Detection of SSH Brute Force Attacks Using Aggregated Netflow Data

Maryam M. Najafabadi, Taghi M. Khoshgoftaar, Chad Calvert, Clifford Kemp
Email: {mmousaarabna2013@fau.edu; khoshgof@fau.edu; ccalver3@fau.edu; cliffkempfl@gmail.com}
Florida Atlantic University, Boca Raton, FL 33431

***Abstract*—The SSH Brute force attack is one of the most prevalent attacks in computer networks. These attacks aim to gain ineligible access to users' accounts by trying plenty of different password combinations. The detection of this type of attack at the network level can overcome the scalability issue of host-based detection methods. In this paper, we provide a machine learning approach for the detection of SSH brute force attacks at the network level. Since extracting discriminative features for any machine learning task is a fundamental step, we explain the process of extracting discriminative features for the detection of brute force attacks. We incorporate domain knowledge about SSH brute force attacks as well as the analysis of a representative collection of the data to define the features. We collected real SSH traffic from a campus network. We also generated some failed login data that a legitimate user who has forgotten his/her password can produce as normal traffic that can be similar to the SSH brute force attack traffic. Our inspection on the collected brute force Netflow data and the manually produced SSH failed login data showed that the Netflow features are not discriminative enough to discern brute force traffic from the failed login traffic produced by a legitimate user. We introduced an aggregation of Netflows to extract the proper features for building machine learning models. Our results show that the models built upon these features provide excellent performances for the detection of brute force attacks.**

***Keywords*-Intrusion Detection; Brute Force; Machine Learning; Aggregated Netflows;**

I. Introduction

The rapidly growing progress in internet-based technology has brought tremendous benefits to our society. Communications and other numerous amounts of services that the internet provides (social media, social networks, online banking, online booking, online shopping, etc.) have brought convenience to our every day lives. On the other hand, the internet has its downside too. A lot of threats are created every day by individuals and organizations to attack computer networks to steal private information and data. This information can be very critical and sensitive, such as social security or bank account information. This has created the need for security technologies to secure users' information and provide reliable computer network environments. In that regard, intrusion detection plays an important role for detection of attacks and securing computer networks.

An Intrusion Detection System (IDS) monitors and analyzes network related data in order to detect malicious activities [11], [17]. The intrusion detection can occur at the host level or at the network level [5]. It depends on the source of data that needs to be monitored and analyzed by the IDS. In a host-based IDS, the network traffic that enters and exits the host is analyzed. Other kinds of data that are available at the host level, such as log information, can also be analyzed to detect the attacks. In a network-based IDS, the traffic that enters and exits a computer network is monitored and analyzed. In this paper, we focus on network-based intrusion detection.

SSH brute force attack is a common type of attack in which the attacker tries to gain access to a remote machine by performing authentication attempts. These attacks are performed by systematically checking all the possible passwords until the correct one is found. Human-chosen passwords are inherently weak. People tend to select simple passwords because they are easier to remember. Sometimes, they don't change the machine's default password or they simply use the user name as the password.

A machine compromised by a brute force attack can cause serious damage by joining botnets, distributing sensitive information, participating in distributed attacks, etc. Brute force attacks are still a prevalent threat for computer networks. The Cisco 2014 annual security report [3] explains that "Although brute-force login attempts are by no means a new tactic for cybercriminals, their use increased threefold in the first half of 2013". In a survey reported by the Ponemon 2014 SSH security Vulnerability Report [4], 51% of the respondent companies have been compromised by SSH brute force attacks in the last 14 months.

At the host level, brute force attacks can be detected by inspecting access logs. If the number of failed logins exceeds a pre-defined threshold in a specific time interval then a brute force attack alert is fired. The host-based detection is not scalable. In addition, if the host does not have its own detection mechanism and gets infected by the brute force attack, it can be a serious danger for the whole network. Providing detection at the network level not only makes the detection mechanism scalable, it also provides security for the hosts that do not have their own detection systems.

