IAP: Intrusion Alert Protocol

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.

Abstract

The Intrusion Alert Protocol (IAP) is an application--level protocol for exchanging intrusion alert data between intrusion detection elements, notably sensor/analyzers and managers in IP networks. The protocol is designed to be independent of the type of data representation. The data definition and formatting model for alerts is described in companion documents of the working group.

1 Introduction

1.1 Purpose

The Intrusion Alert Protocol (IAP) is an application--level protocol for exchanging intrusion alert data. The protocol is designed to provide the necessary transport and security properties to allow sensitive alert data to be sent across IP networks. In addition, the
protocol is designed so that future extensions may use the application layer for configuring IDEF entities.

1.2 Transport layer protocol

IAP uses the transmission control protocol (TCP) [1] as its underlying transport layer mechanism. This protocol is used for a wide variety of IP services. It has a number of features such as congestion control, slow start, etc. that enable it to work over large networks with a wide variety of latency, bandwidth and congestion characteristics. TCP provides reliable and sequenced delivery of data between IP peers.

1.3 Terminology

Terminology is borrowed from [2].

1.4 Overall operation

IAP is primarily oriented towards supporting the transmission of alert data from an intrusion detection sensor/analyser that detects a potential intrusion, to a manager that displays it to a human, logs it to a database or takes appropriate action.

In the simplest case, a sensor/analyser (SA) sends alerts to a manager (M).

\[
\text{SA} \quad \text{------------------->} \quad \text{M}
\]

In a more complex situation, there are more than one intermediaries in the communication path. Two common forms of intermediary are:

Proxy A proxy is a forwarding agent which MAY do some rewriting of the content, and forward the message.

Gateway A gateway is used to translate messages from/to some native format (such as SNMPv3 UDP wire protocol).
A proxy may be used to relay two connections when the communication needs to pass through an intermediary such as a firewall.

\[ SA \longrightarrow P \longrightarrow G \longrightarrow M \]

Here, IDEF data is generated by SA and passed to a proxy P. P connects to a gateway G, which translates alerts into a native format to be consumed by the manager M. In this exchange, the connections between SA and P, and P and G use IAP. The connection between G and M uses the native protocol supported by M.

IAP communication takes place over an IP network using the transport control protocol (TCP). To connect with other networks, gateways should be used to transform protocol data into the native representation.

2 IAP notation

2.1 Entity roles

2.1.1 Server role

An IDEF entity acts in an IAP server role if it serves as the consumer of alert data sent over IAP, that is, a manager in an IDEF setting.

2.1.2 Client role

An IDEF entity acts in an IAP client role if it serves as a producer of alert data sent over IAP, that is, a sensor/analyzer in an IDEF setting.

2.1.3 Proxy role

An IDEF entity in a proxy role does not have an IAP identity. It acts as a relay of messages without understanding their content. It MAY do some rewriting of the content, but in a manner that does not impact the security properties of alerts.

2.2 Augmented BNF
We use the augmented BNF definitions of [3].

3 Protocol parameters

IAP uses a <major>.<minor> numbering scheme to indicate versions of the protocol. The minor number is changed when updates to the protocol add features that do not change the parsing algorithm. The major number is changed when the parsing algorithm is modified.

This document describes version 0.2 of IAP. The version string for this is iap-version: IAP/0.2.

3.1 Media Types

IAP uses Internet Media Types [4] to denote the type of alert data. Media types are used to define the encapsulation of data, in a manner that can be extended without requiring changes to the application protocol.

The only media type used by IAP/0.2 is application/x-idef-alert. It is expected that the IANA will allocate a registered media type for the IDEF alert format. IAP entities MUST accept data sent in this format.

4 Structure of IAP protocol sessions

An IAP protocol is split into four phases.

4.1 Protocol Setup

This is the first phase after a TCP session is established. In this phase, the two parties set up the transport parameters of the protocol. An IAP entity in a proxy role MAY rewrite content to set up the protocol in this phase. This phase uses a request--response form. The primary reason for this phase is to allow the protocol to be proxied in an effective manner.

Response codes are borrowed from the hypertext transfer protocol [3].

