ISA 656, Project 2

G00773152: Huangxin Wang

G00841583: Li Liu

Contents

1 Group member

G00773152: Huangxin Wang
G00841583: Li Liu

2 Security feature design choice

2.1 Password Storage

We implement class “EncryptPassword” which will take the <username, password> pair as input, and salted the password, then hash the salted password, finally it will store encrypted <username, salt, hashed salted password> info in the disk.
Whenever server is up, it will read the encrypted password file from disk, decrypt it, and store the result in the memory.
Note: we hardcode the plaintext password file name and encrypted password file name. They are named “input.txt” and “output” respectively.

2.2 Challenge and response scheme

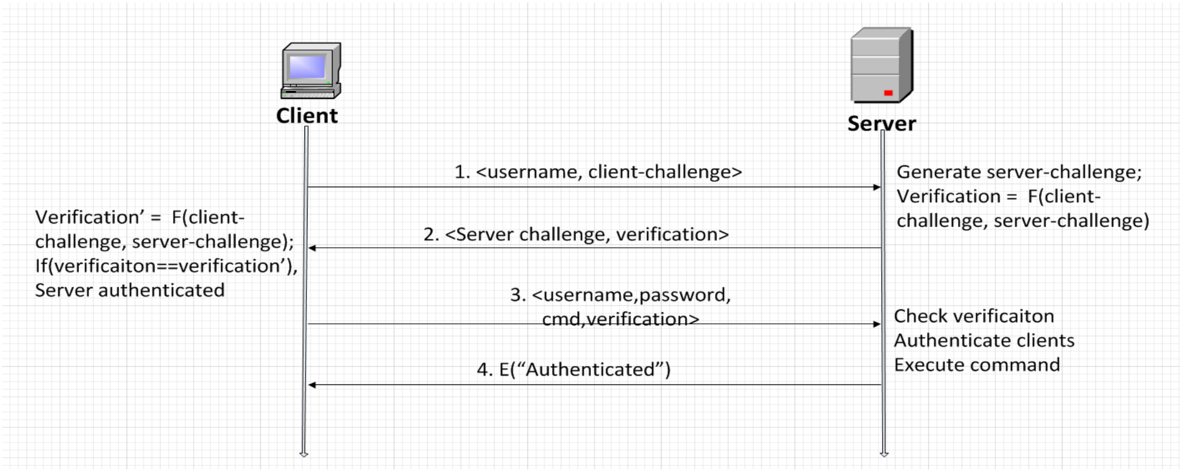
The prerequisite is client and server sharing a secret function beforehand. The challenge and response procedure is shown in the following figure.

Step1: client send connect request to server, the message contain its usename and a random generate challenge.

Step2: server receives the request. It keep generate a random challenge, and use the shared secret function to get a verification by using the client-challenge and server-challenge as input. The server uses the generated verification code to authenticate himself to the client. The message the server send to the client will contain the server challenge and verification.

Step3: The client use the share secret function to perform the same operation, if the output is equal to the verification get from server, then server is authenticated. It will then send his password, username, verification, command to authenticate himself to the server by password authentication. If the client is authenticated, the command will be executed.

Note: since the challenge is randomly generated each time, its freshness will save the challenge and response scheme from reply attack.


Client
Server
1. <username, client-challenge>
Generate server-challenge; Verification = F(client-challenge, server-challenge)
2. <Server challenge, verification>
Verification’ = F(client-challenge, server-challenge); If(verificaiton==verification’), Server authenticated
3. <username,password, cmd,verification>
Check verificaiton
Authenticate clients
Execute command
4. E(“Authenticated”)

