Analysis of Flooding DoS Attacks Utilizing DNS Name Error Queries

Zheng Wang
China Organizational Name Administration Center
Beijing 100028 - China
[e-mail: wangzheng@conac.cn]

Received August 8, 2012; revised September 20, 2012; accepted September 28, 2012; published October 29, 2012

Abstract

The Domain Name System (DNS) is a critical Internet infrastructure that provides name to address mapping services. In the past decade, Denial-of-Service (DoS) attacks have targeted the DNS infrastructure and threaten to disrupt this critical service. While the flooding DoS attacks may be alleviated by the DNS caching mechanism, we show in this paper that flooding DoS attacks utilizing name error queries is capable of bypassing the cache of resolvers and thereby impose overwhelming flooding attacks on the name servers. We analyze the impacts of such DoS attacks on both name servers and resolvers, which are further illustrated by May 19 China's DNS Collapse. We also propose the detection and defense approaches for protecting DNS servers from such DoS attacks. In the proposal, the victim zones and attacking clients are detected through monitoring the number of corresponding responses maintained in the negative cache. And the attacking queries can be mitigated by the resolvers with a sample proportion adaptive to the percent of queries for the existent domain names. We assess risks of the DoS attacks by experimental results. Measurements on the request rate of DNS name server show that this kind of attacks poses a substantial threat to the current DNS service.

Keywords: DNS, Denial of service, Name error queries

This work was supported in part by the National Key Technology R&D Program of China under the grant number 2012BAH16B00 and the National Science Foundation for Distinguished Young Scholars of China under the grant number 61003239.

http://dx.doi.org/10.3837/tiis.2012.10.018

1. Introduction

The Domain Name System is a fundamental and indispensable component of the modern Internet [1][2]. In essence, the DNS is a globally distributed and decentralized database of network identities. Its most common use by far is to resolve host names into Internet addresses. Furthermore, a growing number of other applications, such as SMTP, ENUM and SIP also depend on the DNS in order to route messages through appropriate application level gateways [3]. As a result, the availability of the DNS can affect the availability of a large number of Internet applications. Ensuring the DNS service availability is essential for maintaining the robust and resilience Internet service.

A Denial-of-Service (DoS) attack is an attack in which one or more machines target a victim and attempt to prevent the victim from doing useful work. Almost all Internet services are vulnerable to denial-of-service attacks of sufficient scale. In most cases, sufficient scale can be achieved by compromising enough end-hosts (typically using a virus or worm) or routers, and using those compromised hosts to perpetrate the attack. Such an attack is known as a Distributed Denial-of-Service (DDoS) attack. However, there are also many cases where a single well-connected end-system can perpetrate a successful DoS attack.

Due to its hierarchical structure, the DNS availability depends on a small number of servers that serve the root and other important top level domains. A number of DDoS attacks have been directed against these top level DNS name-servers in recent years [4][5][6][7]. The impact on overall DNS availability is debatable [8][9], but some attacks did succeed in disabling the targeted DNS servers and resulted in parts of the Internet experiencing severe name resolution problems. This demonstrates the potential of attacks to threaten the DNS availability as well as the availability of the Internet.

As a global service, DNS must be highly scalable and offer good performance under high load. In particular, the system must operate efficiently to provide low latency responses to users while minimizing the use of wide-area network resources. The aggressive use of caching is widely believed to contribute much to the scalability of DNS [19]. Besides, caching also mitigates the DoS attacks against the name servers launched from the clients due to the filtering effect of cache for the flooding DNS queries.

We propose in this paper that the DoS attacks utilizing name error query can effectively bypass the cache thus exert the whole query load on the name server. We also analyze the capability and impact of such DoS attacks, and develop a countermeasure that can effectively detect and enhance the DNS resilience against such attacks.

2. Related Work

In the past decade there has been some literature for hardening the DNS against DoS or DDoS attacks. Handley et al [10] have proposed to build a global peer-to-peer replication infrastructure on the records at every caching server. Yang et al [11] have proposed to augment the original DNS hierarchical structure with hierarchical overlay networks. When certain nodes are under DoS attacks, user queries are routed across the overlays to bypass the failed nodes and reach the destination. While both proposals achieve the mitigation of DoS attacks in terms of the DNS hierarchical structure as a whole, they hardly deal with the single point defending problem. Parka et al [12] have developed a lightweight peer-to-peer name lookup service using caching servers as peers at remote sites to provide cooperative lookups during

failures. The method cannot increase the DNS resilience against DDoS attack that target name servers. It also has weakened practicality due to its impractical assumption that caching server operators are cooperative. Ballani et al [13] have proposed to utilize expired records to enhance the DNS availability and resilience. Caching servers can leverage the expired records for DNS resolution rather than solicit the response from the authoritative name servers. However, the DNS DoS attacks proposed in the paper makes caching ineffective thus invalidate the approach.

