*Sindh Univ. Res. Jour. (Sci. Ser.) Vol. 48 **(4D)** 189-196 (2016)*

SINDH UNIVERSITY RESEARCH JOURNAL (SCIENCE SERIES)

DNS Amplification Attack Detection via Flexible Flow (sFlow)

A. A. A. M. ATAN*, M. N. M. M. NOOR, M. N. ISMAIL

Malaysian Institute of Information Technology, University of Kuala Lumpur, Malaysia

Received 28rd August 2016 and Revised 17th October 2016

Abstract: High availability plays a vital role in the Information Age. Ensuring availability involves the protection against network threats that could lead to unavailability, such as a Distributed Denial of Service (DDoS) attack. Relatively, Domain Name System (DNS) amplification attack is one of the biggest DDoS to date. The attack takes advantage of the circumstance that a small DNS query can cause amplified DNS replies. In regard to today's rapid networks, recent defense approaches have focused more on flow-based analysis as alternative countermeasure due to its efficacy in monitoring anomalous behaviour against fast-paced data streaming. However, fixed flow-monitoring application (i.e. NetFlow) is apparently known for its deferred processing mode which causes prominent delays in DDoS detection. Conversely, this paper proposed a method of detecting DNS amplification attack via flexible flow analysis. Preliminary results are based on the utilization of sFlow, immediate cache, and extended flow values involving DNS attributes.

Keywords: DNS amplification attack, DDoS detection, flow-based analysis, flexible flow, sFlow.

1. INTRODUCTION

DDoS attack is a growing issue in the Information Age. The attack typically works by flooding a target with bogus network traffic via various ways to overwhelm its resources and consequently causing cascading failures. Following the rise of IPv6 and Internet of Things (IoT), these attacks are undeniably becoming more abusive. A reflected DDoS is more detrimental because the attack is more distributed and mostly involves servers (amplifiers) that are capable of sending amplified responses.

On March 18 2013 (Anagnostopoulos *et.al.,*2013), a cyber-attack caused by DNS amplification has resulted in what many security experts are calling one of the largest reflected DDoS so far. The attack works by distributing spoofed DNS requests to numerous open resolvers. An open resolver is a DNS server that recursively replies to random DNS queries from anyone all over the cyberspace. Due to spoofing, these servers then would assume as if the DNS requests were sent from the intended target and all DNS responses are redirected (reflected) to that target instead (**Fig. 1).** The attack leverages the fact that a single DNS request (small investment) can generate large DNS responses (amplified returns), where Amplification Factor (AF) is defined as: response size / request size. Cisco (2015) highlighted that DNS amplification attack remains a concern in 2016. Conventional countermeasures like Firewall and IPS mostly rely upon predetermined attack patterns or malicious contents (Bauer *et. al.,* 2012). This however does not apply to protocols like DNS since it transparently involves legitimate traffic. They also do not scale well due to their stateful design, where each packet is compared against the state table for deep packet inspection (Sheth and Thakker, 2013). Hence, the approach would not be able to keep up with inspecting all data at high throughput levels. Thus, an incentive-driven approach is required and can be achieved via flow-based analysis (Nitin *et.al.,* 2012).

The main advantage of flow-based is its feasibility within high-speed networks since it relies on packet headers (Golling *et.al.,* 2014). A flow is a unidirectional sequence of packets that share the values of ingress interface, source/destination IP address with port number, protocol, ToS, and the non-key fields (i.e. durations, packets, bytes, and flows). A flow-based operation typically works by caching incoming packets in the flow cache for aggregation purpose **(Fig. 2).** The aggregated flows are then exported as flow records to a dedicated server (collector) for analysis **(Fig. 3).** As of today, there are two well-known types of flow-based application; NetFlow (fixed) (Chen *et.al.,* 2015) and NetFlow (flexible) (Afaq *et.al.,* 2015).

In this paper, Section 2 discusses the problem statement of fixed flow-monitoring. Section 3 is methodology in which the fundamental concepts of flexible flow are discussed. Section 4 describes the proposed approach to deploy testbed. Section 5 is the discussion, and finally section 6 concludes the paper.

++ Corresponding author: Ahmad Ariff Aizuddin Mohd Atan email: ahmadariffaizuddin@gmail.com
*Faculty of Defiance Science and Technology, National Defence University of Malaysia, Malaysia.

