Hindawi
Security and Communication Networks
Volume 2018, Article ID 9065424, 7 pages
https://doi.org/10.1155/2018/9065424

Research Article

Detection Method for Distributed Web-Crawlers: A Long-Tail Threshold Model

Inwoo Ro,[1,2] Joong Soo Han,[1] and Eul Gyu Im[1]

[1] Department of Computer Science, Hanyang University, Seoul, Republic of Korea
[2] NAVER WEBTOON Corp., Bundang, Gyeoungi-do, Republic of Korea

Correspondence should be addressed to Eul Gyu Im; imeg@hanyang.ac.kr

Received 24 July 2018; Revised 7 November 2018; Accepted 15 November 2018; Published 4 December 2018

Guest Editor: Gabriele Tolomei

This paper proposes an advanced countermeasure against distributed web-crawlers. We investigated other methods for crawler detection and analyzed how distributed crawlers can bypass these methods. Our method can detect distributed crawlers by focusing on the property that web traffic follows the power distribution. When we sort web pages by the number of requests, most of requests are concentrated on the most frequently requested web pages. In addition, there will be some web pages that normal users do not generally request. But crawlers will request for these web pages because their algorithms are intended to request iteratively by parsing web pages to collect every item the crawlers encounter. Therefore, we can assume that if some IP addresses are frequently used to request the web pages that are located in the long-tail area of a power distribution graph, those IP addresses can be classified as crawler nodes. The experimental results with NASA web traffic data showed that our method was effective in identifying distributed crawlers with 0.0275% false positives when a conventional frequency-based detection method shows 2.882% false positives with an equal access threshold.

1. Introduction

Web crawling is used in various fields to collect data [1, 2]. Some web-crawlers collect data even though the target site prohibits crawlers by `robots.txt`. Some web services try to detect crawling activities and to prevent crawlers from accessing web pages through anticrawler methods, but some malicious web-crawlers bypass detection methods by modifying their header values or by distributing source IP addresses to masquerade itself as if they are normal users.

Some companies prohibit web-crawlers from access their web pages because of the following reasons: First, web-crawlers may degrade the availability of web servers. Second, contents in the web servers are regarded as intellectual properties of the companies. A competing company may copy the entire data provided in a web server, and the competing company may provide similar services to clients. Even if individual data is open to be browsed by clients, crawling and collecting data from competitors can be treated as a separate issue. For example, there was a crawling lawsuit between Job Korea, Inc., and Saramin, Inc. Saramin crawled resume data from jobkorea.com, and JobKorea filed a complaint against this action. As a result, the court has imposed a fine on Saramin's crawling activities.

This paper investigates the conventional anticrawling methods and various avoidance techniques and shows that the conventional anticrawling methods cannot stop distributed crawlers. Then we propose a new anticrawling method, i.e., LTM (Long-tail Threshold Model) method which gradually adds the distributed crawlers' node IP addresses to the block list. The experimental results showed that our method can effectively identify distributed crawlers with 0.0275% false positives. In the conventional frequency-based method, when the threshold is increased to detect more crawler nodes, false positives increase accordingly.

This paper is organized as follows: Section 2 describes about conventional anticrawling methods and how distributed crawlers can bypass them. Section 3 explains the technique of detecting distributed crawlers using the long-tail region and characteristics of the region. In Section 4 we compare the false positive rate of the LTM with that of the conventional access frequency-based anticrawling technique.

Section 4 describes how the LTM solves this problem. Sections 5 and 6 summarize the paper and explain future works.

2. Related Works

In this section, we summarized conventional anticrawling methods and their counter crawling measures.

2.1. Filtering Using HTTP Header Information. A basic crawler will send requests without modification on its header information. Web servers can distinguish a legitimate user from a crawler by checking the request header, especially if the User-Agent value has been set properly. This header checking method is a basic anticrawling method [3]. However if a crawler attempts to masquerade itself as a legitimate user, it will replay with the header information from a web browser or with the HTTP header information similar to a browser. This makes it difficult for a web server to determine whether a client is a crawler or a legitimate user by simply checking the request header.

