PDoT: Private DNS-over-TLS with TEE Support

Yoshimichi Nakatsuka\textsuperscript{1}, Andrew Paverd\textsuperscript{2}, Gene Tsudik\textsuperscript{1}

\textsuperscript{1}University of California, Irvine, USA \quad \textsuperscript{2}Microsoft Research Cambridge, UK
DNS

DNS: Domain Name Service, TLD: Top Level Domain
Privacy issues in DNS

ics.uci.edu

Client Application

Client Stub

Recursive Resolver

Name Server (Root)

Name Server (TLD)

Authoritative Name Server

ics.uci.edu: 128.195.1.88

Cache

① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩
DNS-over-TLS [S&P’ 15]

TLS: Transport Layer Security
DNS-over-TLS gaining attraction

What is 1.1.1.1?
1.1.1.1 is a public DNS resolver that makes DNS queries faster and more secure.

Google Security Blog
The latest news and insights from Google on security and safety on the Internet

DNS over TLS
Updated on 01 Nov 2018 | 3 minutes to read | Contributors

RFC 7858 specifies DNS over TLS (Transport Layer Security). This article explains how to provide a DNS over TLS service using BIND 9 and stunnel. The setup of a privacy aggregator is at the end.

DNS over TLS support in Android P Developer Preview
April 17, 2018
Issues of DNS-over-TLS

1. How can clients authenticate resolvers in a scalable manner?

Current state: Clients must add hash of public key manually to configuration file
Issues of DNS-over-TLS

2. How can clients trust the resolver?

Current state: Recursive resolvers can “see” raw queries.
Idea: Run recursive resolver in TEE!
Trusted Execution Environment (TEE): Huh?

- Isolates code and data from privileged software
- Provides several functionalities, including:
  - Memory Security
  - Remote Attestation
- Implementations of TEEs:
  - AMD Secure Encrypted Virtualization (SEV)
  - ARM TrustZone
  - Intel Software Guard Extension (SGX)
Intel SGX

- **Enclave:**
  Secure container containing isolated data & code

- **Memory Security:**
  Enclave Page Cache (EPC):
  Memory reserved for enclaves. Encrypted and Access Controlled.
Intel SGX

- **Remote Attestation**: Determine whether a genuine, remote enclave is running expected code

1. Send data (MRENCLAVE, PK)
2. Verify data
3. Generate digital signature of data
4. Send signature back to A
5. Send digital signature and data
6. Verify signature and data (Binds PK to MRENCLAVE)

MRENCLAVE: Hash of enclave code
PK: Public Key
PDoT: Private DNS-over-TLS with TEE Support

Aim to address the two issues using Secure Memory and RA:

1. How can clients authenticate resolvers in a scalable manner?
   a. Clients can authenticate resolver based on behavior (RA)

2. How can clients trust the resolver?
   a. Host of resolver cannot “see” raw query (Secure Memory)
   b. Client knows what code the resolver is running (RA)
Design Challenges

1. **TEE Limited Functionality**
   a. Applications running in TEE cannot make system calls
   b. Must rely on untrusted host for communication

2. **TEE Memory Limitations**
   a. Typical EPC size is 128MB.
   b. Should avoid page swapping for better performance

3. **TEE Call-in/Call-out Overhead**
   a. Switching from TEE to non-TEE side introduce additional overhead
   b. Should minimize switching back and forth for better performance
Adversary Model

- Malicious recursive resolver operator 🤡
  - Has control of OS and applications (including enclaves)
  - Can meddle packets
System Requirements

1. Query Privacy
   a. Client’s query must not be learned by adversary

2. Response Latency
   a. Present similar response latency as other DoT resolvers

3. Scalability
   a. Process queries generated by realistic number of clients
Setting up Secure Tunnel

Client

- PDoT Cert
- Sig Cert
- R Sig
- R

PDot

Sig

Cert

Verify

Verify

Check

MRENCLAVE

H

MRENCLAVE

Secure Tunnel

PDot

Sig

Cert

R

R Sig

Sig Cert

Application Enclave

PDot Cert

R

MRENCLAVE: Hash of enclave code

R: Data in Report

R Sig: Signature of Data in Report

Sig Cert: Signature Certificate
Security Analysis

- Malicious resolver operator
  - DNS queries flow in TLS tunnel: Cannot learn query (encryption)
  - TLS endpoint inside enclave: Cannot learn query (secure memory)
  - Trick client to establish TLS channel with malicious TEE resolver: Detectable (wrong MRENCLAVE)
Security Analysis

- Malicious resolver operator
  - DNS queries flow in TLS tunnel: Cannot learn query (encryption)
  - TLS endpoint inside enclave: Cannot learn query (secure memory)
  - Trick client to establish TLS channel with malicious TEE resolver: Detectable (wrong MRENCLAVE)
Security Analysis

- Malicious resolver operator
  - DNS queries flow in TLS tunnel: Cannot learn query (encryption)
  - TLS endpoint inside enclave: Cannot learn query (secure memory)
  - Trick client to establish TLS channel with malicious TEE resolver: Detectable (wrong MRENCLAVE)

MRENCLAVE: Hash of enclave code
System Requirements

1. Query Privacy
   a. Client’s query must not be learned by adversary

2. Response Latency
   a. Present similar response latency as other DoT resolvers

3. Scalability
   a. Process queries generated by realistic number of clients
Evaluation Setup

Intel NUC with

- CPU: Intel Pentium Silver J5005
- EPC memory: 128 MB
- RAM: 4 GB
- OS: Ubuntu 16.04
- SGX SDK: ver. 2.2

Used Unbound¹ DNS-over-TLS resolver as baseline

¹: https://nlnetlabs.nl/projects/unbound/about/
Latency of PDoT and Unbound (Warm Start)

- Send 100 queries to each domain
- Reuse TLS tunnel
- Measure time between request & response

- PDoT poses acceptable increase in latency
System Requirements

1. **Query Privacy**
   a. Client’s query must not be learned by adversary

2. **Response Latency**
   a. Present similar response latency as other DoT resolvers

3. **Scalability**
   a. Process queries generated by realistic number of clients
Scalability of PDoT and Unbound

- How many queries per second a resolver can handle?
  ⇒ Measure throughput
- Constant latency at rate
  ⇒ Throughput

- PDoT & Unbound pose similar throughput
System Requirements

1. Query Privacy
   a. Client’s query must not be learned by adversary

2. Response Latency
   a. Present similar response latency as other DoT resolvers

3. Scalability
   a. Process queries generated by realistic number of clients
Caching in DNS-over-TLS and PDoT resolver

Caching can lead to following attacks:
- Measuring time between query and response
- Correlating client and resolver requests

Caching in PDoT is possible, despite small secure memory size.

Based on MRENCLAVE value, PDoT clients can choose between
- Faster, but less private PDoT resolvers with caching
- More private, but slower PDoT resolvers without caching

MRENCLAVE: Hash of enclave code
Conclusion and Future Work

- Proposed PDoT, a novel DNS recursive resolver that operates within a TEE
- PDoT protects privacy of DNS queries even from PDoT operator
- Unoptimized implementation of PDoT shows:
  - Acceptable increase in query latency
  - Throughput matches that of Unbound

- Future work:
  - Port Unbound to Intel SGX and conduct performance comparison
  - Explore methods for improving PDoT’s performance
Thank you!

Questions?

Yoshimichi Nakatsuka (nakatsuy [at] uci.edu)
Appendix