Public Review for multi-context TLS (mcTLS): Enabling Secure In-Network Functionality in TLS

Sharon Goldberg

As TLS (the end-to-end protocol that encrypts and authenticates network traffic) becomes increasingly ubiquitous in the Internet, tension has emerged between the in-network services provided by middleboxes and the privacy and security guarantees offered by TLS. How can a middlebox inspect or manipulate network traffic if it can't read the network traffic?

Several solutions have been proposed to address this problem, and this paper does a great job surveying them. Perhaps the most popular is to install a custom root certificate on the client, certifying a public key that corresponds to a private key that is known to the middlebox. The middlebox then uses this private key to perform a man-in-the-middle attack on the TLS connection: decrypting, introspecting, and then re-encrypting *all* of the TLS traffic that the client sends to *any* server. This situation is less than ideal; the middlebox has unrestrained access to all of the client's traffic, and the servers have no way to authorize (or prevent) the middlebox from introspecting on the connection.

This paper proposes a thought-provoking new point in the design space. Instead of giving a middlebox unrestrained access to all of a client's TLS traffic, this paper proposes mcTLS, a protocol allows client and server to *jointly* agree that a particular middlebox should be authorized to read or write only *certain parts* of the TLS traffic sent between them. To achieve this, the paper has two key contributions:

1) The notion of "contexts", or portions of the TLS traffic that are encrypted and/or authenticated under ephemeral sessions keys that are known to the middlebox. Thus, the middlebox can read (encrypt/decrypt) or write (encrypt/decrypt and authenticate) certain portions of the traffic, depending on the session keys that the client and server agree to provide to the middlebox.

2) A 3-way key exchange protocol that uses TLS certificates to allow client and server to set up ephemeral session keys with a middlebox.

The paper proposes new protocols to achieve the above, and implements and evaluates their performance. The PC agreed that the work brings a fresh and creative new perspectives to the problem, while raising several interesting and thorny questions.

First, what is the right security model for allowing middleboxes to act upon encrypted traffic? Is it necessary to provide a middlebox with full read/write access to portions of the TLS connection, or are there other workable models that might further limit the information exposed to the middlebox? Does permitting a middlebox to rewrite one context impact the privacy/authenticity of other contexts (analogously to the risks associated with mixing HTTP/HTTPS content)? In light of increasing public concern about surveillance and censorship, how should the end-users be informed about which parties are inspecting their network traffic, without eroding trust in TLS?

The paper also raises several technical questions. Perhaps the most challenging of these relates to the 3-way key-exchange protocol. Designing secure and efficient key-exchange protocols is far from trivial; indeed, while TLS/SSL has been around for decades, the community is still wrestling with the security of its key exchange protocol. Moreover, history has shown that changing even a tiny aspect of a cryptographic construction can have major security repercussions. This paper therefore has taken on a major technical challenge—designing a key-exchange protocol for not just two parties (client and server), but actually for three or more (client, server, and middlebox(es))! The protocol proposed in this paper is a variant of that used in TLS, with several modifications. There is still room for future work to rigorously analyze its security using cryptographic techniques or formal methods, or even to study alternate 3-way key exchange protocols. More narrowly, while Figure 1 in the paper proposes a key-exchange protocol designed to support "forward secrecy" (so session keys derived from long-term keys cannot be compromised even if a long-term key is compromised in the future), the implementation evaluated in Section 5 does not. Does forward secrecy impose a performance penalty?

In summary, we thought that the paper presents several fresh ideas that lead to several exciting new avenues for future work. We expect that this work will lead to further research into the growing tension between middleboxes and encrypted communications.