

# SFTP

## [CSED331] Term Project - SFTP

이성환 (20130486)

생명과학과

## Introduction

SFTP(SSH File Transfer Protocol / Secure File Transfer Protocol, SFTP)은 신뢰할 수 있는 데이터 스트림을 통해 파일 접근, 파일 전송, 파일 관리를 제공하는 네트워크 프로토콜입니다. SSH 프로토콜을 통해 실행되며 SSH의 보안 및 인증 기능을 지원합니다.

SFTP는 또한 Password Sniffing 및 Man-in-the-Middle Attack 공격으로부터 보호합니다. 암호화 및 암호화 해시 기능을 사용하여 데이터의 무결성을 보호하고 서버와 사용자를 모두 인증합니다.

이번 프로젝트에선 Linux Ubuntu 환경에서 동작하고 SFTP의 동일한 기능을 지원하는 FTP를 구현합니다.

구현에 요구되는 사항은 아래와 같습니다.

### 1. Experimental setup

- In server side, we execute `python ftp_server.py` to construct server.
- In client side, we execute `python ftp_client.py` to execute client.  
Then, the program will immediately require authentication information.

### 2. Requirements - Correctness

- A client has to authenticate their identity.
- A client can traverse the server directory structure.
- A client can traverse the client directory structure.
- A file needs to be uploaded from client to server
- A file needs to be downloaded from server to client.
- Multiple file could be uploaded from client to server.
- Multiple file could be downloaded from server to client.
- A server can handle more than 100 clients at the same time.

### 3. Requirements - Performance

- The upload speed will be similar to that of existing implementation.
- The download speed will be similar to that of existing implementation.

### 4. Clients Interface

- `cd path` : Change remote directory to 'path'
- `exit` : Quit sftp
- `get [-afPpRr] remote [local]` : Download file
- `lcd path` : Change local directory to 'path'
- `lls [ls-options] [path]` : Display local directory listing
- `lpwd` : Print local working directory
- `ls [-lafhlNrSt] [path]` : Display remote directory listing
- `put [-afPpRr] local [remote]` : Upload file
- `pwd` : Display remote working directory

## Execute

### 1. Server 실행

`ftp_server.py` 에서 아래 코드 부분을 수정합니다.

```
#TODO: Set your INFO
BIND_PORT = 2200
CUSTOM_USER_NAME = 'root'
CUSTOM_PASSWD = '1234'
key_filename = "id_rsa"
```

아래 명령어로 SFTP 서버를 실행합니다. `private_key_file` 에는 RSA private key file의 path를 입력합니다.

```
$ python ftp_server.py private_key_file
Listening for connection ...
```

`Listening for connection ...` 메시지가 뜬다면 클라이언트의 접속이 가능한 상태입니다.

## 2. Client 실행

`ftp_server.py` 에서 아래 코드 부분을 수정합니다. Server와 동일한 RSA로 설정합니다.

```
#TODO : Set your RSA KEY  
key_file = "./id_rsa"
```

아래 명령어로 SFTP 클라이언트를 실행합니다. `user` 에는 접속 username, `host` 는 서버 IP 혹은 url, `port` 는 접속 포트를 입력합니다. 이후 패스워드 입력 창에 패스워드를 입력합니다.

```
$ python ftp_client.py user@host -p port  
Password:
```

```
Hello~!. sftp excute.  
-----Connect Info-----  
Connect User Name | 127.0.0.1  
Connect Host Name | root  
Connect Host Port | 2200  
-----  
[127.0.0.1:2200] Successful connection  
Welcome! sftp connect success~!!  
custom_sftp> █
```

figure1. SFTP Client 접속 화면

## 3. Local 명령어

- `lpwd` : 현재 로컬 디렉토리 위치를 알려줍니다.

```
Hello~!. sftp excute.  
-----Connect Info-----  
Connect User Name | 127.0.0.1  
Connect Host Name | root  
Connect Host Port | 2200  
-----  
[127.0.0.1:2200] Successful connection  
Welcome! sftp connect success~!!  
custom_sftp> lpwd  
[Local] /net_workdir/project  
custom_sftp> █
```

