FQDN-Based Whitelist Filter on a DNS Cache Server Against the DNS Water Torture Attack

Keita Hasegawa, Daishi Kondo, Hideki Tode
Department of Computer Science and Intelligent Systems, Osaka Prefecture University, Osaka, Japan
Email: {hasegawa.keita@com., daishi.kondo@, tode@}cs.osakafu-u.ac.jp

***Abstract*—A distributed denial-of-service (DDoS) attack is a major social issue, such as the Domain Name System (DNS) DDoS attack against Dyn Inc., a DNS provider, which caused serious outages to several web services in 2016. This paper tackles the DNS water torture attack that was observed in the Dyn cyberattack. In the DNS water torture attack, attackers create a large number of unique and unresolvable fully qualified domain names (FQDNs) with random labels and send them to DNS cache servers and authoritative DNS servers, causing these servers to fail. Although countermeasures on DNS cache servers have been proposed to prevent such attack, one drawback of these countermeasures is that they cannot detect malicious DNS queries generated by an advanced DNS water torture attack. To address this shortcoming, we propose an FQDN-based whitelist filter that registers actually existing FQDNs and drops the non-existent ones created by the attackers. This whitelist filter eliminates malicious DNS queries while mitigating the negative impact of falsely dropping legitimate ones.**

***Index Terms*—DNS DDoS, DNS water torture attack, FQDN-based whitelist filter**

I. Introduction

A distributed denial-of-service (DDoS) attack [1] is a type of cyberattack in which the attacker uses malware to turn a large number of computing devices into bots, force them to generate superfluous traffic to the targeted servers, and make them unavailable. In 2016, a Domain Name System (DNS) DDoS attack using the Mirai botnet [2] was launched against Dyn, Inc., a DNS provider that is utilized by many well-known sites such as Amazon, GitHub, and PayPal. The attack traffic reached 1.2 Tbps at its peak, causing these well-known sites to become inaccessible [3]. This paper tackles the DNS water torture attack that was observed in the Dyn cyberattack [4]–[6]. In this attack, bots generate a large number of unique and unresolvable fully qualified domain names (FQDNs) with random labels added to the targeted primary domain names and then send malicious DNS queries including these FQDNs to the targeted DNS cache servers. These massive queries invalidate the cache of the cache servers and also attack the authoritative DNS servers.

Conventional countermeasures against the DDoS attack include the following. A name filter [7], [8] detects the randomness of the FQDNs and filters out attack queries. The rate-limiting method [9] prevents an attack by dropping DNS queries without making a clear distinction between benign and malicious ones when a certain amount of DNS traffic intensity is detected. However, it is possible for these methods to not only overlook malicious queries generated by an advanced DNS water torture attack but also drop legitimate ones. Moreover, stale DNS data (i.e., the time to live (TTL) of the cache has expired) are utilized to continue to provide a name resolution service even when the authoritative DNS servers are down [10]. A side effect of this solution, however, is that, when a content provider such as a content delivery network (CDN) installs a server load balancing function by sequentially changing the IP addresses, the stale DNS data might deactivate this function.

As an alternative countermeasure against the DNS water torture attack, an FQDN-based whitelist filter on a DNS cache server is proposed in this paper. This filter registers only FQDNs that actually exist and are resolvable. Although a certain amount of time is needed to create a whitelist, the filter can eliminate attack queries in this way while mitigating the negative effect of falsely dropping legitimate ones. As this whitelist is based on actually existing FQDNs, and, hence, benign DNS queries are allowed through and DNS cache servers can update their DNS cache, which helps realize load balancing. In the study experiment, we created FQDN-based whitelist filters and evaluated their performance on three different datasets. The evaluation result showed that, aside from reducing the attack threat, our proposed whitelist filter can validate 97% of legitimate DNS queries. In addition, we recommend that a filter for each DNS cache server be created independently.

The remainder of this paper is organized as follows. Section II provides an overview of the DNS water torture attack and the conventional countermeasures against it. Section III introduces our FQDN-based whitelist filter. Section IV presents the result of our experiment evaluation. Section V discusses our findings. Finally, Section VI concludes this paper.