Apart from hardening the current DNS there has been some efforts on redesigning the DNS. Cox et al [14] have proposed a distributed hash table based peer-to-peer infrastructure to provide DNS service. This design lessens the impacts of DDoS targeting a specific server thanks to the inherent distributed nature of peer-to-peer systems. But this work also shows the disadvantage of this peer-to-peer design compared with the current DNS in terms of query response time. Driven by the similar idea, Ramasubramanian et al [15] redesign the DNS by replicating the most popular records across the peer-to-peer system to improve the performance of the lookup service. Some hold different opinions on the direction of DNS redesign, e.g., Deegan et al [16] considered a number of common problems with the DNS, and some of the possible solutions. The study proposed to remove the unnecessary and inefficient distribution and recentralize the publication of data in the DNS. All these redesigning approaches call for a complete overhaul of the DNS structure. But such radical changes require a much more detailed specification of the requirements than is currently available and may hinder their adoption.

For the DNS performance enhancement, Kangasharju et al [17] have proposed a new DNS design consisting of geographically distributed servers, each of which having a complete and up-to-date copy of the entire DNS database. Cohen et al [18] proposed the use of proactive caching to reduce the DNS response delays. Pappas et al [24] proposed to reduce the impact of DDoS attacks against top level DNS servers by setting long TTL values for NS records and their associated A and AAAA records. To protect against the DNS poisoning attacks, Manos et al [25] proposed Anax, a DNS protection system that detects poisoned records in cache. The system can observe changes in cached DNS records, and applies machine learning to classify these updates as malicious or benign. Yan et al [26] proposed a new type of unidirectional flow called IF flow. They also introduced two efficient methods in their DDoS attacks detection and evaluation scheme. While these proposals can potentially improve the resilience and stability of the DNS against DDoS attacks, they do not address the specific attacks proposed in this paper.

3. Name Resolution Procedure

Clients rely on DNS primarily to map service names to the IP addresses of the corresponding servers. Typically, clients issue their queries to a local resolver (recursive server). The local resolver then maps each query to a matching resource record set (hereon simply referred to as a matching record) and returns it in the response. Each record is associated with a time-to-live (TTL) value and resolvers are allowed to cache a record till its TTL expires; beyond this, the record is evicted from the cache. Given a query to resolve, a resolver executes the following actions:

1) Look up the cache for a matching record. If a matching record is found, it is returned as the response.

2) If a matching record is not found in the cache, the resolver uses the DNS resolution process to obtain a matching record. This involves:

(a) Determine the closest zone that encloses the query and has its information cached (if no such zone is cached, the enclosing zone is the root zone and the resolver resorts to contacting the DNS root-servers). For example, given an A record query for the name www.conac.cn, the resolver determines if records regarding the authoritative name servers for the zones .conac.cn, or .cn (in that order) are present in its cache.

(b) Starting from the closest enclosing zone, traverse down the DNS zone hierarchy by querying subsequent sub-zones. The procedure lasts until the zone responsible for authoritatively answering the original query is reached or an error response from a zone's name servers implies that the traversal cannot proceed. In either case, the resolver returns the appropriate response to the client. Also, all responses (including negative responses indicating error) during this resolution process are cached by the resolver. **Fig. 1** illustrates this scenario.

4. Flooding DNS DoS Attacks Utilizing Name Error Queries

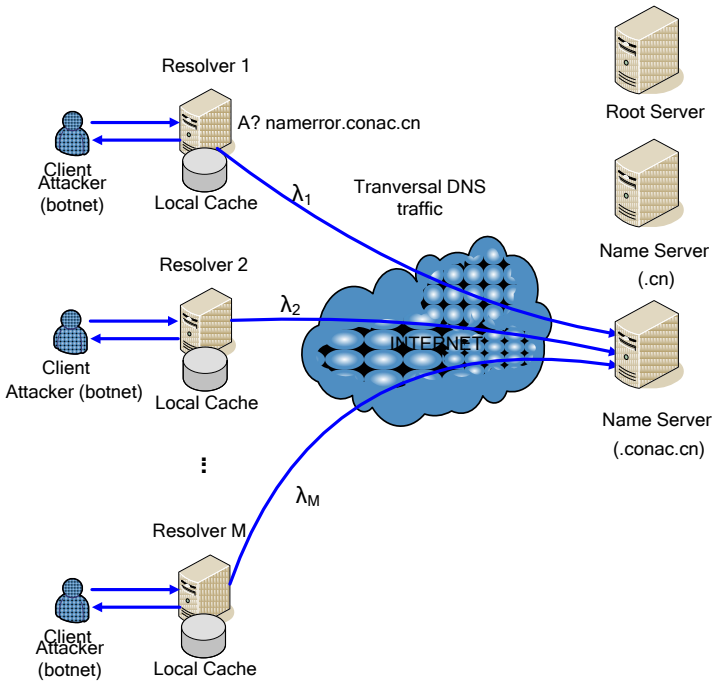

Fig. 1. DNS Name Resolution Procedure

We consider DoS attacks on DNS servers. Attackers flood the name servers of a zone to disrupt the resolution of records belonging to the zone. Consequently, any of its sub-zones also collapse when a zone breaks down. Thanks to the functioning of the local resolver and its caching mechanism, DoS traffic generated by the attacker may be largely blocked by the local cache. This ensures that DoS traffic cannot traverse the resolver to reach the target name server. For example, when the attacker sends an A record query for the name www.conac.cn, the resolver can fetch it from the name server at the first time. So the afterwards queries for the same record can hit the cache until the TTL expires. Generally, we have the following theorem to express the maximum existent name query rate generated from one resolver.