In this paper, we use aggregated Netflow data along with a machine learning approach for the detection of SSH brute force attacks at the network level. Machine learning methods can be used to automatically extract patterns from network data and unlike signature based methods, there is no need to manually extract the patterns [13], [14]. We explain our feature extraction process which is based on using domain knowledge and the analysis of the representative collected data

978-1-5090-0287-0/15 $31.00 © 2015 IEEE
DOI 10.1109/ICMLA.2015.20

in building the predictive models. Extracting discriminative features is a very important step in any machine learning application. Even simple machine learning algorithms can provide good performance results on a well-defined feature set. Conversely, a sophisticated machine learning method can fail in providing good performance results when applied on a poorly defined feature set.

We took into account domain knowledge for analyzing a real collection of network traffic to define discriminative features. We collected real network traffic from a campus network with almost 300 users. Our network operators labeled real brute force attacks in the collected traffic. To analyze the data we extracted Netflows from the data. A Netflow is a summary of network packets that share some common characteristics. In recent years, Netflow analysis has increased in intrusion detection studies because it provides faster analysis than packet analysis. In addition, since no packet payload is used in the construction of a Netflow, Netflow analysis can be applied on encrypted traffic as well.

In addition to the normal data which is already included in the collected network traffic, we also produced some normal data which is similar to the attack data. We generated some failed login attempt data that represents the kind of traffic which is produced when a legitimate user has forgotten/miss-spelled his/her password. The final data is collected over eight days. It also includes failed login data produced by penetration testing. These characteristics make the collected data to include most of the networking and communicating scenarios seen in a real computer network. In other words, the collected data is representative of different network traffic scenarios.

A representative collection of the traffic makes it possible to compare the attack data with normal data in order to introduce the features that can better discriminate these two from one another. If the collected traffic is not representative and does not contain various types of normal network traffic, even a simple feature set might look to work well for discriminating normal and attack traffic. Using such a feature set in building the predictive models can lead to poor performance when new types of traffic show up during the detection time. The collected data should contain a representative amount of normal traffic as well as the normal traffic that might be similar to the attack traffic. This makes it possible to do more in depth comparisons of the normal and attack data and define the features in a way that can be truly discriminative for recognizing attacks from normal traffic.

To introduce the features that can discriminate between normal SSH and brute force attack data, we inspected the collected data by comparing brute force Netflows with normal Netflows which also included failed logins produced by our network operators. Our analysis showed that the brute force Netflows are very similar to the failed login attempts Netflows by legitimate users. This means the Netflow features are not discriminative enough to differentiate between brute force and normal data. We decided to extract features from an aggregation of Netflow data. The aggregation of the Netflows can reflect the number of attempts taken which is the main difference between the brute force traffic and the legitimate failed login attempts traffic. Our results show that, using these features, classifiers provide very good performances for the detection of brute force attacks.

The rest of the paper is organized as follows: Section II provides some related works. Section III explains the main steps in our approach. Section IV provides details about our case study dataset and explains the process of aggregating Netflow data and the features extracted. Section V discusses the machine learning methods utilized. Section VI provides the results of our experiments. Finally, in Section VII we conclude the paper.

II. Related Work

In recent years, machine learning methods have increasingly been used for intrusion detection [19], [16]. Zamani et al. [21] explained that since traditional intrusion detection techniques (such as firewalls, access control mechanism, etc.) have limitations in the detection of sophisticated attacks like denial of service, machine learning methods have started to be used in the problem of intrusion detection with the hope to improve detection rates and adaptability.

Narayan et al. [15] proposed a hybrid classification method utilizing both Naïve Bayes and C4.5 decision trees for intrusion detection. This approach was established with the intent of being used as a pre-processor to an IDS to reduce high dimensionality and time of computation. It utilized the KDD Cup 99 dataset and proposed a three phase approach. Phase one implemented a misuse detection model which was first built using C4.5 and then normal training data was decomposed from the resulting model. From here, multiple single-class Naïve Bayes models were created from the decomposed data subsets for classification. Lastly, a post processing phase using a majority voting algorithm (N-voting) was used to reduce false alarm rates. Other machine learning algorithms were analyzed for the three proposed phases, but the C4.5 and Naïve Bayes classifiers proved to have the greatest performance.

