Encrypted SNI: Privacy and Security

No Name

October 22, 2019

1 Introduction

writeme

2 Encrypted SNI

There are several operational goals for Encrypted SNI [**?**], described below:

- Work with non-ESNI servers to avoid fallback.
- Mitigate replay attacks:
- Avoid widely-deployed shared secrets:
- Prevent SNI-based DoS attacks:
- Do not stick out:
- Forward secrecy:
- Proper security context:
- Split server spoofing:
- Multiple protocol support:

2.1 Draft-04 Design Overview

XXX: include figure of the protocol

2.2 Security and Privacy Goals

ESNI assumes a standard active and on-path Dolev-Yao attacker that can arbitrarily drop, tamper, replay, and forward messages from clients. Fundamentally, a TLS handshake that negotiates ESNI should leak no more information than one which did not negotiate ESNI in the presence of this adversary. This means there are at least two necessary requirements for ESNI:

Figure 1: Early ESNI Client Reaction Attack

1. SNI agreement: A successful TLS handshake implies agreement on the SNI transmitted. This means, among other things, that the client authenticated the server's certificate using the SNI, and that both client and server share the same view of the SNI negotiated.

2. SNI privacy: A successful TLS handshake that negotiates ESNI does so without leaking any information about the underlying SNI. Moreover, the SNI is known only to the client and server (or any recipient of the private ESNI key). We do not require forward secrecy for the SNI encryption.

We may optionally want to hide the fact that ESNI was negotiated, as per the "do not stick out" goal. However, this is primarily only deployment concern. Furthermore, we may also want to hide the fact that a client offered ESNI in its handshake. This may be useful for clients that wish to GREASE [**?**] the extension.

Early versions of ESNI did not achieve these goals. For example, the first version was vulnerable to a client reaction attack shown in Figure 1. The core problem was that servers did not signal to clients whether or not they processed the ESNI extension. This allowed any MITM to complete the handshake – prior to authentication – on behalf of the server with a certificate of its choosing. Adding a nonce to the server's response prohibits this as clients can then check whether or not the nonce is correct.

Despite this fix, draft-04 of ESNI does not achieve the necessary requirements stated above. To show why, we illustrate some more attacks on the protocol.

Probing Attacks : Differences in service configurations can leak information.

HelloRetryRequest Mix and Match : Attacker-chosen key share, client-chosen SNI.

Server Reaction Attacks : Use ticket and server reaction for dictionary attack.

The number of edge cases makes it clear that a formal model of the protocol is needed.

3 Core Protocol

At its core, ESNI is a protocol between a client and server that works as described in Figure **??**. It aims to provide the following guarantees:

- Client: TLS handshake secret known by entity which has the private handshake key share (y), corresponding PSK, private ESNI decryption key, and ENSI nonce.

- Server: TLS handshake secret known by entity which has the private handshake key share (x), corresponding PSK, and ENSI nonce.
- Client and server both agree on the same TLS handshake secret and transcript.

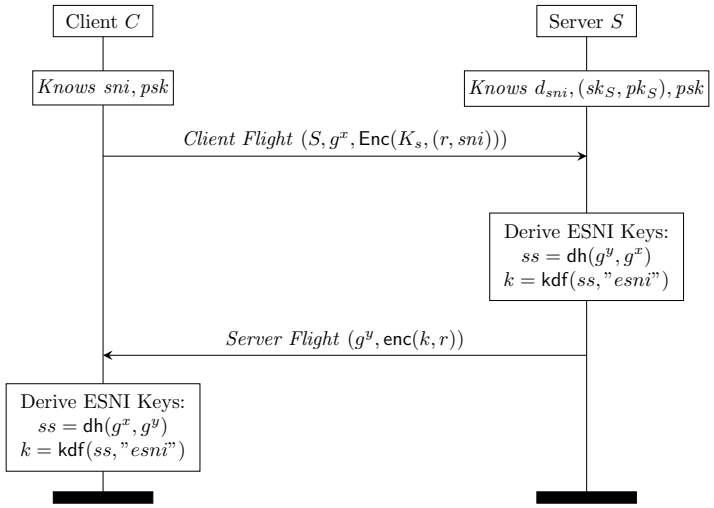

Figure 2: Simple ESNI Protocol without resumption or HelloRetryRequest support.

Moreover, these guarantees must hold for all TLS handshake patterns, including: normal handshakes, resumption handshakes, and HelloRetryRequest handshakes. Considering the full variant, there are five secrets that influence the handshake: private signing key, secret Diffie Hellman key shares, pre-shared key(s), ESNI decryption key (K_S), and the ESNI nonce.

These guarantees require the following properties:

- Backward binding: ESNI contents are bound to the entire ClientHello such that any modification is detectable by servers. ESNI contents are *backward bound* to a ClientHello if it is not possible to modify a CH in any way without causing an ESNI check to fail.
- Forward binding: TLS handshake secrets are bound to the ESNI contents such that knowledge of both ESNI secret(s) and one of the TLS key shares is needed to derive the handshake secrets. ESNI is *forward bound* if it is not possible to learn the TLS handshake secret without both the Diffie Hellman shared secret and ESNI shared secret.

As a consequence of forward binding and the need to interoperate with ESNI-incapable servers, we also require a signalling mechanism for clients to determine

whether or not the ESNI contents were used to protect the handshake secret. (Trial decryption is always an option, though other more practical solutions exist.)

4 Summary of Proposals

4.1 ESNI Proxy Transformation

XXX

4.2 ESNI PSK Binders and Key Schedule Injection

XXX

References