Detection of Severe SSH Attacks Using Honeypot Servers and Machine Learning Techniques

Gokul Kannan Sadasivam^{1,*}, Chittaranjan Hota¹, Bhojan Anand²

¹*Department of Computer Science and Information Systems,
BITS, Pilani – Hyderabad Campus, Hyderabad, Telangana, India – 500078*

²*School of Computing, National University of Singapore, Computing 1,
13 Computing Drive, Singapore – 117417*

E-mail: {gokul; hota}@hyderabad.bits-pilani.ac.in; dcsab@nus.edu.sg

**Corresponding Author*

Received 25 November 2016; Accepted 1 January 2017;
Publication 10 February 2017

Abstract

There are attacks on or using an SSH server – SSH port scanning, SSH brute-force attack, and attack using a compromised server. Attacks using a server could be DoS attack, Phishing attack, E-mail spamming and so on. Sometimes an attacker breaks into a public SSH server and uses it for the above activities. Mostly, it is hard to detect the compromised SSH servers that were used by the attackers. However, by analysing the system logs an organisation can know about the compromises. For an organisation holding several SSH servers, it would be tedious to analyse the log files manually. Also, high-speed networks demand better mechanisms to detect the compromises. In this paper, we detect a compromised SSH session that is carrying out malicious activities. We use flow-based approach and machine learning techniques to detect a compromised session. In a flow-based approach, individual packets are not scrutinised. Hence, it works better on a high-speed network. The data is extracted from a distributed honeypot. The paper also describes the machine learning techniques with appropriate parameters and feature selection technique. A real-time detection model that is tested on a public server is also

Journal of Software Networking, 79–100.

doi: 10.13052/jsn2445-9739.2017.005

© 2017 River Publishers. All rights reserved.

presented. Several analyses proved that J48 decision tree algorithm and the PART algorithm are best suited for detection of SSH compromises. It was inferred that inter-arrival time between packets and the size of a packet payload play a significant role in detecting compromises.

Keywords: SSH Compromises, SSH Attacks, Machine Learning, Feature Selection, Flow-based Analysis.

1 Introduction

SSH stands for “Secure Shell”, and it is a protocol in application layer of TCP/IP stack. It provides remote access to a server, and the interaction is completely protected using standard cryptographic algorithms. The sender encrypts the traffic using a symmetric algorithm like AES. The receiver decrypts the ciphered traffic using a key. The key is usually generated using Diffie-Hellman Key Exchange algorithm.

SSH service has been under attack for the past couple of decades. SSH server provides a convenient repository to launch DoS attacks, spread spam messages, and test new malware. A recent survey report [7] has stated that 50% of SSH enterprise servers have experienced SSH key-related compromises. A survey article [1] by Calyptix Security Corporation states that 25% of network attacks on the Internet is brute force attacks. In fact, most of the administrators are not technically competent to prevent a misuse from happening.

Port scanning is a technique to check if a specific set of ports is open on a public server or not. For carrying out an attack on an SSH server, the first step is to check if the port 22 is available or not. SYN scanning is one of the techniques where TCP packets are sent to an SSH server to detect the presence of an open port. If the port is open, then the server would respond with an SYN packet. If it is closed, then the server would send an RST packet.

Brute force attacks are used to know the username and password pair of an account on an SSH server. The attacker would try different usernames with different password combinations. Guessing could be done manually or using some automated tools. Automated tools contain a database of commonly used usernames and passwords.

During brute forcing, if an username and password pair is correct, then the attacker gets a login shell with access to the server’s hard disk. Consequently, the attacker could do several malicious activities. Some of them are: injecting a virus, installing a bot that executes a DDoS attack, spread spam, steal company’s information or test a newly developed malware.

An SSH honeypot (e.g., Kippo SSH Honeypot [2]) stores all the authenticated sessions in log files. These log files provide information about the attackers and the commands used by them. Also, in an organisation's SSH server the authenticated sessions are detected by viewing the log files. The log files are stored on the SSH server. Hence, detecting authenticated sessions in the peripheral of the network where IDS and firewall are located, is not feasible. The question raised by the authors is "How to detect SSH compromises using the network traffic in the network peripheral?". The solution to this question would benefit deploying such a detection system in the firewall, rather than detecting using log files on individual systems.

