.edu
‡amrifau@gmail.com

Received 20 May 2015
Revised 14 December 2015
Accepted 11 January 2016
Published 16 February 2016

Due to the great increase in the amount of attacks that occur in computer networks, there is an increasing dependence on network intrusion detection systems which monitor and analyze the network data to detect attacks. In recent years, machine learning methods have been used to build predictive models for network intrusion detection. These methods are able to automatically extract patterns from the network data to build detection models. Defining proper features, which help models to better discriminate between normal and attack data, is a critical task. While network attacks vary widely, they share some commonalities. Many attacks, by their nature, are repetitive and exhibit behaviors different from normal traffic. Among these commonalities are self-similarity between attack packets, periodicity and repetition characteristics seen in the attack traffic. In this paper, we study the common behaviors between two different attack types, called RUDY and DNS Amplification attacks, in order to propose new features for building predictive models by using machine learning algorithms. We collected Netflow traffic from an operational ISP network. We introduce a concept called "session" derived from Netflow which incorporates both sides of a network communication to define a network instance. Features are extracted for each session. To demonstrate how the newly defined features work for the task of intrusion detection, we use these features to build intrusion detection models for the detection of RUDY attack, DNS Amplification attack and the combination of these two attacks. To build predictive models we apply four machine learning classification algorithms: two versions of a decision tree algorithm, Naïve Bayes and 5-Nearest Neighbor (5-NN) algorithm. Our results show that the proposed features based on the attack commonalities provide very good prediction results for the detection of two studied attacks on real network traffic.

Keywords: Intrusion detection; machine learning; session network data; RUDY; DNS amplification; feature extraction.

1. Introduction

Nowadays, people use the internet and computers to do banking, shopping, make appointments, fill in forms and all manners of essential services. Our everyday life is becoming increasingly dependent on communication networks technology. With all the private and critical information (social security numbers, bank account information, etc.) being available online, there is a need for providing security and reliable operation of online communication networks. Additionally, in recent years the number of attacks on these networks has dramatically increased which consequently makes the security of computer networks even more important.

Intrusion Detection Systems (IDSs) provide security by monitoring and analyzing network related traffic for malicious activities.[1] There are two types of IDSs: host and network IDSs.[2] A host IDS runs as software installed on an individual host. It monitors network traffic which enters and exits the host to detect attacks. The main downside of host IDSs is that they are not able to detect the attacks that need a network-wide view point to be detected, such as distributed attacks. Network IDSs, on the other hand, are placed in a computer network where they monitor the traffic to/from all network devices and hosts. In comparison to host IDSs which observe only one individual host's data, network IDSs observe the whole network traffic which gives them the view to detect network attacks. Also they are not dependent upon a host's operating system which makes them more scalable. In this work, we study how to build a network intrusion detection model by applying machine learning methods on network data.

The most common operational network IDSs are signature-based systems.[3] These systems consist of a database of attack signatures. Human experts produce the attack signatures by manually analyzing the attack data. The monitored network traffic is matched against this database to detect malicious activities. Producing attack signatures is a time consuming and manually intensive task. Recently, machine learning techniques have been applied to build predictive models for the detection of network attacks.[4,5] Unlike signature-based methods which need manual analysis by human experts to extract attack patterns, machine learning algorithms are able to automatically extract similarities and patterns in the network data with no need form annual analysis.

To apply machine learning algorithms on any kind of data, including network data, the very first step is to extract features from data instances. A feature vector is extracted from each data instance. This is actually a vector of the feature values that represents the data instance. This process transforms the initial data to a set of feature vectors. After this step, the machine learning algorithm is applied on the set of extracted feature vectors to build the predictive models. Figure 1 shows this process.

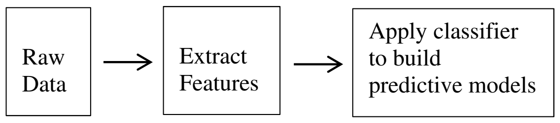

Fig. 1. The steps of building predictive models.

Extracting features is a very critical task in applying machine learning methods. The features should be informative and related to the underlying classification task. For example, extracting hair color as a feature for the detection of lung cancer is neither informative nor related to the detection task. Expert knowledge should be taken into account for the extraction of proper and useful features.

In the current task, which is the detection of the network attacks, an appropriate feature is the one which is useful for the classification of attack traffic from normal traffic. We call these features "discriminative features", because they are able to discriminate between normal and attack data. Introducing discriminative features for intrusion detection requires domain knowledge and expert analyses. In this paper, we define such features based on the commonalities observed in the attack data which makes them different from normal data.

Our main motivation for the work being presented here is that while network attacks are different, they share some behavioral commonalities which can discriminate them from normal data. Many attacks, by their nature, are exceedingly repetitive and exhibit behaviors different from normal traffic. Most of the attacks need to be repetitive in nature for the attack to be impactful. This leads to behavioral characteristics such as self-similarity between attack packets, periodicity and repetition.

In this work, we study RUDY (R U Dead Yet?)[6] and DNS Amplification[7] attacks to define discriminative features for the task of intrusion detection. In our analysis, the collection and analysis of real world internet traffic is key in identifying features that can be used to build machine learning models. The idea that anomalous and malicious traffic share commonalities that can be measured and enumerated is the rational used to define new features for the machine learning-based intrusion detection in this work.

