Refined Detection of SSH Brute-Force Attackers Using Machine Learning

Karel Hynek, Tomáš Beneš, Tomáš Čejka, Hana Kubátová

Refined detection of SSH brute-force attackers using machine learning

Karel Hynek[1,2], Tomáš Beneš[1,2], Tomáš Čejka[2], and Hana Kubátová[1]

[1] FIT CTU in Prague, Czech Republic
{hynekkar,benesto3,kubatova}@fit.cvut.cz
[2] CESNET a.l.e., Czech Republic
{hynekkar,tomas.benes,cejkat}@cesnet.cz

Abstract. This paper presents a novel approach to detect SSH brute-force (BF) attacks in high-speed networks. Contrary to host-based approaches, we focus on network traffic analysis to identify attackers. Recent papers describe how to detect BF attacks using pure NetFlow data. However, our evaluation shows significant false-positive (FP) results of the current solution. To overcome the issue of high FP rate, we propose a machine learning (ML) approach to detection using specially extended IP Flows. The contributions of this paper are a new dataset from real environment, experimentally selected ML method, which performs with high accuracy and low FP rate, and an architecture of the detection system. The dataset for training was created using extensive evaluation of captured real traffic, manually prepared legitimate SSH traffic with characteristics similar to BF attacks, and, finally, using a packet trace with SSH logs from real production servers.

Keywords: SSH · Brute-force · Attack · Security · Network · Monitoring · Malicious · Flow · AdaBoost · Decision tree · Classification

1 Introduction

A brute-force (BF) is a common type of attack that may lead to intrusion and taking control by an attacker. A study [20] published by Ponemon Institute in 2014 claims that 51 % of interviewed companies experienced SSH related adverse events. Unfortunately, the situation has not changed dramatically to this day based on recent annual security reports by Cisco [8] mentioning BF attacks.

Detection of incoming BF attacks against own server can be easily performed using server logs. However, our scope of interest is rather a detection of sources of attacks at the network level. Accurately detected attacks against a remote server originating in the operated network infrastructure are usually an indicator of compromise, i.e., valuable information about suspicious behavior.

More sophisticated attacks are performed according to "low and slow" tactic [14] to be hidden from detection systems. It usually includes a large coordinated botnet attacking multiple various targets, so the number of repeated attempts from a single IP address against a single server can be very low. The attacking IPs therefore surpass the host-based solutions because their individual

contribution does not reach thresholds for blocking. Contrary, it is possible to detect such attacks on the ISP (Internet Service Provider) level, which is our goal, because the attackers' traffic targeting multiple victims is observable.

At the ISP level (in high-speed networks), the packet-based monitoring and traffic analysis is very resource-intensive. Therefore in practice, a flow-based monitoring is used. That means aggregated information about communicating parties is represented by IP flow records, usually in the NetFlow or IPFIX format [9,21]. Even though SSH traffic is encrypted, some published papers show that it is possible to detect BF attacks that use this encrypted protocol (e.g., [16]). Based on our experience, the accuracy of the detection system, which uses traditional IP flows, is limited due to false-positive alerts. For example, our instance of the NEMEA SSH brute-force detector [6,19] (based on the SSHCure [12] algorithm), reported several hosts from the monitored infrastructure as attackers. However, these hosts did not show any other signs of infection or misbehavior.

During the analysis of the relevant traffic, we discovered that it comes from automated tools (e.g., monitoring) and causes false-positive alerts. The tools produce multiple SSH connections periodically with a few seconds/minutes interval, each containing a single command. Despite successful authentication (usually by a public key), the observed IP flows can be easily misinterpreted as BF attacks.

Existing IP Flow-based detection algorithms suffer from the same issues with misinterpreted legitimate traffic, since the traditional IP flow data indeed do not contain information that would distinguish such automated traffic from BF attacks. A typical workaround solution can be a change of threshold values to filter out these connections or to use some whitelisting. Unfortunately, both solutions have disadvantages: a higher value of thresholds decreases detector's sensitivity, and the whitelisting is complicated to maintain. Therefore, we designed a better detector based on machine-learning (ML) and extended IP flow-like data.

Recent papers describe extended IP Flow records (e.g., [5,13,27]) with additional information extracted from unencrypted protocol headers up to application layer (L7). Encryption decreases visibility into the traffic and prevents the extraction of the L7 information. However, the behavior of applications discloses information even in the encrypted traffic, as Anderson et al. described in [3].