2.2. Access Pattern-Based Anticrawling. An access pattern-based anticrawling method classifies legitimate users from crawlers based on the request patterns generated by clients. If a client requests only specific web pages continuously without calls to web pages that should normally be requested, the client will be regarded as a crawler. A crawler performing an aggressive crawling predefines the core web pages that the crawler wants to collect, and the crawler requests specific web pages without requesting unnecessary web pages. In this case, a web server can recognize that the client is not a legitimate user. With a web service that analyzes access patterns of clients, the service can distinguish crawlers from normal users based on predefined normal users' access patterns [3]. Although this approach can recognize a crawler based on access patterns, some crawlers even masquerade their access patterns by analyzing network logs [4].

2.3. Access Frequency-Based Anticrawling. An access frequency-based anticrawling method determines whether a client is a crawler or a legitimate user by the access frequency threshold as the maximum number of access within a specific time window. If the number of requests from a client exceeds a certain threshold within the predefined duration, the web server classifies the client as a crawler [4]. This approach has two well-known problems. First, it has vulnerability against distributed crawlers. If an attacker uses distributed crawlers such as Crawlera, the access rate of each crawler node can be managed to stay lower than the threshold. Second, there is a chance to detect normal users that share a single public IP address as a crawler.

2.4. CAPTCHA. CAPTCHA provides a separate test step to identify users, which must be passed first in order for users to use web services. CAPTCHA can be used to defend against crawlers but has a trade-off between security and user experience, for example, requesting additional actions from users for verification at the login phase. In addition, there is a possibility that a malicious user can reuse a session key after completing the verification normally and can execute crawlers with the session key.

3. Blocking Distributed Crawlers

As described in the previous section, distributed crawlers can bypass conventional anticrawling methods. In this section, we propose a new technique to detect and to block distributed crawlers that could not be defended by conventional anticrawling techniques.

3.1. Required Number of Crawler Nodes. In order for distributed crawlers to collect the entire data of a website, the following conditions must be met:

$$C_n \geq \frac{U_m}{(T_d * 30)} \tag{1}$$

where U_m is the number of items updated in a month, T_d is the maximum number of requests per IP address, C_n is the number of crawler nodes (IP addresses), and 30 is the number of days in a month. T_d multiplied by 30 was to get the number of requests per month.

The crawler node needed to collect all the monthly updated data. The number of monthly update data was divided by the maximum number of requests per month. For example, if there are web service updates U_m (e.g., 30,000) items in a month and the service has a restriction rule that an IP address with more than T_d (e.g., 100) requests will be blocked and an attacker who tries to collect every item from the web service will need C_n (e.g., 10) crawler nodes to avoid the restriction. Therefore, as U_m increases or T_d decreases, C_n should be increased, and C_n numerically indicates the level at which the website is difficult to crawl.

3.2. Generating Long-Tail T_d Zones. The number of items that are updated in a month can not be arbitrarily increased. Therefore, a simple way to prevent distributed crawlers is to decrease T_d, but this will also increase false positives significantly. In this paper, we solve this problem by reversing the general characteristics of web traffic and using the fact that distributed crawlers try to replicate the entire data of a web server. If items are sorted by access rates, we can see the exponentially decreasing curve in the graph, as shown in Figure 1. The most web traffic is concentrated on most frequently requested items [5], and there is a long-tail region that has low access rates. We calculated maximum request counts of this long-tail region and set this value as T_{d3}.

In information theory, unlikely events are more informative than likely events, and the events in the long-tail region are more unlikely than the other events. This means that a web service can find more information from requests in the long-tail region. Hence, when a client keeps requesting items in the long-tail region, a web service can increase the count until it reaches T_{d3}, instead of reaching T_d mean. This means that a web service can set much more sensitive threshold without increasing the false positive rate.

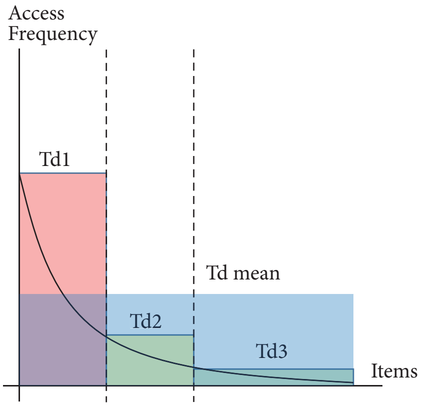


FIGURE 1: Access frequencies per number of connections.

