Detecting DNS Amplification Attacks

Georgios Kambourakis, Tassos Moschos, Dimitris Geneiatakis, Stefanos Gritzalis

Laboratory of Information and Communication Systems Security
Department of Information and Communication Systems Engineering
University of the Aegean, Karlovassi, GR-83200 Samos, Greece
{gkamb, tmos, dgen, sgritz}@aegean.gr

Abstract. DNS amplification attacks massively exploit open recursive DNS servers mainly for performing bandwidth consumption DDoS attacks. The amplification effect lies in the fact that DNS response messages may be substantially larger than DNS query messages. In this paper, we present and evaluate a novel and practical method that is able to distinguish between authentic and bogus DNS replies. The proposed scheme can effectively protect local DNS servers acting both proactively and reactively. Our analysis and the corresponding real-usage experimental results demonstrate that the proposed scheme offers a flexible, robust and effective solution.

Keywords: DNS Security; Denial of Service; DNS Amplification Attacks; Detection and repelling mechanisms.

1 Introduction

Beyond doubt, the Internet is the ultimate terrain for attackers who seek to exploit its infrastructure components in order to achieve an unauthorised access or to cause a Denial of Service (DoS). DoS attacks can be classified into two major categories. In the first one, the adversary featly crafts packets trying to exploit vulnerabilities in the implemented software (service or protocol) at the target side. This class of attacks includes outbreaks like the ping of death [1]. In the second one, the aggressor attempts to overwhelm critical system's resources, i.e. memory, CPU, network bandwidth by creating numerous of well-formed but bogus requests. This type of attack is also well known as flooding. Several incidents in the Internet have been already reported in the literature [2]-[5] as flooding attacks, affecting either the provided service or the underlying network infrastructure. The most severe among them is presented in [2] and is known as Reflection Distributed DoS (RDDoS). Such attacks can cost both money and productivity by rapidly paralyzing services in the target network.

Recent attack incidents verify the catastrophic outcomes of this class of attacks when triggered against key Internet components like Domain Name System (DNS) servers. For example, as reported in [2], in October 2002 eight out of the thirteen root DNS servers were suffered a massive DoS attack. Many other similar attacks were triggered against DNS in 2003 and 2004 [13], [14]. In a recent study, the Distributed

Denial of Service (DDoS) activity in the Internet was analyzed employing a method called “backscatter” [15]. The results of this study showed that nearly 4,000 DDoS attacks are released each week. In February 2006, name servers hosting Top Level Domain (TLD) zones were the frequent victims of enormous heavy traffic loads.

Contrariwise to normal DDoS attacks, where an arsenal of bots mounts an assault on a single targeted server, the new attacks unfold by sending queries to DNS servers with the return address aiming at the victim. In all cases the primary victim may be the local DNS server(s) itself. Bandwidth exhaustion caused affects normal network operation very quickly and incapacitates the target machine. For example, very recently, in May, 2007, US-CERT has received a report that Estonia was experiencing a national DDoS attack. According to the source, the attacks consisted of DNS flooding of Estonia's root level servers. By this time 2,521 unique IP's have been identified as part of the attacking botnets. This situation is far more difficult to prevent because in this case the DNS server performs the direct attack. For instance, in an ordinary DDoS attack, one can potentially block a bot instructed to launch a DDoS attack by blocking the bot’s IP address. Contrariwise, it is not so simple to block a DNS server without affecting and damaging the operation of a corporate network. The amplification factor in such recursive DNS attacks stems from the fact that tiny DNS queries can generate much larger UDP responses. Thus, while a DNS query message is approximately 24 bytes (excluding UDP header) a response message could easily triple that size. Generally, this outbreak takes advantage the fact that the DNS is needed by any service (http, ftp etc) requires name resolution.

