Preventing DNS Amplification Attacks Using the History of DNS Queries with SDN

Soyoung Kim, Sora Lee, Geumhwan Cho, Muhammad Ejaz Ahmed, Jaehoon (Paul) Jeong, and Hyoungshick Kim(✉)

Sungkyunkwan University, Suwon, South Korea
{ksy2608,leesora,geumhwan,ejaz629,pauljeong,hyoung}@skku.edu

Abstract. Domain Name System (DNS) amplification attack is a sophisticated Distributed Denial of Service (DDoS) attack by sending a huge volume of DNS name lookup requests to open DNS servers with the source address spoofed as a victim host. However, from the point of view of an individual network resource such as DNS server and switch, it is not easy to mitigate such attacks because a distributed attack could be performed with multiple DNS servers and/or switches. To overcome this limitation, we propose a novel security framework using Software-Defined Networking (SDN) to store the history of DNS queries as an evidence to distinguish normal DNS responses from attack packets. Our evaluation results demonstrate that the network traffic for DNS amplification attack can completely be blocked under various network conditions without incurring a significant communication overhead.

Keywords: Software-Defined Networking (SDN) · Distributed Denial of Service (DDoS) · Domain Name System (DNS) · DNS amplification attack

1 Introduction

Domain Name System (DNS) amplification attack is a popular form of Distributed Denial of Service (DDoS) attack that relies on the use of publicly accessible open DNS servers to overwhelm a victim system with DNS response traffic [22]. In a typical DNS amplification attack scenario, an attacker uses an extension to the DNS protocol [25] to generate relatively small queries (e.g., about 60 bytes) with a spoofed source address (i.e., the victim's address) to DNS servers. As a result, DNS servers reply with significantly larger responses (e.g., about 4,000 bytes) to exhaust the victim's resources (see Sect. 2.1 for more details). DNS amplification is one of the most notorious and disruptive attack types. In March 2013, a massive 300 Gbps DDoS attack was thrown against the website of Spamhaus which is the anti-spam clearing house organization [9].

However, it is not a trivial task to prevent such attacks since trusted sources (i.e., open DNS servers) are used as the sources of a DDoS attack. Blacklisting the attack machines' IP addresses can also affect legitimate DNS resolutions. To

S.N. Foley et al. (Eds.): ESORICS 2017, Part II, LNCS 10493, pp. 135–152, 2017.
DOI: 10.1007/978-3-319-66399-9_8

make matters worse, DNS requests can easily be spoofed since DNS protocols are based on UDP [3]. Therefore, a proper defense mechanism is needed to mitigate such attacks while minimizing its potential impact on legitimate users.

In recent years, a couple of defense techniques have been developed to mitigate such attacks. Vixie [26] proposed a defense mechanism based on limiting the number of unique responses from a DNS server. However, this defense mechanism could be circumvented by distributing attack packets among a large number of DNS servers.

Another possible strategy is to use the history of DNS queries for checking the "one-to-one mapping" between DNS requests and responses in order to detect orphan DNS responses [8,11,21]. In general, the existing solutions can be categorized into two approaches: (1) using the local memory of switches [8,21] and (2) using the external memory of a remote server [11]. Each approach has its strengths and weaknesses and may not be suitable for certain circumstances. For the first approach (e.g., [8,21]), it is critical to efficiently store the DNS queries because a switch typically has a small memory size. Therefore, the solutions in this category used a space-efficient data structure called Bloom filters to efficiently store this history of DNS queries because a Bloom filter supports probabilistic set membership testing. However, the use of Bloom filters inherently gives erroneous results (i.e., false positives). For the second approach (e.g., [11]), the communication with a remote server is always required to store all DNS query records and check them, which results in a significant communication overhead. In this paper, we proposed a more flexible model by providing a highly robust and scalable data storage for DNS queries using Software-Defined Networking (SDN) [13], which was recently introduced to decouple the data and control planes in network systems. The proposed scheme is designed to store all DNS query records by using an SDN controller even when there is no enough memory to store DNS query records in a switch anymore. Surely, the proposed scheme does not cover all types of DDoS attacks. We focused only on DNS amplification attacks. Our main contributions are as follows:

- We propose a novel mitigation system to fight against DNS amplification attacks by checking the validity of DNS response packets using the history of DNS queries with SDN. The proposed scheme does not need anymore to use a probabilistic method such as Bloom filters and can finally avoid false positives related to DNS amplification attacks (See Sect. 3).
- We show the feasibility of the proposed system by conducting intensive experiments in a controlled environment. Our evaluation results demonstrate that the network traffic for DNS amplification attack can completely be blocked under various network conditions without incurring a significant delay by the communication with the SDN controller (See Sect. 4).

The rest of the paper is organized as follows: Sect. 2 provides the background information about DNS amplification attack and SDN used in the proposed system. Section 3 presents our proposed architecture with the important network components, and our experiment results are presented in Sect. 4. Related work is covered in Sect. 5. Our conclusions and future work are in Sect. 6.

2 Background

In this section, we first explain how a DNS amplification attack can be performed and then provide an overview of SDN usage for the additional storage of the history of a DNS request, when those requests cannot be stored in local switches due to limited memory capacity.

2.1 DNS Amplification Attack

A DNS amplification attack relies on the use of publicly accessible open DNS servers to overwhelm a victim's network bandwidth with DNS response traffic. A DNS server provides the corresponding IP address against the domain name requested by a user. For example, when a user wants to connect to a website, they usually type its domain name (i.e., URL) in the browser. The local DNS server, when receives the domain name request, tries to find the corresponding IP address against the user's request which is then communicated to the user, and that IP address is used to connect to the website. Here, a DNS response packet, delivering the corresponding IP address to the user, is of a much larger size than the user's request. This principle makes a DDoS attack more influential.

