Anti-Methods for Distributed Web-Crawler

Anonymous Author(s)

ABSTRACT

In this paper, we propose a countermeasure against distributed crawlers. We first introduce known crawler detection methods in general and how distributed crawlers bypass these detection methods. Then, we propose a new method that can detect distributed crawlers by focusing on the property that web traffics follow the power distribution. This means when we sort the items by the number of requests, most of the requests are concentrated on the most frequently requested items.[1] And there will be a longtail area that legitimate users are not generally request. But crawlers will request for it because of their algorithms are intend to request iteratively by parsing web service architecture to collect every items they aimed for. By following these two assumptions, we can assume that if some IPs are frequently requesting the items that are located in longtail area, those IPs can be classified as crawler node. We tested this theory by simulating with real world web traffic data that NASA released.

KEYWORDS

Web Crawler, Traffic Analysis, Power Law, Information Theory

1 INTRODUCTION

Web-crawling is used in various fields for collecting data. Some web-crawlers collect data even though the target site is prohibiting crawlers by robot.txt. And even though web services try to detect and prevent crawlers by anti-crawler methods, these malicious web-crawlers bypass detection by modifying their header values or distributing IPs to masquerade as if they are legitimated users. It is a matter of availability and data property issue because even though any service permits viewing of each data, it is prohibited to duplicate an entire dataset. And malicious web-crawlers can cause significant traffic and intellectual property infringement by collecting the entire data from web services. This can have a serious impact on the availability of the target service. In this paper, we introduce the conventional anti-crawling methods and its countermeasures, and show that the conventional anti-crawling methods cannot defend against the distributed crawler. Then we propose the new anti-crawling technique that we call 'node-reducing' which gradually adds the distributed crawler node IPs to the block-list.

2 RELATED WORKS

In this section, we will describe about conventional anti-crawling methods and their counter crawling measures.

(1) HTTP Header Check
A basic crawler will send requests without modifying its header information. And web servers can distinguishes a legitimate user from a crawler by checking the request header, especially User-Agent value has been set properly. This header checking method is a basic anti-crawling method. But if a crawler attempts to masquerade itself as a legitimate user, it will replay the header information from the web browser or form 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) Access Pattern based Anti-Crawling
Access pattern based anti-crawling is a method of classifying legitimate users and crawlers based on the pattern requested by the client. If a client requests only a specific resource continuously without a call to a resource that should normally be requested, the corresponding could be regarded as a crawler. An attacker performing an aggressive crawling predefines the core data that the web service wants to collect, and implements a crawler that requests specific data without requesting unnecessary resources. In this case, the web server can recognize that the client is not a legitimate user. In the case of a web service using advanced approach to access pattern recognition, the service is viewed as a set of consecutive requests from the viewpoint of the user UX, and requests and responses belonging to the same set are chained by including a specific hash value in the cookie. Although this approach can recognize a crawler based on access pattern, some crawlers even masquerade their access pattern by analyzing network logs.[Inwoo Ro]

(3) Access Frequency based Anti-Crawling
Access frequency based anti-crawling is a method that determines a client is whether a crawler or a legitimate user by access frequency rate. A web server can set a threshold limit of access count in an unit time. If a client with a specific IP requests exceeds this limit in pre-defined time, the web server determines that this IP as a crowler node.
This method has two well known problems. First, it has vulnerability against distributed crawler. If an attackers use distributed crawler such as Crawlera, access rate for a crawler node IP will be reduced enough to bypass threshold limit. Second, it could raise false positive error if many users share a public IP.

3 BLOCKING DISTRIBUTED CRAWLER

As described in previous section, distributed crawler can bypass every conventional anti-crawling methods. In this section, we propose a new technique to detect and block distributed crawlers that could not be defended by existing anti-crawling techniques.

3.1 Required Number of Crawler Nodes

In order for the Distributed crawler to collect the entire data of a website, the following conditions must be met.

ACSAC'18, December 2018, San Juan, Puerto Rico
2018.

$$Cn \geq Um/(Td * 30)$$

The above formula can be described as follows.

- Um: Number of items updated in month
- Td: Maximum number of request per IP
- Cn: Number of crawler node(IP)

For an example, if there is a web service updates Um(30,000) items in a month and the service has a restriction rule that if an IP requests more then Td(100) times will be banned, than an attacker who tries to collect every items from the service use at least Cn(10) crawler nodes to avoid restriction.
Therefore, as Um increases or Td decreases, Cn increases. And Cn numerically indicates the level at which the website is difficult to crawl.

