Proceedings of the Twenty-Ninth International
Florida Artificial Intelligence Research Society Conference

RUDY Attack: Detection at the Network Level and Its Important Features

Maryam M. Najafabadi, Taghi M. Khoshgoftaar, Amri Napolitano, Charles Wheelus
Florida Atlantic University
mmousaarabna2013@fau.edu, khoshgof@fau.edu, amrifau@gmail.com, cwheelus@fau.edu

Abstract

Compared to common DoS/DDoS attacks that are destructive and generate massive traffic, the application layer DoS attacks can be slow-and-low which means they occur at a slow rate and do not generate a massive amount of traffic. These attacks appear legitimate in terms of the protocol rules and rates. These characteristics make the detection of these attacks difficult. In this paper, we study the RUDY (R-U-Dead-Yet) attack which is one of the slow-and-low application layer attack types. RUDY attacks can bring down a server by creating long POST HTTP form submissions to the server at a very slow rate which results in application threads at the server side becoming stuck. The mitigation methods against RUDY attacks are mostly host-based. In this paper, we use a machine learning approach for the detection of RUDY attacks as well as determining the important features for their detection at the network level. The network level detection is scalable and it provides detection for hosts that do not have their own detection mechanism. We extract features from bi-directional instances of the network traffic. We then use an ensemble feature selection approach containing 10 different feature ranker methods in order to extract the most important features for the detection of RUDY attacks at the network level.

Introduction

Denial of Service (DoS) attacks are among the most common attack types in computer networks which makes them a major concern in network security. The basic principle of DoS attacks is to make services on a server unavailable to legitimate users by exhausting the server's resources. This can effectively disable the whole network or organization. Although these types of attacks usually don't result in the theft of information or other security loss, they can cost the targeted machine or network a great loss of time and money.

Early DoS attacks were usually done at the TCP level by flooding the targeted server. The server is flooded by receiving a lot of requests sent by the attacker (Damon et al. 2012). DoS attacks at the TCP level exploit the fact that a lot of servers allocate resources at the initial phase of a TCP three-way handshake. In order to flood the server, the attacker would initiate a large amount of TCP connections to the server and would not complete the required number of handshakes. The server's resources would get saturated by the flood of initiated, unfinished TCP connections. To defend against these attacks, the defenders developed some mitigation methods, such as delaying the buffer allocation until the three-way handshake is completed. Therefore, the attackers looked for other vulnerabilities, such as application layer vulnerabilities.

Application layer DoS attacks have increased during recent years [1], [2]. These attacks usually mimic real human behavior and conform to the protocol which make them harder to detect. Layer 7 of the OSI model (Grigonis 2000), application level, supports network applications and end user processes and it provides services to a network application program. It facilitates programs such as web browsers and email services in transferring data and information across the internet through different protocols (e.g. HTTP, SMTP). Unlike common DoS attacks, application layer attacks require very little resources. While network flooding needs several hundred bots to be performed, these attacks can be activated from a single attacker computer with no need for additional bots. This need to limited resources makes layer 7 attacks an increasingly popular strategy to perform DoS attacks.

HTTP (Hypertext Transfer Protocol) (Fielding et al. 1999) is the foundation of data communication for the World Wide Web (WWW). Since this protocol is a connection oriented protocol, it suffers from similar vulnerabilities to TCP. Once a connection is initiated, the resources are allocated to it by the server and remain allocated while the connection is open. The RUDY attack was developed to take advantages of this vulnerability.

The RUDY attack (K.C, Shetty, and H.R 2014) opens concurrent POST HTTP connections to the HTTP server and delays sending the body of the POST request to the point that the server resources are saturated. The detail of this attack is explained in the RUDY Attack section of this paper. This attack sends numerous small packets at a very slow rate to keep the connection open and the server busy. This low-and-slow attack behavior makes it relatively difficult to detect, compared to flooding DoS attacks that raise the traffic vol-

[1]http://www.gartner.com/newsroom/id/2344217
[2]https://blogs.akamai.com/2015/01/q4-2014-state-of-the-internet—security-report-some-numbers.html

ume abnormally.

Most of the mitigation methods against RUDY attacks are host-based which include monitoring of the resources allocations at the server [3]. It includes real time monitoring of the server resources such as CPU, memory, connection tables and more. The advantages of a network-based mitigation are that such a detection is scalable, it provides security for the hosts that don't have their own security union and it is also proper for the detection of distributed attacks. In this paper, we analyze the internet traffic at the network level by using machine learning methods to detect the RUDY attacks. We also expand our analysis by further investigating on what features are more important for the detection of this attack at the network level.