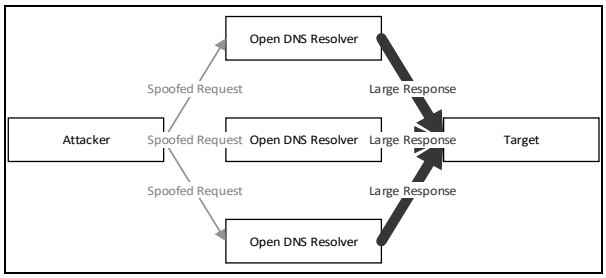


Fig. 1. Theoretical concept of DNS amplification attack

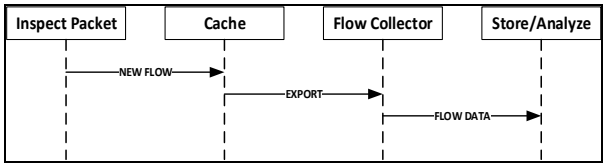


Fig. 2. Standard flow-based operation

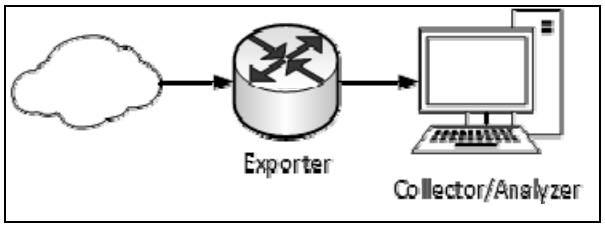


Fig. 3. Standard flow-based component

2. PROBLEM STATEMENT

NetFlow is highly subjected to cache timeout, where at least 5 to 10 minutes of aggregated flows are required before the traffic can be determined as benign or malicious. Though the timeout value can be reduced to a minimum of 1 minute, it would cause inadequate data collection and high false positives. Then again, 5 to 10 minutes duration is contradicting the concept of a near real-time environment. Cache timeout is not the only factor of delays, but the predefined values of fixed flow itself are restricted to give enough information for real-time detections; higher thresholds required. Moreover, flow cache has explicit size limits. In the event of an attack, the amount of incoming traffic would rapidly intensify and lead to cache exhaustion. Existing flows will not be exported until the timeout is reached, and new flows will not be cached until the cache is cleared. This is considered as intrusive if the new flows consist of normal traffic. Most of the flow-based researches and analysis rely upon NetFlow since it was formerly considered as a de-facto industry standard and has large community support back then (Li *et. al.,* 2013) (Hofstede *et. al.,* 2014) (Huistra. 2013) (Rossow. 2014). Nowadays, sFlow is gaining considerable attention due to its association with recent technology like Software-Defined Networking (SDN) and alternative flow records' options.

3. METHODOLOGY

sFlow permits users to add extended flow values that suited the analysis goals (i.e. DNS application data). The data consist of, but not limited to, query ID, query type, query name, return code, and number of returned answers. Conversely, extended flow values might unnecessarily increase the size of flow records. However, not all DNS application data are required. As long as the optimally selected values fulfill the detection criterion, it would not excessively increase the size of flow records. Apart from the flow values, sFlow permits users to choose between normal and immediate cache as well. Immediate cache enables flows to be exported instantly based on certain flow values (e.g. port 53). If a flow is parsed and does not meet the criteria, it is stored in the normal cache instead.

This paper proposed a detection method of DNS amplification attack through the monitoring of optimal DNS application data via sFlow module. The performance of proposed solution includes the measurements and aspects of response time; comprising average detection duration and interval (period between time of attack and time of detection), in conjunction with varying normal to abnormal ratio, network speed, DNS queries, attack sources, and total volume sent.

3.1 DNS Application Data

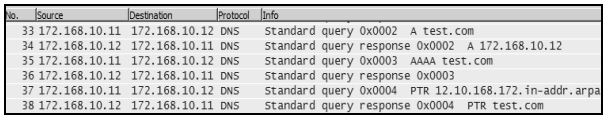
Out of the five highlighted metrics in Section 3, DNS query transaction ID appears suitable to fulfill the requirement of being distinctive and lightweight as only a single counter is needed.

Theoretically, a targeted network shall receive huge sums of traffic from DNS servers while there is no or minimal request involved (Rossow. 2014). Conventionally, this can be computed through the ratio between received and sent bytes. If the ratio is larger than a threshold or equals to 0, then it will be marked as suspicious.

However, the computational process can only be done after a predetermined cache timeout as shorter period will lead to deficient outcomes (false positives). In terms of Query ID, benign DNS traffic should have equivalent Query ID for both queries and replies. Else, there is an attempt of DNS amplification attack. Compared to conventional approaches, Query ID does not require excessive computation of flow values.

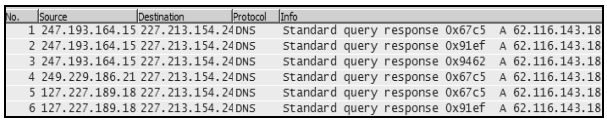
(Fig. 4) shows an example of benign DNS traffic captured from simulated data, where for every standard query (e.g. 0x0002 A test.com), there is a standard query response that matches the transaction ID, in hex value, (e.g. 0x0002 A 172.168.10.12). While **(Fig. 5)** shows a snippet of malicious DNS traffic captured from a virtualized target, where multiple standard query responses with multiple occurrences, do not have its corresponding standard query.

DNS query transaction ID is one of the packet-level information wherein it is not technically visible in the IP packet header or standard flow values (fixed flow).



No.	Source	Destination	Protocol	Info
33	172.168.10.11	172.168.10.12	DNS	Standard query 0x0002 A test.com
34	172.168.10.12	172.168.10.11	DNS	Standard query response 0x0002 A 172.168.10.12
35	172.168.10.11	172.168.10.12	DNS	Standard query 0x0003 AAAA test.com
36	172.168.10.12	172.168.10.11	DNS	Standard query response 0x0003
37	172.168.10.11	172.168.10.12	DNS	Standard query 0x0004 PTR 12.10.168.172.in-addr.arpa
38	172.168.10.12	172.168.10.11	DNS	Standard query response 0x0004 PTR test.com