There are two assumptions in detecting SSH compromises. Firstly, SSH compromise detector detects only those sessions that are actively used by an attacker. Secondly, the real-time detector has a software limitation. The detector considers only unique TCP sessions to find SSH compromises. A TCP session is a unique combination of source IP address, source port number, destination IP address, and destination port number. If there are multiple TCP sessions with same attributes, then it might provide an invalid result. The occurrence of multiple sessions with same source IP address, source port number, destination IP address, and destination port number is rare. Hence, the second assumption can be used reasonably.

For the purpose of machine learning classification, we divide the sessions into two categories. The first category consists of all activities carried out using a compromised SSH server. We name it as "severe" attacks. The second category includes all attacks leading to a compromise. It consists of SSH port scanning, SSH brute-force attack, and brute-force attacks that are successful but no activities thereafter. The second category is named as "not-so-severe" attacks.

SSH protocol uses a strong cryptographic algorithm to hide the payload contents. Since the traffic is encrypted, the communication makes no sense to a casual observer. IDS would not be able to run signature matching on the payload of SSH traffic. However, distinct characteristics between "severe" and "not-so-severe" attacks can be found by choosing appropriate features that can segregate these two attacks using machine learning algorithms.

We employ Machine Learning algorithms, namely, Naive Bayes learner, Logistic Regression, J48 decision tree, Support Vector Machine, OneR, k-NN, and PART to classify the attacks. Suitable features were selected based on domain knowledge, literature survey, and feature selection technique. The performance of machine learning algorithms is evaluated using the metrics accuracy, sensitivity, precision, and F-score.

The remainder of this paper goes as follows: In Section 2, related research work concerning the classification of attacks is mentioned. Section 3 provides three sub-sections detailing the experimental model, feature extraction, and class labelling. Section 4 details the Machine Learning model emphasising the optimal parameters. Section 4 also discusses the analyses done for performance evaluation. In Section 5, the best set of features is extracted using Exhaustive Search technique. In Section 6, a near real-time detection algorithm is explained with a flowchart. Finally, Section 7 concludes the paper with the future scope of work.

2 Related Work

SSH protocol has three sub-protocols: SSH Transport Layer protocol [26], SSH Authentication protocol [24], and SSH Connection protocol [25]. The SSH Transport Layer protocol is used for negotiating the cryptographic algorithms, and the traffic is unencrypted. The application layer traffic for both SSH Authentication protocol and SSH Connection protocol is encrypted. Hence, packet level inspection of SSH Authentication protocol and SSH Connection protocol, using a firewall is unfeasible.

SSH brute-force attacks are detected using the number of DNS PTR Query packets to a DNS Server [9]. When an attack occurs on a campus network, there would be a significant amount of DNS PTR Query packets to the local DNS server. The sample variance in the number of DNS Query packet per minute in a time interval of 10 minutes is used to detect SSH brute-force attacks.

Maryam M. Najafabadi et al., [12] used Machine Learning algorithms to detect SSH Brute force attacks. The algorithms used were 5-NN (Nearest Neighbour), C4.5D and C4.5N (Decision Tree algorithms), and Naive Bayes algorithm. Almost all classifiers performed well in classifying the brute force attacks. 5-NN outperformed all other classifiers with and without source port features.

Legitimate login failures on SSH server portrays a Beta-Binomial Distribution [8]. When there is a distributed brute-force attack, the Global Factor Indicator (GFI) would deviate from the mean of the normal user traffic distribution. This deviation helps to find distributed brute-force attacks.

The behavior of SSH Brute-force attacks can be modelled using a Hidden Markov Model [19]. The model consists of seven hidden states (phases): active scanning phase, inactive scanning phase, active brute-force attack phase, inactive brute-force attack phase, active compromise phase, inactive

compromise phase, and end state. The transition probabilities are computed by observing the flow metrics (flows per second, packets per flow, bytes per packet). This model is used in their research project to build a detection tool called as SSHCure [4, 10].

SSHCure is a plugin for NfSen (NetFlow Sensor) tool [14]. A NetFlow sensor collects NetFlow data from different routers and integrates the data. SSHCure reports SSH attack details, target details, and attacker details. The backend of SSHCure makes use of HMM model [19] to determine SSH attack phases. The accuracy of attack detection was 0.839 and 0.997.

The purpose of Rick Hofstede et al., [6] work is to improve the existing SSHCure plugin. Certain characteristics of the brute-force attack tools were used to detect SSH compromises. It also considers the features of OpenSSH server. SSHCure 3.0 [5] considers the flat traffic nature of brute-force attacks.

