Research Status: Efficient Detection of Distinct Heavy Hitters for Low Rate Attacks

April 5, 2023

Abstract

The objective of this study is to propose an efficient solution for Low-Rate Attacks (LRA), such as scraping attacks that aim to download all the Uniform Resource Identifiers (URIs) of a website. Attackers attempt to evade detection by behaving like regular users while browsing a small set of distinct pages (URI) at small time scales. However, at larger time scales, the attacker becomes a distinct heavy hitter that requests numerous distinct URIs. Although there are several space-efficient and time-efficient methods to detect distinct heavy hitters, they still require excessive memory to track all users over a large time scale. In this paper, an innovative streaming algorithm is proposed to detect the attacker.

1 Introduction

A current challenge faced by Web Application Firewalls (WAFs) [5] is protecting against scraping attacks. Web scraping, also known as web harvesting, is employed to extract content and data from websites. Common uses for web scraping include the theft of copyrighted content [13][9]. Scraping attacks are often executed by scraping bots or scrapers, which are computer programs that automatically retrieve data from the internet. To evade detection, scraping attackers slowly extract data.

A prevalent solution for detecting web scraping involves identifying whether the user is a bot rather than a human, such as using a Captcha-like solution [1][11]. However, this necessitates altering the user experience. Another approach combines machine learning and heuristic features to determine if the activity is bot-driven rather than human-driven [12][2], but this suffers from false-negative rates. An additional method involves rate-limiting solutions, although this method only caps the number of requests rather than the number of distinct requests, resulting in a higher false-positive rate.

In this paper, we adopt a more direct approach. We focus on the challenging and prevalent scenario of Low Rate Attacks (LRA), where attackers send a distinct number of requests within a small window size, similar to regular users. The attacker becomes a distinct heavy hitter only in larger window sizes.

The primary challenge lies in accomplishing this task while maintaining memory and space efficiency, enabling it to reside within a security device such as WAF. To achieve this, we employ streaming algorithms [10][3], which are real-time algorithms designed for processing data streams in a single pass while utilizing minimal storage.

1.1 Formal definition of LRSA

In this context, we focus on two streaming algorithms: heavy hitters (HH) and distinct heavy hitters (DHH). Consider a stream of triples $D = [(t_1, s_1, u_1), \ldots, (t_n, s_n, u_n)]$, where s_i is the source identifier, u_i is the requested URI at time t_i, and $t_{i1 \leq i \leq n}$ is a monotonically increasing series.

Definition 1 (Heavy Hitter)**.** *A source s is a heavy hitter (HH) if its total number of appearances in the stream is greater than $\epsilon \cdot N$, where N is the size of the stream and $0 < \epsilon < 1$.*

Definition 2 (Distinct Heavy Hitter)**.** *For a source s, the distinct heavy hitter counts the number of distinct u, and N represents the number of distinct u's in the stream.*

Let $D_s = [(t_j, s, u_j), \ldots, (t_k, s, u_k)]$ be a substream of all triples from the same source s. Let D_s^I be a substream within a time window $I = [t_b, t_e]$ that contains all triples with time t such that $t_b \leq t \leq t_e$, i.e., from the beginning of time t_b and ending in time t_e. The time window size is denoted as $t_e - t_b$.

Definition 3 (Low Rate Attack (LRA))**.** *Given a stream D, two window sizes a small time window, T_s and a big time window $T_b >> T_s$, and three thresholds $\theta_1, \theta_2, \theta_3$, where $\theta_1 < \theta_2$ and $\theta_2 << \theta_3$, a Low Rate Scraping Attack (LRSA) is a source s that satisfies the following conditions:*

- *The number of **distinct** URLs in the substream D_s^I with a window size of $I = T_s$ is above θ_1 but smaller than θ_2.*
- *The number of **distinct** URLs in the substream D_s^I with a window size of $I = T_b$ is greater than θ_3.*

In essence, the attacker behaves like a regular attacker in a small window size but acts as a DHH in a larger window size.

2 Background and Related Work

2.1 HyperLogLog

HyperLogLog (HLL) [8] is an algorithm designed to estimate the cardinality or zeroth-frequency (i.e., the number of distinct elements) in large data ensembles called multisets, which typically involve

massive streams (read-once sequences). HLL is a version of the LogLog algorithm [6]. The HLL algorithm estimates the cardinality of a multiset S by hashing all its elements and determining the highest position of the leftmost 1-bit, denoted by R. The cardinality of S is then deduced from R using a stochastic averaging process. This process involves splitting S into m buckets based on the first b bits in the binary representation of the hashed values, where $m = 2^b$. The aim is to process the buckets independently and compute an average of R for a more accurate estimation of the multiset cardinality. A sliding window version of HLL also exists [4].