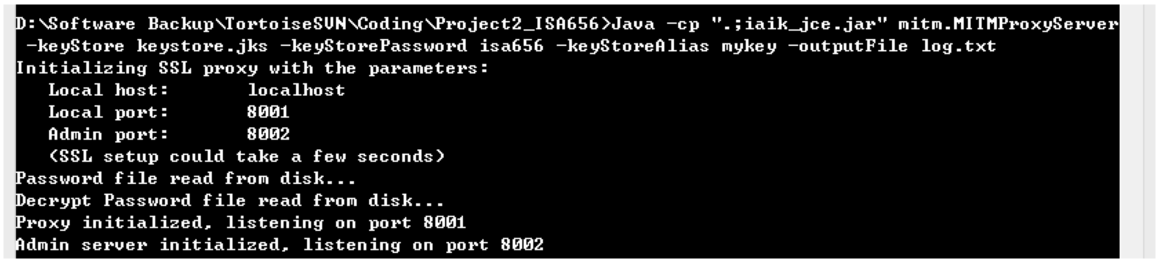
2.3 Password authentication

At the final step of challenge and response, password authentication will be conducted by the server. The server will first lookup the salt of the corresponding user in his memory, then he will use the salt to salt the password, and then hash the salted result. If the hashed result is the same with the one in the memory, the client is then authenticated, otherwise, it will be rejected.

3 Steps to run the system

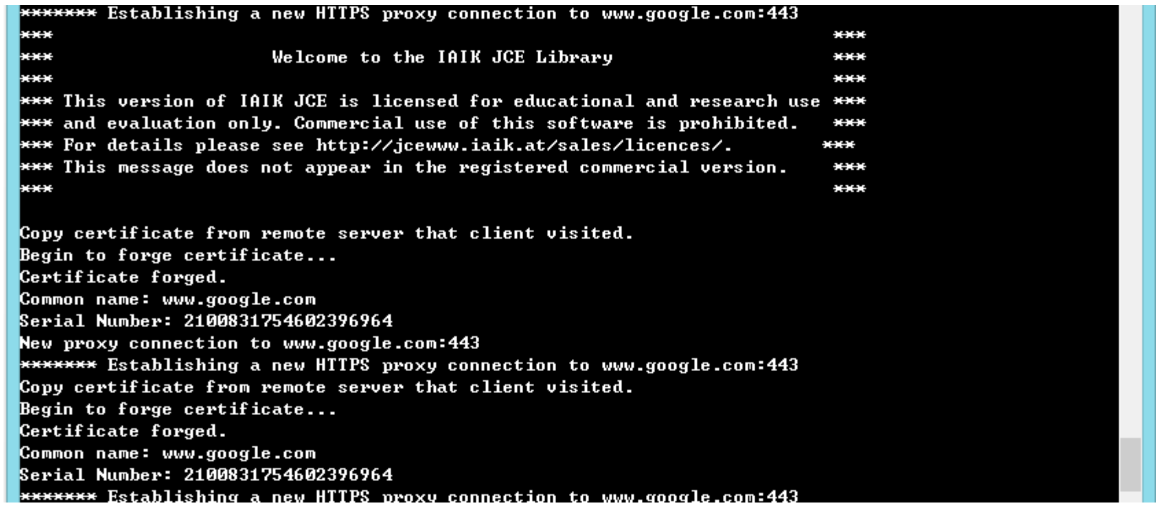
3.1 Run the proxy server (server side)

```
Java -cp ".;iaik_jce.jar" mitm.MITMProxyServer -keyStore keystore.jks -
keyStorePassword isa656 -keyStoreAlias mykey -outputFile log.txt
```



```
D:\Software Backup\TortoiseSVN\Coding\Project2_ISA656>Java -cp ".;iaik_jce.jar" mitm.MITMProxyServer
 -keyStore keystore.jks -keyStorePassword isa656 -keyStoreAlias mykey -outputFile log.txt
Initializing SSL proxy with the parameters:
   Local host:       localhost
   Local port:       8001
   Admin port:       8002
   (SSL setup could take a few seconds)
Password file read from disk...
Decrypt Password file read from disk...
Proxy initialized, listening on port 8001
Admin server initialized, listening on port 8002
```

