"Detecting Man in the Middle Attacks on Ephemeral Diffie-Hellman without Relying on a Public Key Infrastructure in Real-Time Communications"

Alan Johnston, Avaya, Inc., Washington University in St. Louis
alan.b.johnston@gmail.com, alan@ese.wustl.edu
January 20, 2014

Abstract

With the recent revelations about pervasive surveillance on the Internet, there is renewed interest in techniques that protect against passive eavesdropping without relying on a Public Key Infrastructure (PKI). An ephemeral Diffie-Hellman (DH) key agreement can provide such protection, but (without authentication) the exchange is vulnerable to a Man in the Middle (MitM) attack. An example of a protocol that has MitM protection for a DH key agreement is ZRTP, RFC 6189, "ZRTP: Media Path Key Agreement for Unicast Secure RTP." ZRTP provides pervasive surveillance resistant security for Voice over IP (VoIP), video communication, and other real-time communication services. This paper describes the techniques used by ZRTP to detect MitM attacks, and explores whether these techniques could be used to develop a general MitM detection protocol to be used by other non-real-time communication protocols. An example of how ZRTP can provide MitM detection for another protocol, DTLS-SRTP, Datagram Transport Layer Security – Secure Real-time Transport Protocol, is given.

Introduction

The use of ephemeral Diffie-Hellman to protect against passive attackers is well known and well documented. Performing this technique without authenticating the DH key agreement is sometimes referred to as "opportunistic encryption," as described in [draft-farrelll-mpls-opportunistic-encrypt]. The main argument against this approach is its vulnerability to a Man in the Middle (MitM) attack. The basic arrangement is that while Alice and Bob are performing a DH key agreement in order to establish an encrypted connection between them, a third party, Carol, is performing a MitM attack on them by placing herself in the path so she can inspect, modify, or delete any packets exchanged between Alice and Bob. Alice and Bob think they are performing a DH exchange with each other, and the resulting generated secret is known only to them and can be used to derive keys for a secure connection between them. In reality, Carol as an active MitM attacker performs a DH exchange with Alice and a separate DH exchange with Bob. As a result, Carol knows both generated secrets, and hence can read all the encrypted packets sent by Bob and Alice by continuing to be in the middle, decrypting and re-encrypting packets on the fly.

While the principles of how to detect an active MitM attack are known, the detailed design of such approaches is not. There are two main approaches. One involves authenticating the other party in the DH exchange. If a MitM is present, Alice and Bob will be able to detect that they did not perform the DH exchange with each other, but instead with Carol. The second approach involves checking to see if the DH secret generated by Alice and Bob are the same. If a MitM is present, the secrets will be different, since they are different DH exchanges.

ZRTP [ZRTP] is a protocol for generating symmetric keys for Secure Real-Time Transport Protocol (SRTP) using a DH key agreement. Secure Real-Time Transport Protocol (SRTP) provides privacy and authentication for voice, video, and other real-time media sessions. ZRTP [ZRTP] is a protocol for generating symmetric keys for SRTP using a DH key agreement. ZRTP does not rely on PKIs or place any requirements on a signaling channel or protocol, such as Session Initiation Protocol (SIP) [SIP]. Instead, ZRTP uses the second approach to detect an active MitM attack, comparing the DH secret. ZRTP has a number of additional security features, including what it calls "best effort encryption," which is similar to the approach of IPsec Opportunistic Encryption [IPSEC-OE]. ZRTP has a number of open source implementations and commercial products.

Short Authentication Strings for Detecting MitM Attacks

ZRTP detects MitM attacks by checking to see whether the generated secret is the same for Alice and Bob. This is done using a Short Authentication String (SAS). The SAS is generated as a truncated hash of the generated secret. Since the SAS is just a hash of the generated secret, it is not a secret and can be displayed and logged by the protocol. If Alice and Bob can compare the SAS in an out-of-band way, they can detect the presence of a MitM. In this case, "out of band" doesn't mean simply sending it in a protocol message, since the MitM attacker could trivially change the SAS on the fly. Since ZRTP is a VoIP and video communications protocol, one way to compare the SAS is for the communicating parties to read aloud the displayed SAS at any time during the session. This is possible if the clients on each side calculate and render the SAS in the same way. Note that this does not require the parties to know each other's voice, since all they have to do is make sure that

the voice in which they hear the SAS is the same voice as the rest of the conversation. The humans can use any other approach that exchanges the SAS in a way that makes it difficult for the MitM attacker to modify it.

The SAS can only be "short" (e.g., 16 bits long compared to a full length hash of 256 or 512 bits) if a hash commitment of DH public values is performed. See [ZRTP] for a discussion of how the DH hash commitment works.

For protocols in which there aren't humans on both ends, the SAS can still be used to detect a MitM, although this would not typically be done in real time. If the SAS for each connection is logged and stored by their user clients, Alice and Bob can later audit the logs and determine whether a MitM attacker was present. As already mentioned above, this can be done provided both clients calculate and log the SAS in the same way. While this might seem of little value since it can only be done after the fact, a protocol that has the ability to detect MitM attacks can provide protection simply by deterring those who might mount such attacks by raising the risk of the attacker being detected.

Also, the comparison can be done by another protocol, as shown in the section "MitM Protection for Other Protocols."

Key Continuity

Key continuity can be a very useful tool in detecting MitM attacks if repeated connections are likely, and local storage is available (i.e. caching). There are a number of ways in which key continuity can be implemented. One way is key caching if public keys are reused between sessions, as used, for example, in Secure Shell, SSH [SSH]. Another approach is where instead of caching a previous public key, the protocol caches a hash of a previous shared secret. This is the approach used by ZRTP. This can be used to link the connection with previous connections using a hash chain.

