Server Security Research

Ariel Amberden

9/29/2015

Apple's iMessage Encryption and Certificate Transparency

Post "Let's talk about iMessage (again)" obviates the need for data on the server to be kept confidential from the controller of the server. This requires encryption of data kept on the server and public keys to be maintained in a transparent way. Green describes Apple's encryption of iMessage as "good enough" in that it "makes it clear that iMessge encryption is 'end-to-end' and that decryption keys never leave the device." [1] For the time being, encryption for communication and storage on the iSafe server will implement Apple-style encryption, "messages are encryption with a combination of 1280-bit RSA public key encryption and 128-bit AES, and signed with ECDSA under a 256-bit NIST curve." [1] To prevent key substitution attacks, Green brings up CONIKS as the "most complete published variant of [Certificate Transparency]." [1] Google has a Certificate Transparency project that "[provides] an open framework for monitoring and auditing SSL certificates in nearly real time. Specifically, Certificate Transparency makes it possible to detect SSL certificates that have been mistakenly issued by a certificate authority or maliciously acquired from an otherwise unimpeachable certificate authority. It also makes it possible to identify certificate authorities that have gone rogue and are maliciously issuing certificates." [2] The group's website includes information on how to implement certificate transparency in TLS handshakes and how to verify that your server is responding with Certificate Transparency information.

OpenSSL TLS Protocol

OpenSSL will be used to implement the TLS protocol for all network traffic between the app on the device and the server. TLS 1.2 is the most recent version. "A connection always starts with a handshake between a client and a server. This handshake is intended to provide a secret key to both client and server that will be used to cipher the flow. ... The initial handshake can provide server authentication, client authentication or no authentication at all." [3] Server and client authentication will be provided for iSafe TLS implementation through trusted Certificate Authorities. There are several tutorial pages online demonstrating authenticated TLS implementations for the intended C# server platform. Support for secure transport is supported on iOS 5.0 and later. [4]

Zero Knowledge Proof

Green presents a zero knowledge problem framed in a deal in which Google must provide a three-colored graph for cell-tower coverage in which the client must be sure that the solution is correct and Google must convince the client without revealing anything about the solution. [5] In this context, a zero-knowledge protocol must have the following three properties [5],

1. Completeness. If Google is telling the truth, then they will eventually convince me (at least with high probability).
2. Soundness. Google can only convince me if they're actually telling the truth.
3. Zero-knowledgeness. (Yes it's really called this.) I don't learn anything else about Google's solution.

The need for Zero Knowledge is clear since, "Even as late as 2014, highly vulnerable client-to-server connections for services like Yahoo Mail were routinely transmitted in cleartext – meaning that they weren't just vulnerable to the NSA, but also to everyone on your local wireless network. And web-based connections were the good news. Even if you carefully checked your browser connections for HTTPS usage, proprietary extensions and mobile services would happily transmit data such as your contact list in the clear. If you noticed and shut down all of these weaknesses, it still wasn't enough – tech companies would naively transmit the same data through vulnerable, unencrypted inter-datacenter connections where the NSA could scoop them up yet again." [6]

List of security practices to be implemented

1. Encryption
   - 1.1. Encrypt using 1280-bit RSA public key encryption, 128-bit AES, signed with ECDSA under a 256-bit NIST curve
   - 1.2. Apply encryption at phone app level before sending to server. Only the user and financial institution connected with their account should have access to the private keys.
   - 1.3. Implement Zero Knowledge by encrypting data before it leaves the iSafe app, never storing passwords on the server, and prevent server knowledge of meta-data.
   - 1.4. Zero Knowledge prevents servers from storing or transmitting data in clear text so data would retain encryption in server breaches

2. OpenSSL TLS Protocol
   2.1. SSLv2 and SSLv3 must be disabled
   2.2. Control protocol downgrade
   2.3. Implement TLS 1.2 to handle secure communication with server
      2.3.1. Handshake between client and server to provide private key to both, authenticate both server and client
      2.3.2. Implement AES-128 as the cipher suite selection
3. Certificate Transparency
   3.1. Implement support of X509v3 Extension, TLS Extension, and OCSP Stapling for Signed Certificate Timestamp (SCT) in the TLS handshake
   3.2. Key substitution attacks
   3.3. Website spoofing attacks
   3.4. Server impersonation attacks
   3.5. Man-in-the-middle attacks

References

[1] M. Green, "Let's talk about imessage (again)." A Few Thoughts on Cryptographic Engineering, Blog, 2014. http://blog.cryptographyengineering.com/2015/09/lets-talk-about-imessage-again.html.

[2] G. C. T. team and contributors, "Certificate transparency." None, Unknown. http://www.certificate-transparency.org/.

[3] OpenSSL, "Ssl and tsl protocols." Wiki, Unknown. https://wiki.openssl.org/index.php/SSL_and_TLS_Protocols.

[4] Apple, "Cryptographic services guide." Mac Developer Library, 2014. https://developer.apple.com/library/mac/documentation/Security/Conceptual/cryptoservices/SecureNetworkCommunicationAPIs/SecureNetworkCommunicationAPIs.html.

[5] M. Green, "Zero knowledge proofs: An illustrated primer." A Few Thoughts on Cryptographic Engineering, Blog, 2014. http://blog.cryptographyengineering.com/2014/11/zero-knowledge-proofs-illustrated-primer.html.

[6] M. Green, "The network is hostile." A Few Thoughts on Cryptographic Engineering, Blog, 2014. http://blog.cryptographyengineering.com/2015/08/the-network-is-hostile.html.