Fig. 4. Benign DNS packet capture via Wireshark



No.	Source	Destination	Protocol	Info
1	247.193.164.15	227.213.154.24	DNS	Standard query response 0x67c5 A 62.116.143.18
2	247.193.164.15	227.213.154.24	DNS	Standard query response 0x91ef A 62.116.143.18
3	247.193.164.15	227.213.154.24	DNS	Standard query response 0x9462 A 62.116.143.18
4	249.229.186.21	227.213.154.24	DNS	Standard query response 0x67c5 A 62.116.143.18
5	127.227.189.18	227.213.154.24	DNS	Standard query response 0x67c5 A 62.116.143.18
6	127.227.189.18	227.213.154.24	DNS	Standard query response 0x91ef A 62.116.143.18

Fig. 5. Malicious DNS packet capture via Wireshark

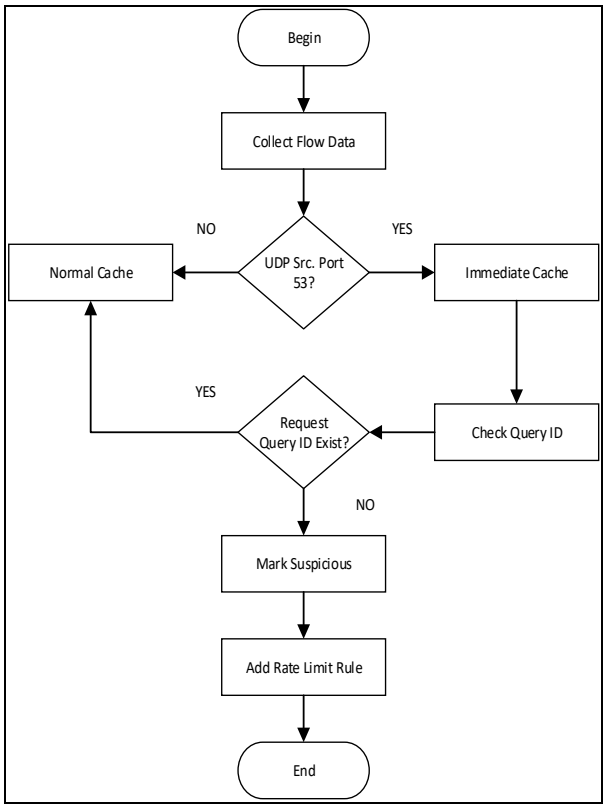


Fig. 6. Flowchart of proposed detection logic

Algorithm I Proposed Detection of DNS Amplification Attack

```
1: function GetDNSResponse
2: Fresponses = {IPsrc, IPdest, Psrc, Pdest, Prot, Bytes}
3: FDNS = {QID}
4: export Fresponses by Psrc to Fexported
5: for each Fresponse in Fexported do
6: Fext. = Fresponse.FDNS = {IPsrc, IPdest, Psrc, Pdest, Prot,
   Bytes, QID}
7: if QID > 1 or QID = 0 then
8: return "Fresponse.IPsrc suspicious"
9: end if
10: end for
11: end function
```

As a result, flexible flow compensates for the drawback of standard flow values by adding selective metadata extracted from IP packet payload onto the detection logic.

3.2 Prospective Detection

Hypothetically, the proposed detection logic classifies between normal and abnormal traffic by utilizing flexible flow values, in which any dissimilarity or absence of DNS query transaction ID would signify an attack. In relation, the module intends to solve issues regarding limitation of flow values and cache overload. In practice, it is aimed to work below than 1 minute of data, as a trade-off between the collection of data and the preference to work in near real-time.

Both flowchart in **(Fig. 6)** and pseudocode of Algorithm I show that the proposed detection logic starts by collecting flow data from the forwarding device (collector) and check the predefined flow values in the packet header to see if the traffic originated from DNS servers. If the source port number belongs to other than 53, the flow is stored in the normal cache, else, it is stored in the immediate cache.

Next, DNS traffic stored in the immediate cache is further scrutinized by looking into additional flow values or DNS application data to see if there is a matching request query ID in the flow records. If yes, the traffic is classified as a normal DNS response, else, it is marked suspicious. In order to pre-validate the hypothesis, a series of experiments are conducted on a testbed encompassing varying network profiles and attack scenarios.

4. TESTBED

(Fig. 7) shows the initial setup of the testbed, where all research instruments are mainly set on a single machine. The machine runs a 64-bit Windows 7 with Intel Core i5-3337U CPU @ 1.80GHz and 8 GB RAM. The topology consists of one controller, single switch, and two hosts (attacker and victim).

To produce the presented topology, a hypervisor (VirtualBox) is installed along with a virtual machine appliance (SDN Hub Tutorial VM). The virtual machine runs a 64-bit Ubuntu 14.04 LTS with Intel Core i5-3337U CPU @ 1.80GHz and 2 GB RAM. On top of that, the virtual machine is preloaded with a network emulator (Mininet), virtual switch/exporter (Open vSwitch), flow analyzer/collector (sFlow-RT; where the detection logic resides and enabled), and customizable APIs for the detection logic.

Prior to producing synthetic data, a number of network profiles are selected for the experimental procedure.

4.1 Network Profile

Though the basic setup of the experiment is based on Fig. 7, number of attacker (DNS servers) varies depending on each network profile.

The profiles are selected based on the studies and surveys done by (van Rijswijk-Deij *et. al.,* 2014) and (Ryba *et. al.,* 2015).

