Machine Learning for Detecting Brute Force Attacks at the Network Level

Conference Paper · November 2014

DOI: 10.1109/BIBE.2014.73

5 authors, including:

Maryam M Najafabadi
Florida Atlantic University
16 PUBLICATIONS **308** CITATIONS
SEE PROFILE

Taghi Khoshgoftaar
Florida Atlantic University
413 PUBLICATIONS **8,694** CITATIONS
SEE PROFILE

Clifford Kemp
Indian River State College
8 PUBLICATIONS **34** CITATIONS
SEE PROFILE

Naeem Seliya
Ohio Northern University
79 PUBLICATIONS **1,986** CITATIONS
SEE PROFILE

2014 IEEE 14th International Conference on Bioinformatics and Bioengineering

Machine Learning for Detecting Brute Force Attacks at the Network Level

Maryam M. Najafabadi, Taghi M. Khoshgoftaar, Clifford Kemp, Naeem Seliya, and Richard Zuech
{mmousaarabna2013@fau.edu; khoshgof@fau.edu; cliffkempfl@gmail.com; nseliya@gmail.com; rzuech@fau.edu}
Florida Atlantic University, Boca Raton, Florida, USA

Abstract — The tremendous growth in computer network and Internet usage, combined with the growing number of attacks makes network security a topic of serious concern. One of the most prevalent network attacks that can threaten computers connected to the network is brute force attack. In this work we investigate the use of machine learners for detecting brute force attacks (on the SSH protocol) at the network level. We base our approach on applying machine learning algorithms on a newly generated dataset based upon network flow data collected at the network level. Applying detection at the network level makes the detection approach more scalable. It also provides protection for the hosts who do not have their own protection. The new dataset consists of real-world network data collected from a production network. We use four different classifiers to build brute force attack detection models. The use of different classifiers facilitates a relatively comprehensive study on the effectiveness of machine learners in the detection of brute force attack on the SSH protocol at the network level. Empirical results show that the machine learners were quite successful in detecting the brute force attacks with a high detection rate and low false alarms. We also investigate the effectiveness of using ports as features during the learning process. We provide a detailed analysis of how the models built can change as a result of including or excluding port features.

Keywords — Brute force attack, network flow, network-level attack detection, machine learning.

I. INTRODUCTION

Any host connected to the public Internet or even a private computer network is under a constant threat of Internet or insider attacks. In today's computing environments, network attacks have become a critical problem for the security of computer networks. With an increase in dependence on the Internet and computer networks (e.g., cloud computing, social networking, online forms, etc.), malicious behaviors have become commonplace. Attacks and malicious activities over a network must be detected to ensure the intended functioning of the system and assure secure transfer of user data.

Network attacks can be detected, or even prevented, by monitoring and analyzing relevant data at the network level or at the host level; however, high speed networks and the utilization of encrypted traffic data make such detections challenging. Host-based detection uses internal software installed on the host to monitor arriving traffic received by that host. This attack detection software used by the host also has access to the internal logs, such as users' login activities, running processes and applications, and other relevant data – which it can utilize for attack detection. A key downside of host-based detection is the inability to detect distributed attacks that have become prevalent in today's Internet and ubiquitous computing environments. Detection of such attacks requires a broader view and inspection of the network data. Network-based detection provides such a perspective.

In network-based detection, all of the data traversing the network can be monitored and analyzed. In order to accomplish this, the detection element must be positioned at the network transit points. In comparison to host-based detection software, which is deployed only on a particular host, a network-based detection scheme is more scalable. It is not dependent on the host's operating system (OS). In addition to providing protection for hosts that do not have their own protection mechanism, network-based detection is the only possibility in situations where there is no direct access to particular hosts.

In network-based detection, since all of the data passing through the network needs to be monitored for malicious activity, the detection scheme/model needs to be fast and efficient. One solution is the application of flow-based analysis. In recent years, network security research has started to focus more on flow data to detect attacks. Flow analysis involves monitoring and analysis of network flows instead of packets. Network flow is an aggregation of packets that share some identical network features/behaviors during a specific time interval, i.e. window. The five most common features that are used in the definition of a network flow is a 5-tuple key consisting of source IP, destination IP, source port, destination port, and network protocol. Since network flow is an aggregation of network packets, analyzing network traffic at the flow-level speeds up the detection processing time as compared to analyzing each individual packet because the amount of data to be processed is reduced. Since flow data is based only on packet headers and not packet payload information, it is a better option for the detection of attacks that contain encrypted payload within the traffic.

