INTERNET-DRAFT Greg Hudson
Expires: November 21, 2000 ghudson@mit.edu
MIT

Security in the Instant Message and Presence Protocols
draft-hudson-impp-security-00.txt

1. Status of this Memo

2. Abstract

This document describes the security issues and options associated with instant messaging and transmission of presence as they are being considered in the IMPP working group.

This document uses many terms described in [RFC 2778], which describes a model for instant messaging.

3. IMPP architecture and payload characterization

The architecture favored by the IMPP working group involves two different kinds of network elements, clients and servers. Clients represent IMPP entities (senders, instant inboxes, presentities, or watcher), and only speak to the server of their home domain. Servers communicate with each other in order to carry out presence operations.

A server may choose to communicate with entities other than IMPP clients; in this case, the server is called a gateway. For instance, MIT might set up a server which appears to be an IMPP server to other servers, but which communicates with Zephyr clients instead of standard IMPP clients.

A server may choose to communicate with "clients" only on the local host, without using the standard protocol, or with only one "client" built into the server program.

Servers for a domain are located using a SRV [RFC 2782] lookup in the DNS, similar to how email servers for a domain are located using an MX lookup. Also as with email, there is a fallback to an A record lookup if no SRV record exists.

Compared to email, IMPP traffic is expected to be characterized by larger numbers of smaller messages. Currently, a heavy email user subscribed to many email lists might receive a few hundred messages in a day, while a user of many instant messaging "chat rooms" or IRC channels might receive a few hundred messages in an hour or less.

The forms of the IMPP transfer protocols are not finalized. This document will be written assuming that the transfer protocols between clients and servers and between servers and servers use long-lived TCP connections.

4. Goals

The security goals of the IMPP system are presented in [RFC 2779]. They can be summarized as follows:

* Secrecy: Eavesdroppers should not be able to read message content. Ideally, they should not be able to do traffic analysis either, beyond watching where the actual IP packets go.

* Authentication and integrity: It should be difficult to forge IMPP operations or modify them in transit.

5. General methods

Security goals can be achieved in one of two general ways:

1. Data can be authenticated or encrypted at the level of the transfer protocol or, equivalently, at the level of the underlying transport protocol (e.g. using TLS or IPsec). This method requires the communicating parties to trust the intermediary servers to do their job properly.

2. Data can be authenticated and/or encrypted from the sender to the receiver, even though they do not enjoy a direct connection. This method does not require the communicating parties to trust any intermediaries, but does require them to agree on a security mechanism and key management structure.

Of course, these two methods can be layered together for added protection. And it is possible to mix the two methods; for instance, presence information might be authenticated from a presence service to a watcher, which would require the presentity's server to be trusted but not the watcher's.

6. Secrecy

Preventing passive eavesdroppers from reading traffic, including headers, is relatively easy. At the beginning of each transfer connection, the two parties can conduct an anonymous Diffie-Hellman key exchange and encrypt the transfer stream using a symmetric algorithm. This scheme requires no infrastructure and would prevent passive eavesdroppers from reading traffic, but a man-in-the-middle attacker can defeat this scheme while remaining invisible to the two parties. Also, there is currently no standardized protocol method that I know of for doing anonymous Diffie-Hellman key exchange, selecting symmetric encryption algorithms, and deriving keys.

Secrecy of IM and presence payloads can also be achieved in an end-to-end manner if the sender can obtain the a public key for the receiver or can somehow negotiate a shared secret with the receiver. This category of method requires infrastructure, of course, and does not prevent attackers from doing traffic analysis.

7. Authentication

Authentication is a fundamentally more difficult problem than secrecy. Authentication between two unfamiliar parties cannot be achieved without some kind of infrastructure, and authentication infrastructures are inherently limited in strength. In considering the options below, note that each option proves something slightly different about the sender.

7.1. Transfer-level or transport-level authentication

Authentication at the transfer protocol or transport protocol level has some attractive properties:

* Assuming the transfer protocol uses long-lived TCP connections, security associations can be set up at the beginning of a connection and efficient symmetric protocols can be used to encrypt and integrity-protect traffic.

