Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 5, 2011

P. Hoffman
VPN Consortium

December 2, 2010

Specifying That a Server Supports TLS
draft-hoffman-server-has-tls-01

Abstract

A server that hosts applications that can be run with or without TLS may want to communicate with clients whether the server is hosting an application only using TLS or also hosting the application without TLS. Many clients have a policy to try to set up a TLS session but fall back to insecure if the TLS session cannot e set up. If the server can securely communicate whether or not it can fall back to insecure tells such a client whether or not they should even try to set up an insecure session with the server. This document describes the use cases for this type of communication and a secure method for communicating that information.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on June 5, 2011.

1. Introduction

Most client-server application standardized in the IETF has two modes: an insecure mode that involves no authentication or integrity protection, and a secure mode that requires (at a minimum) that the client authenticate the server and set up a communication channel with integrity protection. In most cases, the secure mode is achieved by starting a TLS session and, when successful, running the insecure mode inside of it.

People within the IETF and application developers have historically had widely varying views on what a client should and should not do about the two modes. Phrases like "assured security" and "client flexibility" are used, often without clear definition. Deployed clients and servers from different vendors act differently for the two modes, often relegating the control of the two modes to "advanced" configuration options (if such control is given at all).

Section 2 of this document lays out the choices for clients and servers for handling the two modes in different circumstances, and gives specific semantics for each type of client and server. Section 3 gives a protocol for a domain owner to specify whether they offer one or both modes for any given application. Section 4 defines how to implement various policies using the protocol. Using the protocol given here, a server can completely specify what it offers and allows a client to reliably choose which mode it wants to use.

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 RFC 2119 [RFC2119].

2. Security Options for Clients and Servers

This section describes the different types of clients and servers that deal with insecure protocols that can be secured by wrapping the protocol in TLS. It also describes the types of security policies that those clients and servers can embody. It explicitly does not argue that one policy is better than another in any particular environment; instead, it assumes that the server operator and the client implementor (and, hopefully, the human operating the client)

can make that decision themselves if given the proper tools.

This discussion assumes a client-server protocol that is defined for an insecure fashion, and is also defined for a secure fashion that uses a TLS session for security. For example, "HTTP run over port 80" and "HTTP-in-TLS run over port 443" would meet this definition; "SMTP without STARTTLS" and "SMTP with STARTTLS" (see [RFC3207]) would also meet this definition. Some peer-to-peer protocols might meet this definition if the startup actions resemble the typical client-server interaction, but this discussion makes no extra attempt to cover such protocols.

Given a particular client application configuration, there are three interesting types of clients:

Insecure Only (CIO) -- The client is configured to only attempt communication for the application in its insecure form. For example, a POP client might be configured to only try insecure POP on port 110.

Secure Only (CSO) -- The client is configured to only attempt communication for the application in its secure, TLS-wrapped form. For example, a POP client might be configured to only try secure POP on port 995.

Allows Fallback From Secure to Insecure (CFB) -- The client is configured to attempt communication for the application in its secure, TLS-wrapped form, but if it fails to set up a TLS session, the client will attempt to attempt communication to the same server using the insecure form.

Given a particular server configuration, there are three interesting types of servers:

Insecure Only (SIO) -- The server responds without TLS on the main port for the application. For example, a host for a web server only responds to HTTP requests on port 80.

Secure Only (SSO) -- The server responds using TLS on the TLS-specific port for the application. For example, a host for a web server only responds to HTTP requests on port 443. Alternately, if the application supports in-band security update (such as STARTTTLS for SMTP), the server responds on the normal port, tries to establish a TLS session, and does not proceed with the protocol if a TLS session cannot be established.

Serves Both Secure and Insecure (SSB) -- The server responds without TLS on the main port for the application *and* responds using TLS on the TLS-specific port for the application, such as both ports 80 and 443 for HTTP. Alternately, if the application supports in-band security update (such as STARTTTLS for SMTP), the server responds on the normal port, tries to establish a TLS session, and proceeds with the normal protocol if a TLS session cannot be established.

In this taxonomy, a CIO can always communicate with an SIO and SSB. A CSO can communicate with an SSO, and can communicate with an SSB as long as the TLS session is set up successfully. A CFB can communicate with an SIO, an SSO, and an SSB.

Given this, a host that wants clients to only use the secure form of a protocol MUST only be configured to be SSO; a client that wants to only communicate with a server securely MUST only be configured to be CSO.

