

T. CALONNE
T. JUILLARD

3 mai 2020

Manuel utilisateur

Cette documentation est à destination des utilisateurs de l'application ChatHack. Il sera décrit l'ensemble des actions à réaliser en environnement de production afin d'utiliser l'application de façon fonctionnelle. De plus, il sera mentionné les erreurs connues qu'un utilisateur puisse rencontrer.

SOMMAIRE

1.Lancement de l'application	3
Lancement du serveur de mot de passe	3
Lancement du serveur ServerChatHack	4
2.Connexion des clients	4
Lancement des clients ClientChatHack	4
Connexion sans mot de passe	4
Connexion avec mot de passe	5
3.Communication en mode public	5
4.Communication en mode privé	6
Initialisation d'une conversation privée avec un utilisateur	6
a. La réponse est "yes"	6
b. La réponse est "no"	7
5.Echange de fichiers	7
6.Erreurs connues	8
Erreurs prévues au développement de la version 1.0	8
Connexion	8
Envoi de messages privées	8
Echange de fichier	9
Environnement windows	9
Environnement linux	9

1.Lancement de l'application

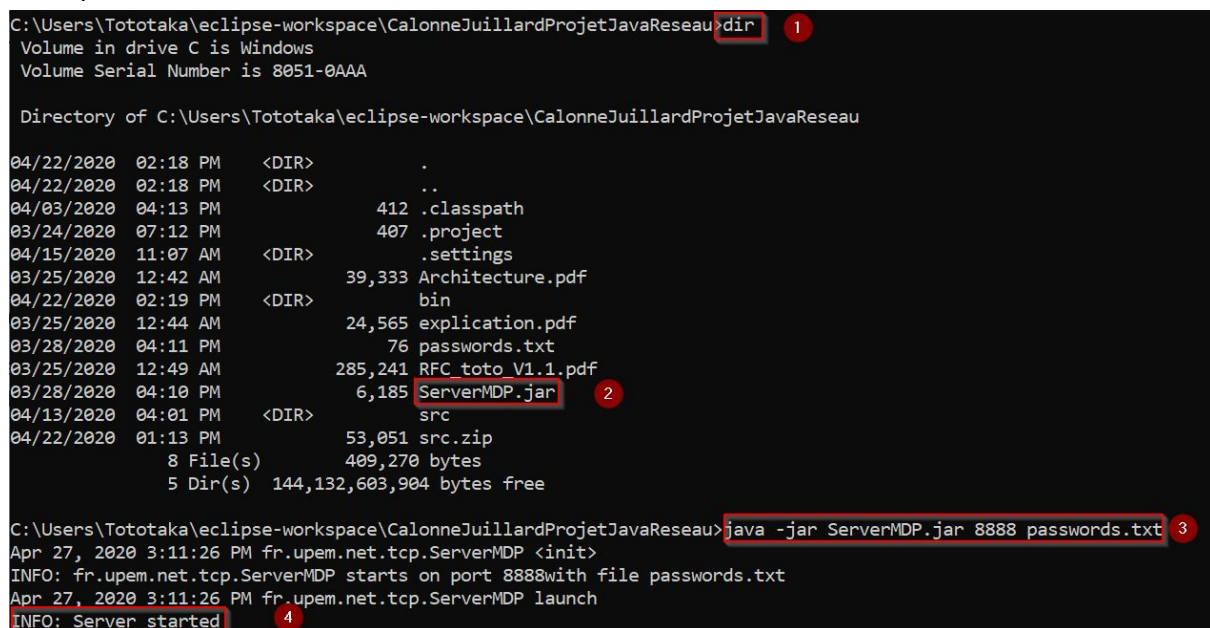
Lancement du serveur de mot de passe

Pour lancer le serveur de mot de passe il faut se mettre dans le répertoire du projet, à l'endroit où se trouve le fichier ServerMDP.jar et ouvrir un terminal.

Dans ce terminal, tapez la commande suivante :

java -jar ServerMDP.jar 8888 passwords.txt

Exemple d'utilisation :