* Proper authentication at the transfer level can allow improved secrecy, preventing even man-in-the-middle attackers from eavesdropping or performing traffic analysis.

* Domains can reuse their existing security infrastructure to authenticate clients.

Between a client and its home server, authentication is essentially a solved problem. Either SASL [RFC 2222] or SPNEGO [RFC 2478] allow clients to use a variety of methods of varying strengths to authenticate to a server and to protect traffic. Server operators can decide what sorts of authentication they will allow from clients.

Between servers for two domains, however, the problem is harder. Although SASL or SPNEGO can be used between domains, most of the existing mechanisms within those frameworks are not as useful between domains, and the problem of configuring a minimum security level between domains is essentially unsolvable without help from the protocol specification.

Following are descriptions of two possible options for transfer-level authentication between domains. Neither is particularly optimal.

7.1.1. Using DNS KEY records for inter-domain authentication

If we wish to authenticate that a server is properly representing the DNS domain it claims to be representing, then it would make some amount of sense to use the DNS as the authority for keys. The secure DNS specification [RFC 2065] envisions that DNS might be used to store public keys for application protocols. Ideally these keys would be signed to protect against DNS spoofing attacks. Public keys in the DNS could be used in a variety of ways for authenticating domains; one option would be authenticated Diffie-Hellman key agreement as described in [RFC 2631].

This scheme has some nice properties:

* Because servers look up each others' key records in the DNS, and because DNS records usually expire after a relatively short time period, there is no certificate revocation problem, and key roll-over is possible by including multiple keys in the RRset.

* Because servers look up each others' key records in the DNS, this mechanism can be made optional without allowing an attacker to spoof a domain by claiming that the domain does not support security. If a domain has a public key for IMPP in the DNS, then it would not be allowed to authenticate to another secure domain without the private key.

* Even in the absence of signed key records, an attacker would have to perform a DNS spoofing attack to forge a request. This would be considerably better than email, where forging email is as trivial as lying about the from address.

* Because the same authority is used for keys as for server lookup, there is no concern that a certification authority separate from the relevant DNS authorities might be issuing certificates to parties other than the authorized domain administrators.

However, there are some important disadvantages to this scheme:

* Secure DNS is essentially vapor at the current time. Some

implementations exist, but we have essentially no operational experience with it.

* Upon receiving a connection and an authentication request, a server would have to go out and perform a DNS query to get the public key for the domain the connection initiator claims to be. This may not be a real problem, since server implementations already have to perform DNS queries to initiate connections to other domains.

* There is no standard mechanism for authenticating domains using public keys stored in the DNS; a new one would have to be invented.

7.1.2. Using TLS and X.509 certificates for inter-domain authentication

TLS [RFC 2246] describes a transport-layer security protocol using **X.509 certificates. This approach is currently in wide use for** securely authenticating World Wide Web sites to web users and for protecting traffic (especially for passwords and/or credit card information used in web commerce). TLS could be used for mutual authentication between domains, since it allows for certificates to be passed in both directions.

The advantages of this scheme are obvious: it is an existing standard with significant deployment experience (although most of it with certificates only going in one direction) with existing publicly available implementations. However, it poses the following problems:

* Certificates typically expire after a long period of time, much longer than a typical DNS expiry time. If a private key is compromised, TLS provides no mechanism for revoking the corresponding certificate.

* Domains would provide each other with the entire certificate chain leading up to the CA as part of the transfer connection; servers do not and cannot look up certificates in any outside directory. This is potentially convenient for the server implementors, but means there is no way to look up whether a domain supports security or not. So if the TLS mechanism is optional, a domain has no way of knowing whether to trust another party's assertion that they represent a domain which does not employ the mechanism.

* Although TLS allows the receiving party to specify to the initiating party which certification authorities it respects, it does not allow the initiating party to specify that information to the receiving party. So a server with certificate chains leading to two different CAs does not know which certificate chain to present upon receiving a connection from another server.