In this paper we focus on DNS amplification attack suggesting a novel, practical and effective solution to mitigate its consequences. Our repelling mechanism can protect local DNS servers both proactively and reactively. Specifically, it can proactively alert administrators before the attack affects DNS server operation, and reactively by automatically blocking bots’ IP addresses at the firewall or the edge router(s). This means that every local network host is well protected too, in case that it is the actual target of the attack taking place. Actually, some bogus DNS replies will reach the target host at the first stages of the attack, but as soon as an alert is generated all subsequent falsified DNS replies will be dropped at the perimeter. We also evaluate our mechanism considering real-usage scenarios, false positives and false negatives. The rest of the paper is organized as follows. Next section focuses on DNS DoS flooding attacks, while Section 3 presents the existing countermeasures and remedies proposed so far. Section 4 introduces and evaluates the proposed mechanism, in terms of response time, false negatives and false positives. Section 4 draws a conclusion giving also some pointers for future work.

2 Flooding Attacks and the Domain Name System

2.1 General description and Problem statement

The main goal of any flooding attack is the expeditious consumption of critical system resources in order to paralyse the provided services and make them

unavailable to its legitimate users. Assuming that such an attack takes place against or exploits a critical component like the DNS it is very likely that would quickly incapacitate the overall network's services making it unavailable to any legitimate user. Several researchers have pointed out the threat of flooding attacks using recursive DNS name servers open to the world. For instance, according to a recent study [17], which is based on case studies of several attacked ISPs reported to have on a volume of 2.8 Gbps, one event indicated attacks reaching as high as 10 Gbps and used as many as 140,000 exploited name servers.

Flooding attacks against DNS are similar to other well documented Internet services flooding attacks and could be launched in two distinct ways. In the first case the attacker sends a large number of bogus DNS requests either from a single or multiple sources, depending on the flooding architecture utilized [4], [5]. An example of multiple sources flooding architecture attack against a DNS is depicted in Figure 1. According to this scenario, the attacker orchestrates usually innocent hosts, called bots, to simultaneously generate fake DNS requests aiming at disrupting the normal DNS operation by consuming its resources; mainly memory and CPU.

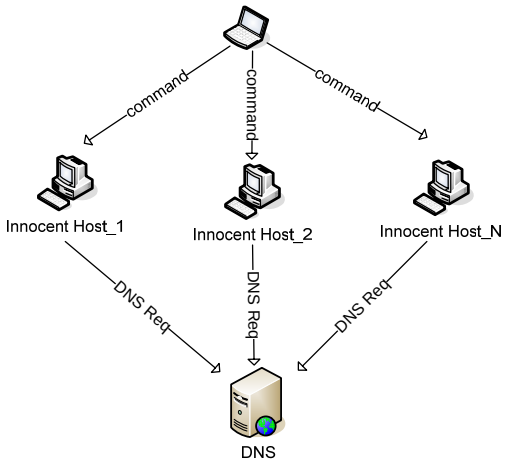

Fig. 1 Multiple sources flooding attack architecture

On the other hand, the most sophisticated and "modern" attacks exploit the DNS components themselves in an attempt to magnify flooding attack consequences. Putting it another way, in a DNS amplification attack scenario, the attacker exploits the fact that small size requests could generate larger responses. Especially, new RFC specifications supporting IPv6, DNS Secure, Naming Authority Pointer (NAPTR) and other extensions to the DNS system, require name servers to return much bigger responses to queries. The relation between a request and the corresponding response is known as the amplification factor and is computed using the following formula:

$$\textit{Amplification Factor} = \textit{size of (response)} / \textit{size of (request)}$$

The bigger the amplification factor is, the quicker the bandwidth and resource consumption at the victim is induced. Consequently, in the case of DNS amplification

attack the aggressor is based on the fact that a single DNS request (small data length) could generate very larger responses (bigger data length). For example, in the initial DNS specification [8] the DNS response was restricted up to 512 bytes length, while in [9] even bigger. The attack unfolds as follows: The attacker falsifies the source address field in the UDP datagram to be that of a host on the victims' network. Using the spoofed address, a DNS query for a valid resource record is crafted and sent to an intermediate name server. The latter entity is usually an open recursive DNS server, which forwards the final response towards the target machine as illustrated in Figure 2. The attacker will repeatedly send the query to the intermediate name server but with all the responses going to the victim network. Potentially, the adversary could consume the entire bandwidth of a T1 line by generating a few thousand responses.