Compared to the traditional IP Flow data representing one-directional "connections", modern monitoring systems are able to pair both directions into so called *biflow* records. Additionally, Joy exporter [4] adds various additional traffic features at the packet level (such as length and inter-packet gaps of individual packets). The whole feature vectors can be afterward used as an input for ML models — Anderson et al. focus on the detection of malware traffic.

Our paper focuses on addressing false-positives issues in prior works. We have decided to design a detection mechanism using extended IP flows with a minimal subset of the packet-level features created by Joy exporter. The additional information in IP flows should make the resulting model resilient against false-positive detection and more accurate. In contrast to the model, that uses traditional IP flows only, our approach can distinguish between automated traf-

fic from BF attacks, because it takes advantage of additional information such as individual packets length.

To reach our goal, we have worked on the following contributions:

- We have created an annotated training and testing dataset using manually created traffic and automatically captured real traffic of SSH protocol. It contains legitimate flows and BF attacks. The dataset consists of over 30,000 extended biflow records of SSH, about half of them are BF attacks,
- We have experimentally evaluated over 70 traffic features and several ML models to select a suitable subset of (11) features, and a feasible model that achieves the best results. This demonstrates that a small number of extracted traffic features can provide very good accuracy of BF attacks detection.
- We have described an architecture of a detection system consisting of data preprocessing, ML-based detector, and a knowledge base storage for post-processing and filtering detected raw events. According to our experiments, the system performs better than the detection based on pure IP flow data.

This paper is divided as follows. Section 2 describes the related work of SSH BF attacks detection. Section 3 describes the ML method we used. Section 4 describes the obtaining and labeling our dataset. Section 5 contains results of our experiments. Section 6 concludes this work.

2 Related Work

There are many published approaches to the detection of SSH BF attacks. We can classify them as *host-based* and *network-level*. A host-based detection and prevention can be performed either by an application natively, or by external software that does an automatic analysis of the system logs and mitigation of the suspicious traffic using, e.g., host's firewall. A well-known example of this approach is Fail2Ban [11]. This solution usually is not capable of detecting coordinated attacks from many hosts (botnet).

Analysis of SSH brute-force traffic in [1] describes that attacks from botnets are well-coordinated, and dictionaries are efficiently distributed among bots. This main disadvantage of the host-based approach can be solved by a special architecture described in [25]. The proposed architecture shares information between involved hosts to detect or prevent SSH brute-force attacks. However, the proposed solution has only been simulated with a handful of servers and does not provide any comparison with the real-world environment. It also does not provide any consideration for drawbacks associated with communication between large amounts of hosts.

The *network-level* approach depends on observation points. However, it is not affected by any of these disadvantages. On the other hand, it has to deal with a massive amount of data. Therefore, Jonker et al. published a study where they described a specific characteristic of an SSH brute-force attack [15]. They analyzed flat traffic showing that the BF attack has a specific number of packets and duration. Based on these observations, the BF attacks can be detected on network-level only from aggregated data.

The paper [12] proposed an IP Flow-based intrusion detection system for SSH attack called SSHCure. It uses *packet per flow* (PPF) metric and a *minimal number of flows* within a 1 or 5-minute window. The authors build the algorithm on the assumption published in [24] regarding 3 phases of BF attacks:

1. **Scan Phase** – The attacker scans an IP address block to find out a host with running SSH server.
2. **Brute-force Phase** – The attacker is trying to login to a smaller subset of IP addresses using the brute-force attack.
3. **Die-off Phase** – After the successful break-in, there is still traffic from the attacker, but it is much smaller. This residual traffic represents commands executed on the victim machine.

Nowadays, the first Scan Phase can be skipped by attackers, since targets can be found using scanner services like Shodan [23] or Censys [7]. This kind of specialized search engine is designed to scan and gather information about devices and systems accessible from the Internet. With a simple query, we can obtain thousands of IP addresses with the open SSH port.

The SSHCure solves the scenario by allowing attackers to enter the Brute-force Phase without passing through the Scan Phase. The paper [12] presented 98.4 % accuracy of the detector without false-positives. It is worth noting that the algorithm was validated on a dataset with only 130 attacks, where all of the attackers performed the Scan Phase.