We collected real network data from an operational ISP network. Our network experts analyzed and manually labeled the collected data. We focused on two different types of attacks: Real DNS Amplification inherently found in the data and RUDY (R U Dead Yet?) attack traffic generated through penetration testing. DNS Amplification attack attempts to flood the victim machine by DNS traffic. The attacker directs DNS traffic to the victim machine by sending spoofed DNS requests to the DNS server. RUDY is an application layer attack. The attacker creates simultaneous HTTP POST connections to abuse server resources. To study our newly defined features we use two versions of decision tree algorithms in our machine learning analysis.

The common occurrence of encrypted traffic and the increase in the volume of the data transferred on computer networks make the detection of attacks over these networks challenging. In recent years, Netflow-based analyses have been introduced to address these issues.[8,9] Netflow is an aggregation of the network packets that have some network features such as source IP, destination IP, source port, destination port and protocol in common. Since in Netflow-based analysis we are analyzing an aggregation of packets instead of analyzing every single packet, the analysis takes place faster. Therefore, Netflow-based analysis is appropriate for high speed networks. No payload traffic is analyzed, so it is also a proper approach for analyzing encrypted traffic.

Netflow is a uni-directional aggregation of packets. Like communication between people, network data context is largely dependent on hearing both sides of the conversation. Network traffic is similar in the sense that the inbound (traffic targeting the network) and outbound traffic (traffic leaving the network) typically results from one another. By analyzing the inbound and outbound Netflows separately, context can be obscured. In our analyses we propose the concept of a "session". A session contains corresponding outbound and inbound Netflows. We define the "session" in a way that it incorporates both sides of the communication in the definition of a network instance.

We extracted network sessions from the Netflow data collected in our data collection process. We extracted session-based features for each traffic instance, i.e., a session. In our analysis we applied four different classification methods for the task of detection of network attacks in the network data. The classification methods we used are 5-Nearest Neighbors (5-NN), two versions of the C4.5 decision tree algorithm and Naïve Bayes (NB).

We applied two main sets of experiments. In the first set of our experiments, we aimed to see how well the session-based features perform in the detection of each of the two studied attacks separately. We applied the classification methods for the detection of each of the RUDY and DNS Amplification attacks separately. Since our main goal is to introduce a feature set that can be used generally for the detection of different types of network attacks, in the second set of our experiments we combined the data from RUDY and DNS Amplification attacks. We then applied our classification methods to detect these two attacks together. Our results suggest that the provided session-based features perform very well in the detection of the two studied attacks in the network data.

The remainder of this paper is structured as follows. Section 2 provides some related works on machine learning-based intrusion detection and the two specific attacks studied in this paper. Section 3 provides information about the attacks studied in this work. Section 4 provides details about our case study dataset. Section 5 discusses the machine learning methods utilized. In Sec. 6, we discuss our results. In Sec. 7 we discuss our work. Finally, in Sec. 8 we present our conclusions and future works.

2. Related Work

Machine learning methods have been used to build predictive models for the task of network intrusion detection.[4,5,10,11] These methods are able to automatically extract patterns and similarities in the network data that manual analysis or other methods might not be able to detect. The extracted patterns are then used to apply predictive tasks on the newly seen network data. Machine learning methods have been used in the detection of network attacks in order to reduce the high amount of false alarms seen in the traditional methods such as firewalls. In this section, we explain some of the previous works done in using machine learning methods in the detection of attacks. We also review some works that have been done for the detection of RUDY and DNS Amplification attacks in particular.

Sinclair *et al.*[12] employed genetic algorithms and decision trees to build intrusion detection models. They classified connections by using five features: source IP, destination IP, source port, destination port and protocol. They did not discuss what happens if a new IP or port is observed which is not included in any of the produced intrusion detection rules.

Mukkamala *et al.*[13] compare Support Vector Machine (SVM) and neural networks methods for building intrusion detection models on the KDD dataset.[14] The intrusion detection models are built as binary classifiers to predict attack data from normal data (all different attack types in KDD data set are labeled as attack). They build the models by using all 41 features of KDD datasets. They also applied a feature selection method[15] to select most significant features (13 of 41) and build the models by using only these features. Their results show that SVM consistently (using either 41 or 13 features) outperforms neural network in terms of training time and accuracy of detection. Peddabachigari *et al.*[16] compare decision tree and SVM to build IDSs. They applied their analysis on DARPA dataset. Their results show that decision tree gives better overall performance than SVM. In these works, the authors used the pre-defined features in the DARPA and the KDD datasets to build the intrusion detection predictive models. In the current work, we study the common behaviors between attacks to introduce new features for the task of intrusion detection.

DNS Amplification attack and RUDY attack have been studied in some works. Kambourakis *et al.*[17] proposed a mechanism to detect DNS Amplification attacks based on the one-to-one strict mapping of DNS requests (queries) and responses. Under normal DNS operation, the client sends a name resolution request to the DNS server and the server responds back a DNS reply. In a DNS Amplification attack, the client is receiving DNS responses without sending a DNS request. Based on this idea they employ a monitor to record both DNS requests and responses using the IPtraf tool.

For every DNS message received, the detection system determines whether the message is a request or a response message. If it is a response message then a look up is done to find the corresponding request message. If the response does not match with any of the DNS requests logged previously in a given timeframe then it is marked as suspicious. When the ratio of the suspicious responses exceeds a predefined threshold, then an alert is generated. In Ref. 18, they extended the aforementioned paper[17] work by incorporating bloom filters to speed up the process of detection.

Rastegari *et al.*[19] applied neural networks for the detection of a DNS Amplification attack. They extracted three features from 20 s long time windows of network traffic. The number of received bits at the server during the time window, average size of the received packets and the number of lost DNS packets are three features they extracted from each time window. They applied three different neural network methods on simulated network traffic: Back Propagation (BP), Radial Basis Function (RBF) and Self-Organizing Map (SOM). They conclude that the BP neural network provided the best detection rate for the detection of DNS Amplification attacks. In addition, for remediation of DNS Amplification attacks several security advisories and guidelines have been issued in US-CERT[20] and Ref. 21.