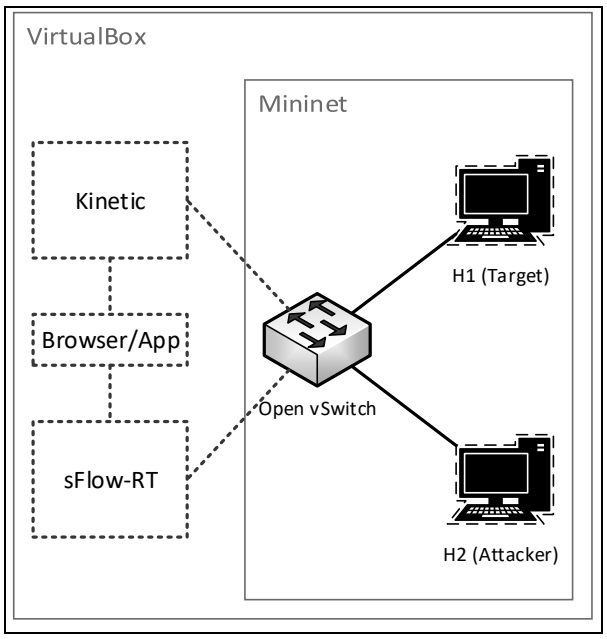


Fig. 7. Proof-of-concept topology

In order to simulate attack scenarios and generate the preliminary results, the following profiles are considered:

- 25 Test Runs (each)
- Average Attack Duration (seconds)
- Average Detection Duration (epoch)
- Interval (period between attack and detection)
- Ratio (Normal/Abnormal): 10% 50% 100%
- Total Volume Sent (Mbps): 100 500 1000
- Query Types: ANY A TXT
- DNS Servers: 2, 4, 8, 12, 22
- Speed: 100Mbps 500Mbps 1Gbps

4.2 Attack Scenario

As described in Section 1, DNS reflection works by sending spoofed DNS requests to a number of DNS servers that will automatically respond to that DNS requests. Hence, before simulating the amplification attack, two main aspects should be taken into account; attacker should be able to spoof the IP address of target, and DNS servers should be able to send large DNS responses to the requester. Two Python scripts are used for the attacker and DNS servers. Both scripts are coded to suit the controlled environment (supervised variables). In order to allow both scripts to be accessible through the corresponding Mininet hosts, they are placed in the root directory of SDN Hub Tutorial VM by default. The first script (open_resolver), which is based on DNSLib, is issued on additional hosts' terminal (i.e. other than attacker and target) via command python open_resolver.py, to mimic the application of open resolvers. DNSLib is a Python library that provides the framework of a server.

Table-1: A snippet of detection result via proposed solution (10% attack ratio with 100Mbps volume sent)

ID	*(%)*	*(Mbps)*	*Query*	*DNS*	*Attack Duration*	*Detection Time*	*Interval*
A1	10	100	ANY	2	21.17	13958	±8.17
A2	10	100	A	2	28.30	13905	±15.3
A3	10	100	TXT	2	43.15	13174	±30.15
A4	10	100	ANY	4	41.70	13890	±28.7
A5	10	100	A	4	48.73	13492	±35.73
A6	10	100	TXT	4	63.40	13312	±50.4
A7	10	100	ANY	8	81.75	13908	±68.75
A8	10	100	A	8	88.16	13538	±75.16
A9	10	100	TXT	8	103.38	13401	±90.38
A10	10	100	ANY	12	121.77	13805	±108.77
A11	10	100	A	12	128.38	13338	±115.38
A12	10	100	TXT	12	143.75	13299	±130.75
A13	10	100	ANY	22	221.76	13969	±208.76
A14	10	100	A	22	228.80	13538	±215.8
A15	10	100	TXT	22	243.39	13526	±230.39

The second script (dnsamp), which is based on Scapy, is issued on attacking host's terminal via command python dnsamp.py, to craft spoofed DNS requests. Scapy is a Python library that offers the customization and generation of packets for a wide range of protocols. Accordingly, findings of the experiments are compared with existing solutions done by previous related works.

5. PRELIMINARY DISCUSSION

This section evaluates the efficiency and effectiveness of using DNS Query ID to detect DNS amplification attack on synthetic dataset.

The overall preliminary results of this method can be referred in **(Table-1), (Fig. 8, 9, and 10).**

In (**Table-1-3)**, the first row denotes the record ID, second row denotes the attack ratio between normal and abnormal traffic, third row denotes the total volume sent in Mbps, the fourth row denotes the query types where the size (bytes) of each query is approximately ±4096 for ANY, ±2900 bytes for A, and ±512 for TXT. The fifth row denotes the number of open resolvers, the sixth row denotes the average attack duration in seconds, the seventh row denotes the average detection duration in epoch, and the last row denotes the period, in seconds, between attack and detection (interval). The obtained results are observed and analyzed by taking

into account the network speed, total test runs, average attack and detection duration, as well as the period between attack and detection, induced from the dataset.

As stated in Section 3, the proposed solution is aimed to work below than 1 minute of data, as a trade-off between collection of data and preference to work in near real-time. In terms of practicality, emphasis is given more to detection speed over accuracy, provided that the proposed solution is able to achieve a minimum of 90% accuracy in less than a minute.

From (**Table-1, Fig. 8, 9, and 10), i**t is observed that the average detection time increases as the total volume sent rises.

