マルチメディア通信と分散処理 111－32
コンピュータセキュリティ 20－32
（２００３．２．２８）

公開 DNS 通信のセキュリティ確保

力武 健次 [‡][⋆]　中尾 康二 [‡]　野川 裕記 [⋆]　下條 真司 [⋆]

DNS（ドメイン名システム）にはトランスポート層にインターネット全体のセキュリティを弱めかねない基本的な欠点がある。 DNS データベースのトランザクションは大部分が UDP 上でなされており、サービス拒否攻撃に対してシステム全体が影響を受けやすい。本論文では公開されたインターネットゲートウェイシステムにおいて、 UDP による DNS サービスの提供のリスクについて論じる。その代案として、 DNS のトランスポート層への T/TCP（トランザクショナル TCP ）の導入を提案し、実装実験による評価を通じて、 T/TCP がパフォーマンスの損失を抑えつつ、 DNS トラフィックの制御をより容易にすることを示す。

Securing Public DNS Communication

Kenji Rikitake[‡][⋆], Koji Nakao[‡], Hiroki Nogawa[⋆] and Shinji Shimojo[⋆]

The DNS (Domain Name System) has a fundamental weakness on the transport layer, which may affect the overall security of the Internet. The DNS database transaction is mostly performed over UDP, which makes the whole system susceptible to denial-of-service attacks. In this paper, we first discuss the risk of providing DNS service through UDP access on publicly-exposed Internet gateway systems. We then propose introducing T/TCP (Transactional TCP) to the DNS transport layer as an alternative. We evaluate an experimental implementation and show how T/TCP is effective to improve the controllability of the DNS traffic while the performance degradation is minimal.

1 Introduction

One of the most important and mission-critical subsystems of the Internet Protocol Suite is DNS (Domain Name System) [1, 2]. Almost all Internet applications depend on DNS for the domain name resolution. Electronic mail messages use domain names to choose the source and destination addresses. The Web is fully dependent on domain names to properly choose the appropriate servers.

The integrity of Internet depends on how DNS is managed. The management is not only limited to the consistency of the database contents (Resource Records, RRs) but also to ensure the reliable access between the servers and the resolvers (clients). If DNS server-resolver communication is intentionally forged or spoofed for a security attack, the attacker can forward an email message to another server for monitoring or rewriting the contents, or redirect a Web service request to another server to provide a negative publicity by forged contents.

Massive attacks have been going on to publicly-exposed DNS subsystems, such as the server running on a corporate Internet gateway, or even to the Root Servers [3]. DNS has a fundamental weakness on the transport layer of using UDP [4] for the database queries, which makes the system prone to DoS (Denial-of-Service) attacks.

Cryptographic protocols such as DNSSEC [5] have been extensively studied for authentication of

[‡] 株式会社 KDDI 研究所
[⋆] 大阪大学 サイバーメディアセンター
[‡] KDDI R&D Laboratories, Inc.
{kenji,nakao}@kddilabs.jp
[⋆] Cybermedia Center, Osaka University
{nogawa,shimojo}@cmc.osaka-u.ac.jp

DNS traffics. Most of the DNS programs, however, are *not capable* of performing the cryptographic extension. For the production-level systems, the overall security of DNS should be considered *without the cryptographic extensions*.

In this paper, we focus on the transport layer security of DNS, and propose an alternative DNS transport with T/TCP (Transactional TCP) [6, 7]. We describe how T/TCP helps ensuring the transport security of DNS, by showing the practicality of replacing the current DNS queries of UDP with T/TCP, through the performance analysis and evaluation.

In the later sections, we first describe the security risks of public UDP services in Section 2, and analyze the transport layer behavior and requirements of DNS in Section 3. We then describe the T/TCP fundamentals and the advantages to traditional TCP in Section 4, and the evaluation results of T/TCP used as a DNS transport in Section 5. We conclude this paper on Section 6 with a discussion of the possible application fields of T/TCP to improve DNS security.

2 The Security Risks of Providing Public UDP Services Including DNS

Providing a *public* service on Internet, which means making the server accessible globally from anywhere by anybody, imposes a risk of security attacks. In case of UDP services, the risk is even greater. CERT Coordination Center advises disabling unneeded UDP services on each host [8].