Theorem 1: Let the zone of attacker's target name server has a total of N resource records sets RR_1, RR_2, …, RR_N., and their TTLs are T_1, T_2,…, T_N respectively. The maximum existent name query rate generated from one resolver is

$$q_{\max}^{v} = \sum_{i=1}^{N} \frac{1}{T_i} \tag{1}$$

Proof: For each existent resource records set RR_i, the cached data exists for the time span of T following one response from the target name server, so the maximum query rate for it is $\frac{1}{T_i}$.Summation of the maximum query rate for each existent resource records set is expressed in Equation (1).

Normally, the zone size N is less than 10000, and the TTL ranges from several seconds to a couple of days or mouths. Assume the zone size is 10000 and the average TTL is one hour, then $q_{\max}^{v} = \frac{10000}{3600} < 3qps$. This order of query load is far below the peak capability of most DNS name servers. In the worse case, the attacker may compromise more than one clients (e.g. botnet [20]) served by different resolvers to launch the DoS attack. In this way, the accumulated query rate can be multiplied by the number of the involved resolvers. Despite all that, the overall query rate keeps within the resolution capability of ordinary name servers.

The quey rate is largely bounded by the number of names available for the attacks. Theoretically, query for nonexistent name has no limitaions on the number of names. This significantly changes its pattern of query rate compared with the query for existent name. For each query for nonexistent name, the name server responses with SOA records and name error RCODE. The negative caching retains the negative response for the queried name according to the minimum TTL set by the SOA record, thus all negative caching for one zone's data uses the same TTL. Since the nonexistent names that can be utilized to attack the target zone can be fabricated as many as the attacker like, the theoretical query rate can approximate infinity. Let the TTL of negative caching is T_N and the number of available nonexistent names is N_N, the maximum query rate $q_{\max}^{e}$ can be expressed as follows

$$q_{\max}^{e} = \frac{N_N}{T_N} \tag{2}$$

So we have

$$q_{\max}^{e} \to \infty, \text{ if } N_N \to \infty \tag{3}$$

The only boundary of the maximum query rate of one resolver is the query rate accumulated by all clients of it. If the number of compromised clients by the attacker within one resolver is N_c, and their maximum query rate are $q_{\max}^{1}, q_{\max}^{2}, \ldots, q_{\max}^{N_c}$. Then the maximum query rate from the resolver $q_{\max}^{e,1}$ can be expressed as follows

$$q_{\max}^{e,1} = \sum_{i=1}^{N_c} q_{\max}^{i} \tag{4}$$

From the Equation (4), we can see that the name error DoS attack is a real potential threat to the name server. Moreover, this threat can be aggravated through utilizing clients belonging to more than one resolver. Under this name error DoS attack, the resources of a name server can be exhausted and thereby prevented from resolving queries, both for the existent names and for the nonexistent names. One major indicator of name server overload is the response delay,

and this makes the resolver maintain large amount of parallel sessions waiting for the response until time out.

Assume the time out is T_{to}, the mean query rate of one resolver is $q_{\max}^{e}$, the service time distribution at the target name server is arbitrary with mean $\bar{x}_i$ and the mean service rate of server is $\mu_i = 1/\bar{x}_i$. We can derive the amount of parallel sessions maintained in the resolver.

The time for which the *k*th query for nonexistent name has to wait is

$$t_k = (k+1)\bar{x}_i - k/q_{\max}^{e} \tag{5}$$

When *k*th query is longer than T_{to} or time out, then it is removed from the waiting sessions. Then

$$t_k > T_{to} \tag{6}$$

Thus the first time out query is

$$k^{*} = \left\lceil \frac{T_{to} - \bar{x}_i}{\bar{x}_i - 1/q_{\max}^{e}} \right\rceil \tag{7}$$

Where $\lceil \square \rceil$ is the upper integer function.

The number of queries getting response is

$$k^{'} = \left\lfloor \frac{k^{*}}{q_{\max}^{e} * \bar{x}_i} \right\rfloor \tag{8}$$

Where $\lfloor \square \rfloor$ is the lower integer function.

The number of sessions maintained in the resolver is

$$k^{''} = k^{*} - k^{'} \tag{9}$$

We assume that the arrival process of queries to the name server is an independent Poisson process with rate λ and the server implements a Processor Sharing (PS) scheduling policy, where all the requests share the server's capacity equally and continuously [1]. Under the above assumptions, the name server behaves as an-PS queue and the average delay of request is therefore given by the following expression [21]

$$\bar{T}_i = \frac{1}{\mu_i - \lambda} \tag{10}$$

For the case of DoS attack, nonexistent name queries can occupy the majority of total query rate. Thus (for attacks compromising more than one resolver)

$$q_{\max}^{e,1} \to \lambda \tag{11}$$

Substitute Equation (11) into (10), then $\bar{T}_i$ in Equation (10) into $\bar{x}_i$ in Equation (7), (8) and (9), we can get the more accurate expression of the number of sessions calculated as Equation (9).

We can see that in the scenario of the DoS attacks utilizing name error query, not only the name server suffers from the resource exhaustion, but also the local resolvers can break down due to the lowered resolution service of name server.