3.2 Generating Long-tail Td zone

The number of items that are updated in a month can not be arbitrarily increased. Simple way to prevent distributed crawler is to decrease Td. But this will also increase false positive significantly. In this paper, we solve this problem by reversing the general characteristics of web traffic and the fact that the crawler tries to replicate the entire data.
If you sort the items by access rate, you can see the exponentially decreasing form as shown in the following figure. This means most of the web traffic is concentrated on most frequently requested items. And there is a long-tail region that has low access rate. We calculated max request count of this longtail region and set this value as Td3.

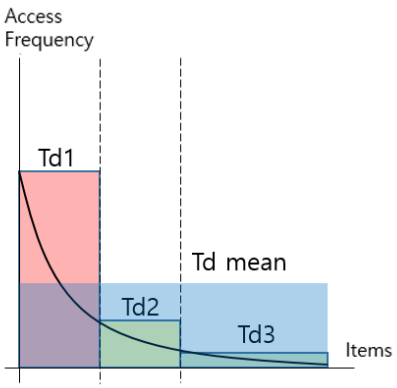

Figure 1: Access Frequency per number of connections

In information theory, unlikely event is more informative then likely event. And the event that a request to an item in long-tail region is much unlikely then the upper items. This means the web service can find more information from a request to the long-tail region.
Hence, when a client keep request the items in long-tail region, the web service increase the count until it reaches Td3, instead of reaching Td mean. This means a web service can set much more sensitive threshold without increasing false positive error rate.

3.3 Node Reducing with Long-tail Region

In order for an attacker to collect the entire data from the service, he or she must also access the items in the long-tail (Td3) interval. However, the attacker does not know exactly which item the item he is accessing belongs to. Using this information asymmetry, service providers can easily identify IPs that are accessed more frequently than long-tailed segments. These identified crawler IPs will included in block-list and the number of IPs in blcok-list will be called Cm. If we start to increase the Cm value through the long-tail interval, the attacker will crawl with a smaller number of IPs, and Cm will increase in the Td3 interval.

- Cm: Number of crawler node(IP) blocked by service
- long_t: Ratio of items included in long-tail region

So attackers must satisfy the following inequality. Cn - Cm is the number of non-blocked crawler nodes and this should be greater then the right term.

$$Cn - Cm \geq (Um * long\_t)/(Td3 * 30)$$

On the service provider side, Cm should be greater then right term to block distributed crawlers.

$$Cm > Cn - (Um * long\_t)/(Td3 * 30)$$

If a particular IP accesses an item in the long-tail region with more than the Td3 value determined by the above formula, it can be included in the block-list.

3.4 Dummy Items

The service provider may include a dummy item to detect the crawler in addition to the actual service target item. The item is normally inaccessible to the general user through the UI. For example, it exists as an HTML tag but it is not displayed on the screen due to the attribute setting or the case where the ordinary user is not interested because it exists on the index but is not in the real world. Such a dummy item may approach a crawler that performs sequential access but it can maintain a relatively low threshold value because the accessibility of the general user is low and it generates a lower interval than the long-tail interval derived from the traffic can do.

4 EXPERIMENT

The experiment was designed to verify classification performance between crawlers and actual web traffics. The key goals were maximum number of crawler nodes that the detection algorithm can identify and false positive, compare to normal access frequency based anti-crawling.
We used the web traffic log that NASA released. Details and experimental method of data are as follows.

Figure 2: Experiment Design

4.1 Web Traffic Data

(1) Source
NASA released a total of 1,891,715 access logs for the month of July 1995. In this paper, the log is parsed into csv format and composed of 4 columns including IP, date, access target and access result. The total number of connected IPs is 81,978 and the number of items is 21,649. The most accessed items received 111,116 requests as '/images/NASA-logosmall.gif'.

(2) Data Pre-Processing and Traffic Distribution
In order to prevent duplication of data used in modeling and experimental data in the time series data, Long-tail was constructed by using data from 1 to 24 days in 30 matching data. Traffic verification was performed from the 25th to the last day Data. When accessing html files, gif extension files are removed from the long-tail group in order to prevent cumulative access values from increasing in duplicate while accessing gif extension files together. And we excluded the request log which was not accessed successfully from the experiment. Finally we made the test set as below table.

Table 1: Pre-Processed Web Traffic Data

Num of Item	7,649
Num of Long-tail	5,355
Mean of Request	184.76
Mean of Long-tail	1.88

As a result, we made a pre-processed traffic data set consists of 7,649 items from 21,649 raw data. And 5,355 items were included in the long-tail region. The mean of the total request count was 184.76, and the mean of long-tail region was 1.88. It means we can set about 6880% more sensitive threshold to the crawler detection algorithm.

Table 2: Experiment Data

	Ratio	mean	max	count
Td1	> 0.5%	21,250	76,040	38
Td2	0.5% - 30%	264	7,043	2,256
Td3	30% >	1.88	9	5,355

And also we could find NASA traffic data also has power distribution as shown in Figures 2, 3 and 4.

