A Suggested Improvement to SSHv2

Jonathan Herzog
The MITRE Corporation
WN99B000041

December 10, 1999

The SSH (Secure SHell) protocol has become one of many de-facto Internet standards for secure log-on and other network services across an insecure connection. It is commonly used to secure connections such as `telnet` or `rsh`, for secure file transfers (via the `scp` command), as a secure tunnel for other connections such as `http` or `X`, or as a VPN component.

A second version of the protocol ([3, 4, 1, 2]) has been formulated by an IETF working group, and is being currently being considered for RFC status. This version of the protocol was the subject of investigation by the MITRE Corporation, whose analysis identified an area of concern. In this note, we present a brief overview of the SSHv2 protocol, explain the area of concern, and suggest a solution.

SSH is a client-server protocol. In its second incarnation, the SSH protocol is actually composed of three sub-protocols, layered upon each other:

- The Transport layer, which is the focus of this note, contains two exchanges and provides two services:

  - At the beginning of a connection, the two parties negotiate the cryptographic algorithms to be used.
  - The client and server engage in "key exchange," which includes server authentication as well as key exchange. The exact mechanism used is negotiated as part of the exchange above.

  Both the algorithm negotiation and the key exchange/server authentication are required at the beginning of a connection, but can re-occur at any time (as desired by the participants).

  The Transport layer also:

  - Breaks streams of data from higher protocols into packets for transmission, and combines incoming packets back into streams of data, and
  - Provides cryptographic services such as integrity and confidentiality, and non-cryptographic services such as compression. The algorithms used for these services are negotiated as above.

- The Authentication layer relies on the secure tunnel established at the Transport layer, and provides mechanisms for client authentication. Typically, these mechanisms will be based on a name-password pair (traveling within the secure tunnel) but may rely on more secure methods.

- The Connection layer multiplexes many connections through the mutually-authenticated tunnel established by the Authentication layer.

The protocol is designed to be modular, in that the client and server can negotiate acceptable settings and algorithms for each service separately. The algorithms for confidentiality can be negotiated separately from those for integrity, for example, and the algorithms for client-to-server communication can be different than those for server-to-client. For interoperability reasons, the specification contains a set of algorithms which must be supported by all implementations.

Of concern here is the key exchange algorithm *diffie-hellman-group1-sha1*, which is the only key exchange algorithm defined by the standard. This exchange proceeds as follows:

1. The client chooses a random number x, and sends to the server the value $g^x \bmod p$, where p is a large prime number and p, g are given in the specification.

2. The server chooses its own random number y, and sends to the client

   - $g^y \bmod p$,
   - its certificate, and
   - a signed hash of all previous messages (and the secret key $g^{xy} \bmod p$).

The area of concern focuses upon the fact that the specification fixes the values of p and g. The key exchange scheme described above is called Diffie-Hellman (where the certificate and signed hash have been added to prevent man-in-the-middle attacks), and it relies upon the difficulty of the discrete logarithm for its security. Under some conditions, the discrete log is easy to compute, and for this reason the value of p must be chosen carefully. For example, it is easy to compute the discrete logarithm when $p-1$ has only small prime factors, and hence p is usually chosen so that $(p-1)/2$ is itself prime. Likewise, some care must be given to the choice of g so that the subgroup generated by g is relatively large, but this is usually an easier choice to make than the choice of p.

Because a poor choice of p or g has potential pitfalls, it is not inherently a bad decision to re-use the same values in many runs of Diffie-Hellman. Indeed, it may actually be desirable for interoperability reasons.[1] In the case of SSH, however, there is no performance considerations that would benefit from the use of a standard p value. Indeed, prudent design considerations might actually indicate the opposite.

[1] For example, if one were to create certificates containing Diffie-Hellman values, one might insist on the use of global values, so that any two certificates can be used together.

The best known algorithms for generalized discrete logarithms are still quite slow, but the majority of time is actually consumed in easily parallelizable precomputations about the group (choice of p and g) in general. Once these precomputations are finished, then any discrete logarithm in that group is easily found. Given that most SSH connections will be created using the only key exchange algorithm defined in the specification, the time and expense required to break the majority of SSH connections is therefore only slightly greater that the time and expense required to perform the aforementioned precomputations for the group specified in the standard.

One time-honored rule for security design is that the value of the data being protected should be less than the expense required to break the security system. Given that SSH is a popular choice of security protocol for remote login and VPN applications, the value of all data protected by SSH might indeed exceed the expense of the precomputations required above.

For this reason, we believe that it should be recommended to the SSH IETF working group that the current key exchange algorithm, *diffie-hellman-group1-sha1*, be supplemented by another standard key exchange algorithm that allows the p and g values to be negotiated. Although it might be advisable to provide a list of recommended strong values in the specification (or elsewhere), the recommended algorithm should allow the client and server to negotiate a pair of mutually acceptable values independent of said list. Just as the connection terminates if no mutually acceptable encryption scheme can be found, the connection should terminate if no mutually acceptable set of Diffie-Hellman values can be found.

The specific details of this algorithm can be left open, to be decided upon by the members of the SSH IETF working group.

References

[1] T. Ylonen, T. Kivinen, and M. Saarinen. SSH authentication prototcol. Internet draft, November 1997. Also named draft-ietf-secsh-userauth-01.txt.

[2] T. Ylonen, T. Kivinen, and M. Saarinen. SSH connection prototcol. Internet draft, November 1997. Also named draft-ietf-secsh-connect-03.txt.

[3] T. Ylonen, T. Kivinen, and M. Saarinen. SSH prototcol architecture. Internet draft, November 1997. Also named draft-ietf-secsh-architecture-01.txt.

[4] T. Ylonen, T. Kivinen, and M. Saarinen. SSH transport layer prototcol. Internet draft, November 1997. Also named draft-ietf-secsh-transport-01.txt.