figure2. SFTP Client lpwd 명령어 실행 화면

- `lls [ls-options] [path]` : 현재 로컬 디렉토리에 위치한 파일 리스트를 보여줍니다.

```

Hello~!. sftp excute.
-----Connect Info-----
Connect User Name | 127.0.0.1
Connect Host Name | root
Connect Host Port | 2200
-----
[127.0.0.1:2200] Successful connection
Welcome! sftp connect success~!!
custom_sftp> lpwd
[Local] /net_workdir/project
custom_sftp> ll
arg_parse.py      ftp_client.py  id_rsa          shell_cmd.py    test            test_client.py
demo_server.log   ftp_server.py  id_rsa.pub      termB_FTP.pdf   test.txt
custom_sftp> █

```

figure3. SFTP Client ll 명령어 실행 화면

```

[Local] /net_workdir/project
custom_sftp> ll
arg_parse.py      ftp_client.py  id_rsa          shell_cmd.py    test            test_client.py
demo_server.log   ftp_server.py  id_rsa.pub      termB_FTP.pdf   test.txt
custom_sftp> ll -l
total 144
-rw-r--r-- 1 root root    599 Jun 29 04:30 arg_parse.py
-rw-r--r-- 1 root root      0 Jul  2 10:50 demo_server.log
-rw-r--r-- 1 root root  8109 Jul  3 02:47 ftp_client.py
-rw-r--r-- 1 root root 13974 Jul  3 02:49 ftp_server.py
-rw-r--r-- 1 root root  1823 Jul  2 04:24 id_rsa
-rw-r--r-- 1 root root   399 Jul  1 07:37 id_rsa.pub
-rw-r--r-- 1 root root   303 Jun 28 08:34 shell_cmd.py
-rw-r--r-- 1 root root 96415 Jun 28 06:40 termB_FTP.pdf
drwxr-xr-x 4 root root   128 Jun 30 05:20 test
-rw-r--r-- 1 root root     4 Jul  2 11:24 test.txt
-rw-r--r-- 1 root root   446 Jul  2 03:31 test_client.py

```

figure4. SFTP Client ll -l 명령어 실행 화면

```

custom_sftp> ll ../
asn1  asn2  asn3  project  python3-ping
custom_sftp> █

```

figure5. SFTP Client ll ../ 명령어 실행 화면

```

custom_sftp> ll ../ -l
total 0
drwxr-xr-x 10 root root 320 Apr  8 03:38 asn1
drwx----- 6 root root 192 Jun 29 05:36 asn2
drwx----- 6 root root 192 Jun 11 00:12 asn3
drwxr-xr-x 13 root root 416 Jul  2 11:24 project
drwxr-xr-x 6 root root 192 Jun 10 22:01 python3-ping
custom_sftp> █

```

figure6. SFTP Client ll ../ -l 명령어 실행 화면

- `lcd path` : 로컬 디렉토리 위치를 변경합니다.

```
custom_sftp> lpwd  
[Local] /net_workdir/project  
custom_sftp> lcd ..  
custom_sftp> lpwd  
[Local] /net_workdir  
custom_sftp> █
```

figure7. SFTP Client lcd .. 명령어 실행 화면

```
custom_sftp> lpwd  
[Local] /net_workdir  
custom_sftp> lcd net_workdir/test  
custom_sftp> lpwd  
[Local] /net_workdir/net_workdir/test  
custom_sftp> █
```

figure8. SFTP Client lcd net\_workdir/test 명령어 실행 화면

## 4. Remote 명령어

- `pwd` : 현재 원격 디렉토리 위치를 알려줍니다.

```
Hello~!. sftp excute.  
-----Connect Info-----  
Connect User Name | 127.0.0.1  
Connect Host Name | root  
Connect Host Port | 2200  
-----  
[127.0.0.1:2200] Successful connection  
Welcome! sftp connect success~!!  
custom_sftp> pwd  
[Remote] /root  
custom_sftp> █
```

figure9. SFTP Client pwd 명령어 실행 화면

- `ls [ls-options] [path]` : 현재 원격 디렉토리에 위치한 파일 리스트를 보여줍니다.

```
custom_sftp> ls
demo_server.log ftp_server1.py ftp_server2.py ftp_server3.py
ftp_server4.py ftp_server5.py id_rsa id_rsa.pub test.txt
```