3.3. Node Reducing with Long-Tail Region. In order for an attacker to collect the entire data from a web service, the attacker must also access items in the long-tail region. However, the attacker does not know exactly which items belong to the long-tail region. Using this information asymmetry, service providers can easily identify IP addresses that accessed the items more frequently than other IP addresses. These identified crawler IP addresses will be included in the block list and the number of IP addresses in the block list will be C_m. If we start to increase the C_m value through the long-tail interval, the attacker will crawl with a smaller number of IP addresses, and C_m will be increased in the T_{d3} interval.

(i) C_m: the number of crawler nodes (IP addresses) blocked by a web service

(ii) long_t: ratio of items included in the long-tail region

So attackers must satisfy the following inequality to avoid our blocking method. C_n - C_m is the number of nonblocked crawler nodes and this should be greater than the right-hand side term of the following expression:

$$C_n - C_m \geq \frac{(U_m * long\_t)}{(T_{d3} * 30)} \quad (2)$$

On the service provider side, C_m should be greater than the right-hand side term to block distributed crawlers.

$$C_m > C_n - \frac{(U_m * long\_t)}{(T_{d3} * 30)} \quad (3)$$

If a particular IP address accesses an item in the long-tail region with more than the T_{d3} value determined by the above formula, it can be included in the block list.

3.4. Dummy Items. The service provider may add dummy items to detect crawlers, and dummy items are inaccessible from legitimate users because there are no user interfaces for dummy items or hidden. There are few ways to generate dummy items, it may exists as an HTML tag but it is not displayed on the screen by the attribute setting or it may contains a garbage information that normal users shell not be interested. But a crawler that performs sequential access to the service will may access the dummy items. By this characteristic, dummy items can work as extension of the long-tail region. In this paper, we will not include dummy items in experiments for fair comparisons with real traffic logs which do not contain any dummy items.

TABLE 1: Comparison of log data.

	# of access	# of Items	# of users
NASA	1,701,011	7,649	76,040
Hyderabad	3,208,200	16	81,087

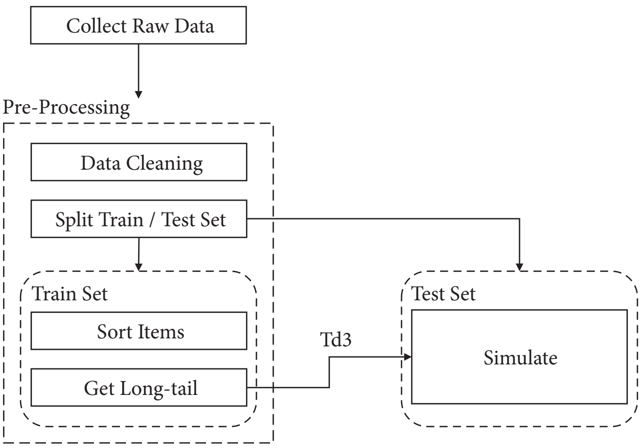


FIGURE 2: Experimental design.

4. Experiments

Our experiments were designed to evaluate the classification performance of our crawler detection module for web traffic. We compared our LTM (Long-tail Threshold Model) method with a normal access frequency-based anticrawling method on the maximum number of crawler nodes and false positive rates. We used the real web traffic logs that NASA released in 1995 [6].

Even though the web traffic logs of NASA are more than 20 years old, there are two factors why we used this dataset in our experiments. One is the number of users, and the other is the number of items in the site. The number of users is important because the traffic patterns can be biased by some users if the number of users is small. For an example, as shown in Table 1, even though the Hyderabad log has a large number of access from a number of users, it would be difficult to build a long-tail zone for simulation since the number of items is small.

To accomplish this purpose, we developed a python-based data tool and a simulator. In the data preprocessing tool, raw traffic data is preprocessed as shown in Figure 2 to count access frequencies for individual URLs and classify sets belonging to the long-tail region. The simulator determines based on preprocessed data whether the accessing node is a crawler whenever a new access occurs.