Supposing that the attacker employs a distributed architecture similar to that presented in Figure 2, it is obvious that the bandwidth and resources consumption rate at the victim increase very rapidly. Furthermore, it should be noted that the attacker featly spoofs all query requests to include a specific type of DNS resource in order the authoritative DNS server to generate large responses. This task could be managed either by discovering which DNS servers store RRs that when requested create large responses or by compromising a DNS server and deliberately include a specific record – also known as the amplification record - that will create a large response. An example of this technique, exploiting large TXT records which is introduced in Extended DNS (EDNS) [9]. As stated in [17] by combining different response types, the amplification effect can reach up to a factor higher than 60. After that, the attacker collects a list of open recursive name servers that will recursively query for, and then return the amplification record he/she created. Even a list of known name servers may be more than adequate. As stated in [17] there is a 75% chance that any known name server is an open resolver too, thus a copy of a TLD zone file may be sufficient. A detailed description of DNS amplification attacks is presented in [6].

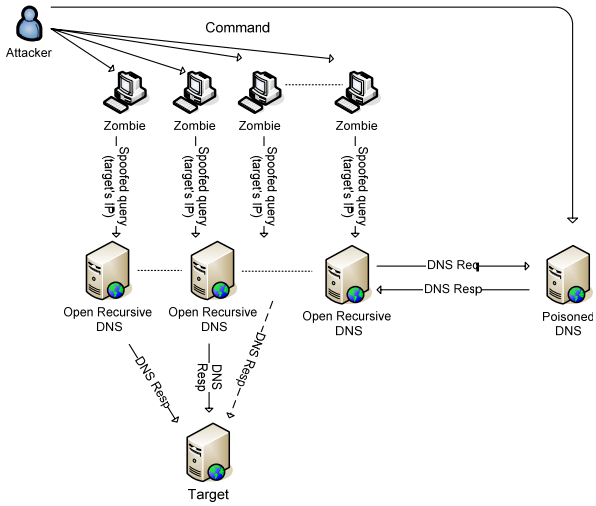

Fig. 2 General Architecture of a DNS amplification attack

2.2 Protection Mechanisms

In this section we present known countermeasures to defend against amplification attacks. Generally, in order to shield against DNS DDoS attacks different protection layers must be deployed. Having these mechanisms acting simultaneously, it is very possible to build a more secure, redundant and robust DNS infrastructure and shield our network against this category of attacks.

DNS employs UDP to transport requests and responses. As a result, the malicious user is able to fabricate the appropriate spoofed DNS requests very easily. Thus, as a first level of protection it should be introduced a spoof detection / prevention mechanism like the ones proposed in [10]-[13]. In some cases such mechanisms are implemented as part of a stateful firewall as well. Moreover, to mitigate DNS cache poisoning and Man-In-The-Middle (MITM) attacks, which usually are launched at the early stages of a DNS amplification attack, additional security mechanisms should be employed. These are necessary in order to ensure the integrity and origin authentication of the DNS data that reside either in RR cache or in the zone file [10],[14].

Apart from well accepted practices to securely configure DNS servers [19], another effective remediation, at least against outsiders, is to disable open recursion on name servers from external sources and only accepting recursive DNS originating from trusted sources. This tactic substantially diminishes the amplification vector [18]. Available data until now reveal that the majority of DNS servers operate as open recursive servers. The Measurement Factory [17] reports that more than 75% of domain name servers of approximately 1.3 million sampled permit recursive name service to arbitrary querying sources. This leaves abandoned name servers to both cache poisoning and DoS attacks.