For example, Alice and Bob have connected using ZRTP before, and have stored a hash of the secret used during that connection. The stored hash could be retrieved and used to authenticate the current connection as it is being established. This can let Alice know that she was connecting to the same Bob she connected with previously. If this chain of connections continues, a MitM attack could be detected by an unexplained break in the chain, if "Bob" does not know the previous secret known to Alice.

ZRTP takes this simple previous shared secret caching concept one step further. Key continuity caching of previous shared secrets can also be combined with the SAS, so that users don't have to always check the SAS every time to detect a MitM attack. During the DH exchange, ZRTP allows Alice and Bob to determine whether they share cached secrets. As a result, after a successful SAS comparison between Alice and Bob is performed, there is no need to check the SAS for subsequent connections as long as the hash chain is intact. If the hash chain ends, then Alice and Bob need to check the SAS again, and another hash chain is started.

User Interface Considerations

If an SAS is used, the calculation and rendering needs to be carefully designed and standardized. ZRTP uses two rendering schemes: hex digits or phonetically distinct words, based on a word list originally developed for PGP [PGP-WORDS]. Since more than one rendering is defined, the rendering must be agreed upon between the endpoints for each connection during the ZRTP handshake. Having one side render the SAS as hex digits and the other side render it using words will not work.

In some cases, an alternative identifier (besides that used/provided by the protocol) can be used by humans for endpoints. For example, a cached public key or previous shared secret cache could be labeled by the human when it is stored locally. For example, the first time Bob and Alice connect, Bob could choose this label for Alice's cached information: "Alice at Home." When Alice connects to Bob using this same device/endpoint, "Alice at Home" would be rendered to Bob.

Middlebox or Gateway Support

In general, security is strongest when it is defined and deployed end-to-end. With some protocols and applications, however, a server or gateway must be between the two ends, generically referred to as a middlebox. In cases where a middlebox must be involved, it is possible to designate a particular middlebox as "trusted" by a user with an enrollment/provisioning step that still achieves some end-to-end goodness. However, this should be avoided wherever possible.

This is very common in real-time communication, where the middlebox is known as a PBX or a Media Server. These devices often need to have access to the media flows, so the media packets cannot be encrypted end-to-end. For these situations, ZRTP allows the configuration of a "trusted MitM" which still permits MitM detection, even though there is more than one DH exchange in the path.

For example, both Alice and Bob might each have a trusted middlebox that they utilize. The resulting protocol connection between them would then have three legs to the connection: Bob to his middlebox, between the two middleboxes, and from Alice's middlebox to Alice. Alice and Bob can still validate the SAS, which would be the SAS calculated from the DH exchange between the two middleboxes. Of course, a compromise of either middlebox would render the entire connection vulnerable.

MitM Protection for Other Protocols

The previous sections have discussed the MitM detection techniques that ZRTP uses to protect the ZRTP protocol. Many of these techniques are general and could be applied to other protocols that use an ephemeral Diffie-Hellman key agreement as part of opportunistic encryption. This section discusses how ZRTP can provide MitM protection for another protocol [draft-johnston-rtcweb-zrtp], suggesting that the approach could be generalized to provide protection for different protocols in a variety of application scenarios.

If a protocol performs a DH exchange and logs the SAS in a standard way, it is possible for another protocol to provide MitM detection for this protocol. An example of this is the use of ZRTP to provide MitM detection for DTLS-SRTP key agreement [DTLS-SRTP] as used in WebRTC [WEBRTC]. DTLS-SRTP uses self-signed certificates that cannot be authenticated using a PKI. The original DTLS-SRTP specification relied on a signaling server-based PKI for authentication [SIP-IDENTITY]. This approach has been found to be un-deployable, and the STIR IETF Working Group [STIR-WG] is currently trying to fix this problem. For WebRTC, a new approach using an Identity Service [draft-ietf-rtcweb-security-arch] has been proposed to provide MitM protection for DTLS-SRTP. However, this approach would also rely on a PKI. In contrast, ZRTP can provide MitM protection for DTLS-SRTP without a PKI as described in [draft-johnston-rtcweb-zrtp] using the WebRTC data channel. In many cases, ZRTP can automatically detect a MitM in the DTLS-SRTP key exchange without even requiring the users to compare the SAS.

Conclusion

This paper has discussed the detection of MitM attackers for protocols using an ephemeral Diffie-Hellman key agreement to protect against pervasive surveillance. While the techniques, such as SAS and key continuity have been optimized specifically for VoIP and video communication applications, they can be effectively applied to other protocols. The MitM protections provided by ZRTP have been shown to be useful for providing MitM protection for the DTLS-SRTP protocol as used in WebRTC. In the future, a generic MitM detection protocol could be standardized for the IETF and be useful for other protocols implementing opportunistic encryption.

References

[draft-farrelll-mpls-opportunistic-encrypt] http://tools.ietf.org/html/draft-farrelll-mpls-opportunistic-encrypt

[ZRTP] http://tools.ietf.org/html/rfc6189

[SIP] http://tools.ietf.org/html/rfc3261

[IPSEC-OE] http://tools.ietf.org/html/rfc4322

[SSH] http://tools.ietf.org/html/rfc4253

[PGP-WORDS] https://en.wikipedia.org/wiki/PGP_word_list

[DTLS-SRTP] http://tools.ietf.org/html/rfc5763

[WEBRTC] http://tools.ietf.org/html/draft-ietf-rtcweb-overview

[SIP-IDENTITY] http://tools.ietf.org/html/rfc4474

[STIR-WG] https://tools.ietf.org/wg/stir/charters

[draft-ietf-rtcweb-security-arch] http://tools.ietf.org/html/draft-ietf-rtcweb-security-arch

[draft-johnston-rtcweb-zrtp] http://tools.ietf.org/html/draft-johnston-rtcweb-zrtp