IKT-Sicherheitskonferenz 2022
Young Researchers' Day

Integration des Noise Schlüsselaustauschprotokolls in Tox

Tobias Buchberger

Competence Center for IT-Security
FH Campus Wien - University of Applied Sciences
Favoritenstrasse 226, 1100 Vienna, Austria

tobias.buchberger@fh-campuswien.ac.at

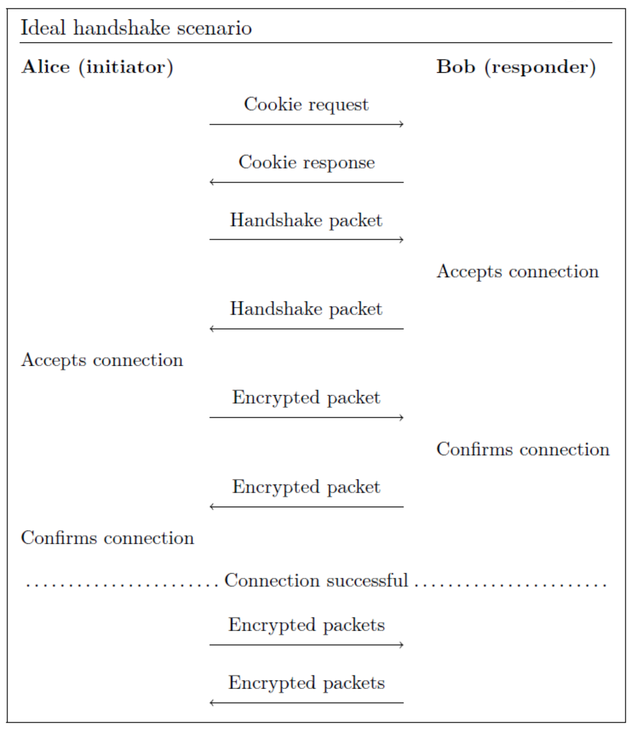
Einführung

Tox ist ein Peer-to-Peer Instant-Messaging Protokoll, das zum Ziel hat sichere Kommunikation zu ermöglichen. Das Tox-Protokoll ist in C implementiert und die Programmbibliothek heißt „c-toxcore“. Die Entwicklung von Tox hat im Anschluss an Edward Snowdens Veröffentlichungen hinsichtlich der Überwachung der internetbasierten Kommunikation durch die US National Security Agency (NSA) begonnen. Es ist als Ende-zu-Ende verschlüsselte und verteilte Alternative zu Skype gedacht. Tox ermöglicht unter anderem Instant-Messaging und Sprach- bzw. Videotelefonie.

„Key Compromise Impersonation“ Schwachstelle

Die kryptographischen Verfahren für den Schlüsselaustausch (X25519), die Authentifizierung (Poly1305) und die symmetrische Verschlüsselung (XSalsa20) sind aktueller Stand der Technik. Tox’ authentifizierter Schlüsselaustausch während des Handshakes funktioniert, ist aber ein selbsterstelltes kryptographisches Protokoll. Dieser Schlüsselaustausch ist für „Key Compromise Impersonation“-Angriffe (KCI) anfällig. Im Fall von Tox kann dies einem*r Angreifer*in, der*die den privaten Langzeit-X25519-Schlüssel einer Tox-Benutzerin Alice kompromittiert hat, unter bestimmten Voraussetzungen ermöglichen, sich gegenüber Alice als beliebige*r Tox-Benutzer*in auszugeben. Dies befähigt einen*r Angreifer*in Man-in-the-Middle-Attacken durchzuführen.

Ablauf eines Tox-Handshake zum Aufbau einer gesicherten Session:



Vereinfachter Schlüsselaustauch zwischen A und B:

- Message 1: $A \rightarrow B$

$$XAEAD(key = ECDH(S_A^{priv}, S_B^{pub}), payload = E_A^{pub})$$

- Message 2: $B \rightarrow A$

$$XAEAD(key = ECDH(S_B^{priv}, S_A^{pub}), payload = E_B^{pub})$$

- Session Key Derivation

$$ECDH(E_A^{priv}, E_B^{pub}) = ECDH(E_B^{priv}, E_A^{pub})$$

Vereinfachter KCI-Angriff von M auf A:

- Message 1: $M \rightarrow A$

$$XAEAD(key = ECDH(S_A^{priv}, S_B^{pub}), payload = E_M^{pub})$$

- Message 2: $A \rightarrow M$

$$XAEAD(key = ECDH(S_A^{priv}, S_B^{pub}), payload = E_A^{pub})$$

- Session Key Derivation

$$ECDH(E_A^{priv}, E_M^{pub}) = ECDH(E_M^{priv}, E_A^{pub})$$

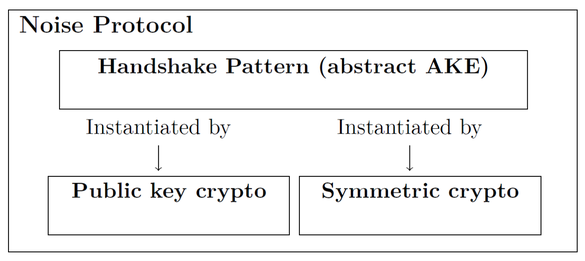
Implementierung eines sicheren Schlüsselaustausches basierend auf dem Noise Protocol Framework

Das Noise Protocol Framework (Noise) von Trevor Perrin ist dazu gedacht, für den Entwurf von Protokollen zum Aufbau sicherer Kommunikationskanäle basierend auf dem Diffie-Hellman-Schlüsselaustausch, verwendet zu werden. Noise stellt sogenannte Handshake-Patterns für unterschiedliche Anwendungsfälle zur Verfügung. Die Sicherheits-Eigenschaften dieser Patterns sind formal verifiziert und sind u.a. Forward Secrecy, glaubhafte Abstreitbarkeit und Resistenz gegen KCI. Noise-Protokolle werden z.B. bei WireGuard und WhatsApp eingesetzt.
Mithilfe von Noise wird ein neuer Schlüsselaustausch mit Resistenz gegen KCI für das Tox-Protokoll konzipiert. Der Noise-basierte Schlüsselaustausch verwendet das "IK" Pattern.
Der Name des resultierenden Noise-Protokolls ist Noise_IK_25519_ChaChaPoly_SHA512. Daher basiert auch der neue Schlüsselaustausch auf X25519, aber die Verschlüsselung während dem Schlüsselaustausch wird anstelle von XSalsa20 (welches von Noise nicht unterstützt wird) mit ChaCha20 durchgeführt.
Die Noise-C Programmbibliothek von Rhys Weatherley wird verwendet, um den neuen Schlüsselaustausch in c-toxcore umzusetzen. Zwei Funktionen werden zu Noise-C hinzugefügt, um die Schlüssel für symmetrische Verschlüsselung nach einem erfolgreichen Handshake abzufragen. Diese Schlüssel werden anschließend mit der bereits existierenden XSalsa20-Poly1305-Verschlüsselung verwendet, um Nachrichten zu senden.

Das Noise "IK" Handshake Pattern:

```
IK:
        <- s
        ...
        -> e, es, s, ss
        <- e, ee, se
```

Komponenten eines Noise-Protokolls:



Future Work

Als nächster Schritt ist angedacht das in Tox verwendete Noise-Protokoll explizit für Tox/c-toxcore zu entwickeln, anstatt die Noise-C Programmbibliothek zu verwenden. Dies würde die Abhängigkeit von Noise-C für c-toxcore entfernen und durch eine reduzierte Anzahl von Quellcodezeilen die Angriffsoberfläche verringern.

Weiterführende Informationen

Tox Website: https://tox.chat/
GitHub Repository der c-toxcore Programmbibliothek: https://github.com/TokTok/c-toxcore/
Noise Protocol Framework Spezifikation/Website: https://noiseprotocol.org/
GitHub Repository der Noise-C Programmbibliothek: https://github.com/rweather/noise-c

N. Unger, S. Dechand, J. Bonneau, S. Fahl, H. Perl, I. Goldberg, and M. Smith. SoK: Secure Messaging. In *2015 IEEE Symposium on Security and Privacy*, pages 232–249. 2015.

N. Kobeissi, G. Nicolas, and K. Bhargavan. Noise Explorer: Fully Automated Modeling and Verification For Arbitrary Noise Protocols. *In 2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 356–370. 2019.