ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA

HỆ ĐIỀU HÀNH (MỞ RỘNG) (CO201D)

Optimal strong-password authentication protocols

GVHD: Nguyễn Lê Duy Lai
SV thực hiện: Trịnh Cao Thắng – 2014550

Tp. Hồ Chí Minh, Tháng 12/2022

Contents

1 Cơ sở lý thuyết

1.1 Xác thực (Authenciation)

Authencication (Xác thực) : là quá trình kiểm tra danh tính của một tài khoản trên hệ thống hiện tại thông qua một hệ thống xác thực. Nếu như không có quá trình xác thực thì hệ thống không xác định được danh tính của tài khoản đăng nhập vào là hợp pháp hay không để đưa ra những phản hồi thích hợp. Quá trình xác minh được sử dụng nhiều ở các hệ quản trị nội dung (Content Management System) liên quan đến quản lý, tương tác người dùng thông qua form đăng ký được xác nhận dựa trên tên tài khoản người dùng và mật khẩu.

Quá trình xác minh có thể được thực hiện bằng nhiều phương thức.

- Mật khẩu : Khi người dùng truy cập, hệ thống sẽ không lưu lại mật khẩu vào cơ sở dữ liệu mà sẽ lưu giá trị của mật khẩu khi đưa qua hàm băm (md5, SHA1, ...). Do tính chất hàm băm là hàm 1 chiều và không thể bị đảo ngược, nếu không may cơ sở dữ liệu bị hacker xâm nhập, họ cũng không thể biết được mật khẩu của người dùng. Quá trình xác minh bằng mật khẩu này có nhiều biến thể như Swipe Pattern PIN (dùng phổ biến trong các điện thoại Android) và các loại mật khẩu dùng 1 lần (dùng trong các giao dịch quan trọng như giao dịch ngân hàng)
- Hệ thống mã hóa sử dụng khóa công cộng : còn gọi là Public-key Cryptography, bao gồm một cặp khóa công cộng và khóa bí mật (pk, sk). Để dăng nhập vào hệ thống, người dùng cần sử dụng khóa cá nhân và hệ thống sẽ xác nhận danh tính người dùng bằng cách sử dụng khóa công cộng. Hacker cho dù có kháo công cộng cũng không thể biết được bí mật nếu không có khóa cá nhân.
- Sinh trắc học : Quá trình xác nhận sử dụng sinh trắc học là quá trình sử dụng dấu vân tay, tròng mắt, hoặc khuôn mặt để xác thực thông tin người dùng. Một ưu điểm là định danh và mật khẩu luôn đi cùng nhau nên người dùng dùng không cần phải lo lắng chuyện quên mật khẩu khi đăng nhập.

Mọi phương thức xác minh người dùng luôn có những rủi ro và thách thức như mất mật khẩu, mất khóa cá nhân, bị đánh cắp dấu vân tay hay những thủ thuật tấn công của các hacker nếu quá trình xác thực không được hiện thực đúng đắn.

1.2 Giao thức xác minh tối ưu sử dụng mật khẩu mạnh (Optimal strong-password authencication protocols)

Đối với các giao thức xác thực người dùng sử dụng mật khẩu thông thường, tất cả thông tin người dùng được lưu trong cơ sở dữ liệu của server. Nếu Server không có khả năng bảo mật cao, các hacker có thể thâm nhập vào server và lấy được tất cả thông tin cần thiết. Một cách để phòng ngừa việc này là đưa các thông tin quan trọng trong cơ sở dữ liệu qua hàm băm một chiều, nhằm vẫn bảo mật được thông tin bởi các hacker. Tuy nhiên, không phải tất cả thông tin đi qua hàm băm đều được bảo vệ thành công bằng cách này. Mọi hàm băm đều có những điểm yếu nhất định, quan trọng nhất là không phải mật khẩu nào cũng được đặt một cách có tính bảo mật (123456, thang123,...). Đối với những mật khẩu được đặt một cách thờ ơ như vậy, hacker có thể bruteforce để tìm ra được mật khẩu gốc.

Quá trình xác minh sử dụng mật khẩu mạnh tối ưu có một số đặc điểm vượt trội hơn các giao thcuws thông thường : sử dụng nhiều hàm băm lên một dữ liệu, sử dụng mật khâu có nhiều khả năng bảo mật, ... Hơn nữa, trong giao thức này, server chỉ chọn thuật toán băm tại thời điểm người dùng nhập xong mật khẩu và gửi request đến server để xác minh, vì vậy các hacker có thâm nhập vào log tương tác của server và user cũng không thể biết được hàm băm mà server sử dụng là gì, khiến việc đánh cắp thông tin trở nên khó khăn hơn.