2.3 Limitations

Although the generic countermeasures and remedies referred in previous subsection could decrease the chances of potential attackers to launch a flooding attack, are not able to provide an effective solution against DNS amplification attacks. More specifically, it is well known that these mechanisms are employed only by a limited number of DNS servers. As a result many DNS servers are unprotected or misconfigured, which in turn are exploited by aggressors in order to amplify the hazardous effects of flooding attacks as described previously. Moreover, solutions like DNS Secure [10] do not offer an efficient countermeasure against flooding attacks as already argued in [15]. In addition, these mechanisms do not provide any security against (malevolent) insiders, who are responsible for many security incidents. On the top of that, the traffic generated in a DNS amplification attack seems to be normal, so the prevention of such an attack could not be achieved only with the employment of the security mechanisms presented in the previous section. Therefore, the introduction of a specific detection / prevention mechanism against DNS amplification attacks should be considered mandatory.

To the best of our knowledge until now the only method that specifically addresses DNS amplification attacks is the DNS-Guard one [20]. This approach involves

several policies that generate some form of cookies for a DNS server to implement origin authentication; that is to verify whether each incoming request is indeed from where the request datagram says it is from. However, the main problem with DNS-Guard is that it introduces large traffic and delay overhead and mandates wide scale deployment.

3. The proposed solution

Hereunder we describe and evaluate the proposed solution. It is stressed that our mechanism is primarily designed to effectively protect local DNS servers. As mentioned in the introduction local network hosts are also protected but indirectly. Actually, some bogus DNS replies will reach the host-victim at the first stages of the attack, but as soon as an alert is generated all subsequent falsified DNS replies will be dropped at the perimeter. In any case protecting local network hosts is rather a simple task to accomplish. That is, having the firewall to only accept traffic coming from trusted DNS servers. However, this solution is not possible to implement in a DNS server; blocking the 53 port would have undesired implications to the DNS service itself.

3.1 Description

The proposed mechanism is based on the one-to-one strict mapping of DNS requests (queries) and responses. Specifically, under DNS normal operation, once a client requests a name resolution sends a request towards the appropriate DNS, which is responsible to create the corresponding response. Nevertheless, when a DNS amplification attack is taking place, the targeted DNS server receives responses without having previously sent out the corresponding request. As a result, such data, characterised as orphan pairs, must be immediately classified as suspicious.

Based on the aforementioned simple but fruitful idea, we employ a monitor to record both DNS requests and responses using the IPtraf tool [16]. At the same time, our custom-made Hypertext Preprocessor (PHP) based tool, namely DNS Amplification Attacks Detector (DAAD), process on-the-fly the captured network data, which are stored in the appropriate MySQL database (see Table 1 & 2). Thereby, the incoming DNS traffic is classified as suspicious or not and generate the corresponding alert in the case of an undergoing attack. Note, for example, that the second line of Table 2 (response) matches with the first line of Table 1 (request). The architecture employed by the proposed scheme is depicted in Figure 3, while the overall DAAD's detection logic is presented in Figure 4. The interface of the DAAD tool is publicly accessible at: http://f6tmos.samos.aegean.gr/~tmos (username: user & password: kalimera!). All the corresponding source code is also available by the authors upon request.

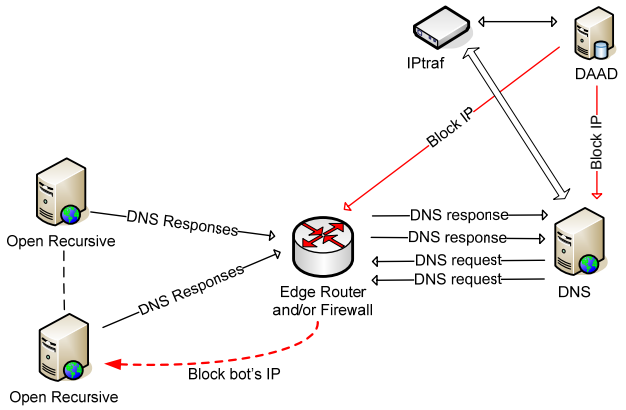

Fig. 3 The proposed DNS Amplification Detection Architecture

Table 1. An Example of the DNS requests Table

Source IP	Source Port	Destination IP	Destination Port
195.251.162.96	32790	195.251.128.5	53
195.251.162.96	32790	194.177.210.210	53
195.251.162.96	32790	194.177.210.210	53
195.251.162.96	32790	195.251.177.9	53
195.251.162.96	32790	192.33.4.12	53
195.251.162.96	32790	192.5.6.32	53
195.251.162.96	32790	192.12.94.32	53