II. Background

A. The DNS Water Torture Attack

Fig. 1 shows an overview of the DNS water torture attack [4]–[6]. In this attack, the attacker first uses malware to infect vulnerable computing devices and make them bots. Then, the attacker sends an attack command to these bots to generate malicious DNS queries containing FQDNs with random labels (e.g., *(generated_label)*`.victim.com`) and to send them to the DNS cache servers. As these FQDNs are not maintained by the authoritative DNS server of the targeted `victim.com` domain, the corresponding DNS responses do not exist in the DNS cache servers. Therefore, the DNS cache servers will forward these attack queries to the `root` and `com`

978-3-903176-32-4 © 2021 IFIP

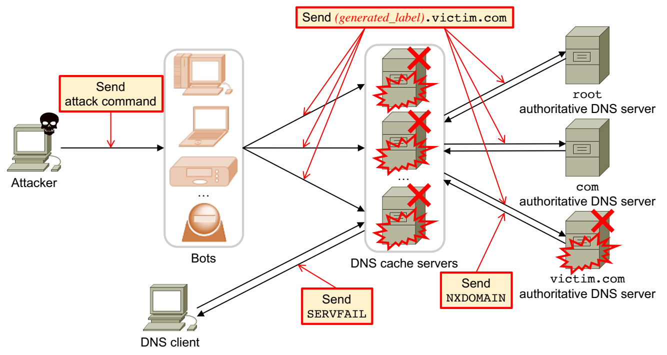

Send (generated_label).victim.com
Send attack command
Attacker
Bots
DNS cache servers
root authoritative DNS server
com authoritative DNS server
victim.com authoritative DNS server
Send SERVFAIL
Send NXDOMAIN
DNS client

Fig. 1: Overview of the DNS water attack.

authoritative DNS servers, and these queries will eventually reach the authoritative DNS server of `victim.com`. At this point, an `NXDOMAIN` message stating that the FQDN does not exist is sent to the DNS cache servers from the authoritative DNS server of the `victim.com` domain. This results in the generation of the negative cache and the deletion of the positive cache in the DNS cache servers, which degrades the performance of these servers [11]. When no cache server can resolve the FQDN because of the attack, a legitimate DNS client will receive a `SERVFAIL` message indicating that the server cannot respond properly for some reason. By increasing the number of exploited DNS cache servers, the bots can send more malicious queries to the targeted authoritative DNS server, without any of the DNS cache servers noticing that an attack is happening simply on the basis of the traffic volume.

B. Conventional Countermeasures

To minimize the negative impact of the DNS water torture attack, we propose a countermeasure for use in DNS cache servers.

1) Name Filter: A name filter [7], [8] detects the randomness of FQDNs on the basis of features such as the bigram of the leftmost label or the FQDN and drops any suspected attack queries. However, legitimate queries with the randomness of FQDNs such as disposable domain [12]–[15] might also be excluded. Moreover, any FQDNs that are not managed by the authoritative DNS server and do not have the randomness can be used to perform an attack, which a name filter cannot completely prevent. Thus, a name filter is not effective at dealing with such an advanced attack.

2) Rate Limiting: The rate-limiting method [9] is a method that is used to detect a large volume of traffic and to drop DNS queries on DNS cache servers to regulate excessive traffic volume. With this method, the volume of traffic to the targeted authoritative DNS server can be limited; however, some attack queries might still reach the target and, at the same time, legitimate queries might be dropped, which are some of the limitations of this method [16]. In addition, when attack queries are sent to multiple DNS cache servers, they can reach each of the servers at such a low rate that performing rate limiting is unnecessary. Thus, such an advanced attack invalidates the rate-limiting method.

3) Serve Stale: Serve stale [10] can continue to provide a name resolution service by utilizing an expired cache. Even if an attack occurs and the authoritative DNS server is down, the stale cache can mitigate the attack and continue to provide service [17]. However, when a content provider such as a CDN installs a server load balancing function by sequentially changing the cache, the stale cache might deactivate this function. For instance, Akamai uses a short TTL of 20 seconds to distribute the load [18].

