Danish: HTTPS DANE Validation on Linux

Andrew McConachie

Image courtesy Wikimedia Commons

DANE Overview

- Presentation Assumptions
  - Basic knowledge of DNS and DNSSEC
  - Basic knowledge of the Public Key Infrastructure (PKIX)
    - X.509 Certificates
    - Transport Layer Security (TLS)
- DNS-Based Authentication of Named Entities (DANE)
  - Defined in RFCs 6698, 7218, 7671
  - Ties X.509 certificate trust to DNS
  - Starting to be used in SMTP but very little use for HTTPS
- TLSA Resource Record
  - Contains the cryptographic hash of an X.509 certificate
  - Plus other stuff ..

TLSA Records

```
_443._tcp.www.digid.nl. 900 IN TLSA 3 0 1
DDC85B7EDAA3F3C65A34AEAD4C5A36DB2677065F659D5A554AB56E2C2EDC5F8E
```

- Certificate Usage
  - 0 – PKIX-TA (DANE && local store)
  - 1 – PKIX-EE (DANE && local store)
  - 2 – DANE-TA (DANE only)
  - 3 – DANE-EE (DANE only)
- Selector
  - 0 – Full X.509 Certificate
  - 1 – Subject Public Key Information (SPKI) “public key”
- Matching Type
  - 0 – No hashing used
  - 1 – SHA256
  - 2 – SHA512

Danish History

- First presented at ICANN 59 in June 2017
  - Written in Python
  - Only ran on OpenWRT
- Rewrote Danish in 2019
  - Written in Rust
  - Supports both middlebox and host operation on Linux
  - BSD-3 licensed
- I have been DANE validating HTTPS traffic for about 3 years
  - Primarily an exercise in NXDOMAIN generation
  - I have also found some TLSA records in the wild

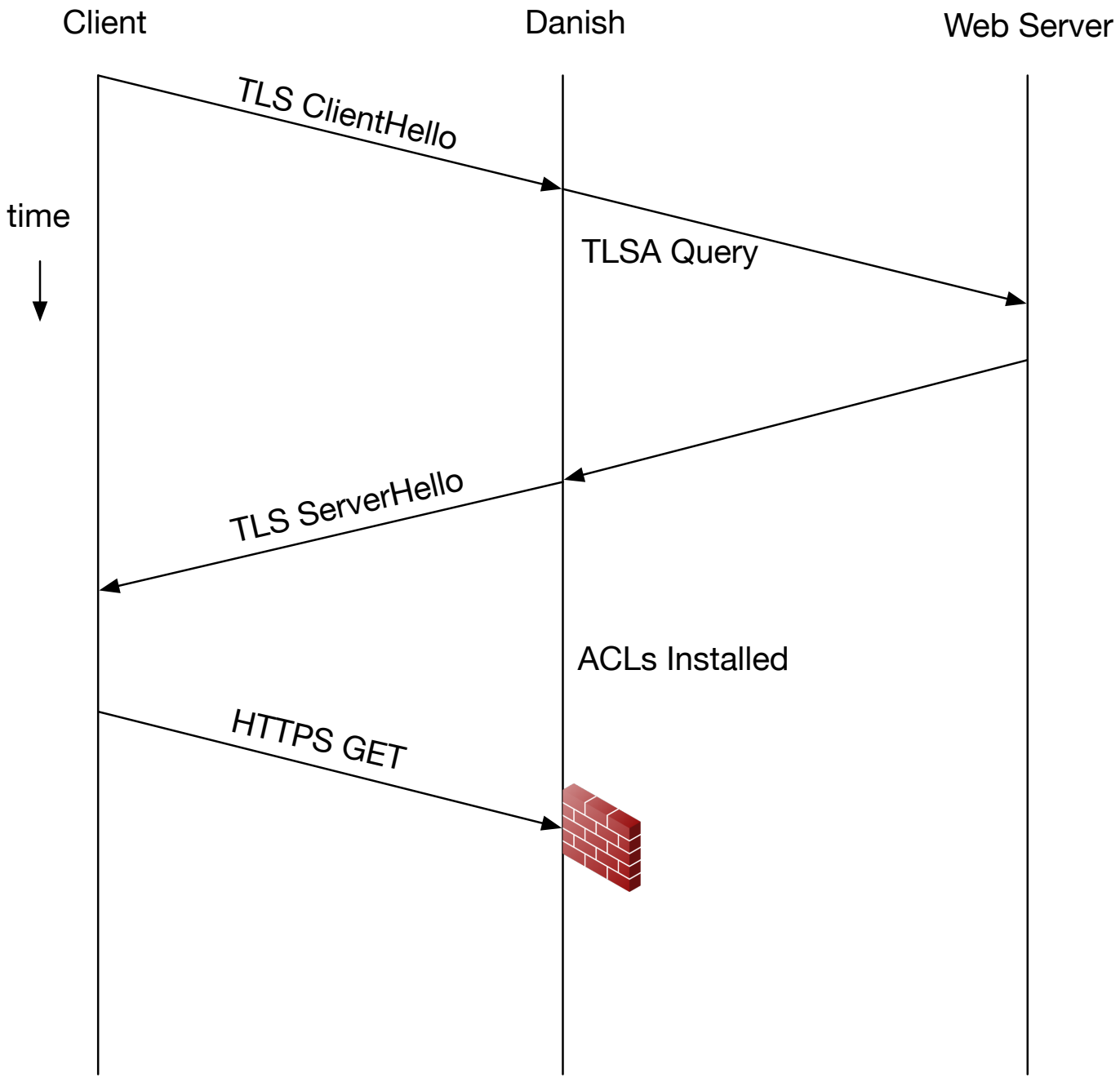
What is Danish?