Brute force attacks are one of the most prevalent types of attacks in computer networks [1], [2]. In a brute force attack on the SSH protocol the attacker tries to log in to a user's

978-1-4799-7502-0/14 $31.00 © 2014 IEEE
DOI 10.1109/BIBE.2014.73

account, and continues trying different passwords on the victim's machine to reveal the login password. Typically, attackers use automated software that generates different combinations of passwords to attempt against the victim's machine. Unfortunately, human-chosen passwords are inherently weak because they are selected from a limited domain of the user's knowledge. Moreover, the need for memory retention/recall of the passwords aids to the weakness of passwords. For example, a recent article by CBS News presents the top 25 common passwords of 2013, revealing the use of very weak passwords such as "123456", "qwerty", "abc123", "sunshine", etc. This makes it easier for an attacker to find the correct password by trying different possible password permutations. Another key reasons brute force attacks are popular are the continued use of default auto-generated passwords and using the username as the password. While mandated change of default passwords is increasing, some old servers that do not facilitate this provide opportunities for a brute force attack on the SSH protocol.

The research on the detection of brute force attacks has generally focused on detection at the host level. At the host level detection, access logs are inspected and if the number of failed login attempts in a specific time exceeds a predefined threshold number an alert is fired. In this work we study the detection of brute force attack at the network level. It scales better in comparison to a host-based detection scheme. Additionally, network-based detection also is critical in detecting network-based attacks and provides some protection for devices that do not have internal (host level) protection. We investigate using machine learners for automated detection of brute force attacks (on the SSH protocol) at the network level based on flow data.

Our case study data is collected from a real-world network data from a production computer network. We extract network flows from the full packet captures using a tool named SiLK [3]. The data is then labeled by professional network experts to detect brute force attacks. Finally, each flow is labeled as being a brute force attack or not a brute force attack to generate a labeled network flow dataset. We prefer not to use the usual "attack" and "normal" label notations largely because the traffic data labeled as "normal" might include attacks other than brute force attack – the focus of our study. Our intention in this work is the detection of brute force attacks that's why we found labeling as "brute force", "not brute force" more appropriate in this case.

We investigate four different machine learners (classifiers) for detecting brute force attacks. They include K-Nearest-Neighbor with K = 5 (5-NN), Naïve Bayes (NB), and two versions of the C4.5 Decision Tree (C4.5D and C4.5N). Details of these learners are provided in Section III. Using four different learners presents a broader analysis on the ability of machine learning to detect brute force attacks at the network level. To provide evaluation results on the models' performance, 5-fold cross-validation is used. Our results demonstrate that using machine learning methods is a very effective approach for the detection of SSH brute force attacks at the network level. We also investigate the effect of using port information for the detection of brute force attacks. This is conducted by repeating the above modeling and evaluation for brute force attack detection, but without using ports data as features in building the detection models.

The remainder of this paper is organized as follows. Section II provides related works on the topic of brute force attack and different detection approaches. Section III presents our case study data, the classifiers utilized, and the main empirical designs. In Section IV, we discuss our results. Finally in Section V, we conclude our work and provides suggestions for the future work.

II. RELATED WORK

The SANS institute called the brute force attack "the most common form of attack to compromise servers facing the Internet" in its 2007 Top-20 Security Risks report. The prevalence of brute force attack has been investigated in different studies. Bezut et al. [4] studied four months of SSH brute force attack data collected from honeypot machines. The authors concluded that every individual SSH port on the Internet is very likely to experience brute force attacks. Owens et al. [5] study brute force attacks observed on three different networks – a residential system with a DSL Internet connection, a university campus network, and an Internet-connected small business. Their studies show that brute force attacks applied on these different networks are very similar. Moreover, their observations suggest that many brute force attacks are based on precompiled lists of user name and passwords which are widely shared. In addition to the two works discussed above, there are other studies that also suggest that brute force attacks are one of the most prevalent attacks on the Internet [1], [6], [7].

Host-based detection techniques for brute force attacks are based on counting the number of failed login attempts from a specific host during a specific time interval. If this number exceeds a predefined number of attempts threshold the host is blocked. The approach stems from the fact that automated software used in a brute force attack will try testing more wrong passwords than a legitimate user who has forgotten their password during the specified time interval. DenyHosts [8], BlockHosts [9], and BruteForce-Blocker [10] are some of the common host-based detection techniques.

