Internet Draft
Expiration: November 2003
File: draft-ietf-rap-cops-tls-06.txt
Jesse Walker
Amol Kulkarni
Intel Corp.

COPS Over TLS

Last Updated: May 23, 2003

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of [RFC2026].

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

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].

Abstract

This memo describes how to use TLS to secure COPS connections over the Internet.

Please send comments on this document to the rap@ops.ietf.org mailing list.

Table Of Contents

1 Introduction

COPS [RFC2748] was designed to distribute clear-text policy information from a centralized Policy Decision Point (PDP) to a set of Policy Enforcement Points (PEP) in the Internet. COPS provides its own security mechanisms to protect the per-hop integrity of the deployed policy. However, the use of COPS for sensitive applications such as some types of security policy distribution requires additional security measures, such as data privacy. This is because some organizations find it necessary to hide some or all of their security policies, e.g., because policy distribution to devices such as mobile platforms can cross domain boundaries.

TLS [RFC2246] was designed to provide channel-oriented security. TLS standardizes SSL and may be used with any connection-oriented service. TLS provides mechanisms for both one- and two-way authentication, dynamic session keying, and data stream privacy and integrity.

This document describes how to use COPS over TLS. "COPS over TLS" is abbreviated COPS/TLS.

2 COPS Over TLS

COPS/TLS is very simple: use COPS over TLS similar to how you would use COPS over TCP (COPS/TCP). Apart from a specific procedure used to initialize the connection, there is no difference between COPS/TLS and COPS/TCP.

3 Separate Ports versus Upward Negotiation

There are two ways in which insecure and secure versions of the same protocol can be run simultaneously.

In the first method, the secure version of the protocol is also allocated a well-known port. This strategy of having well-known port numbers for both, the secure and insecure versions, is known as 'Separate Ports'. The clients requiring security can simply connect to the well-known secure port. The main advantage of this strategy is that it is very simple to implement, with no modifications needed to existing insecure implementations. Thus it is the most popular approach. The disadvantage, however, is that it doesn't scale well, with a new port required for each secure implementation. Hence, the IESG discourages designers from using the strategy.

The second method is known as 'Upward Negotiation'. In this method, the secure and insecure versions of the protocol run on the same port. The client connects to the server, both discover each others' capabilities, and start security negotiations if desired. This method usually requires some changes in the protocol being secured so that it can support the upward negotiation. There is also a high handshake overhead involved in this method.

3.1 The COPS/TLS approach

COPS/TLS uses a combination of both these approaches to achieve simultaneous operation with COPS/TCP. Initially, the authors had hoped to use the Separate Ports strategy for implementing COPS/TLS, however, due to the reluctance of the IESG to assign a well-known port, they settled on the following approach.

When the COPS/TLS server is initialized, it SHOULD bind to any non-well-known port of its choice. The standard COPS server running over TCP MUST know the TCP port on which COPS/TLS is running. How this is achieved is outside the scope of this document.

The system acting as the PEP also acts as the TLS client. It needs to first connect to the COPS/TCP server, from where it can be redirected to the COPS/TLS server.

During the initial negotiation with the COPS/TCP server, the Message Integrity Object MUST be used to authenticate the validity of the COPS messages. The use of the integrity object is described in [RFC2748]. How the keys indicated by the Integrity Object are shared between the Client and Server is outside the scope of this document.

3.2 Object Format and Error Codes

This section describes the ClientSI object sent in the ClientOpen message and the error codes the server returns.

3.2.1 The ClientSI object format

```
      0          1           2          3
+----------+----------+----------+----------+
|    Length (Octets)  | C-Num=9  | C-Type=2 |
+----------+----------+----------+----------+
|       Protocol      |        Flags        |
+----------+----------+----------+----------+
|          :          :          :          |
//         :          :          :         //
+----------+----------+----------+----------+
|       Protocol      |        Flags        |
+----------+----------+----------+----------+
```

Protocol:
     1 = TLS

Flags:

0 = Protocol Support Optional
1 = Protocol Support Required

This ClientSI object MUST be included with the ClientOpen message (Client Type = 0) when the client supports security. For each supported protocol, there MUST be a 32 bit Protocol+Flags pair appended to the object. At present, only one protocol (TLS) is described. However, the ClientSI object definition is general enough to allow addition of new protocols in the future.
If multiple protocols are supported by the client, it MUST ensure that no more than one has the 'Protocol Support Required' flag set. Note that it is also valid to mark all protocols as optional.

