TLS SNI Encryption

Author: Erik Nygren (erik+ietf@nygren.org)

2014-05-15

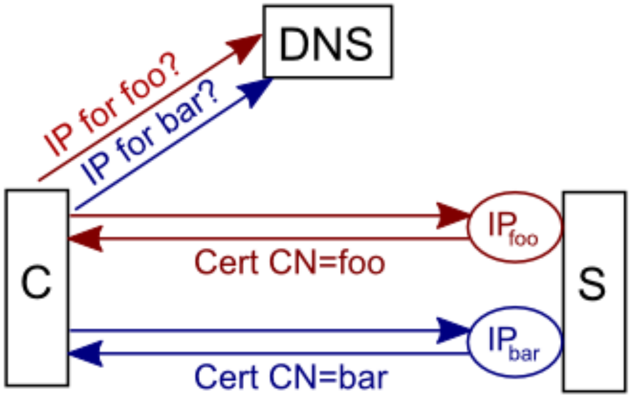
Why SNI?

- Multi-tenant hosting of sites
  - *Server needs to know which certificate to return*
  - *Load-balancers need to know where to steer TCP connections*
- Without SNI, must resort to an IP address per cert
  - *This means potentially hundreds of millions of IPv4 addresses wasted*
  - *IP-to-cert associations leaks information to passive eavesdroppers*
- With SNI:
  - *Not all servers behind an IP may be in the same security domain*
  - *(eg, with a TCP-terminating but not TLS-terminating demultiplexer)*

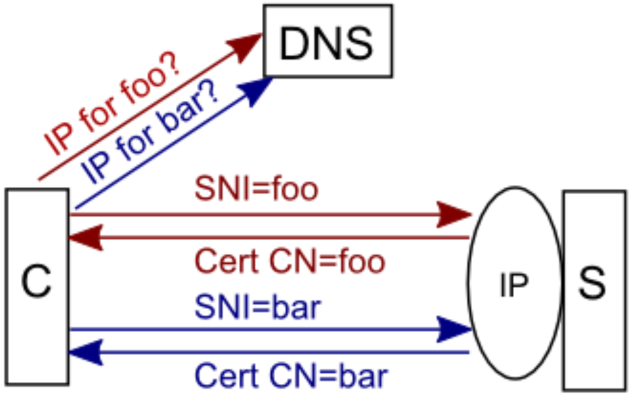
SNI Transition Challenge

- Transition challenge: only ~85% of clients send an SNI header
  - *Older Android, Windows XP, custom clients, and others do not send one*
  - *Requiring SNI isn't yet an option for many sites and blocks scaling to "TLS everywhere" with IPv4*
  - *Lack of incremental deployability is a problem*
- Without requiring SNI, waste millions of IPv4 addresses
  - *SAN and wildcard certs only help so much (e.g., with hundreds of thousands of hostnames)*

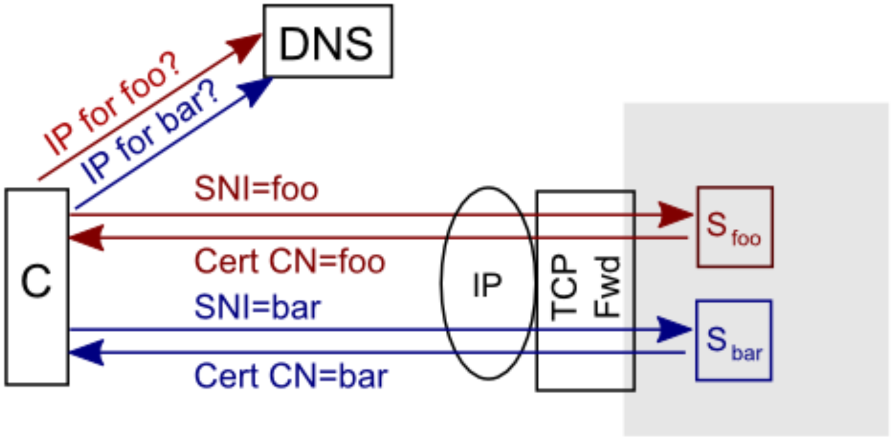
Without SNI



With SNI



With SNI



The Privacy Challenge

- Passive listeners (Eve) can observe which site (Host/ServerName) is being visited

- SNI primarily makes things worse for the cases where it is most needed (multi-tenant)
  - *Eve can just ask the IP for its certificate, so a privacy issue even without it*

- Even if the SNI is encrypted:
  - *Little-to-no benefit if DNS is in-the-clear*
  - *Doesn't stop traffic analysis due to nature of underlying HTTP flows*

- Requiring encrypted SNI server-side for all requests would actually make things better
  - *Likely an impossible transition challenge (no fall-back options)*

SNI Encryption Challenges

- Adds extra RTTs and extra complexity
  - *Current proposal also vulnerable to active attacks*
- Many resulting-but-necessary mitigations/work-arounds eliminate most privacy gains:
  - *Separate IPs-per-server*
  - *Identifier in request (eg, server_key_label in PredictedParameters, if poorly implemented)*
- Building features vulnerable to active attacks into TLS makes it hard to reason about
  - *May make more sense to put OE at a lower layer?*

Options for TLS 1.3 - part 1/2

- Leave SNI as-is in-the-clear for now
  - *Provides additional information to passive eavesdroppers for multi-tenant server IPs*
- Opportunistically encrypt SNI (as per draft-rescorla-tls13-new-flows)
  - *May force some sites to put off using TLS or to use server IPs per cert*
  - *May still provide too much information to passive eavesdroppers based on keyid in handshake*
  - *Adds additional RTTs and complexity in many cases*
  - *Information still leaked in the DNS until/unless it is secured*

Options for TLS 1.3 - part 2/2

- Use Opportunistic Encryption at a lower layer to protect handshake
  - *For example: tcpcrypt or ipsec*
  - *Benefit: having things vulnerable to active attacks in TLS makes it hard to reason about*
- Put handshake bootstrap into the DNS
  - *Opt-in (ie, requires putting records in the DNS)*
  - *Ties benefits to improving security of the DNS*
  - *May still provide too much information to passive eavesdroppers based on server_key_label in handshake*
  - *Does not add additional TCP roundtrips but may require additional DNS roundtrips*
  - *Requires careful design to enable deployability*

Appendix: Example Sketch of Handshake Bootstrap in the DNS

- New "Service Binding" ("B") record:

  _https._b.www.example.com B "service=server1.example.com,
  port=443, alpn=h2, handshake_params_key=68sgjbjfsd8fyjgbsgd7863,
  handshake_params_token=5sdfkj335, pri=5,
  dane_cert_name=version83.ca.example.com"