Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 13, 2010

E. Rescorla
RTFM, Inc.
M. Ray
S. Dispensa
PhoneFactor
N. Oskov
Microsoft

November 09, 2009

Transport Layer Security (TLS) Renegotiation Indication Extension
draft-rescorla-tls-renegotiation-00.txt

Status of this Memo

Copyright Notice

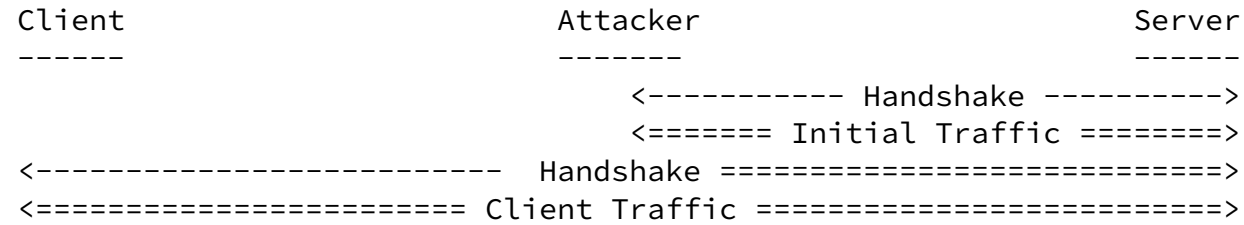
Abstract

SSL and TLS renegotiation are vulnerable to an attack in which the attacker forms a TLS connection with the target server, injects content of his choice, and then splices in a new TLS connection from a client. The server treats the client's initial TLS handshake as a renegotiation and thus believes that the initial data transmitted by the attacker is from the same entity as the subsequent client data. This draft defines a TLS extension to cryptographically tie renegotiations to the TLS connections they are being performed over, thus preventing this attack.

Table of Contents

1. Introduction

TLS [RFC5246] allows either the client or the server to initiate renegotiation--a new handshake which establishes new cryptographic parameters. Unfortunately, although the new handshake is carried out over the protected channel established by the original handshake, there is no cryptographic connection between the two. This creates the opportunity for an attack in which the attacker who can intercept a client's transport layer connection can inject traffic of his own as a prefix to the client's interaction with the server. The attack proceeds as shown below:



To start the attack, the attacker forms a TLS connection to the server (perhaps in response to an initial intercepted connection from the client). He then sends any traffic of his choice to the server. This may involve multiple requests and responses at the application layer, or may simply be a partial application layer request intended to prefix the client's data. This traffic is shown with == to indicate it is encrypted. He then allows the client's TLS handshake to proceed with the server. The handshake is in the clear to the attacker but encrypted over the attacker's channel to the server. Once the handshake has completed, the client communicates with the

server over the new channel. The attacker cannot read this traffic, but the server believes that the initial traffic to and from the attacker is the same as that to and from the client.

If certificate-based client authentication is used, the server will believe that the initial traffic corresponds to the authenticated client identity. Even without certificate-based authentication, a variety of attacks may be possible in which the attacker convinces the server to accept data from it as data from the client. For instance, if HTTPS [RFC2818] is in use with HTTP cookies [REF], the attacker may be able to generate a request of his choice validated by the client's cookie.

This attack can be prevented by cryptographically binding renegotiation handshakes to the enclosing TLS channel, thus allowing the server to differentiate renegotiation from initial negotiation, as well as preventing renegotiations from being spliced in between connections. An attempt by an attacker to inject himself as described above will result in a mismatch of the extension and can

thus be detected This document defines an extension that performs that cryptographic binding. The extension described here is similar to that used for TLS Channel Bindings [I-D.altman-tls-channel-bindings].

2. Conventions Used In This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Extension Definition

This document defines a new TLS extension: "renegotiation_info", which contains a cryptographic binding to the enclosing TLS connection (if any) for which the renegotiation is being performed. The "extension data" field of this extension contains a "Renegotiation_Info" structure:

```
          struct {
```

```
          opaque renegotiated_connection<0..255>;
        } Renegotiation_Info;
```

All TLS implementations SHOULD support this extension. TLS clients SHOULD generate it with every handshake and TLS servers SHOULD generate it in response to any client which offers it.

The contents of this extension are specified as follows.