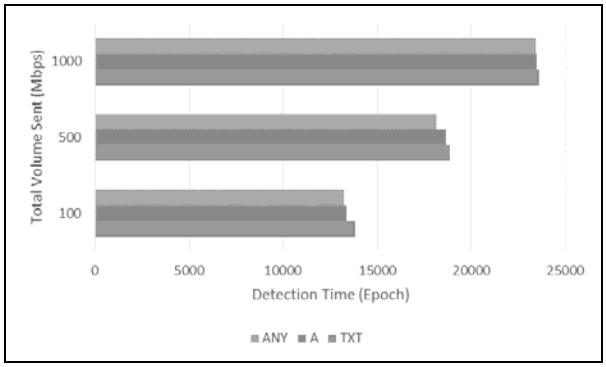


Fig. 8. 10% Attack ratio over 1Gbps link

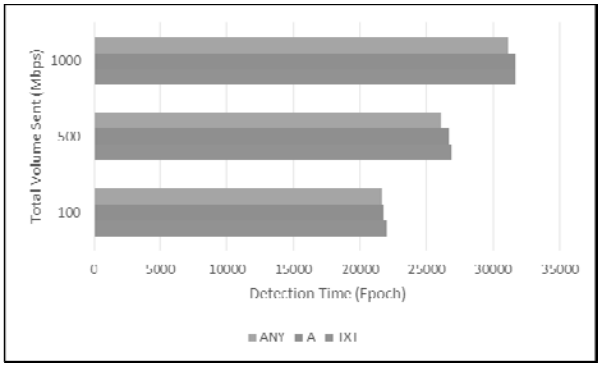


Fig. 9. 50% Attack ratio over 1Gbps link

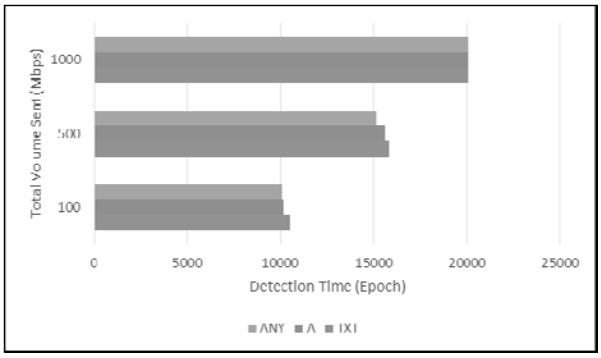


Fig. 10. 100% Attack ratio over 1Gbps link

This is due to the average duration it takes for an attack to complete its cycle; finish distributing all malicious traffic. It is also observed that there is an increment in the average detection time for the 50% attack ratio compared to 10% and 100%. This is considered nonstandard, though considerably acceptable, since the average detection time pattern is projected to be decreasing as the attack ratio increases.

In Table-1, record ID A1, A4, and A7 show that the interval value increases as the number of open resolvers rises (i.e. ±8.17 seconds, ±28.47 seconds, and ±68.75 seconds). This means that a more open resolvers involved in an attack, the quicker the malicious traffic is detected due to the increasing levels of distinctiveness in DNS Query ID.

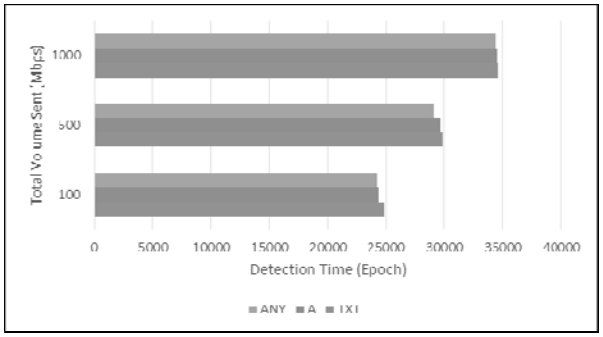


Fig. 11. 10% Attack ratio over 500Mbps link

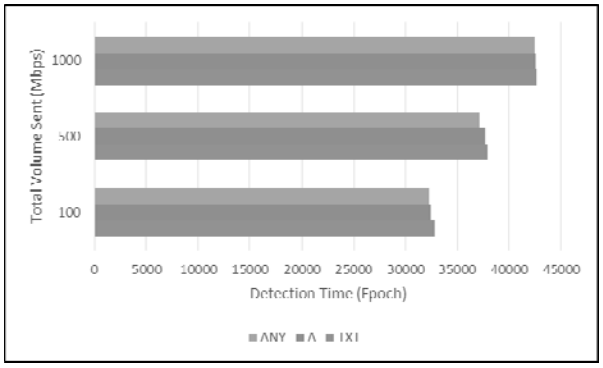


Fig. 12. 10% Attack ratio over 100Mbps link

(Fig. 8, 9, and 10) show that there is a slight decrement in the average detection time, when measured in epoch, as the response size for each type of query increases. This conforms the principal of DNS amplification attack, where bigger response size leads to the rapid saturation of network link, hence speeds up both average attack and detection duration. It should be noted that the results in and Fig. 8, 9, and 10, are tested with 1Gbps of network speed.

Fig. 11 represents the results for 500Mbps, while Fig. 12 represents the results for 100Mbps. Based on **(Fig. 8, 11, and 12),** it is perceived that both average

attack and detection duration increases as the network speed decreases. This corresponds to the postulation that the slower the speed, the more time it takes for an attack to complete its cycle, thus slows down detection time. Likewise, the overall pattern for 500Mbps and 100Mbps are gradually comparable to 1Gbps of network speed.