III. Proposal

This paper proposes an FQDN-based whitelist filter for use in a DNS cache server against the DNS water torture attack. This filter eliminates malicious queries while reducing the negative impact of falsely excluding legitimate ones.

A. FQDN-Based Whitelist Filter

This filter checks the received DNS response in the DNS cache server, and when the response code (`RCODE`) [19] is `NOERROR`, the filter registers the FQDN of this DNS response as a list item. This ensures that the FQDNs registered in the whitelist actually exist. Because the malicious queries caused by the DNS water torture attack do not exist, they can be eliminated by the filter. As this whitelist is based on actually existing FQDNs, and, hence, benign DNS queries are allowed through and DNS cache servers can update their DNS cache, which helps realize load balancing. The filter proposed in this paper creates the whitelist by focusing on the `A` record, `AAAA` record, and `RCODE` message because the record type of most queries from DNS clients is either an `A` or an `AAAA` record [20].

When an authoritative DNS server under attack is identified, the whitelist filter is applied only to the FQDN whose primary domain name is maintained by the server. Thus, FQDNs other than the FQDNs including the targeted domain are allowed through even when the attack is active.

B. Methods to Activate the FQDN-Based Whitelist Filter

In addition to the rate-limiting method, the whitelist filter can be launched by some methods, and, here, we propose two methods. First, a direct signal from the targeted authoritative DNS server can activate the whitelist filter by informing the DNS cache servers that an attack on the former is happening. This method can launch the whitelist filter even if the attack query rate is low, unlike with the rate-limiting method. Second, the frequency of occurrence of each `RCODE` message can be used to activate the whitelist filter because the `RCODE` message of most DNS responses is `NOERROR`; however, a large amount of `NXDOMAIN` traffic is generated during an attack. By measuring the amount of `NXDOMAIN` traffic within a certain period, this method will be able to detect an attack and launch the whitelist filter. Owing to the space limitation of this paper, the actual evaluation is not discussed in this paper.

C. Problems with the FQDN-Based Whitelist Filter

When the proposed whitelist filter is activated during an attack, the attack will fail. However, legitimate users will not be able to resolve the FQDNs of the targeted domains that are not in the whitelist. Nevertheless, given the nature of the cache, it can be reused in the future to store data that have been used once; thus, it is highly possible that the cache of DNS responses is expected to be frequently accessed by users. *In other words, it is also highly possible that the FQDNs that users frequently resolve are registered in the whitelist.* Therefore, we hypothesize that legitimate users will not be significantly affected by the false dropping of legitimate DNS queries even during an attack.

IV. Experiments

We created whitelists using three different DNS traffic datasets: a dataset captured at our laboratory (Laboratory), an open dataset provided by IEEE (IEEE) [21], and a dataset captured at our university (University). We define the acceptance rate of FQDN and of the DNS queries as the percentage of the number of FQDNs accepted by the whitelist filter relative to that of all the FQDNs and as the percentage of the number of DNS queries accepted by the whitelist filter relative to that of all the DNS queries, respectively, and we calculated and evaluated these performance metrics.

A. Acceptance Rate

TABLE I: Acceptance rate

Dataset	FQDN	DNS query
Laboratory	62.5%	98.1%
IEEE	63.9%	98.9%
University	57.6%	95.1%
Average	61.3%	97.4%

Table I shows the acceptance rates of the FQDNs and DNS queries, which were computed by five-fold cross-validation. All the rates of the DNS queries were quite high, which indicates that, although the proposed method is naive, it exhibits sufficient filtering performance and does not have a large negative impact on legitimate users even during an attack. On the other hand, the acceptance rates of FQDNs were low. From the average of the acceptance rates of the FQDNs and DNS queries obtained from the three datasets, DNS queries with dropped FQDNs, which comprised 38.7% ($= 100 - 61.3$) of all FQDNs, accounted for only 2.6% ($= 100 - 97.4$) of all the DNS queries. This means that the dropped FQDNs generally have a small number of access operations and that FQDNs with a large number of access operations are registered in the whitelist. Thus, we validated our hypothesis as described in Section III-C.