Table 2. An Example of the DNS responses Table

Source IP	Source Port	Destination IP	Destination Port	Status
194.177.210.210	53	195.251.162.96	32790	OK
195.251.128.5	53	195.251.162.96	32790	OK
195.251.177.9	53	195.251.162.96	32790	OK
192.33.4.12	53	195.251.162.96	32790	OK
192.5.6.32	53	195.251.162.96	32790	OK
192.12.94.32	53	195.251.162.96	32790	OK
204.13.161.15	53	195.251.162.96	2481	SUSPICIOUS

In a nutshell, when a DNS message is received the DAAD engine determines whether the message is a response or a request. For any received request or response the DAAD tool creates a new entry to the request / response table (see Tables 1 & 2 accordingly). Once a message is identified as a response the DAAD module checks for the existence of the corresponding request in the queries table by performing an SQL lookup. If the response does not match with none of the requests logged previously in a given timeframe then is marked as suspicious (see the last line of Table 2). Additionally, as soon as the number of suspicious messages exhibits a given administrator-specified threshold an alert is generated and firewall rules are automatically updated to block the attacker's data as depicted in Figure 3. All the parameters in the aforementioned procedure, i.e. timeframe, threshold, can be

dynamically updated and depend on the administrator's security policies in the specific network domain. It should be stated that the proposed solution could be also introduced as part of a statefull firewall. Currently, as mentioned in Section 2.2, statefull firewalls are able to protect DNS only against unauthorized request.

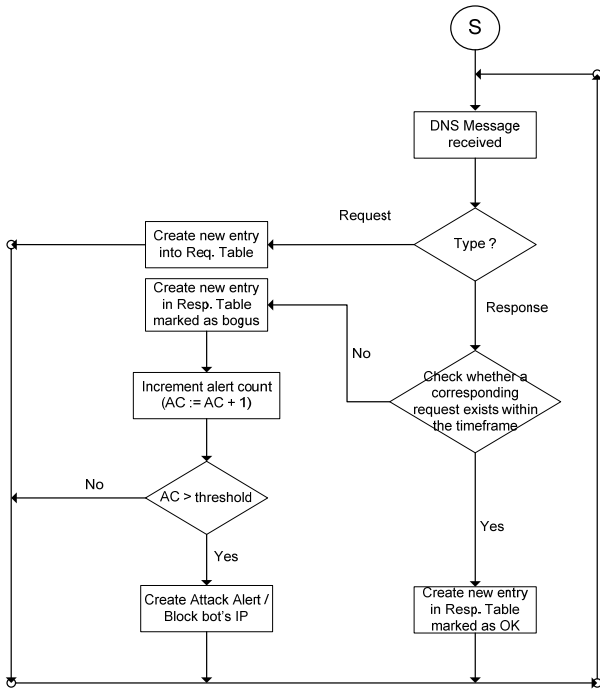

Fig. 4 DAAD's engine detection logic

3.2 Evaluation

In order to evaluate the accuracy of the proposed mechanism we employed the architecture presented in Figure 3. A common desktop machine which incorporates a Pentium IV 2,8GHz processor with 768 MB RAM and 80 GB IDE hard disk was configured to serve as the local DNS server. DAAD was installed in the same machine with the DNS server. Of course, this is the worst case in terms of performance and it is utilized here deliberately. For peak performance DAAD should be placed in a separate independent machine in parallel with the DNS server. Two email servers - which consult 6 black lists of email addresses - and a whole sub-network of our university was instructed to hit this DNS machine. This means that under normal operation the specific machine was processing more than 30,000 DNS queries per hour. It is worth noting that during all experiments no false negative was generated.