Akihiro Satoh, Yutaka Nakamura, and Takeshi Ikenaga [18] have proposed a methodology to detect successful or unsuccessful brute-force attack using Ward Clustering algorithm. The features considered are packet order, packet size, and packet direction. The model takes a TCP session and checks for the existence of SSH connection sub-protocol. The mere existence proves that a brute-force attack is successful. Their model does not mention about the SSH compromise sessions without any activity after the compromise. Also, their paper doesn't discuss the practical applicability of their model and the possibility for real-time detection.

This paper is an extension of the conference paper [17] done by the same authors. The conference paper describes the machine learning model and the feature selection.

3 Data Collection

3.1 Experimental Setup

A honeynet system was installed and configured in BITS, Pilani – Hyderabad campus. Gokul et al., [16] had built the system on a DELL PowerEdge server. The system comprised low-interaction honeypot tools running on different virtual machines. The physical server with a public IP address collected network traffic continuously. Figure 1 shows the entire honeynet architecture. The system was active from 22nd July 2014 to 16th August 2014. The honeypot ran incessantly, day and night, without interruption. Kippo SSH Honeypot [2] was used to garner SSH traffic.

Figure 1 Honeynet architecture.

The honeypot system allowed connections on several ports including SSH, MySQL, MS-SQL, and HTTP. Among all the network services, SSH had a significant amount of traffic (approximately 11,000 TCP sessions). This traffic comprises of SSH scanning attacks, SSH brute-force attacks, and SSH compromises (successful SSH connections). All the SSH sessions used password-based authentication because it was the only authentication method enabled on the kippo server.

3.2 Classes

There are two categories of SSH attacks: severe and not-so-severe. Severe attacks comprise all pcap files which had a successful login attempt followed by an execution of a couple of Unix commands. A successful connection could start with a brute force attack. However, it is necessary that the attacker tried, at least, one Unix command on the SSH server.

Table 1 Traffic classes

Severity Level	Traffic Type	Number of TCP Flows
Severe Attacks	SSH Compromise with one or more Unix commands	14
Not-So- Severe Attacks	SSH Compromise with no Unix commands	58
	SSH Brute-force attacks	10308
	SSH Port Scanning	552

Not-so-severe attacks comprise the following types.

1. A successful connection with no Unix commands. It happens because the attacker is only interested in knowing the login credentials. Once the brute force attack is successful, the connection is closed.
2. An unsuccessful connection comprising only login failure messages. It is a common brute force attack which is unsuccessful.
3. In port scanning attacks, two techniques were observed.
 - (a) SYN scanning
 - (b) TCP Connect scanning

The authors used two approaches for tagging the traffic files with its corresponding classes. Firstly, programs were written to extract scanning attempts. Secondly, manual inspection of the pcap files with its respective log files was carried out to obtain the correct classes. Table 1 provides the list of SSH attacks.

The class label is chosen as “severe attack” because a severe attack will damage a computer system. SSH compromise is one in which an attacker is carrying out malicious activities. Hence, this will cause harm to one’s system and external systems. On the other hand, a not-so-severe attack will not cause any internal damage to the system. It will take some of the network bandwidths but will not cause any deletion or modification of the file system data.

3.3 Extraction of Features

TCP flow packets were processed to obtain a set of statistical attributes. The following features were computed from each flow pcap file.

1. Total number of packets
2. Total number of received packets
3. Total number of sent packets
4. The sum of all packet bytes

5. The sum of all received packet bytes
6. The sum of all sent packet bytes
7. The sum of all payload bytes
8. The sum of all received payload bytes
9. The sum of all sent payload bytes
10. Mean inter-arrival time between received packets
11. Variance of inter-arrival time between received packets
12. Total number of packets with ACK flag set
13. Total number of packets with PSH flag set
14. Total number of packets with RST flag set

All the above features were obtained through domain knowledge and literature survey. The range of values in each one of the features are different. Since the algorithms “Logistic Regression”, “Support Vector Machine”, and “k-NN” perform better with normalised data, each of the features in the data set is normalised using max-min scaling.

4 Machine Learning Model

The network traffic was captured in pcap file format using the popular “tcpdump” tool. The pcap file was completely processed to remove unwanted network traffic. This traffic includes Ubuntu updates/upgrades, legitimate traffic, and honeypot status messages. Splitcap tool [13] was used to separate the merged pcap file into individual TCP flows.