A different solution, proposed in [22], combines extended flow with ML techniques. The authors defined a new structure called *sub-flow*, which is an ordered sequence of packet sizes of one IP flow. The *sub-flow* is than used in the clustering algorithm for the detection of the SSH authentication protocol and specific authentication packets. This information is then processed along with the inter-arrival time of authentication packets and compared with the predefined threshold. They achieved very high precision with 99 % of successfully detected attacks.

The advanced ML detection methods were studied in [17]. The authors extracted 18 features from an aggregated extended flow. The aggregated flow is a set of flows with the same source IP, destination IP, and destination port observed within a 5-minute time window. Using the aggregated flows, they avoided the problem of similarity between failed login produced by legitimate users and brute-force attacks. They separated those classes by the number of BF suspicious flows within the time window.

Features extracted from aggregated flows are then used as input for Naive Bayes, K-NN (exactly 5-NN), and C4.5 decision tree algorithm. All algorithms classified with similar precision with AUC metric around 0.995 on a dataset with around 1,000 aggregated flows. This study proved that the ML detector works sufficiently even with a straightforward algorithm such as a decision tree.

Automated tools that use SSH as a transport protocol (e.g., Zabbix, git, rsync, ansible) can create many independent connections having very similar flow features as BF attacks. Based on our experience, detection presented in [22] and [17] would very likely suffer from the high amount of false-positive detection

results. The detection models presented in prior works depend on thresholding suspicious connections identified by standard side-channel features (average inter-packet time, average packet size and so on), and they do not take advantage of the knowledge of SSH protocol auth phase. Therefore, the short and periodic SSH connections are nearly indistinguishable from unsuccessful logins. Although, in the case of [17], extension of the feature set with the information that we present in the following sections will help to separate successful logins from unsuccessful ones. However, even with this modification, we doubt that the same ML-based threshold model would be transferable to other networks since it is data-dependent, and the thresholds are tuned for one particular network traffic.

3 Our Approach

This paper proposes a *network-level* detection architecture to decrease the number of false-positives with the same or higher level of sensitivity, i.e., more accurate. It uses extended IP Flow features provided by the Joy exporter and ML algorithm to distinguish successful and unsuccessful logins. The extended IP flows are obtained from backbone network traffic. The main advantage here is the detection of attackers using the "low and slow" tactic.

The results (identified login failures) are afterward processed by a threshold filter, like in host-based solutions. This allows tuning for the policies of the target network. Contrary to [17], our ML-based algorithm uses protocol-dependent features only, therefore, it is easily transferable to different networks.

The architecture consists of four parts shown in Figure 1 (inside a dashed border), which will be explained in more detail in the following sections.

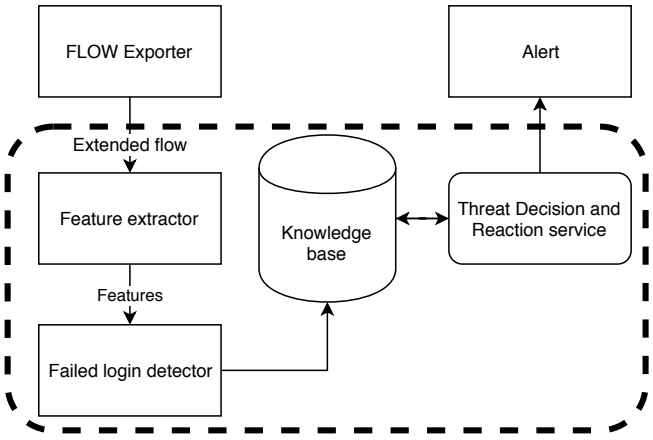

Fig. 1. Overview of the SSH brute-force attack detector

3.1 Feature extractor

The Feature extractor processes extended IP Flows and converts them into a table of features to be usable by ML. The main idea is that the used exporter (Cisco Joy in our case) provides information about every length of packet, but does not provide aggregated statistical features like mean values that we need for the ML method. The Feature extractor takes extended flow as input and calculates the required features for the Failed login detector.

The feature selection is one of the most important parts because the feature set affects the accuracy of the ML classifier. During our analysis, we started with a set of over 70 features. Our deployment target is a high-speed network, where a high number of extracted characteristics is impractical because every single feature extraction consumes many resources. We needed to balance the accuracy and computational complexity of obtaining features.