Figure 1 shows an overview of the DNS amplification attack. An `Attacker` sends a request using small DNS query with a spoofed IP address (`Victim`'s IP address) to `Open DNS server`. Then `Open DNS server` returns a response to the `Victim` with several times larger packets than the DNS request. Preventing this attack is difficult since DNS requests from the attacker include the spoofed IP address ("`10.0.0.1`"), and DNS responses are sent to the victim with "`10.0.0.1`" as its IP address instead of the original requester. Unlike an ordinary DDoS attack, it is not simple to prevent this form of attacks since trusted sources (i.e., DNS servers) are used for DDoS attack; blocking attack machines' IP addresses might affect and damage normal network operations. To

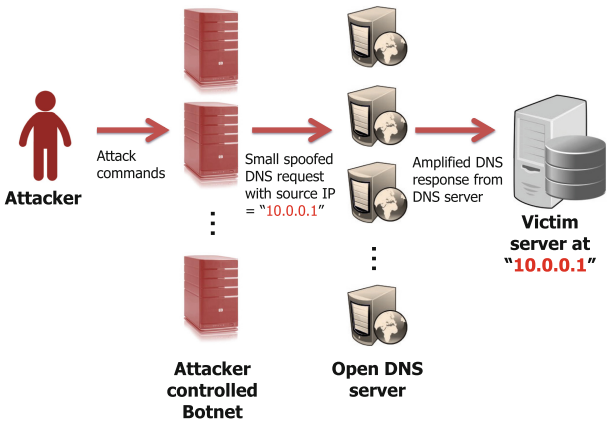


Fig. 1. Overview of DNS amplification attack.

make matters worse, DNS requests can easily be spoofed since DNS protocols are based on UDP. A proper defense mechanism should mitigate attacks, while avoiding a major impact on legitimate users.

In this paper, we consider two types of DNS packets ("`A` type of packet" and "`ANY` type of packet"). Normally, when a user requests DNS query, "`A` type of packet" is used for requesting a query and the response packet is typically less than 512 bytes that contains only an IP address which the user wants to find. Otherwise, if an attacker uses `ANY` type of packet as a query (about 60 bytes) of the request, it can return a response of about 4,000 bytes, resulting in about 50x amplification [24]. Therefore, the use of `ANY` type of packet is more effective for the DNS amplification attack.

2.2 Overview of SDN

SDN is a novel networking paradigm that decouples the control plane from the data plane. This separation can be realized by a well-defined programming interface between a switch and an SDN controller [15]. The SDN controller enforces direct control over the data plane elements (e.g., switch) with network applications using an OpenFlow protocol [2], as shown in Fig. 2. The OpenFlow switch performs forwarding functions which allows user space control at flow level processing on the network [1]. The SDN controller can manage and control the OpenFlow switch since the OpenFlow switch forwards packets according to the predefined rules in its flow table received from the SDN controller [16]. For example, a forwarding component in the OpenFlow switch consults the flow table for a proper rule to forward the incoming packet. If the flow rule is found, the component forwards the packet according to the rule. Otherwise, the OpenFlow switch asks the SDN controller, and then a new flow rule is enforced to the OpenFlow switch by the SDN controller. Also, network operators can deploy

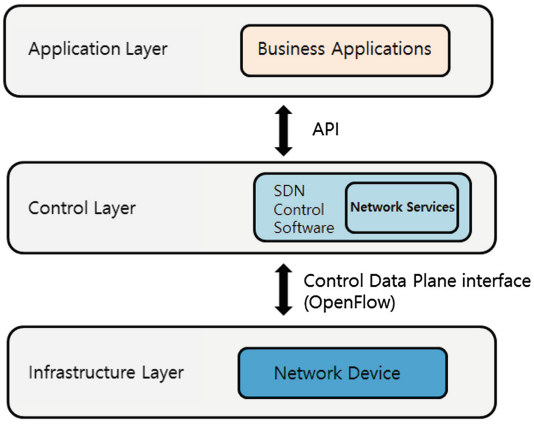


Fig. 2. Overview of SDN architecture.

applications of their choice without manually deploying and excessively configuring the networks. For example, with the use of SDN, network operators can easily implement a firewall application and deploy it at the application layer. Besides, the SDN controller enforces the partial functionality of the firewall in the switch. In this case, the cost of deploying new network resources can be avoided which results in a cost effective solution.

3 Proposed System

In this section, we explain how the proposed system can be used to block the packets for a DNS amplification attack.

3.1 Overview

We propose a new DNS amplification attack mitigation scheme using an "one-to-one strict mapping" method between DNS requests and responses in order to detect orphan DNS responses. Our proposed scheme can detect such responses by checking whether there exists a DNS request (generated by a benign host) that matches to a given DNS response. Consequently, a DNS response, which matches a DNS query requested by the victim, is only allowed to reach the victim's machine.

Figure 3 gives an overview of the proposed scheme. To check "one-to-one strict mapping" between DNS requests and responses, the DNS requests generated by benign hosts are first stored in the local memory of a switch. If the switch has no available memory space anymore, further DNS requests are stored in the memory of an external network entity (e.g., the SDN controller or another remote server).