3.2 Use browser to visit https sites (server side)

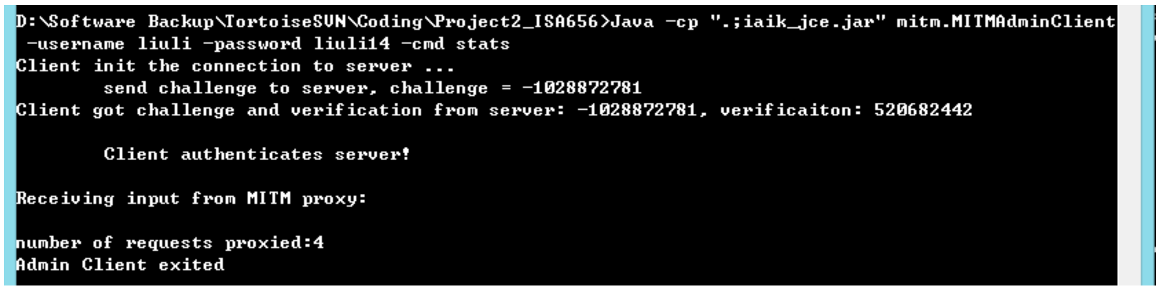


```
******* Establishing a new HTTPS proxy connection to www.google.com:443
***                                                                      ***
***                    Welcome to the IAIK JCE Library                   ***
***                                                                      ***
*** This version of IAIK JCE is licensed for educational and research use ***
*** and evaluation only. Commercial use of this software is prohibited.  ***
*** For details please see http://jcewww.iaik.at/sales/licences/.       ***
*** This message does not appear in the registered commercial version.  ***
***                                                                      ***

Copy certificate from remote server that client visited.
Begin to forge certificate...
Certificate forged.
Common name: www.google.com
Serial Number: 2100831754602396964
New proxy connection to www.google.com:443
******* Establishing a new HTTPS proxy connection to www.google.com:443
Copy certificate from remote server that client visited.
Begin to forge certificate...
Certificate forged.
Common name: www.google.com
Serial Number: 2100831754602396964
******* Establishing a new HTTPS proxy connection to www.google.com:443
```

3.3 Use another CMD to run admin client, use command stats/shutdown (client side)

stats: List how many requests were proxied

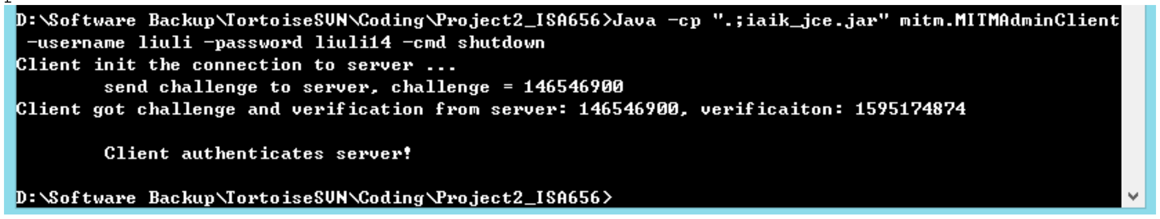
```
Java -cp ".;iaik_jce.jar" mitm.MITMAdminClient -username liuli -
password liuli14 -cmd stats
```



```
D:\Software Backup\TortoiseSVN\Coding\Project2_ISA656>Java -cp ".;iaik_jce.jar" mitm.MITMAdminClient
 -username liuli -password liuli14 -cmd stats
Client init the connection to server ...
        send challenge to server, challenge = -1028872781
Client got challenge and verification from server: -1028872781, verificaiton: 520682442

        Client authenticates server!

Receiving input from MITM proxy:

number of requests proxied:4
Admin Client exited
```

shutdown: shutdown the MITM proxy server

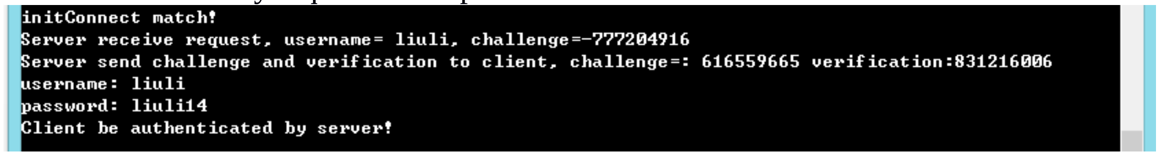
```
Java -cp ".;iaik_jce.jar" mitm.MITMAdminClient -username liuli -
password liuli14 -cmd shutdown
```



```
D:\Software Backup\TortoiseSVN\Coding\Project2_ISA656>Java -cp ".;iaik_jce.jar" mitm.MITMAdminClient
 -username liuli -password liuli14 -cmd shutdown
Client init the connection to server ...
        send challenge to server, challenge = 146546900
Client got challenge and verification from server: 146546900, verificaiton: 1595174874

        Client authenticates server!

D:\Software Backup\TortoiseSVN\Coding\Project2_ISA656>
```