3.2.2 Error Codes and Sub-Codes

This section adds to, and modifies, the error codes described in section 2.2.8 (Error Object) of [RFC2748].

```
Error Code: 12 = Redirect to Preferred Server:
            Sub-code:
              0 = Regular redirect (no security necessary)
              1 = Use TLS
Error Code: 16 = Security Failure
            17 = Security Required
```

A new error sub-code has been added to the pre-existing error code 12. The sub-code informs the client that it SHOULD use TLS when connecting to the redirected server. In the future, more sub-codes may be added to specify additional protocols.

Error Code 17 SHOULD be used by either Client or Server if they require security but the other side doesn't support it.

4 Usage Scenarios

When the client needs to open a secure connection with the server, it SHOULD first connect to the non-secure port, and send a Client Open message with a ClientType=0.

The policies implemented on the client dictate whether security is mandatory or optional.

If the policies specify that security is mandatory, the above-mentioned ClientSI object MUST be included in the Client Open message. This object MUST list one protocol as required by setting the corresponding flag to 1.

If the policies do not explicitly specify that a secure connection is required, the client SHOULD include the ClientSI object, listing protocol support as optional.

Note that if the client's policies specifically prohibit a secure connection, it MAY attempt to establish a non-secure connection.

Based on the client's policies and the server's policy requirements for the client, the following scenarios occur:

4.1 Security Mandatory on both, Client and Server

The server MUST send a ClientClose message with a Redirect object, redirecting the client to the COPS/TLS secure port. Additionally, the error object included in the ClientClose message MUST have the error code = 12 and sub code = 1.

4.2 Security Mandatory on Client and Optional on Server

The server SHOULD send a ClientClose message with a Redirect object, redirecting the client to the COPS/TLS secure port. Additionally, the error object included in the ClientClose message MUST have the error code = 12 and sub code = 1.

If the server does not redirect the client to the secure port, it MUST send a ClientClose with the error code 16.

4.3 Security Optional on Client and Mandatory on Server

The server MUST send a ClientClose message with a Redirect object, redirecting the client to the COPS/TLS secure port. Additionally, the error object included in the ClientClose message MUST have the error code = 12 and sub code = 1.

4.4 Security Optional on both, Client and Server

The server SHOULD send a ClientClose message with a Redirect object, redirecting the client to the COPS/TLS secure port. Additionally, the error object included in the ClientClose message MUST have the error code = 12 and sub code = 1.

Optionally, the server MAY proceed to establish an insecure connection over COPS/TCP.

4.5 Security Mandatory on Client but not supported by Server

The server MUST send a ClientClose with the error code 16.

4.6 Security Optional on Client but not supported by Server

The server SHOULD attempt to establish a non-secure connection with the client.

4.7 Security Mandatory on Server but not supported by Client

If security is required by the server it MUST send a ClientClose with the error code 16.

4.8 Security Optional on Server but not supported by Client

The server it MAY attempt to establish a non-secure connection with the client.

5 Secure Connection Initiation

Once the PEP receives a redirect from the COPS/TCP server, it initiates a connection to the PDP to the secure COPS port. When this succeeds, the PEP system sends the TLS ClientHello to begin the TLS handshake. When the TLS handshake completes, the PEP MAY initiate the first COPS message. All COPS data MUST be sent as TLS "application data". Normal COPS behavior follows.

All PEP implementations of COPS/TLS MUST support an access control mechanism to identify authorized PDPs. This requirement provides a level of assurance that the policy arriving at the PEP is actually valid. The access control mechanism implemented is outside the scope of this document. PEP implementations SHOULD require the use of this access control mechanism for operation of COPS over TLS. When access control is enabled, the PEP implementation MUST NOT initiate COPS/TLS connections to systems not authorized as PDPs by the access control mechanism.