Kumagai et al. [11] calculated the sample variance of the total PTR resource record based DNS query packet traffic of campus networks servers which were under brute force attack. Their observations show there is a change in this statistical value when the SSH brute force attack occurs. However, they did not provide any threshold limit for the change detection. In addition, the authors did not provide accuracy evaluation for their detection method. Mobin et al. [12] also use statistical analysis for the detection of brute force attacks. They calculate a parameter that summarizes aggregate activity. Significant change in this parameter can demonstrate a distributed brute force attack.

Malecot et al. [13] use information visualization for the detection of distributed brute force attack. For each local host, the hosts attempting to connect to the local host are shown in a mapping structure, denoted as a quad tree. The idea is that coordinated attackers appear in quad trees of multiple hosts and it makes their detection possible. However, this approach needs a network expert's analysis and is not able to detect the attacking hosts automatically. Their study focuses on recurring brute force attacks, and concludes that whenever there is a SSH port open on the Internet, there is a very good possibility that the attackers will perform brute force attack on it. In addition, their study shows that the attackers most likely target professional servers rather than workstations.

III. Experimental Design

In this section, we explain the main steps we took in our empirical studies. First, a new dataset is built from network traffic data collected from a real-world production computer network. Second, the network flow data is extracted from the packet data captured from the network using the SiLK traffic analysis tool [3]. Brute force attack flows (and consequently labeling not brute force attack instances) are then labeled in the data by network experts. The four classifiers, as briefly stated earlier, are used to build classification models on the labeled flow data. Towards studying the effect of using ports data on detecting brute force attacks, we conduct two sets of experiments: (1) a feature set containing source and destination ports data, and (2) a feature set without ports data. The remainder of the section presents the above steps in further detail.

A. Data collection and labeling

We collected full packet data from a live production network over a 24 hour period. A production network is a live computer network connected to the Internet that reflects volume and diversity of real world network traffic. Our case study involves a campus network with approximately 300 users, four different subnets and multiple servers such as: domain controller, Web, FTP, Email, and DNS servers. The data collection server collects all the traffic entering and exiting the network via three collector sensors located at different parts of the network. Snort [14] log files are used to extract full packet data in the collection server. Subsequently, we then extract network flow data from the full packet captures using SiLK.

Network flow describes network sessions in terms of an aggregation of packets that share some certain properties during a specific time interval. These properties are some key packet features, which most commonly are source IP address, destination IP address, protocol type, and source and destination ports (for UDP and TCP packets). We note that our definition of network flow is based on the IPFIX standard. Once a flow record has been initiated there are only two ways it can be terminated. The IPFIX standard states that when no data for a flow has been received within 30 seconds of the last packet, the flow record is terminated, and when a flow has been open for 30 minutes, the flow record is terminated and a new flow record is initiated.

We consider 8 different features for each flow. Features and their descriptions are shown in Table I. We do not use IP addresses as predictive features to avoid making out analysis special/customized to just the particular network of our case study. During our analysis we realized that there were some source IPs that are just producing attack data. Eliminating the source IPs prevents the classifiers from simply considering attacker source IPs for the detection of attacks. Since we want to investigate the effect of having ports in the feature set we conduct two kinds of experiments. In one experiment we use ports within the features set, while in the other experiment we do not use ports as features in the process of building models.

After producing the flow data, manual analysis is conducted by network experts in order to detect and label the brute force attacks in the data. Such analysis is done based on Snort alerts as well as analysis of top talkers. Top talkers consist of IPs with the most number of connections, amount of time spent per connection, total aggregated connection, number of aggregated packet and most amounts of bytes coming in and leaving the network. They originate from tools found in SiLK such as rwstats, rwcut and rwfilter. Correlating the statistical results achieved by analyzing top talkers and Snort alerts along with applying visual analysis helps the network experts to detect the SSH brute force attacks. After the labeling is done, each single network flow in the data is labeled as being a brute force attack or not a brute force attack. The final labeled data is thus ready for training the machine learners.