A method was implemented by Haque and Alkharobi [8], using the NSL KDD Cup dataset and aims to be used alongside an IDS to augment its prediction performance for upcoming data. Naïve Bayes, Random Forest, and PART were used to classify the KDD dataset along with various feature reducing schemas. Their results showed that by utilizing the aforementioned classifiers, they experienced a significant improvement in overall execution time.

Tang and Cao [18] aimed to analyze the efficiency and performance of both Neural Networks (NN) and Support Vector Machines (SVM) when classifying varying types of attacks. Their work considered the following four attack types: Probing, Denial of Service (DoS), User to Root (U2R), and Remote to User (R2U). Analysis was conducted using the KDD Cup 99 dataset. In their findings, SVM was superior to NN in false alarm rates as well as detection rates with all four attacks. NN was found to outperform SVM only in accuracy.

Some studies have been done for the detection of brute force attacks. Work conducted by Javed and Paxson [10] outlines a method for the detection of distributed SSH brute force attacks. A two-phase approach is considered to first identify attack epochs in order to determine if an attack has occurred, and secondly, to classify the hosts as either participants or non-participants. Their method was evaluated using 8 years of SSH login records collected at the Lawrence Berkley National Laboratory. The first phase consists of an aggregate site analyzer that monitors the probability distribution of the parameter for excessive change and then denotes such changes as possible attacks. The second phase of this method is accomplished by denoting common characteristics of legitimate users, singleton brute-forcers and low-rate distributed brute-forcers. The results found that, during the aggregation analyzer phase, 99 attack epochs were detected with 9 false alarms reported, i.e., a nearly 10% false positive rate. Classification of the attack host proved better results, detecting a total of 9,306 hosts with only 37 benign hosts misclassified.

Network flows have also been utilized in the detection of brute force attacks. By analyzing information found in the network flow, the amount of data that must be mitigated is lessened, making attacks more visible. Drasar et al. [6] analyzed the impact that this methodology has based on different types of brute force attack detection methods. In their work, five methodologies were examined; signature-based, similarity-based, automated action detection based on a time window heuristic and a Fourier transform, and entropy time changes. Each approach was deemed successful in detecting attacks but each came with their own considerations. Signature-based approaches were simple but other data sources were required to help eliminate false positives. Similarity-based methods ended up being overly generic, and therefore, they were highly dependent on the parameters identified. Both automated detection methods were capable of detecting constant periodic traffic alluding to ongoing attacks. The entropy time series method was very scalable and fast to deploy.

Hofstede et al. [9] also utilized network flows to aid in the detection of SSH attacks such as brute force. The authors propose the use of their open source IDS SSHCure which has been updated to incorporate detection algorithms capable of identifying if an attack was successful. Their algorithms are based upon a Hidden Markov Model which assumes that attacks feature one or more of three attack phases: scan, brute force, and/or compromise. Traffic possibly consisting of brute force attacks is identified by considering hosts sending flows with a particular range of packets per flow (PPF). Their results showed that their IDS was capable of close to 100% accuracy, however, it was observed that retransmissions can cause false positives. This is due to the fact that retransmissions are not contained in Netflow, except for presence of increased packet and byte counts.

In this paper, we apply machine learning for the detection of brute force attacks by using Netflow data. We use the aggregation of Netflows to extract the features that are more discriminative than normal Netflow features to discriminate between brute force and normal SSH data.

III. Our Approach

To build the predictive models for the detection of SSH brute force attacks at the network level we followed these steps:

1- Collecting representative network traffic.
2- Analyzing the collected data and using the domain knowledge to extract features.
3- Using machine learning algorithms to build the predictive models for the detection of attacks.