5. One Example: May 19 China's DNS Collapse

The DoS attack utilizing name error query described in Section 3 has been perhaps largely employed to flood DNS servers by attackers. However, the most recent reported one gluing our attention may be May 19 China's DNS Collapse.

Starting from 21:50 on May 19, 2009, Internet users in Jiangsu, Anhui, Guangxi, Henan, Gansu, and Zhejiang, reported that they suffered slow Internet speeds or were unable to visit some websites. The Internet in these provinces were recovered after the rescure efforts such as resolver cache flushing. According to China Telecom, the network failure was led by the domain name system failure of Baofeng.com, the website of the Chinese music player provider, and the failure further caused the surge of DNS server visits and the decrease of processing performance of the network.

A more detailed report by Liu from China Telecom [22], one of the major ISPs of china, presents a first sight into who suffers what, and thus provides us some tips for the cause of this collapse. It is reported that Internet users got few responses from their local DNS servers (largely ISP server) and Operators observed their boxes (DNS servers, Layer-4 switches, …) overloaded. More close observations (as illustrated in **Fig. 2**) include:

1) baofeng.com became the hottest domain.
2) Servfail messages showed up when trying nslookup baofeng.com names.
3) The recursive session number reached the limit.

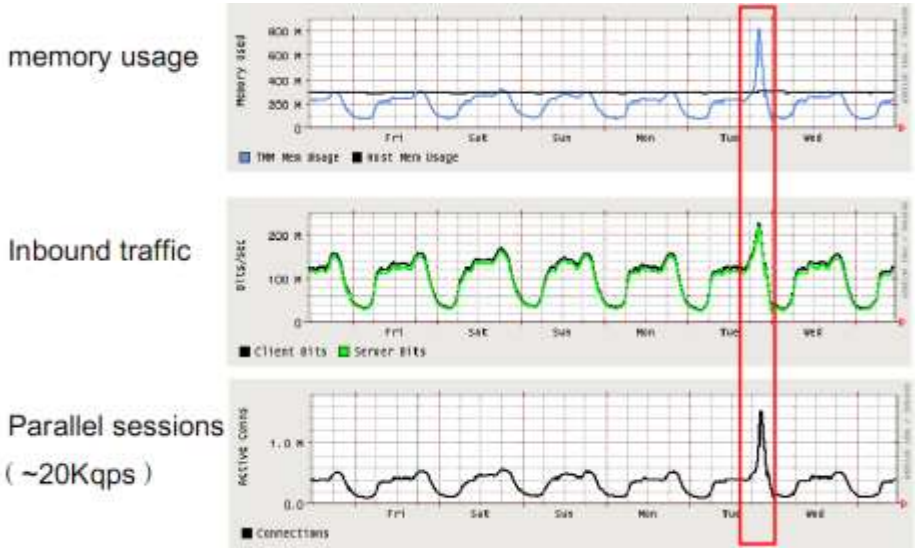

Fig. 2. DNS Node Observations

As seen from these clues, the root cause of this collapse can be traced back to the DoS attacks utilizing name error query. The victim name server is the server of baofeng.com zone. It was flooded by name error queries from clients within several resolvers. The number of recursive session was increased especially for those resolvers whose clients are the source of the attacks.

6. Detection of the Attacks

The detection approach is based on the monitoring of the negative responses in the cache.

Despite that comparing the incoming DNS query traffic from the clients and the DNS responses from the name servers is also applicable for the detection purpose, we believe that the most appropriate method should have minimal effects on the normal DNS resolution service. Periodical counting of negative cached responses need no extra data structures, nor incurs additional procedures for the processing of DNS packets. So we adopt the approach of negative caching processing for the sake of computational efficieny.

At the first step, we want to find the most popular zone flooded by a large amount of name error queries. To do this, the count of name error answers in the cache is calculated grouped by zone name. This count indicates the number of name error queries for a zone in the recent TTL time span. In the case of attacks, this count should be highly skewed for different zones (or name servers). The attack is unlikely to appear in a large number of zones, because the query load would thereafter be distributed among them, which mitigates the severity of attacks on a particular zone. So we only need to label the top-K most requested zones as he possible victim zones. We can assume a threshold value T_z to determine the zones under attack. If the query count for a zone is larger than T_z, the zone is identified as the victim zone. Long time learning is required to obtain the normal rate level of name error queries, based on which T_z is set. The appropriate value of T_z is also dependent of the time interval over which statistics and analysis are done. Let the time interval be T. Small T reflects transient and accuate traffic fluctuations, which makes it advantageous for a sensitive detection system. However, the cost of small T lies in the huge resource comsuption incured by computational pressure of a short interval. By comparion, large T cuts the cost of resources by a prolonged processing interval while the detection granularity becomes coarser. Here an appropriate value of T that we recommend may be the TTL of negative caching. This is a tradeoff between the processing complexity and the detection granularity . Furthermore, the TTL of negative caching is the largest value for T that ganrantees that there is no expiration of cached negative answers in the time interval of detection processing. Otherwise, more frequent name error queries experiencing cache expiration more than once amid T is unreasonably deems as the same traffic volume as those retaining in the cache during T.