Recently, machine learning methods have been applied for intrusion detection applications (Tsai et al. 2009), (Najafabadi et al. 2014). Machine learning methods are able to extract patterns and similarities in the data in order to do classification tasks, e.g. the detection of attack from normal traffic in the network data. Extracting discriminative features is a very critical step in any machine learning task. Usually, this step is done through feature engineering. Feature engineering is the process of using domain knowledge about the data in order to extract potentially-useful features. However, not all the extracted features from feature engineering are equally important for building the predictive models. Some feature can be redundant or irrelevant. Feature selection is a pre-processing step in applying machine learning methods to build the predictive models. The goal is to increase the overall effectiveness of the model by removing redundant and irrelevant features without adversely affecting the classification performance (Najafabadi, Khoshgoftaar, and Seliya 2016).

In this paper, we use machine learning algorithms in order to build predictive models for the detection of RUDY attacks at the network level. We also use feature selection methods to determine which of the defined features are more important for the detection of RUDY attacks. Using less features increases the efficiency of the detection method and provides faster analysis. In addition, the selected features reveal the important characteristics of the RUDY attack that are beneficial for its detection.

We used the SANTA dataset (Wheelus et al. 2014) for our experiments. To determine the important features for the detection of the RUDY attack, we applied 10 different feature ranker methods and aggregated their results. We applied three classification methods to build the predictive models. Finally, we applied ANOVA (Berenson, Goldstein, and Levine 1983) to compare the predictive models performances when the whole feature set with those models when the selected feature sets are used. Our results show that the selected feature set provides very similar performance results to the case where all the features are used in the building of the predictive models. The selected features include the features which represent three main characteristics, including the traffic size, the self-similarity between packets and the traffic velocity.

The rest of the paper is organized as follows. The Related Work section contains the previous research regarding feature selection in the intrusion detection domain. In the Case Study section we explain the SANTA data. In the Machine Learning Methods section we explain our machine learning approach which includes the classifiers we used in our analysis, the feature selection methods and the performance metrics used. The Results section presents the results of our work. Finally the Conclusion section concludes the paper and presents the topics for the future works.

Related Work

Elimination of redundant and insignificant features in an intrusion detection task leads to a simplified and faster analysis. Reducing the complexity may also provide better accuracy. The problem of feature selection in intrusion detection has been studied in different works.

In (Sung and Mukkamala 2002) two wrapper-based feature selection methods are used: performance-based ranker and SVM-specific ranker. The experiments are done on the DARPA dataset [4]. In the performance-based ranker, one input feature is eliminated at a time, the classifier is then trained and tested. The classifier's performance is compared with that of the original classifier which uses all the features. Based on the performance comparison, the importance of the feature is identified by a set of rules and each feature is categorized as being "important", "secondary" or "unimportant". In the SVM-specific feature ranker, the features are ranked based on their contribution in the support vector decision function. The procedure then calculates weights from the support vector function and ranks the importance of the features by the absolute values of the weights. According to the feature ranks, they are again categorized as "important", "secondary" and "unimportant". SVM and Neural Networks are used to build the classification models and test them by using only the important features. Their results show that using important features for each class (attacks and normal) gives the best performance. The testing time decreases, the accuracy for the normal class increases while the performance of the other classes are similar to the performance of using the whole feature set. They listed the features which were selected as important by both feature selection methods, however, the important features for each specific class are not presented.

Onut and Ghorbani (Onut and Ghorbani 2007) proposed a ranking mechanism to evaluate the effectiveness of different features for the detection of different types of attacks. Their statistical feature ranking process is designed as a 3-tier method. The ultimate goal is to calculate the probability of each individual feature being able to detect one of the main types of attacks defined in the used dataset (DARPA). The higher the probability, the better the feature is for the detection of attacks. Once all the probabilities are computed, the features are ranked based on their corresponding probability values. The top selected features for the DoS attack

[3]https://www.incapsula.com/ddos/attack-glossary/rudy-r-u-dead-yet.html

[4]http://www.ll.mit.edu/ideval/data/

are the ones correlating to Internet Control Message Protocol (ICMP) such as the number of ICMP bytes sent by the source IP or the number of ICMP packets sent by the source IP.

