Отчёт по лабораторной работе “Схема Лэмпорта (S/KEY)”.

Выполнили: Кандинский Алексей и Суханов Арсений.

Реализация протокола.

Схема Лэмпорта (S/KEY) представлена ниже. Наша реализация обладает некоторыми особенностями, о которых пойдет речь ниже.

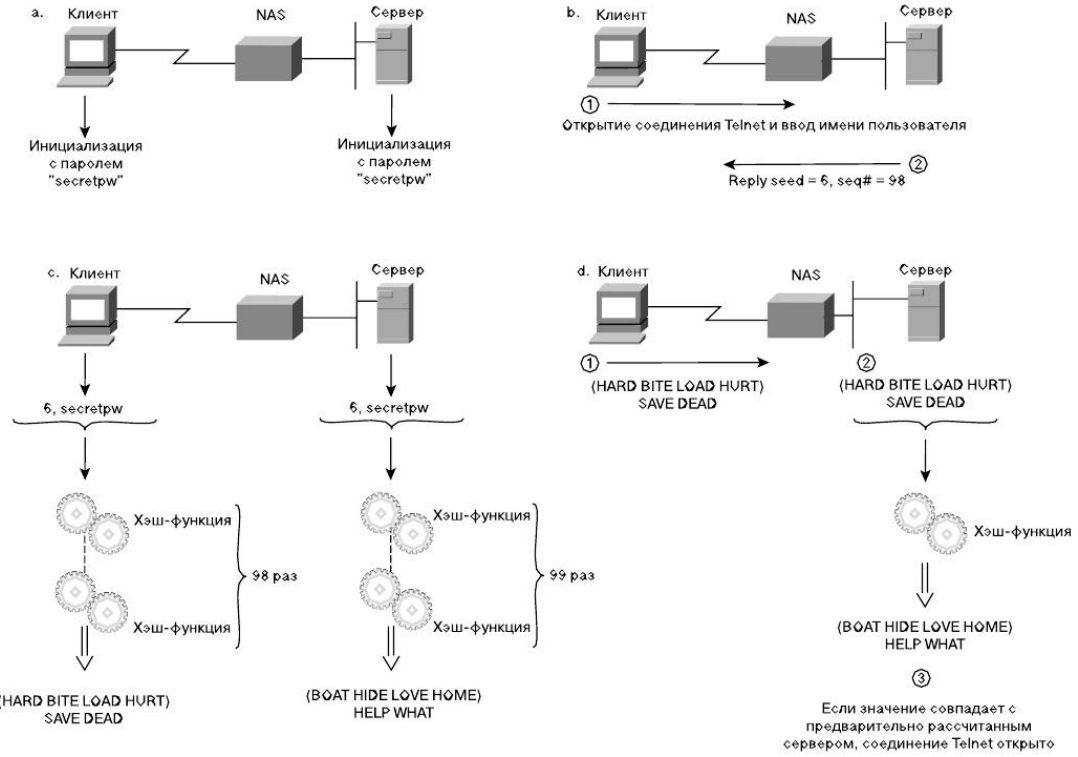

Рисунок 13. Функционирование системы S/Key

Для инициализации на стороне сервера используется команда: keyinit <user> <password> <seed> <sequence number>. Эту команду использует администратор для задания нового/изменения старого пользователя. Пояснение:

1. user - имя пользователя;
2. password - пароль пользователя от 1 до 16 символов;
3. seed - от 1 до 16 символов английские строчные и цифры;
4. sequence number - количество итераций для вычисления хэша, не включая предварительный шаг.

После инициализации сервера можно запускать клиент, который запрашивает у пользователя его имя (user), которое сразу же отправляется на сервер (пакет инициализации). Сервер пытается найти пользователя с именем user в своей базе данных (словаре). Если сервер отвечает “User not recognized”, то пользователь не найден, клиент завершает работу, иначе клиент получает seed и sequence number.

Затем, клиент находит конкатенацию seed и password. После выполняется подготовительный шаг - безопасная хэш-функция однократно вычисляется для конкатенации seed и password. Затем, безопасная хэш-функция выполняется sequence number раз над результатом подготовительного этапа.