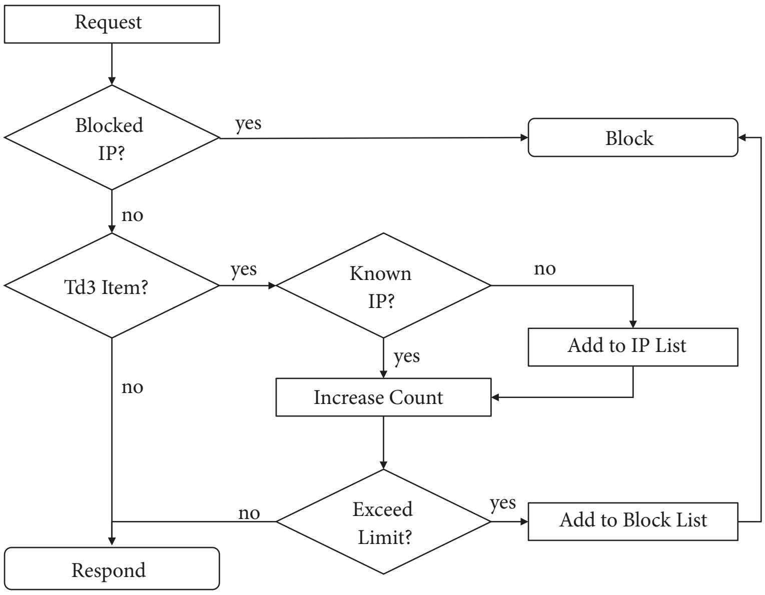


Figure 3: Crawler detection flow.

Table 2: Preprocessed web traffic data.

Total Items	7,649
Long-tail Items	5,355
Mean of Access Counts (Total Items)	184.76
Mean of Access Count (Long-tail Items)	1.88

4.1. Web Traffic Data

4.1.1. Data Source. NASA released a total of 1,891,715 access logs for the month of July 1995. We parsed these logs into the csv format that composed of four columns including IP address, date, an access target, and an access result. The total number of connected IP addresses was 81,978 and the number of items was 21,649.

4.1.2. Data Preprocessing and Traffic Distributions. We performed three steps in the preprocessing phase for the experiments. The first step splits logs into two datasets: a training set and a testing set. Among NASA access logs, first 24-day logs are set as a training set and the last of logs as a testing set. The second step filters out some access logs to calculate more accurate access counts. Some requests are merged as a single request to prevent duplicated counting. For an example, when a user accesses an html file, they also get access to image files that are linked. This can force a multiple increase of access count. Therefore we removed some requests for image files. In addition, we excluded the request logs that the access results are not success from the experiment. After the completion of the preprocessing phase, we have a test set as shown in Table 2.

As a described in Table 2, we built a preprocessed traffic data set that consists of 7,649 items from 21,649 raw data, and it has a long-tail region that consists of 5,355 items. The mean value of total access counts was 184.76, and the mean access count of the long-tail region was 1.88. The difference between two mean values simply shows that we can set a more sensitive threshold in the crawler detection algorithm.

Table 3 describes the characteristics of the most frequently accessed group (T_{d1}) to the least frequently accessed group (T_{d3}) when sorted by frequency of access. Ratio refers to the interval in which each group is located when all items are sorted by access count. For example, T_{d2} is a group located between the top 0.5% and 30%. Mean of Access refers to the average number of access per item belonging to each group, and Maximum Access refers to the maximum access count among items in each group. The term Items means the number of items belonging to each group.

The key part of the experiment was to simulate the algorithm with the real web traffic data. Table 3 shows distributions of the NASA data. We checked whether the NASA data represent power distribution after sorting with access frequencies, and the traffic frequencies have power distributions as shown in Figures 4, 5, and 6.

Figure 4 shows the group of the most frequently requested items.

Figure 6 shows the long-tail region of sorted results. We set the T_{d3} threshold value as 20 which is about twice larger than the maximum access rate of the long-tail region. Setting a T_{d3} threshold value in LTM has a heuristic part because web services have different purposes and circumstances.