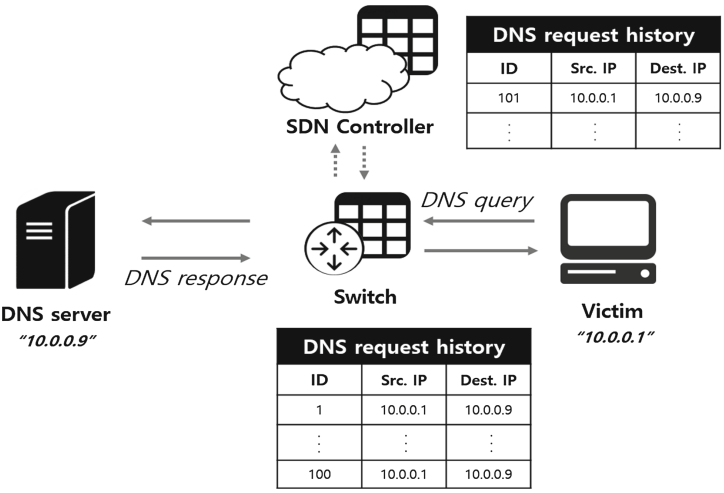


Fig. 3. Logical structure of the proposed scheme.

3.2 Main Components

The proposed system consists of two main components: switch and SDN controller. Here, we describe the primary function of each component to mitigate DNS amplification attacks.

Switch. In the proposed system, we assume that a switch is SDN compatible and has a limited memory capacity. Similar to previous proposals [8,21], the local memory of a switch is used to store the information about DNS request records (e.g., the source and destination IP addresses in a DNS request message). In SDN, unlike traditional network switches, a switch can forward packets based on the rules configured by an SDN controller [14]. We note that each switch constitutes the first line of defense against DNS amplification attacks. When a switch receives a packet, the high-level behaviors of the switch are as follows (see Fig. 4):

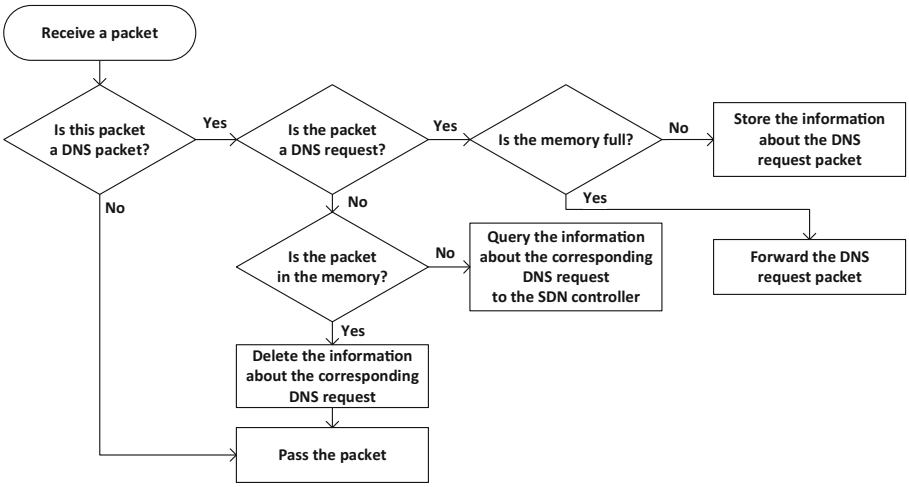


Fig. 4. Flowchart for a switch when receiving a packet.

1. The switch first checks whether the received packet is related to a DNS query. If the packet is not related to DNS, the switch just passes the packet; otherwise, the switch identifies the packet type.
2. If the received packet is "DNS request", the switch checks whether the DNS request information can be stored into the local memory of the switch itself. If the switch has a sufficient memory, the record is stored into its local memory; otherwise, the switch forwards the DNS request information to its predesignated SDN controller so that the DNS request information should be stored in the memory of the SDN controller (or another external host with a sufficiently large memory for storing DNS request records).

3. If the received packet is "DNS response", the switch verifies the validity of the received packet by checking whether there exists the DNS request record exactly matched to the DNS response in its local memory. If the validity of the DNS response is successfully verified, the received packet is passed and then the stored DNS request record is finally deleted because it is already consumed; otherwise, the next step can be redivided into two cases. In First, when its local memory is not full, the packet is simply dropped. Second, when its local memory is full of DNS request records, the switch forwards the DNS response packet to its SDN controller for checking the validity of the DNS response via the SDN controller.

SDN Controller. An SDN controller is a logically centralized network entity that manages flows to enable more flexible, customized, and intelligent networking. Naturally, when there is no enough memory to store DNS query records in a switch anymore, the SDN controller can establish or modify the rules to dynamically forward such information from the switch to an external database server with a large memory capacity needed to store it. Without loss of generality, we assume that an SDN controller itself has a large memory capacity to store the information about DNS request records. That is, the SDN controller can be used as an external storage server to store the history of DNS queries and provide it (if needed). When the SDN controller receives a packet, the high-level behaviors of the SDN controller are as follows (see Fig. 5):

1. The SDN controller first checks whether the received packet is related to a DNS query. If the packet is not related to DNS, the SDN controller processes the packet as normal; otherwise, the SDN controller identifies the packet type.

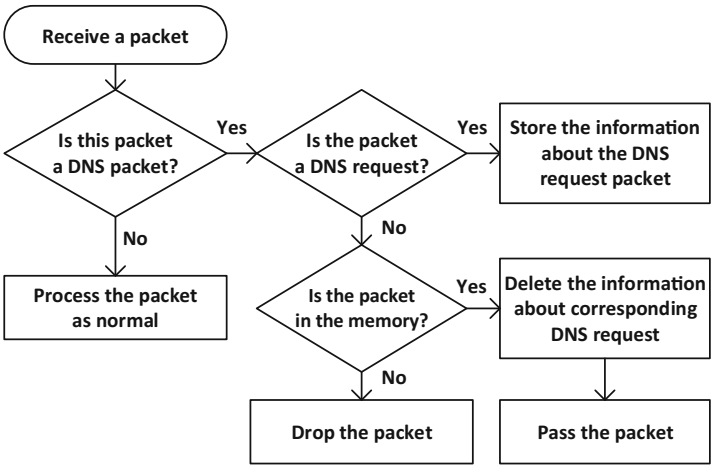


Fig. 5. Flowchart for an SDN controller when receiving a packet.