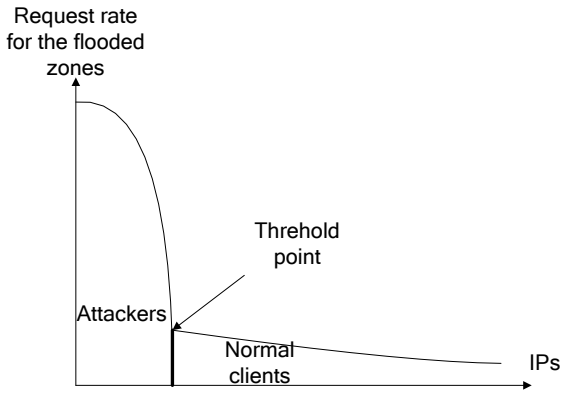

Fig. 3. Sorted Query Rate Graph for the Flooded Zones

When the zones under attack are identified from the candidate top-K zones, we then need to detect those clients that launch the flooding attacks to the zones. Similar to the processing of victim zones, the name error answers from the flooded zone in the cache are grouped by client IPs and counted. The resulting query count distribution differs from the victim zones in that client IP space is usually much larger than the victim zone space. Generally, there should be a

gap between the name error query rate of the normal clients and the malicious ones. So the threshold point is the first point in the first "valley" of the sorted query rate graph (as illustrated in **Fig. 3**). However, it is still hard to predict the range of number of malicious clients. Here we use a step method to find the threshold. If the threshold misses the current top-K client list, increase K by a step of K_i. This procedure progresses until the appropriate threshold is found. The empirical initial value may be around 100.

Let the sorted query rate for all N IPs is q_1, q_2, …, q_N, and $q_1 > q_2$, …, $> q_N$. The definition of "valley" can be formulated as follows:

Definition: Clients 1, 2, …, i are the attackers if i is the first point in the first "valley" of the sorted query rate graph or the minimal number that satisfies

$$\frac{q_{i-1} - q_i}{q_i - q_{i+1}} > F_{th} \tag{12}$$

Where F_{th} is the threshold whose empirical value can be 10. The flowchart of the attack detection procedure is shown in **Fig. 4**.

7. Defense against the Attacks

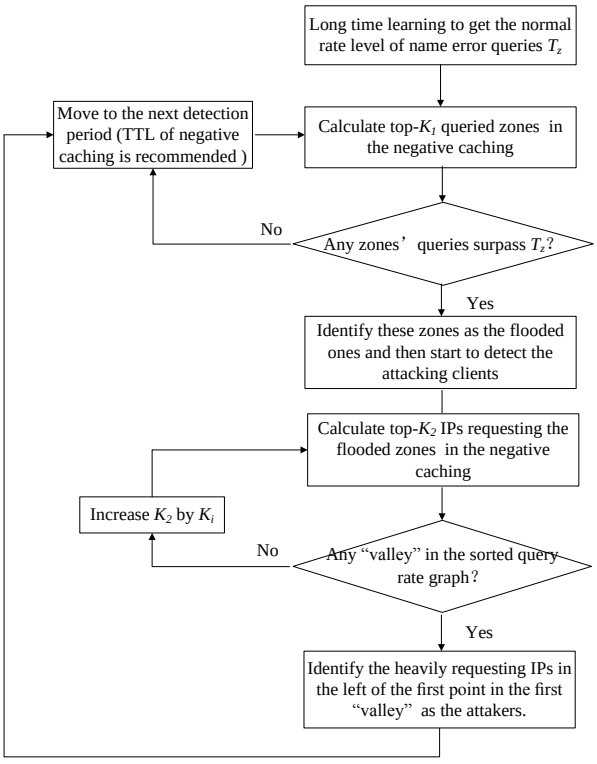

Fig. 4. Attack Detection Procedure

In the DoS attack detection process, malicious clients and their target zone are identified. This makes it possible for the defense against the attacks. Filtering out queries for the target zone from the malicious clients is perhaps the simplest solution, but not the most appropriate one. Due to the dynamic and variation nature of the attack behaivor, some clients deemed maclious in the previous round of detection may return to the normal behavior during the current round

of detection. But the simple filtering approach would block all queries from the malicious clients for the target zone inclusive of the existent namesfor *T*. To react to the attack mitigation from the clients timely, we trace the status of the malicious clients and take measures correspondingly. We use a sample technique to selectively offer resolution service to some queries issued by the detected malicious clients. The current name error proportion can be estimated by minitoring the response to the queries. The sample proportion is adjusted and renewed according to the previously estimated name error proportion. In the close-loop control process, more percentage of queries for existent names in this roundmeans more sample proportion in the next round. In extreme cases, the sample proportion reaches 100% if the percentage of queries for existent names is high enough.