4.2. Simulation. In this paper, we implemented simulations for two purposes. One is to check whether LTM would be able to detect and to disable a distributed crawler IP address group and the other one is to check false positive rates when the actual web traffic is inputted to LTM. The overall crawler detection flow is shown in Figure 3.

Table 3: Experiment data.

	Percentage	Mean of Access	Maximum Access	Items
T_{d1}	> 0.5%	21,250	76,040	38
T_{d2}	0.5% - 30%	264	7,043	2,256
T_{d3}	30% >	1.88	9	5,355

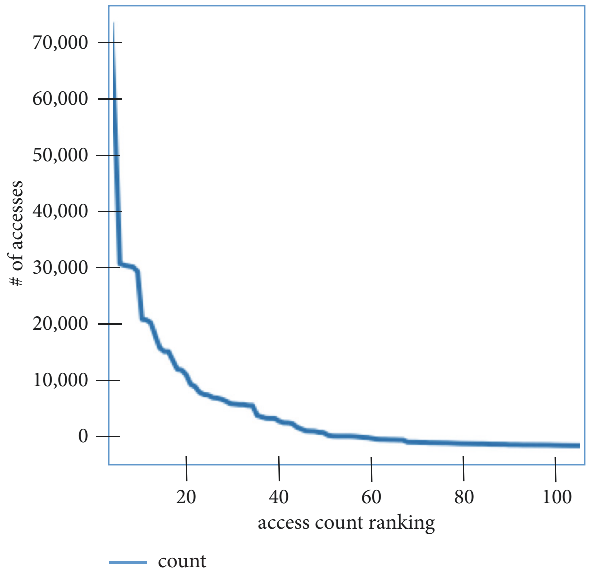


Figure 4: Access count in T_{d1}.

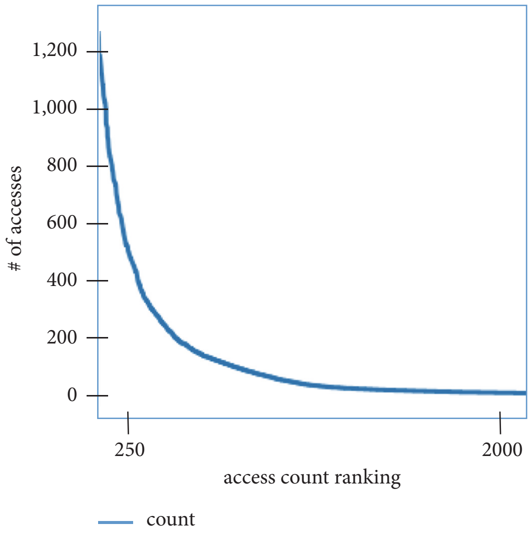


Figure 5: Access count in T_{d2}.

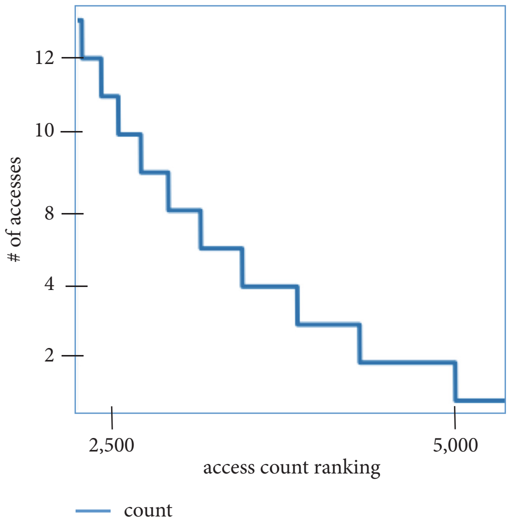


Figure 6: Access count in the long-tail region.

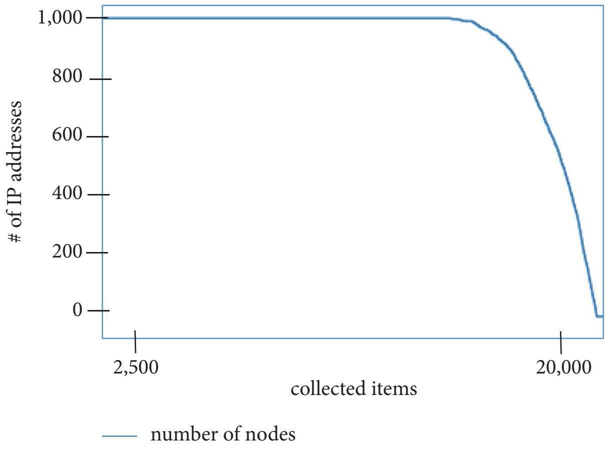


Figure 7: Number of IPs reduced by detection.