B. Applying machine learning methods

We chose four classification learners for our analysis: 5-Nearest Neighbor (5-NN), two forms of C4.5 Decision Trees (C4.5D and C4.5N), and Naive Bayes (NB). These learners were all chosen due to their relative ease of computation and their dissimilarity from one another. Different learners are used in our study since a broader analysis on the ability of machine learning algorithms in detecting brute force attacks can be investigated. We build all models using the WEKA machine learning toolkit [15], and using its default parameters' settings. For all classification models, "brute force" class was considered as the positive class and "not brute force" class was considered as the negative class. A brief description of each learner including their relevant parameter settings is presented below; however, for additional details the reader is referred to [15].

K-nearest-neighbors or K-NN is an instance learning and lazy learning algorithm. It only uses the training data to build the learnt hypothesis. The predicted class for every test instance is derived from the classes of the K closest samples to that instance (in our study, K=5). Since for each test sample K-NN needs to calculate its distance to all the training samples to specify the K nearest samples to the given sample, K-NN has a $O(n^2)$ complexity that makes it a computationally expensive algorithm.

C4.5 decision tree (implementation of the j48 decision tree in WEKA) is a tree-based algorithm in which a decision tree

structure is determined. Each branch divides the samples into 2 or more other branches based on the values of one of the features in the data sample. The C4.5 algorithm uses a normalized version of Information Gain to decide the hierarchy of useful features in building tree branches. The more information gain the feature has the higher it appears in the tree structure. In this study, we employed a version of C4.5 using the default parameter values from WEKA (denoted C4.5D) as well as a version (denoted C4.5N) with Laplace smoothing activated and tree-pruning deactivated.

The Naïve Bayes algorithm uses Bayes' theorem to calculate the posteriori probability of an instance being a member of a specific class. Unfortunately, it is very difficult to calculate the posteriori probability directly. Therefore, certain assumptions are made along using Bayes' theorem to calculate posteriori probabilities. While, these assumptions make Naïve Bayes a relatively weak learner, it is a fast classifier.

The four classification models are trained and evaluated using cross-validation (CV) based on the labeled dataset generated with the network flow data for our case study. A 5-fold cross-validation is used in our experiments. Moreover, to mitigate any bias due to a random lucky/unlucky split to create the folds, the cross-validation based model training and evaluation is repeated 4 times, and the average performance of the machine leaners across the 4 runs is then evaluated. The classification performance metric used in our study is the effective and commonly used Area Under the Receiver Operating Characteristic Curve (AUC). The AUC builds a graph of the

Table I: Features used for flow analysis

Feature	Description
Source Port	Source port seen in the flow
Destination Port	Destination port seen in the flow
Number of Packets	Total number of packets seen in the flow
Number of Bytes	Total number of bytes seen in the flow
Duration	Flow duration time
Flow Flags	Cumulative OR of all the TCP flags seen in the flow
Initial Flags	The flags of the first packet seen in the flow
Session Flags	Cumulative OR of all the TCP flags seen in the flow except the TCP flags of the first packet

True Positive Rate (TPR) vs. the False Positive Rate (FPR) as the classifier decision threshold is varied, and then uses the area under this graph as the performance across all decision thresholds. AUC demonstrates the trade-off between TPR and FPR, where higher AUC values indicate a high TPR and low FPR which is preferable in the current application, i.e. network attack detection.

We performed two kinds of analysis based on including/excluding ports in our feature set. Popular victim services of SSH brute force attacks usually use a well-known Internet port such as TCP/22, although System Administrators can and sometimes do change the destination port for the SSH service to something different than the standard default port. Also, attacks usually use source ports greater than 1024. As ports can provide some information which directs us to detection of an attack we decided to apply the experiments with and without using ports data as features in our network flow samples, and observe how providing port information can modify the results. Our results suggest that including ports in the feature set can improve the classification performance and it makes changes in the built model based on the type of the classifier used. However, one significant benefit of not using ports in the dataset is that since destination ports for the SSH service can arbitrarily be changed by Systems Administrators to other values, not including ports in the dataset produces a more robust model which can accommodate the scenario of when the SSH service is not running under the standard default port.

IV. RESULTS

For each classifier, 4 runs of 5-fold cross-validation is applied. This produces 20 AUC values for each classifier. The results shown in Table II are some statistics calculated over the AUC values of each classifier. The results present AUC values for both cases of using and not using ports in the feature set. As the standard deviation (std) of AUC values are low we can use the mean of AUC values to compare the performance of different classifiers. We observe that, overall, the classifiers are performing well in the detection of brute force attacks as shown by AUC values greater than 0.97. The 5-NN learner has the highest classification performance with an AUC value of 0.9998 when using ports in its feature set and an AUC value of 0.9902 while not using ports in its feature set – the two models for 5-NN are relatively similar. The classifier performances shown in Table II suggest that using machine learning algorithms for the detection of SSH brute force attacks produces very good results with a high detection rate and low false alarm rate.