UDP provides the following two functionalities:

- selecting data flow between different application services by assigning *port numbers* to each service;
- the per-packet checksum to ensure the data integrity of each packet.

On the other hand, what UDP does *not* provide are as follows:

- the retransmission functionality to provide a reliable communication between the application programs under data errors and packet losses;
- the state of being connected or not;
- the state of direction of connection, which distinguishes the clients and the servers.

The stateless nature of UDP is a vulnerability for the application protocols. While UDP is efficient on a reliable link when retransmission is not needed, it is less controllable at a firewall using a packet filter than TCP [9], since the UDP exchange does not have a state. The packet filter must be configured to leave the service port publicly accessible, which exposes a vulnerability, since it allows an unlimited access to the port. DoS attacks over UDP is much easier to perform than those over TCP, since the attacker does not have to manage the state of the connections.

The volume of UDP traffics is also difficult to control. UDP packets must be directly processed by the application software, which makes the software vulnerable to DoS (Denial-of-Service) attacks, since the content of the packets are directly passed onto the application software without being verified. This also indicates that the operating system on which the server runs should be capable of handling the volume of all incoming packets, whichever the packets are legitimate or not.

Some of the attacks to the known vulnerabilities of UDP-based protocols include the worm attack to Microsoft SQL Server [10], which caused a major traffic disruption in Korean and worldwide networks [11]. This attack used the UDP port 1434 assigned for The SQL Server Resolution Service, which can be abused to initiate a perpetual ping exchange between two Microsoft SQL servers [12]. Another well-known vulnerable UDP-based protocols is RPC (Remote Procedure Call) [13]. These examples show that opening unprotected UDP ports to public may result in harmful exploits.

Most of the well-known UDP-based protocols, such as DHCP, NTP, SYSLOG, and TFTP can be isolated from the public networks since they are of limited use in a private network environment. The UDP service of DNS, however, *must* be provided by all Internet sites for the domain name resolution by running a public server on the Internet, as stated in the Section 6.1.3.2 of RFC1123 [14]. The risk of DNS server attacks can not be reduced by the DNS delegation which many organizations and individuals have to the domain name authority to Internet service providers (ISPs), since only the targets are changed to the ISPs from their subscribers, which may be even more risky because the ISPs are delegated thousands of domain name authorities.

Moreover, DNS programs have already been exposed under persistent attacks [15, 16], so the transport layer should be protected as robust as possible. If the DNS database queries could efficiently performed over another transport protocol than UDP, public Internet gateway systems would no longer need to provide an open UDP service, and the systems could be much more secure and easier to defend. We discuss the possibility of the protocol replacement in the later sections.

3 DNS Transport Layer Issues

DNS server-resolver communication protocol is collectively defined by many Internet RFCs (Request For Comments). The three important RFCs are:

- RFC1034 [1], which specifies the architecture of DNS;
- RFC1035 [2], which specifies the implementation details;
- RFC1123, which specifies the DNS usage as a part of the Internet host requirements.

DNS has two major forms of data exchange between servers and resolvers as follows:

RR Queries: this occurs between the servers and resolvers. Most of the real-world traffic of the queries is over UDP, though TCP is also allowed and supported by the majority of servers.

Zone Transfer: this occurs between two servers for replication of a set of RRs to obtain redundancy against a possible server failure. This is performed solely over TCP.

We focus on the RR queries in this section, since the our goal is to improve the efficiency and security of the DNS database lookup.

While both UDP and TCP are allowed on the initial DNS queries, TCP must be used for a large-size reply. Section 4.2.1 of RFC1035 explicitly restricts the size of UDP queries and answers to 512 bytes. Section 6.1.3.2 of RFC1123 shows that a DNS server *must* service UDP queries and it *should* service TCP queries, and allows private agreement of servers and resolvers to solely use TCP for the queries.

Section 4.1.1 of RFC1035 specified the DNS header format. In the format, the `TC` bit is set when a server sends a truncated reply, due to length greater than that permitted on the transport. The `djbdns` [17] behavior of the resolver which receives a UDP answer with the `TC` bit set is to reissue the request to the server using TCP [18] all over again. This means the query reply longer than 512 bytes is always sent back by TCP, after waiting a UDP exchange solely for the notification purpose.