In its most primitive form, this negotiation is initiated by an IAP client (sensor/analyzer) by issuing an iap-connect-request command.

A corresponding IAP server (manager) MAY choose to accept this connection, and respond using an iap-response command, with the status code set to 200 to denote success. The parties can then proceed to the next phase.

Alternatively, the entity MAY reject the connection request by
setting the status code to 4xx to denote failure. In particular, the 403 Forbidden command MAY be used by the entity in server role.

An intermediate entity in a proxy role MAY, upon receiving the request, rewrite the iap-connect-request command. A proxy MAY append the iap-proxy-via command to the connection request before passing it on to the receiving entity. The version string used by the proxy MUST be the same as the one specified in iap-connect-request. This is important to ensure that newer versions of IAP can work with older proxies.

A proxy SHOULD relay the server’s response back to the client without rewriting it. A proxy MUST verify that existing proxy-via headers in the request don’t match its own identity in order to limit the damage from misconfigured proxies. Otherwise, it MUST send a bad gateway (502) response to the client that requested the connection.

4.2 Security Setup

After the protocol setup phase, an entity in a proxy role MUST be transparent. It MUST not alter any data being relayed between the client and server.

A single request--response pair is used to upgrade the security of a connected transport. An entity in a client role, upon receiving an iap-connect-response command without any errors, issues a iap-upgrade-request command.

A server should expect the iap-upgrade-request command after sending an iap-response reply indicating acceptance of the connection. The server SHOULD terminate if the request is not received, or some other request is received. Upon receiving the request, it SHOULD send an iap-response reply with the status code set to 101 to indicate acceptance of the upgrade.

It MAY also send a 500 to denote server configuration error, if for instance, it is not yet configured and does not have a certificate.

Once the client receives an iap-response message indicating success of its request to upgrade the security of the connection, it initiates the TLS 1.0 handshake protocol [5]. This protocol negotiates the protocol version, cryptographic algorithms, mutual authentication and generates shared secrets for the next phase.

If the parties complete the TLS handshake protocol successfully, they enter exchange final setup request--response pairs, using the TLS record protocol. These pairs are exchanged after a successful handshake and is used by the parties to verify connection parameters.
Any entities in a proxy role MUST forward data transparently. End entities detect any attempts to manipulate or inject messages into the data stream.

First, the client informs the server about the IAP version it intends to employ for the session using a iap-version-verify message. The server matches this against the version in the iap-connect-request header, and issues a iap-response reply. The server then sends its own iap-version-verify message which is verified by the client, which responds with a iap-response reply. The status code 200 SHOULD be used to indicate success.

The protocol and security setup phases are now complete, and the channel is ready to communicate alert data.

4.3 Secured data transport

This is the main phase of the protocol. In this phase, encoded IDEF alerts are sent by the client (sensor/analyzer) to the server (manager) over the TLS record layer.

In addition to data in the form x-idef-alert, compatible clients and servers MAY send other data using this transport. A peer, upon receipt of data that it is unable to decode (unknown IAP content type), SHOULD reject the data. It MAY choose not to terminate the connection. This allows clients (analyzers) supporting richer content types to communicate with legacy servers (managers).

4.4 Termination

Termination can be initiated by either peer by sending a TLS close-notify alert (section 7.2.1). The recipient, on receipt of this alert, should in turn respond with a close-notify alert and the parties can close the connection gracefully.

4.5 Example

In this example, an analyzer A is configured to connect to proxy P. P connects to a manager M to deliver the alerts. The following diagram depicts the abstract message flow for the case where there are no errors, and a connection is set up.

```
A                           P                               M
--------------->
iap-connect-request
```

Gupta
5 IAP Wire Protocol

IAP uses a subset of the HTTP/1.1 syntax to send IDEF alerts. The request--response protocol is modeled on HTTP, with the exception
that the each request and response must be prefixed with the IAP version number during the setup phase.

All requests must be responded to using an iap-response message indicating whether the recipient was able to successfully interpret (and act on) the request. Response codes are borrowed from HTTP/1.1.