Let the minimum percentage of queries for existent names when offer complete service (or the sample proportion is 100%) be p_h and the minimum sample proportion be S_{min}. And when the percentage for existent names falls below p_l, we use the minimum sample proportion. This can be expressed as

$$S(t+1)=\begin{cases} S_{\min} & p(t)<p_l \\ f(p(t)) & p_l<p(t)<p_h \\ 1 & p(t)>p_h \end{cases} \tag{13}$$

For percentages between p_l and p_h, the sample proportion for the next round is set according to the current percentage. The relationship between sample proportion and percentage for existent names should at least follow the monotone increasing law. For example, we can adopt a simple linear relationship (blue line in **Fig. 5**). So the sample function $f(p(t))$ in Equation (13) is

$$f(p(t))=S_{\min}+\frac{p(t)-p_l}{p_h-p_l}*(1-S_{\min}) \qquad p_l<p(t)<p_h \tag{14}$$

Another possible relationship can be convex to the below in order to punish the low section of percentage for existent names (red line in **Fig. 5**).

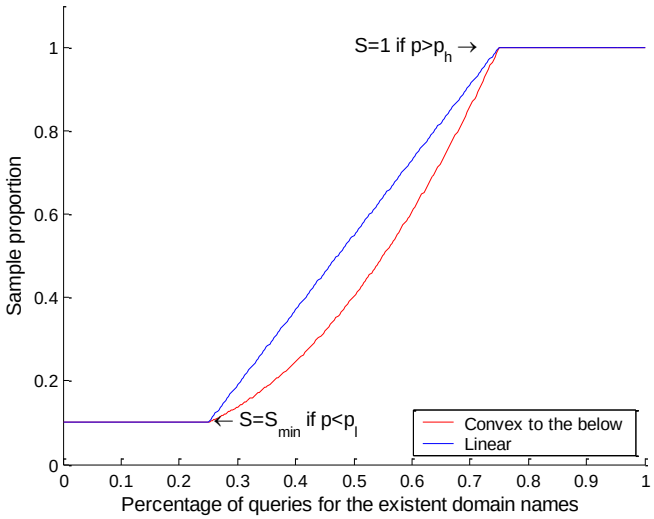

Fig. 5. Sample Function

This adaptive counter measure provides opportunities for the detected attacking clients to regain service before the next round of detection process. The flowchart of the attack defense procedure is shown in **Fig. 6**.

8. Experimental Results

We set up a testbed to evaluate the effects of DNS DoS attack using the name error query. The testbed consists of three nodes: one DNS resolver, one DNS client, and one authoritative name server.

In the tests, the client and the authoritative name server are on our local campus network. Since the DNS resolver service provided by some ISPs may return a redirect result to the name error query, we use Google public DNS resolver which is claimed to return the resolution results with absolutely no redirection.

The zone administrated by the authoritative name server is delegated from a subdomain of our organization. It is temporarily created and only used when testing the DoS attack and is not accessed by other clients.

The authoritative name server run BIND (version 9.3.1) and the DNS requests are generated through the DIG tool of BIND [23]. We rely on the client to send the DNS requests for error names of the zone, which are then forwarded to the name server monitored on our local network. From the log file of the name server, we can see the status of the requests arriving at it. The SOA minimum TTL value is set as 10 seconds.

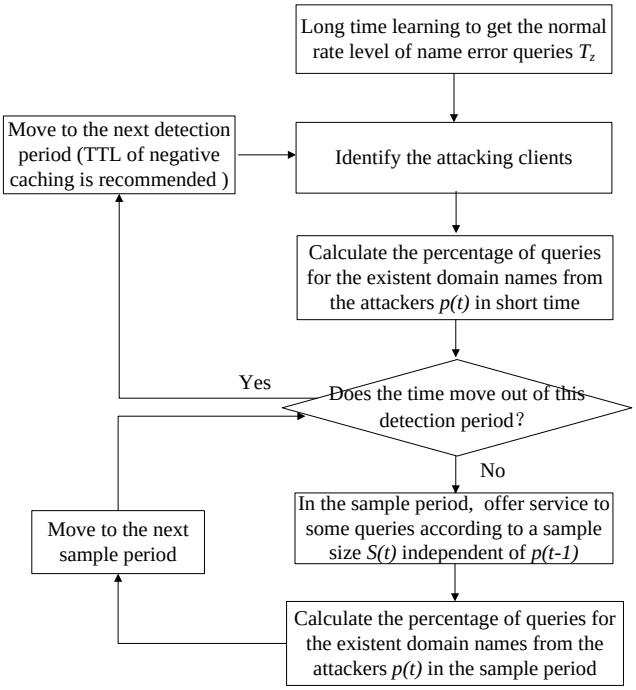

Fig. 6. Attack Defense Procedure

We first evaluate the attacks by DNS requests using one name error query. In this test, the client sends only one name error query for 10,000 times at a near constant rate. If the negative answer is cached until the TTL expires, the arrival interval of queries should approximate TTL

plus a small residual time. The time intervals of all requests are shown in **Fig. 7**. We note that the minimum value is 10.09, which is quite close to the value of TTL. So it can be concluded that the resolver caches the negative answer exactly according to the SOA minimum TTL value. And this is the prerequisite for our further experiments.