1.3 Hàm băm (Hash)

Hàm băm (Hash functions) là hàm dùng để sinh ra giá trị băm (hash value) từ những khối dữ liệu. Hàm băm là hàm không thể bị đảo ngược, nên từ giá trị băm, không có cách nào để có thể khôi phục về

dữ liệu gốc, do đó nó được dùng nhiều trong việc xác minh.

Kí hiệu hàm băm:

$$f : \{0,1\}^{0,1\ldots,\infty} \longmapsto \{0,1\}^k$$

trong đó :

- f là tên hàm băm (các hàm sẵn có hoặc tự chế)
- $\{0,1\}^n$ là dữ liệu biểu diễn dưới dạng nhị phân với số bit là n

Vì dữ liệu có thể rất lớn, và giá trị băm thì chỉ có một số bit nhất định, nên sẽ có trường hợp hai giá trị đầu vào cho cùng một giá trị băm, trường hợp này ta gọi là đụng độ (collision). Nếu hàm băm nào đó xảy ra đụng độ một cách dễ dàng, thì cũng có nghĩa là ta có thể tìm được hai dữ liệu mà có chung một giá trị băm, hoặc cho một mật khẩu mà mình không biết, nhưng mình có thể tìm được một mật khẩu khác mà có chung một giá trị băm với mật khẩu đó.

1.4

2 Cơ sở lý thuyết toán học

2.1 Nhóm (Group) - Vòng (Ring) - Miền (Field) - Miền hữu hạn (Finite Field)

Một group hay abelian group là một tập hợp G cùng với một toán tử nhị phân (binary opearator) $*$ thuộc G sao cho :

- $\exists \in G$ (e còn được gọi là phần tử đồng nhất) sao cho : $\forall e \in G,\ a * e = e * a = a$ (Đây là tính chất đồng nhất)
- $\forall a, b, c \in G,\ a * (b * c) = (a * b) * c$ (Đây là tính chất kết hợp)
- $\forall a, b \in G,\ a * b = b * a$ (Đây là tính chất giao hoán)
- $\forall a\ \exists a^{'} \in G,\ a * a^{'} = a^{'} * a = e$ (Đây là tính chất nghịch đảo)

Ví dụ : $\mathbb{Z}/n\mathbb{Z}$, là tạp hợp các số nguyên modulo n và chỉ bao gồm phép toàn $+$, là một group. Tập số nguyên vô hạn $\mathbb{Z}$ cũng là một group.

Một vòng (ring) là một tập hợp R mà được bao đóng bởi hai phép toán là $+$ và $\times$ và thỏa mãn những tính chất sau :

- R là một abelian group dưới toán tử $+$
- Tính chất kết hợp :

$$\forall a, b, c \in R : a \times (b \times c) = (a \times b) \times c$$

- Tính chất phân phối :

$$\forall a, b, c \in R : a \times (b + c) = a \times b + a \times c\ and\ (b + c) \times a = b \times a + c \times a$$

Ví dụ : Các tập hợp $\mathbb{Z}/n\mathbb{Z}$, $\mathbb{Z}$ cũng là các vòng. Tập tất cả đa thực có hệ số nguyên $\mathbb{Z}[x]$ là một vòng.

Một field là tập hợp F được bao đóng bởi 2 toán tử $+$ và $\times$ và thỏa mãn các tính chất :

- F là một abelian group dưới toán tử $+$
- $F - \{0\}$ là một abelian group dưới toán tử $\times$

Nếu một miền (field) có số lượng phần tử là hữu hạn thì được gọi là miền hữu hạn (finite field) hay Galois field

Ví dụ : Nếu p là số nguyên tố thì tập $\mathbb{Z}/p\mathbb{Z}$ là một field, và thường được kí hiệu là $\mathbb{F}_p$

Cho $x \in \mathbb{F}_p$, ta nói *order* của x (kí hiệu là $ord(x)$) trong $\mathbb{F}_p$ là một số nguyên k nhỏ nhất với $k > 1$ sao cho : $x^k \equiv x \pmod{p}$
. Nếu $k = p$. ta nói x là số khởi tạo miên $\mathbb{F}_p$ (finite field generator)

