Security Design Document

CS557 Project 2

Yunhe Tang and Xuebin He

1. **The adversary capabilities expected, and the goals of the adversary (his payoffs).**

   The adversary is capable of see the source code, file content of the program. But the adversary is forbidden to delete or modify the file existed. However, deleting or changing any of the files can only corrupt the program but no information will be leaked to adversary. In this sense, the system is still secure. In addition, since the java runnable file is rely on JRE stalled in local computer, the situation that the JRE library is maliciously modified is eliminated from our consideration.

2. **The system goals, briefly summarizing the intended functionality, and emphasizing the security goals you want to achieve.**

   There are several secure aspects that we concern about.

   1) User authentication: Distributed File System (DFS) is consists of several nodes that communicate with each other. Only admin is permitted to launch/shutdown a node. We ask user to input admin password to verify if the user has the authority to act on node.
   2) Confidentiality: All the data transmit inside DFS network is encrypted. Therefore anyone outside the network is incapable of knowing what is going on in DFS network.
   3) Data integrity: we take advantage of Digital Signature to verify if the content of the message is modified by others during transmission.
   4) End point authentication: Combining Digital Signature and Certificate Authentication, we are capable of provide verify that if the peer is really the person who (s)he claimed to be. CA is a trusted third party organization that distributes certificates.

3. **User profile: What do you expect the user to do to keep your system secure, and why do you think the user will want to do that? Why do you think the user will know how to do that?**

   To use our system safely, the user must try to prevent any unauthorized users log into their operating system (the most effective way to be secure). If the adversary somehow get the access into the system, as long as (s)he does not have the admin password, the data is still unreadable. For essential data, we can store it on other node, making the data secure even if the local data file is deleted. If the password file is deleted, whoever launch the node can set a new password for this node. This design is merely for convenience. In this way anyone can set their password and test our program without any information from us.

4. **Security mechanisms: What mechanisms does your project use to achieve its security goals?**

   Security goals are achieved by two ways of cryptographies: Symmetric Cryptography and Asymmetric Cryptography.

   1) We take advantage of AES encryption algorithm to encrypt the data that should be stored in file. We take the MD5 value of the admin password as a key to encrypt the data.
   2) We design and implement a simple version of a TSL. The algorithm we use is RSA algorithm. We designed our own CA to distribute the certificates. Here there are two pre-conditions: CA is trustable and process of issuing certificates is secured. There are mainly three main function on this part:
      - Key generating and certificate retrieving.
      - Perform Handshaking procedure to establish a TSL connection after TCP connection is on.
      - Encrypt and decrypt messages after the TSL is on.

   NOTE: the private/public key for CA is fixed (in file CA_key.txt), while the private/public key for each node is generated at runtime dynamically.

User Guide

This is a guide for users (anti-team) to get to know on to run our program.

Initially their three sub-directories in directory “DFS_TSL Test”, each named “Node[x]”. [x] stands for number, for example Node1, Node2…