We successfully reduced the set by removing features with low information gain ratio. Then we continued with a reduction based on performance, and we ended up with 11 selected features that achieved excellent results among multiple ML algorithms while being easily extracted. The final set of features with a description is shown in Table 1.

The feature set includes the length of four specific packets from 9^{th} to 12^{th} positions (only SSH protocol packets are counted, TCP control packets or possible re-transmissions are filtered out by the flow exporter). These packets have been chosen based on our knowledge of the SSH protocol and experimental evaluation of the feature set during its selection. These packets (usually with the same size) carry a response of an SSH server to unsuccessful login attempts. The first packets belong to the SSH handshake (which is not relevant for the detection), and the authentication process starts from the 9^{th} packet (shown in Figure 2). Repeated attempts to log into a target server can be observed within one flow by significant similarity of the sizes of the server response packets included in the feature set. To best of our knowledge, there is none prior work that uses these features to distinguish between BF attack and benign traffic.

Our original feature set included also used version of SSH protocol, client name, supported cipher-suites, and other clients related information. According to our results, these features do not improve brute-force attack detection. We found out that the attackers are using clients like *OpenSSH, PUTTY, libssh* that are also present in the benign traffic. Therefore, the information gain ratio value of client-related features is lower than 0.3.

3.2 Failed login detector

The detector is an ML-based model that performs binary classification on extracted features. The goal is to classify each flow as a successful or a failed login attempt. The classification results of individual flows are then stored in the Knowledge base.

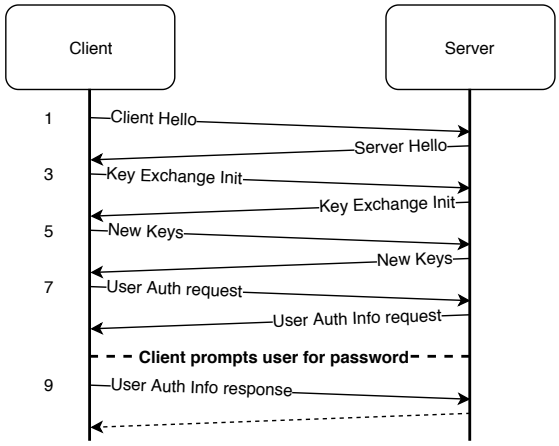

Fig. 2. Conversation diagram of SSH handshake [28,10].

3.3 Knowledge base

The knowledge base is the long term memory of the detector. It covers all information about authentication attempts observed during at least 24 hours in the past. The source IP, destination IP, timestamp, and type of login are stored and used for attack detection in Threat decision and reaction service.

3.4 Threat Decision and Reaction service

The Threat Decision and Reaction service processes stored login reports and decides whether to send an alert. Unlike the traditional host-based solutions, the flows obtained from backbone traffic gives us information about multiple victims of a single IP address. This fact gives us a higher probability of detecting "low and slow" attacks of botnets. The overall and complex information about failed login gives us also other possibilities. For example, when one IP address is constantly failing to log in to multiple servers and suddenly succeeds, it might be an indicator of a possible break-in.

This part can also mitigate potential imprecision of ML-based Failed login detector. The probability of misclassification multiple times in a row is very low. Therefore the false classifications are not going to reach the threshold value. The possible false negatives will only cause increased latency of the attack detection by a negligible number of attempts.

4 Dataset Creation

The most important premise to create a good ML model is a proper dataset. The quality of the model is directly linked with the quality of information contained in the dataset. Due to the lack of useful public datasets available online, we

Table 1. Description of features used for detection of unsuccessful login.

Num.	Name	Inf. Gain ratio	Description
1	duration	0.836	The duration of TCP connection.
2	numPktsIn	0.666	Number of packets transferred from client to server.
3	numPktsOut	0.754	Number of packets transferred from server to client.
4	bytesIn	0.758	Number of bytes transferred from client to server.
5	bytesOut	0.756	Number of bytes transferred from server to client.
6	avgIpt	0.888	Average inter-packet time.
7	medIpt	0.942	Median inter-packet time.
8	dp9bytes	0.837	The length of 9^{th} packet in flow.
9	dp10bytes	0.868	The length of 10^{th} packet in flow.
10	dp11bytes	0.837	The length of 11^{th} packet in flow.
11	dp12bytes	0.754	The length of 12^{th} packet in flow.

have decided to create our own, which should contain malicious and benign SSH traffic. There are two methods on how to create a dataset of SSH traffic with correct labeling.