As already mentioned, upon receiving a DNS reply the DAAD tool must decide if it is legitimate or suspicious. To do so, DAAD must check against a subset of

previously DNS queries logged into the database. However, frequent SQL lookups substantially affect DAAD's performance. Thus, every incoming DNS reply must be checked not against a big subset of queries, but those issued before a carefully tuned timeframe. DAAD operation showed that the bigger this time-window is, the lesser false alarms are recorded. On the other hand, as already mentioned, increasing this timeframe, DAAD's performance reduces. Moreover, setting this timeframe too high there is a small - and the only - possibility to generate false negatives. For instance, consider the following example when timeframe is set to 30 secs: our DNS server Bob sends a request towards the DNS server Alice at time 00:00. Alice responds to the request by sending a valid reply at time 00:01. Considering the rare case that Alice is also a bot it can bombard Bob with bogus replies for the next 29 secs without being identified by DAAD. Corresponding tests in our network showed that this timeframe becomes optimum when set at 2 seconds.

Every one minute, which is the minimum allowed value[1], DAAD performs a check if there is an undergoing attack by examining the number of suspicious packets logged. As presented in Table 3, which consolidates DAAD operation for a 12 hour time interval (from 08:00 to 20:00), false positives span between 4 and 31 Thus, depending on the network traffic, false alarms can be safely prevented if the number of suspicious replies gathered within this 1 min interval is set between 500 and 1,000. Having this threshold exceeded an alarm is generated.

Table 3. DAAD statistics for a 12 hour interval - no attack occurred (timeframe = 2 seconds, threshold to activate alarm = 500, check for attack every 1 min, flush database check every 1 min if it contains more than 5,000 records)

Time	Requests	Responses	False Positives	Requests delay avg (secs)	Responses delay avg (secs)
08-09	32.819	31.303	20	0.5578	0.5723
09-10	31.655	30.254	18	0.5767	0.5908
10-11	31.965	30.650	4	0.6031	0.6276
11-12	39.260	37.136	28	0.5997	0.6269
12-13	42.852	40.777	20	0.6068	0.6314
13-14	33.383	31.875	9	0.6496	0.6630
14-15	35.346	33.580	9	0.5783	0.6056
15-16	36.108	34.528	31	0.5857	0.6121
16-17	34.424	32.976	6	0.5575	0.5838
17-18	31.281	29.884	11	0.5543	0.5726
18-19	34.776	32.664	6	0.5544	0.5860
19-20	30.133	28.421	4	0.5496	0.5707

Our experiments showed that letting the database to constantly grow it will eventually crash at about 2,500,000 records. Of course, this value is implementation specific, but without dispute we need a policy for flushing periodically the database, especially in case of an attack (see Figure 5). Therefore, every one minute DAAD

[1] As it was placed into the operating system scheduler - the clock daemon in Unix (Cron)

examines the size of the database. If it contains more than 5,000 requests, then DAAD removes all requests that their timeframe is greater than 2 secs. More importantly, the same tactic, i.e. periodically reduce the size of the database, is followed in case of an attack as well (see Figure 5). This happens since smaller database means better performance. It is stressed that this arrangement concerns the DNS requests only, not the replies. In case of an attack the incoming messages (bogus DNS replies) will increase very rapidly but without affecting the overall DAAD performance, since the SQL lookups take into account only the requests. Replies are also removed from the database but far less frequent than requests. Removed data can also be transferred to another database to serve as logfiles at a later time.