2. If the received packet is "DNS request", the SDN controller stores the DNS request information into the memory of the SDN controller.
3. If the received packet is "DNS response", the SDN controller verifies the validity of the DNS response message by checking whether there exists the DNS request record exactly matched to the DNS response in its memory. If the validity of the DNS response is successfully verified, the received packet is passed and then the stored DNS request record is finally deleted because it is already consumed; otherwise, the packet is just dropped because it means that the received DNS response is an orphan DNS response which could be used for a DNS amplification attack.

4 Evaluation

This section presents the evaluation results of the proposed scheme against DNS amplification attacks. We conducted experiments with various parameters such as the bandwidth of network link and the number of DNS servers used for DNS amplification attacks.

For evaluation, we measured several metrics such as the ratio of successfully delivered packets and the packet delivery time between normal hosts while performing DNS amplification attacks.

4.1 Experiment Setup and Procedure

To evaluate the performance of the proposed mitigation scheme against DNS amplification attacks, we used Mininet 2.3.0d1 (http://mininet.org/) because we need to conduct large-scale network experiments with varying the important parameters such as the number of DNS servers and bandwidth. Mininet works on Ubuntu 12.04.3 with VMware Workstation 12.5.4. Since Mininet provides Open vSwitch 1.10.0 and POX controller 0.2.0, we used them to implement the proposed system. As described in Sect. 3, we also implemented the storage of DNS query records at the Open vSwitch and the POX controller, respectively. In our experimental environment, the network topology consists of a POX controller, an Open vSwitch connected to two end hosts (victim/benign host), eight DNS servers, and one attack host (See Fig. 6).

In our experiments, we performed DNS amplification attacks on the constructed Mininet network topology. We considered two different types of DNS requests: `A` type with about 400–500 bytes of DNS responses and `ANY` type with about 3,000–4,000 bytes of DNS responses [23]. The number of attack packets generated from a single DNS server is 10,000 per second. While conducting the attack, the victim sends and receives a 64-byte packet per second to/from a benign host for simulating an exemplary networking scenario. For each test, we repeated a DNS amplification attack 10 times and computed the average metric to reduce the bias of the test result.

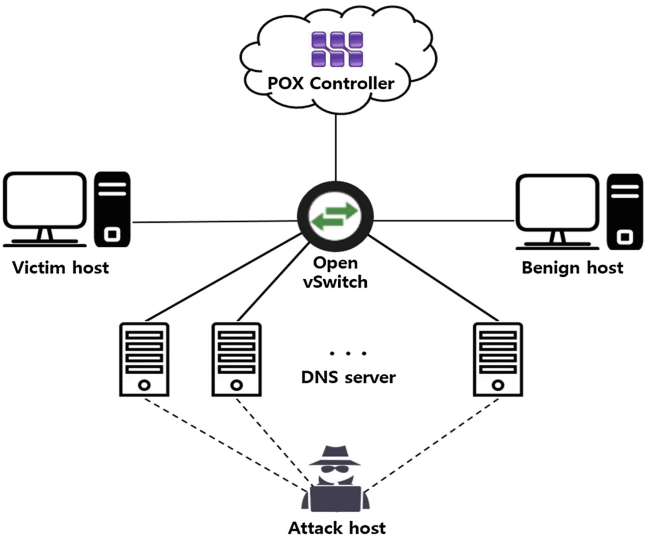


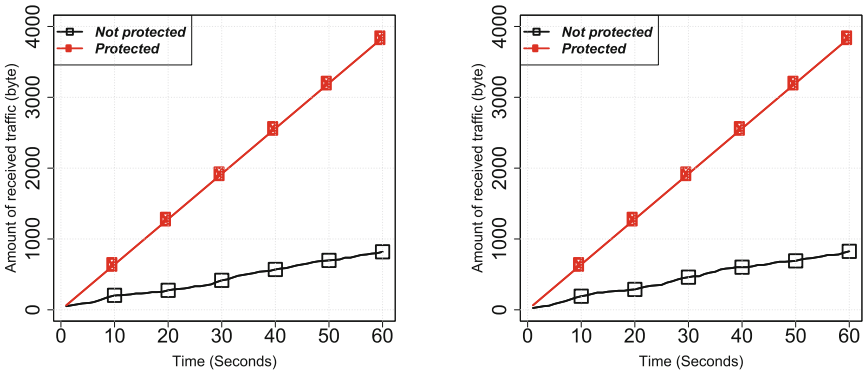
Fig. 6. Mininet network topology for experiments.

4.2 Experiment Results

Effectiveness of the Mitigation Scheme. To evaluate the performance of the proposed scheme, we first analyzed the cumulative amount of the successfully delivered traffic from the benign host to the victim host during 60 s when a DNS amplification attack takes place. We compared this metric with (`Protected`) and without (`Not protected`) the proposed mitigation scheme. For the comparison of `A` type and `ANY` type, the victim host's bandwidth was set to 5 Mbps since the attack with `A` type did not work when that bandwidth was more than 5 Mbps. Also, eight DNS servers were used to perform DNS amplification attacks. The experiment results are shown in Fig. 7.