Similarly, PDP COPS/TLS implementations MUST support an access control mechanism permitting them to restrict their services to authorized PEP systems only. However, implementations MUST NOT require the use of an access control mechanism at the PDP, as organizations might not consider the types of policy being deployed as sensitive, and therefore do not need to incur the expense of managing credentials for the PEP systems. If access controls are used, however, the PDP implementation MUST terminate COPS/TLS connections from unauthorized PEP systems and log an error if an auditable logging mechanism is present.

6 Connection Closure

TLS provides facilities to securely close its connections. Reception of a valid closure alert assures an implementation that no further data will arrive on that connection. The TLS specification requires TLS implementations to initiate a closure alert exchange before closing a connection. It also permits TLS implementations to close connections without waiting to receive closure alerts from the peer, provided they send their own first. A connection closed in this way is known as an "incomplete close". TLS allows implementations to reuse the session in this case, but COPS/TLS makes no use of this capability.

A connection closed without first sending a closure alert is known as a "premature close". Note that a premature close does not call into question the security of the data already received, but simply indicates that subsequent data might have been truncated. Because

TLS is oblivious to COPS message boundaries, it is necessary to examine the COPS data itself (specifically the Message header) to determine whether truncation occurred.

6.1. PEP System Behavior

PEP implementations MUST treat premature closes as errors and any data received as potentially truncated. The COPS protocol allows the PEP system to find out whether truncation took place. A PEP system detecting an incomplete close SHOULD recover gracefully.

PEP systems MUST send a closure alert before closing the connection. Clients unprepared to receive any more data MAY choose not to wait for the PDP system's closure alert and simply close the connection, thus generating an incomplete close on the PDP side.

6.2. PDP System Behavior

COPS permits a PEP to close the connection at any time, and requires PDPs to recover gracefully. In particular, PDPs SHOULD be prepared to receive an incomplete close from the PEP, since a PEP often shuts down for operational reasons unrelated to the transfer of policy information between the PEP and PDP.

> Implementation note: The PDP ordinarily expects to be able to signal end of data by closing the connection. However, the PEP may have already sent the closure alert and dropped the connection.

PDP systems MUST attempt to initiate an exchange of closure alerts with the PEP system before closing the connection. PDP systems MAY close the connection after sending the closure alert, thus generating an incomplete close on the PEP side.

7 Port Number

The first data a PDP expects to receive from the PEP is a Client-Open message. The first data a TLS server (and hence a COPS/TLS server) expects to receive is the ClientHello. Consequently, COPS/TLS runs over a separate port in order to distinguish it from COPS alone. When COPS/TLS runs over a TCP/IP connection, the TCP port is any non-well-known port of the PDP's choice. This port MUST be communicated to the COPS/TCP server running on the well-known COPS TCP port. The PEP may use any TCP port. This does not preclude COPS/TLS from running over another transport. TLS only presumes a reliable connection-oriented data stream.

8 Endpoint Identification and Access Control

Implementations of COPS/TLS MUST use X.509 v3 certificates conforming to [RFC2459] to identify PDP and PEP systems. COPS/TLS systems MUST perform certificate verification processing conforming to [RFC2459]. In case the Certificate Authority cannot be accessed, communication MAY revert to insecure.

If a subjectAltName extension of type dNSName or iPAddress is present in the PDP's certificate, that MUST be used as the PDP identity. Otherwise, the most specific Common Name field in the Subject field of the certificate MUST be used.

Matching is performed using the matching rules specified by [RFC2459]. If more than one identity of a given type is present in the certificate (e.g. more than one dNSName name, a match in any one of the set is considered acceptable.), the COPS system uses the first name to match, except as noted below in the IP address checking requirements. Names may contain the wildcard character * which is considered to match any single domain name component or component fragment. For example, *.a.com matches foo.a.com but not bar.foo.a.com. f*.com matches foo.com but not foo.bar.com.