Comparing the classification results from two kinds of experiments with or without the inclusion of ports in the feature set, shows that when we use ports as features the classification performance of 5-NN and C4.5N doesn't really improve significantly. In contrast, the performance of C4.5D does improve slightly and the classification performance of Naïve Bayes increases from 0.9707 to 0.9975. The Naïve Bayes algorithm is traditionally a weak algorithm, and in our study for the case of not using ports in the feature set, the other classifiers outperform it. However, by adding ports to its feature set Naïve Bayes yields similar performance results compared to 5-NN and C4.5D.

The constructed tree structures of the C4.5D and C4.5N are the same at the respective first levels, but then C4.5N adds more levels to its tree structure. The likely reason is that no pruning is performed in C4.5N. Figure 1 shows a segment of the common structure in the C4.5N and C4.5D trees (a sample case is presented) while not using ports in the feature set. Due to space limitation the whole tree is not shown. Considering

the common part of the tree structures in C4.5D and C4.5N we observe that when we don't use ports in the feature set "number of bytes", "flow flags" and "number of packets" features appear in the first, second and third levels of the trees respectively. "Duration" then sometimes appears on the fourth level.

The main characteristic of a brute force attack is a relatively high number of failed login attempts in a specific time interval. Here the maximum time interval we consider is 30 minutes with respect to the definition of our extracted network flows that we provide in Section III.A. Hence, it is reasonable to see a specific high number of packets in a brute force flow. However, as failed login attempts do not include big packets the number of bytes in the respective flow is not high. On the other hand, failed login attempts in a SSH brute force flow record will still contain all TCP flags seen in a complete TCP communication since failed logins still build a complete TCP connection. Our experiment included the FIN, SYN, PSH, and ACK TCP flags as part of these SSH brute force flow records, however in general the TCP SYN and TCP ACK flags will always be present at a minimum in order to indicate a successful TCP connection. The decision tree classifier reflects these characteristics in the resulting tree structure. At the first level the number of bytes in the flow is examined, and if the number of bytes is less than a specific amount then flow flags are checked in the next step to see whether all the TCP flags necessary for a completed TCP connection are seen in the flow. In the third level, the number of packets is checked that should be relatively high. Duration of the flow is then used in the fourth level to make the classification more accurate based on the time duration attackers tend to choose for applying the attack.

When including ports in the feature set, the features "initial flags", "number of packets" and "destination port" construct the first, second and third levels of the tree structure respectively. Most of the time destination port leads to a labeled leaf and there are no more branches after that. Sometimes source port comes after the destination port as the fourth level to refine the classification.

Upon comparing the tree structure when not using ports in the feature set with the tree structure when using ports in the feature set, we observed the following: (1) In the latter, the feature "number of bytes" is not used, and (2) The feature "number of packets" and a flag related feature appeared at the high levels of the tree structures. In this case however, "initial flags" which represents the flags of the first packet is used in contrast to "flow flags" which represents all the flags seen in the whole flow. Based on these two observations, using ports in the feature set has resulted in: the tree eliminating a characterizing feature such as "number of bytes", using only the "initial flags" feature instead of the "flow flags" feature, and adding destination port to the third level. Sometimes source port is used at the fourth level after the destination port to assist with classification. Avoiding some discriminating features and using ports features instead, along with increasing the classification performance, suggests that port features can be helpful in the detection of brute force attacks using C4.5D. However, we note that using ports make the tree structure larger due to the fact that ports are actually categorical features with a lot of different values in which there is one branch for each port value. Moreover, larger trees are more likely to have the over-fitting problem in machine learning.

Even though using ports is helpful in C4.5D it doesn't really make significant performance change in C4.5N. The likely reason is that in C4.5N the tree structure is completed and includes finer branches (lower-level) than in C4.5D. In fact, no pruning is done in C4.5N. In both cases of using/not-using ports in the features set, C4.5N yields a complete tree. Consequently, it provides relatively similar classifier performance results in both cases, suggesting that ports might not be very useful when making a more sophisticated tree.

