Computer Communications 32 (2009) 736–739

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Security flaw of Hölbl et al.'s protocol

Jorge Munilla, Alberto Peinado *

Dpto. Ingeniería de Comunicaciones, E.T.S.I.Telecomunicación, University of Málaga, Malaga, Spain

ARTICLE INFO

Article history:
Received 23 June 2008
Received in revised form 25 November 2008
Accepted 27 November 2008
Available online 7 December 2008

Keywords:
Authentication
Hash
Cryptanalysis
Password

ABSTRACT

Recently, Hölbl et al. [M. Hölbl, T. Welzer, B. Brumen, Improvement of the Peyravian–Jeffries's user authentication protocol and password change protocol, Computer Communications 31 (2008) 1945–1951] have proposed an improvement of Peyravian–Jeffries's user authentication protocol and password change protocol [M. Peyravian, C. Jeffries, Secure remote user access over insecure networks, Computer Communications 29 (5–6) (2006) 660–667]. Peyravian–Jeffries's scheme suffers from an active off-line password-guessing attack [J. Munilla, A. Peinado, Off-line password-guessing attack to Peyravian–Jeffries's remote user authentication protocol, Computer Communications 30 (1) (2006) 52–54], and Hölbl et al. state that their improved protocol overcomes this weakness. However, we show in this paper that although this proposed protocol prevents this active attack, it remains vulnerable to a passive (simpler) off-line password-guessing attack.

1. Introduction

User authentication based on passwords is largely used to control the access to resources or confidential information residing within networks. In a password scheme, the user authentication protocol is a procedure to achieve user authentication and, the password change protocol is a procedure which allows an authenticated user to change his password. Thus, it is assumed that every user has a unique identification, *id*, and a password *pw* to access the system. While *id* is considered to be public information, *pw* is considered to be private or secret information. A database with this information, along with the services which the user is allowed to access, must be usually maintained by the server responsible for administering and limiting access to users.

Password authentication protocols are very subject to:

- *Replay attack: An* attacker impersonates a legitimate user through the reuse of information obtained (eavesdropping) in previous executions of the protocol.
- *Dictionary attack or password-guessing attack: This* attack is possible when weak passwords with low entropy are used (most users select passwords from a small subset of the full password space), and involves an attacker simply systematically trying passwords, one at a time, until the correct one is found [2–4,9].
- *Stolen-verifier attack: An* attacker gains access to the information stored inside the system (server who stores the passwords). This information can subsequently be used by the attacker to impersonate legitimate users.

Password change protocols are very subject to:

- *Denial-of-service attack (DoS): This* attack prevents or inhibits the normal use or management of communications facilities (e.g. cause the server to reject a legitimate user).

In 2006, Peyravian and Jeffries [8] proposed two schemes of protocols to perform remote user authentication and password change over insecure networks in a secure manner. The first scheme does not use any private-key or public-key infrastructure, and only employs a collision-resistant one-way hash function (e.g. SHA-1). A hash function can be defined as a computationally efficient function mapping binary strings of arbitrary length to binary strings of some fixed length [6]. To be of cryptographic use, this hash function is typically chosen such that it is computationally infeasible to find two distinct inputs which hash to a common value (collision resistant), and that given a specific hash value, it is computationally infeasible to find the input (one-way). Furthermore, hash functions are generally public, and everyone can apply it to any input. The second scheme proposed by Peyravian and Jeffries combines a collision-resistant one-way has function with the Diffie–Hellman key agreement scheme (DH scheme) [1]. They claimed that the first scheme was secure against DoS attacks, whereas the DH scheme provided protection against off-line password-guessing attack and DoS attacks. However, Munilla and Peinado [7], and Shim [10], independently showed that this DH-based scheme was still vulnerable to an off-line password-guessing attack.

Hölbl et al. [5] have recently proposed an improved variant of Peyravian–Jeffries's DH-based scheme in order to overcome the above mentioned weakness. The authors claim that their protocol

* Corresponding author. Tel.: +34 95 213 1305; fax: +34 95 213 2027.
E-mail address: apeinado@ic.uma.es (A. Peinado).

0140-3664/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2008.11.034

provides protection against off-line password-guessing attacks and, as Peyravain–Jeffries' protocol, only employs the Diffie–Hellman key agreement scheme and collision-resistant one-way hash functions. However, in this paper we present another off-line password-guessing attack against this improved protocol. This attack is even easier to carry out than the attacks presented in [7,10] on Peyravian–Jeffries's DH-based scheme, since these attacks are active, the attacker has to make a connection with the client, whereas the attack on Hölbl et al.'s protocol is totally passive. Due to the password change are extensions of the user authentication protocols, they will not be described here, and we will focus on the user authentication protocols.

The remainder of this chapter is structured as follows. Section 2 reviews the user authentication protocol proposed by Peyravian and Jeffries and its cryptanalysis. Section 3 describes the Hölbl et al.'s user authentication protocol. The passive password-guessing attack against this improved protocol is presented in Section 4. And finally, some conclusions are given in Section 5.