RUDY[22] is a slow rate application layer DOS attack, also called "low and slow". It targets the HTTP protocol. The attacker seeks to launch concurrent HTTP requests that are difficult to serve back. It leads to the depleting application resources and bringing down the victim's website. RUDY attack has been studied in Ref. 23, where Damon *et al.* present a hands on lab experiment for RUDY attack. They mainly focused on simulated attack scenarios.

Different mitigation methods have been proposed for the RUDY attack.[6] Some mitigation methods track server memory, CPU usage, and connection table to identify the abuse of resources which can include long and idle open network connections. Other methods monitor the open server connections. Such methods and the method provided for detection of DNS Amplification attack in Ref. 17 are host-based intrusion detection methods. In this paper, we are studying a network level detection investigation of RUDY attack.

3. Attacks

We studied two types of attacks in this paper to define our features. Our network experts studied the characteristics of RUDY and DNS Amplification attacks and used the knowledge to define new features for the detection of attacks. In this section, we explain the studied attacks and their characteristics that lead us to define our new features. The general approach is to differentiate the behavior inherent in normal traffic from anomalous and attack traffic in a manner that produces features that exhibit clear numerical differences. This is the approach we have taken in producing the SANTA[24] dataset. This approach differs significantly from the other datasets such as the KDD 99 (Ref. 14) and Kyoto[25] data sets. The collected data is not simulated and it is real network traffic.

DNS Amplification: In this attack, the instigator aims to flood a target with DNS response traffic by forging the target machine's source IP (known as spoofing) and sending DNS name lookup request to an open DNS server.[7] Domain Name System (DNS)[26] translates a domain name such as www.google.com to the numerical IP address which is needed to make connections among computers worldwide. When a client needs to look up a domain name, it queries the DNS server to resolve the name. The DNS server responds the client's DNS query by sending it back the information of the requested domain name.

In DNS Amplification attack, the attacker sends the DNS request to the server with a forged IP which belongs to the target's machine. When the DNS server responds to this DNS request, the response is actually sent to the target machine. The spoofed requests sent by attackers are of type "any", which results in the response to include all known information about a DNS zone. Therefore, the inbound packets to the intermediate DNS server are considerably smaller than the resulting outbound packets that are sent to the target machine, thus the attack is amplified. The attacker attempts to flood the target's machine by sending a lot of spoofed DNS requests to the DNS server.

The repetitive and periodic nature of the DNS Amplification attack, which is required for the attack to be effective, allows for the quantification of new attributes that provide metrics for repetition and periodicity. In other words, these attacks, by their very nature, will provide very different values for attributes that measure repetition and periodicity than normal traffic would. Further the packets of this session exhibit a very high degree of self-similarity, thus allowing for self-similarity measurements elucidated in terms of convergence. DNS Amplification also differs from normal DNS traffic in terms of velocity.

RUDY: RUDY (R U Dead Yet?) is a slow rate HTTP POST denial of service attack that exploits a weakness in the HTTP protocol that was originally designed to allow web services to support users with slow connections (e.g., dial up connections). The instigator attempts to open a few number of connections to the targeted server and keeps them open as long as possible to abuse the server's resources.

RUDY attacks take advantage of the fact that an HTTP POST operation allows for the connection to remain open indefinitely in cases where the post data arrives very slowly; for example one byte per second. The attacker starts an HTTP POST operation and then delays the completion of the transaction by sending very small packets to keep the session open without completing the POST operation. The attacker initiates simultaneous HTTP POST connection to the server. Since the server is hanging while waiting for the rest of the HTTP POST request, simultaneous connections ultimately create a denial of service condition. The subtle nature of this attack makes it an interesting case for study.

The RUDY attacks exhibits a very low inbound velocity, and will be quite periodic and the packets are relatively similar. These characteristics make RUDY attacks an outlier from normal HTTP traffic. This is due to the fact that normal HTTP traffic is a series of requests and responses, while RUDY is one unending request that can never receive a response.

4. Case Study Data

4.1. *Data collection and labeling*

Among the data collection studies we have done in the past,[24,27] we used data from our previous data collection, SANTA dataset.[24] The SANTA dataset is collected network traffic from an operational network of a commercial Internet Service Provider (ISP). The ISP does not provide service to any consumer accounts and also hosts a wide range of server traffic that is accessed by users from across the internet, including web servers, email servers, DNS servers and other various internet services that are common to ISPs. The collected traffic includes a mixture of varying types of internet traffic. Some of the traffic is generated from customer networks that access the internet through the ISP network, including email, browser, DNS and all other types of internet traffic that an average business might generate in the course of day-to-day operations.

The operational nature of the ISP network allows for observation of the wide network anomalies and attacks. The operational nature of the data makes it an excellent choice for the study of network intrusion detection. Our network experts labeled real DNS Amplification attack in the collected data. The RUDY attack was produced by penetration testing in which the network expert produced RUDY attack data toward the ISP network from an external host.

The data is collected from two border routers that connect the ISP network to the outside world. Since the collected data is only border traffic and no internal traffic (from one internal host to another internal host) is included, two categories can be defined for the collected traffic: inbound and outbound traffic. Outbound traffic is the traffic that originates from any IP addresses inside the network. Inbound traffic is all other packets that are actually originating from outside the ISP network and are targeting the ISP network. We use these two concepts in defining our network features. We use these concepts to define our network instances, called sessions, which present a bi-directional concept of the traffic.