Bhoria et al. (Bhoria and Garg 2013) used the NSl KDD dataset [5] to find the relevant set of features for the detection of DoS attacks. They did not use any specific feature selection method to select the features. Three sets of features are formed and compared together. The first set includes all the 41 features in the KDD dataset. The second set includes 28 features from the 41 features in the KDD dataset which are selected by combining basic and time based traffic features. The third set includes 8 features; however, the authors did not explain specifically how these feature sets are chosen. They applied the decision tree algorithm on these three sets of features along with cross validation to compute the performance values. Their results show that the feature set with 8 features provides the best classification accuracy as well as the shortest classification time.

In this paper, we apply feature selection methods in order to find the important features for the detection of RUDY attacks based on network traffic. The selected feature set not only provides more efficient predictive models for the detection of RUDY attacks, it also shows what kind of features should be focused on more for the detection of these attacks, as well as quite likely for other attacks with similar behavior to RUDY attacks.

The RUDY Attack

The Rudy attack is a slow rate application layer DoS attack. This attack attempts to open a relatively low number of connections to the targeted machine over a period of time and keeps them open as long as possible to keep the machine's resources suspended. Eventually these open sessions exhaust the targeted machine and make it unavailable to the legitimate users. The low and slow traffic associated with this attack makes it hard for the traditional mitigation tools to detect.

RUDY attack exploits a weakness in the HTTP protocol which was originally designed to provide service to the users with very slow rate traffic (such as dial up users). RUDY attacks take advantages 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 sends a legitimate HTTP POST request with an abnormally long "content length" header field. Then it starts injecting the content at a very slow rate (usually one byte at a time) to the server. The long "content length" field prevents the server from closing the connection. The information is not only sent in very small chunks, but also it is sent at a very slow rate. On the server side, this traffic creates a massive backlog of application threads, while it does not close the connection because of the long "Content-Length" field. The attacker launches simultaneous connections to the server. Since the server is hanging while waiting for the rest of these HTTP POST requests, eventually its connection table gets exhausted and the server crashes.

[5]http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

Case Study Data

SANTA Dataset

The SANTA dataset (Wheelus et al. 2014) is collected from an operational network of a commercial Internet Service Provider (ISP). The network data includes a mixture of varying types of internet traffic. The ISP hosts a wide range of server traffic that is accessed by users from across the internet, including web servers, email servers, and other various internet services that are common to internet providers. The customer networks accessing the internet through the ISP network generate traffic such as email, browser and all other types of internet traffic that an average business might generate in the course of day-to-day operation.

The data is collected from two border routers that connect the ISP network to the outside world, therefore the collected traffic does not include the internal traffic (from one internal host to another internal host). Since the collected data only includes the border traffic the features are defined based on two concepts: inbound and outbound traffic. Inbound traffic is the traffic which is targeting the network and the packets are originating from the hosts outside of the network. Outbound traffic is the traffic which is leaving the network and the originating hosts are the ones in the network which are sending the packets to the outside world. The two concepts are used in order to define bi-directional network instances which include both inbound and outbound packets. This allows the extracted features to include inbound and outbound traffic information simultaneously for one network instance. The RUDY attack traffic was produced through penetration testing.

Network instance and extracted features

Each instance is a network session in the SANTA dataset , (Najafabadi, Khoshgoftaar, and Wheelus 2015). A network session is constructed by grouping the corresponding inbound and outbound Netflows. For example an inbound HTTP request would be paired with the corresponding HTTP response, which is outbound traffic, to make a session. Once a session is built, the features are extracted from it. A Netflow is a uni-directional aggregation of network packets. Conversely, a session is a bi-directional aggregation of the network packets. Like communication between people, the network data context is largely dependent on hearing both sides of the conversation. Similar to a conversation, in the network traffic the inbound traffic and the outbound traffic typically result from one another. The context can suffer from analyzing inbound and outbound Neflows separately. The concept of "session" incorporates both sides of a network conversation (inbound and outbound Netflows) in the definition of a network instance.

Three main categories of features are defined along with features that are more analogous to those used in other datasets, such as DARPA. The three main categories are: self-similarity, periodicity and velocity related features. Self-similarity represents the similarity between the packets. It is calculated by computing the variance of size of