2. Peyravian–Jeffries's user authentication protocol

In this section, we review the DH-based Peyravian–Jeffries's user authentication protocol (DH–PJ protocol). It is assumed that the user and the server have agreed on the password (pw) to be used, but the server, to prevent stolen-verifier attacks, does not store pw. Instead it stores a password digest value. More specifically, the server generates a table with the values id and $idpw\_dig = hash(id, pw)$.

For a better understanding, we summarize next some notations that are used throughout this paper:

id: user identity (unique and public),
pw: user password (private and possibly weak),
rc,rs: session-independent random numbers chosen by the client and the server, respectively,
p: a large prime number,
g: a primitive element for $GF(p)$, where $GF(p)$ is the set of integers $\{0, 1, \ldots, p-1\}$ with arithmetic operations defined modulo p,
x,y: session-independent random non-zero integer exponents $x, y < (p-1)$, chosen by the user and the server, respectively,
hash: a collision-resistant one-way hash function,
idpw_dig: $hash(id, pw)$,
$\oplus$: bitwise XOR operation.

2.1. Description of DH–PJ

The DH–PJ protocol comprises the following steps (see Fig. 1):

S.1. The user submits his id and pw to the client.
S.2. The client generates a random value rc, and chooses a large prime p, and a primitive element g for $GF(p)$. The client also chooses a random non-zero integer $x < (p-1)$, and computes $g^x \bmod p$. Next, the client sends $id, rc, p, g, g^x \bmod p$ to the server.
S.3. The server generates a random value rs. The server also chooses a random non-zero integer $y < (p-1)$, and computes $g^y \bmod p$. Then, the server obtains the value $(g^x)^y \bmod p = g^{xy} \bmod p$. Using the random values rc and rs, the computed $g^{xy} \bmod p$, and $idpw\_dig$, the server generates

$$challenge = rs \oplus hash(g^{xy} \bmod p, idpw\_dig, rc) \quad (1)$$

Next, the server sends $g^y \bmod p$ and $challenge$ to the client.

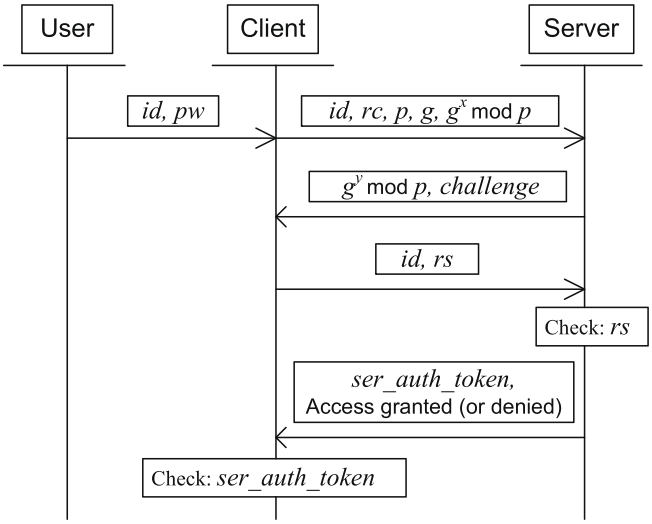

Fig. 1. Peyravian–Jeffries's DH-based user authentication protocol.

S.4. The client computes $idpw\_dig = hash(id, pw)$ using id a pw received from the user, and $g^{xy} \bmod p$, by raising the received $g^y \bmod p$ to x. Next, the client retrieves rs by computing

$$rs = challenge \oplus hash(g^{xy} \bmod p, idpw\_dig, rc) \quad (2)$$

Next, the client sends id and rs to the server.
S.5. The server verifies that the received rs is the same as the one it generated. If they are different, it sends a message refusing the user's request to access the system. Otherwise, the user is authenticated and the server generates a one-time authentication token as follows:

$$ser\_auth\_token = hash(g^{xy} \bmod p, idpw\_dig, rc, rs) \quad (3)$$

and sends it to the client.
S.6. The client verifies the validity of the token to authenticate the server.

2.2. Cryptanalysis of DH–PJ

DH–PJ protocol is not resistant to impersonation attacks (active). In this kind of attacks, the attacker impersonates one or another part depending on the side that is authenticated at first place. If the client's authentication token is sent first, the attacker will impersonate the server to the client. Otherwise, when the server is authenticated at first place, the attacker will impersonate the client to the server.

In this case, the first authentication token is sent by the client to the server ($S.4$), and therefore, the attacker takes the server role. The following steps are performed to breach the system security (see Fig. 2):

S.1. A legal user submits his id and pw to the client.
S.2. The client generates rc, and chooses p, a primitive element g for $GF(p)$, and a random non-zero integer $x < (p-1)$. Then, the client computes $g^x \bmod p$, and sends $id, rc, p, g, g^x \bmod p$ to the (fake) server.
S.3. The attacker, or fake server, chooses any known number as $challenge*$ and any known non-zero integer as $y*$. Then, he computes $g^{y*} \bmod p$, and sends it and $challenge*$ to the client.
S.4. The client computes $idpw\_dig = hash(id, pw)$ using id a pw received from the user, and g^{xy*}, by raising the received $g^{y*} \bmod p$ to x. Then, the client retrieves rs as follows: