Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 13, 2011

P. Hoffman
VPN Consortium

March 12, 2011

Specifying That a Server Supports TLS
draft-hoffman-server-has-tls-04

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 be 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 September 13, 2011.

1. Introduction

Most client-server applications standardized in the IETF have 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 [RFC5246] 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 to the user 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, and to specify a client policy preference. 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. This configuration may be offered for reasons such as if the client doesn't trust the CA that the server uses to identify itself.

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. A host for a web server that only responds to HTTP requests on port 80 is an example of this.

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.

It is expected that client configuration will be per-host. That is, a client that is CSO for some hosts might be CFB for other hosts. The server configuration, of course, applies to all clients accessing it.

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 only allows clients to 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.

Note that a host might want to serve both insecure and secure form of a protocol, but wants clients to only use the secure form. For example, the insecure form might immediately do an upgrade to the secure form, or it might do a protocol-based redirection to a server doing the secure form. Such a host will want to be able to indicate that, even if it has both secure and insecure ports for a protocol open, it wants clients that can be configured as CFB or CSO to only be configured as 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 any of 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.

Note that the protocol described here is not a discovery protocol. It is perfectly reasonable for a server to be running a service in an

insecure fashion, a secure fashion, or both, without using the protocol that is described here. The purpose of the protocol is to let a client find out, securely, whether a particular server protocol is being run securely, not whether it is being run at all.

3. The HASTLS Resource Record

The HASTLS resource record type, whose value is TBD1, lists an insecure/secure port pair that is served on the host named by the domain name for the application and protocol given in the query. 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.

The presentation format is:

_appname._protoname.hostname IN HASTLS ins-port sec-port pol-pref

The application name ("appname") and protocol name ("protoname") being queried are the same as are used in the SRV RRtype described in [RFC2782] The insecure port number (called "ins-port"), the secure port number (called "sec-port"), and the client policy preference (called "pol-pref") are each two-octet positive integers.

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, "ins-port" and "sec-port" have the same value. A HASTLS record MUST NOT have both the "ins-port" and "sec-port" set to 0.

A query for a particular application MAY return more than one HASTLS resource record, and conformant clients MUST be able to process multiple responses from a single query. For example, a site that offers HTTP on both port 80 and port 8080 might return two records, one for port 80 and its secure counterpart (if any), and one for protport 8080 and its secure counterpart (if any).

The HASTLS record is not useful for service discovery. Clients MUST NOT make any assumptions about an application for which there is not a HASTLS record; the lack of a HASTLS record for a particular application says nothing about whether or not the service is offered on the host on a specific port.

The client policy preference octet specifies the host's preference for client policy. It has two possible values:

0 -- The host's administrator has no client policy preference for this protocol.

1 -- If the client could be configured as either CFB or CSO for this protocol, and the host's administrator was able to configure the client for the protocol, that administrator would configure the client as CSO. Stated another way, the host's administrator does not want any CFB client to access the host for this protocol.

Note that specifying 1 for the client policy preference when a host does not support a protocol securely makes no sense, but it also does no harm. Further, for a host that is SSO, both policy preferences have an identical result.

For example, the server at www.example.com offers SMTP both securely and insecurely. The host's SMTP administrator has a client policy preference that CFB clients not access the host. The HASTLS record would look like:

```
_smtp._tcp.www.example.com IN HASTLS 25 25 1
```

Another example is the server at www.example.com offering HTTP only securely. The resulting HASTLS records could be either:

```
_http._tcp.www.example.com IN HASTLS 0 443 0
    or
_http._tcp.www.example.com IN HASTLS 0 443 1
```

[[NEED TO ADD: wire format.]]

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 ins-port to a non-zero number and the sec-port to 0. A server whose policy is that it is an SSO would set the ins-port to 0 and the sec-port to a non-zero number. A server whose policy is that it is an SSB would set both the ins-port and sec-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 communicating with a server that has set its client policy preference to 1 MUST NOT try to communicate insecurely with that server, even if the server has set the ins-port to a non-zero number; this is the equivalent of temporarily setting its policy

for the server to CSO for that application. This temporary policy based on the server's client policy preference overrides any others for the server.

- A client whose policy is that it is a CIO MUST NOT try to communicate insecurely with a server that has the ins-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 sec-port set to a non-zero number.
- A client whose policy is that it is a CSO MUST NOT try to communicate with the server if an ins-port value is given.
- A client whose policy is that it is a CFB MUST NOT try to communicate securely with a server that has the sec-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 ins-port set to 0.
- A client whose policy is that it is a CFB trying to communicate with a server whose sec-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

In order to prevent a man-in-the-middle (MITM) attack where the attacker can change DNS responses, the data in the HASTLS record needs to be received securely by a DNS requester, such as through validated DNSSEC. At this time, there is no common method for a application client to know whether or not the data it receives from the DNS has been protected with DNSSEC unless that application has a validating resolver, or has an API to the operating system and the operating system has a validating resolver.

Some MITM attacks can do a partial impersonation of the secure server if the client does not have good indications when an SSL connection is established (see, especially, the "sslstrip" attack described in various places in 2009). This protocol cannot protect against all such attacks because the attack is actually on the user interface, not the security of the TLS connection.

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.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

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

7.2. Informative References

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

Appendix A. Protocols That Have Both Second Ports and Upgrade Paths

There is a significant open issue with the current proposal: some protocols (hopefully few) have multiple ways of using TLS. One that was mentioned is HTTP, which has both "start TLS on TCP port 443" and "use STARTTLS on TCP port 80 as described in RFC 2817". POP and IMAP have the same issue.

One possibility is for this protocol to say "always use the second port" or "always use in-stream upgrade". This would give the client unambiguous instructions, but would not work for servers that do not implement the specified option.

Another possibility is to have the HASTLS response indicate which way the secure version is supported. This is more flexible than the first proposal, but also more complicated for the client. It also is possibly overkill if there are only two or three protocols of interest.

This issue needs to be resolved before the proposal is finished.

Author's Address

Paul Hoffman
VPN Consortium

Email: paul.hoffman@vpnc.org