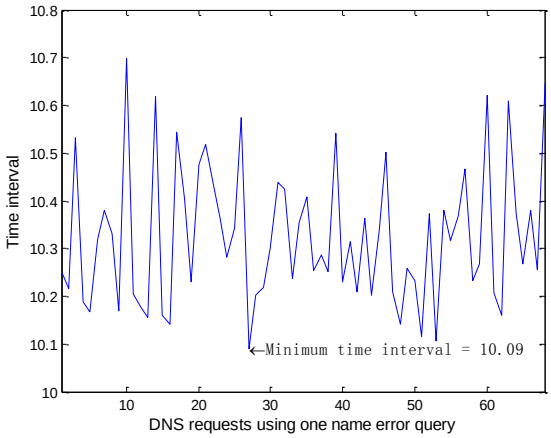

Fig. 7. Time Intervals for DNS Requests Using One Name Error Queries

We then measure the effects when DNS requests use different name error queries. A list of error names is requested in a cycle sequence. We have the name space large enough to ensure that the time taken by requesting all error names sequentially in the list exceed the TTL. So all queries miss the negative cache and therefore reach the name server. Then we can estimate the arrival/sending interval of the name error queries. The request rate averaged in 10 seconds and the cumulative distribution of intervals are shown in **Fig. 8** and **Fig. 9** respectively. **Fig. 8** tells us that the average number of requests in a TTL is about 60.4. Thus as indicated in Section 3, the minimum number of name error queries is 61.

We then use DNS requests for the minimum number of error names. When the size of error name list is 61, all queries are also expected to miss in the cache. So the request rate should match the value measured in the previous experiment. **Fig. 10** and **Fig. 11** illustrate the request rate averaged in 10 seconds and the cumulative distribution of intervals respectively. We can see that both the request rate and the cumulative distribution of intervals agree well with the previous measurements, which validates our analysis in Section 3.

9. Conclusions and Future Work

This paper explored the DNS DoS attacks utilizing name error query. Our analysis, example, and experimental results show that this kind of attacks is the potential threats to the current DNS service. We also proposed some simple approaches to effectively detect and defend such attacks.

To more comprehensively quantify the impact of such attacks using real-world traces, we are in the process of obtaining traces collected at DNS resolvers of large ISPs. And we aim to determine the impact of different aspects of these attacks. For example, assuming an attack that takes down the DNS root-servers, how many queries for a typical resolver will fail?and how many will benefit from having the proposed detection and defense measures in place? We also aim to implement this detection and defense approach into common DNS resolvers (for

example, BIND, DBJDNS, etc.).