2007-6-14 4:5:1 - requests=5036 - responses=4855 - suspicious=0 - Empty Database
2007-6-14 4:9:1 - requests=1257 - responses=2557 - suspicious=921 – **Attack**
2007-6-14 4:10:1 - requests=361 - responses=2322 - suspicious=1223 - **Attack**
2007-6-14 4:11:1 - requests=235 - responses=952 - suspicious=572 - **Attack**
2007-06-14 04:29:01 - requests=5007 - responses=4848 - suspicious=1 - Empty Database
2007-06-14 04:46:02 - requests=5288 - responses=4988 - suspicious=3 - Empty Database
2007-06-14 05:00:02 - requests=5233 - responses=4833 - suspicious=5 - Empty Database
2007-06-14 05:15:01 - requests=5360 - responses=5094 - suspicious=1 - Empty Database
2007-06-14 05:28:02 - requests=5223 - responses=4942 - suspicious=8 - Empty Database

Fig. 5 Snapshot of the logfile that becomes updated when the database flushes

Every record in the database, either DNS request or response, is associated with two distinct times. The first one is the time taken from the iptraf tool. That is the exact time the packet came in to or left the local network. The other one is the time the corresponding record was appended to the database. Subtracting these times we get the overall delay for each MySQL transaction. This time includes processing time and packet characterization time duration (for responses only) as legitimate or suspicious as well. As shown in Table 3 the average delay time for both queries and responses span between 0.5496 and 0.6630 seconds. Naturally, this time greatly depends on the size of the database and the specified timeframe. These times also attest that in average, whether under attack or not, DAAD performs nearly the same.

Another valuable remark is that the number of requests is always greater that the number of responses. Our experiments showed that under normal traffic the total number of responses is about 95% of the issued requests. Having this relation disrupted means that something goes wrong. For example, when self-launching an attack for 5 min duration we recorded 25,606 requests and 68,575 responses. A snapshot of the logfile that is updated every time the database flushes is depicted below.

Last but not least, we present further down DAAD results gathered during a 20 min duration self-attack. According to the attack scenario, the aggressor generates spoofed DNS requests and sends it towards the local DNS server, trying to cause a DoS. The relation between the number of DNS queries and replies - including the number of bogus packets received - is shown in Figure 6. Also, to be able to compare with values presented previously in Table 3 we report hereunder some comparative key metrics in Table 4.

Table 4. Comparative key metrics in seconds for the DAAD tool: Under attack vs. Normal operation

	Requests delay avg	Replies delay avg	Max	Min	St. Deviation requests	St. Deviation replies
Under attack	0.6076	0.6504	0.9870	0.3846	0.1900	0.1028
Normal operation	0.5811	0.6036	0.6630	0.5496	0.0297	0.0292

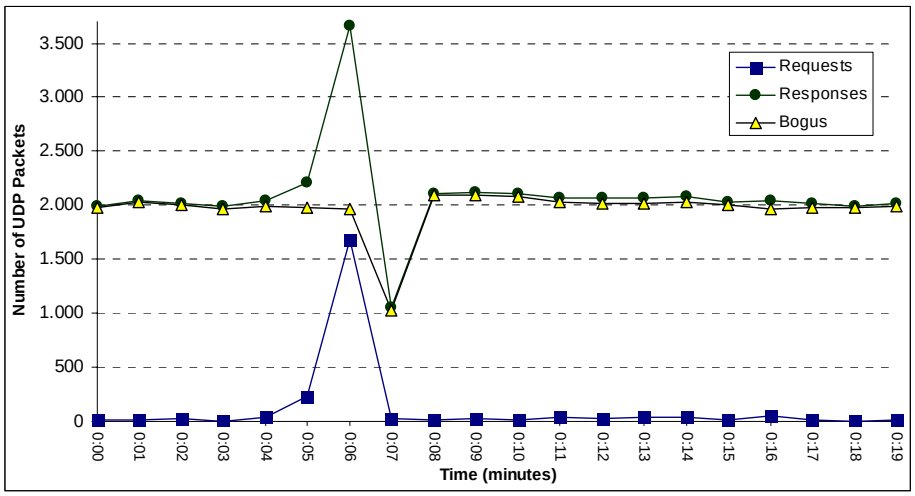

Fig. 6 Relation of DNS requests and responses (including bogus ones) during a 20 min duration self-attack