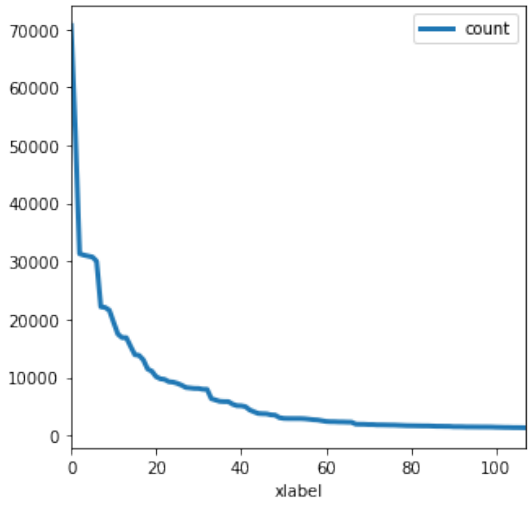

Figure 3: Access Count in Td1

Figure 2 shows the group of the most frequently requtestd items.

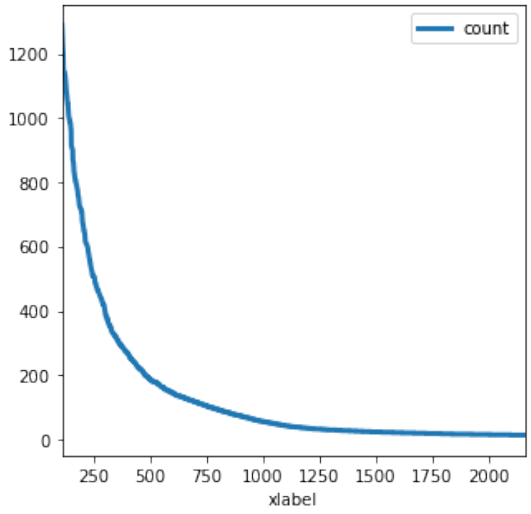

Figure 4: Access Count in Td2

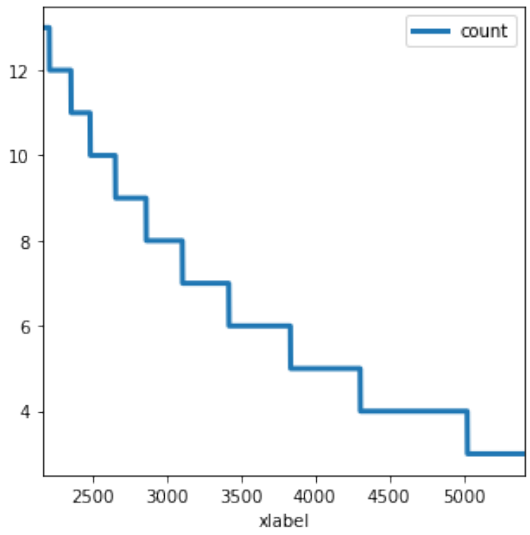

Figure 5: Access Count in Long-tail

Figure 4 shows the long-tail region of sorted result. We can find Td3 threshold value from this group.

4.2 Simulation

In this paper, we implemented two kinds of simulation. One is to check whether it is possible to detect and disable the crawler IP group by performing node reduction through items belonging to the long-tail, and the other is to check false positive when the actual traffic is input to the same detection logic.

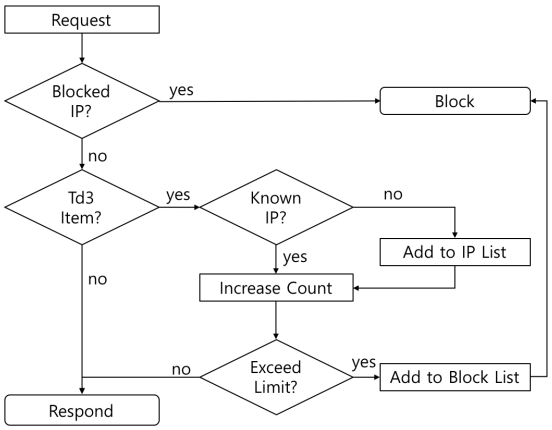

Figure 6: Crawler Detection Flow

(1) Simulators
The simulator is implemented using python. The parameters are the size of the distributed IP set used by the crawler, the long-tail list, the entire item list, and threshold values used for detection. The implementation method allows the Crawler IP Set to access each item by traversing the entire item list, and accesses the IP in the crawler distributed IP set at each access. When the crawler accesses a long-tail entry, it adds the IP to the warning dictionary and increments the access count by one. However, if the same IP accesses the same item, it does not increase the access count because it is not related to the purpose of crawling. When the access count exceeds the threshold value, Node Reducing is implemented by adding the corresponding IP to the banned list. Figure 5 below shows an example of running a crawler using 100 distributed IPs for 7,649 items. The number of long-tail items is 5,355 and the threshold is set to 20.