2.2 Phép toán XOR

Phép XOR (kí hiệu là $\oplus$) là toán tử 2 dầu vào $a = \{0, 1\}$ và $b = \{0, 1\}$, và có kết quả là 0 nếu $a \neq b$ và 1 nếu $a = b$. Bảng chân trị của phép toán XOR được biểu diễn như sau :

a	b	a $\oplus$ b
0	0	0
0	1	1
1	0	1
1	1	0

Toán tử XOR có các tính chất sau :

- Kết hợp : $a \oplus (b \oplus c) = (a \oplus b) \oplus c$
- Giao hoàn : $a \oplus b = b \oplus a$
- Đồng nhất : $a \oplus 0 = a$
- Nghịch đảo : $a \oplus a = 0$

3 Hệ thống đã có

3.1 Giới thiệu sơ bộ

Nhằm mục đích xác minh, trong quá khứ đã hiện thực được một giao thức xác minh được cải tiến và sử dụng bởi các người dùng remote bằng cách sử dụng thanh USB. Giao thức này duy trì những thông tin kết nối nhằm dùng để xác minh giữa server và user. Những user ở bất cứ vị trí địa lý nào đều có thể truy cập vào những thông tin được cung cấp bởi admin mà được lưu ở server nhờ có thanh USB. Những thanh USB này có một ID tiêu chuẩn, được đăng ký trước với server. Server sẽ cho phép người dùng đăng nhập khi và chỉ khi ID của thanh USB là hợp lệ. Những thanh USB này còn chứa hai khóa bí mật (secret key) và khóa công cộng (public key), nhằm tăng hiệu suất cho quá trình xác minh của giao thức.

Trong quá trình dăng nhập, remote users sử dụng những thông tin được lưu trữ trong thanh USB, tuy nhiên server chỉ sử dụng những thông tin mà đucợ gửi từ users tại thời điểm users đăng nhập và server không sử dụng những thông tin được lưu trữ trong thanh USB. Giao thức này bao gồm 5 giai đoạn khác nhau được biểu diễn tường minh bằng 4 giai đoạn chính :

- Giai đoạn khởi tạo (Initialization Phase)
- Giai đoạn đăng ký (Registration Phase)
- Giai đoạn đăng nhập và xác minh (Login and Authentication Phase)
- Giai đoạn liên quan đến mật khẩu

3.2 Kí hiệu quy ước

Những kí hiệu sử dụng trong các quá trình của bài cáo báo này được quy ước như sau :

- id_i : khóa chính (định danh của người dùng i)
- pw_i : mật khẩu của người dùng i
- S : kí hiệu cho server
- sk, pk : khóa bí mật và khóa công cộng của server S
- p, q : là 2 số nguyên tố trong đó $p = 2q + 1$
- g : là số khởi tạo finite field $\mathbb{F}_p$ với $ord(g) = p$
- H : là tên hàm 1 chiều
- T : mốc thời gian (timestamp)
- ΔT : thời gian delay đường truyền cực đại
- $\mathbb{Z}_q$: là vòng chứa các số nguyên modulo q
- $||$: phép nối chuỗi
- $\mathbb{Z}_p^*$: là tập $\mathbb{Z}_q$ nhưng chỉ bao gồm toán tử $\times$
- n : là số session xác minh (authentication sessions)

3.3 Giai đoạn khởi tạo (Initialization Phase)

Là giai đoạn server khởi tạo các giá trị cần thiết cho việc xử lý yêu cầu đăng ký của người dùng ở giai đoạn kế tiếp. Server thực hiện tổng cộng 4 bước sau :

- Khởi tạo q là một số nguyên tố an toàn (safe prime), do đó $p = 2q + 1$ cũng là số nguyên tố.
- Chọn một giá trị khởi tạo $g \in \mathbb{Z}_q^*$
- Chọn một khóa bí mật ngẫu nhiên $sk \in \mathbb{Z}_q^*$
- Chọn một thuật toán hash 1 chiều H, tính khóa công cộng $pk \equiv g^{sk} \pmod p$

3.4 Giai đoạn đăng ký (Registration Phase)

Trong giai đoạn này, server nhận từ users mỗi cặp giá trị (id_i, pw_i) và thực hiện đăng ký cho user. Những hành động này được mô tả qua các bước như sau :

- User U gửi một cặp giá trị (id_i, pw_i) cho server S
- Server tính toán y_i :
$$y_i = H(id_i||sk) \oplus H(pw_i)$$
- Server lưu id_i vào một bảng id_table và trả về bộ 5 giá trị (pk, y_i, H, p, q) về cho user. User lưu bộ 5 giá trị này vào một file XML.