figure10. SFTP Client ls 명령어 실행 화면

```
custom_sftp> ls -l
total 708
-rw-r--r-- 1 root root 657508 Jul  3 07:26 demo_server.log
-rw-r--r-- 1 root root   6666 Jul  1 04:30 ftp_server1.py
-rw-r--r-- 1 root root   6956 Jul  3 06:07 ftp_server2.py
-rw-r--r-- 1 root root  14260 Jul  3 07:34 ftp_server3.py
-rw-r--r-- 1 root root   1473 Jul  1 07:42 ftp_server4.py
-rw-r--r-- 1 root root   8319 Jul  3 05:19 ftp_server5.py
-rw----- 1 root root   1823 Jul  1 04:01 id_rsa
-rw-r--r-- 1 root root    399 Jul  2 10:23 id_rsa.pub
-rw-r--r-- 1 root root     10 Jul  3 07:26 test.txt
```

figure11. SFTP Client ls -l 명령어 실행 화면

```
custom_sftp> ls ../
bin boot dev etc home lib lib64 media mnt net_workdir opt
proc root run sbin srv sys tmp usr var wget-log
```

figure12. SFTP Client ls ../ 명령어 실행 화면

```

custom_sftp> ls -l ../
total 84
drwxr-xr-x  1 root root 4096 Apr  7 09:31 bin
drwxr-xr-x  2 root root 4096 Feb  1 17:09 boot
drwxr-xr-x  5 root root  360 Jul  1 04:24 dev
drwxr-xr-x  1 root root 4096 Jul  1 07:32 etc
drwxr-xr-x  1 root root 4096 Jun 30 08:42 home
drwxr-xr-x  1 root root 4096 Apr  7 09:31 lib
drwxr-xr-x  2 root root 4096 Mar 27 00:00 lib64
drwxr-xr-x  2 root root 4096 Mar 27 00:00 media
drwxr-xr-x  2 root root 4096 Mar 27 00:00 mnt
drwxr-xr-x  8 root root  256 Jun 28 06:38 net_workdir
drwxr-xr-x  2 root root 4096 Mar 27 00:00 opt
dr-xr-xr-x 172 root root    0 Jul  1 04:24 proc
drwx-----  1 root root 4096 Jul  3 07:23 root
drwxr-xr-x  1 root root 4096 Jul  3 05:08 run
drwxr-xr-x  1 root root 4096 Apr  7 09:31/sbin
drwxr-xr-x  2 root root 4096 Mar 27 00:00 srv
dr-xr-xr-x 12 root root    0 Jun 30 08:01 sys
drwxrwxrwt  1 root root 4096 Jul  3 06:46 tmp
drwxr-xr-x  1 root root 4096 Mar 27 00:00 usr
drwxr-xr-x  1 root root 4096 Mar 27 00:00 var
-rw-r--r--  1 root root    0 Jul  1 04:00 wget-log

```

figure13. SFTP Client ls -l ../ 명령어 실행 화면

- `cd path` : 원격 디렉토리의 위치를 변경합니다.

```

custom_sftp> pwd
[Remote] /root
custom_sftp> cd ../home
custom_sftp> pwd
[Remote] /home
custom_sftp> █

```

figure13. SFTP Client cd ../home 명령어 실행 화면

## 5. 파일 전송 명령어

- `put [-afPpRr] local [remote]` : 로컬 파일을 원격으로 업로드합니다.

```

Hello~!. sftp excute.
-----Connect Info-----
Connect User Name | 127.0.0.1
Connect Host Name | root
Connect Host Port | 2200
-----
[127.0.0.1:2200] Successful connection
Welcome! sftp connect success~!!
custom_sftp> ls
bin demo_server.log ftp_server.py id_rsa id_rsa.pub
custom_sftp> ll
ftp_client.py ftp_server.py id_rsa id_rsa.pub local.txt test
custom_sftp> put local.txt
local.txt
/net_workdir/project/local.txt
/root/local.txt
[Local] /net_workdir/project/local.txt -> [Remote] ./local.txt
custom_sftp> ls
bin demo_server.log ftp_server.py id_rsa id_rsa.pub local.txt
custom_sftp> ll
ftp_client.py ftp_server.py id_rsa id_rsa.pub local.txt test
custom_sftp> █

