TT-DNS: Trustworthy Third-Party DNS Service with Privacy Preserving

Poster # 2, 2 pages

1 INTRODUCTION

DNS Privacy is a concern and several solutions are proposed (e.g., DNSCrypt[1] for DNS message confidentiality, and PPDNS[7] and Anonomous DNS[6] for DNS client privacy), but these solutions focus on "channel attacks", not "end point attacks". In recent years, DNS is more and more offloaded to third-party service providers (e.g., Google public DNS[2], OpenDNS[3]). In these scenarios, *whether these non-authoritative DNS servers are honest and trustworthy becomes a security concern for DNS clients.*

We design a *privacy-preserving and trustworthy* third-party DNS service, named TT-DNS. Our goal is to protect DNS clients' privacy, which is defined as the association between the client's identity (i.e., IP address) and its queried names (i.e., URL or ENUM [4, 5]), and the adversaries include those on the communication channel and on the DNS servers. TT-DNS achieves the following goals: (1) the channel attackers cannot violate clients' privacy (**security**), (2) the DNS server attackers cannot violate clients' privacy (**security**), and (3) the system is deployable on the current Internet (**practicality**).

TT-DNS consists of a DNS agent and a DNS resolver. The DNS agent runs on the DNS client's end host, which intercepts and sends the DNS queries to a DNS resolver, and the DNS resolver further queries the Internet DNS service (e.g., authoritative DNS server or trustworthy local ISP DNS server) and replies to the DNS agent.

The whole communication channel includes two segments, and the attacks may come from both of them. On the communication channel between the DNS agent and the DNS resolver, TT-DNS applies a secure communication protocol, and on that between the DNS resolver and the Internet DNS, TT-DNS applies *obfuscation mechanism.*

The server attacks include in-memory information steal in the *runtime* and software forgery in *initialization* time. To defend attacks from runtime untrusted server operators (i.e., server attackers), TT-DNS is implemented on a trusted execution environment (TEE, we use Intel SGX in this poster); thus, the *runtime* memory of the DNS resolver could be protected.

To defend from initialization time software forgery, we also design an initialization protocol that combines the Intel SGX and PKI, so that (1) the software integrity of the DNS resolver can be attested by the auhoritative developer, and (2) the attestation result can be passed to the DNS clients. We prototype TT-DNS and evaluate it to show its feasibility, and also point out the space for future improvement.

2 DESIGN

The workflow of serving a client's DNS query involves two communication channels. On the channel between the client and the DNS resolver, a client query and its response are encrypted using an authenticated encryption protocol (i.e., AE; e.g., AES-GCM); on the channel between the resolver and the trustworthy DNS source, the query is obfuscated with many fake queries. In this design, we assume it is practical to incrementally deploy the DNS agent on the client side (along with the AE encryption), but it is not to make radical changes to the existing Internet architecture (so we use an obfuscation mechanism).

In the obfuscation mechanism, the DNS resolver is initialized with an obfuscation degree N and a domain $NameCache$ with a few names. For each incoming encrypted query, it is decrypted and the name in query is extracted. Then a query set of size N is generated. All queries in the set are sent to the trustworthy DNS source and the returned results are put into the cache. Finally, the client's query result is fetched from the cache and returned to the client.

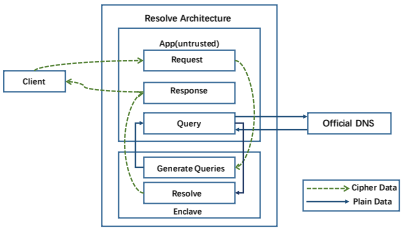

Figure 1: Implementation

Figure 1 shows the implementation of the DNS resolver algorithm on SGX. An SGX application consists of trusted parts (in enclave) and untrusted parts. In the DNS resolver implementation, we put the global variables $ClientKey$ and $NameInQuery$ and the functions $GenerateQuerySet$ and $ResolveName$ into SGX enclave. In the trusted code and data segment, the client's query is decrypted, the name in query is cached, the query set is generated and sent out of the enclave, and the final response $< name, IP >$ is encrypted. In the untrusted code and data segment, the encrypted query and response are exchanged with the client, and the plaintext query set and responses are exchanged with the trustworthy DNS servers.

In the *initialization* time, the DNS clients also need to verify whether it is the genuine version of DNS resolver that is loaded into the memory. This process involves four parties: DNS clients, the DNS resolver, the authoritative TT-DNS developer, and Intel Attestation Service (IAS). We combine the original SGX attestation[1] with Public Key Infrastructure (PKI) to achieve this verification. The detailed protocol is shown in (Figure 2).