3.5 Giai đoạn đăng nhập và xác minh (Login and Authentication Phase)

Sau khi đăng ký thành công, user bắt đầu đăng nhập vào server bằng việc nhập vào id_i và pw_i và sử dụng file XML đã lưu trước đó. Giai đoạn đăng nhập và xác minh diễn ra như sau :

- User lấy lại file XML đã lưu trữ trước đó và thực hiện đăng nhập bằng cách gửi id_i và pw_i lên server
- User U chọn một số ngẫu nhiên $a \in \mathbb{Z}_p^*$
- User tính $y_i^{'}, c_i$ và d_i

$$y_i^{'} = y_i \oplus H(pw_i)$$
$$c_i = g^a \pmod p$$
$$d_i = pk^a \pmod p$$

- Tính $v_i = H(id_i \,||\, y_i^{'} \,||\, c_i \,||\, d_i \,||\, T_1)$ với T_1 là mốc thời gian hiện tại của user
- User U gửi bộ 4 giá trị (id_i, c_i, v_i, T_1) đến server S
- Server S nhận request bộ 4 giá trị của user và bắt đầu kiểm tra 2 điều kiện sau
  - Kiểm tra xem id_i có nằm trong id_table hay không
  - Kiểm tra bất đẳng thức : $T_2 - T_1 < \Delta T$ với T_2 là mốc thời gian hiện tại của server.
- Nếu 2 điều kiện trên **cùng thỏa** thì server sẽ thực hiện các bước tiếp theo như sau :
  - Server sẽ tính $Y_i^{'}, d_i^{'}$:

$$Y_i^{'} = H(id_i||sk)$$
$$d_i^{'} = c_i^{sk} \equiv g^{sk.a} \equiv pk^a \equiv d_i \pmod p$$

  - Server tính $V_i = H(id_i \,||\, Y_i^{'} \,||\, c_i \,||\, d_i \,||\, T_1)$
  - Nếu $V_i = v_i$, server tính $m_i = H(id_i \,||\, d_i^{'} \,||\, T_3)$ với T_3 là mốc thời gian hiện tại của server và gửi bộ 2 giá trị (m_i, T_3) đến user U
- User nhận bộ 2 giá trị và sau đó thực hiện kiểm tra :
  - T_3 có hợp lệ hay không (thường là hợp lệ)
  - $M_i = H(id_i \,||\, d_i \,||\, T_3)$ có bằng m_i hay không.
- Nếu cả hai điều kiện cùng thỏa thì server S đã xác minh thành công user U, ngược lại thì server xác minh không thành công
- Sau khi xác minh lẫn nhau thành công giữa server và user, mỗi bên sẽ tính toán một $session\ keys_k$
  - U : $s_k = H(d_i)$
  - S : $s_k = H(d_i^{'})$

3.6 Hoạt động thay đổi mật khẩu (Password change activity)

User U có thể thay đổi mật khẩu tùy ý bằng cách chọn một mật khẩu mới pw_i^* và tính $y_i^* = y_i \oplus H(pw_i) \oplus H(pw_i^*)$ và thay $y_i = y_i^*$

3.7 Hiện thực

Em hiện thực 3 phase chính bằng *SageMath* , là ngôn ngữ phát triển bởi Python nhưng đẩy mạnh vào các hàm built-in cho toán học mật mã. Và để thuận tiện cho việc tính toán (nhất là thời gian tính finite field generator g), em chọn p, q có số bit là 128. Thêm vào đó, em chọn hàm băm là *md5*, nhưng chọn các hàm khác vẫn cho cùng một kết quả xác thực thành công

```
from Crypto.Util.number import *
import hashlib
from pwn import xor
import random

# User information
id = "HappyHacker22"
pw = "123456"

# Initialization phase
delta_T = 1
while(True):
    q = getPrime(128)
    if (isPrime(2*q + 1)):
        p = 2*q + 1
        break
Fp = GF(q, modulus = "primitive")
g = Fp.gen()

# Choose md5 as a secure one way hash function
sk = Fp.random_element()
pk = g**sk
id_table = []

print("Initialisation phase : ")
print("p =", p)
print("q =", q)
print("Finite field generator g =", g)
print("The hash function is fixed to be : md5")
print("Secret key (kept secret) sk =", sk)
print("Public key pk =", pk)
print("Maximum transmission delay = ", delta_T)

print("\nUser information :")
print("Username :", id)
print("Password :", pw)

# Register phase
h1 = b""
h2 = b""
h1 = hashlib.md5((id.encode() + long_to_bytes(sk))).digest()
h2 = hashlib.md5(pw.encode()).digest()
y = xor(h1, h2)
id_table.append(bytes_to_long(id.encode()))

print("\nRegister phase :")
print("Public key pk =", pk)
print("y_i =", bytes.hex(y))
print("H : md5")
print("p =", p)
print("q =", q)

# Login and authentication phase

# User side
a = Fp.random_element()
y_dot = xor(y, hashlib.md5(pw.encode()).digest())
c = g**a
d = pk**a
T_1 = 10

v = hashlib.md5(id.encode() + y_dot + long_to_bytes(c) + long_to_bytes(d) + long_to_bytes
    (T_1)).digest()