8.1 PDP Identity

Generally, COPS/TLS requests are generated by the PEP consulting bootstrap policy information identifying authorized PDPs. As a consequence, the hostname or IP address for the PDP is known to the PEP. How this bootstrap policy information arrives at the PEP is outside the scope of this document. However, all PEP implementations MUST provide a mechanism to securely deliver or configure the bootstrap policy. In particular, all PEP implementations MUST support a mechanism to securely acquire the signing certificate of the authorized certificate authorities issuing PDP certificates, and MUST support a mechanism to securely acquire an access control list or filter identifying its set of authorized PDPs.

PEP implementations that participate in multiple domains, such as those on mobile platforms, MAY use different certificate authorities and access control lists in each domain.

Organizations may choose to deliver some or all of the bootstrap policy configuration from an untrusted source, such as DHCP. In this circumstance, COPS over TLS provides no protection from attack when this untrusted source is compromised.

If the PDP hostname or IP address is available via the access control mechanism, the PEP MUST check it against the PDP's identity as presented in the PDP's TLS Certificate message.

In some cases the bootstrap policy will identify the authorized PDP only by an IP address of the PDP system. In this case, the subjectAltName MUST be present in the certificate, and it MUST include an iPAdress format matching the expected name of the policy server.

If the hostname of the PDP does not match the identity in the certificate, a PEP on a user oriented system MUST either notify the user (PEP systems MAY afford the user the opportunity to continue with the connection in any case) or terminate the connection with a bad certificate error. PEPs on unattended systems MUST log the error to an appropriate audit log (if available) and MUST terminate the connection (with a bad certificate error). Unattended PEP systems

MAY provide a configuration setting that disables this check, but then MUST provide a setting which enables it.

8.2 PEP Identity

When PEP systems are not access controlled, the PDP need have no external knowledge of what the PEP's identity ought to be and so checks are neither possible nor necessary. In this case, there is no requirement for PEP systems to register with a certificate authority, and COPS over TLS uses one-way authentication, of the PDP to the PEP.

When PEP systems are access controlled, PEPs must be PKI clients in the sense of [RFC2459]. In this case, COPS over TLS uses two-way authentication, and the PDP MUST perform the same identity checks for the PEPs as described above for the PDP.

When access controls are in effect at the PDP, PDP implementations MUST have a mechanism to securely acquire the signing certificates of the certificate authorities issuing certificates to any of the PEPs they support.

9 Other Considerations

9.1 Backward Compatibility

The client and server SHOULD be backward compatible with peers that do not support security. A client SHOULD be able to handle errors generated by a server which does not understand the ClientSI object mentioned above. Similarly, if a server receives a ClientOpen for Client type=0, which does not contain the ClientSI object, it SHOULD assume that the client wishes to open a non-secure connection and proceed accordingly.

9.2 IANA Considerations

This draft defines some new error codes and sub codes which require IANA approval. Section 3.2.2 has more details on these codes.

10 Security Considerations

This entire document concerns security.

11 Acknowledgements

This document freely plagiarizes and adapts Eric Rescorla's similar document [RFC2818] that specifies how HTTP runs over TLS. Discussions with David Durham, Scott Hahn and Ylian Sainte-Hillaire also lead to improvements in this document.

12 References

12.1 Normative References

[RFC2026] Bradner, S., "The Internet Standards Process - Revision 3", RFC 2026, October 1996

[RFC2119] Bradner, S., "Key Words for use in RFCs to indicate Requirement Levels", RFC 2119, March 1997.

[RFC2748] Durham, D., Boyle, J., Cohen, R., Herzog, R., Rajan, R., Sastry, A., "The COPS (Common Open Policy Service) Protocol", RFC 2748, January 200.

[RFC2459] Housley, R., Ford, W., Polk, W., Solo, D., "Internet Public Key Infrastructure: Part I: X.509 Certificate and CRL Profile", RFC 2459, January 1999.

[RFC2246] Dierks, T., Allen, C., "The TLS Protocol", RFC 2246, January 1999.

12.2 Informative References

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

13 Author Addresses

Jesse R. Walker
Intel Corporation
2111 N.E. 25th Avenue
Hillsboro, OR 97214
USA
jesse.walker[at]intel.com

Amol Kulkarni
Intel Corporation
JF3-206
2111 N.E. 25th Avenue
Hillsboro, OR 97214
USA
amol.kulkarni[at]intel.com