The first method is to generate malicious traffic ourselves, which can be easily labeled afterward. There are several popular tools to perform a brute-force attack on an SSH server such as THC HYDRA [26] or NCrack [18]. Every tool has a wide variety of options on how to change the characteristics of the attack. Generating malicious traffic from these tools with multiple settings is an extremely time-consuming process. Additionally, resulted dataset would contain only specific kind of brute-force traffic that may not correspond to a real network.

The second method is to use existing traffic from a public server, which is very likely already enduring some form of SSH brute-force attacks. The main downside is that we expose our production server to a potential risk of a successful attack. We did not want to use a custom honeypot or testing server because the resulting dataset would not have the same variety of incoming traffic. Therefore we used Fail2Ban. Contrary to the default behavior that blocks the traffic from an attacker completely, we have prepared an action script to redirect it into an isolated SSH service acting like a honeypot[3].

Our aim was to simulate the same service like the original server (e.g., use of the same certificates), so the attacker should not be able to recognize a change. However, the honeypot has disallowed all user accounts, so it is ensured that every login fails, and we can capture traffic of attacks without interruption. The architecture of the described data capture is depicted in Figure 3.

[3] `https://github.com/CESNET/traffic-datasets/tree/master/ssh/f2b`

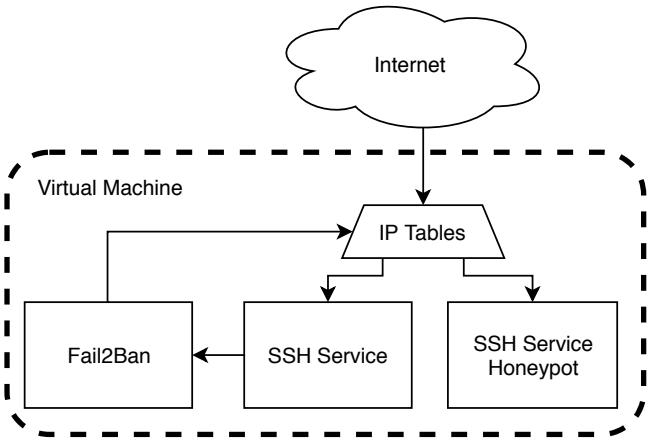

Fig. 3. Architecture of our data collection machine

The captured traffic in PCAP was converted by Joy exporter into extended IP flows. We use the ML approach to detect unsuccessful login so that we could annotate each flow record from the server access log.

Using this method, we created a dataset (*dataset1*) of 35,000 labeled SSH flows, from which 22,000 were malicious, and 13,000 were benign. This dataset represents a realistic traffic that has a slightly higher number of malicious login attempts, which is a similar characteristic as in the real environment. We could not use the exact ratio as it is observed in backbone network because the number of malicious attempts would be very high compared to the number of successful logins.

The dataset is composed of Brute-Force attacks, SSH interactive sessions (authentication by password or public key), file transfer, monitoring by Zabbix, etc. The main advantage of this method is the richness of real attack traffic, which is not restricted to a set of given tools, and easy setup of the traffic capture, which can be deployed on many machines.

Our dataset also covers the main variability in the ssh authentication phase by containing connections encrypted by multiple block and also stream ciphers. Unexpectedly, according to our experiments, the behavior of the server in the authentication phase does not differ between various implementations.

5 Validation and experimental results

This section describes the results of our measurements of the ML-based Failed Login detector and the overall accuracy of proposed detection architecture.

5.1 Accuracy of Failed Login detector

We have considered five ML methods for the flow classification, which are fast and lightweight. They were Ada-Boosted Random tree, Naive Bayes, 5-NN, C4.5

Decision tree, and Random Forest. We evaluated all the models on the *dataset1* (described in Section 4) using 5-fold cross-validation. The overall accuracy is very similar across all evaluated algorithms (except Naive Bayes). The detailed results are shown in Table 2. For further evaluations, we selected the one with the highest accuracy, the Ada-Boosted Random tree.

Table 2. Measured accuracy across all considered ML algorithms.