The maximum average detection time recorded, counting in the discrepancies with 50% attack ratio, is approximately within 50 seconds; fulfilling the requirement of the proposed solution to work below 1 minute of data.

Relatively, both (Huistra. 2013) and (Rossow. 2014) are among the current and closely related researches in terms of identifying reflected DDoS attack via flow-based analysis. For comparative study purpose, their methods are replicated in the research framework as well.

Table-2: A snippet of detection result based on method (10% and 50% attack ratio with 100Mbps volume sent)

ID	*(%)*	*(Mbps)*	*Query*	*DNS*	*Attack Duration*	*Detection Time*	*Interval*
B1	10	100	ANY	2	20.59	60883	±-39.41
B2	10	100	A	2	29.18	60874	±-30.82
B3	10	100	TXT	2	42.88	60187	±-17.12
B4	10	100	ANY	4	42.82	60724	±-17.18
B5	10	100	A	4	47.74	60356	±-12.26
B6	10	100	TXT	4	64.48	60239	±4.48
B7	10	100	ANY	8	80.95	60868	±20.95
B8	10	100	A	8	89.45	60452	±29.45
B9	10	100	TXT	8	102.70	60326	±42.7
B10	10	100	ANY	12	122.20	60739	±62.2
B11	10	100	A	12	127.10	60273	±67.1
B12	10	100	TXT	12	144.18	60120	±84.18
B13	10	100	ANY	22	220.44	60856	±160.44
B14	10	100	A	22	229.15	60496	±169.15
B15	10	100	TXT	22	242.51	60454	±182.51
B16	50	100	ANY	2	25.47	49896	±-23.53
B17	50	100	A	2	26.87	49454	±-22.13
B18	50	100	TXT	2	48.26	49071	±-0.74
B19	50	100	ANY	4	43.20	49953	±-5.8
B20	50	100	A	4	47.45	49691	±-1.55
B21	50	100	TXT	4	69.88	49530	±20.88

5.1 Comparison with Huistra on Synthetic Dataset

This section evaluates the comparison between proposed solution and existing solution done by (Huistra. 2013).

(Huistra. 2013) proposed a detection of DNS amplification attack using fixed flow data generated from NetFlow-configured router. The detection logic is based upon total number of packets per flow and average number of bytes per flow, while the threshold is subjected to optimization and dependent on network profiles. For evaluation purpose, the method is only tested with 1Gbps of network speed.

(According to Huistra. 2013), regular DNS responses are inspected to correspond below the value of 1000 packets per flow or the average of 900 bytes per flow. In terms of threshold, the flows are considered suspicious if total number of packets per flow exceeds 1000 packet counts, or if average number of bytes per flow exceeds 900 bytes for flows with less than 1000 packet counts. The reduction of these values would raise inaccuracy or false positive rates.

Accordingly, any defined threshold in this detection logic is multiplied 100x when tested upon non-sampled dataset as the dataset from this network is sampled at a rate of 1:100 for brevity.

The overall preliminary results of this method can be referred in **(Table-2), (Fig. 13, 14, and 15).**

From record ID B1 and B16 in Table-2, it is observed that the average detection time decreases as the attack ratio increases (i.e. ±60 seconds and ±49 seconds). Unlike the proposed solution, this method parallels the projected pattern, where the more attacking traffic are being distributed, the sooner they are being detected. This is due to the fact that the method is based upon statistical analysis (total packets and average bytes) instead of pattern matching (DNS Query ID).

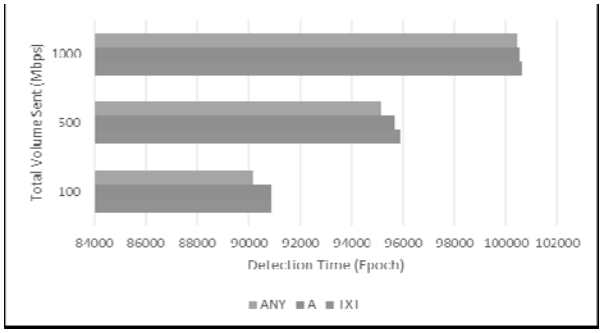


Fig. 13. 10% Attack ratio over 1Gbps)