A Java program interfacing with the jnetpcap library [21] was written to extract relevant features from each flow pcap file. Some of the characteristics (mean inter-arrival time, the variance of inter-arrival time) were computed using statistical formulae. Weka tool [3] was used to execute the Machine Learning algorithms.

4.1 Machine Learning Algorithms

Some machine learning supervised algorithms were chosen to evaluate the performance of classification. The algorithms are Logistic Regression (LR), Support Vector Machine (SVM), Decision Tree (J48), OneR, PART, k-Nearest Neighbour, and Naive Bayes (NB) algorithm. Logistic Regression, SVM, and k-Nearest Neighbour (k-NN) performs well for a dataset with only numeric attributes. Decision Tree is one of the widely used algorithms in the field of information security. Even though Naive Bayes algorithm is well suited for nominal attributes, it can be applied even for numerical attributes. ZeroR is used as a baseline classifier.

Figure 2 Machine learning model.

Logistic Regression uses a sigmoid function to determine the correct output class. In Equation (1), L is the cost function, Y_i is the output class in binary values (0,1), and X_i is a normalised data sample. P_{SA} and P_{NSA} stand for the probability of a severe and that of a not-so-severe attack, respectively. θ is the vector of LR coefficients, and r is the ridge parameter. The ridge parameter used is 1×10^{-8} . Quasi-Newton method finds the optimal values of θ .

$$L = - \sum_{i=1}^n (Y_i \log P_{SA}(X_i) + (1 - Y_i) \log P_{NSA}(X_i)) + r\theta^2 \quad (1)$$

SVM is a binary linear classifier that outputs an hyperplane with a large margin between the positive and the negative samples. Sequential Minimal

Optimization (SMO) [15] computes the coefficients of the SVM algorithm. The objective function considers the slack variables (ϵ_i) and the penalty constant C . The value of ϵ_i is 1×10^{-12} and penalty constant C is 1. The kernel method used in SVM is a normalised polynomial function with the degree of the polynomial as 24. In Equation (2), k' is a kernel function and d is the degree of the polynomial. In Equation (3), k is the normalised kernel function.

$$k'(X_i, X_j) = [X_i \cdot X_j]^d = [X_i \cdot X_j]^{24} \quad (2)$$

$$k(X_i, X_j) = \frac{k'(X_i, X_j)}{\sqrt{k'(X_i, X_i) \cdot k'(X_j, X_j)}} \quad (3)$$

J48 [23] is the Java version of the decision tree algorithm C4.5 (Revision 8). The splitting criterion is decided by two parameters: Information Gain and Information Gain Ratio. The information gain of an attribute should be greater than the average information gain of all the attributes, and the information gain ratio should be maximum. Information Gain Ratio is the ratio of the information gain and the intrinsic value of an attribute. The expression for intrinsic value calculation is given in Equation (4). In Equation (4), $IV(A)$ is the intrinsic value for attribute A , n is the number of different values in attribute A , N_i is the number of samples having i^{th} value in attribute A , and N is the total number of samples. The pruned and the unpruned version produced the same results. For the pruned version, the confidence factor is 0.25.

$$IV(A) = - \sum_{i=1}^n \frac{N_i}{N} \log \frac{N_i}{N} \quad (4)$$

Naive Bayes algorithm [11, 20] utilises the conditional probability formula of Bayes Theorem. Naive Bayes helps in predicting the conditional probabilities of all classes and chooses the one with the highest probability. When the features are numerical, each feature can be treated as a particular distribution. All features in this work portray a Gaussian distribution. Hence, Gaussian distribution is used to determine the correct class.

PART [3, 20] is a rule-based classifier based on Separate-and-Conquer strategy. The confidence threshold for pruning is 0.25. The minimum number of instances per rule is 2. With these settings, PART did reasonably good.

OneR [3] is a rule-based classifier that performs well for most of the datasets in different domains. For OneR algorithm, the numeric attributes are discretized into several intervals. The minimum number of samples in a bucket

is set to 6. In fact, the results are unchanged even when the minimum number of samples in a bucket is 3 or 4 or 5 or 6.

k-NN [11, 20, 22] is an instance-based learning algorithm. The similarity measure considered here is Euclidean distance. Cross-validation (hold-one-out) provided the optimal value for the number of nearest neighbours. The number of nearest neighbours for this dataset is 3. The results were same when weights were added.