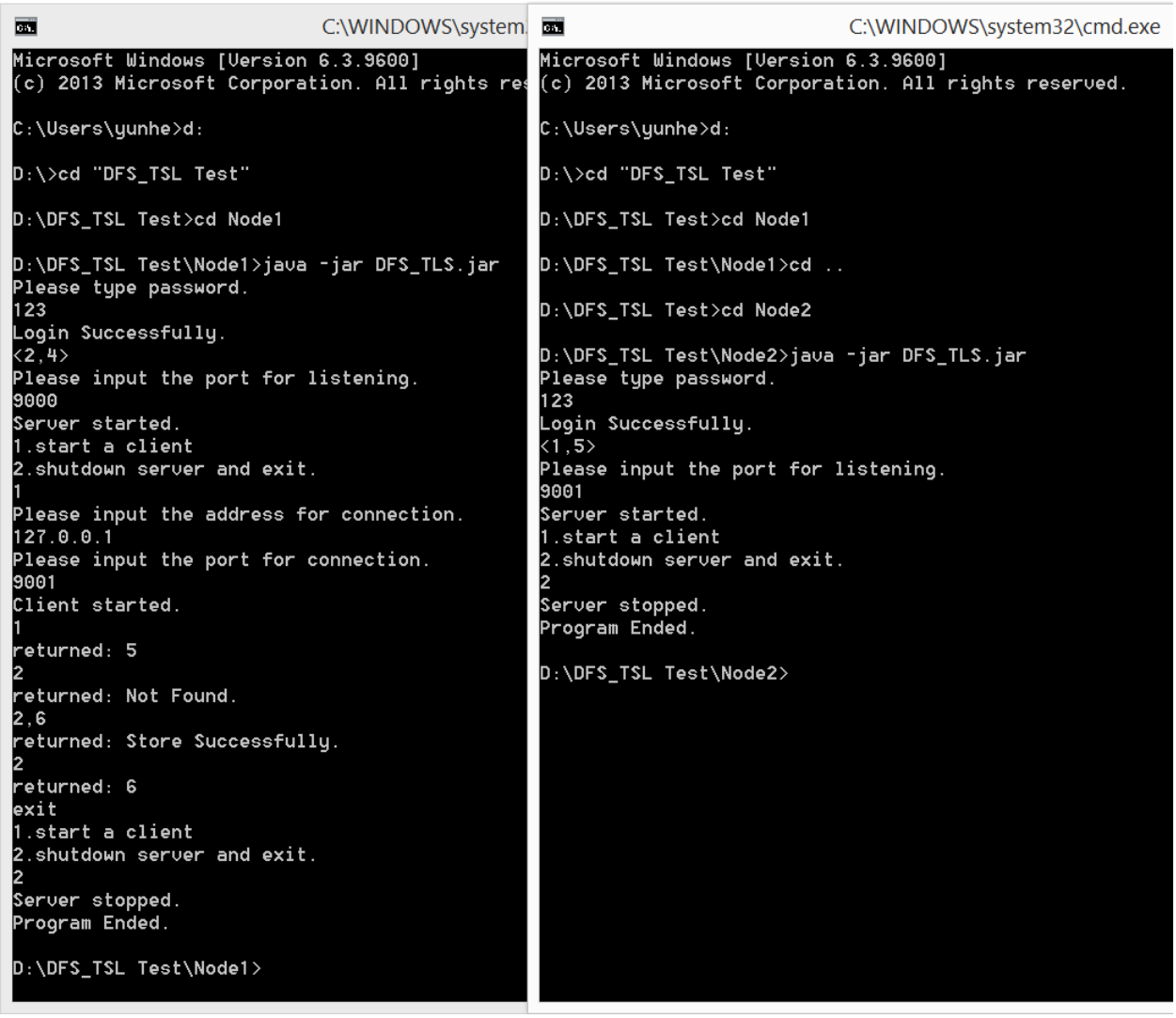
Namely each “Node[x]” directory represent a node. Inside the Node[x] directory, there are at most four files in it (Initially it may contains just two). They are:

1) DFS_TSL.jar: a java runnable file. Type “java –jar DFS_TSL.jar” under current directory to run the file in terminal.
2) CA_key.txt: Stores CA’s private and public key.
3) StoredData.txt: Data is store in it.
4) Tmp.txt: admin password is stored in it.

You can create more nodes by simply copy and paste a Node[x] directory under “DFS/_TSL Test” folder with renaming.

Each node can only communicate with one other node. To store/retrieve data on specific node, create a client and input the target IP address and port number to connect to it. Then type number in format “[Num]” or “[Num],[Num]” to store or retrieve data.

A concrete example is on next page.


C:\WINDOWS\system
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights res
C:\Users\yunhe>d:
D:\>cd "DFS_TSL Test"
D:\DFS_TSL Test>cd Node1
D:\DFS_TSL Test\Node1>java -jar DFS_TL$.jar
Please type password.
123
Login Successfully.
<2,4>
Please input the port for listening.
9000
Server started.
1.start a client
2.shutdown server and exit.
1
Please input the address for connection.
127.0.0.1
Please input the port for connection.
9001
Client started.
1
returned: 5
2
returned: Not Found.
2,6
returned: Store Successfully.
2
returned: 6
exit
1.start a client
2.shutdown server and exit.
2
Server stopped.
Program Ended.
D:\DFS_TSL Test\Node1>
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.
C:\Users\yunhe>d:
D:\>cd "DFS_TSL Test"
D:\DFS_TSL Test>cd Node1
D:\DFS_TSL Test\Node1>cd ..
D:\DFS_TSL Test>cd Node2
D:\DFS_TSL Test\Node2>java -jar DFS_TL$.jar
Please type password.
123
Login Successfully.
<1,5>
Please input the port for listening.
9001
Server started.
1.start a client
2.shutdown server and exit.
2
Server stopped.
Program Ended.
D:\DFS_TSL Test\Node2>