```

figure14. SFTP Client put local.txt 명령어 실행 화면

```

custom_sftp> ll
ftp_client.py ftp_server.py id_rsa id_rsa.pub local.txt test
custom_sftp> ls
bin demo_server.log ftp_server.py id_rsa id_rsa.pub local.txt remote_folder
custom_sftp> ls remote_folder

custom_sftp> put local.txt remote_folder
/root/remote_folder/local.txt
[Local] /net_workdir/project/local.txt -> [Remote] ./remote_folder/local.txt
custom_sftp> ls remote_folder
local.txt
custom_sftp> █

```

figure15. SFTP Client put local.txt remote\_folder 명령어 실행 화면

- `get [-afPpRr] remote [local]` : 원격 파일을 로컬로 다운로드합니다.



```

Hello~!. sftp excute.
-----Connect Info-----
Connect User Name | 127.0.0.1
Connect Host Name | root
Connect Host Port | 2200
-----

[127.0.0.1:2200] Successful connection
Welcome! sftp connect success~!!
custom_sftp> ls
bin demo_server.log ftp_server.py id_rsa id_rsa.pub local.txt remote.txt remote_folder
custom_sftp> ll
ftp_client.py ftp_server.py id_rsa id_rsa.pub local.txt test
custom_sftp> get remote.txt
[Remote] ./remote.txt -> [Local] /net_workdir/project/remote.txt
custom_sftp> ll
ftp_client.py ftp_server.py id_rsa id_rsa.pub local.txt remote.txt test
custom_sftp> █

```

figure16. SFTP Client get remote.txt 명령어 실행 화면

```

Welcome! sftp connect success~!!
custom_sftp> ls
bin demo_server.log ftp_server.py id_rsa id_rsa.pub local.txt remote.txt remote_folder
custom_sftp> ll
ftp_client.py ftp_server.py id_rsa id_rsa.pub local.txt local_folder remote.txt test
custom_sftp> ll local_folder
custom_sftp> get remote.txt local_folder
[Remote] ./remote.txt -> [Local] /net_workdir/project/local_folder/remote.txt
custom_sftp> ll local_folder
remote.txt
custom_sftp> █

```

figure17. SFTP Client get remote.txt local\_folder 명령어 실행 화면

- `exit` : sftp 접속을 종료합니다.

```

Hello~!. sftp excute.
-----Connect Info-----
Connect User Name | 127.0.0.1
Connect Host Name | root
Connect Host Port | 2200
-----

[127.0.0.1:2200] Successful connection
Welcome! sftp connect success~!!
custom_sftp> exit
Quit sftp
root@b6cd8ba91630:/net_workdir/project# █

```

figure18. SFTP Client exit 명령어 실행 화면

```
Listening for connection ...
(Check the new thread) 127.0.0.1:41420
Got a connection!
Auth attempt with key: c47cbde2915e911580543e1c4b209fb7
Authenticated!
[127.0.0.1:41420] mssg_00_connect
[127.0.0.1:41420] mssg_01_init_pwd
[127.0.0.1:41420]
[127.0.0.1:41420] Socket is closed
```

figure19. SFTP Server 접속이 끊어진 로그 화면

## 6. 클라이언트 다중 접속

최대 200개의 다중 접속을 허용하도록 설정했습니다.