Producing network traffic is an essential step in any intrusion detection study. A lot of previous work in the application of intrusion detection have been done on the KDD Cup 99 dataset [1]. This dataset is outdated and does not reflect the current trends in today's computer networks. It is simulated network traffic and does not contain the noise and randomness seen in real network traffic. The models built on such data do not provide the same performance in a real computer network.

To fulfill the gap between intrusion detection studies and the actual deployment of these systems in real computer networks, real network traffic should be used to build the predictive models. In addition, our previous study [12] has shown that the collected data has to be representative enough to provide realistic results that are applicable on real computer networks. Representative network traffic means the data includes different scenarios seen in real computer networks, such as data streaming, interactive traffic, etc.

We collected real network traffic from a campus network. The Netflows were extracted from this data. Our network operators labeled real brute force attack Netflows. Section IV explains the details of our data collection process for providing a representative network traffic.

After the data is collected, the next step is to define discriminative features for the task of building machine learning models. To define the features that are powerful enough to discriminate between a brute force attack and normal SSH traffic we decided to compare the collected brute force Netflows with the normal Netflows. By looking at the data, we realized that the number of packets and the number of bytes in the Brute force Netflows are different from the normal SSH Netflows. Compared to the upload/download and even successful login traffic, the brute force attack Netflows had a relatively fewer number of packets and bytes.

Although, it appeared that Netflow features are able to discriminate between brute force attacks and normal SSH traffic, we decided to compare brute force traffic with the failed login data that a normal user who has forgotten his/her password will produce. The reasons is that a brute force attack is a sequence of attacker's failed login attempts to sign in to the user's account until the correct password is found. Therefore, it is expected that a brute force attack might be similar to failed login data. The difference is that in legitimate failed login data the number of attempts is limited, while in the brute force attack the number of attempts is high.

The comparison between the brute force data and the failed login data showed that the failed login Netflows produced by the legitimate users can be very similar to the failed login data produced by the brute force attack. The reason is that most of the time, the attackers change their source port when sending a new login attempt. This causes a new Netflow to be generated for each login attempt which is similar to the login attempts Netflows produced by legitimate users.

Based on the comparing brute force data with failed login data, Netflow features are not enough to discriminate the brute force data from the normal failed login data. We decided to provide new features. The main difference between brute force attacks and normal failed login data is the number of failed login attempts made. In the brute force attack data the number of attempts is more than in the data from failed logins by a legitimate user. To reflect this characteristic in our features, we decided to extract the features not from the Netflows, but from an aggregation of Netflows which is independent of the sender IP's source port. Altering the source port in producing attack attempts makes each attack attempt produces a different Netflow. In fact, we are aggregating the attacker's attempts in one single instance by aggregating the Netflows with the same characteristics but different source ports. Such an attack instance has more numbers of Netflows than an instance related to failed login attempts by a legitimate user.

We aggregated all the Netflows with the same source IP, destination IP and destination port in one single instance in 5 minutes intervals. We extracted the features from each aggregated instance. The extracted features are explained in more details in Section IV. After the features are extracted, four different machine learning algorithms are applied to build the predictive models. The used machine learning methods are explained in Section V.

IV. Case Study Data and Aggregation of Netflows

Data was collected from a live campus network utilized by both students and faculty. Students specifically used the network to upload and download assignments via SSH connections. The server was configured with one public IP allowing traffic from both local and outside users. Traffic was collected over a period of eight days through the use of port mirroring with Cisco switches and in some instances by placing a network tap (nTAP).

Amongst the natural traffic being captured, penetration testing was also done to simulate failed login traffic. This traffic was generated to represent users who have possibly forgotten their password and would attempt multiple logins before either successfully logging in or giving up. A script was run on the network which would generate several failed logins before ultimately giving up and quitting. This script was designed in such a way as to not trigger a Snort alert for brute force. Since certain Snort rules will flag any communication that attempts more than five unsuccessful logins in one minute, the script made sure to stop before this threshold and wait at least a minute before attempting more logins.

