Sistemas de Gestão de Identidade

Trabalho I

O presente trabalho consiste em implementar uma versão simplificada do **CHAP - Challenge Handshake Authentication Protocol**, protocolo de autenticação baseado num segredo partilhado (*password*), que permite a um programa cliente autenticar um utilizador junto de um programa servidor, sem nunca enviar o segredo pela rede.
O protocolo deverá constar pelo menos dos seguintes passos:

1. A aplicação servidor conhece um conjunto de credenciais de utilizadores (*usernames* e *passwords*) que deverão ser armazenados num ficheiro, sendo as passwords obrigatoriamente guardadas sob forma de *digest* (p.ex. MD5).
2. A aplicação cliente começa por pedir ao utilizador o seu *username*.
3. O cliente abre uma conexão para o servidor (utilizando p.ex. a socket API), envia o nome do utilizador em claro e fica à espera de receber uma resposta.
4. O servidor verifica que conhece o nome do utilizador e o *digest* da respectiva password armazenado.
5. Se for o caso, envia de volta uma mensagem de *challenge* ao cliente contendo um *nonce* N de 64 bits (8 bytes), obtido p.ex. pela concatenação do valor do tempo sistema (32 bits - 4 bytes) com um número inteiro aleatório (32 bits - 4 bytes), e fica à espera da resposta do cliente.
6. O cliente pede ao utilizador a sua password P e calcula o seu *digest* (p.ex.MD5), obtendo assim Dp = MD5(P), com 128 bits (16 bytes).
7. O Cliente efectua a concatenação do *nonce* N com o *digest* Dp da password, obtendo N || Dp e calcula a resposta R = MD5 (N || Dp) que envia de volta ao servidor (16 bytes).
8. O servidor realiza a mesma concatenação que o cliente a partir do valor do *digest* Dp da *password* do utilizador que tem armazenada e do *nonce* N que enviou, obtendo R' = MD5 (N || Dp).
9. Se R == R', então o cliente está de posse da password certa e está a responder directamente ao servidor, pois o nonce é válido, provando portanto a sua identidade: o servidor envia uma mensagem de sucesso ao cliente
10. Se R != R´ não há prova de autenticação e o servidor envia uma mensagem de erro ao cliente.

Nota: Alternadamente, pode ser utilizada a função de *hash* SHA1, com a diferença que os *digests* terão 160 bits (20 bytes) em vez de 128 bits (16 bytes) no caso do MD5.

Referências:

- Protocolo CHAP (Challenge Handshake Authentication Protocol) www.ietf.org/rfc/rfc1994.txt e netlab.ulusofona.pt/im/teoricas/IM-02-IdMgmt.pdf
- Funções que permitem obter o tempo sistema e gerar números aleatórios em Linux: linux.die.net/man/2/time e linux.die.net/man/3/rand_r
- Exemplo de geração de um nonce: netlab.ulusofona.pt/cr/praticas/nonce.c
- Exemplo de cálculo do digest MD5: netlab.ulusofona.pt/cr/praticas/md5.c