Network Working Group	E. Rescorla	
Internet-Draft	RTFM, Inc.	
Intended status: Standards Track	M. Ray	
Expires: May 24, 2010	S. Dispensa	
	PhoneFactor	
	N. Oskov	
	Microsoft	
	November 20, 2009	

TOC

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

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.

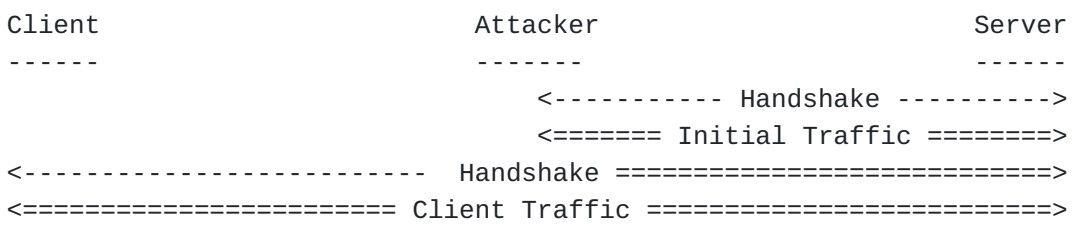
Status of this Memo

Table of Contents

TOC

1. Introduction

TLS [RFC5246] (Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," August 2008.) 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] (Rescorla, E., "HTTP Over TLS," May 2000.) 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] (Altman, J., Williams, N., and L. Zhu, “Channel Bindings for TLS,” March 2010.).

---

2. Conventions Used In This Document

TOC

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] (Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” March 1997.).

---

3. Extension Definition

TOC

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 the "renegotiated_connection" field is of zero length in both the ClientHello and the ServerHello. Thus, the entire encoding of the extension is: ff 01 00 01 00. The first two octets represent the extension type, the third and fourth octet the length of the extension itself, and the final octet the zero length byte for the "renegotiated_connection" field.

- 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] (Altman, J., Williams, N., and L. Zhu, “Channel Bindings for TLS,” March 2010.)

*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

TOC

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.

4.1. Client Considerations

TOC

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. Note, however, that merely because the server does not acknowledge the extension does not mean that it is vulnerable; it might choose to reject all rehandshakes. However, it is not possible for the client to determine purely via TLS mechanisms whether this is the case 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. However, it is expected that many TLS servers that do not support renegotiation (and thus are not vulnerable) will not support this extension either, so in general, such behavior would not work well.
While this specification does not require the client to send RI on initial handshakes, clients which choose not to do so have no mechanism for determining whether the server is operating in a vulnerable mode (and provide no such indication to the server) and are therefore relying entirely on the server refusing to renegotiate in the absence of the extension as opposed to explicitly indicating to the server that the initial handshake is in fact the first one on the connection.

4.1.1. Renegotiation Info Support Cipher Suite

TOC

[[OPEN ISSUE: Should this feature be added?]]
This document defines a special TLS cipher suite "RENEGOTIATE_WITH_RI", with code point 0xNN, 0xMM. Clients which support this extension MUST include this cipher suite in every handshake. When the server receives a Client Hello message which (1) contains this cipher suite (2) does not include RI and (3) the server thinks is doing renegotiation it MUST reject it with a fatal "handshake_failure" alert. Because servers ordinarily ignore unknown cipher suites, this cipher suite can be added safely on any handshake, thus allowing detection and prevention of the MITM attack described above. Servers MUST NOT select this cipher suite in any handshake, as it does not correspond to any valid cipher suite. Note that a minimal client which does not support renegotiation at all can simply use this cipher suite in all initial handshakes. Any compliant server will reject any (apparent) attempt at renegotiation by such a client. Clients which do support renegotiation MUST implement Section 3 (Extension Definition) as well.

4.2. Server Considerations

TOC

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. Broken Servers

TOC

SSLv3 does not support extensions and thus it is not possible to Both SSLv3 (at least later drafts) and TLS 1.0/1.1 require implementations to ignore data following the ClientHello (i.e., extensions) if they do not understand it. However, some SSLv3 and TLS 1.0 implementations incorrectly fail the handshake in such case. Client implementations

which offer extensions sometimes will respond to such failures by falling back to an extensionless mode. This practice can be exploited by a MITM to cause a client which would ordinarily offer the renegotiation extension not to do so. Note that this consideration does not apply to implementations which ignore extensions since the ordinarily TLS Finished messages protect that negotiation.

When combined with a server which allows renegotiation without the extension (which, per Section 5 (Security Considerations), is NOT RECOMMENDED) this allows a downgrade attack. Accordingly, clients which offer this extension SHOULD NOT fall back to extensionless modes upon handshake errors. Any SSLv3 or TLS implementation which chooses to address this issue by refusing to renegotiate at all MUST at minimum ensure that the extension is ignored (this is simply a restatement of existing requirements). Thus, any such handshake failure can be assumed to represent either an attack or a vulnerable server; in either case the best practice is not to continue the connection. Even servers which refuse to renegotiate SHOULD reply with an empty RI extension because this signals that they have been upgraded.

---

5. Security Considerations

TOC

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.

While this extension mitigates the man-in-the-middle attack described in the overview, it does not resolve all possible problems an application may face if it is unaware of renegotiation. It is possible that the authenticated identity of the server or client may change as a result of renegotiation.

By default, TLS implementations conforming to this document MUST verify that once an identity has been authenticated within the TLS handshake, it does not change on subsequent renegotiations. For certificate based cipher suites, this means bitwise equality of the end-entity certificate. If the other end attempts to authenticate with a different identity, the renegotiation MUST fail. If the server_name extension is used, it MUST NOT change when doing renegotiation.

A TLS library MAY provide a means for the application to allow identity and/or server_name changes across renegotiations, in which case the application is responsible for tracking the identity associated with data it is processing. This may require additional API facilities in the TLS library.

6. IANA Considerations

TOC

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.
IANA [shall add/has added] TLS cipher suite number 0xNN,0xMM with name RENEGOTIATE_WITH_RI to the TLS Cipher Suite registry.

7. Acknowledgements

TOC

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, the Project Mogul team as well as Pasio Eronen.

8. References

TOC

8.1. Normative References

TOC

[RFC2119]	Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).
[RFC5246]	Dierks, T. and E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.2,” RFC 5246, August 2008 (TXT).

8.2. Informative References

TOC

[I-D.altman-tls-channel-bindings]	Altman, J., Williams, N., and L. Zhu, “Channel Bindings for TLS,” draft-altman-tls-channel-bindings-10 (work in progress), March 2010 (TXT).
[RFC2818]	Rescorla, E., “HTTP Over TLS,” RFC 2818, May 2000 (TXT).

Authors' Addresses

TOC

	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