Algorithm	Accuracy
Ada-Boosted tree	99.47 %
Naive Bayes	92.09 %
5-NN	99.39 %
C4.5 Decision tree	99.46 %
Random Forest	99.38 %

The Ada-boost used ten iterations of creating decision trees with a maximal depth of 10, applied pruning, and minimal leaf size of 2. These parameters provide a good balance between accuracy and possible over-fitting.

The trained model achieved an accuracy of 99.47 % with a pessimistic AUC value of 0.998. Detailed classification results are displayed in form of confusion matrix in Table 3.

Table 3. Failed login detector confusion matrix.

		Ground truth		Class precision
		Successful login	Failed login	
Classified as	Successful login	13,318	130	99.03 %
	Failed login	56	21,698	99.74 %
	Class recall	99.58 %	99.40 %	

5.2 Accuracy of the whole architecture

The results of the ML method are further filtered by the Threat Decision and Reaction service (as it was mentioned in Section 3.4). It triggers an alert only when a several consecutive failed login attempts from a single IP address exceeds a threshold, which was set to 3 in our experiments because it is the default threshold value in Fail2Ban [11]. Therefore, occasional false-positive results of the ML detection usually do not result in an alert. When the detector is evaluated as a whole, including the thresholding, the overall precision on the primary dataset is 100 % with no false positives.

For that reason, we decided to make an additional dataset (*dataset2*) to evaluate the system on real data further. For this dataset, we captured 86,000 bidirectional SSH flows (from 564 clients) from the backbone traffic (peering link between CESNET2 and GÉANT). Table 4 shows overall number of flow classification. We can see that the vast majority of login is marked as unsuccessful and originate from a relatively small number of IP addresses.

Creating *dataset2* is equivalent to deployment on the real network, but the offline evaluation has the advantage of the possibility of a detailed examination of detected anomalies and misclassifications.

Table 5 shows the number of detected brute-force attackers (IP addresses) in the *dataset2* compared to the NEMEA SSH brute-force detector [6,19], which is based on the SSHCure algorithm [12]. We can see, that ML detector is much more sensitive and detected around 40 % more attackers than the NEMEA detector.

Table 6 shows several statistics of our evaluation of the results of the detection. Since the higher sensitivity can increase the number of false-positives, we checked the traffic from the dataset2 and the results of ML-based and traditional detection. To evaluate the false-positive rate, we used AbuseIPDB [2], which provides a list of IP addresses associated with malicious activity. It can be observed that most of the IP addresses detected by the evaluated detectors are listed on AbuseIPDB as malicious. We considered these as true-positives. The remaining ones are possibly false-positives. Since there are only 8 of them, we can investigate them manually.

All three attackers that were detected by NEMEA and not detected by our ML-based method are false-positives. In this case, a simple investigation based on reverse DNS query revealed that these hosts use SSH legitimately for periodic remote access to other servers (e.g., because of monitoring).

The classification of four IP addresses that were not reported on AbuseIPDB but were detected by our method was much more difficult. We marked 2 of them as attackers because they performed a lot of short connections to multiple destinations within one second. The classification of the other two IP addresses was much more difficult. One of them performed short connections with a 5-minute interval from different SSH clients (PUTTY, openSSH1.2 and so on) to multiple IP addresses. The other IP address irregularly shortly connected to one host. Even though we found the behavior highly suspicious, we marked those cases as false-positives.

The last candidate for false-positive IP address detected by both detectors was, according to the SSH-client field, a Raspberry Pi. It was connecting to multiple IP addresses every 10 minutes (default ban time for Fail2Ban). Based on our experience and similarity to other attacks, we marked it as a brute-force attack.

The summary of the results per detector is shown in Table 7. We can observe that the new ML-based approach allows detecting significantly more attackers, while also slightly reducing the false positive rate.

Besides *dataset1* and *dataset2*, we evaluated both detection algorithms using specific traffic samples of a legitimate communication. The samples were gener-

Table 4. The overall number of login classification in *dataset2*. The IP address is counted as unsuccessful, when we detected at least one unsuccessful login.