For both `A` and `ANY` types, we can see that the cumulative amount of the successfully delivered traffic from the benign host to the victim host increased linearly over time when the proposed scheme was applied (`Protected`) while that amount tends to increase relatively slow without the proposed scheme (`Not protected`). In "`Not protected`", both types received were around 21% of packets; 819 bytes out of 3,840 bytes for `A` type and 826 bytes out of 3,840 bytes for `ANY` type, respectively, even when 60 s elapsed. In "`Protected`", however, the victim host successfully received 100% of the packets from the benign host under the same DNS amplification attack.

Effects of the Victim Host's Bandwidth. We discuss how the performance of the proposed mitigation scheme may change with the victim host's bandwidth. To reduce the impact of DNS amplification attacks, a possible straightforward approach is to increase the victim host's bandwidth. We analyzed how the packet

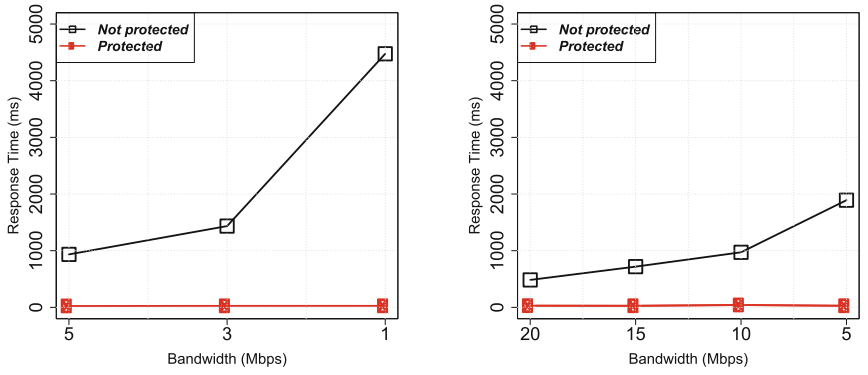


(a) Received traffic for A type. (b) Received traffic for ANY type.

Fig. 7. Amount of received traffic over time.

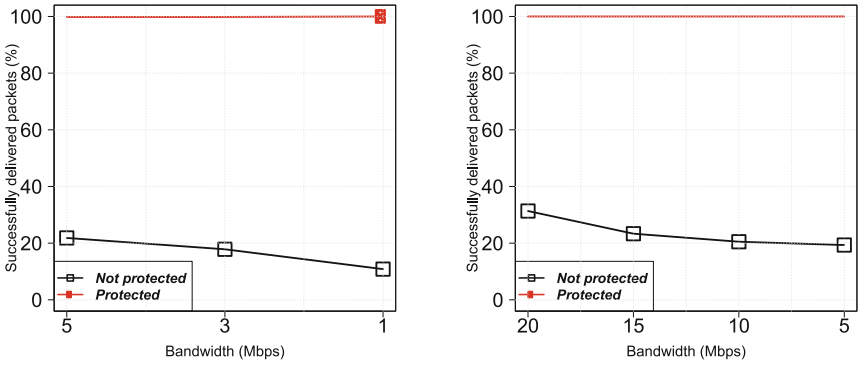
delivery time and the ratio of successfully delivered packets were influenced with the victim host's bandwidth.

For an improved analysis of the effects of victim host's bandwidth, we used different experiment parameters for each of `A` and `ANY` types since both types of DNS responses have different scales. We used 1 Mbps, 3 Mbps, and 5 Mbps, respectively, as the victim host's bandwidth for `A` type DNS amplification attacks. We also used 5 Mbps, 10 Mbps, 15 Mbps and 20 Mbps, respectively, as the victim host's bandwidth for `ANY` type DNS amplification attacks. Also, eight DNS servers were used to perform DNS amplification attacks. The experiment results are shown in Figs. 8 and 9.



(a) Delivery time for A type. (b) Delivery time for ANY type.

Fig. 8. Delivery time with bandwidth.



(a) Delivery ratio for A type. (b) Delivery ratio for ANY type.

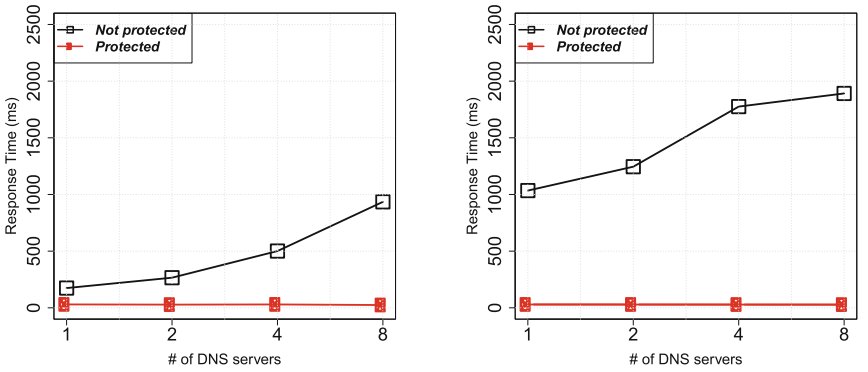
Fig. 9. Percentage of successfully delivered packets with bandwidth.

Figure 8 shows how the packet delivery time from the benign host to the victim host changed with the victim host's bandwidth. For both `A` and `ANY` types, the packet delivery time in "`Protected`" is always significantly faster than that in "`Not protected`". As the victim host's bandwidth decreased, in "`Protected`", the packet delivery time is still less than 32 ms on average for both types and remains stable regardless of the victim host's bandwidth while, in "`Not protected`", the packet delivery time greatly increased from 935 to 4,475 ms for `A` type and from 484 to 1,891 ms for `ANY` type, respectively.

Figure 9 shows how the ratio for successfully delivered packets sent from the benign host to the victim host changed with the victim host's bandwidth. In "`Protected`", the victim host successfully received all packets sent from the benign host. However, in "`Not protected`", for both types of DNS responses, the delivery ratio rather increased with the victim host's bandwidth. When the victim host's bandwidth greatly increased, the packet delivery ratios of `A` type and `ANY` type are only 21.83% and 31.33%, respectively.