5.1 Alert transfer encoding

Alert content is transferred using a content length delimiter in the header, following HTTP/1.1. The content length header is mandatory.

5.2 Syntax

In the wire protocol, requests and responses are terminated by a pair of CRLF sequences, following HTTP/1.1. The following definitions from [3] are used.

+ host
  + DIGIT
  + Chunked-Body

IAP entities MUST not use any trailer fields in a chunked body.

An IAP message is denoted by iap-message.

iap-message =
  ( iap-connect-request |
  iap-upgrade-request |
  iap-verify |
  iap-content |
  iap-response )

The version of the protocol is denoted by iap-t-version

iap-t-version = "IAP/0.2"
A client or server can denote its intent to send data as part of the iap-verify message:

iap-t-sender = "IAP-Role: Sender" CRLF

5.3 Protocol messages
5.3.1 Responses

An iap-response denotes the status code returned by an IAP entity in response to a request.

iap-response = iap-t-version SP 3DIGIT CRLF

5.3.2 Connection Request

A iap-connect-request denotes a request message sent by an IAP client to establish an IAP protocol connection.

iap-connect-request
  = iap-t-connect-request
     *( iap-t-via )

iap-t-connect-request
  = iap-t-version SP "CONNECT" SP host CRLF

iap-t-via
  = iap-t-version SP "VIA" SP host CRLF
5.3.3 Upgrade Request

An iap-upgrade-request contains a notification from the client that it wishes to upgrade the security of the connection.

iap-upgrade-request
 = iap-t-version SP
   "Upgrade: TLS/1.0" CRLF

5.3.4 Verify

An iap-verify request contains a notification requesting that the peer verify the version of IAP being used, and optionally, the mode in which the entity wishes to operate (send requests or receive responses).

iap-verify
 = iap-t-version SP
   "IAP-Version: 0.2" CRLF
   [ iap-t-sender ]

5.3.5 Alert Content

The iap-content message is an encoded alert sent using IAP.

iap-content
 = iap-t-content-header
   CRLF
   iap-t-content-body
   CRLF
iap-t-content-header
  = iap-content-type
  iap-content-length

iap-content-type  = "Content-Type:" SP
  "application/x-idef-alert"
  CRLF

iap-content-length
  = "Content-Length: " SP
  1*DIGIT CRLF

iap-t-content-body
  = *OCTET CRLF

5.4 Example

Here is a transcript of a scenario in which an IDEF sensor/analyzer A
wishes to send alerts to manager M. The proxy P mediates this
connection. After the connection has been set up, A sends an IDEF
alert 64 octets in length with each octet set to 0xFF.

A                        P                   M
iap-connect-request
--------------->
IAP/0.2 CONNECT M.DOM.ORG CRLF
CRLF

iap-connect-request
--------------->
IAP/0.2 CONNECT M.DOM.ORG CRLF
IAP/0.2 VIA P.DOM.ORG CRLF
CRLF

iap-response
<---------------
IAP/0.2 200 CRLF
CRLF

iap-response
<---------------
IAP/0.2 200 CRLF
CRLF
[proxy is now a transparent forwarding agent]
iap-upgrade-request
----------------->
IAP/0.2 Upgrade: TLS/1.0 CRLF
CRLF

iap-response
<--------------
IAP/0.2 101 CRLF
CRLF

(TLS handshake negotiation)

[TLS handshake completed: data sent using the TLS record layer]

iap-verify
<---------------
IAP/0.2 IAP-Version: 0.2 CRLF
IAP-Role: Sender CRLF
CRLF

iap-response
<--------------
IAP/0.2 200 CRLF
CRLF

iap-verify
<---------------
IAP/0.2 IAP-Version: 0.2 CRLF
CRLF

iap-response
<--------------
IAP/0.2 200 CRLF
CRLF

iap-content
<---------------
Content-Type: application/x-idef-alert CRLF
Content-Length: 64 CRLF
CRLF (end of header)
64 * 0xFF octet (IDEF alert data)
CRLF (end of alert body)
CRLF (end of iap-message)

iap-response
<--------------
IAP/0.2 200 CRLF
CRLF
6 Implementation Considerations

6.1 TCP connection initiation

The entity that initiates a TCP connection will be variable, and dependent on the exact deployment model. A common scenario is that of sensor/analyzers outside a security perimeter, with the manager inside. In such configurations, the manager MAY initiate the connection in line with perimeter security requirements.