	Marked as unsuccessful login	Marked as successful login	Total
Flows	85,322	538	85,860
IP addresses	463	101	564

Table 5. Number of detected brute-force attackers from our evaluation datasets

		NEMEA		Total
		Detected	Not detected	
ML	Detected	315	129	444
	Not detected	3	—	3
Total		318	129	447

Table 6. The number of IP addresses detected as attackers which are also listed on AbuseIPDB. The last column shows results of our investigation of those not listed.

	Listed on AbuseIPDB	Not listed	Confirmed FPs
Detected by both detectors	314	1	0
Detected by ML only	125	4	2
Detected by NEMEA only	0	3	3

ated as a traffic from a robot accessing a server via SSH every second performing a single command. The NEMEA detector, i.e., detection based on pure IP flows misclassified this type of traffic as brute-force attacks, meanwhile, our ML-based detection results were correct.

Although our detector achieved excellent results, the amount of reduction of false positives is smaller than we have expected. However, the overall results show that the ML-based algorithm performs much better than the traditional one with almost a third more detected attackers. The active protective systems that use our detector are going to detect more attackers and lower the risk of a successful attack.

6 Conclusion

Brute-force (BF) attacks are a prevalent type of malicious traffic that can be observed by monitoring systems. Successful attacks belong to high severity adverse events. Additionally, it is a typical activity of malware. Therefore, it is essential to focus on their accurate detection.

Modern monitoring systems for high-speed networks use IP flows to represent the observed traffic, and there are published works that describe how to detect BF attacks based on this aggregated information. Some relevant related works and their benefits/drawbacks were described in Section 2. However, after using a flow-based detection algorithm on a real network, we have discovered a significant number of cases where some types of legitimate traffic were misclassified as attacks, which caused false alerts.

Our motivation was to find an improved approach to BF attacks detection at the network level. The aim was to preserve or improve accuracy and decrease the

Table 7. The number of reported attackers and false positives for both detectors

	Total reported attackers	False positives
Detected by NEMEA	318	3 (0.94 %)
Detected by ML	444	2 (0.45 %)

number of false-positive results (without any need for whitelisting or other exceptions that must be maintained). Specifically, our presented detection method focuses on better recognition of legitimate SSH traffic with successful authentication. Contrary to the existing IP flow-based systems, we have designed and evaluated a detection architecture based on machine learning.

To achieve our goal, we have created a large annotated *dataset1* for training and evaluation that contains legitimate communication over SSH (including traffic of automated tools that can cause false-positive alerts due to its flow characteristics) and various BF attacks. Additionally, *dataset2* was prepared using a packet capture of real network traffic to compare the results of the existing non-ML detection and our new ML-based detection. The process of the datasets creation was also described in detail in this paper.

Using the prepared training dataset, which we made publicly available[4], we were able to select 11 traffic features, and train&evaluate several different ML models. Most of them achieved over 99 % accuracy of classifying individual flows as successful or failed login attempts. The best model, AdaBoosted tree, which we selected for further experiments, achieved 99.47 % accuracy.

When the classified flows are fed into the threshold based detector, the whole system achieves perfect accuracy of detection on *dataset1*, i.e., all BF attackers are successfully reported while there is no false alert.

The method is also successful in real-life comparison. On *dataset2*, which represents real SSH traffic on a large network, it detected about 40 % more attackers than an older method implemented in NEMEA, while it also generated less false alerts.

As our future work, we want to continue in our investigation of encrypted network traffic and its analysis. Using the IP flows extended by feature vectors that represent packet-level information, we believe it is possible to classify the encrypted traffic and detect a malicious activity of the network hosts.

Acknowledgment

This work was supported by the Grant Agency of the CTU in Prague, grant No. SGS20/212/OHK3/3T/18 funded by the MEYS of the Czech Republic and the project Reg. No. CZ.02.1.01/0.0/0.0/16_013/0001797 co-funded by the MEYS and ERDF

References

1. Abdou, A., Barrera, D., van Oorschot, P.C.: What Lies Beneath? Analyzing Automated SSH Bruteforce Attacks. In: Technology and Practice of Passwords (2016)
2. AbuseIPDB making the internet safer, one IP at a time. [online] Available: `https://www.abuseipdb.com/` (Oct 2019)
3. Anderson, B., McGrew, D.: Identifying Encrypted Malware Traffic with Contextual Flow Data. In: ACM Workshop on Artificial Intelligence and Security (2016)
4. Anderson, B., McGrew, D., Perricone, P., Hudson, B.: Joy - a package for capturing and analyzing network flow data and intraflow data. [online] Available: `https://github.com/cisco/joy` (Oct 2019)