- Linux daemon for validating HTTPS DANE
  - Sniffs TLS Handshake traffic with lib-pcap
- If validation fails ACLs are installed to deny traffic
  - Uses iptables extensions to deny access to specific TLS SNIs
- Uses lib-resolv as DNS stub resolver
  - Does not perform DNSSEC validation
- Can be run on firewalls or end-hosts
  - iptables chains FORWARD or OUTPUT
- Can block SNI based on DNS Response Policy Zones (RPZ)
  - Look up TLS SNI in DNS
  - If NXDOMAIN → install ACLs
- Supports TLS 1.0 - 1.2, IPv4/IPv6

Operation

1. Sniff HTTPS TLS ClientHello and ServerHello messages
   - Parse Server Name Identifier (SNI) from ClientHello
   - Parse X.509 Certificates from ServerHello
2. Perform DNS TLSA lookup for comparison
3. If no TLSA RR found → Do Nothing
4. If X.509 Certificate and TLSA RR match → Do Nothing
   - Danish has no local certificate store
5. Else install ACLs to block client traffic to offending web server
   - 2 short lived ACLs to force TCP timeout
   - 1 long lived ACL to prevent further egressing TLS ClientHellos with matching SNI
     - Installed for both IPv4 and IPv6

Validation Failure



Validation Failure ACL

```
# iptables --list -n
..
Chain danish_69763cbbe640ba4f2d86 (1 references)
target     prot opt source               destination
DROP       tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:443 STRING match
"|0000001b00190000166261642e6d6964646c65626f782d64616e652e6f7267|" ALGO name bm TO 65535 /*
bad.middlebox-dane.org */
DROP       tcp  --  192.168.1.153        95.179.156.120      tcp spt:33256 dpt:443
DROP       tcp  --  95.179.156.120       192.168.1.153       tcp spt:443 dpt:33256
RETURN     all  --  0.0.0.0/0            0.0.0.0/0
```

HTTPS TLSA RRs in the Wild

Never see certificate usage 0 or 2.

```
_443._tcp.access.ripe.net.              IN      TLSA    3 0 1
_443._tcp.danetools.com.                IN      TLSA    3 1 1
_443._tcp.defcon.org.                   IN      TLSA    3 0 2
_443._tcp.digid.nl.                     IN      TLSA    3 0 1
_443._tcp.www.freebsd.org.              IN      TLSA    3 1 1
_443._tcp.frobbit.se.                   IN      TLSA    3 1 1
_443._tcp.login.enterprise-email.com.   IN      TLSA    3 0 1
_443._tcp.mail.pab.ro.                  IN      TLSA    3 1 1
_443._tcp.mijnoverheid.nl.              IN      TLSA    1 0 1
_443._tcp.mijn.overheid.nl.             IN      TLSA    1 0 1
_443._tcp.overheid.nl.                  IN      TLSA    1 0 1
_443._tcp.posteo.de.                    IN      TLSA    3 1 1
_443._tcp.sys4.de.                      IN      TLSA    3 1 1
```

CLI Options

- -c, --chain <OUTPUT | FORWARD>
  - iptables/ip6tables top chain [default: OUTPUT]
- -i, --interface <device>
  - pcap device to listen to [default: eth0]
- -s, --sub-chain <sub_chain>
  - iptables/ip6tables sub-chain for ACLs [default: danish]
- -r, --rpz
  - Enable Response Policy Zone (RPZ) checking [default: disabled]
  - If DNS query for SNI fails install ACLs
- No option to enable/disable IPv6 support
  - If ip6tables is present IPv6 support is enabled, otherwise disabled

Lessons Learned

- I’m still experimenting with this
  - Code is likely bug ridden
- It’s a race
  - Sometimes Danish installs the ACLs in time, sometimes not
  - Much better chance at winning the race in a middlebox than a host
- We usually think of HTTPS as being something for web browsers
  - Many HTTPS sessions are not initiated by web browsers
  - HTTPS is the new TCP
- TLSA RRs for HTTPS exist in the wild ☺
- pcap-filter(7) for IPv6 still not at feature parity with IPv4 ☹
  - Berkeley Packet Filter (BPF) support for IPv6 is still primitive compared to IPv4
  - Requires more work in user space to support IPv6 (therefore slower)

Future Work / Crazy Ideas

- Danish
  - More testing and bug fixing
  - More work on cross compilation to different targets
    - Currently only testing on amd64
  - Add syslog support
  - Add support for other protocols
    - IMAP, POP3, client SMTP, DNS over TLS
- Add HTTPS DANE validation support to popular scripting languages
  - Perl, php, Ruby, Python, JS, libcurl, etc
- Survey HTTPS TLSA RRs in the wild

Thank You!

And please send me bug reports.

github.com/smutt/danish-rust

www.middlebox-dane.org/