Another common scenario is that of remote sensor/analyzers being managed by a service provider. In this case, the sensor/analyzer MAY contact a proxy on the perimeter, which in turn connects to the remote manager in a service provider’s network.

The communications protocol should allow chained proxies to carry IAP data across multiple security perimeters.

The first message is sent by the client (sensor/analyzer) regardless of whether it or the manager initiated the TCP session. In cases where a client and server wish to establish duplex connections, such as for configuration, the peers SHOULD use two separate TCP connections. The server SHOULD add a iap-sender message to denote that it wishes to send requests for such configuration connections. The peers should then expect requests to come from the server, which get responded to by the client. This allows the protocol to support bidirectional request-response systems in a simple manner.

6.2 Urgent data

Urgent data at the TCP layer MUST NOT be used by an entity using IAP. Endpoints SHOULD terminate a connection upon receipt of urgent data.

6.3 TLS/1.0 protocol

The TLS 1.0 protocol MUST be used. An IDEF entity MUST not allow older protocol HELLO messages, and reject attempts to negotiate an older version of the protocol. The following TLS ciphersuite MUST be supported in line with recommendations from the tls working group:

+ TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA

The recommendations in sections 2.1 and 2.2 of [6] must be followed by IAP client and server implementations.

6.4 Mandatory client certificates
In line with the requirement for strong mutual authentication, client certificates for sensor/analyzers acting in an IAP client role are mandatory, and an IAP entity MUST verify the peer’s certificate’s content during the TLS handshake.

6.5 Certificate conventions

One or more of the following key usage extensions MUST be specified in X.509v3 certificates issued to an IAP client or server entity:

+ IAP_ALERT_GENERATOR
+ IAP_ALERT_CONSUMER
+ IAP_IDEF_CONFIGURATOR

Object identifiers (OIDs) for these extensions will be specified in a companion document. The presence of the key usage extension means that the signer believes that the holder of the certificate is allowed to function in the corresponding IAP role. The extension MUST be marked critical to reduce the accidental usage of the certificate for other purposes.

An entity in a IAP server role MUST have the IAP_ALERT_CONSUMER key usage extension in its certificate. Similarly, an entity in a IAP client role MUST have the IAP_ALERT_GENERATOR key usage extension in its certificate.

6.6 Verification of peer identity

As the peer identity is initially sent in the clear, it is essential that the IAP client and server entities verify the identity of their peer using criteria based on their public key certificates. Also, the version of the protocol MUST be verified to stop protocol downgrade attacks. The mechanism specified in section 2.4 of [6] for verifying peer certificates must be followed.

6.7 TLS session resumption

The entities MUST be configured to support TLS session resumption. Because of the possibility of external entities maliciously terminating IAP sessions, clients and servers MAY attempt to resume a session even if the TLS close-notify alert was not received before the transport closed.

7 Security Considerations

As IAP is intended to be used for carrying security-sensitive data,
we will list a number of security considerations.

7.1 Reliable and sequenced delivery

Although reliable and sequenced delivery can be built on top of UDP, this usually reimplements some of the protocol features of TCP. Certain deployment scenarios may prefer fast unreliable delivery with the manager doing a "best effort" attempt to keep up with the flow of alerts. This proposal does not address such a scenario. One potential alternative for this scenario is to use the SNMP trap as a means to represent the alert. We remark that the above scenario is at odds with a number of items in section 6 of the requirements document of the working group.

7.2 TCP handshake

TCP uses a three--message handshake mechanism to set up connections. This opens the protocol up to certain well-known denial of service attacks. This protocol does not address this vulnerability. The effect of such attacks can be mitigated by a number of techniques, such as restricting the set of peer IP addresses allowed to connect, etc.

7.3 Key Management

For a public--key based communications model to be useful, good key management principles must be used for the lifecycle of public key certificates. The pkix working group of the IETF is defining mechanisms that can be used for this purpose.

