社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2003－ＱＡＩ－8 (9)
2003／7／7

DoS 攻撃に対する DNS サーバの資源消費解析

力武 健次 *★ 菅谷 史昭 ‡ 中尾 康二 ‡ 野川 裕記 ★ 下條 真司 ★

DNS（ドメイン名システム）は恒常的に DoS（サービス拒否）攻撃の対象になっている。DNS はデータベースの検索と結果のやり取りに UDP を使っているため、DNS サーバを UDP パケット生成を繰り返して狙うことは容易にできる。本論文では、模擬 DoS 攻撃に対する DNS サーバプログラムの計算量的資源消費解析を行い、異なる条件下でサーバや OS がどのような挙動を示すかを評価する。

Resource Consumption Analysis of DNS Servers against DoS attacks

Kenji Rikitake‡*, Fumiaki Sugaya‡, Koji Nakao‡, Hiroki Nogawa★ and Shinji Shimojo★

DNS (Domain Name System) servers have persistently been a target of DoS (Denial-of-Service) attacks. Since DNS uses UDP for the database lookup and result exchange, targeting DNS servers by repetitive UDP packet generation is highly feasible. In this paper, we perform a computational resource consumption analysis of DNS server software programs against simulated DoS attacks, and evaluate how the server and the operating system behave under the different types of conditions.

Keywords: Internet, DNS, Security, System Performance, Denial-of-Service Attacks

1 Introduction

DNS (Domain Name System) [1, 2] is one of the most critical subsystems of the IP (Internet Protocol) Suite. DNS defines the mapping between the domain names and various resources including IP addresses and MX (Mail eXchanger) host names. Looking up the DNS is mandatory for a Web browser to find out the IP address of the server. Mail transfer agents find out the destination agent by looking up the DNS for the MX RRs (Resource Records) of the destination domain.

* 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University
rikitake@ist.osaka-u.ac.jp
‡ (株) KDDI 研究所
KDDI R&D Laboratories, Inc.
{kenji,fsugaya,nakao}@kddilabs.jp
* 大阪大学サイバーメディアセンター
Cybermedia Center, Osaka University
{nogawa,shimojo}@cmc.osaka-u.ac.jp

DNS database lookup is mostly performed over UDP (User Datagram Protocol) [3]. The UDP is chosen since in most cases each query request and reply is fit into a single packet since the size of exchanged data between DNS servers and clients is small (≤ 512 bytes), and to keep the server overhead low.

Most DNS servers also support the lookup over TCP (Transmission Control Protocol) [4]. On a DNS lookup, however, TCP is currently only used for exchanging larger data exceeding 512 bytes, since it has the overhead of making and breaking the connections and maintaining the state inside each host.

The DNS dependency of UDP causes an administrative issue when to manage a globally-accessible Internet system, such as a gateway exposed to the global Internet, since the system is prone to UDP-based DoS attacks targeted at the DNS server. When a host runs a DNS server for providing publicly-accessible domain data, the

```
/* the UDP socket descriptor */
int s;
/* number of packets
   between usleep()s */
int num;
/* UDP payload data content */
char *p;
/* UDP payload data length */
int size;

 for (;;) {
  /* a small-time pause for
     the packet rate control,
     not required for flooding */
  usleep(1);
  /* looping for DoS */
  for (i = 0; i < num; i++) {
   /* sending a UDP packet */
   send(s, p, size, 0);
  }
 }
```

Fig. 1 A Simple C Code of UDP DoS

Host name	CPU type	Clock speed (MHz)	Memory size (Mbytes)
Host A	Pentium	200	128
Host B	Celeron	1300	512
Host C	Pentium III (Mobile)	1200	256

(All hosts run FreeBSD 4.8-RELEASE)

Table 1 Specification of Hosts for The Experiments

Host name	Gen. rate†	Acc. rate⋆
Host A	10.9	18.4
Host B	154	114
Host C	157	123

(unit: kilopackets/sec)

†using localhost interface
⋆receiving through a 100BASE-TX Ethernet switch

Table 2 Zero-Payload UDP Packet Maximum Generation and Acceptance Rates

TCP and UDP Port 53 *must* be left open for the public access. Leaving a UDP port open to the Internet means that the open port can be exploited for a DoS (Denial-of-Service) attack, since generating a UDP packet stream is a trivial task for the attackers.