Table 1: Description of features extracted from sessions

Feature Name	Description
Protocol	Transmission protocol
IO match	Whether the inbound Netflow has an associated outbound Netflow record (Boolean)
Duration	The elapsed time, from the earliest of the associated inbound or outbound Netflow until the end of the later Netflow
Bytes	Total size for the session in bytes
Packets	Total number of packets in the session
Inbound Session Convergence	Self-similarity of the inbound packets in the session is determined by examining the variance in size of the inbound packets
Outbound session convergence	Self-similarity of the outbound packets in the session is determined by examining the variance in size of the outbound packets
Repetition	The ratio of the number of most common packet
Periodicity	Standard deviation of packet size (bytes/packets) within the session
Inbound velocity pps	Velocity of inbound traffic measured in packets per second
Inbound velocity bps	Velocity of inbound traffic measured in bits per second
Inbound velocity bpp	Velocity of inbound traffic measured in bytes per packet
Outbound velocity pps	Velocity of outbound traffic measured in packets per second
Outbound velocity bps	Velocity of outbound traffic measured in bits per second
Outbound velocity Bpp	Velocity of outbound traffic measured in bytes per packet
RIOT packets	Ratio of inbound to outbound traffic measured in packets
RIOT Bytes	Ratio of inbound to outbound traffic measured in bytes
Flags	Cumulative OR of all the TCP flags seen in this session
Class	Class label (Attack or Normal) associated with the Netflows within the session

the inbound and outbound packets in one network instance, i.e. session. To measure the periodicity, the variance of the difference between timestamps of the inbound or outbound packets in one session is calculated. Velocity features are defined by providing bits per second, packets per second, and bytes per packet for both inbound and outbound traffic. The descriptions of the extracted features are shown in Table 1.

Applying Machine Learning Methods

Feature Selection Methods

Feature selection (Guyon and Elisseeff 2003) aims to examine a dataset to find which features are most important to the class label. In this study, we apply feature ranking methods. Feature ranking methods use different techniques to assign a score to each feature. The features are ranked based on these scores in order from best to worst. The top N features are then selected as the results of the feature selection method. There is no standard for deciding upon a specific number of features (N), therefore this decision should be left to the discretion of the practitioners. In this study, we apply 10 different feature ranker methods on our dataset. These methods provide ten different ranking lists of the features.

The feature ranking methods used in this study are: F-Measure (F), Geometric Mean(GM), Kolmogorov-Smirnov statistic (KS), Mutual Information (MI), Area Under the Receiver Operating Characteristic Curve (ROC), Fisher Score (FS), Signal to Noise Ratio (S2N), Chi Squared (CS), Information Gain (IG), Gain Ratio (GR). These can be divided into three groups: threshold based, first order statistics, and common literature techniques. Threshold-based feature selection techniques use the feature values as the posterior probabilities to estimate the classification error (This includes F, GM, KS, MI, ROC). First order statistics based methods use first order statistical measures such as mean and standard deviation to measure the relevance of the features (this includes FS and S2N) and the techniques commonly used in the literature (CS, IG and GR).

By aggregating the feature ranking lists and expert analysis we selected seven features. Once the features are ranked in ten different ranked lists, we counted, for each feature, how often the feature appears at the 1st-place, at the second-place and so on among all ten ranking lists. When sorting the features based on how frequently they appear toward the top of the ten ranked lists, we realized that there are some natural cut-offs where some features tend to appear more at the beginning of the ranked lists and the others tend to appear at the end of the ranked lists. Based on this information, we determined that the features that appear more than 4 times at the top seven positions in the rank lists are of our greatest interests. Further investigations can consider more features, however we decided to go with 7 features since we only had 18 features to begin with and choosing more features seems to obviate the purpose of feature selection.

Classifiers

To build the predictive models, we chose three classification algorithms : K-Nearest Neighbor (K-NN) and two forms of C4.5 decision trees (C4.5D and C4.5N). These learners were all chosen due to their popular use in machine learning applications as well as their relative ease of computation. Using these learners provides a broader analysis from a data mining point of view. We built all models using the WEKA machine learning toolkit (Hall et al. 2009).

K-nearest-neighbors or K-NN is an instance learning algorithm. K-NN stores the training instances in the memory. When predicting the class of a new instance, its distance or similarity to all the training instances stored in the memory is calculated. The algorithm uses the K (in our study, K=5) closest instances to the test instance to decide its class.

C4.5 decision tree (implementation of the J48 decision tree in WEKA) is a tree-based learning algorithm. In these algorithms a decision tree is built based on the training data.

Each branch of the tree represents a feature in the data which divides the instances into more branches based on the values which that feature can take. The leaves represent the final class label. The C4.5 algorithm uses a normalized version of Information Gain to decide the hierarchy of features in the final 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.