- If this is the initial handshake for a connection, then this field is of zero length in both the ClientHello and the ServerHello.
- For ClientHellos which are renegotiating, this field contains the verify_data from the Finished message sent by the client on the immediately previous handshake. For current versions of TLS, this will be a 12-byte value. Note that this value is the "tls-unique" channel binding from [I-D.altman-tls-channel-bindings]
- For ServerHellos which are renegotiating, this field contains the concatenation of the verify_data values sent by the client and the server (in that order) on the immediately previous handshake. For current versions of TLS, this will be a 24-byte value.

The above rules apply even when TLS resumption is used.

Upon receipt of the "renegotiation_info" extension, implementations which support the extension MUST verify that it contains the correct contents as specified above. If the contents are incorrect, then it

MUST generate a fatal "handshake_failure" alert and terminate the connection. This allows two implementations both of which support the extension to safely renegotiate without fear of the above attack.

4. Backward Compatibility

Existing implementations which do not support this extension are widely deployed and in general must interoperate with newer implementations which do support it. This section describes considerations for backward compatible interoperation. [[OPEN ISSUE: The normative strength of these recommendations needs to be discussed.]]

4.1. Client Considerations

If a client offers the "renegotiation_info" extension and the server does not respond, then this indicates that the server either does not support the extension or is unwilling to use it. Because the above attack looks like a single handshake to the client, the client cannot determine whether the connection is under attack or not.

If clients wish to ensure that such attacks are impossible, they MUST terminate the connection immediately upon failure to receive the extension without completing the handshake. Otherwise, they may be performing client authentication and thus potentially authorizing the data already sent by the attacker even if the client itself sends no data. Note that initially deployment of this extension will be very sparse and thus choosing to terminate the connection immediately is likely to result in significant interoperability problems.

4.2. Server Considerations

If the client does not offer the "renegotiation_info" extension, then this indicates that the client does not support the extension or is unwilling to use it. Note that TLS does not permit servers to offer unsolicited extensions. However, because the above attack looks like two handshakes to the server, the server can safely continue the connection as long as it does not allow the client to rehandshake. If servers wish to ensure that such attacks are impossible they MUST NOT allow clients who do not offer the "renegotiation_info" extension to renegotiate with them and SHOULD respond to such requests with a "no_renegotiation" alert [RFC 5246 requires this alert to be at the "warning" level.] Servers SHOULD follow this behavior.

4.3. SSLv3

SSLv3 does not support extensions and thus it is not possible to securely renegotiate with SSLv3. Deployments wishing to renegotiate securely will need to upgrade to at least TLS 1.0.

5. Security Considerations

The extension described in this document prevents an attack on TLS. If this extension is not used, TLS renegotiation is subject to an attack in which the attacker can inject their own conversation with the TLS server as a prefix of the client's conversation. This attack is invisible to the client and looks like an ordinary renegotiation to the server. The extension defined in this document allows renegotiation to be performed safely. Servers SHOULD NOT allow clients to renegotiate without using this extension.

6. IANA Considerations

IANA [shall add/has added] the extension code point XXX [We request 0xff01, which has been used for prototype implementations] for the "renegotiation_info" extension to the TLS ExtensionType values registry.

7. Acknowledgements

This vulnerability was originally discovered by Marsh Ray. The general concept behind the extension described here was independently invented by Steve Dispensa, Nasko Oskov, and Eric Rescorla. Comments and refinements were received from Jesse Walker and the rest of the Project Mogul team.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

8.2. Informative References

[I-D.altman-tls-channel-bindings]
Altman, J., Williams, N., and L. Zhu, "Channel Bindings for TLS", draft-altman-tls-channel-bindings-07 (work in progress), October 2009.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

Authors' Addresses

Eric Rescorla
RTFM, Inc.
2064 Edgewood Drive
Palo Alto, CA 94303
USA

Email: ekr@rtfm.com

Marsh Ray
PhoneFactor
7301 W 129th Street
Overland Park, KS 66213
USA

Email: marsh@extendedsubset.com

Steve Dispensa
PhoneFactor
7301 W 129th Street
Overland Park, KS 66213
USA

Email: dispensa@phonefactor.com

Nasko
Microsoft
One Microsoft Way
Redmond, WA 98052
USA

Email: nasko.oskov@microsoft.com