In this paper, the authors first explain how a DoS attack using UDP can be performed in Section 2, and analyze the behavior of an OS (Operating System) on handling a large-volume UDP traffic in Section 3. We describe the issues of UDP handling on DNS servers in Section 4, and conclude the paper in Section 5.

2 UDP DoS Attacks

DoS attacks are one of the most popular forms of security attacks performed over Internet links. UDP packets, as well as the ICMP (Internet Control Message Protocol) [5] packets, are popular for DoS attacks for the following reasons:

- For an effective DoS attack the packets should be generated and transferred as fast as possible. Since UDP is a connection-less protocol, the attacker does not need to wait for the connections to be established.
- The received UDP packets must immediately be forwarded to the listening server program. The server must determine the action to take for each packet, so the processing power is always consumed for each UDP packet. Even if the server does not exist, the UDP header must be processed by the protocol stack to determine whether to discard the payload or not.
- When performing a DDoS (Distributed DoS) attack, coordinating the attacker hosts is the overhead for the attacker. For UDP applications, the attacker does not need to synchronize with each other, since each UDP packet consists a complete message and instruction for the target server.

Figure 1 is an example of simplified C code of performing DoS for the FreeBSD OS, excerpted from a publicly-available source code [6]. It infinitely repeats the `send()` system call, while periodically pausing by the `usleep()` system call. Thanks to the recent processing performance improvement of computers, a recent i386-architecture PC can generate enough traffic to paralyze an old PC by such a simple code.

3 OS Handling Overhead of UDP

We performed a simple experiment to measure the maximum UDP receiving performance of each PC host shown in Table 1 by using the code shown in Fig. 1. These hosts run FreeBSD 4.8-RELEASE as the OS, and are directly connected to a 100BASE-TX switch, so the switch is the only device between the hosts.

Table 2 shows the results of the UDP zero-payload (i.e., the corresponding IP packet size = 28 bytes) packet generation using the local-host network interface, and the maximum number of accepted packets by the kernel through a 100BASE-TX Ethernet interface. All hosts listed in Table 2 showed the system CPU time percentage rate of 85 ~ 90 while accepting the UDP traffics.

The number of UDP packets received were measured through the output of function `badport_bandlim()` in the FreeBSD 4.8-RELEASE kernel source code file `/sys/netinet/ip_icmp.c`, which reports the number of ICMP-unreachable events per second. During the experiments, the events were caused by UDP packets sent to a UDP port which does not have the listening process [7], so the reported number equals to the number of UDP packets per second accepted by the kernel.

Note that the maximum packet forwarding rate of the 100BASE-TX switch was ≈149kpackets/sec, so the switch does not affect much to the packet transfer between the hosts. We also tested a cross-cable connection between the Host B and C, but we only observed the maximum packet rate from Host C to B was ≈50kpackets/sec, while that from Host B to C was ≈84kpackets/sec, much smaller than the rates with the 100BASE-TX switch.

We have found that DoS attack can be performed even by using a more complex programming language, such as Perl. We observed that the number of packet generated by a Perl 5 code equivalent to that shown in Fig. 1 was about 30 ~ 35 percent of that in the C code. Using the Perl code, a DoS attack from Host B to Host A was effectively performed with the tranmission packet rate of ≈45kpackets/sec.

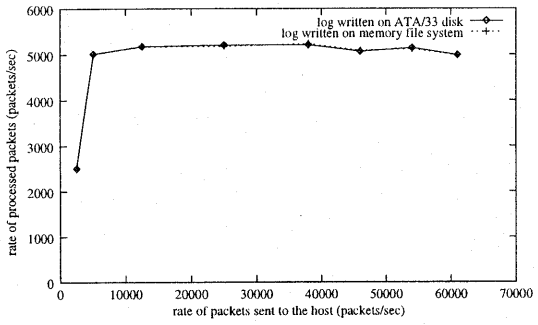

Fig. 2 Performance of `tinydns` against UDP DoS Packets

4 UDP Performance of A DNS Server

While the OS overhead of UDP handling is significant on evaluating DoS effectiveness [8], profiling actual DNS server under the DoS conditions is essential to find out the weakness inside.

A DNS server program must process each UDP packet, so the number of processed packet corresponds to the performance of the program. We tested the `tinydns` program, a UDP non-recursive DNS query server in a popular DNS software `djbdns` [9], with the default compile option (`gcc -O2`).

DoS packets for the tests were generated by the code of Fig. 1, and the packets were sent from Host B to Host C, which the `tinydns` was running.