* A certificate chain leads to a certification authority which may not be the same as the authority which controls the root DNS zone. The risk exists that a certification authority might hand out certificates to parties other than authorized domain owners.

7.2. End to end authentication

End to end authentication has some nice properties relative to transfer-level or transport-level authentication:

* The parties involved do not have to trust the servers between them.

* You can use end to end authentication to authenticate facts other than authorized use of a particular IMPP identifier. Two familiar parties can use manually exchanged keys to authenticate their actual identities. Similarly, two previous unfamiliar parties can exchange keys and use them to authenticate the fact that they are the same two parties conversing over a long period of time, even if neither of them is certain of who the other one is.

* Servers don't have to do any work to support end to end authentication between users; the computational work is pushed out to the edges. This is good for traditional Internet deployments involving powerful client machines, although it poses problems for battery-operated wireless clients.

Unlike transfer-level or transport-level authentication, end to end authentication cannot be leveraged to protect against traffic analysis (although it can be leveraged to protect against eavesdropping of IM and presence payloads), and it does not allow administrative domains to reuse their existing key security infrastructure if it does not happen to gel with the end to end security mechanism in use. Finally, end-to-end authentication does not apply (or at least applies differently) to requests between users and servers, such as for a presentity's request to change its presence information.

7.2.1. End to end authentication using existing MIME-based mechanisms

PGP/MIME [RFC 2015] and S/MIME [RFC 2633] are existing methods for authenticating MIME payloads. They are designed for mail, a non-interactive medium characterized by larger, less frequent payloads, and may not perform adequately for large, smaller payloads. The bandwidth overhead imposed by PGP or S/MIME signatures may be unacceptable for one-line messages, and the computation overhead of doing a public key operation for each instant message may be unacceptable for battery-powered devices. However, they are existing standards with available implementations.

7.2.2. End to end authentication using new mechanisms

A new, more interactive MIME-based mechanism could be designed in order to reduce the bandwidth and computation overheads of PGP/MIME and S/MIME. For instance, two parties communicating secretly over many messages could establish a shared secret in the first message, and then use efficient symmetric algorithms to protect the remaining messages.

A mechanism along these lines could help enormously for users participating in IM redistribution lists or "chat rooms," but only if the redistribution list is considered an "end" and is trusted to properly represent the identities of users who send through it. In the absence of such trust, it is difficult to do better than one public key operation per message for a redistribution list scenario.

8. Conclusions

Security in the IMPP area is a difficult problem, and none of the solutions presented above is very satisfactory, at least by itself.

10. References

[RFC 2015]
M. Elkins. "MIME Security with Pretty Good Privacy (PGP)". RFC 2015,
October 1996.

[RFC 2065]
D. Eastlake, 3rd, C. Kaufman. "Domain Name System Security
Extensions." RFC 2065, January 1997.

[RFC 2222]
J. Myers. "Simple Authentication and Security Layer (SASL)." RFC
2222, October 1997.

[RFC 2246]
T. Dierks, C. Allen. "The TLS Protocol Version 1.0." RFC 2246,
January 1999.

[RFC 2478]
E. Baize, D. Pinkas. "The Simple and Protected GSS-API Negotiation
Mechanism." RFC 2478, December 1998.

[RFC 2631]
E. Rescorla. "Diffie-Hellman Key Agreement Method. E. Rescorla." RFC
2631, June 1999.

[RFC 2633]
B. Ramsdell, Ed.. "S/MIME Version 3 Message Specification." RFC
2633, June 1999.

[RFC 2778]
M. Day, J. Rosenberg, H.Sugano. "A model for Presence and Instant
Messaging." RFC 2778, February 2000.

[RFC 2779]
M. Day, S. Aggarwal, G. Mohr, J. Vincent. "Instant Messaging /
Presence Protocol Requirements."

11. Author's address

Greg Hudson
Massachusetts Institute of Technology
Cambridge, MA 02139
ghudson@mit.edu