<pre>Hello~!. sftp excute. -----Connect Info----- Connect User Name   127.0.0.1 Connect Host Name   root Connect Host Port   2200  [127.0.0.1:2200] Successful connection Welcome! sftp connect success~!! custom_sftp&gt; ls bin demo_server.log ftp_server.py id_rsa id_rsa.p ub local.txt remote.txt remote_folder custom_sftp&gt; cd ../home custom_sftp&gt; pwd [Remote] /home custom_sftp&gt; █</pre>	<pre>Hello~!. sftp excute. -----Connect Info----- Connect User Name   127.0.0.1 Connect Host Name   root Connect Host Port   2200  [127.0.0.1:2200] Successful connection Welcome! sftp connect success~!! custom_sftp&gt; ls bin demo_server.log ftp_server.py id_rsa id_rsa.pub local.txt remote.txt remote_folder custom_sftp&gt; pwd [Remote] /root custom_sftp&gt; █</pre>	<pre>Hello~!. sftp excute. -----Connect Info----- Connect User Name   127.0.0.1 Connect Host Name   root Connect Host Port   2200  [127.0.0.1:2200] Successful connection Welcome! sftp connect success~!! custom_sftp&gt; ls bin demo_server.log ftp_server.py id_rsa id_rsa.p ub local.txt remote.txt remote_folder custom_sftp&gt; cd .. custom_sftp&gt; ls bin boot dev etc home lib lib64 media mnt net_wor kdir opt proc root run sbin srv sys tmp usr var w get-log custom_sftp&gt; pwd [Remote] / custom_sftp&gt; █</pre>
---	---	---

figure18.SFTP Client 다중 접속 화면

```

[127.0.0.1:41408] socket is closed
(Check the new thread) 127.0.0.1:41408
Got a connection!
INFO:paramiko.transport:Connected (version 2.0, client paramiko_2.7.1)
Auth attempt with key: c47cbde2915e911580543e1c4b209fb7
INFO:paramiko.transport:Auth granted (publickey).
Authenticated!
[127.0.0.1:41408] mssg_00_connect
[127.0.0.1:41408] mssg_01_init_pwd
(Check the new thread) 127.0.0.1:41410
Got a connection!
INFO:paramiko.transport:Connected (version 2.0, client paramiko_2.7.1)
Auth attempt with key: c47cbde2915e911580543e1c4b209fb7
INFO:paramiko.transport:Auth granted (publickey).
Authenticated!
[127.0.0.1:41410] mssg_00_connect
[127.0.0.1:41410] mssg_01_init_pwd
(Check the new thread) 127.0.0.1:41412
Got a connection!
INFO:paramiko.transport:Connected (version 2.0, client paramiko_2.7.1)
Auth attempt with key: c47cbde2915e911580543e1c4b209fb7
INFO:paramiko.transport:Auth granted (publickey).
Authenticated!
[127.0.0.1:41412] mssg_00_connect
[127.0.0.1:41412] mssg_01_init_pwd
[127.0.0.1:41408] mssg_02_ls /root
[127.0.0.1:41410] mssg_02_ls /root
[127.0.0.1:41412] mssg_02_ls /root
[127.0.0.1:41412] mssg_03_cd /root/..
[127.0.0.1:41412] mssg_02_ls /
[127.0.0.1:41408] mssg_03_cd /root/../home

```

figure19.SFTP Server 다중 접속 로그 화면 (로컬에서 실험 진행, 클라이언트 별로 포트가 다름)

## Discussion & Conclusion

이번 프로젝트를 통해 SSH와 SFTP의 기본 원리를 이해할 수 있었습니다. 특히나 SSH 연결은 기존의 소켓 통신보다 보안을 강화한 프로토콜로 과제로 구현해야 했던 소켓 프로그램보다 구현이 조금 더 까다로웠습니다. 또 서버는 다중 접속을 허용해야 하기 때문에 멀티 스레드를 이용하였습니다. SSH 통신을 통해 원격 디렉토리를 검색하고 이를 로컬로 통신을 하였습니다. 파일 전송을 시도하면 해당 클라이언트 스레드에서 SFTP를 위한 소켓을 서버와 새로 연결하고 해당 연결을 통해 파일 전송을 진행하며 파일 전송을 완료하면 SFTP 소켓은 종료되고 다시 SSH 연결을 위한 소켓을 열어 통신을 합니다. 이를 통해 지속적인 연결을 유지하여 파일 전송과 원격 디렉토리 검색을 이후에도 시도할 수 있습니다.

이번 프로젝트를 통해 SFTP 서버와 클라이언트를 구현 하면서 학기 중 수업을 통해 배웠던 지식을 정리 할 수 있었습니다.