Безопасная хэш-функция описана в RFC1760, кратко опишем суть её работы. Данная функция состоит из однократного применения функции хэширования, в данной реализации использовалась функция MD5, которая имеет выход 128 бит, затем результат хэширования сворачивается до 64 бит, при помощи операции xor, примененной к 1,3 и 2,4 частям 128 битного хэша.

Результат, полученный клиентом отправляется на сервер. Сервер запоминает полученный результат и выполняет над ним безопасную хэш-функцию 1 раз. Если результат хэширования совпал с тем, что хранится в БД (словаре), то хэш в БД перезаписывается на полученный от клиента результат, уменьшает sequence number на 1 и сервер открывает соединение (отсылает клиенту сообщение “Connection established”), иначе сервер отправляет клиент сообщение “Wrong password” (при этом клиент завершает работу).

Когда соединение установлено и sequence number становится равным -1, т.е. одноразовые пароли заканчиваются, то сервер посылает “Sequence number became -1. Use keyinit command!”, клиент в данном случае просит пользователя ввести команду keyinit. Также, команду keyinit может ввести пользователь при открытом соединении.

Команда “keyinit <пароль> <семя> <число итераций>” позволяет изменить пользователю свой пароль, семя и число итераций. При получении команды keyinit от пользователя клиент проверяет количество полей, которые ввел пользователь, если их меньше 4, то есть, возможно, не задано некоторое поле, то клиент просит повторить ввод, иначе, клиент выполняет конкатенацию seed и password, подготовительный этап и вычисляет безопасную хэш-функцию нужное число итераций, после отсылает команду keyinit <хэш> <семя> <число итераций - 1> на сервер. Сервер валидирует полученные данные и если валидация успешна то, изменяет эти данные у себя в БД (словаре).

Для реализации использовался язык python. Код представлен ниже.

Файл client.py

```
import socket
import hashlib

def s_key_secure_hash(msg):
    hash = hashlib.md5(msg).hexdigest()
    part_0 = int(hash[0:8], 16)
    part_1 = int(hash[8:16], 16)
    part_2 = int(hash[16:24], 16)
    part_3 = int(hash[24:32], 16)

    part_0 ^= part_2
    part_1 ^= part_3

    out = (part_0 << 32) | part_1
    out = out.to_bytes((out.bit_length() + 7) // 8, byteorder='little')

    return out

def change_pass(s, client_sock):
    s = s.split(" ")
    passwd = s[1]
    seed = s[2]
    rounds = int(s[3])
    hsh = seed + passwd
    hsh = s_key_secure_hash(hsh.encode())

    for i in range(rounds):
```

```
            hsh = s_key_secure_hash(hsh)

        hsh = hsh.decode('cp437')
        command = f"keyinit {hsh} {seed} {rounds-1}"
        client_sock.sendall(command.encode())

    def main():
        client_sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        client_sock.connect(('127.0.0.1', 12345))

        login = input("Введите логин:")
        client_sock.sendall(login.encode())
        resp = ((client_sock.recv(1024)).decode())


        if resp == "User not recognized":
            print("Пользователь не найден")
            return

        elif resp:
            resp = resp.split(" ")
            seed = resp[0]
            rounds = int(resp[1])

            passwd = input("Введите пароль:")

            #вычисляем нужные хэши и сохраняем их
            hsh = seed + passwd
            hsh = s_key_secure_hash(hsh.encode())
            for i in range(rounds):
                hsh = s_key_secure_hash(hsh)
            client_sock.sendall(hsh)

            while True:
                resp = client_sock.recv(1024).decode()
                if resp == "Sequence number became -1. Use keyinit command!":
                    keyinit = ""
                    while not keyinit.startswith("keyinit ") or
len(keyinit.split(" ")) != 4:
                        keyinit = input("Пожалуйста, используйте keyinit <пароль>
<семя> <число итераций> , так как ваши одноразовые пароли закончились!\n")

                    change_pass(keyinit, client_sock)
                    continue

                elif resp == "Connection established":
                    print("Соединение с сервером уставновлено")
                    print("Чтобы выйти введите close")

                    while True:
                        s = input("Ваше сообщение серверу:")
                        if s.startswith("keyinit "):
                            if len(s.split(" ")) != 4:
                                print("Введите команду корректно: keyinit
<пароль> <семя> <число итераций>")
                                continue
                            else:
                                change_pass(s, client_sock)
```

```
                    elif s == "close":
                        client_sock.sendall(s.encode())
                        client_sock.close()
                        print("Соединение закрыто")
                        return

                    else:
                        client_sock.sendall(s.encode())

                    resp = client_sock.recv(1024).decode()
                    print(f"Ответ сервера: {resp}")
            elif resp == "Wrong password":
                print("Неправильный пароль")
                return
    else:
        print("Соединение не удалось!")
        return

if __name__ == '__main__':
    main()
```