We collected both packet and Netflow data. Packet data is simply the raw packets that are traversing through the borders of the network. It includes the whole packet and payload data. Many anomaly and intrusion detection methods use a method called deep packet inspection to analyze the payload data in order to detect attacks. This is a very computationally expensive approach. Furthermore, nowadays the internet traffic is moving toward encrypted traffic. In the event that payload is encrypted, the deep packet inspection may not be discernible in any reasonable time frame or even at all. This points to a clear need of analyzing the network traffic without payload such as inspecting packet headers, Netflow data or data derived there from.

Netflow, originally introduced by Cisco,[28] is actually an aggregation of network packets that share some network features in common. The most used set of features in the definition of Netflow are the 5-tuple features: source IP, destination IP, source port, destination port and protocol. Since Netflow is an aggregation of packets its analyses is faster than analyzing every single packet in the network. Since payload inspection is not involved in the definition of Netflow, Netflow analysis can be applied to encrypted traffic. We collected Netflow version 9 data from the ISP network routers.

4.2. *Network sessions and defined features*

Considering the common behaviors between two studied attacks (explained in Sec. 3), we proposed new features in a way that represent these characteristics. To define the network features, we construct our data instances by introducing a new concept called "session". In our dataset each data instance is a network session. A network session is constructed by grouping related inbound and outbound Netflows together. A related pair of inbound and outbound Netflows is defined by considering the correspondence between their 5-tuple features. For example, the source IP of the inbound Netflow should be equal to the destination IP of the outbound Netflow. This implies that an inbound DNS request packet produces a flow record in the Netflow data that is later paired with the corresponding outbound flow record to produce a session. Once grouped by session, we extract our defined features for each session.

The unidirectional nature of packet and Netflow data do not provide a complete context for the network traffic. Similar to the communication between people where the context is largely dependent on hearing both sides of the conversation; in the network traffic it is important to represent the network traffic in a bidirectional context which includes both sides of the traffic (inbound and outbound traffic). Our definition of a network session represents each traffic instance with a bi-directional context. The number of extracted sessions in our collected data is shown in Table 1.

Due to the fact our features are extracted from sessions rather than Netflow or packets and while some of the features are analogous to prior datasets, most of the features are unique. A Boolean variable named "IO match" confirms that each inbound flow corresponds to an outbound flow in a session.

We defined three main categories of features along with other features that are more analogous to the features used in other datasets such as DARPA. The three categories of defined features are: self-similarity, periodicity and velocity related features. Self-similarity features represent the similarity between the packets. This is shown by calculating the variance of size of the inbound and outbound packets in a session. To calculate periodicity, we measure the variance of the difference in time intervals between each inbound packet that belongs in the same session. This variance of the difference between timestamps is used as the periodicity feature. Velocity features are defined by providing bits per second, packets per second and bytes per packet for both inbound and outbound traffic. The list of the features extracted from the session data is provided below. For each feature, its abbreviated name which is used in our data analysis is provided in the parenthesis in front of the feature.

Table 1. Dataset information.

Data type	Number of extracted sessions
Normal	29,292
RUDY	729
DNS amplification	988

(1) **Protocol (proto):** The transmission protocol of the session (TCP, UDP, etc.)

(2) **IO match (io_match):** This is a Boolean feature which shows whether the inbound Netflow has an associated outbound Netflow in the underlying session.

(3) **Session duration (duration):** The elapsed time, from the earliest of the associated inbound or outbound Netflow until the end of the later Netflow.

(4) **Session size in Bytes (bytes):** The total size of the session in bytes.

(5) **Session size in packets (packets):** The total size of the session in packets.

(6) **Inbound session convergence (in_conv):** The self-similarity between inbound packets in the session. This feature is calculated by examining the variance of all the inbound packets included in the session.

(7) **Outbound session convergence (out_conv):** The self-similarity between outbound packets in the session. This feature is calculated by examining the variance of all the outbound packets included in the session.

(8) **Session periodicity (periodicity):** The measure of periodicity within a session. This feature is calculated by examining the variance of time stamp differences between inbound packets.

(9) **Inbound velocity pps (invel_pps):** The velocity of inbound traffic measured in packets per second.

(10) **Inbound velocity bps (invel bps):** The velocity of inbound traffic measured in bytes per second.

(11) **Inbound velocity bpp (invel_bpp):** The velocity of inbound traffic measured in bytes per packet.

(12) **Outbound velocity pps (outvel_pps):** The velocity of outbound traffic measured in packets per second.

(13) **Outbound velocity bps (outvel_bps):** The velocity of outbound traffic measured in bytes per second.

(14) **Outbound velocity bpp (outvel_bpp):** The velocity of outbound traffic measured in bytes per packet.

(15) **RIOT packets (riotp):** The ratio of inbound to outbound traffic measured in packets.

(16) **RIOD Bytes (riotb):** The ratio of inbound to outbound traffic measured in bytes.

(17) **Session flags (flags):** Cumulative OR of all the TCP flags seen in this session. In our data representation, each TCP flag is shown by a letter: A (ACK), U (URG), S (SYN), R (RST), F (FIN), P (PSH), E (ECE) and C (CWR) The cumulative OR of the TCP flags in a session is shown by appending the corresponding TCP

flags' letters. For example, flags = "ARS" means that the TCP flags, ACK, RST and SYN are seen in this session.

(18) **Originating ASN (origasn):** The ASN that announced the external source IP address.