2.2 Sample and Hold

Sample and Hold (SH) [7] is an algorithm used to identify Heavy Hitters (HH) within a specific measurement interval. The algorithm creates a set, with its size determined by the ϵ defining the HH. SH samples each packet with a certain probability. If a packet is sampled and the flow it belongs to does not have an entry in the set, a new entry is created. Once an entry is created for a flow, the algorithm updates the entry for every subsequent packet belonging to that flow. As a result, a corresponding counter for the flow is maintained in a hash table within the set until the end of the measurement interval.

3 Proposed LRA Detection Solution

A straightforward solution for detecting LRA involves determining the size of the large window T_b and monitoring the distinct count for each source. A naive approach would involve constructing an HLL sketch[8] for each source and tracking the distinct count URL, identifying the attacker when it reaches the threshold θ_2. However, this method is not scalable as it would require 64GB of memory to track 1 million sources. Instead, our approach tracks only the distinct HH since the attacker must be a distinct HH. We adapt the S&H[7] algorithm to track distinct HH by inserting a new source into the set using a hash function $H((s, URL)) \rightarrow [0, 1]$, which determines the probability of a source entering the set. Once a source is in the set, we track its distinct URLs using HLL[8], and stop tracking it if the estimate of its distinct heavy hitters is significantly lower than the others. The key idea of our proposed solution's correctness is that the attacker is active in multiple small windows, and hence will eventually enter the set with high probability. This memory-efficient method requires only 1.6GB for one million sources. Our method's advantages include ease of implementation, compatibility with WAF without affecting user experience, and adaptability to work with any window size. We are currently optimizing the algorithm.

4 Current Status

We have developed the following components to conduct experiments with our approach:

- A simulation of an LRA attacker with configurable parameters of T_b, T_s, and $\theta_1, \theta_2, \theta_3$, where the attacker simulates a scraping attack on a website.
- An implementation of our solution that combines HyperLogLog and S&H.

At present, we are computing the success rate of detecting the attacker. Preliminary results suggest a detection rate above 90%, with a minimal memory requirement of less than $2GB$. The specific values are affected by the attack parameters and the quantity of simultaneous users.

References

[1] Luis von Ahn, Manuel Blum, Nicholas J Hopper, and John Langford. Captcha: Using hard ai problems for security. In *International conference on the theory and applications of cryptographic techniques*, pages 294–311. Springer, 2003.

[2] Andoena Balla, Athena Stassopoulou, and Marios D Dikaiakos. Real-time web crawler detection. In *2011 18th International Conference on Telecommunications*, pages 428–432. IEEE, 2011.

[3] Ziv Bar-Yossef, TS Jayram, Ravi Kumar, D Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 1–10. Springer, 2002.

[4] Yousra Chabchoub and Georges Heébrail. Sliding hyperloglog: Estimating cardinality in a data stream over a sliding window. In *2010 IEEE International Conference on Data Mining Workshops*, pages 1297–1303. IEEE, 2010.

[5] Victor Clincy and Hossain Shahriar. Web application firewall: Network security models and configuration. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 835–836. IEEE, 2018.

[6] Marianne Durand and Philippe Flajolet. Loglog counting of large cardinalities. In *European Symposium on Algorithms*, pages 605–617. Springer, 2003.

[7] Cristian Estan and George Varghese. New directions in traffic measurement and accounting. In *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 323–336, 2002.

[8] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. In *Discrete Mathematics and Theoretical Computer Science*, pages 137–156. Discrete Mathematics and Theoretical Computer Science, 2007.

[9] Pedro Marques, Zayani Dabbabi, Miruna-Mihaela Mironescu, Olivier Thonnard, Alysson Bessani, Frances Buontempo, and Ilir Gashi. Detecting malicious web scraping activity: a study with diverse detectors. In *2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 269–278. IEEE, 2018.

[10] Bengt Rosen. Asymptotic theory for successive sampling with varying probabilities without replacement, i. *The Annals of Mathematical Statistics*, pages 373–397, 1972.

[11] Ved Prakash Singh and Preet Pal. Survey of different types of captcha. *International Journal of computer science and information technologies*, 5(2):2242–2245, 2014.

[12] Athena Stassopoulou and Marios D Dikaiakos. Crawler detection: A bayesian approach. In *International Conference on Internet Surveillance and Protection (ICISP'06)*, pages 16–16. IEEE, 2006.

[13] Rida Yaqoob, Muhammad Haris, Munam Ali Shah, et al. The price scraping bot threat on e-commerce store using custom xpath technique. In *2021 26th International Conference on Automation and Computing (ICAC)*, pages 1–6. IEEE, 2021.