哈爾濱工業大學

网络安全实验报告

题　　目 基于口令的认证过程实现

专　　业 计算机科学与技术

学　　号 7203610316

学　　生 符兴

指 导 教 师 王彦

一、实验目的

掌握随机函数的使用，掌握散列函数，加解密函数的使用。包的格式与发包的顺序，可以等同于协议的三要素。掌握程序与数据库的链接。

二、 实验内容

1. 客户端输入用户名，口令，随机产生认证码，使用散列函数计算用户名与口令的散列值 1，使用散列值 1 与认证码计算散列值 2，将用户名，散列值 2，认证码明文传送到服务器端。

2. 服务器端以数据库（如 access）保存用户名和散列值 1 的对应关系。收到客户端信息后，以同样的方法计算散列值 2'。 如散列值 2'=散列值 2，则认证成功，成功后用散列值 1 加密认证码发送给客户端。客户端解密后写到指定文件。用户可以修改自己的密码。。

三、实验过程

基于口令的认证过程实现

实验基本信息：

实验环境：Windows11 x64，Mysql8.0

编程语言：Python

1. 需求分析

需要编写服务端和客户端相应的程序，并且按照一定的协议实现消息的加密发送，其中比较重要的功能函数有：

(1) 随机认证码的生成

(2) 散列函数

(3) DES 加解密算法

2. 程序结构

(1) 客户端程序的大致结构为：

① 通过 TCP 协议连接服务器。

② 正确连接服务器后，用户键入用户名、口令，等待认证。

③ 生成随机认证码 a，计算用户名和口令的散列值 b；然后计算随机认证码 a 和散列值 b 的散列值 c。

```
# 服务器连接
serverSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
serverSocket.connect((args.addr, args.port))
print("已成功连接上服务器")

# 获取用户名、口令
userName = input("请输入登录用户名: ")
password = input("请输入登录口令: ")

# 生成随机认证码
certCode = certCodeGeneration(16)
print("所生成的随机认证码为:  {}".format(certCode))

# 计算散列值1
hash1 = hashlib.md5((userName+password).encode("utf-8"))
hash1Str = hash1.hexdigest()

# 计算散列值2
hash2 = hashlib.md5((hash1Str+certCode).encode("utf-8"))
hash2Str = hash2.hexdigest()
```

图 1 准备数据阶段代码截图

④ 将用户名，散列值 c 以及随机认证码明文发送到服务器端，等待服务器的认证。

⑤ 若认证成功，可以通过随机认证码对服务端发送来的消息进行解密，并写入相应的文件中。在这里，消息的加密算法为 DES，加密的密钥是散列值 b。

```
# 将用户名，散列值2，认证码明文传送到服务器端
sendData = userName+" "+hash2Str+" "+certCode
sendData = sendData.encode("utf-8")
serverSocket.send(b'000'+sendData)

# 接收服务端发送来的数据
recvData = serverSocket.recv(100)
if recvData[0:3] == b'001':
    desC = DESModel()
    recvStr = desC.decrypt(recvData[3:], hash1Str[0:8])
    with open("./{}_{}.txt".format(userName, int(time.time())),"w") as fil
        file.writelines(recvStr+"\n")
else:
    print("登陆失败，请检查密码")
    exit(0)
```

图 2 等待服务器发送认证结果

⑥ 用户可以向服务器发起修改密码的请求。

⑦ 获取用户新的密码，并重复 2 的过程；但是在发送时候需要将散列值 b 发送给服务端，因为服务端需要保存该散列值用于后续的认证。

```
if recvData[0:3] == b'001':
    newPass = input("请输入新的密码: ")
    # 生成随机认证码
    certCode = certCodeGeneration(16)
    print("所生成的随机认证码为:  {}".format(certCode))

    # 计算散列值1
    hash1 = hashlib.md5((userName+newPass).encode("utf-8"))
    hash1Str = hash1.hexdigest()

    # 计算散列值2
    hash2 = hashlib.md5((hash1Str+certCode).encode("utf-8"))
    hash2Str = hash2.hexdigest()

    # 将用户名，散列值1，认证码明文传送到服务器端
    sendData = userName+" "+hash1Str+" "+certCode
    sendData = sendData.encode("utf-8")
    serverSocket.send(sendData)

    # 接收服务端发送来的数据
    recvData = serverSocket.recv(100)
    if recvData[0:3] == b'001':
        desC = DESModel()
        recvStr = desC.decrypt(recvData[3:], hash1Str[0:8])
        with open("./{}_{}.txt".format(userName, int(time.time())),"w"
            file.writelines(recvStr+"\n")
        print("密码修改成功")
    else:
        print("修改密码失败")
        exit(0)
```

图 3 用户修改密码过程

(2) 服务端程序的大致结构为：

① 监听连接请求

```
data = client.recv(100)
if data[0:3] == b'000':
    data = data[3:]
    data = data.decode("utf-8")
    data = data.split(" ")
    hashFromSql = self.getUserHash(data[0])
    if hashFromSql == "Not Found":
        print("当前用户 {} 未进入数据库".format(data[0]))
    else:
        tmpHash = hashlib.md5((hashFromSql+data[2]).encode("utf-8")
        if tmpHash.hexdigest() == data[1]:
            print("当前用户 {} 通过认证".format(data[0]))
            desC = DESModel()
            enCode = desC.encrypt("wolaile", hashFromSql[0:8])
            client.send(b'001'+enCode)
        else:
            print("当前用户 {} 认证失败".format(data[0]))
            client.send(b'000000')
```