The limit imposed by UDP packet size leads to a restriction that answers to the DNS queries for the Root Servers must be fit into 512 bytes, The UDP size limitation means a restriction which only 13 IPv4 servers can be specified in the SOA answer [*1]. If the number of Root Servers were increased or some of the servers also announced the IPv6 addresses by the AAAA RR, the answer may easily exceed the 512-byte size limit [18], so the query answer for the Root Servers will not be able to be carried over UDP. If TCP can be used as a primary protocol of DNS queries, this problem will be solved, since it has no limitation of the data size per packet.

DNS programs has its own retransmission and timeout algorithms for the UDP transport, since UDP does not have them. For example, `djbdns` uses the timeout algorithm [19] of waiting 3, 11, and 45 seconds respectively for each UDP recursive queries, and terminates the operation if nothing received after retransmitting three times. This retransmission strategy works well when the packet loss rate of the network is small. When the packet loss rate is very high, however, it may cause delay of the completion of query processes, either succeeded or failed, since only four or less packets are sent for each query. On the other hand, TCP has its own retransmission algorithm implemented, which scales well on various conditions of links.

4 T/TCP and Traditional TCP

T/TCP is an extension of TCP, designed for a transactional use between a connection-based client-server communication which fits very well

[*1] The RRs of the query answer for the Root Domain is 1 SOAs, 13 NSes and 13 As, 493 bytes in total.

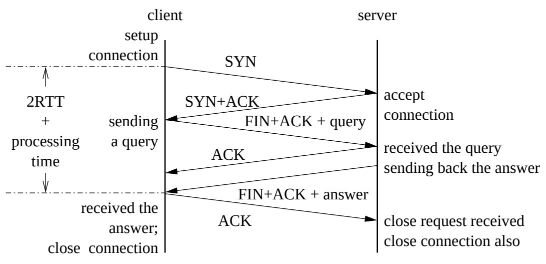

Fig. 1 Traditional TCP Time Line

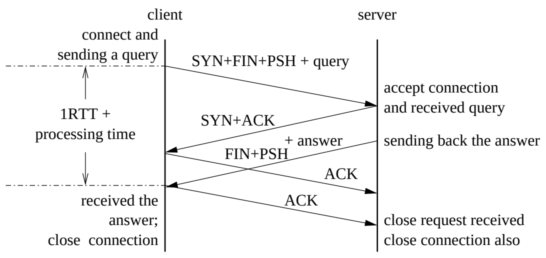

Fig. 2 T/TCP Time Line

to DNS database exchange. The transactional model of communication proceeds as the following sequence:

- the client sends a request to the server;
- then the server sends back the reply;
- the exchange completes and the link is disconnected.

Using traditional (non-transactional) TCP for the transactional model sequence requires two round-trip exchanges. Figure 1 shows the time line of traditional TCP. It shows that the first of the two exchanges is solely for setting up a TCP connection, while the second one is actually used for the data exchange.

On the other hand, using T/TCP requires only one round-trip exchange, which is the same as in the UDP case. Figure 2 shows the time line of T/TCP. It shows that the first packet sent from the client to the server carries the query data as well as the connection request. Putting the query data on the same packet for the connection request is performed by using the CC (Connection Count) options, introduced by T/TCP, to indicate the support and to avoid duplicate old connections.

Note that in either time line figures of Fig. 1 or Fig. 2, the meaning of the `ACK` bit in TCP header is left unchanged. The packet filtering rules of allowing only *established* connections of TCP are applicable to T/TCP with no need to change.

T/TCP has another feature to shorten the time spent in TCP `TIME_WAIT` state which is to complete full-duplex closing of a connection and to allow old duplicate TCP segments to expire. The amount of time spent in the `TIME_WAIT` is traditionally specified as twice the MSL (Maximum Segment Lifetime). T/TCP specifies the length of `TIME_WAIT` as eight times the RTO (Retransmission Timeout) when the connection duration is less than the MSL. This means that the length of `TIME_WAIT` state is shortened from 60 to 4 seconds on FreeBSD 4.6.2-RELEASE. A smaller length of `TIME_WAIT` state means a smaller size requirement to the network control block, and an increase of number of TCP connections which a server host can simultaneously handle.