The results in Table I are supported by Fig. 2. As for the definition of the FQDN ranking, an FQDN is ranked by its number of access operations. The ranking and the number of access operations follow Zipf's law, and the FQDN with a higher ranking is included in more DNS queries.

B. Difference Between the Datasets

Table II shows the acceptance rate of applying the test dataset to the whitelist, each of which was created by another dataset. Although the laboratory dataset is not a subset of the

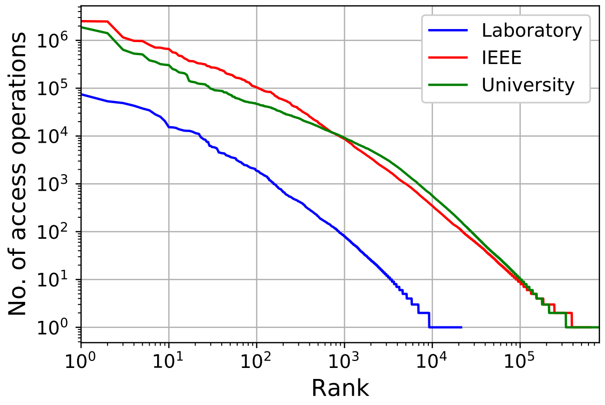


Fig. 2: FQDN ranking vs. the number of access operations.

TABLE II: Acceptance rate by applying the test dataset to the whitelist dataset created by another dataset

Test dataset	Whitelist dataset	FQDN	DNS query
Laboratory	IEEE	20.9%	64.5%
	University	59.5%	96.8%
IEEE	Laboratory	0.7%	53.3%
	University	5.4%	73.6%
University	Laboratory	0.1%	36.2%
	IEEE	0.3%	18.3%

university dataset, the acceptance rate of the DNS queries was relatively high. However, except for this case, the acceptance rates of the FQDNs and DNS queries were much lower than those shown in Table I. This indicates that each group of users has different access trends for the FQDNs. Therefore, the proposed whitelist should be created for each DNS cache server.

C. Number of Required Days to Create the Whitelist

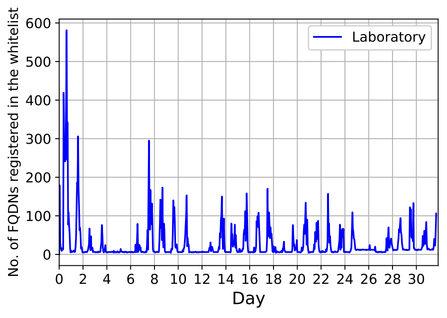
Fig. 3 shows the changes in the number of FQDNs added to the whitelist, which describes the number of days required to create the whitelist. As can be seen in Fig. 3(a)-(c), the number of registered FQDNs increased significantly on the first day and that of the added FQDNs did not increase significantly after the 8th, 3rd, and 16th day for the laboratory, IEEE, and university datasets, respectively. Moreover, the figure shows that the number of added FQDNs did not become zero at some point. Therefore, the present experimental environment concludes that the number of days required to create the whitelist is 16 days and that at least 1 day may be enough to make a half-baked but sufficient filter.

V. Discussion

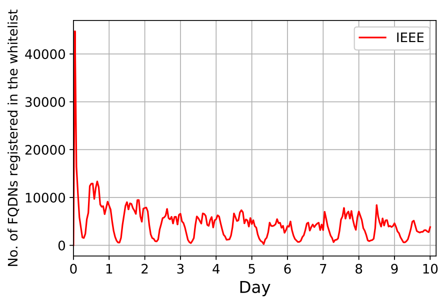
In this section, we treat the collection of dropped FQDN samples in each dataset as one dataset and discuss it.