In all the above algorithms (except k-NN), 10-fold stratified cross-validation was employed to differentiate training data and testing data. In each fold, the dataset was randomly chosen. Before application of each fold, the dataset was randomised again.

4.2 Performance of ML Algorithms

In this section, we describe the performance of the chosen set of algorithms. For each algorithm, the confusion matrix is computed. The confusion matrix is shown in Table 2.

In Table 2, TS, FS, TNS, and FNS represents True Severe Attack, False Severe Attack, True Not-So-Severe Attack, and False Not-So-Severe Attack, respectively. The confusion matrix for the eight ML algorithms is shown in Table 3.

Since the dataset is unbalanced, metrics like sensitivity, precision, and F-score would give a better picture about the classification.

Table 2 Confusion matrix

	Predicted: Severe Attack	Predicted: Not-So-Severe Attack
Actual: Severe Attack	TS	FNS
Actual: Not-So-Severe Attack	FS	TNS

Table 3 Confusion matrix for ML algorithms

Algorithm	TS	FS	TNS	FNS
ZeroR	0	0	10918	14
SVM	8	3	10915	6
kNN	8	1	10917	6
OneR	9	1	10917	5
LR	10	4	10914	4
NB	12	5	10913	2
PART	13	0	10918	1
J48	13	0	10918	1

$$Accuracy = \frac{TS + TNS}{TS + TNS + FS + FNS} \quad (5)$$

$$Sensitivity = \frac{TS}{TS + FNS} \quad (6)$$

$$Precision = \frac{TS}{TS + FS} \quad (7)$$

$$F - score = \frac{(1 + \beta^2)(prec. \times sens.)}{(\beta^2 prec.) + sens.} \quad (8)$$

The accuracy (as given in Equation (5)) provides the ratio between the attacks correctly classified to the overall attacks. A Machine Learning algorithm that provides a low value of accuracy will also have a low value for sensitivity and precision. Hence, accuracy is the first performance metric that should be checked for a Machine Learning algorithm. As shown in Table 4, the accuracy is more than 99% for all the chosen algorithms. Neglecting ZeroR, all other algorithms are eligible candidates for performance comparison.

Sensitivity (as provided by Equation (6)) is the fraction of severe attacks that are correctly predicted by a classifier. In comparison to all the algorithms, J48 decision tree algorithm and PART algorithm had the highest sensitivity of 92.86%. Naive Bayes Algorithm and Logistic Regression were comparatively better than Support Vector Machine and k-NN algorithm, which were able to detect only 57.14% of all the severe attacks.

Precision (as provided by Equation (7)) conveys the fraction of predictions that are correct. J48 and PART predictions are very accurate. As seen in column 4 of Table 4, the precision of the J48 and PART algorithm hit the maximum of 100%. The results of k-NN and OneR are closer to the J48 algorithm.

Table 4 Performance evaluation

Algorithm	Accuracy	Sensitivity	Precision	F_2 Score
ZeroR	0.9987	0.0000	–	–
SVM	0.9992	0.5714	0.7273	0.5970
kNN	0.9994	0.5714	0.8889	0.6154
OneR	0.9995	0.6429	0.9000	0.6818
LR	0.9993	0.7143	0.7143	0.7143
NB	0.9994	0.8571	0.7059	0.8219
PART	0.9999	0.9286	1.0000	0.9420
J48	0.9999	0.9286	1.0000	0.9420

The performance of SVM is minutely better than Naive Bayes and Logistic Regression. Naive Bayes, which had a better sensitivity, has a relatively poor precision.

F-score (as given in Equation (8)) is the weighted average of sensitivity and precision. The value of β determines the balance between sensitivity and precision. When $\beta > 1$, the F-score gives more preference to sensitivity than precision. In this work, sensitivity is given more importance than precision. Hence, the authors have chosen a β of 2. F-score reaches its best value at 1 and worst value at 0. As previously observed, the J48 and PART algorithm had the maximum F_2 score. Irrespective of the poor precision value by Naive Bayes algorithm, it had a better F_2 score. It is due to the balance provided by its sensitivity value. The F_2 score of Logistic Regression remained the same as its sensitivity and precision. SVM had an approximate value close to its sensitivity.