T/TCP is fully backward-compatible with the traditional TCP. When the server is T/TCP-aware, it can identify the client is T/TCP-aware or not by referring to the CC options sent from the client. On the other hand, when the server is not T/TCP-aware, it discards the data payload on the first packet with `SYN` flag to avoid TCP `SYN`-flooding, so the client to wait for an additional error timeout period for each transaction. Nevertheless, the backward compatibility is still retained.

5 Evaluation of T/TCP on DNS

We tested T/TCP as a DNS transport by modifying the program code of `djbdns` on FreeBSD 4.6.2-RELEASE using `dummynet` [20], and measured the performance and behavior. The details of `djbdns` modification are listed as follows:

- adding a function to set the `TCP_NOPUSH` socket flag, and an interface to `sendto()` system call for `djbdns` socket library;
- changing the DNS resolver interface functions called from the `djbdns` programs to use T/TCP instead of traditional TCP;
- changing `dnscache`, the DNS cache program, to use T/TCP for accepting the connections and external lookups.

The conditions of DNS query time measurements are as follows:

Table 1 Total Elapsed Time of 1000 Sequential DNS Queries to a `dnscache` Server (in seconds)

	local	Ether	ADSL
RTT (ms)	≈0.04	≈0.4	60~70
UDP	0.22	2.40	67.77
T/TCP	0.52	8.70	74.70
TCP	0.53	8.92	138.80

RTT: Round-Trip Time

- `dns_resolve()`, a DNS resolver function of `djbdns`, is called for each query. A modified version is used to perform TCP-only DNS queries. `DNSCACHEIP`, The environment variable is set to choose the appropriate `dnscache` to test.
- Each query contains a request to the NS RRs of the Root Domain (`"."`), which `dnscache` can answer solely by referring to a configuration file `root/IP/@`, with no external or internal lookup.

Table 1 shows the result of measuring the difference of query processing time between UDP, T/TCP and TCP for different types of links. We used a local interface, a 100BASE-TX Ethernet, and an ADSL (Asynchronous Digital Subscriber Link) of an Internet service provider.

For the local interface and Ethernet links, UDP is the fastest, since the number of packets exchanged for each query differs; 2, 5, and 6 for UDP, T/TCP, and TCP, respectively. On the other hand, the testing of the ADSL link shows that the overhead of T/TCP to UDP is only 10% of the total time, while TCP takes about twice as much as UDP does. This is consistent with the time line explanation on Section 4.

We also performed a test to evaluate how the random packet loss rate affects the query success rates of UDP and T/TCP. Since UDP exchange takes 59 seconds as the maximum value by the `djbdns` retransmission algorithm, the value of T/TCP timeout to determine the success of query is extended from the default value of 10 to 60 seconds on both the server and resolver sides. We used two hosts connected with a 100BASE-TX link and `dummynet` for the network delay simulation. 100 concurrent queries was conducted.

Figure 3 shows the result in the case of 500 milliseconds delay. In this case, UDP and T/TCP

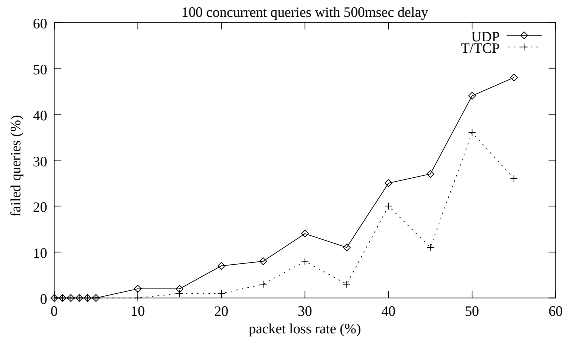

Fig. 3 Query Failure Rates of UDP and T/TCP for Link with 500ms Delay

showed little difference at the low packet loss rates ≤ 10%. As the packet loss rate increases, the difference between UDP and T/TCP also increase. At the packet loss rate of 55%, more than a half of UDP queries fails, while T/TCP queries only failed about 25% of the whole queries. This indicates T/TCP is effective for decreasing the worst-case failure rate for DNS queries in the networks of high packet loss rates.