Performance Metrics

We use Area Under the receiver operating characteristic Curve (AUC) as well as True Positive Rate (TPR) and False Positive Rate (FPR) as the evaluation metrics. The Receiver Operating Characteristic (ROC) curve is a graph of the TPR vs FPR. In the current application, TPR is the percentage of the RUDY attack instances that are correctly predicted as Attack. FPR is the percentage of the Normal data which is wrongly predicted as Attack by the model. The ROC curve is built by plotting TPR vs FPR as the classifier decision threshold is varied. The area under the ROC graph is calculated as the AUC performance metric. A higher value of AUC means higher TPR and lower FPR which is preferable in an intrusion detection applications.

To evaluate the performance values we used 5 fold cross validation. In 5 fold cross validation, the data is divided into 5 non-overlapping parts (folds). In each iteration, one part is kept out as the test data and the other four parts are used as the training data. The final performance values are calculated by aggregating the performance values of the models being tested on each of 5 parts of the data. In order to decrease the bias of randomly selected folds, we applied four runs of 5 fold cross validation to provide each performance value.

Results

We selected seven features by using the ensemble method explained in the Feature Selection Methods sub-section. By looking at the ranking lists side by side and the frequency of different features appearing in different positions in the ranked lists, we decided to select the features that appear more than 4 times at the 7 first positions of all the ranked lists as the final selected features. The selected features and their frequency in the first 7 positions of the ten ranked lists are shown in Table 2.

We applied three classification algorithms on the data with the whole feature set and with the 7 selected features. The performance values are achieved through 4 runs of 5 fold cross validation. The cross validation results on the whole feature set and on the selected feature set are shown in Table 3 and Table 4 respectively. The AUC values being more than 0.99 plus the high TPR and low FPR values indicate that the machine leaning methods perform very well in the detection of RUDY attacks with the whole or selected feature sets.

We applied ANOVA analysis in order to compare whether the selection of feature sets significantly affects the performance. ANOVA analysis determines whether there is significant difference between the mean value of the independent groups. We applied one-way ANOVA analysis with the factor being whether the whole or selected feature set is used. We chose a significance level of 5% for this ANOVA analysis and a “Prob>F” score of less than 0.05 is considered to be statistically significant. The ANOVA results are shown in Table 5. The results show that there is no significant difference in the selection of feature sets which means the selected feature set performs very similar to the whole feature set in the detection of RUDY attacks.

Table 2: Selected Features by the ensemble of rankers.

Feature	Number of occurrences
Outbound session convergence	5
Inbound session convergence	5
Packets	5
Bytes	6
RIOT Bytes	6
Outbound velocity bpp	6
Outbound velocity bps	7

Table 3: Cross Validation Results on the whole feature set

Classifier	AUC	TPR	FPR
C4.5N	0.9988	0.9873	0.000282
C4.5D	0.9940	0.9866	0.000307
5-NN	0.9999	0.9883	0.000316

Table 4: Cross Validation Results on the selected feature set with 7 features

Classifier	AUC	TPR	FPR
C4.5N	0.9983	0.9907	0.00029
C4.5D	0.9996	0.9890	0.00041
5-NN	0.9944	0.9897	0.000265

By looking at the selected features shown in Table 2 we observe that the features are in correlation with the RUDY attack behavior explained in RUDY Attack section. In a RUDY attack scenario, the attacker is sending small packets in a slow rate and the server is just responding TCP acknowledgement packets to the incoming slow rate attack packets which makes the response, i.e, outbound traffic, have a small number of bytes per packets. This also affects the speed in which the packets are sent out, i.e. bits per second/packets. These characteristics can be represented in Outbound velocity bps and Outbound velocity bpp features. On the other hand, the small packet sizes sent by the attacker and the short responses can be represented in the overall size of a session (Bytes and Packets) features as well as RIOT Bytes which shows the ratio of inbound to outbound traffic in bytes representing the relative size of inbound to outbound traffic.

Another important characteristic of the RUDY attack is the self-similarity between the request packets sent by the attacker and the response packets sent by the server. The two features, Inbound session convergence and Outbound session convergence, represent the self-similarity between inbound and outbound packets in a session respectively. So it is expected to see these features among the important features selected for the RUDY attack.