Effects of the Number of DNS Servers. We now move to the discussion on the performance of the proposed mitigation scheme when the number of DNS servers increased by fixing the victim host's bandwidth as 5 Mbps. It is important to show that the proposed mitigation scheme is highly robust even when many open DNS servers are used for a DNS amplification attack. We analyzed how the packet delivery time and the ratio of successfully delivered packets were influenced with varying the number of DNS servers from 1 to 8.

Figure 10 shows how the packet delivery time from the benign host to the victim host changed with the number of DNS servers. For both `A` and `ANY` types, the packet delivery time in "`Protected`" is always significantly faster than that in "`Not protected`". As the number of DNS servers increased to 8, in "`Protected`", the packet delivery time is still less than 29 ms on average for both types and remains stable regardless of the number of DNS servers while,

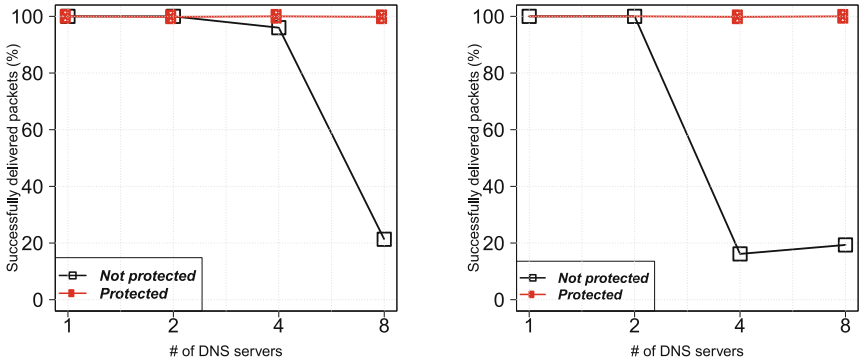


(a) Packet delivery time for A type. (b) Packet delivery time for ANY type.

Fig. 10. Packet delivery time with the number of DNS servers.

in "`Not protected`", the packet delivery time continuously increased from 174 to 935 ms for `A` type and from 1,034 to 1,891 ms for `ANY` type, respectively.

Figure 11 shows how the ratio for successfully delivered packets sent from the benign host to the victim host changed with the number of DNS servers. In "`Protected`", the victim host successfully received all packets sent from the benign host. In "`Not protected`", however, the packet delivery ratio was greatly decreased when the number of DNS servers is 8 for `A` type; the packet delivery ratio was greatly decreased from 4 to 8 DNS servers for `ANY` type.



(a) Delivery ratio for A type. (b) Delivery ratio for ANY type.

Fig. 11. Percentage of successfully delivered packets to the victim from benign host by increasing number of DNS servers, with 5 Mbps of bandwidth.

4.3 Comparison with the Existing Solutions

In this section, we discuss the advantages and disadvantages of the proposed mitigation scheme compared with the existing solutions [8,21] that also used the history of DNS records. Unlike our proposal, those solutions were designed to focus on the construction of efficient storage for DNS query records by using Bloom filters [5]. Bloom filter is a space-efficient probabilistic data structure that uses a one-way hash function. Unfortunately, Bloom filters inherently give false positives when the memory size in a switch is small to cover DNS queries to be stored. The proposed scheme was designed to use external servers' storage when there is no enough memory to store DNS query records in a switch. Therefore, there is no false positive in the proposed scheme since the information about all DNS requests is eventually stored. However, the proposed scheme may incur the communication overhead required to communicate with the external server (i.e., SDN controller). We analyzed how the false positive ratio of the previous solution using a Bloom filter and the communication overhead of the proposed scheme were affected with the available memory size of switch and the communication speed for SDN controller.

Given time T in seconds to keep the information of a DNS request in a Bloom filter, the maximum DNS request packet rate R (packets per second called pps) of a link, and the size M (bits) of the memory required to store DNS request records, false positive P can be calculated by the following formula [8,21]:

$$P = e^{(-1)\cdot(M/2TR)\cdot(\ln 2)^2}$$

Table 1 shows the false positive ratio and the number of error packets in the existing solutions with the available memory size of a switch. We assumed T was 3 s according to previous work [8]. We also assumed that the most widely used network switches (Cisco 350 series, Cisco 500 series, and Cisco 350X series) were deployed, respectively, for the proposed system. Here, the maximum available memory size of each switch can be calculated by excluding the size of the firmware from the overall memory size. The available memory sizes of Cisco 350

Table 1. False positive ratio and number of error packets in the existing solutions using a Bloom filter with the size of available memory in a switch.

Attack rate (pps)	False positive ratio			Number of error packets		
	Cisco 350 series (5.16 MB)	Cisco 500 series (21.94 MB)	Cisco 350X series (230.87 MB)	Cisco 350 series (5.16 MB)	Cisco 500 series (21.94 MB)	Cisco 350X series (230.87 MB)
1,000,000	3%	0%	0%	31,249	0	0
5,000,000	50%	5%	0%	2,499,983	262,205	0
10,000,000	71%	23%	0%	7,071,044	2,289,998	2
15,000,000	79%	37%	0%	11,905,481	5,614,546	485
20,000,000	84%	48%	0%	16,817,900	9,570,784	8,581

series, Cisco 500 series, and Cisco 350X switches are 5.16 MB, 21.94 MB, and 230.87 MB, respectively.

From Table 1, we can see that a large number of false positives occurs when either the memory size is about 5 MB or the attack rate is significantly higher (e.g., 10,000,000 pps). In such environments, existing solutions would not be effective in filtering out attack packets. In contrast to previous approach, as mentioned above, false positives do not appear in the proposed scheme because all DNS request records are stored.