Our capture machine was configured with the Security Onion IDS (v.12.04.5.2) and utilized Snort (v.2.9.7.3) to perform full packet captures (pcaps) of all data passing through the network daily. Once the initial data capture was completed, Wireshark was utilized to merge all daily pcaps into a single capture and filter out unneeded communications such as broadcast, IPv6, UDP, or ARP traffic. Next, Snort was run against the resulting pcap to produce an alert file to be used for labeling. This alert file would help to identify which Netflows were associated with various attack types. Once the alert file had been produced, the pcap was run through SiLK [2] to extract the necessary Netflow information and apply into the csv file format for labeling and analysis.

Netflow, also referred to as session data, represents a high-level summary of network conversations. A Network flow record is identified based upon the standard 5-tuple attribute set that makes up a conversation: source IP, destination IP, source port, destination port, and transport protocol. Based upon which Netflow standard is being implemented, other attribute fields can also be produced. As stated prior, SiLK was used in our experiments to extract Netflows and utilizes the IPFIX standard for flow generation.

Once the Netflow csv and alert file had been generated, labeling was conducted by attempting to associate each Netflow to an alert fired by Snort. If a Netflow aligned with the time-stamp of a brute force associated alert, then that Netflow was labeled as Attack. If no alert could be associated with the Netflow, then it was labeled as Normal. During the later aggregation process, if any Netflow in a particular aggregation had been labeled as Attack, then the entire resulting aggregation was also labeled as Attack. If no Netflows in the aggregation had triggered alerts, then it remained a Normal aggregation.

Aggregation of Netflows was performed over a five minute time window. A shorter time window might not include enough attack data if the attack data is sent with some delays. A larger time window, on the other side, can delay the detection of the attack. We decided to use five minute time intervals as a trade-off between the short and large time windows. Within this window, Netflows were aggregated based on three common features: source IP (sIP), destination IP (dIP), and destination port (dPort). Once the Netflows were aggregated, 19 features were calculated and/or extracted for evaluation based upon a Netflow's packet size, byte count, duration, etc. A description of each extracted feature can be found in Table 1. We did not include sIP, dIP and dPort in this table because we did not use them in applying the machine learning methods. This makes the approach to be able to be generalized by not being dependent on the IPs and ports seen in a specific network. The aggregated dataset includes 425 Attack instances and 558 Normal instances.

V. Applying Machine Learning Methods

To build the predictive models, we chose four classification algorithms : 5-Nearest Neighbor (5-NN), two forms of C4.5 decision trees (C4.5D and C4.5N), and Naïve Bayes (NB).

TABLE I: Description of features extracted from aggregated data

Feature Name	Description
sIP	Source IP
numOfNetflows	Number of Netflows in aggregation
avgPkts	Average number of packets within aggregation
medPkts	Median number of packets within aggregation
stdPkts	Standard deviation of packets within aggregation
avgBytes	Average number of bytes within aggregation
medBytes	Median number of bytes within aggregation
stdBytes	Standard deviation of bytes within aggregation
avgPktSz	Average packet size (bytes/packets) within aggregation
medPktsz	Median packet size (bytes/packets) within aggregation
stdPktSz	Standard deviation of packet size (bytes/packets) within aggregation
avgDur	Average duration length (in seconds) within aggregation
medDur	Median duration length (in seconds) within aggregation
stdDur	Standard deviation of duration length (in seconds) within aggregation
flags	All flags triggered from all Netflows within aggregation
initialFlags	All initial flags triggered from all Netflows within aggregation
sessionFlags	All session flags triggered from all Netflows within aggregation
class	Class label (Attack or Normal) associated with the Netflows within aggregation

These learners are commonly used in machine learning applications [20]. Using these learners provides a broader analysis from a data mining point of view. We built all models using the WEKA machine learning toolkit [7].

K-nearest-neighbors or K-NN is an instance learning or lazy learning algorithm. This algorithm delays building the predictive models until the testing phase. 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.