4.2.1. Distributed Crawler Detecting Simulation. We used Python to implement the LTM simulator. The required parameters are (1) the size of the distributed IP address set used by the crawler, (2) the long-tail list, (3) the entire item list, and (4) threshold values used for detection. When the crawler accesses an item in the long-tail region, LTM increases the access count of the source IP address. When an access count of an IP address exceeds the threshold, LTM adds the corresponding IP address to the block list. Figure 7 shows an example of running a crawler using 1,000 distributed IP addresses with the threshold value of 20. The long-tail ratio was 70.001% since the number of items in the long-tail region was 5,355 while the total number of items was 7,649.

We can observe that the crawler IP address set is gradually reduced until all the IP addresses are totally blocked. When the first crawler node IP address exceeds the T_{d3} threshold

Table 4: Experiment data.

	Threshold	Max Node	False Positive
LTM	10	426	0.1239%
LTM	20	222	0.0275%
LTM	35	128	0.0046%
FBA	10	573	8.8064%
FBA	20	299	2.8819%
FBA	35	173	0.8903%
FBA	100	60	0.0367%

and was blocked, the node reducing count increases exponentially. This is because other crawler nodes get more burdens and has to access more items when a crawler node was blocked.

4.2.2. Node Reducing Result. Experiments were performed with threshold set to 20 and the crawler set consisting of 222 nodes. As a result, LTM detected entire crawler nodes. False positives were about 0.0275% which is much less than that of the conventional frequency-based crawler detection method. In Table 4, we compared the results of LTM with a normal FBA (frequency-based anticrawling) method.

Detection performance of LTM against distributed crawlers can vary depending on the number of items and a long-tail ratio. Considering this limitation, simulation was conducted using old NASA traffic data (1995); total number of items were quite smaller than usual modern web services. If there is a service with 10 times data and similar access frequency distributions, our proposed method could detect distributed crawlers which consists of 2000 nodes.

In the experiments, we compared the detection capability and the false positives between LTM and FBA with the same threshold and environments. In below graph, LIMIT_LTM is the detection number limit of LTM with given threshold (x-axis) and LIMIT_FRQ is the detection number limit of FBA with given threshold values.

Since LTM uses the long-tail region instead of entire items, FBA detects more crawler nodes with the equivalent threshold value as shown in Figure 8. Figures 9 and 10 show the comparison of false positive rates of two methods.

As we can see in Figures 9 and 10, as the threshold value in x-axis increases false positive rates and the number of detectable crawler nodes are both decreased. In the case of LTM, we can see that the false positive rates are less than 0.15% even with the low threshold values. However, in case of FBA, the threshold value should be set to more than 76 to have similar false positive rates, which can affect the detection performance.

In the experiments, LTM reached the minimum false positive rate of 0.0046% when FBA reached only 0.0367%, which means that LTM has up to 500% better detection score than the classic FBA method with the similar level of false positive rates. Even though FBA can trade off the detection score, FBA generates 798% more false positive rates. When we set the threshold to 35 which is the value that LTM reached the

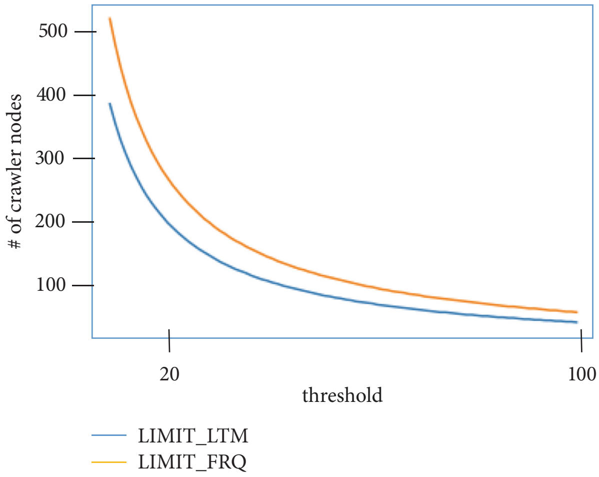


Figure 8: Number of detectable crawler nodes.