```

```
# Server side
T_2 = 13
assert (T_2 > T_1) & (T_2 - T_1 > delta_T)
Y_dot = hashlib.md5(id.encode() + long_to_bytes(sk)).digest()
D = c**sk
assert D == d

V = hashlib.md5(id.encode() + Y_dot + long_to_bytes(c) + long_to_bytes(d) + long_to_bytes
    (T_1)).digest()
assert v == V

T_3 = 15
m = hashlib.md5(id.encode() + long_to_bytes(D) + long_to_bytes(T_3)).digest()

# User side
assert (T_3 > T_2) & (T_3 - T_2 > delta_T)
M = hashlib.md5(id.encode() + long_to_bytes(d) + long_to_bytes(T_3)).digest()

print("\nLogin and authentication phase : ")
if (m == M):
    print("The server S has successfully authenticated to the user U")
else:
    print("Authentication failed")

# Session keys calculation
session_key_U = hashlib.md5(long_to_bytes(d)).hexdigest()
session_key_S = hashlib.md5(long_to_bytes(D)).hexdigest()

print("User session key :", session_key_U)
print("Server session key :", session_key_S)
```

Code 1: Hiện thực giao thức xác minh bằng MD5

4 Hệ thống đề xuất

Hệ thống xác minh đề xuất này dựa trên độ bảo mật cao của việc áp dụng hash nhiều lần 1 giá trị, cũng như độ phức tạp về toán học cũng được giảm thiểu một chút. Để chứng minh việc chọn hàm băm nào cũng cho cùng một kết quả thành công, em sẽ dùng hàm băm SHA256

4.1 Giai đoạn đăng ký (Registration Phase)

User U sử dụng id_i và pw_i để đăng ký lên server, như vậy, user thực hiện các bước sau :

- Tính $y = H^2(pw_i \oplus 1) = H(H(pw_i \oplus 1))$
- Gửi bộ 3 giá trị (id_i, y, n) lên server, với n bắt đầu là 1, thể hiện số lần đăng nhập vào hệ thống của người dùng
- Server tạo một khóa bí mật $sk \in \mathbb{Z}_p$ với p là số nguyên tố. Sau đó server sẽ lưu trữ bộ 3 giá trị $(H(id_i \,||\, sk), y, n)$ vào cơ sở dữ liệu

4.2 Giai đoạn đăng nhập (Login phase)

Sau khi đăng ký thành công, user sẽ đăng nhập vào server sử dụng 2 giá trị là id_i, pw_i nhập từ phía user. Sau đó user sẽ tính 3 gía trị c_1, c_2, c_3 :

$$c_1 = H(pw_i \oplus n) \oplus H^2(pw_i \oplus n)$$

$$c_2 = H^2(pw_i \oplus (n+1)) \oplus H(pw_i \oplus n)$$

$$c_3 = H^3(pw_i \oplus (n+1))$$

Cuối cùng, user gửi bộ 5 giá trị $(id_i, pw_i c_1, c_2, c_3)$ lên server

4.3 Giai đoạn xác minh (Authentication phase)

Giai đoạn xác minh diễn ra như sau :

- Server nhận bộ 5 giá trị $(id_i, pw_i c_1, c_2, c_3)$ từ user
- Server sẽ thay id_i bằng $H(id_i \mid\mid sk)$ tìm trong database xem có giá trị nào thỏa không. Nếu không thỏa thì báo về user là "Xác thực thất bại vì ID không phù hợp"
- Server sẽ kiểm tra 3 điều kiện sau. Nếu không thỏa 1 trong 3 điều kiện này thì sẽ báo về user : "Xác thực thất bại vì mật khẩu không đúng"

$$H(pw_i \oplus n) = c_1 \oplus H^2(pw_i \oplus n)$$

$$H^2(pw_i \oplus (n+1)) = c_2 \oplus H(pw_i \oplus n)$$

$$c_3 = H^3(pw_i \oplus (n+1))$$

- Nếu cả 3 điều kiện đều thỏa, thì báo về người dùng : "Xác thực thành công"