A. Access Counts of the Dropped FQDNs

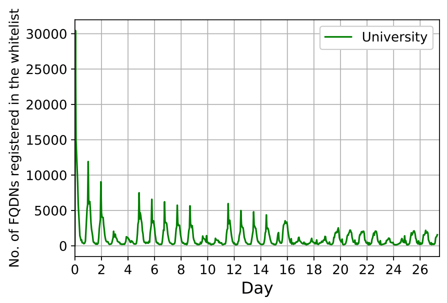
Fig. 4 shows the percentage of the dropped FQDNs for each access count. As can be seen in the figure, the dropped FQDNs generally have a smaller number of access operations. Even if such an FQDN is dropped, it usually has little effect on legitimate users. However, there are some FQDNs that have



(a) Laboratory.



(b) IEEE.



(c) University.

Fig. 3: Changes in the number of FQDNs registered in the whitelist.

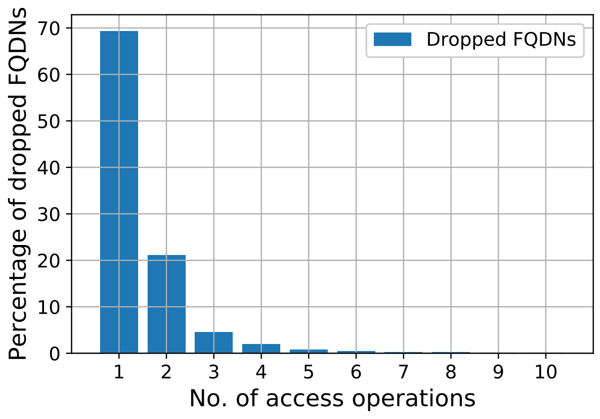


Fig. 4: Access counts of the dropped FQDNs.

a larger number of access operations when a whitelist filter is applied, and, therefore, these FQDNs should be allowed to pass. In addition, when the FQDN with a popular primary domain is accessed a few times but is dropped, the legitimate user might be affected.

B. Popular Dropped FQDNs

Using the Majestic dataset [22], which includes 1 million popular primary domains, we investigated whether the dropped FQDNs had popular domains. Table III shows the ratio of dropped FQDNs included in the top 1K, 10K, 100K, and 1M domains, and the average number of FQDNs per primary domain. Around half of the FQDNs had popular primary domains, and, therefore, the whitelist filter should allow such FQDNs to pass to reduce the negative impact on legitimate users when these FQDNs are falsely dropped. As for the top 1K primary domains, each primary domain has around 503 FQDNs on average, which indicates that the more popular primary domains hold more FQDNs.

C. Characteristics of the Dropped FQDNs

There are three types of dropped FQDNs: “FQDN that is not popular and is only accessed a few times,” “FQDN that is accessed many times within a certain period of time,” and “FQDN that has a popular disposable domain [12]–[15].” The

TABLE III: Percentage of dropped FQDNs included in the top n domains and average number of FQDNs per domain

Top n domains	Percentage of dropped FQDNs	Average no. of FQDNs
1K	44.9%	503.0
10K	49.6%	83.5
100K	56.3%	31.0
1M	63.7%	13.9

first one can be dropped without any problem as it is not normally accessed by users.

The second one is generated at a certain event. If an access operation on such an FQDN happens during an attack, this attack will have a negative impact on legitimate users. However, even when the filter is applied, it can register the FQDN when there is a certain number of access operations on the FQDN.

The third one is used for various purposes such as video services, social networking services, and security improvements. This FQDN is temporarily created by a DNS client for an authoritative DNS server to directly manage the user’s state, and it does not cause a cache hit. According to the previous study on disposable domains [12]–[15], a disposable domain is automatically created and is used only once. Therefore, because these features are similar to those of attack queries, it is very difficult to distinguish whether the DNS query is malicious.

VI. Conclusion

This paper tackled the DNS water torture attack, and to provide a countermeasure against this attack, it proposes an FQDN-based whitelist filter for a DNS cache server. By utilizing real-world DNS traffic datasets, we showed that the proposed method prevented all attack queries and minimized the negative effect of falsely dropping legitimate queries during an attack. In addition, we analyzed the FQDNs of the dropped legitimate queries and examined whether it was possible to reduce the negative impact on legitimate users. In the future, we plan to further investigate disposable domains to determine the difference between legitimate queries including these domains and the attacked ones, and to evaluate the malicious traffic reduction rate by introducing the proposed method.