Table 5: ANOVA Results

	Df	Sum Sq	Mean Sq	F value	Prob>F
Whole/selected feature set	1	6.00e-07	6.190e-07	0.04	0.843
Residuals	118	1.85e-03	1.568e-05		

Conclusion

The RUDY attack is an application layer HTTP denial of service attack which exploits the server behavior of supporting users with slower connections. It injects the application POST body traffic at a very slow rate to the server to make the application threads wait for the end of a non-ending POST. By initiating simultaneous POST connections, the RUDY attack exhausts the server connection table and leads to a denial of service attack. Most of the mitigation methods for this attack take place at the host level which includes monitoring the servers resource consumption, such as CPU and memory, as well as behavior analysis of open server connections. In this paper, we provide a machine learning approach for the detection of the RUDY attack at the Network level. We used feature selection methods to determine which features are more important for the detection of the RUDY attack at the network level. We used an ensemble feature selection approach including 10 different feature ranking methods to investigate what features are more important for the detection of this attack. Based on our results the features related to the traffic size, self-similarity between traffic packets and the velocity related features are more important for the detection of RUDY attacks. These features provide a very good classification performance for the detection of RUDY attacks which based on our ANOVA results is not significantly different from the classification performance of the whole feature set. This information can be used in introducing features for the detection of other types of attacks similar to the RUDY attack. In future work, we would like to study more application level DoS attacks and the important features for their detection.

Acknowledgment: We acknowledge partial support by the NSF (CNS-1427536). Opinions, findings, conclusions, or recommendations in this material are the authors' and do not reflect the views of the NSF.

References

Berenson, M. L.; Goldstein, M.; and Levine, D. 1983. *Intermediate Statistical Methods and Applications: A Computer Package Approach 2nd Edition*. Prentice Hall.

Bhoria, M. P., and Garg, K. 2013. Determining feature set of dos attacks. *International Journal of Advanced Research in Computer Science and Software Engineering* 3(5):875–878.

Damon, E.; Dale, J.; Laron, E.; Mache, J.; Land, N.; and Weiss, R. 2012. Hands-on denial of service lab exercises using slowloris and rudy. In *Proceedings of the 2012 Information Security Curriculum Development Conference*, InfoSecCD '12, 21–29. New York, NY, USA: ACM.

Fielding, R.; Gettys, J.; Mogul, J.; Frystyk, H.; Masinter, L.; Leach, P.; and Berners-Lee, T. 1999. Hypertext transfer protocol – http/1.1. Request for Comments: 2616.

Grigonis, R. 2000. *Computer Telephony Encyclopedia*. C M P Books, 1st edition.

Guyon, I., and Elisseeff, A. 2003. An introduction to variable and feature selection. *J. Mach. Learn. Res.* 3:1157–1182.

Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; and Witten, I. H. 2009. The weka data mining software: An update. *SIGKDD Explor. Newsl.* 11(1):10–18.

K.C, P.; Shetty, S.; and H.R, N. 2014. A comprehensive study on distributed denial of service attacks and defense mechanisms. *International Journal of Computer Applications (0975 8887)* 15–20.

Najafabadi, M. M.; Khoshgoftaar, T. M.; Kemp, C.; Seliya, N.; and Zuech, R. 2014. Machine learning for detecting brute force attacks at the network level. In *Bioinformatics and Bioengineering (BIBE), 2014 IEEE International Conference on - workshop on Big Data and data analytics applications*, 379–385.

Najafabadi, M. M.; Khoshgoftaar, T. M.; and Seliya, N. 2016. Evaluating feature selection methods for network intrusion detection with kyoto data. *International Journal of Reliability, Quality and Safety Engineering*.

Najafabadi, M. M.; Khoshgoftaar, T. M.; and Wheelus, C. 2015. Attack commonalities: Extracting new features for network intrusion detection. In *Proceedings of the 21st ISSAT International Conference on Reliability and Quality in Design*, 46–50.

Onut, I.-V., and Ghorbani, A. A. 2007. Features vs. attacks: A comprehensive feature selection model for network based intrusion detection systems. In *10th International Conference, ISC 2007*, volume 4779, 19–36. Springer.

Sung, A. H., and Mukkamala, S. 2002. Feature selection for intrusion detection using neural networks and support vector machines. *Transportation Research Record Journal of The Transportation Research Board 1822* 33–39.

Tsai, C.-F.; Hsu, Y.-F.; Lin, C.-Y.; and Lin, W.-Y. 2009. Review: Intrusion detection by machine learning: A review. *Expert Systems with Applications* 36(10):11994–12000.

Wheelus, C.; Khoshgoftaar, T. M.; Zuech, R.; and Najafabadi, M. M. 2014. A session based approach for aggregating network traffic data - the santa dataset. In *Bioinformatics and Bioengineering (BIBE), 2014 IEEE International Conference on - workshop on Big Data and data analytics applications*, 369–378.