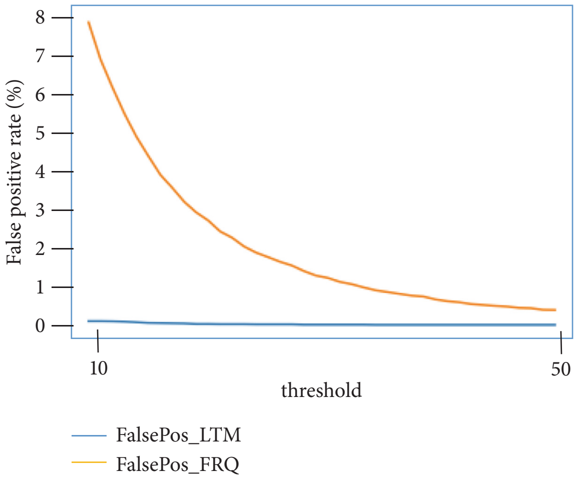


Figure 9: False positive: 1 to 50.

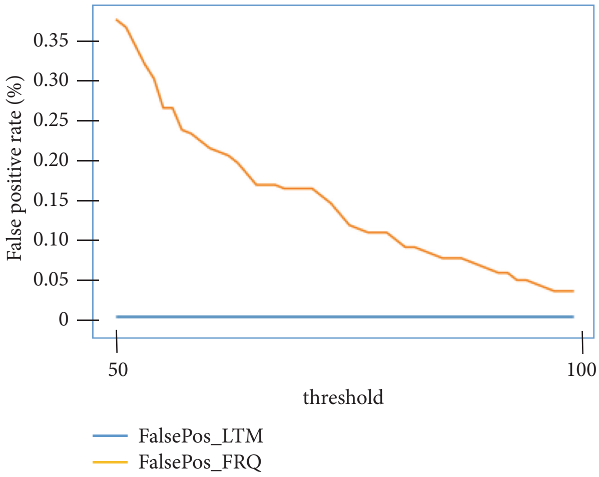


Figure 10: False positive: 50 to 100.

minimum false positive rate, FBA generated 19,400% more false positive rate than LTM.

5. Conclusion

In this paper, we introduced LTM (Long-tail Threshold Model) and showed that how LTM can detect distributed crawlers effectively that other previous methods are vulnerable. By simulating with the real web traffic data, LTM effectively identified distributed crawlers and showed a significantly low level of false positive rates. Illegal web crawling against web services becomes a serious security threat. Considering that there are some crawler developer using distributed crawler proxy service [7] for illegal purposes, LTM could improve data security of web services.

6. Future Works

Web traffic generally tends to generate traffic bursts at certain times [5]. Although the experiments of this paper are based on actual traffic logs, the data used in this paper did not include any complicated cases like adding new items or traffic bursts occur with external reasons since the time period of the dataset used in the experiments contains only one-month data. However real web services should challenge with these complicated circumstances. In order to apply the results of this paper more securely to actual services, it is necessary to study whether the item movement levels and the threshold value of long-tail area can be maintained based on actual traffic data for traffic burst occurrence cases.

Data Availability

Our experiments used publicly available data, i.e., NASA traffic logs.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was partially supported by the Civil-Military Technology Cooperation Program (UM17312RD3) and was partially supported by the Human Resources program in Energy Technology (no. 20174010201170) of the Korea Institute of Energy Technology Evaluation and Planning (KETEP) grant funded by the Ministry of Trade, Industry and Energy of the Korea government.

References

[1] A. Balla, A. Stassopoulou, and M. D. Dikaiakos, "Real-time web crawler detection," in *Proceedings of the 2011 18th International Conference on Telecommunications, ICT 2011*, pp. 428–432, Cyprus, May 2011.