5. Applying Machine Learning Methods

We chose two versions of C4.5 decision tree, NB and 5-NN algorithms for our analysis. We built all models using the WEKA machine learning toolkit.[29] We chose these classifiers because they have shown to work well for the application of network intrusion detection.[30–32] For all classification models, attack class was considered as the positive class and normal class was considered as the negative class. Applying our analysis with four different classifiers provides a more comprehensive analysis from a machine learning perspective.

C4.5 decision tree[33] (implementation of the j48 decision tree in WEKA) is a tree-based algorithm in which a decision tree structure is created. Each branch divides the samples into two or more other branches based on the values of one of the features in the data sample. The C4.5 algorithm uses a normalized version of Information Gain to decide the hierarchy of useful features in building tree branches. The information gain is calculated based on the entropy and conditional entropy. Entropy measures the disorder and randomness in a closed system. Information gain is calculated as below:

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v).$$

Information gain measures how much information we gain by splitting the data using a particular attribute. In other words, information gain tells us how much of the uncertainty is reduced by using a particular attribute to split the data. The higher the amount of uncertainty is reduced, the higher the prediction outcome would be. So more the information gain the feature has, the higher it appears in the tree structure.

In this study, we employed a version of C4.5 using the default parameter values from WEKA (denoted C4.5D) as well as a version (denoted C4.5N) with Laplace smoothing activated and tree-pruning deactivated.

K-Nearest Neighbors or K-NN[34] is an instance learning and lazy learning algorithm. In this kind of learning, no general model is learnt prior to the prediction task. Instead, when a prediction request is made, the instance is compared to the training data to predict its class. In the K-NN algorithm, the predicted class for every test instance is derived from the classes of the K closest samples to that instance (in our study, $K = 5$). The instance is classified by the majority votes of its neighbors. Since for each test sample K-NN needs to calculate its distance to all the training samples to specify the K nearest samples to the given sample, K-NN has a $O(n^2)$ complexity that makes it a computationally expensive algorithm.

The NB[35] algorithm uses Bayes' theorem to calculate the posteriori probability of an instance being a member of a specific class. Unfortunately, it is very difficult to calculate the posteriori probability directly. Therefore, a strong assumption of features being independent is made using Bayes' theorem to calculate posteriori probabilities. While, this assumption makes NB a relatively weak learner, it is a fast classifier. This classifier usually is used as a base classifier for evaluating the performances of other classifiers.

The four classification models are trained and evaluated using Cross-Validation (CV). In an m-fold CV, first the whole data is divided into m equal parts. The idea is that each of the m parts are going to be used exactly one time as the test dataset while the other parts of data are used as the training dataset. In each iteration, one of the data parts is left out to be used as the test dataset and the other $m-1$ parts of the data are used as the training dataset to build the predictive model. When the model is built, the left out data part is used to test the model and calculate the performance metrics. The process is done m iterations for each data part to be used exactly once as the test dataset. The final evaluation metrics are calculated by aggregating the evaluation metrics calculated in each iteration.

A 5-fold CV is used in our experiments. Moreover, to mitigate any bias due to a random lucky/unlucky split to create the folds, the CV-based model training and evaluation is repeated four times, and the average performance of the machine learners across the four runs is then evaluated.

We provide three different evaluation metrics to show the models' performances: True Positive Rate (TPR), False Positive Rate (FPR) and Area Under the receiver operating characteristic Curve (AUC). TPR is the percentage of attack instances that are correctly predicted as attack by the model. FPR is the percentage of the normal data which wrongly is predicted as attack by the model. The AUC builds a graph of the TPR versus the FPR as the classifier decision threshold is varied, and then uses the area under this graph as the performance across all decision thresholds. AUC demonstrates the trade-off between TPR and FPR, where higher AUC values indicate a high TPR and a low FPR (which is preferable in the current application, i.e., network attack detection).

6. Results

In this section, we provide the performance results we achieved through applying CV along with our machine learning analysis on the collected data. For each classifier, four runs of 5-fold CV are applied. Since the standard deviation of evaluation metrics produced in four runs of 5-fold CV is very low we simply provided the mean value across the four runs of 5-fold CV for each performance metric.

6.1. *The results on each attack*

The evaluation results on each attack type are provided in Table 2. The evaluation results show that all the four classifiers are providing very good performance by

Table 2. Cross validated results for RUDY and DNS Amplification attacks separately.

Classifier	AUC	TPR	FPR
DNS Amplification attack			
C4.5D	0.9998	0.9979	0
C4.5N	1	1	0
NB	0.9989	0.9979	0.0001
5-NN	1	0.9989	0
RUDY attack			
C4.5D	0.9941	0.9866	0.0003
C4.5N	0.9989	0.9973	0.0003
NB	0.9859	0.9938	0.1625
5-NN	0.9999	0.9883	0.0003

using the newly defined features. Although, NB is a weak classifier, it provides high performance values. 5-NN also has a high performance. Its performance is similar to the C4.5N performance. The C4.5N algorithm performs the best.

The results show that the newly defined features work well for the detection of attacks with different classifiers. Also among the decision tree algorithms, C4.5N is performing better than C4.5D. In C4.5N, no pruning is done after the tree is constructed. So the tree structure is deeper and more complicated which results also in longer and more complicated classification rules.