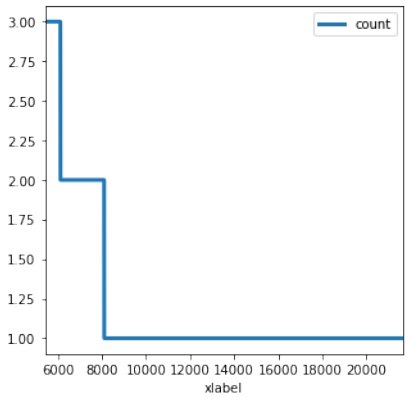

Figure 7: Number of IPs using crawling

The IPs included in the crawler IP set gradually accumulate the access count, and the node reduction starts from the point when the access count of the entire crawler node group increases beyond number of nodes * threshold. Another function of the simulator is to input the request request to the crawler simulator based on the actual web traffic log. This is implemented to confirm the case where the simulator judges the actual web traffic as a crawler.

(2) Node Reducing Result
Experiments were performed with threshold set to 30, and the crawler set consisting of up to 222 nodes was completely detectable. If the number of nodes exceeded 300, it was not detected at all. False positives were 1.33 cases per day, which was 0.0312% of the daily average IP number of 3,631. The following figure is a graph of the process of reducing 222 crawler sets on the simulator.

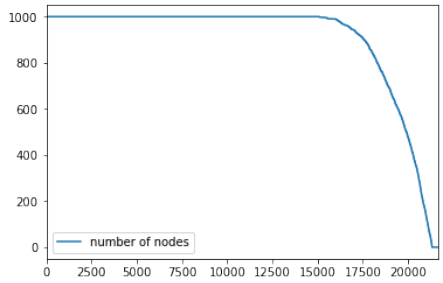

Figure 8: Number of IPs reduced by detection

Table 3: Experiment Results

Num of Item	7,649
Mean of Td3	1.88
Max Node	222
False Positive	0.0312%

This is based on NASA traffic data in 1995, and can be applied to more or less crawler sets depending on the number of items the site has or the length of the long tail.
A total of eight false positives occurred in 6 days of data. Six IPs were recognized as crawler nodes except for duplicate detection. The number of requests generated per month from each IP is as follows.

Table 4: IP and domains generated requests

IP or domain	Number of requests
156.80.168.122	117
163.205.180.17	564
dwkm206.usa1.com	167
jalisco.engr.ucdavis.edu	424
jbiagioni.npt.nuwc.navy.mil	2124
sputnix.cas.und.nodak.edu	101

156.80.168.122 and sputnix.cas.und.nodak.edu, which generated relatively few requests, were detected as crawler nodes because the requests of these IPs were concentrated on a certain date, 29.7% of them were in the long-tail area Of the respondents.

5 CONCLUSION

In this paper, we introduce a node reducing method that identifies the IP set of distributed crawlers and gradually reduces IP by using the property that web traffic follows the power law. The node reducing scheme has shown a very low level of false positives against distributed crawlers using multiple IPs, effectively identifying crawler sets.

6 FUTURE WORKS

Web traffic generally tends to generate traffic bursts at certain times. [1] Although the experiment of this paper is based on actual traffic log, since the time point of the data used in the experiment is one month, it does not include cases where a new item is added or an issue occurs and a traffic burst occurs. In order to apply the results of this paper more securely to actual services, it is necessary to study whether the item movement level and threshold value of long-tail area can be maintained based on actual traffic data for traffic burst occurrence cases.

7 REFERENCES

[1] M.V Simkin and V.P. Roychowdhury, "A theory of web traffic" https://arxiv.org/pdf/0711.1235.pdf
[2] Density Estimation for Statistics and Data Analysis
[3] Explaining World Wide Web Traffic Self-Similarity
[4] Research on Detection Algorithm of WEB Crawler
[5] Design and Implementation of Scalable, Fully Distributed Web Crawler for a Web Search Engine
[6] Design and Implementation of a Distributed Crawler and Filtering Processor
[7] URL Assignment Algorithm of Crawler in Distributed System Based on Hash
[8] Design and Implementation of a High-Performance Distributed Web Crawler
[9] Feature evaluation for web crawler detection with data mining techniques
[10] Crawler Detection: A Bayesian Approach
[11] Real-time Web Crawler Detection [12] An investigation of web crawler behavior: characterization and metrics