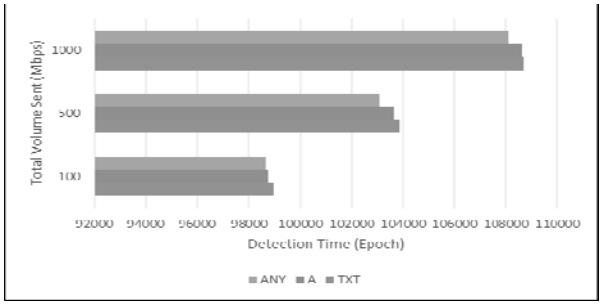


Fig. 14. 50% Attack ratio over 1Gbps

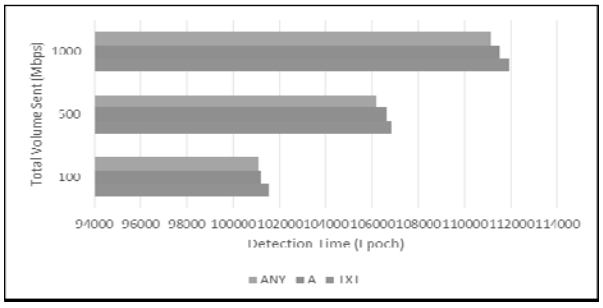


Fig. 15. 100% Attack ratio over 1Gbps

However, it is also observed that this method works better if the number of open resolvers involved are comparatively higher. Based on the generated data, record ID B5 and B20 in Table-3 show that the method is only able to detect an attack after its cycle is completed (network link saturated) if the number of open resolvers are relatively lower (±4 and below), as seen through the calculated negative intervals (i.e. ±-12.26 seconds and ±-1.55 seconds). Then again, the potential damage is expected to be trivial in some real case scenarios since it would lessen the amplification effect to have smaller amounts of open resolver and defeat the sole purpose of the attack. Conversely, the maximum average detection time recorded is approximately within 300 seconds (over 100Mbps link); 6 times more than the proposed solution.

Table-3: A snippet of detection result based on method (10% attack ratio with 100Mbps volume sent)

ID	*(%)*	*(Mbps)*	*Query*	*DNS*	*Attack Duration*	*Detection Time*	*Interval*
C1	10	100	ANY	2	22.97	90739	±-37.03
C2	10	100	A	2	27.58	90715	±-32.42
C3	10	100	TXT	2	44.27	90133	±-15.73
C4	10	100	ANY	4	40.56	90661	±-19.44
C5	10	100	A	4	49.82	90216	±-10.18
C6	10	100	TXT	4	62.15	90131	±-27.85
C7	10	100	ANY	8	82.81	90745	±-7.19
C8	10	100	A	8	87.93	90354	±-2.07
C9	10	100	TXT	8	104.68	90249	±14.68
C10	10	100	ANY	12	120.86	90652	±30.86
C11	10	100	A	12	129.46	90147	±39.46
C12	10	100	TXT	12	142.39	90141	±52.39
C13	10	100	ANY	22	222.89	90728	±132.89
C14	10	100	A	22	227.78	90337	±137.78
C15	10	100	TXT	22	244.80	90257	±154.8