The decision tree algorithms build the predictive rules by using the dataset attributes. This provides a way to interpret how the newly defined attributes are contributing in the detection of attacks. In the two versions of the decision tree structures built on the whole data from DNS Amplification attack, "Outbound velocity Bpp" is selected as the first discriminative feature in the tree hierarchy. As we explained before in DNS Amplification attack, the response DNS traffic that is actually the outbound traffic is amplified. So even though the DNS request traffic sent by the attacker might look like a normal DNS request, the response DNS traffic which is directed to the victim machine is larger than what is normally seen in response DNS traffic. "Outbound velocity Bpp" represents the velocity of outbound traffic (in this case DNS response) measured in bytes per packet. The larger this feature is, the larger the DNS response would be. In the built decision trees, most of the DNS Amplification attack traffic falls below the branch when this attribute is more than the threshold selected by the decision tree algorithm. This indicates how the introduced attributes are used to reflect characteristics of the attack in the decision tree structure in order to discriminate between attack and normal traffic.

Considering the two versions of the decision trees built on the whole RUDY attack data, "Outbound velocity Bpp" feature is selected at the first level. In the RUDY attack the server is just responding TCP acknowledgment packets to the incoming slow rate attack packets. Despite what normally happens in a HTTP communication no data is actually responded by the server. Therefore, the server response, i.e., outbound traffic, is small compared to normal HTTP outbound traffic. This makes the “Outbound velocity Bpp” feature to be low in RUDY traffic.

In a normal HTTP communication, the HTTP response is larger than the corresponding HTTP request which results in the outbound traffic being larger than the inbound traffic. However, in the RUDY attack, even the small packets sent by the attacker are larger than the HTTP acknowledgment response packets. That is why, in the tree structures, the “RIOT Bytes” feature is selected at the second level. Higher values of this attribute correlate with RUDY attack.

6.2. *The results on combination of attacks*

Our main goal is to introduce network traffic features which are useful for the detection of different types of attacks that share some commonalities in their behaviors (repetition, self-similarity between packets and periodicity). To demonstrate if the session-based features work well when applied on a dataset with presence of different attacks, we combined RUDY and DNS Amplification attacks data. We then applied the classification algorithms on this combined dataset. This shows how well the features are performing in the detection of attacks in a dataset where different attack types exist. The corresponding results are shown in Table 3.

The CV results for the combination of attacks show that among the four classification methods, three of them (5NN, C4.5D and C4.5N) are still performing very well for the detection of attacks. On the contrary, NB is not providing good classification performance. NB is a very simple algorithm with a very strong assumption (the features are independent). Overall the results suggest that the features are performing well for the detection of two attacks, when they are combined.

Again, by analyzing the tree structures we can see the contribution of features in building the predictive rules. The C4.5D decision tree of the whole data from combined attack dataset is shown in Fig. 2. It can be seen that the introduced features are contributing in the decision tree structure of the combination of RUDY and DNS Amplification attacks. The “Outbound velocity Bpp” (outvel_bpp) and “RIOT bytes” (riotb) are used in the very first levels of the tree structure to discriminate between Normal and Attack data. This is similar to what we previously explained about the tree structures of RUDY and DNS Amplification attacks. Since these features show up at the first levels of the RUDY attack decision tree and DNS

Table 3. Cross validated results for the combination of RUDY and DNS attack.

Classifier	AUC	TPR	FPR
C4.5D	0.9962	0.9921	0.0003
C4.5N	0.9993	0.9923	0.0003
NB	0.8836	0.7499	0.1337
5-NN	0.9998	0.9938	0.0003

```
outvel_bpp <= 1486
|  outvel_bpp <= 40
|  |  riotb <= 1.3125: Normal
|  |  riotb > 1.3125
|  |  |  flags =ARS: Normal
|  |  |  flags = APRS: Attack
|  |  |  flags = APR: Attack
|  outvel_bpp > 40
|  |  outvel_bps <= 28800
|  |  |  out_conv <= 42107.58333: Normal
|  |  |  out_conv > 42107.58333
|  |  |  |  out_conv <= 45977.71829: Attack
|  |  |  |  out_conv > 45977.71829: Normal
|  |  outvel_bps > 28800
|  |  |  bytes <= 2286
|  |  |  |  invel_pps <= 22
|  |  |  |  |  out_conv <= 35822.33414
|  |  |  |  |  |  riotb <= 0.516667: Attack
|  |  |  |  |  |  riotb > 0.516667: Normal
|  |  |  |  |  out_conv > 35822.33414: Attack
|  |  |  |  invel_pps > 22
|  |  |  |  |  riotp <= 1.1: Normal
|  |  |  |  |  riotp > 1.1
|  |  |  |  |  |  invel_bpp <= 85: Attack
|  |  |  |  |  |  invel_bpp > 85: Normal
|  |  |  bytes > 2286
|  |  |  |  proto = UDP
|  |  |  |  |  duration <= 729.868: Normal
|  |  |  |  |  duration > 729.868: Attack
|  |  |  |  proto = TCP: Normal
|  |  |  |  proto = ICMP: Normal
outvel_bpp > 1486
|  proto = UDP: Attack
|  proto = TCP: Normal
```

Fig. 2. Decision tree of the combination of attacks.

Amplification attack decision tree separately, it is expected to see them again at the first levels of the decision tree of the combination of these two attacks.

Figure 2 also shows that other session-based features which represent repetition and self-similarity between packets. The features such as out_conv, outvel_bps, invel_pps and invel_bpp cooperate in the decision tree structure to discriminate between Normal and Attack traffic. This, again demonstrates the ability of the session-based features in the detection of attacks.

The provided results suggest that the newly defined attributes perform very well in the detection of the two studied attacks. As it is stated earlier, these attributes are defined based on studies done on the commonalities of different attacks seen in real network traffic.

7. Discussion

Nowadays, the amount of network attacks on network traffic has increased dramatically. There is an important need for intrusion detection models that are able to detect different types of attacks. There are some aspects that should be considered in a network intrusion detection study. We considered these aspects in our current work and we encourage the practitioners to take them into account in an intrusion detection study.