References

[1] "Aws best practices for ddos resiliency." [Online]. Available: https://d0.awsstatic.com/whitepapers/Security/DDoS_White_Paper.pdf

[2] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the mirai botnet," in *USENIX Security*, 2017, pp. 1093–1110.

[3] "Dyn analysis summary of friday october 21 attack." [Online]. Available: https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/

[4] "Water torture: A slow drip dns ddos attack." [Online]. Available: https://secure64.com/2014/02/25/water-torture-slow-drip-dns-ddos-attack/

[5] X. Luo, L. Wang, Z. Xu, K. Chen, J. Yang, and T. Tian, "A large scale analysis of dns water torture attack," in *CSAI*, 2018, pp. 168–173.

[6] H. Griffioen and C. Doerr, "Taxonomy and adversarial strategies of random subdomain attacks," in *IFIP NTMS*, 2019, pp. 1–5.

[7] T. Yoshida, K. Kawakami, R. Kobayashi, M. Kato, M. Okada, and H. Kishimoto, "Detection and filtering system for dns water torture attacks relying only on domain name information," *Journal of Information Processing*, vol. 25, pp. 854–865, 2017.

[8] L. Chen, Y. Zhang, Q. Zhao, G. Geng, and Z. Yan, "Detection of dns ddos attacks with random forest algorithm on spark," in *BDNT*, vol. 134, 2018, pp. 310–315.

[9] "Recursive client rate limiting in bind 9.9.8, 9.10.3 and 9.11.0." [Online]. Available: https://kb.isc.org/docs/aa-01304

[10] "Rfc 8767 serving stale data to improve dns resiliency." [Online]. Available: https://tools.ietf.org/html/rfc8767

[11] L. Shafir, Y. Afek, A. Bremler-Barr, N. Peleg, and M. Sabag, "Dns negative caching in the wild," in *ACM SIGCOMM Posters and Demos*, 2019, pp. 143–145.

[12] Y. Chen, M. Antonakakis, R. Perdisci, Y. Nadji, D. Dagon, and W. Lee, "Dns noise: Measuring the pervasiveness of disposable domains in modern dns traffic," in *IEEE/IFIP DSN*, 2014, pp. 598–609.

[13] S. Hao and H. Wang, "Exploring domain name based features on the effectiveness of dns caching," *SIGCOMM Comput. Commun. Rev.*, vol. 47, no. 1, pp. 36–42, Jan. 2017.

[14] R. Laurens, H. Rezaeighaleh, C. C. Zou, and J. Jusak, "Using disposable domain names to detect online card transaction fraud," in *IEEE ICC*, 2019, pp. 1–7.

[15] Y. Zeng, Y. Zhang, T. Zang, X. Chen, and Y. Wang, "A linguistics-based stacking approach to disposable domains detection," in *IEEE ICNP*, 2019, pp. 1–4.

[16] J. Mirkovic and P. Reiher, "A taxonomy of ddos attack and ddos defense mechanisms," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, pp. 39–53, Apr. 2004.

[17] G. C. M. Moura, J. Heidemann, M. Müller, R. de O. Schmidt, and M. Davids, "When the dike breaks: Dissecting dns defenses during ddos," in *ACM IMC*, 2018, pp. 8–21.

[18] K. Fujiwara, A. Sato, and K. Yoshida, "Dns traffic analysis – cdn and the world ipv6 launch," *Journal of Information Processing*, vol. 21, no. 3, pp. 517–526, 2013.

[19] "Rfc 2929 domain name system (dns) iana considerations." [Online]. Available: https://tools.ietf.org/html/rfc2929

[20] N. Ishikura, D. Kondo, I. Iordanov, V. Vassiliades, and H. Tode, "Cache-property-aware features for dns tunneling detection," in *ICIN*, 2020, pp. 216–220.

[21] M. Singh, M. Singh, and S. Kaur, "Ti-2016 dns dataset," 2019. [Online]. Available: https://dx.doi.org/10.21227/9ync-vv09

[22] "The majestic million." [Online]. Available: https://majestic.com/reports/majestic-million