J48 and PART have the highest score for all the different performance metrics. J48 chooses the best features using entropy and information gain. It means feature selection is built into decision tree algorithms. The reason for J48's best performance could be its feature selection mechanism. Hence, feature selection method was applied to find the best set of features.

5 Feature Selection

Feature Selection [20] is a process of determining the redundant and unnecessary features in a data set. It reduces the computational cost involved in training a data set. It also eliminates data overfitting. By decreasing the bias, it helps achieve more accuracy. The three standard techniques are Subset selection (Wrapper method), Filter method and Embedded approach. All techniques use heuristics to estimate the best features. Heuristics approaches may not always produce the best set of features.

Exhaustive search is a Wrapper method that tries each combination of features to obtain the best feature set. Hence, it is guaranteed to produce the correct answer. However, the downside is the computation time required. Given the dimensionality of our data set, it was feasible to use this technique to obtain the best features.

Naive Bayes is the performance evaluation algorithm used in the exhaustive search. Exhaustive search technique produced the best feature set in the order mentioned in Table 5, with accuracy as the performance metric.

Accuracy alone is not sufficient to decide on the best feature set. Hence, sensitivity, precision, and F-score were computed to find the best feature set.

Table 5 Accuracy of different feature sets

Feature Set	Accuracy
2	0.99894
9	0.99914
4, 10	0.99923
1, 4, 10	0.99934
2, 4, 10	0.99943
6, 10	0.99954
5, 6, 10	0.99963
4, 5, 6, 10	0.99969
6, 7, 10	0.99973
9, 10	0.99991

Figure 3 NB best feature set performance.

Figure 3 shows the performance metrics for various feature sets generated by the feature selection technique. From Figure 3, it could be inferred that the set comprising features 9 and 10 produces the optimal performance.

The features f9 and f10 are used to evaluate the performance of all algorithms considered. The accuracy, sensitivity, precision, and F_2 score have improved prominently for Logistic Regression and k-NN (as shown in Table 6). On the other hand, SVM had a slight dip in accuracy and precision. The decrease was due to the misclassification of 14 different not-so-severe

Table 6 Performance with best features

Algorithm	Accuracy	Sensitivity	Precision	F_2 Score
SVM	0.9990	1.0000	0.5600	0.8642
3-NN	0.9997	0.9286	0.8667	0.9155
OneR	0.9995	0.6429	0.9000	0.6818
LR	0.9997	0.9286	0.8667	0.9155

attacks. Importantly, SVM classifies all the severe attacks leading to a sensitivity of 100%. Moreover, the performance of OneR was unchanged.

Figure 4 shows the pruned decision tree generated by the J48 algorithm. The feature f10 corresponds to the mean inter-arrival time between received packets. The feature f9 corresponds to the sum of all sent payload bytes. As shown in Figure 4, feature f10 was able to segregate most of the not-so-severe attacks (10907 attacks). Feature f9 did justice in separating most of the severe attacks (13 attacks). If the mean inter-arrival time of received packets is greater than 1651 milliseconds, then it is highly likely that the attack is not-so-severe. If there are more than 3216 sent payload bytes in a session, then it would probably be a severe attack.

6 Near Real-Time Detection

Real-time detection of SSH severe attacks helps to mitigate the problems caused by the attacks. Figure 5 is the flowchart for detection of SSH severe attacks in real-time. A Java application depicting the flowchart is built with the help of libraries: jnetpcap, commons-math3-3.5, and other libraries (created

Figure 4 J48 decision tree.

Figure 5 Flowchart for real-time detection of SSH severe attacks.

by authors). The application is multi-threaded in nature and makes use of shared resources. The shared resources are accessed using synchronised method calls. It is near real-time because a TCP session has to end to correctly detect it. The only assumption is that two sessions cannot have the same source IP address and source port number.

The first thread does the network packet capture. Jnetpcap library provides methods to capture packets from a live network. In order to process the packets

quickly, each time a packet is received it is inserted into a queue. The queue size is 100,000 packets.

The second thread dequeues every packet from the queue and processes it. Each packet is analysed carefully and placed in the appropriate session file. Each session is determined by a unique combination of source port number, destination port number, source IP address, destination IP address and protocol type. Along with the packet, the time the packet was received, the FIN flag and the RST flag are stored.

The third thread iterates through all session files and passes them for final classification. It checks for three different conditions before closing the sessions. These conditions are described below.