Using an updated dataset: Any intrusion detection model should be applied on a network traffic dataset to analyze its performance. Many of previous research works on network intrusion detection were done on KDD 99 dataset. It is an old dataset and it does not provide the current normal and attack trends in the network traffic. Practitioners need to work on newer datasets which include current network traffic trends. In this study we collected real network traffic from an ISP network. Our network practitioners manually labeled the data as normal and attack data.

Defining discriminative features: It is very important to define powerful features when building intrusion detection models. Practitioners should study different types of attacks as well as normal data to define the features which are able to discriminate between normal and attack data. Our main objective in this work is to introduce some features which represent the common characteristics seen in the different types of attacks. We studied the characteristics of two attacks, RUDY and DNS Amplification to define the discriminative features. The common characteristics in the behavior of these attacks are considered to define our features. We also considered the bidirectional nature of network traffic in the definition of our features by introducing a new concept, named session. A session represents a bidirectional instance of the network traffic.

8. Conclusion

In this paper, we studied the common behaviors between two types of attacks called, DNS Amplification attack and RUDY attack, to define discriminative features for the task of building machine learning models for the detection of these attacks. Extracting features is a very important step in building predictive models. The features should be informative and related to the underlying task. Our motivation is that even though attacks are different, they share some commonalities in their behaviors. Those commonalities can be studied to define the features that can discriminate between normal and attack traffic.

We collected network traffic from a real ISP network. Our network experts labeled real DNS Amplification attacks in the network. We also produced RUDY attack data by performing penetration testing to the target network. The expert analysis of the attack data yields to several commonalities amongst different attack types, such as self-similarity between attack packets, periodicity and repetition.

To extract the features we defined each network instance as a session. A session is an aggregation of inbound and outbound Netflows which includes bi-directional traffic information. Unlike a Netflow which only provides uni-directional information, sessions provide more information about a network communication. The nature of the attack commonalities is considered in the production of new network session features. We used these features to build predictive models for the detection of RUDY and DNS Amplification attacks separately and when these attacks are combined in one dataset. To build the predictive models, we used four different classification algorithms: two versions of C4.5 decision tree, NB and 5-NN. Our results show that the newly defined features are providing very good results for the detection of these attacks in the network. Incorporating expert knowledge of the attacks to define network features makes the predictive models provide good performance results.

For future work, we intend to collect more network data and incorporate other types of attacks to provide more comprehensive analysis of the performance of the proposed features.

Acknowledgment

The authors gratefully acknowledge partial support by the National Science Foundation, under grant number CNS-1427536. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

1. K. Scarfone and P. Mell, Guide to Intrusion Detection and Prevention Systems (IDPS), NIST Special Publication 800-94 (2007).
2. N. Das and T. Sarkar, Survey on host and network based intrusion detection system, *Adv. Netw. Appl.* **6**(2) (2014) 2266–2269.
3. A. Lazarevic, V. Kumar and J. Srivastava, Intrusion detection: A survey, in *Managing Cyber Threats, Issues, Approaches, and Challenges*, Springer, US (2005), pp. 19–80.
4. M. M. Najafabadi, T. M. Khoshgoftaar, C. Kemp and N. Selia, Machine learning for detecting brute force attacks at the network level, *14th IEEE Int. Conf. Bioinformatics and Bioengineering — Workshop on Big Data and Data Analytics Applications*, Boca Raton, FL (2014), pp. 379–385.
5. R. Sommer and V. Paxson, Outside the closed world on using machine learning for network intrusion detection, *2010 IEEE Symp. Security and Privacy (SP)*, Oakland, CA, USA (2010), pp. 305–316.
6. R.U.D.Y. (R-U-Dead-Yet?), Available at http://www.incapsula.com/ddos/attack-glossary/rudy-r-u-dead-yet.html.
7. DNS Amplification Attack, Part of the network security glossary, Available at http://whatis.techtarget.com/definition/DNS-amplification-attack (2013).
8. R. Hofstede, V. Barto, A. Sperotto and A. Pras, Towards real-time intrusion detection for NetFlow and IPFIX, *9th Int. Conf. Network and Service Management* (2013), pp. 227–234.
9. W. Zhenqi and W. Xinyu, NetFlow based intrusion detection system, *Int. Conf. MultiMedia and Information Technology* (2008), pp. 825–828.
10. P. Sangkatsaneea, N. Wattanapongsakorna and C. Charnsripinyob, Practical real-time intrusion detection using machine learning approaches, *Comput. Commun.* **3**(18) (2011) 2227–2235.
11. R. Wald, T. M. Khoshgoftaar, R. Zuech and A. Napolitano, Network traffic prediction models for near and long-term predictions, *2014 IEEE Int. Conf. Bioinformatics and*