[4] `https://github.com/CESNET/traffic-datasets/tree/master/ssh/`

5. Cejka, T., et al.: Using Application-Aware Flow Monitoring for SIP Fraud Detection. In: Intelligent Mechanisms for Network Configuration and Security (2015)
6. Cejka, T., et al.: NEMEA: A framework for network traffic analysis. In: 12th International Conference on Network and Service Management (CNSM) (2016)
7. Censys. [online] Available: `https://censys.io` (Oct 2019)
8. Cisco 2018 annual cybersecurity report. [online] Available: `https://rfc-editor.org/rfc/rfc3954.txt` (Oct 2019)
9. Claise, B.: Cisco Systems NetFlow Services Export Version 9. RFC 3954 (Oct 2004). https://doi.org/10.17487/RFC3954
10. Cusack, F., Forssen, M.: Generic message exchange authentication for the secure shell protocol (SSH). Tech. rep. (Jan 2006). https://doi.org/10.17487/rfc4256
11. Fai12ban. [online] Available: `http://www.fai12ban.org/wiki/index.php/Main_Page` (Oct 2019)
12. Hellemons, L., Hendriks, L., Hofstede, R., Sperotto, A., Sadre, R., Pras, A.: SSHCure: A flow-based SSH intrusion detection system. In: Dependable Networks and Services (2012)
13. Hendriks, L., et al.: Threats and surprises behind IPv6 extension headers. In: Network Traffic Measurement and Analysis Conference (TMA) (2017)
14. Hota, C., Sadasivam, G.K., Bhojan, A.: Honeynet Data Analysis and Distributed SSH Brute-Force Attacks (Mar 2018)
15. Jonker, M., Hofstede, R., Sperotto, A., Pras, A.: Unveiling flat traffic on the internet: An SSH attack case study. In: International Symposium on Integrated Network Management (IM) (2015)
16. Najafabadi, M.M., Khoshgoftaar, T.M., Kemp, C., Seliya, N., Zuech, R.: Machine learning for detecting brute force attacks at the network level. In: IEEE International Conference on Bioinformatics and Bioengineering (2014)
17. Najafabadi, M.M., Khoshgoftaar, T.M., Calvert, C., Kemp, C.: Detection of SSH Brute Force Attacks Using Aggregated Netflow Data. In: 14th International Conference on Machine Learning and Applications (ICMLA) (2015)
18. ncrack - Network authentication cracking tool. [online] Available: `https://nmap.org/ncrack/` (Oct 2019)
19. NEMEA Bruteforce detector. [online] Available: `https://github.com/CESNET/Nemea-Detectors/tree/master/brute_force_detector` (Oct 2019)
20. Ponemon 2014 SSH security vulnerability report. [online] Available: `https://energycollection.us/Energy-Security/Ponemon-2014-SSH.pdf` (Oct 2019)
21. Sadasivan, G., Brownlee, N., Claise, B., Quittek, J.: Architecture for IP Flow Information Export. RFC 5470 (Mar 2009). https://doi.org/10.17487/RFC5470
22. Satoh, A., Nakamura, Y., Ikenaga, T.: SSH dictionary attack detection based on flow analysis. In: 12th International Symposium on Applications and the Internet IPSJ, 2012
23. Shodan. [online] Available: `https://www.shodan.io` (Oct 2019)
24. Sperotto, A., et al.: Hidden Markov Model Modeling of SSH Brute-Force Attacks. In: Integrated Management of Systems, Services, Processes and People in IT (2009)
25. Thames, J.L., Abler, R., Keeling, D.: A distributed active response architecture for preventing SSH dictionary attacks. In: IEEE SoutheastCon 2008
26. THC HYDRA V. Hauser, "The Hacker Choice (THC) - Hydra". [online] Available: `https://www.thc.org/thc-hydra/` (Oct 2019)
27. Velan, P., Čeleda, P.: Next Generation Application-Aware Flow Monitoring. In: Monitoring and Securing Virtualized Networks and Services (2014)
28. Ylonen, T.: The Secure Shell (SSH) Transport Layer Protocol. Tech. rep. (Jan 2006). https://doi.org/10.17487/rfc4253