4 Conclusions and Future work

Name servers can be maliciously used as DDoS attack amplifiers. If this is done on an ongoing basis with a large number of open name servers, it can quickly flood the victim's IP address with responses from thousands (or tens of thousands) of name servers, thereby exhausting the victim's available network bandwidth. The actual target of the attack may be the local DNS server or any host inside the local network. At any rate, the former entity will suffer the consequences of the attack first of any other. Likewise to the Smurf attack, the critical factor here is the amplification effect that is based on the fact that tiny queries can potentially generate much larger UDP packets in response. In this paper several aspects of these attacks were discussed and analyzed. On the top of that, we presented a novel, practical and efficient mechanism, namely DAAD, to defend against them. In its current pilot stage the proposed solution is practical and easy to implement in any network realm. Moreover, test results showed that is effective and can be easily parameterized to fit properly into any network domain. As future work we shall investigate alternative and more efficient data stores like Bloom Filters [21]. This would not only improve the performance of the DAAD tool, but make it scalable as well.

References

1 Cert Advisory CA-1996-26, "Denial of Service Attack via ping", http://www.cert.org/advisories/CA-1996-26.html, Dec. 1997.

2 Gibson, S., "DRDoS Distributed Reflection Denial of Service",http://grc.com/dos/drdos.htm, 2002.

3 Glenn C., Kesidis, G., Brooks, R. R. and Suresh Rai, "Denial-of-Service Attack-Detection Techniques" IEEE Internet computing 2006.

4 Peng, T., Leckie, C. and Kotagiri, R., "Survey of Network-based Defense Mechanisms Countering the DoS and DDoS Problems", to appear in ACM Computing Surveys.

5 Mirkovic, J. et al., Internet Denial of Service: Attack and Defense Mechanism.

6 Security and Stability Advisory Committee, "DNS Distributed Denial of Service (DDoS) Attacks", http://www.icann.org/committees/security/dns-ddos-advisory-31mar06.pdf, March 2006.

7 Mockapetris P., "Domain Names – Concepts and Facilities", RFC 1034, November 1987.

8 Mockapetris P., "Domain Names – Implementation and Specification", RFC 1035, Nov. 1987.

9 Vixie P., "Extension Mechanisms for DNS", RFC 2671, August 1999.

10 Arends, R., Austein, R., Larson, M., Massey, D., Rose, S., "DNS Security Introduction and Requirements", RFC 4033, March 2005.

11 Arends, R., Austein, R., Larson, M., Massey, D., Rose, S., "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.

12 Arends, R., Austein, R., Larson, M., Massey, D., Rose, S., "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.

13 Guo, F., Chen, J., and Chiueh, T., "Spoof Detection for Preventing DoS Attacks against DNS Servers", In Proceedings of the 26th IEEE international Conference on Distributed Computing Systems , July 2006

14 Chandramouli, R. and Rose, S. "An Integrity Verification Scheme for DNS Zone file based on Security Impact Analysis", In Proceedings of the 21st Annual Computer Security Applications Conference, Dec. 2005.

15 Atkins, D., Austein, R., "Threat Analysis of the Domain Name System (DNS)", RFC 3833, Aug. 2004.

16 IPTraf - An IP Network Monitor, http://iptraf.seul.org/.

17 Vaughn, R. and Evron, G., "DNS Amplification Attacks, A preliminary release", March 2006.

18 ICANN Report, "DNS Distributed Denial of Service (DDoS) Attacks", Security and Stability Advisory Committee (SSAC), March 2006.

19 Vixie, P., SAC004, Securing The Edge, http://www.icann.org/committees/security/sac004.txt.

20 Guo, F., Chen, J. and Chiueh, T. "Spoof Detection for Preventing DoS Attacks against DNS Servers," in Proc. of ICDCS 2006.

21 Bloom, B., "Space/time trade-offs in hash coding with allowable errors" Communications of ACM, 13(7), pp. 422-426, July 1970.