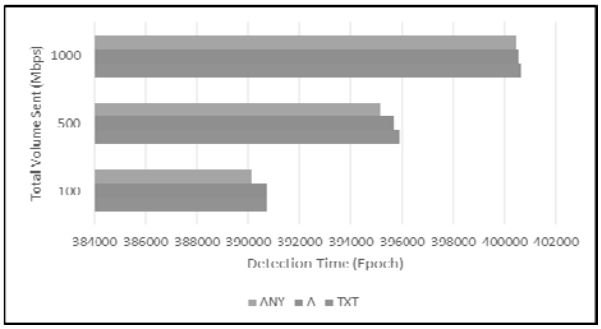


Fig. 16. 10% Attack ratio over 1Gbps link via Rossow

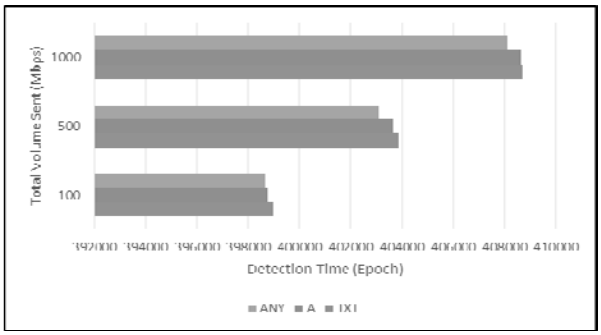


Fig. 17. 50% Attack ratio over 1Gbps link via Rossow

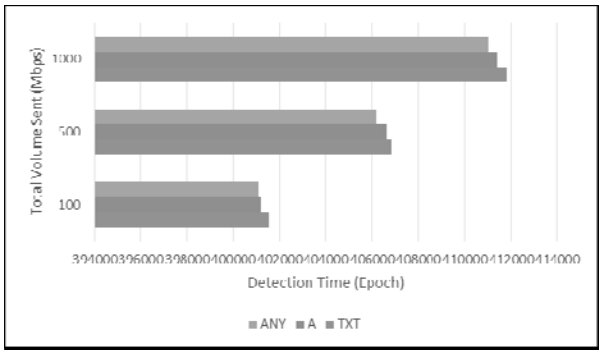


Fig. 18. 100% Attack ratio over 1Gbps

5.2 Comparison with Rossow on Synthetic Dataset

This section evaluates the comparison between proposed solution and existing solution done by (Rossow 2014). The author proposed a fixed flow-based solution to detect DNS amplification attack using parameters obtained from NetFlow-configured exporter. The detection logic is based upon number of received bytes-per-pairflow, and ratio of sent and received bytes, while the threshold is subjected to optimization and dependent on network profiles. For evaluation purpose, the method is only tested with 1Gbps of network speed. According to (Rossow 2014), regular DNS responses are inspected to correspond below the value of 100,000 bytes-per-pairflow. Furthermore, it is inspected that the numbers of replies for a regular host should parallel the numbers of queries. In terms of threshold, the flows are considered suspicious if number of received bytes-per-pairflow exceeds 100,000 bytes, and ratio of sent and received bytes equals to zero: bytes to server / bytes to client (since a client would rarely send requests to the server, bytes to server value remains mostly zero). The overall preliminary results of this method can be referred in Table-3, **Fig**. 16, 17, and 18. Though the hypothetical concept is practically similar to the proposed solution, in which a targeted network shall receive huge sums of traffic from DNS servers without any request involved, this method is still based upon statistical analysis instead of pattern matching. This means that the detection is susceptible to delays in the same way as Huistra (2013), where the attack would not be detected until it exceeds the threshold value. Moreover, it also reflects that the overall pattern of the method is technically equivalent to Huistra (2013); when the attacking traffic increases, the detection time decreases. Nonetheless, record ID C1, C4, and C7 in Table-3 show that there is a minor difference in terms of the number of open resolvers required for this method to operate adequately (±12 and above), as seen through the calculated negative intervals (i.e. ±-37.03 seconds, ±-19.44 seconds, and ±-7.19 seconds). The maximum average detection time recorded is approximately within

600 seconds (over 100Mbps link); 12 times more than the proposed solution.

6. CONCLUSION

This paper addressed the significance of flexible flow-based analysis and substitute flow values in detecting DNS amplification attack. Based on average detection duration and interval, preliminary results showed that the proposed solution rationalizes the hypothesis presented in Section 3, in which the detection logic should at any rate perform better than the related studies (below 1 minute of data). Additionally, the method includes an alternative to ease memory consumption of flow exporters by precluding cache timeout.

6.1 Future Work

Although the testing is based on emulated network, experimental output should be commutable to network traces involving actual malicious DNS responses. Further refinements will focus on the evaluation of proposed solution with authentic dataset, which include the measurements and aspects of false positives, false negatives, accuracy, precision, and recall, as well as its integration with a reasonable mitigation approach to diminish amplified DNS traffic, based on traffic shaping mechanism in security-centric SDN.

REFERENCES:

Afaq. M., S. Rehman, and W. C. Song, (2015). Large Flows Detection, Marking, and Mitigation based on sFlow Standard in SDN. Journal of Korea Multimedia Society, 18 (2): 189-198. DOI:10.9717/kmms..18.2.189.

Anagnostopoulos. M., G. Kambourakis, P. Kopanos, G. Louloudakis, and S. Gritzalis, (2013). DNS Amplification Attack Revisited. Computer Security, 39 (0167-4048):475-485. DOI:10.1016/j.cose.2013.10.001.

Bauer. E., R. Adams, (2012). Denial of Service Attacks. In: Reliability and Availability of Cloud Computing, Wiley, Hoboken, NJ, 159-160. ISBN: 9781118177013.

Chen. Z., Y. Wen, J. Cao, W. Zheng, J. Chang, Y. Wu, G. Ma, M. Hakmaoui, and G. Peng, (2015). A Survey of Bitmap Index Compression Algorithms for Big Data. Tsinghua Science and Technology, 20 (1): 100-115. DOI:10.1109/TST.2015.7040519.

Cisco., (2015). Annual Security Report.Cisco Public Information,California,USA.http://www.cisco.com/web /offers/lp/2015-annual-security-report/index.html (Accessed on March 17, 2016).

Golling M., R. Koch, and R. Hofstede, (2014). Towards multi-layered intrusion detection in high-speed networks. 2014 6th International Conference on Cyber Conflict, June 3-6, IEEE Xplore Press, Estonia, 191-206. DOI: 10.1109/CYCON.2014.6916403.

Hofstede. R., P. Celeda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, (2014). Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX. Communications Surveys & Tutorials, 16 2037-2064. DOI:10.1109/COMST.2014.2321898.

Huistra. D., (2013). Detecting Reflection Attacks in DNS Flows. In: 19th Twente Student Conference on IT, Enschede, Netherlands.

Li. B., J. Springer, G. Bebis, and M. H. Gunes, (2013). A survey of network flow applications. Computer Applications, 36 (567-581): 1084-8045. DOI:10.1016/j.jnca.2012.12.020.

Nitin. T., S. R. Singh, and P. G. Singh, (2012). Intrusion Detection and Prevention System (IDPS) Technology- Network Behavior Analysis System (NBAS). Engineering Sciences, 1 51-56. URL:http://www.isca.in/IJES/Archive/v1i1/9.ISCA-JEngS-2012-032.pdf.

Rossow. C., (2014). Amplification Hell: Revisiting Network Protocols for DDoS Abuse. Proceedings of the 2014 Network and Distributed System Security, Feb. 23-26, Internet Society, USA, pp: 1-15. DOI: 10.14722/ndss.2014.23233.

Ryba. F. J., M. Orlinski, M. Wählisch, C. Rossow, and and Schmidt, (2015). Amplification and DRDoS Attack Defense - A Survey and New Perspectives.http://arxiv.org/abs/1505.07892 (Accessed on February 5, 2015).

Sheth. C., and R. Thakker, (2013). Performance Evaluation and Comparison of Network Firewalls under DDoS Attack. Computer Network and Information Security, 5 (60-67): 2074-9090. DOI:10.5815/ijcnis.2013.12.08.

Van Rijswijk-Deij, R., A. Sperotto, and A. Pras, (2014). DNSSEC and Its Potential for DDoS Attacks - A Comprehensive Measurement Study. Proceedings of the 14th ACM Internet Measurement Conference, Nov. 5-7, ACM, USA, 449-460. DOI: 10.1145/2663716.2663731.