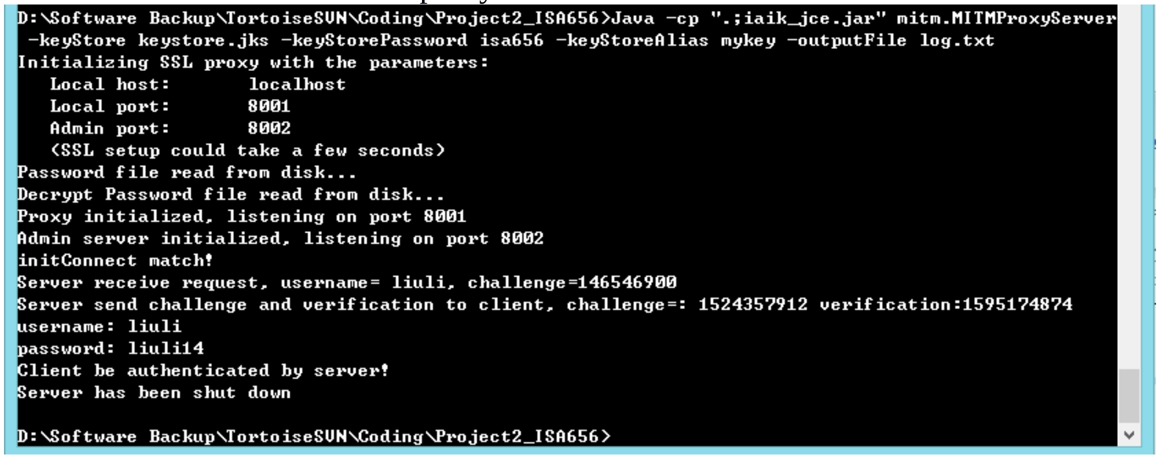
3.4 What happens on admin server (proxy server) side

stats: List how many requests were proxied



```
initConnect match!
Server receive request, username= liuli, challenge=-777204916
Server send challenge and verification to client, challenge=: 616559665 verification:831216006
username: liuli
password: liuli14
Client be authenticated by server!
```

shutdown: shutdown the MITM proxy server



```
D:\Software Backup\TortoiseSVN\Coding\Project2_ISA656>Java -cp ".;iaik_jce.jar" mitm.MITMProxyServer
 -keyStore keystore.jks -keyStorePassword isa656 -keyStoreAlias mykey -outputFile log.txt
Initializing SSL proxy with the parameters:
   Local host:       localhost
   Local port:       8001
   Admin port:       8002
   (SSL setup could take a few seconds)
Password file read from disk...
Decrypt Password file read from disk...
Proxy initialized, listening on port 8001
Admin server initialized, listening on port 8002
initConnect match!
Server receive request, username= liuli, challenge=146546900
Server send challenge and verification to client, challenge=: 1524357912 verification:1595174874
username: liuli
password: liuli14
Client be authenticated by server!
Server has been shut down

D:\Software Backup\TortoiseSVN\Coding\Project2_ISA656>
```

4 keystore: name and passwords

Name: “mykey”
Password: “isa656”