The appearance of an RST message indicates the immediate closure of a session. If an RST packet was received/sent as the last packet, then the thread will immediately pass the session file for classification.

In many OS kernels, a KeepAlive timer is used to prevent long idle connections. The time-out depends on the OS and is usually 2 hours. For the purpose of real-time detection, the timer is set to 10 minutes. If the last packet was received more than KeepAlive time (i.e., 10 minutes), then all packets in the session file are passed for attack classification.

The machine that performs active close waits to receive a FIN message from the other end. After receiving a FIN message, it sends an ACK message and waits for 2MSL. After 2MSL the session is closed. After receiving/sending two FIN messages, the timer starts for 2MSL. After 2MSL the session is closed. 2MSL value is set as 60 seconds.

The classification is done using J48 decision tree algorithm (Figure 4). If it is an SSH compromise session, then a firewall rule is added to drop all packets from the session IP address. The firewall used is iptables. If there are more than a pre-defined number of compromises, an alert could be sent to the administrator to change the password. After changing the password, all the iptables rules can be removed safely.

The time taken to build the model using J48 is 0.26 seconds. The training time depends on the number of training samples in the dataset. In real-time detection, a session has to be completed before determining the type of attack. The amount of time to classify a session as severe attack depends on two factors: (a) Waiting time for a session to end, (b) Classification time. The waiting time is determined by the duration of a TCP connection. Assuming, the number of packets in a session as n , the Big-Oh notation is given by $O(n)$. The time to classify a sample is negligible and considered as $O(1)$. Therefore, the total time to detect a severe attack is $O(n)$.

7 Conclusion

We have proposed a machine learning model to classify SSH attacks based on the attack nature. We found that the feature set {f9, f10} is significant in determining SSH compromises. After analyses on the performance of several machine learning algorithms, the performance of J48 and PART seemed to be promising and produced better results. In future, the authors would like to test the practicality of this model in building better intrusion detection systems.

References

- [1] Calyptix (2015). *Top 7 Network Attack Types in 2015*. Available at: <http://www.calyptix.com/top-threats/top-7-network-attack-types-in-2015-so-far>
- [2] Desaster (2014). *Kippo ssh Honeypot*. Available at: <https://github.com/desaster/kippo>
- [3] 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, 10–18.
- [4] Hellemons, L., Hendriks, L., Hofstede, R., Sperotto, A., Sadre, R., and Pras, A. (2012). “Sshcure: a flow-based ssh intrusion detection system” in *Dependable Networks and Services: Lecture Notes in Computer Science*, Vol. 7279, (Berlin: Springer Verlag), 86–97.
- [5] Hofstede, R., and Hendriks, L. (2015). “Unveiling sshcure 3.0: Flow-based ssh compromise detection,” in *Proceedings of the International Conference on Networked Systems, NetSys 2015*, Cottbus: Brandenburg University of Technology Cottbus-Senftenberg.
- [6] Hofstede, R., Hendriks, L., Sperotto, A., and Pras, A. (2014). Ssh compromise detection using netflow/ipfix. *SIGCOMM Comput. Commun. Rev.*, 44, 20–26.
- [7] Venafi Inc (2014). *Ssh Security Vulnerability Report*: Available at: https://www.venafi.com/assets/pdf/Ponemon_2014_SSH_Security_Vulnerability_Report.pdf, 2014.
- [8] Javed, M., and Paxson, V. (2013). “Detecting stealthy, distributed ssh brute-forcing,” in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, CCS*, (New York, NY: ACM), 85–96.
- [9] Kumagai, M., Musashi, Y. D., Romana, A. L., Takemori, K., Kubota, S., and Sugitani, K. (2010). “Ssh dictionary attack and dns reverse