This taxonomy exposes a problem with the way that clients and servers interact today: a CIO that starts an insecure communication with a server, or a CFB that falls back to insecure communication with a server, has no idea whether the site they wish to communication with even hosts an insecure server. The server might be configured to be SIO or SSO or SSB, but the client cannot tell. If a CIO or CFB client knows ahead of time that a host did not support insecure communication, the client would not even start communication because it would either just waste time waiting for a timeout, or it would communicate with an impostor.

3. The HASTLS Resource Record

The HASTLS resource record type, whose value is TBD1, lists all of the pairs of insecure/secure port pairs that are served on the host named by the domain name. It only applies to applications that are secured with TLS, not to applications that have insecure and secure versions that use some other security protocol. It applies to TLS used over any transport (which will usually be TCP, but can also be SCTP and others), and also applies to DTLS.

Data in the HASTLS record MUST be received securely by a DNS requester, such as through validated DNSSEC.

The presentation form is:

```
IN HASTLS (portpair[1] ...)
```

where "portpair" consists of exactly four octets: two octets for the insecure port number (called "insecure-port"), and two for the secure port number (called "secure-port"). At least one portpair needs to be present, but many can be listed; not including any pairs of ports is explicitly undefined.

If a server does not offer one of the the two services, that service is indicated by port 0. For protocols that use in-band signaling for security upgrades, "insecure-port" and "secure-port" have the same value. Setting both the "insecure-port" and "secure-port" to 0 in a portpair is explicitly undefined.

For example, a server that offers SMTP both securely and insecurely, and offers HTTP only securely, would have a HASTLS record of:

```
www.example.com IN HASTLS (25 25 0 443)
```

[[NEED TO ADD: show example with hex values instead of decimal values.]]

4. Implementing Policy with HASTLS

Servers that have a policy to declare the server as SIO, SSO, or SSB can use HASTLS to announce that policy for each application it serves. A server whose policy is that it is an SIO would set the insecure-port to a non-zero number and the secure-port to 0. A server whose policy is that it is an SSO would set the insecure-port to 0 and the secure-port to a non-zero number. A server whose policy is that it is an SSB would set both the insecure-port and secure-port to a non-zero number.

The conformance requirements for a client using the HASTLS record depend on the policy configured for the client or the server:

- A client whose policy is that it is a CIO MUST NOT try to communicate insecurely with a server that has the insecure-port set to 0.

- A client whose policy is that it is a CSO MUST only try to communicate securely with a server that has the secure-port set to a non-zero number; it MUST NOT try to communicate with the server on the insecure-port value given.

- A client whose policy is that it is a CFB MUST NOT try to communicate securely with a server that has the secure-port set to 0.

- A client whose policy is that it is a CFB MUST NOT try to communicate insecurely with a server that has the insecure-port set to 0.

- A client whose policy is that it is a CFB trying to communicate with a server whose secure-port is set to a non-zero number SHOULD first try to communicate securely over the secure port unless it knows from other sources that the TLS session will not be set up properly.

5. IANA Considerations

This document requests that IANA allocate a new DNS resource record type called HASTLS from the data types range; it will have the value TBD1.

Submission template:

A. Submission Date: Date of this document
B. Submission Type: New RRTYPE
C. Contact Information for submitter: Author of this document
D. Motivation for the new RRTYPE application: Contents of this document
E. Description of the proposed RR type: Contents of this document
F. What existing RRTYPE or RRTYPEs come closest to filling that need and why are they unsatisfactory: None are even close to that given in this document.
G. What mnemonic is requested for the new RRTYPE (optional): HASTLS
H. Does the requested RRTYPE make use of any existing IANA Registry or require the creation of a new IANA sub-registry in DNS Parameters: No
I. Does the proposal require/expect any changes in DNS servers/resolvers that prevent the new type from being processed as an unknown RRTYPE (see [RFC3597]): No
J. Comments: None

6. Security Considerations

If the HASTLS information is received by the client system without security, an attacker could change the HASTLS information to fool the client into thinking that a host provides insecure application services and/or does not provide secure application services. Thus, cryptographic protection of the contents of the HASTLS information (such as with DNSSEC) is mandatory.

7. References

7.1. Normative References

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

7.2. Informative References

[RFC3207] Hoffman, P., "SMTP Service Extension for Secure SMTP over Transport Layer Security", RFC 3207, February 2002.

Author's Address

Paul Hoffman
VPN Consortium

Email: paul.hoffman@vpnc.org