Session Key Retrieval in J-PAKE Implementations of OpenSSL and OpenSSH

Sébastien Martini – seb@dbzteam.org

September 12, 2010

1 Description

This issue affects the implementations of J-PAKE [1] in OpenSSL [2] and OpenSSH [3]. These implementations referred as experimental [4, 5] and work-in-progress [5], both contain the same flaw, namely, there aren’t adequately verifying the public parameters received from untrusted parties. These parameters must be reduced modulo \( p \) in order to prevent an attacker to bypass important non-modular checks. This deficiency may enable an attacker not knowing the secret password to confine the computations of her victim into a small subgroup [6] eventually leading her to always derive her session key from the value \( K = 1 \). Like J-PAKE, this issue is symmetric, meaning that in the usual client/server model, the attacker could be a client trying to authenticate to a server, or a server trying to impersonate another server to an honest client.

2 Modified Protocol Rounds

Eve, in order to perform her attack, modifies the rounds 1 and 2 of the original protocol (see [1] section 3) to send to her victim Alice carefully selected values instead of the randomly chosen ephemeral ones.

- **Round 1’**: Eve randomly selects \( x_1 \) exactly like in the original first round, then she picks \( x_2 = 0 \) and \( g^{x_2} = p + 1 \) and calculates a knowledge proof of 0 using the term \( g^{x_2} = p + 1 \) in the hash computation. She then sends out these values to Alice.

- **Round 2’**: Eve selects \( B = 1 \) along with a knowledge proof of zero and sends out these values to Alice.

Alice follows the original protocol but mistakenly does not reduce \( g^{x_2} \mod p \) thus both parties validate every steps of the protocol. Which eventually lead Eve and Alice to invariably share the same value \( K = 1 \).

3 Remarks

3.1 OpenSSH

The value of \( x_1 \) in OpenSSH is required to be different than zero and the value of \( B \) must be strictly greater than one. Hence, round 2’ must be modified to take the value of a congruent of 1 mod \( p \) different than 1.

3.2 SRP

[7] demonstrated the effective need for validating SRP [8] input values. Hence, this issue may also apply to uncareful SRP implementations using non-modular verifications.
References