- resolution traffic in campus network,” in *Proceedings of the Intelligent Networks and Intelligent Systems (ICINIS), 3rd International Conference*, Washington, DC, 645–648.
- [10] Hofstede, R., and Hendriks, L. (2016). Available at: <https://sourceforge.net/projects/sshcure/>, 2016.
 - [11] Mitchell, T. M. (1997). *Machine Learning*. New York City: McGraw-Hill.
 - [12] 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 *Proceedings of the Bioinformatics and Bioengineering (BIBE), 2014 IEEE International Conference*, Rome, 379–385.
 - [13] Netresec (2013). *Splitcap Tool*. Available at: <http://www.netresec.com/?page=SplitCap>
 - [14] NfSen (2011). *Nfsen*. Available at: <http://nfsen.sourceforge.net/>
 - [15] Platt, J. C. (1998). *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*. Technical Report MSR-TR-98-14. Microsoft Research.
 - [16] Sadasivam, G. K., and Hota, C. (2015). “Scalable honeypot architecture for identifying malicious network activities,” in *Proceedings of the 2015 International Conference on Emerging Information Technology and Engineering Solutions (EITES)* (Rome: IEEE), 27–31.
 - [17] Sadasivam, G. K., Hota, C., and Anand, B. (2016). “Classification of ssh attacks using machine learning algorithms,” in *Proceedings of the 2016 6th International Conference on IT Convergence and Security (ICITCS)* (Rome: IEEE), 1–6.
 - [18] Satoh, A., Nakamura, Y., and Ikenaga, T. (2012). “Ssh dictionary attack detection based on flow analysis,” in *Proceedings of the 2012 IEEE/IPSJ 12th International Symposium on Applications and the Internet (SAINT)* (Rome: IEEE), 51–59.
 - [19] Sperotto, A., Sadre, R., Boer, P.-T., and Pras, A. (2009). “Hidden markov model modeling of ssh brute-force attacks,” in *Proceedings of the 20th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management: Integrated Management of Systems, Services, Processes and People in IT, DSOM '09*, (Berlin: Springer-Verlag), 164–176.
 - [20] Tan, P.-N., Vipin, K. and Steinbach, M. (2006). *Introduction to Data Mining*. London: PEARSON.
 - [21] Sly Technologies (2016). *jNetPcap Library*. Available at: <http://jnetpcap.com>

- [22] Witten, I. H. (2016). *Weka Class IBk*. Available at: <http://weka.sourceforge.net/doc.dev/weka/classifiers/lazy/IBk.html>
- [23] Witten, I. H. (2016). *Weka Source Code*. Available at: <http://www.cs.waikato.ac.nz/ml/weka/downloading.html>
- [24] Ylonen, T., and Lonvick, C. (2006a). *The Secure Shell (ssh) Authentication Protocol*. Ed. RFC. Available at: <https://www.rfc-editor.org/rfc/rfc4252.txt>
- [25] Ylonen, T., and Lonvick, C. (2006b). *The Secure Shell (ssh) Connection Protocol*. Ed. RFC. Available at: <https://www.rfc-editor.org/rfc/rfc4254.txt>
- [26] Ylonen, T., and Lonvick, C. (2006c). *The Secure Shell (ssh) Transport Layer Protocol*. Ed. RFC. Available at: <https://www.ietf.org/rfc/rfc4253.txt>

Biographies

G. K. Sadasivam is a Lecturer in Department of Computer Science, BITS, Pilani Hyderabad Campus, India. He received the B.E. degree from Anna University (Main Campus), Chennai, India, in 2004, and the M.Sc. degree in Computer Engineering from the National University of Singapore (NUS), Singapore, in 2007. He completed his second M.Sc. Degree in Computer Science from the Northwestern Polytechnic University, California, The United States of America, in 2012. He is currently pursuing Ph.D. in Computer Science at BITS, Pilani – Hyderabad Campus, India.

C. Hota is a Professor and Associate Dean (Admissions) at Birla Institute of Technology and Science-Pilani, Hyderabad, India. He is also responsible for managing the Information Processing Unit at BITS-Hyderabad that takes care of ICT needs of the entire institute. He was the founding Head of Dept. of Computer Science at BITS, Hyderabad. Prof. Hota did his Ph.D. in Computer Science and Engineering from Birla Institute of Technology & Science, Pilani. He has been a visiting researcher and visiting professor at University of New South Wales, Sydney.

B. Anand, Senior Lecturer in School of Computing, National University of Singapore. He has received Ph.D. in Computer Science from National University of Singapore. He has received several awards including Dean's graduate research achievement award, University first rank and State rank scholarship. His thesis was nominated for Wang Gungwu Medal & Prize which is the top most award of the university. He was a mentor and visiting scholar at MIT Gambit lab (2008), USA. His works are published in premier conferences including IEEE-INFOCOM, ACM-SIGCOM, ACM-Mobisys, ACM-Netgames, IFIP-ICEC and ACM-Multimedia on a wide range of topics. His current research interests focus on Interactive Virtual Environment Design, Cybersecurity, Systems and Networking.