7.4 Message length

Traffic analysis may allow an observer to induce the type of alert from the size of the message. If this is a threat, IDEF entities SHOULD pad their data so that it observes some known distribution (such as the uniform distribution) over time.

7.5 Proxy considerations

As the proxy transparently forwards encrypted traffic after connections are established, it is prudent to deploy the proxy in a manner that it can’t be used to violate the perimeter security policy.

8 Acknowledgements

This document makes heavy use of prior work in the IETF on HTTP, MIME and TLS. Their effort is gratefully acknowledged. Members of the
IETF’s intrusion detection working group (idwg) have made extensive comments that are reflected in the draft.

9 Bibliography


A Author’s Address

Dipankar Gupta
Hewlett-Packard
690 E Middlefield Road, MS 31R
Mountain View, CA 94043, USA
Fax: +1(650)919-8066
Email: dg@mayfield.hp.com

B Full Copyright Statement

Copyright (C) The Internet Society (1999). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are
included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Table of Contents

1  Introduction ........................................ 1
1.1 Purpose ............................................ 1
1.2 Transport layer protocol ............................ 2
1.3 Terminology ........................................ 2
1.4 Overall operation ................................. 2
2  IAP notation ........................................ 3
2.1 Entity roles ........................................ 3
2.1.1 Server role ..................................... 3
2.1.2 Client role ..................................... 3
2.1.3 Proxy role ..................................... 3
2.2 Augmented BNF ..................................... 3
<p>| 3 | Protocol parameters ........................................ | 4 |
| 3.1 | Media Types ............................................... | 4 |
| 4 | Structure of IAP protocol sessions .................... | 4 |
| 4.1 | Protocol Setup ............................................ | 4 |
| 4.2 | Security Setup ............................................ | 5 |
| 4.3 | Secured data transport ................................... | 6 |
| 4.4 | Termination ................................................ | 6 |
| 4.5 | Example .................................................... | 6 |
| 5 | IAP Wire Protocol ......................................... | 7 |
| 5.1 | Alert transfer encoding .................................. | 8 |
| 5.2 | Syntax ....................................................... | 8 |
| 5.3 | Protocol messages ......................................... | 9 |
| 5.3.1 | Responses .................................................. | 9 |
| 5.3.2 | Connection Request ....................................... | 9 |
| 5.3.3 | Upgrade Request ........................................... | 10 |
| 5.3.4 | Verify ...................................................... | 10 |
| 5.3.5 | Alert Content ............................................. | 10 |
| 5.4 | Example ..................................................... | 11 |
| 6 | Implementation Considerations ........................... | 13 |
| 6.1 | TCP connection initiation .................................. | 13 |
| 6.2 | Urgent data ................................................ | 13 |
| 6.3 | TLS/1.0 protocol .......................................... | 13 |
| 6.4 | Mandatory client certificates ........................... | 13 |
| 6.5 | Certificate conventions ................................. | 14 |</p>
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.6</td>
<td>Verification of peer identity</td>
<td>14</td>
</tr>
<tr>
<td>6.7</td>
<td>TLS session resumption</td>
<td>14</td>
</tr>
<tr>
<td>7</td>
<td>Security Considerations</td>
<td>14</td>
</tr>
<tr>
<td>7.1</td>
<td>Reliable and sequenced delivery</td>
<td>15</td>
</tr>
<tr>
<td>7.2</td>
<td>TCP handshake</td>
<td>15</td>
</tr>
<tr>
<td>7.3</td>
<td>Key Management</td>
<td>15</td>
</tr>
<tr>
<td>7.4</td>
<td>Message length</td>
<td>15</td>
</tr>
<tr>
<td>7.5</td>
<td>Proxy considerations</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>Acknowledgements</td>
<td>15</td>
</tr>
<tr>
<td>9</td>
<td>Bibliography</td>
<td>16</td>
</tr>
<tr>
<td>A</td>
<td>Author’s Address</td>
<td>16</td>
</tr>
<tr>
<td>B</td>
<td>Full Copyright Statement</td>
<td>16</td>
</tr>
</tbody>
</table>