The 5-NN classifier provides similar performance by including or excluding ports in the feature set. It can be interpreted that even by not considering the ports in the feature set the samples are still distributed in the feature space in a way that samples with the same label are distributed close to each other.

Naïve Bayes has the largest performance improvement when using ports in its feature set. In Naïve Bayes, features are assumed to be independent. Although it is not a realistic assumption, Naïve Bayes works well in many cases. Hence, it seems reasonable to assume that feature dependence could be compensated for by the addition of more features to the feature set, and results in improved performance.

It seems clear that using ports as features has different effects on the built classification models based on their nature; however, overall it improves the performance. One can decide to use ports in a feature set for the detection of brute force attacks based on these results; however, some considerations need to be taken into account. Ports are categorical features which can take a value ranging from 1 to 65535. It is not practical and unlikely to see all these values when building a classification model for detecting brute force attacks. As a result, some new port values can be seen at test time that have not been seen during training. This causes the classifier to rely on other features to make the classification decision. A classifier like C4.5D that ignores some characterizing features in building its model because of the presence of ports in its feature set might not work well attempting to classify samples with new port values at test time. In future works we plan to investigate this matter by testing the models on new collected data. Overall, while ports act as discriminative features, they might make a classification model too specific to the training data and can also cause the learner to eliminate some other important discriminating features when building the model. This can happen because rule-based models, like decision trees, tend to prefer using categorical features with a lot of different values when building their models.

To use the discriminating power of ports while not letting the high number of values of those features affect a classifier like decision tree in such a way that it ignores other good

features, one solution would be to use domain knowledge to make a coarser categorization on the port values – we plan to examine this aspect in our future work. Using the domain knowledge on the brute force attack ports can be separated into groups which represent whether they are likely to be participants in a brute force attack or not.

Table II: Average AUC statistics from 4 runs of 5-fold CV

Including ports in the feature set					
Classifier	**mean**	**std**	**median**	**min**	**max**
5NN	0.9988	0.0009	0.9990	0.9964	0.9999
C4.5D	0.9979	0.0008	0.9979	0.9960	0.9993
C4.5N	0.9893	0.0046	0.9880	0.9826	0.9987
NB	0.9975	0.0016	0.9979	0.9938	0.9997
Not including ports in the feature set					
Classifier	**mean**	**std**	**median**	**min**	**max**
5NN	0.9902	0.0019	0.9907	0.9902	0.9934
C4.5D	0.9880	0.0023	0.9885	0.9819	0.9907
C4.5N	0.9892	0.0017	0.9895	0.9846	0.9918
NB	0.9707	0.0040	0.9708	0.9634	0.9783

V. Conclusion

The brute force attack has been and still is, one of the most prevalent attacks on the Internet. In this paper we investigate the use of machine learning methods to detect brute force attacks at the network level by using flow data. To this end we collected real word data which was labeled by network experts. Using the labeled data, we trained 4 different classifiers and evaluated their results based on average AUC values obtained across 4 runs of 5-fold cross-validation. Our results show that with the application of machine learning methods, we can achieve very good prediction results in detecting SSH brute force attacks.

Further, we investigated the effect of using ports in the feature set when training the classification models. Our results suggest that even while ports can improve the classification, it might result in classifiers, such as the decision tree, to ignore some other good features when building the model. This can affect the classification performance in the case of seeing new ports that had not been encountered in the training dataset. This happens because ports are categorical features which can take a large number of potential values. It leads to a bias by the decision tree classifier to use ports in its structure and make a large number of different rules based on different port values instead of making more general rules based on the other discriminating features. Moreover, if at some later time a new port value is seen the classifier is likely to have a difficult time labeling that particular instance. We plan to investigate this matter in our future works by testing models on new collected data.

For future work we aim to collect additional network flow data from the production network and test the trained model to observe its performances on the training data versus the new test data, thus, evaluating how the fitted model will behave on a real-world unseen test dataset. Future work may also include applying machine learning methods in the detection of distributed brute force attacks at the network level.

References

[1] E. Alata, V. Nicomette, M. Kaaniche, M. Dacier, "Lessons Learned from the Deployment of a High-interaction Honeypot," in *Dependable Computing Conference, 2006. EDCC '06. Sixth European*, Coimbra, 2006.

[2] "Hewlett-Packard Development Company. Top Cyber Security Risks Threat Report for," 2010. Available: http://dvlabs.tippingpoint.com/toprisks2010.