[2] D. Han, K. S. Han, B. Kang, H. Han, and E. G. Im, "Peer-to-Peer botnet traffic analysis and detection," *Information*, vol. 15, no. 4, pp. 1605–1624, 2012.

[3] A. Stassopoulou and M. D. Dikaiakos, "Crawler detection: A Bayesian approach," in *Proceedings of the International Conference on Internet Surveillance and Protection, ICISP'06*, France, August 2006.

[4] R. Inwoo, K. Ku-Hwan, and I. Eul Gyu, "Intelligent Crawler for Dark Web Crawling," in *Proceedings of the Conference on Information Security and Cryptography-Winter*, p. KIISC, 2017.

[5] M. V. Simkin and V. P. Roychowdhury, "A theory of web traffic," *EPL (Europhysics Letters)*, vol. 82, no. 2, p. 28006, 2008.

[6] NASA, *NASA-HTTP log*, 1995, ftp://ita.ee.lbl.gov/traces/NASA_access_log_Jul95.gz.

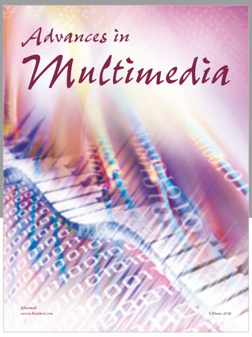
[7] scrapinghub, *Crawlera - The World's Smartest Proxy Network*, 2018.

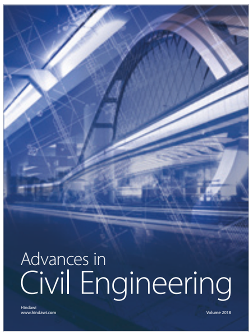
Journal of
Engineering

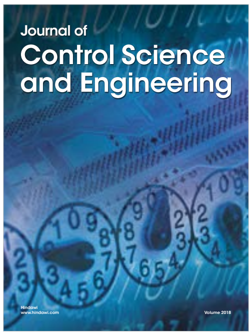
The Scientific
World Journal

International Journal of
Rotating
Machinery

Journal of
Sensors

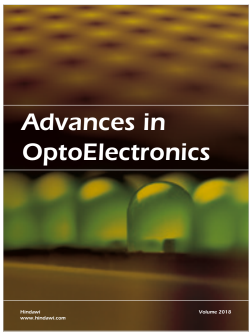

Advances in
Multimedia


Advances in
Civil Engineering


Journal of
Control Science
and Engineering

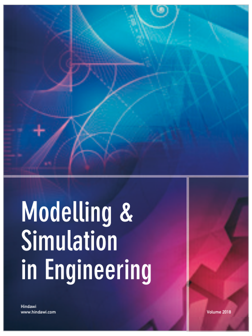
Journal of
Robotics

Journal of
Electrical and Computer
Engineering

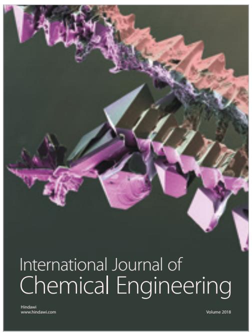

Advances in
OptoElectronics

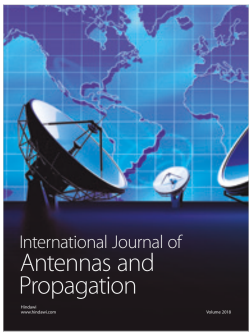
VLSI Design

International Journal of
Navigation and
Observation

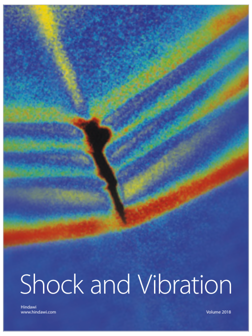

Modelling &
Simulation
in Engineering

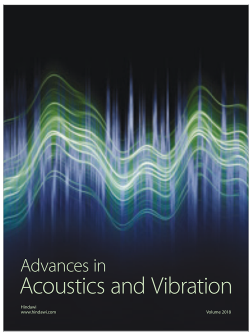
International Journal of
Aerospace
Engineering


International Journal of
Chemical Engineering


International Journal of
Antennas and
Propagation

Active and Passive
Electronic Components


Shock and Vibration


Advances in
Acoustics and Vibration