图 4 服务端认证过程

② 在收到一个连接认证请求时，服务端需要对其进行认证。首先通过用户名在数据库查找用户对应的散列值 b，然后将散列值 b 和接收到的随机认证码生成新的散列值 c′，将新散列值 c′和接收的散列值 c 进行比对；如果二者相同则认证成功；如果二者不同，则认证失败；

③ 如果当前用户认证成功，则服务端会使用用户的散列值 b 对一个消息进行加密并将其发送给客户端；

④ 如果当前用户发送修改密码请求，服务端会接受来自客户端的用户名，散列值 b 和随机认证码；然后将数据库中该用户对应的散列值更新为新的散列值 b；完成更新后向客户端发送响应。

```
elif data[0:3] == b'002':
    client.send(b'001000')
    data = client.recv(100)
    data = data.decode("utf-8")
    data = data.split(" ")
    hashFromSql = self.updateHash(data[0], data[1])
    if hashFromSql == "Not Update":
        client.send(b'000000')
        print("当前用户 {} 更新密码失败".format(data[0]))
    else:
        print("当前用户 {} 更新密码成功".format(data[0]))
        desC = DESModel()
        enCode = desC.encrypt("wolaile", data[1][0:8])
        print(enCode)
        client.send(b'001'+enCode)
```

图 5 服务端相应客户端修改密码请求

(3) DES 加解密算法的大致结构为：由于算法限制，这里的密钥取得是所输入散列值的前 8 位；

```
def encrypt(self, text, key):
    try:
        text = text.encode('utf-8')
        cryptor = DES.new(key.encode("utf-8"), self.mode, key.encode("utf-8"))
        length = 16
        count = len(text)
        if count < length:
            add = (length - count)
            text = text + ('\0' * add).encode('utf-8')
        elif count > length:
            add = (length - (count % length))
            text = text + ('\0' * add).encode('utf-8')
        self.ciphertext = cryptor.encrypt(text)
        return b2a_hex(self.ciphertext)
    except:
        return ""
```

图 6 DES 加密

```python
def decrypt(self, text, key):
    try:
        cryptor = DES.new(key.encode('utf-8'), self.mode, key.encode('utf-8'))
        plain_text = cryptor.decrypt(a2b_hex(text))
        # return plain_text.rstrip('\0')
        return bytes.decode(plain_text).rstrip('\0')
    except:
        return ""
```

图 7 DES 解密

四、实验结果

(1) MySql 数据库保存的用户信息：

图 8 数据库保存的信息

(2) 程序运行截图：

```
(base) D:\NetworkSecurity\lab4>python client.py
已成功连接上服务器
请输入登录用户名: test
请输入登录口令: 111
所生成的随机认证码为:  b44Wm0f6265VOZoh
32
所生成的散列值1为 4061863caf7f28c0b0346719e764d561
所生成的散列值2为 91682a65718ffada60296fc363b47919
是否需要修改数据[0不修改，1修改]: 1
请输入新的密码: 5555
所生成的随机认证码为:  EmbRV316I5jOr7p2
32
所生成的散列值1为 ce3463b17b552a3d74e9603836db71c3
所生成的散列值2为 02ed0a5e3c6dbc61478316118476fb32
密码修改成功

(base) D:\NetworkSecurity\lab4>python client.py
已成功连接上服务器
请输入登录用户名: test
请输入登录口令: 5555
所生成的随机认证码为:  j8fIYjOXfyr6bYGx
32
所生成的散列值1为 ce3463b17b552a3d74e9603836db71c3
所生成的散列值2为 b3017ab273669735d7f4f518aa30f259
是否需要修改数据[0不修改，1修改]: 0
已正确接收，再见
```

图 9 客户端流程截图

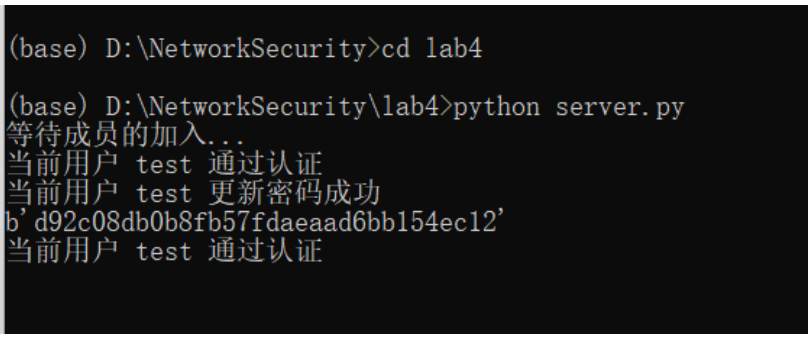

图 10 服务端流程截图

(3) 客户端将消息解密并写入文件中（这里使用了自定义的消息）：

图 11 客户端解密消息写入文件

五、心得体会

(1) 进一步了解对消息加密发送的过程，并掌握了散列函数，DES 加密算法。

(2) 更深入了解了网络中两台主机之间发送数据的具体过程，并通过自定义的协议实现对消息的加密传送，并且在接收端可以正确解密并写入对应的文件中；