[3] SiLK: https://tools.netsa.cert.org/silk/.

[4] R. Bezut and V. Bernet-Rollande, "Experimental Study of Dictionary Attacks on SSH," Technical report, University of Technology of Compiegne, 2010.

[5] J. Owens and J. Matthews, "A Study of Passwords and Methods Used in Brute-Force SSH Attacks," in *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2008.

[6] D. Ramsbrock, R. Berthier and M. Cukier, "Profiling Attacker Behavior Following SSH Compromises," in *Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2007.

[7] C. Seifert, "Analyzing Malicious SSH Login Attempts," http://www.symantec.com/connect/articles/analyzing-malicious-ssh-login-attempts. [Accessed 2006].

[8] DenyHosts: http://denyhosts.sourceforge.net/.

[9] BlockHosts: www.aczoom.com/blockhosts/.

[10] D.Gerzo.BruteForceBlocker: danger.rulez.sk/index.php/bruteforceblocker/.

[11] M. Kumagai, Y. Musashi, D. Arturo, L. Romana,K. Takemori, S. Kubota, and K. Sugitani, "SSH Dictionary Attack and DNS Reverse Resolution Traffic in Campus Network," in *Intelligent Networks and Intelligent Systems (ICINIS), 2010 3rd International Conference on*, Shenyang, 2010.

[12] M. Javed and V. Paxson, "Detecting Stealthy, Distributed SSH Brute-Forcing," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, Berlin, Germany, 2013.

[13] E. L. Malecot, Y. Hori, K. Sakurai, J. Ryou, and H. Lee, "(Visually) Tracking Distributed SSH BruteForce Attacks?," in *In 3rd International Joint Workshop on Information Security and Its Applications*, 2008.

[14] M. Roesch, "Snort - Lightweight Intrusion Detection for Networks," in *Proceedings of the 13th USENIX Conference on System Administration*, Seattle, Washington, 1999.

[15] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P.

Reutemann and I. H.Witten, "The WEKA Data Mining Software," *SIGKDD Explorations,* vol. 11, no. 1, pp. 10-18, 2009.

[16] J. Vykopal, T. Plesnik and P. Minarik, "Network-based Dictionary Attack Detection," in *Future Networks, 2009 International Conference on*, Bangkok, 2009.

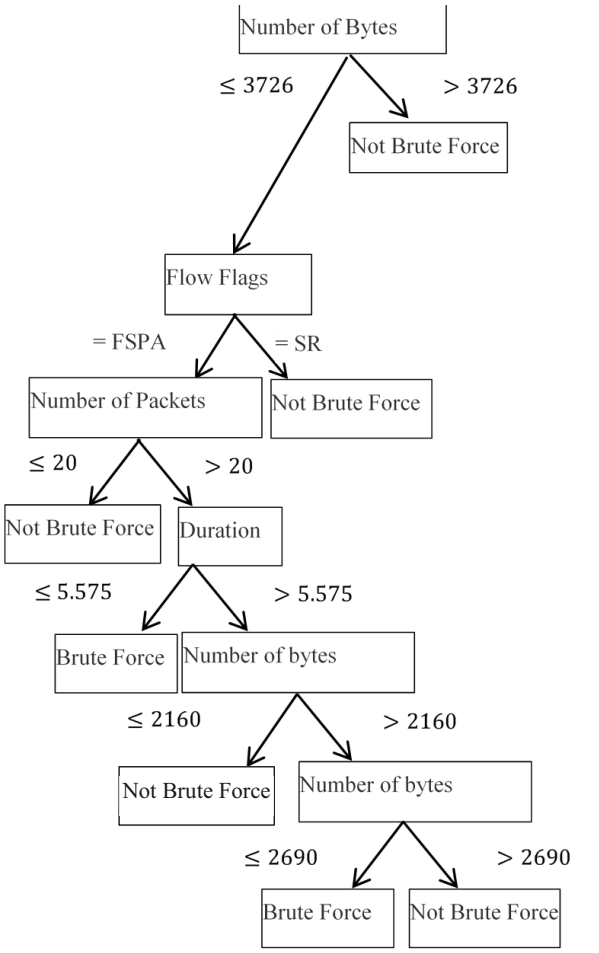

Figure 1: Partial view of the common parts of C4.5D and C4.5N decision trees when not using ports in the feature set