4.4 Hiện thực

Trong phần hiện thực, em mặc định rằng các ràng buộc về mốc thời gian của server và user là hợp lệ, cũng như chỉ có một tài khoản được tạo và thuật toán cho hash là SHA256

```python
from Crypto.Util.number import *
import hashlib
from pwn import xor
import random

id = "HappyHacker22"
pw = "HappyPhoenix22"
p = getPrime(512)
sk = long_to_bytes(random.randrange(0, p))
database = []
n = 1
def hash(s):
    return hashlib.sha256(s).digest()
def findID(db, id, sk):
    for i in db:
        if (id.encode() + sk in i):
            return True
    return False

y = hash(hash(xor(pw, b"\x01")))
store = [id.encode() + sk, y, n]
database.append(store)

temp1 = hash(xor(pw, long_to_bytes(n)))
temp2 = hash(hash(xor(pw, long_to_bytes(n))))
c1 = xor(temp1, temp2)

temp1 = hash(hash(xor(pw, long_to_bytes(n + 1))))
temp2 = hash(xor(pw, long_to_bytes(n)))
c2 = xor(temp1, temp2)

c3 = hash(hash(hash(xor(pw, long_to_bytes(n + 1)))))

received_id = "HappyHacker22"
received_pw = "HappyPhoenix22"

if (findID(database, received_id, sk)):
    print("Found matched ID!")
    pw = received_pw

    lhs1 = hash(xor(pw, long_to_bytes(n)))
    rhs1 = xor(c1, hash(hash(xor(pw, long_to_bytes(n)))))

```

```
    lhs2 = hash(hash(xor(pw, long_to_bytes(n + 1))))
    rhs2 = xor(c2, hash(xor(pw, long_to_bytes(n))))

    lhs3 = c3
    rhs3 = hash(hash(hash(xor(pw, long_to_bytes(n + 1)))))

    if ((lhs1 == rhs1) & (lhs2 == rhs2) & (lhs3 == rhs3)):
        print("Authentication completed!")
    else:
        print("Authentication failed!")
else:
    print("Invalid ID!")
```

Code 2: Hiện thực giao thức xác minh đề xuất bằng MD5

5 Demo Code

Em có demo code trên 3 video :

- Video hiện thực hệ thống sẵn có
  1. https://drive.google.com/file/d/1ngrIMKF6qKjXQ6Ek_gvUHcPjUBfe8Akv/view?usp=share_link
  2. https://drive.google.com/file/d/1HNY8lLX1X6YGvWTshMDn0tiuJSVoPvTM/view?usp=share_link
- Video hiện thực hệ thống đề xuất : https://drive.google.com/file/d/14K5o48kfrHu-3ClnWwWLdZmdrt2HU0j1/view?usp=share_link

Toàn bộ bài làm (báo cáo, source code), em để ở link Github : https://github.com/HappyFalcon22/KSTN-OS-Project

6 Kết luận

Em đã hiện thực được một giao thức xác minh người dùng tối ưu khác và về lý thuyết là bảo mật mạnh hơn hệ thống xác thực đã có. Ở trong hệ thống sẵn có, các giá trị chỉ đi qua hàm hash một lần, vì vậy trong quá trình xác minh, tồn tại các trường hợp nhiều giá trị giống nhau được truyền qua mỗi lần giao tiếp, tạo cơ hội cho các hacker can thiệp vào dữ liệu. Ở hệ thống đề xuất, dù không đầy đủ các chức năng bằng nhưng hệ thống sử dụng hash nhiều lần cùng 1 giá trị (multiple hash function), do đó các giá trị mà hacker nhìn thấy có tỉ lệ là cùng một giá trị hash ra là thấp hơn, do đó cơ hội cho hacker cũng ít hơn.

7 Tài liệu tham khảo

1. *An Optimal Strong Password Authentication Protocol with USB Sticks*. D.Vikram , truy cập từ https://eprint.iacr.org/2014/327.pdf
2. *Optimal Strong Password Protocols*. Sanya Jain, Avani Kasat, truy cập từ https://www.ijariit.com/manuscripts/v7i3/V7I3-1466.pdf
3. Lý thuyết XOR, truy cập từ https://cryptohack.org/challenges/general/
4. *A computational introduction to number theory and algebra*. Victor Shoup.