However, the proposed mitigation scheme could incur some communication delay between SDN controller a switch when the memory of a switch is full. We analyzed this overhead with the memory size of a switch. We assumed the communication delay between SDN controller and a switch was 0.0416 ms according to the SDN controller benchmarks in [27]. Table 2 shows the average communication delay per packet between the SDN controller and a switch.

Table 2. Average communication delay for SDN controller and number of requests to be forwarded in the proposed scheme with the size of available memory in a switch.

Attack rate (pps)	Communication delay			Number of requests to SDN controller		
	Cisco 350 series (5.16 MB)	Cisco 500 series (21.94 MB)	Cisco 350X series (230.87 MB)	Cisco 350 series (5.16 MB)	Cisco 500 series (21.94 MB)	Cisco 350X series (230.87 MB)
1,000,000	0.019 ms	0 ms	0 ms	458,984	0	0
5,000,000	0.037 ms	0.022 ms	0 ms	4,458,984	2,698,993	0
10,000,000	0.039 ms	0.032 ms	0 ms	9,458,984	7,698,993	0
15,000,000	0.040 ms	0.035 ms	0 ms	14,458,984	12,698,993	0
20,000,000	0.040 ms	0.037 ms	0 ms	19,458,984	17,698,993	0

From Table 2, we can see that the proposed scheme may incur some communication delay; Cisco 500 series switches take a delay of 0.032 ms on average when the attack rate is 10,000,000 pps. Unsurprisingly, the worst case communication overhead (0.040 ms) can be found when the attack rate is 20,000,000 with Cisco 350 series switches. Those numerical analysis results demonstrate that the communication time overhead incurred by the proposed scheme seems acceptable enough in practice.

5 Related Work

DNS amplification attack is more harmful than other DDoS attacks due to the fact that the attack packets are sent by DNS servers cannot simply be blacklisted because DNS servers are trustworthy network entities. After Vaughn and Evron [23] reported their preliminary results on DNS amplification attacks, a number of countermeasures have been proposed.

The most intuitive approach is to prevent attackers from using spoofed IP addresses. Senie and Ferguson [19] and Katsurai et al. [12] proposed packet filtering methods in which edge routers block packets delivered from invalid sources to a local network. Snoeren et al. [20] proposed a traceback method to find the origins of IP packets. Bremler-Barr and Levy [6] proposed an authentication method to check the authenticity of the source addresses for incoming packets. However, those solutions generally require significant changes to the existing Internet infrastructure which are unlikely to be implemented in the near future. Unlike those solutions, DNS guard [10] did not require such fundamental changes in the Internet infrastructure because it tried to detect spoofed DNS requests using cookies. However, this approach requires the deployment of additional new servers between hosts and root servers, which is also not acceptable for some environments. We can see that the real world Internet is still vulnerable to several types of IP spoofing attacks [4].

A general approach to preventing DDoS attacks is to limit the number of packets delivered from a particular host. For DNS amplification attacks, Vixie [26] particularly proposed a mechanism called response rate limiting to limit the rate of responses from the DNS server and dropping the responses that exceeds the rate limit. However, such solutions are still susceptible to the use of a large number of open DNS servers. Verma et al. [24] proposed a distributed architecture with multiple DNS servers to detect a DNS amplification attack by accumulating the DNS request rates of those DNS servers that are involved in the DNS amplification attack. It needs to deploy the detection system on DNS servers and adjust complicated protocol to share the rate of DNS request between DNS servers.

Another approach is to detect the attacks by analyzing the DNS traffic. Deshpande et al. [7] and Rastegari et al. [18] proposed defense mechanisms using neural networks and a probabilistic model with several traffic statistics, respectively. Lexis and Mekking [17] proposed a visualization method to identify patterns in DNS traffic. Such approaches might be effective in detecting and classifying the attack traffics at the expense of false positives which restrict legitimate users from using DNS servers.

Recently, a promising technique was introduced by using the history of DNS queries to identify orphan DNS responses. Kambourakis et al. [11] proposed a method to check the "one-to-one mapping" relationship between DNS requests and responses. In their proposal, a mapping relationship was stored on an external database server. Consequently, the external database server was always used for storing every DNS request and checking every DNS response, respectively, at which its communication cost may not be acceptable in real world applications. To overcome this limitation, Sun et al. [21] proposed a technique using two Bloom filters in order to store the "one-to-one mapping" relationship between DNS requests and responses in the local memory of a switch. Bloom filters were used to support probabilistic membership queries with a small memory space. Di Paola and Lombardo [8] also proposed a similar technique using Bloom filters with a slight modification of the detection process [5]. Such mitigation tech-

niques and our proposed scheme share a common goal of blocking unmatched DNS responses. Nevertheless, the proposed scheme differentiates itself from other techniques by using a novel network model called SDN that can be used to store the history of DNS request at any network entity in a flexible manner. The proposed scheme aims to avoid the possibility of false-positives that are inherently incurred in previously proposed systems [8,21] when the number of DNS request increases because a network switch has limited memory space for storing all DNS requests. We propose a hybrid approach that takes the advantages of both approaches in order to support a "one-to-one strict mapping" method and simultaneously minimize the communication overhead with an external network entity such as the database server.

6 Conclusion

DNS amplification attack is a reflection-based DDoS attack. Since trusted severs such as open DNS servers are used as sources of attacks, it is not easy to stop such attacks. Previous defense mechanisms are not effective enough under the resource constrained switches having small memory sizes because they could incur false positives that cannot easily be ignored. In this paper, we propose a novel mitigation scheme against DNS amplification attacks by providing a highly scalable and centralized data storage for DNS request using SDN. Unlike the existing solutions using a probabilistic method, the proposed solution can remove the possibility of false positive packets, thus it can completely prevent DNS amplification attacks without incurring a significant delay by the communication with the SDN controller.