Файл server.py.

```
import socket
import hashlib
import string


def hashstep(msg):
    hashed = hashlib.md5(msg).hexdigest()
    part_0 = int(hashed[0:8], 16)
    part_1 = int(hashed[8:16], 16)
    part_2 = int(hashed[16:24], 16)
    part_3 = int(hashed[24:32], 16)
    part_0 ^= part_2
    part_1 ^= part_3
    out = (part_0 << 32) | part_1
    out = out.to_bytes((out.bit_length()+7)//8,'little')
    return out

def hashFunc(msg:str,seqn:int):
    hsh = hashstep(msg.encode())
    for i in range(seqn):
        hsh = hashstep(hsh)
    return hsh

def keyinitValidation(l:list):
    alphanumerics = string.ascii_lowercase + string.digits
    if len(l) == 4:
        if isinstance(l[1].encode('cp437'),bytes) and len(l[1]) == 8:
            if len(l[2]) > 0 and len(l[2]) < 17:
                for i in l[2]:
                    if alphanumerics.find(i) == -1:
                        return False
                for i in l[3]:
                    if string.digits.find(i) == -1:
                        return False
                if len(l[3]) > 1 and l[3][0] == '0':
                    return False
                return True
    return False

def main():
    d = dict()
```

```
        #d["user"]=[b"\x19\x90\x8c\x11r\xf6\xf8]","security",99]#bytes,str,int
        while True:
            cmd = input("Enter command:\n1 - add/update user\n2 - start server\n3
- show database\nexit - quit\nCommand:")
            if cmd == "1":
                key = input("Enter keyinit <user> <password> <seed> <sequence
number>:\n")
                splitlist = key.split(" ")
                hsh = hashFunc(splitlist[3]+splitlist[2],int(splitlist[4]))
                d.update({splitlist[1]:[hsh,splitlist[3],int(splitlist[4])-1]})
                continue
            elif cmd == "2":
                listener = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
                listener.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEADDR,1)
                IP = "127.0.0.1"
                PORT = 12345
                listener.bind((IP,PORT))
                listener.listen(0)

                while True:
                    try:
                        connection,address = listener.accept()
                        data = connection.recv(1024).decode()
                        userdata = d.get(data)
                        if  userdata == None:
                            connection.send("User not recognized".encode())
                        else:
                            connection.send((userdata[1] + " " +
str(userdata[2])).encode())
                            password = connection.recv(1024)
                            hashres = hashstep(password)
                            if hashres == userdata[0]:
                                seqn = userdata[2]-1
                                if seqn == -1:
                                    while True:
                                        connection.send("Sequence number became
-1. Use keyinit command!".encode())
                                        msg = connection.recv(1024).decode()
                                        if msg.startswith("keyinit "):
                                            splitlist = msg.split(" ")
                                            if keyinitValidation(splitlist):

d.update({data:[splitlist[1].encode('cp437'),splitlist[2],int(splitlist[3])]})
                                                break
                                else:
                                    d.update({data:[password,userdata[1],seqn]})
                                connection.send("Connection
established".encode())
                                while True:
                                    msg = connection.recv(1024).decode()
                                    if msg.startswith("keyinit "):
                                        splitlist = msg.split(" ")
                                        if keyinitValidation(splitlist):

d.update({data:[splitlist[1].encode('cp437'),splitlist[2],int(splitlist[3])]})
                                            connection.send("Keyinit command used
successfully".encode())
                                        else:
                                            connection.send("Keyinit command used
unsuccessfully".encode())
                                    elif  msg == "close":
                                        break
                                    else:
                                        connection.send(("Server recieved from
user:" + msg).encode())
                            else:
                                connection.send("Wrong password".encode())
                    except Exception:
```

```
                    continue
            elif cmd == "3":
                print(d)
            elif cmd == "exit":
                return
            else:
                print("Wrong command!")

if __name__ == '__main__':
    main()
```