Bioengineering (*BIBE*) — *Workshop on Big Data and Data Analytics Applications* (2014), pp. 362–368.
12. C. Sinclair, L. Pierce and S. Matzner, An application of machine learning to network intrusion detection, in *Computer Security Applications Conf.*, *1999.* (*ACSAC '99*) *Proc. 15th Annual*, Phoenix, AZ (1999), pp. 371–377.
13. S. Mukkamala, G. Janoski and A. Sung, Intrusion detection using neural networks and support vector machines, in *2002 Int. Joint Conf. Neural Networks*, *2002. IJCNN '02. Proc.*, Honolulu, HI (2002), pp. 1702–1707.
14. KDD CUP DATA, Available at http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (1999).
15. A. Sung, Ranking importance of input parameters of neural networks, *Expert Syst. Appl.* **15**(3–4) (1998) 405–411.
16. S. Peddabachigari, A. Abraham and J. Thomas, Intrusion detection systems using decision trees and support vector machines, *Int. J. Appl. Sci. Comput.* (2004) 118–134.
17. G. Kambourakis, T. Moschos, D. Geneiatakis and S. Gritzalis, Detecting DNS amplification attacks, *Crit. Inf. Infrastruct. Secur.* **5141** (2008) 185–196.
18. S. D. Paola and D. Lombardo, Protecting against DNS reflection attacks with bloom filters, *Detection of Intrusions and Malware*, *and Vulnerability Assessment*, Vol. 6739 (Springer, 2011), pp. 1–16.
19. S. Rastegari, M. I. Saripan and M. F. A. Rasid, Detection of denial of service attacks against domain name system using neural networks, *Artificial Neural Networks — ICANN* (2011), pp. 118–123.
20. US-CERT, The continuing denial of service threat posed by DNS recursion (v2.0) (2013).
21. R. Chandramouli and S. Rose, Secure domain name system (DNS) deployment guide, Recommendations of the National Institute of Standards and Technology (NIST) (2013).
22. Juniper Networks, "Defending Against Application-Layer DDOS attacks", 2013, Available at http://js-technology.fr/wp-content/uploads/2015/06/DDOS-secure-white-paper.pdf.
23. E. Damon, J. Dale, E. Laron, J. Mache, N. Land and R. Weiss, Hands-on Denial of service lab exercises using SlowLoris and RUDY, in *Proc. 2012 Information Security Curriculum Development Conf.*, Kennesaw, Georgia (2012), pp. 21–29.
24. C. Wheelus, T. M. Khoshgoftaar, R. Zuech and M. M. Najafabadi, A session based approach for aggregating network traffic data — The SANTA dataset, *2014 IEEE Int. Conf. Bioinformatics and Bioengineering* (*BIBE*) — *Workshop on Big Data and Data Analytics Applications*, Boca Raton, FL (2014), pp. 369–378.
25. J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue and K. Nakao, Statistical analysis of honeypot data and building of Kyoto, in *Proc. First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*, New York, NY, USA (2011), pp. 29–36.
26. P. Mockapetris, RFC 1034, Domain Names — Conceptsand Facilities, Internet Standard (1987).
27. R. Zuech, T. M. Khoshgoftaar, N. Seliya, M. M. Najafabadi and C. Kemp, A new intrusion detection benchmarking system, in *Proc. Twenty-Eighth Int. Florida Artificial Intelligence Research Society Conf.* (*FLAIRS*) *2015*, Hollywood, Florida (2015), pp. 252–256.
28. E. B. Claise, Rfc 3954: Cisco systems netflow services export version 9, Cisco Systems (2004).

29. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I. H. Witten, The WEKA data mining software, *SIGKDD Explorations* **11**(1) (2009) 10–18.
30. M. M. Najafabadi, T. M. Khoshgoftaar and A. Napolitano, A comparison of feature selection strategies for identifying malicious network sessions, in *21st ISSAT Int. Conf. Reliability and Quality in Design*, Philadelphia, Pennsylvania, USA (2015), pp. 161–167.
31. M. M. Najafabadi, T. M. Khoshgoftaar and C. Kemp, The importance of representative network data on classification models for the detection of specific network attacks, in *21st ISSAT Int. Conf. Reliability and Quality in Design*, Philadelphia, Pennsylvania, USA (2015), pp. 59–64.
32. M. M. Najafabadi, T. M. Khoshgoftaar and C. Wheelus, Attack commonalities: Extracting new features for network intrusion detection, *21st ISSAT Int. Conf. Reliability and Quality in Design*, Philadelphia, Pennsylvania, USA (2015), pp. 46–50.
33. L. Rokach and O. Maimon, *Data Mining with Decision Trees*: *Theory and Applications* (World Scientific Publishing Co., Inc., 2008).
34. N. S. Altman, An introduction to kernel and nearest-neighbor nonparametric regression, *Am. Stat.* **46**(3) (1992) 175–185.
35. S. J. Russell and P. Norvig, *Artificial Intelligence*: *A Modern Approach*, 2nd edn. (Prentice Hall, 2002).

About the Authors

Maryam M. Najafabadi received her B.S. degree in Computer Science from Isfahan University of Technology and her M.S. degree in Artificial Intelligence from Amirkabir University of Technology in 2008 and 2011, respectively. Currently, she is a Ph.D. candidate in the Department of Computer and Electrical Engineering and Computer Science at Florida Atlantic University. Her research interests include Data Mining, Machine Learning, and Network Security.

Taghi M. Khoshgoftaar is Motorola Endowed Chair Professor of the Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University and the Director of NSF Big Data Training and Research Laboratory. His research interests are in Big Data Analytics, Data Mining and Machine Learning, Health Informatics and Bioinformatics, and Software Engineering. He has published more than 500 refereed journal and conference papers in these areas. He is the conference co-chair of the *IEEE International Conference on Machine Learning and Applications* (*ICMLA 2016*). He is the workshop chair of the *IEEE IRI Health Informatics Workshop* (2016). Also, he is the Co-Editor-in-Chief of the *Big Data* journal.

Amri Napolitano received the Ph.D. and M.S. degrees in Computer Science from Florida Atlantic University in 2006 and 2009, respectively, and the B.S. degree in Computer and Information Science from the University of Florida in 2004. His research interests include Data Mining and Machine Learning, Evolutionary Computation, and Artificial Intelligence.