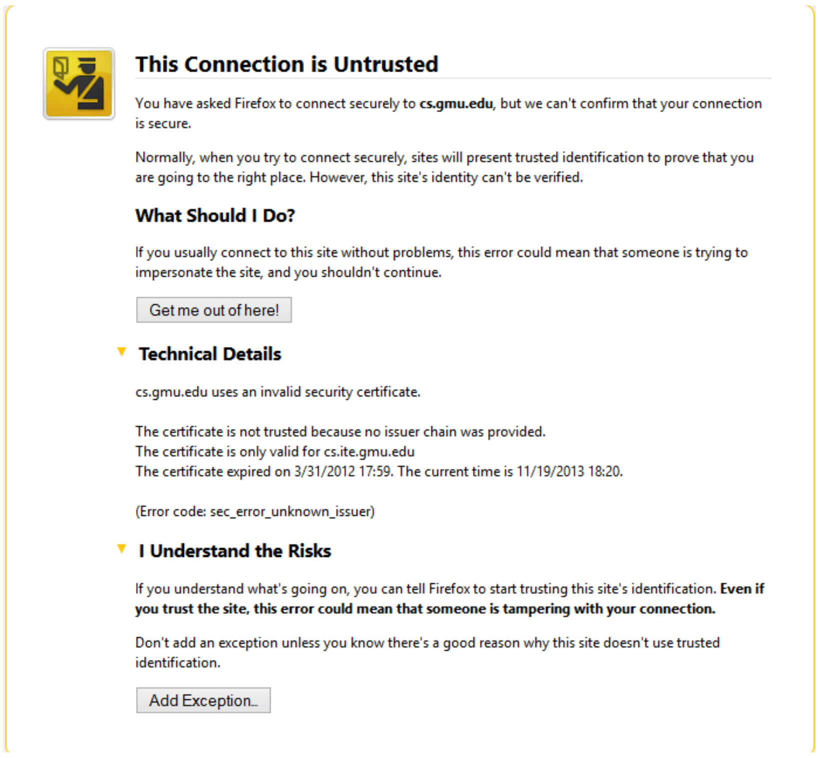
5 A copy of proxy log file from the test

See log.txt file.
This is a section of sample taken from this file:

```
--- localhost:50399->www.google.com:443 opened --
--- www.google.com:443->localhost:50399 opened --
------ localhost:50399->www.google.com:443 ------
G
------ localhost:50399->www.google.com:443 ------
ET /s?gs_rn=31&gs_ri=psy-
ab&pq=duck&cp=1&gs_id=1ik&xhr=t&q=p&es_nrs=true&pf=p&output=search&scli
ent=psy-
ab&oq=&gs_l=&pbx=1&bav=on.2,or.r_qf.&bvm=bv.56643336,d.dmg&fp=6a6b436c0
215037f&biw=1600&bih=763&bs=1&tch=1&ech=2&psi=mbqLUvTvJrKx4AP6pICADw.13
84888981503.7 HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:25.0) Gecko/20100101
Firefox/25.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://www.google.com/
Cookie:
PREF=ID=ad2cd3c8e7c09a4a:U=362f245a34a1586c:FF=0:TM=1384635201:LM=13846
44354:S=GaBiSAcgos1f5jdf;
NID=67=ZiBnoworen4OsEdTUsFj2mFzJZxMSMrt30nsiM2IQYUzd40NeBpgpkmA9Zj-
HhJHDOjKUxIeOZXYF5qo0uKzzel5fDUsRQBLqI7CIciI1vXGSrwUoT_d6JC8hERU09yz
Connection: keep-alive
```

6 A short answer to the following question: How would you change a web browser to make it less likely that a user would be fooled by an attack like the one you implemented?

In out implementation of MITM attack using proxy, the web browser did inform us the untrusted of the website. However, this is not enough for user to learn the possible danger of this warning, and users most likely have no idea what this means to them. So often the users just proceed anyway. So it is better to have more user friendly, less technical information display to users when security issues arise.


This Connection is Untrusted

You have asked Firefox to connect securely to **cs.gmu.edu**, but we can't confirm that your connection is secure.

Normally, when you try to connect securely, sites will present trusted identification to prove that you are going to the right place. However, this site's identity can't be verified.

What Should I Do?

If you usually connect to this site without problems, this error could mean that someone is trying to impersonate the site, and you shouldn't continue.

Technical Details

cs.gmu.edu uses an invalid security certificate.

The certificate is not trusted because no issuer chain was provided.
The certificate is only valid for cs.ite.gmu.edu
The certificate expired on 3/31/2012 17:59. The current time is 11/19/2013 18:20.

(Error code: sec_error_unknown_issuer)

I Understand the Risks

If you understand what's going on, you can tell Firefox to start trusting this site's identification. **Even if you trust the site, this error could mean that someone is tampering with your connection.**

Don't add an exception unless you know there's a good reason why this site doesn't use trusted identification.

In addition, in terms of for how long to trust the certificate which has potential risk, the default option is permanent. As the lazy client are always likely to accept the default settings, once client click “trust” for once, he will get no warning of the certificate from then on. Therefore we suggest to set the default option as temporally in order to remind the lazy clients of the potential risk more often.