The Naïve Bayes algorithm uses Bayes' theorem with the strong assumptions of features being independent to calculate the posteriori probability of an instance being a member of a specific class. While, this assumption makes Naïve Bayes a relatively weak learner, it is a fast classifier.

We use Area Under the receiver operating characteristic Curve (AUC) as the evaluation metric. The Receiver Operating Characteristic (ROC) curve is a graph of the True Positive Rate (TPR) vs False Positive Rate (FPR). In the current application, TPR is the percentage of the brute force 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 intrusion detection applications.

To evaluate the performance values we used 5 fold cross validation. In the 5 fold cross validation, the data is divided to 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 train 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.

VI. RESULTS

We built four models by using four different classification algorithms explained in Section V. Table II shows the performance results. Each AUC value in the table is the average of 20 AUC values acquired during applying four runs of 5 fold cross validation and the std values reflect the standard deviation between those values.

The results show that the predictive models perform very well in the detection of SSH brute force attacks. Different classification algorithms are providing good results. This indicates that the features extracted from the network traffic are discriminative enough for recognizing attack versus normal traffic.

Although all the classification algorithms provide close performance results, decision tree algorithms are easier to interpret. The final decision tree structures show the features contributions in the classification task. Among the two decision trees, C4.5N has better performance with less standard deviation than C4.5D. In the C4.5N, numOfNetflows is selected at the first level in the decision tree structure. We extracted features from an aggregation of Netflows instead of Netflows themselves because this way the instances that belong to brute force attacks would have more Netflows than the instances that belong to the failed login attempts produced by legitimate users that have forgotten their password. So it is not unexpected that numOfNetflows is selected at the first level of the tree structure to discriminate between normal and attack traffic. The next feature selected in the second level of the tree structure is avgPktSz. This feature represents the average packet sizes (bytes/packets) in the aggregated instance. Since a brute force attack is a sequence of login attempts, the average packet size is usually not very high and below a threshold in the tree structure.

TABLE II: Cross Validation Results

Classifier	AUC	AUC std
Naïve Bayes	0.994586	0.00279617
C4.5N	0.99648205	0.00438882
5-NN	0.9965878	0.003975975
C4.5D	0.98712355	0.011668849

VII. Conclusion

In this paper, we explained the process of building machine learning models for the detection of SSH brute force attacks in network traffic. We explained how we incorporated domain knowledge as well as analysis of the real representative collections of network traffic to define discriminative features. We collected our network traffic for the underlying intrusion detection task (detection of brute force attack) from a campus computer network which provides real normal and attack data. We also added data similar to attack network traffic (failed login data produced by legitimate users that have forgotten their passwords) to the collected data. We analyzed brute force versus normal SSH traffic in order to define the features that can be discriminative enough to discriminate between normal and attack traffic. Our analysis showed that attackers tend to change their source port when sending a new login attempt. This results in a new Netflow being built for each attack login attempt. These Netflows are very similar to the Netflows produced in a login attempt by a user that has forgotten his password. The main difference is the number of launched attempts (Netflows) which can be presented in an aggregated instance. The Netflows whose start time falls in the same 5 minute interval with the same source IP, detination IP and destination port are aggregated into one network instance. The features extracted from aggregated Netflows can discriminate between brute force attacks and normal SSH traffic, especially the failed login traffic by legitimate users which can be similar to attack traffic. We used four different classification algorithms to build the predictive models. Our results show that the predictive models perform very well for the detection of SSH brute force attacks. Our approach in defining discriminative features by taking into account the domain knowledge and analysis of representative traffic containing normal data similar to the attack data can be used in building machine learning based-models for the detection of other types of attacks as well. For future work, we aim to compare the performance of the aggregated features and Netflow features for the detection of brute force attacks.

Acknowledgements. 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] "Kdd cup data." [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[2] "Silk." [Online]. Available: https://tools.netsa.cert.org/silk/

[3] "Cisco 2014 annual security report," Cisco, Tech. Rep., 2014. [Online]. Available: http://www.cisco.com/web/offer/gist_ty2_asset/Cisco_2014_ASR.pdf

[4] "Ponemon 2014 ssh security vulnerability report," Venafi, Tech. Rep., 2014.