6 Conclusion and Further Works

In this paper, we proposed to use T/TCP as a DNS transport, evaluated the protocol by an implementation, and showed that T/TCP is an effective alternative to enhance the overall security of DNS by increasing the reliability of the query processing, while maintaining the controllability of TCP on the firewalls.

DNS has become the only mandatorily-required UDP protocol which a firewall connected to the public Internet must support for non-private exchange. While simply prohibiting the UDP queries of DNS may work, it will increase the consumption of the server host resources, as TCP exchange requires the connection blocks inside the operating system kernel. As shown in Section 4, T/TCP shortens the timeout state length, which largely affects the maximum processing capability of a server host, to 1/15 of the traditional TCP. This will reduce the average resource consumption of the server host. And as shown in Section 5, T/TCP is a practical solution to replace UDP DNS lookups on an ADSL or other kinds of network, which have larger latencies, and which many of the end-user Internet sites use. The zone transfer exchange of DNS will benefit from T/TCP for the fast startup and earlier closing of connections as well. If the workload increase due to the T/TCP resource consumption is of concern, the expertise for Web servers are applicable to reduce the impact.

Using T/TCP for DNS lookups from mobile equipments may be effective. Links from those devices often fails because of the high packet loss rate. T/TCP work better than UDP in such a case. Even in a lower packet loss rate, T/TCP has only 10% of query time overhead than UDP in a wide-area network environment which has the RTT of $\geq$ 60 milliseconds. Changing the current UDP queries to T/TCP is a practical solution for mobile equipments, since it eliminates a requirement for UDP protocol stack and gives more control on the firewall between Internet and the networks of the equipments.

We consider three major issues should be discussed for the further works:

- the detailed security analysis on T/TCP;
- how T/TCP can be used on other UDP-based applications;
- estimating resource consumption overhead of DNS migration from UDP to T/TCP.

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] Vixie, P., Sneeringer, G. and Schleifer, M.: Events of 21-Oct-2002. `http://f.root-servers.org/october21.txt`.

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

[5] Eastlake, D.: Domain Name System Security Extensions (1999). RFC2535.

[6] Braden, R.: Extending TCP for Transactions – Concepts (1992). RFC1379.

[7] Braden, R.: T/TCP – TCP Extensions for Transactions Functional Specification (1994). RFC1644.

[8] CERT/CC: UDP Port Denial-of-Service Attack. CERT Advisory CA-1996-01, `http://www.cert.org/advisories/CA-1996-01.html`.

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

[10] CERT/CC: MS-SQL Server Worm. CERT Advisory CA-2003-04, `http://www.cert.org/advisories/CA-2003-04.html`.

[11] CNET News.com: Computer worm slows global Net traffic. `http://news.com.com/2100-1001-982131.html`.

[12] CERT/CC: Microsoft SQL Server 2000 contains denial-of-service vulnerability in SQL Server Resolution Service. CERT Vulnerability Note VU#370308, `http://www.kb.cert.org/vuls/id/370308`.

[13] CERT/CC: Integer Overflow In XDR Library. CERT Advisory CA-2002-25, `http://www.cert.org/advisories/CA-2002-25.html`.

[14] R. Braden (Editor): Requirements for Internet Hosts – Application and Support (1989). RFC1123.

[15] CERT/CC: Denial-of-Service Vulnerability in ISC BIND 9. CERT Advisory CA-2002-15, `http://www.cert.org/advisories/CA-2002-19.html`.

[16] CERT/CC: Buffer Overflows in Multiple DNS Resolver Libraries. CERT Advisory CA-2002-19, `http://www.cert.org/advisories/CA-2002-19.html`.

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

[18] Gudmundsson, O.: DNSSEC and IPv6 A6 aware server/resolver message size requirements (2001). RFC3226.

[19] Bernstein, D. J.: User's guide to name resolution. `http://cr.yp.to/djbdns/resolve.html`.

[20] Rizzo, L.: Dummynet: a simple approach to the evaluation of network protocols, *Computer Communication Review*, Vol. 27, No. 1, pp. 31–41 (1997).