We extend the attestation process from the original three parties (i.e., application, service provider, and IAS) to four parties, which includes DNS clients, DNS resolver, an authoritative verifier, and IAS. The protocol combines Intel Remote Attestation and Public Key Infrastructure (PKI), and finally it achieves the following results. First, the DNS clients can verify the integrity and the in-enclave execution of the DNS resolver software; second, the symmetric session key can be set up between the enclave and the client. The detailed protocol is shown in (Figure 2).

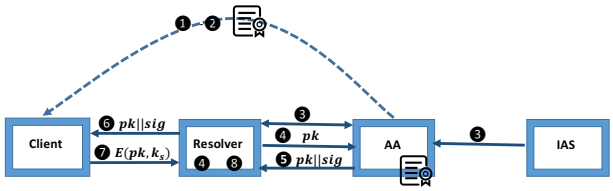

Figure 2: Initialization

(1) An Authoritative Agent AA (i.e., the authoritative TT-DNS developer) is deployed on the Internet, which has its own certificate and secret key.[2]
(2) The client installs the DNS agent, and also gets the certificate of AA. We assume the certificate is installed out of band of the following DNS queries, and the client trusts the AA (i.e., the trust anchor).
(3) When the DNS resolver boots up, it performs the remote attestation with the AA. AA would keep a secure communication connection with the resolver's enclave.
(4) The DNS resolver generates a public-secret key pair $< pk, sk >$. It keeps the sk with in the enclave and sends pk to the AA.
(5) AA signs the pk with its own secret key, and sends a ticket $pk||sig$ to the resolver enclave.
(6) For each DNS client who would issue DNS query, it would first ask for the ticket $pk||sig$ from the resolver, and validate the signature sig of the resolver public key pk with the AA's certificate.
(7) If $pk||sig$ passes the check, the DNS agent would randomly choose a session key k_s and send its encryption $E(pk, k_s)$ to the DNS enclave.
(8) The DNS enclave would decrypt and store the session key and its client IP <clientIP, k_s>.
(9) The following DNS queries and responses would follow the procedure in the workflow.

[1]The original Intel attestation has three parties: the application (the DNS resolver), the service provider (the TT-DNS developer), and IAS.
[2]It can be part of the Internet authoritative DNS.

3 EVALUATION AND CONCLUSION

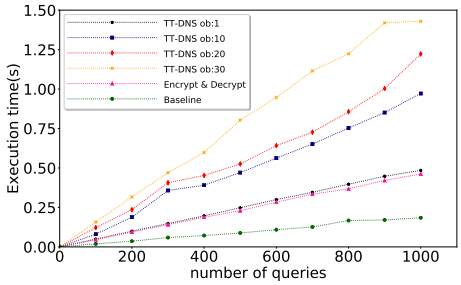

Figure 3: Overall performance

We prototype and profile the overall performance of TT-DNS. There are three parameters in the experiment: whether the AE procotol (AES-GCM mode) is enable, whether the SGX protection is enabled, the obfuscation degree N. Figure 3 is a preliminary experiment that shows the total execution time of DNS queries with varying number of client queries. The baseline is a DNS resolver that only relay DNS queries and responses, which takes about 0.18ms each. Adding client side encryption/decryption would increase the time to be 0.46ms. Implementing the obfuscation algorithm (with degree 1) would slightly increase the per-query execution time to be 0.48ms. And finally, with the obfuscation degree increasing, the per-query execution time is 0.97ms, 1.22ms and 1.42ms with degree 10, 20 and 30.

we conclude that TT-DNS is feasible to provide service on the Internet, and the most overhead is introduced by encryption/decryption and the multiple obfuscating queries. Thus, we plan to accelerate TT-DNS by batching and parallelizing the two procedures in the future.

REFERENCES

[1] [n. d.]. Homepage of DNSCrypt. https://www.dnscrypt.org/. ([n. d.]). Accessed May 1, 2019.
[2] [n. d.]. Homepage of Google Public DNS. https://dns.google.com/. ([n. d.]). Accessed May 1, 2019.
[3] [n. d.]. Homepage of OpenDNS. https://www.opendns.com/. ([n. d.]). Accessed May 1, 2019.
[4] [n. d.]. RFC 1034. https://tools.ietf.org/html/rfc1034. ([n. d.]). Accessed May 1, 2019.
[5] Sergio Castillo-Perez and Joaquin Garcia-Alfaro. 2008. Anonymous resolution of dns queries. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer, 987–1000.
[6] Giuseppe Di Bella, Cettina Barcellona, and Ilenia Tinnirello. 2013. A secret sharing scheme for anonymous DNS queries. In *AEIT Annual Conference 2013*. IEEE, 1–5.
[7] Yanbin Lu and Gene Tsudik. 2010. Towards plugging privacy leaks in the domain name system. In *2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P)*. IEEE, 1–10.