[5] N. Das and T. Sarkar, "Survey on host and network based intrusion detection system," *International Journal of Advanced Networking and Applications*, vol. 6, no. 2, pp. 2266–2269, Dec. 2014.

[6] M. Drasar and J. Vykopal, "Flow-based brute-force attack detection," in *Advances in IT Early Warning*, F. Verlag, Ed. Oxford: Stuttgart, 2013, pp. 41–51.

[7] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009. [Online]. Available: http://doi.acm.org.ezproxy.fau.edu/10.1145/1656274.1656278

[8] M. E. Haque and T. M. Alkharobi, "Adaptive hybrid model for network intrusion detection and comparison among machine learning algorithms," *International Journal of Machine Learning and Computing*, vol. 5, no. 1, pp. 17–23, 2015.

[9] R. Hofstede, L. Hendriks, A. Sperotto, and A. Pras, "Ssh compromise detection using netflow/ipfix," *Computer Communication Review*, pp. 20–26, 2014.

[10] M. Javed and V. Paxson, "Detecting stealthy, distributed ssh brute-forcing," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer; Communications Security*, ser. CCS '13. New York, NY, USA: ACM, 2013, pp. 85–96. [Online]. Available: http://doi.acm.org/10.1145/2508859.2516719

[11] A. K. Jones and R. S. Sielken, "Computer system intrusion detection: A survey," University of Virginia, Tech. Rep., 1999. [Online]. Available: http://www.princeton.edu/~rblee/ELE572Papers/Fall04Readings/IntrusionDetection_jones-sielken-survey-v11.pdf

[12] 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 *21th ISSAT International Conference on Reliability and Quality in Design*, Philadelphia, PA, USA, 2015, pp. 59–64.

[13] M. M. Najafabadi, T. M. Khoshgoftaar, C. Kemp, N. Seliya, , and R. Zuech, "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*, 2014, pp. 379–385.

[14] M. M. Najafabadi, T. M. Khoshgoftaar, and C. Wheelus, "Attack commonalities: Extracting new features for network intrusion detection," in *21th ISSAT International Conference on Reliability and Quality in Design*, Philadelphia, PA, USA, 2015, pp. 46–50.

[15] A. Narayan and T. J. Parvat, "An intrusion detetion system, with machine learning model combining hybrid classifiers," *Journal of Multidisciplinary Engineering Science and Technology*, vol. 2, no. 4, pp. 647–651, 2015.

[16] P. Sangkatsanee, N. Wattanapongsakorn, and C. Charnsripinyo, "Practical real-time intrusion detection using machine learning approaches," *Computer Communications*, vol. 34, no. 18, pp. 2227–2235, Dec. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2011.07.001

[17] K. A. Scarfone and P. M. Mell, "Sp 800-94. guide to intrusion detection and prevention systems (idps)," Gaithersburg, MD, United States, Tech. Rep., 2007.

[18] H. Tang and Z. Cao, "Machine learning-based intrusion detection algorithms," *Journal of Computational Information Systems*, vol. 5, no. 6, pp. 1825–1831, 2009.

[19] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, "Review: Intrusion detection by machine learning: A review," *Expert Systems with Applications*, vol. 36, no. 10, pp. 11 994–12 000, Dec. 2009. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2009.05.029

[20] J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano, "Experimental perspectives on learning from imbalanced data," in *Proceedings of the 24th International Conference on Machine Learning*, ser. ICML '07. New York, NY, USA: ACM, 2007, pp. 935–942. [Online]. Available: http://doi.acm.org/10.1145/1273496.1273614

[21] M. Zamani and M. Movahedi, "Machine learning techniques for intrusion detection," *CoRR*, vol. abs/1312.2177, 2013. [Online]. Available: http://arxiv.org/abs/1312.2177