We tested in two cases, one for logging into a file system on an ATA/33 disk, the other for logging into an asynchronous MFS (Memory File System), to evaluate how the storage performance affects the overall performance.

Figure 2 shows that after reaching the performance limit, the overall performance slightly decreased as the processing overhead of incoming packets increased. We also found that the logging file system did not affect on the processing performance.

`tinydns` logs a report of each UDP request using `multilog`, a general-purpose logging tool of `daemontools` [10]. Fig. 3 and Fig. 4 show the CPU usage, observed from the `top` command results, for each program and the total of the two programs.

Figure 3 shows that the CPU usage increased even after it reached the processing limit. The

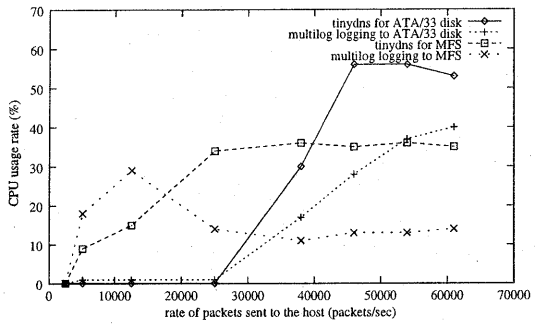

Fig. 3 Per-process CPU Usage of `tinydns` and `multilog` programs

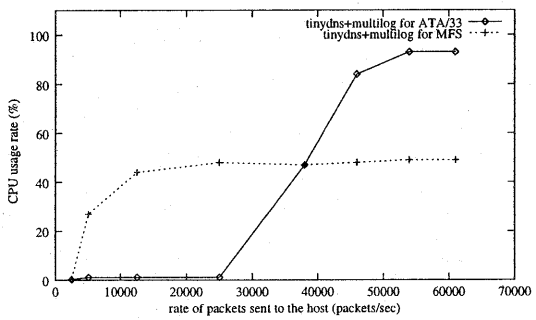

Fig. 4 Total CPU Usage of `tinydns` and `multilog` programs

usage of `tinydns` was always larger than that of `multilog`.

Figure 4 shows that the total CPU usage in the case of the file system on the ATA/33 disk rapidly increased after the incoming packet rate of 25000 packets/sec, while the usage in the MFS case was rather proportional to the overall processed packets. We suspect the device access latency of the ATA/33 disk contributes to the difference of behavior.

5 Conclusions and Further Works

In this paper, we presented a simple DoS-generation program is effective to paralyze other hosts. We also evaluated the behavior of a DNS server program by the CPU usage and the response rate against the zero-payload UDP DoS packets.

We need to consider the following issues for the further works:

- Establishing more accurate and easy-to-measure metrics for the processing resource usage (since we cannot use a profiler such as `gprof` for a daemon program);
- Testing against DNS query packets which actually have valid meanings (i.e., valid RRs) to find the effective DoS patterns;
- Locating the real bottleneck for increasing processing performance of a DNS server by detailed profiling.

DNS server and resolver functionalities are included in not only on the computer hosts but also on the embedded hardware products such as the broadband routers. The further analysis and development of countermeasure against DoS attacks is essential to improve the overall DNS security.

Acknowledgements

Our thanks go to Mr. Tohru Asami, the president and CEO of KDDI R&D Laboratories, Inc., for supporting our research activities.

References

[1] Mockapetris, P. V.: Domain names – concepts and facilities (1987). RFC1034 (also STD13).

[2] Mockapetris, P. V.: Domain names – implementation and specification (1987). RFC1035 (also STD13).

[3] Postel, J.: User Datagram Protocol (1980). RFC768 (also STD6).

[4] Postel, J.: Transmission Control Protocol (1981). RFC793 (also STD7).

[5] Postel, J.: Internet Control Message Protocol (1981). RFC792 (also STD5).

[6] STACKD: `std.c`. `http://packetstormsecurity.nl/groups/ldm/std.c`.

[7] Wright, G. R. and Stevens, W. R.: *TCP/IP Illustrated, Volume 2*, Addison–Wesley (1995).

[8] Otsuka, T.: Computational Resource Consumption of BIND against DoS, Technical report, Faculty of Engineering, Osaka University (2003). Graduation Research Report for the Bachelor's degree.

[9] Bernstein, D. J.: `djbdns`. `http://cr.yp.to/djbdns.html`.

[10] Bernstein, D. J.: `daemontools`. `http://cr.yp.to/daemontools.html`.