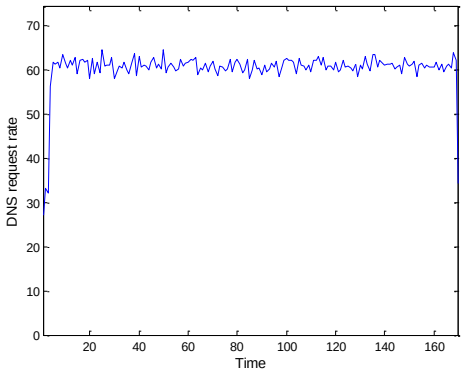

Fig. 8. Request Rate

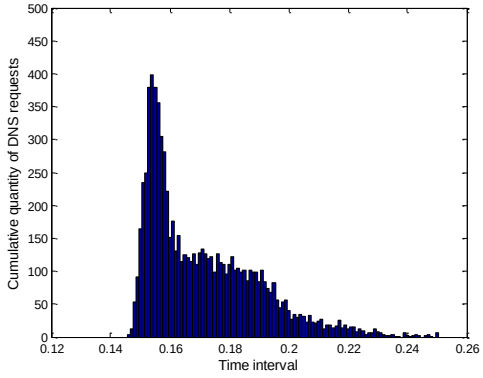

Fig. 9. Cumulative Distribution of Intervals

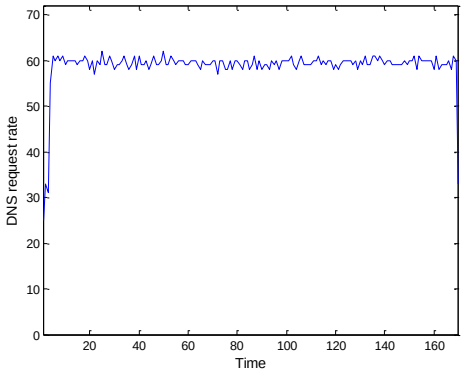

Fig. 10. Request Rate

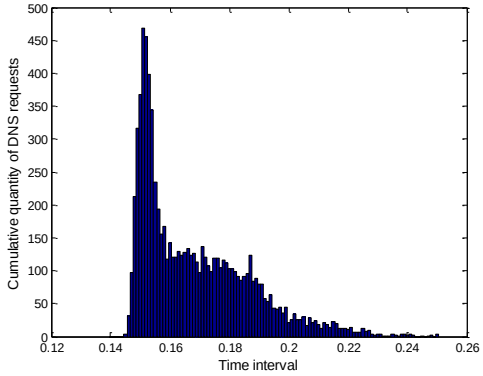

Fig. 11. Cumulative Distribution of Intervals

References

[1] P. Mockapetris, Domain names – concepts and facilities, *Internet Request for Comments* (*RFC 1034*), November 1987.
[2] P. Albitz and C. Liu, *DNS and BIND*, O'Reilly and Associates, 1998.
[3] H. Rood, "What is in a name, what is in a number: some characteristics of identifiers on electronic networks", *Telecommunications Policy*, vol.24, pp.533-552, 2000. Article (CrossRef Link)
[4] Name server DoS Attack October 2002, *http://www.caida.org/projects/dns-analysis/*, 2002. (last retrieved in October 2011)
[5] UltraDNS DOS Attack, *http://www.theregister.co.uk/2002/12/14/*, 2002. (last retrieved in September 2012)
[6] DoS Attack against Akamai, *http://news.com.com/2100-1038_3-5236403.html/*, 2004. (last retrieved in October 2011)
[7] ICANN Factsheet for the February 6, 2007 Root Server Attack. *http://www.icann.org/announcements/ factsheet-dns-attack-08mar07.pdf*, 2007. (last retrieved in October 2011)
[8] Events of 21-Oct-2002, *http://d.root-servers.org/october21.txt*, 2002.
[9] DNS FAQ, *http://www.cs.cornell.edu/People/egs/beehive/faq.html*, 2004. (last retrieved in

September 2012)
[10] M. Handley and A. Greenhalgh, "The Case for Pushing DNS", In *Proc. of the 4th ACM Workshop on Hot Topics in Networks (Hotnets)*, 2005.
[11] H. Yang, H. Luo, Y. Yang, S. Lu, and L. Zhang, "HOURS: Achieving DoS Resilience in an Open Service Hierarchy", In *Proc. of the 2004 International Conference on Dependable Systems and Networks Proceedings (DSN)*, pp. 83-93, 2004.
[12] K. Parka, V. Pai, L. Peterson, and Z. Wang," CoDNS: Improving DNS Performance and Reliability via Cooperative Lookups", In *Proc. of the 6th conference on Symposium on Operating Systems Design & Implementation*, pp. 14-29, 2004.
[13] H. Ballani and P. Francis, "A Simple Approach to DNS DoS Defense". In *Proc. of the 5th ACM Workshop on Hot Topics in Networks*, pp. 67-72, 2006.
[14] R. Cox, A. Muthitacharoen, and R. Morris, "Serving DNS Using a Peer-to-Peer Lookup Service", In *Proc. of the 1st International Workshop on Peer-to-Peer Systems* , pp. 155-165, 2002.
[15] V. Ramasubramanian and E. Sirer, "The Design and Implementation of a Next Generation Name Service for the Internet," In *Proc. of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications,* pp. 331-342, 2004. Article (CrossRef Link)
[16] T. Deegan, J. Crowcroft, and A. Warfield, T"he Main Name System: An Exercise in Centralized Computing," *ACM SIGCOMM Computer Communication Review*, vol.35, no.5, pp.5-14, 2005. Article (CrossRef Link)
[17] J. Kangasharju and K. Ross, "A Replicated Architecture for the Domain Name System", In *Proc. of the 19th Annual Joint Conference of the IEEE Computer and Communications*, pp. 660-669, 2000.
[18] E. Cohen and H. Kaplan, "Proactive Caching of DNS Records: Addressing a Performance Bottleneck," In *Proc. of the 1st IEEE/IPSJ International Symposium on Applications and the Internet (SAINT)*, pp. 85-94, 2001.
[19] J. Jung, E. Sit, H. Balakrishnan, and R. Morris, "DNS performance and the effectiveness of caching," *IEEE/ACM Transactions on Networking*, vol.10, no.5, pp.589-603, 2002. Article (CrossRef Link)
[20] Million-PC botnet threatens consumers. *http://www.infomaticsonline.co.uk/vnunet/news/2167474/million-pc-botnet-threatens*, 2006. (last retrieved in February 2012)
[21] L. Kleinrock, *Queueing Systems*, vol.2, Wiley-Interscience, 1976.
[22] Ziqian Liu, Lessons learned from May 19 China's DNS collapse, Presentation at the 2nd DNS-OARC Workshop. Beijing, China, Nov. 2009, *https://www.dns-oarc.net/files/workshop-200911/Ziqian_Liu.pdf*. (last retrieved in September 2012)
[23] Bind website, *http://www.isc.org/products/BIND/*.(last retrieved in September 2012)
[24] V. Pappas and E. Osterweil. Improving "DNS Service Availability by Using Long TTL Values." *IETF Internet-Draft* (*draft-pappas-dnsop-long-ttl-04*), 2012.
[25] A. Manos, D. David, L. Xiapu, P. Roberto, L. Wenke and B. Justin. "A centralized monitoring infrastructure for improving DNS security". In *Proc. of the 13th International Conference on Recent Advances in Intrusion Detection*, pp. 18-37, 2010.
[26] Ruoyu Yan, Qinghua Zheng and Haifei Li. "Combining Adaptive Filtering and IF Flows to Detect DDoS Attacks within a Router". *KSII Transactions on Internet and Information System*, vol.4, no.3, pp. 428-451, 2010.

Zheng Wang received the MS degree in Electrical Engineering from Institute of Acoustics, Chinese Academy of Sciences in 2006, and the PhD degree in Computer Science from Computer Network Information Center, Chinese Academy of Sciences in 2010. He is currently the director of Joint Labs in China Organizational Name Administration Center. His research interests are in the areas of network architecture, Domain Name System, and information systems.