As part of future work, we plan to analyze the overhead for memory lookup procedure for the proposed system in addition to its communication delay. We also intend to conduct real world experiments through the deployment of the proposed system at a university network, and analyze its performance in a real-world setting.

Acknowledgment. This work was supported in part by the MSIP/IITP (No. 2016-0-00078) and the ITRC (IITP-2017-2012-0-00646). Authors would like to thank all the anonymous reviewers for their valuable feedback.

References

1. Open vSwitch. http://openvswitch.org
2. OpenFlow. https://www.opennetworking.org/sdn-resources/openflow
3. Anagnostopoulos, M., Kambourakis, G., Kopanos, P., Louloudakis, G., Gritzalis, S.: DNS amplification attack revisited. Comput. Secur. **39**, 475–485 (2013)
4. Beverly, R., Bauer, S.: The Spoofer project: inferring the extent of source address filtering on the Internet. In: Proceedings of the 1st USENIX Workshop on Steps to Reducing Unwanted Traffic on the Internet (2005)
5. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM **13**, 422–426 (1970)

6. Bremler-Barr, A., Levy, H.: Spoofing prevention method. In: Proceedings of the 24th IEEE International Conference on Computer Communications (2005)
7. Deshpande, T., Katsaros, P., Basagiannis, S., Smolka, S.A.: Formal analysis of the DNS bandwidth amplification attack and its countermeasures using probabilistic model checking. In: Proceedings of the 13rd IEEE Conference on High-Assurance Systems Engineering (2011)
8. Di Paola, S., Lombardo, D.: Protecting against DNS reflection attacks with bloom filters. In: Holz, T., Bos, H. (eds.) DIMVA 2011. LNCS, vol. 6739, pp. 1–16. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22424-9_1
9. Gallagher, S.: How Spamhaus' attackers turned DNS into a weapon of mass destruction (2013). https://arstechnica.com/information-technology/2013/03/how-spamhaus-attackers-turned-dns-into-a-weapon-of-mass-destruction/
10. Guo, F., Chen, J., Chiueh, T.C.: Spoof detection for preventing DoS attacks against DNS servers. In: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems (2006)
11. Kambourakis, G., Moschos, T., Geneiatakis, D., Gritzalis, S.: Detecting DNS amplification attacks. In: Lopez, J., Hämmerli, B.M. (eds.) CRITIS 2007. LNCS, vol. 5141, pp. 185–196. Springer, Heidelberg (2008). doi:10.1007/978-3-540-89173-4_16
12. Katsurai, Y., Nakamura, Y., Takahashi, O.: A proposal of a countermeasure method against DNS amplification attacks using distributed filtering by traffic route changing. In: Proceedings of the 9th International Workshop on Informatics (2015)
13. Kim, H., Feamster, N.: Improving network management with software defined networking. IEEE Commun. Mag. **51**, 114–119 (2013)
14. Kloti, R., Kotronis, V., Smith, P.: Openflow: a security analysis. In: Proceedings of the 21st IEEE International Conference on Network Protocols (2013)
15. Kreutz, D., Ramos, F.M., Verissimo, P.E., Rothenberg, C.E., Azodolmolky, S., Uhlig, S.: Software-defined networking: a comprehensive survey. Proc. IEEE **103**, 14–76 (2015)
16. Lara, A., Kolasani, A., Ramamurthy, B.: Network innovation using openflow: a survey. IEEE Commun. Surv. Tutor. **16**, 493–512 (2014)
17. Lexis, P., Mekking, M.: Identifying patterns in DNS traffic. Technical report, University of Amsterdam (2013)
18. Rastegari, S., Saripan, M.I., Rasid, M.F.A.: Detection of denial of service attacks against domain name system using machine learning classifiers. In: Proceedings of the 18th World Congress on Engineering (2010)
19. Senie, D., Ferguson, P.: Network ingress filtering: defeating denial of service attacks which employ IP source address spoofing. IETF RFC 2827 (1998)
20. Snoeren, A.C., Partridge, C., Sanchez, L.A., Jones, C.E., Tchakountio, F., Kent, S.T., Strayer, W.T.: Hash-based IP traceback. In: Proceedings of the 15th ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (2001)
21. Sun, C., Liu, B., Shi, L.: Efficient and low-cost hardware defense against DNS amplification attacks. In: Proceedings of the 24th IEEE Global Communications Conference (2008)
22. US-Cert: Alert (TA13-088A) DNS Amplification Attacks. https://www.us-cert.gov/ncas/alerts/TA13-088A (2013)
23. Vaughn, R., Evron, G.: DNS amplification attacks (2006). http://crt.io/DNS-Amplification-Attacks.pdf

24. Verma, S., Hamieh, A., Huh, J.H., Holm, H., Rajagopalan, S.R., Korczynski, M., Fefferman, N.: Stopping amplified DNS DDoS attacks through distributed query rate sharing. In: Proceedings of the 11st International Conference on Availability, Reliability and Security (2016)
25. Vixie, P.: Extension mechanisms for DNS (EDNS0). IETF RFC 2671 (1999)
26. Vixie, P.: DNS Response Rate Limiting (DNS RRL). ISC-TN-2012-1-Draft1 (2012)
27. Zhao, Y., Iannone, L., Riguidel, M.: On the performance of SDN controllers: a reality check. In: Proceedings of the 1st IEEE Conference on Network Function Virtualization and Software Defined Network (2015)