Detection of HTTPS brute-force attacks in high-speed computer networks

Jan Luxemburk, Karel Hynek

Faculty of Information Technology, Czech Technical University in Prague

Introduction

Between 2015 and 2019, the percentage of encrypted web traffic from the Firefox web browser had raised from 35 % to 85 %, and it is expected to continue growing. This is good news, but it brings a new challenge to network operators. There is less visibility into the network, and therefore it is harder to distinguish between legitimate and malicious traffic. One of the most common attack is brute-force attack on web applications. The presented solution operates on the network level and detects brute-force attacks from NetFlow data using machine-learning methods.

Motivation

The thesis focuses on the detection of brute-force attacks from the view of hosting providers and ISP. In [1], three major security threats were identified (1) brute-force attacks result in an increased load on the underlying infrastructure; (2) following a compromise, malicious scripts can be installed, such as remote access shells; (3) web applications can be misused for a range of illegal activities: malware distribution, botnets and DDoS attacks participation. In such cases, the entire IP space owned by the ISP may get blacklisted; thus, a **single customer's security mistake potentially impacts the whole infrastructure including other costumers' services**. Detecting brute-force attacks can be done in multiple ways. Host-based detection can be realized with CAPTCHA or IP-based authentication blockers like Fail2ban. However, such blockers must be implemented and maintained by the customers and are hard to configure and maintain at scale.

NetFlow

A network flow is defined as a set of IP packets passing an observation point in the network during a certain time interval, such that all packets have the same flow key. Traditional 5-tuple flow key is: source and destination IP addresses, source and destination ports, and a transport protocol

Datasets

We created datasets consisting of:

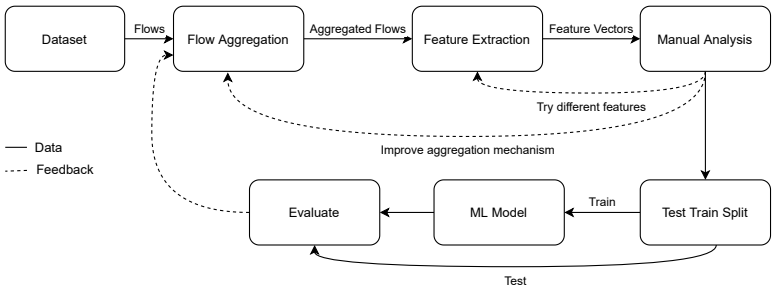
Brute-force traffic data generated by the brute-force simulator. This data includes attacks from Patator, Thc-Hydra, and Ncrack towards these web applications: WordPress, Joomla, Moodle, Mediawiki, Phpbb, Discourse, and Ghost.

Benign traffic from CESNET's backbone network. The data was captured at the perimeter of CESNET's infrastructure in Autumn 2019 and Spring 2020.

Extracted Features

Feature Name
Application Data Records Count
Sent/Received Ratio
Request Size STD
Response Size STD
Mean Request Packets
Mean Response Packets
Roundtrips Count
Roundtrips per Second
Request Response Autocorrelation

Experiment workflow: Feature extraction and tuning of ML algorithm.



Novel Monitoring Approach — Aggregated Flow

Flows were aggregated by the key into: source address, destination address, destination port, and TLS server name indicator. The omission of source port causes aggregation of all flows originating from a specific source address towards the target web application. This solves the problem with attack tools like Thc-Hydra that open a new connection for every attempt. In contrast with the usual practice, the flow aggregation is not time-based. Instead, flows are being aggregated until the number of packets is bigger than some threshold.

Packet-Level Characteristics

Different applications show distinctive properties when their network traffic is analyzed in terms of packet sizes (PS) and inter-packet times (IPT), also called inter-arrival times (IAT). A set of features is extracted from the first N packets of the network flow. Consecutive packets in the same direction are merged, and the result is a sequence of lengths of requests and responses.

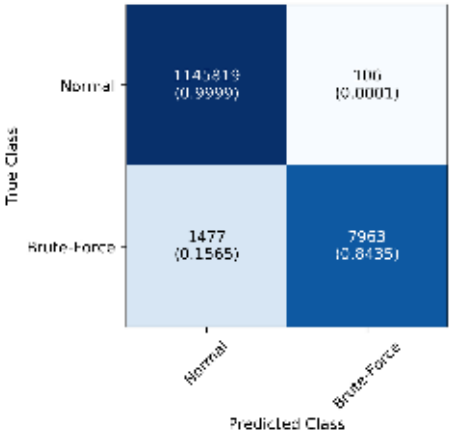
Autocorrelation

The Pearson correlation coefficient of the request-response sequence with a shifted version of itself is calculated. Note that the sequence is encoded as positive numbers for requests and negative numbers for responses. The purpose is to find a periodic signal in the sequence, which would indicate repeated actions in the communication.

Results and Conclusion

The experiments showed that a network-based defense is indeed viable. The LightGBM classifier has the best **recall of 0.84**, and we would consider it as the best brute-force classifier with the right balance between the number of detected brute-force attacks and the number of false positives. The exact numbers of false positives, false negatives, etc., are shown in the confusion matrix. **The resulting detection accuracy is better then similar state-of-the-art solutions.**

	$F_{0.5}$ Score	Recall	FP Rate
Base-line Tree	0.754	0.719	1:500
AdaBoost	0.902	0.723	1:25000
LightGBM	0.953	0.844	1:10000



References

[1] R. Hofstede, M. Jonker, A. Sperotto, and A. Pras, "Flow-Based Web Application Brute-Force Attack and Compromise Detection," vol. 25, no. 4, pp. 735–758. [Online]. Available: https://doi.org/10.1007/s10922-017-9421-4