The screenshot shows a Windows command prompt window. The title bar indicates the path: C:\Users\Tototaka\eclipse-workspace\CalonneJuillardProjetJavaReseau. The prompt is at the same path. A red box with a '1' highlights the 'dir' command. The output shows a directory listing of the current folder, including files like .classpath, .project, .settings, Architecture.pdf, bin, explication.pdf, passwords.txt, RFC toto V1.1.pdf, and ServerMDP.jar. A red box with a '2' highlights 'ServerMDP.jar'. Below the listing, the command 'java -jar ServerMDP.jar 8888 passwords.txt' is entered, highlighted with a red box and a '3'. The output shows the server starting on port 8888. A red box with a '4' highlights the final output line: 'INFO: Server started'.

```
C:\Users\Tototaka\eclipse-workspace\CalonneJuillardProjetJavaReseau>dir 1
Volume in drive C is Windows
Volume Serial Number is 8051-0AAA

Directory of C:\Users\Tototaka\eclipse-workspace\CalonneJuillardProjetJavaReseau

04/22/2020  02:18 PM    <DIR>          .
04/22/2020  02:18 PM    <DIR>          ..
04/03/2020  04:13 PM             412 .classpath
03/24/2020  07:12 PM             407 .project
04/15/2020  11:07 AM    <DIR>          .settings
03/25/2020  12:42 AM      39,333 Architecture.pdf
04/22/2020  02:19 PM    <DIR>          bin
03/25/2020  12:44 AM      24,565 explication.pdf
03/28/2020  04:11 PM              76 passwords.txt
03/25/2020  12:49 AM     285,241 RFC toto V1.1.pdf
03/28/2020  04:10 PM      6,185 ServerMDP.jar 2
04/13/2020  04:01 PM    <DIR>          src
04/22/2020  01:13 PM     53,051 src.zip
               8 File(s)      409,270 bytes
               5 Dir(s)  144,132,603,904 bytes free

C:\Users\Tototaka\eclipse-workspace\CalonneJuillardProjetJavaReseau>java -jar ServerMDP.jar 8888 passwords.txt 3
Apr 27, 2020 3:11:26 PM fr.upem.net.tcp.ServerMDP <init>
INFO: fr.upem.net.tcp.ServerMDP starts on port 8888with file passwords.txt
Apr 27, 2020 3:11:26 PM fr.upem.net.tcp.ServerMDP launch
INFO: Server started 4
```

Avec :

- 1: la commande “**dir**” (sous windows) et “**ls**” (sous linux) permet de lister le contenu du répertoire courant
- 2: On a bien le nom du fichier présent dans le répertoire courant
- 3: La commande a taper pour lancer le serveur de mots de passe (ici le numéro de port 8888 peut être différent)
- 4: On voit bien que le serveur a démarré

Lancement du serveur ServerChatHack

Pré-requis : Le serveur de mot de passe “**ServerMDP**” doit être démarré.

Pour lancer le serveur ServerChatHack il faut se placer dans le répertoire où se situe le .jar et ouvrir un terminal.

Dans ce terminal, tapez la commande suivante :

java -jar ServerChatHack.jar 7777 8888

```
thomas@Homard:~/workspace/CalonneJuillardProjetJavaReseau$ ls 1
Architecture.pdf  capture      capture.zip  cvec.pdf  explication.pdf  RFC_toto_V1.1.pdf  ServerMDP.jar  srcOPTIMISE.zip
bin              Capture.PNG  ClientChatHack.jar  doc       passwords.txt    ServerChatHack.jar 2 rc          srcSAVE.zip
thomas@Homard:~/workspace/CalonneJuillardProjetJavaReseau$ java -jar ServerChatHack.jar 7777 8888 3
Le serveur est opérationnel. 4
```

Avec :

- 1: la commande “**dir**” (sous windows) et “**ls**” (sous linux) permet de lister le contenu du répertoire courant
- 2: On a bien le nom du fichier présent dans le répertoire courant
- 3: La commande à taper pour lancer le serveur ServerChatHack (ici le numéro du port 7777 peut être différent, mais le numéro du port 8888 doit être le même que celui utilisé au moment de lancer le serveur de mots de passe).
- 4: On voit bien que le serveur a démarré

2.Connexion des clients

Lancement des clients ClientChatHack

Pré-requis : Le serveur de mot de passe “**ServerMDP**” et le serveur “**ServerChatHack**” doivent être démarrés.

On peut se connecter avec ou sans mot de passe au client. Vous trouverez comment vous connecter dans cette section.

Connexion sans mot de passe

Dans un terminal ouvert à l’emplacement du fichier ClientChatHack, tapez la commande suivante :

java -jar ClientChatHack.jar localhost 7777 ../ client1

```
thomas@Homard:~/workspace/CalonneJuillardProjetJavaReseau$ ls 1
Architecture.pdf ClientChatHack.jar 2 passwords.txt ServerMDP.jar srcSAVE.zip
bin doc RFC_toto_V1.1.pdf src
capture explication.pdf ServerChatHack.jar srcOPTIMISE.zip
thomas@Homard:~/workspace/CalonneJuillardProjetJavaReseau$ java -jar ClientChatHack.jar localhost 7777
7 ../ client1 3
Vous etes bien connecté 4
```

Avec :

- 1: la commande “**dir**” (sous windows) et “**ls**” (sous linux) permet de lister le contenu du répertoire courant
- 2: On a bien le nom du fichier présent dans le répertoire courant
- 3: La commande à taper pour lancer le client ClientChatHack (ici le numéro du port 7777 doit être le même que celui mis pour le serveur ServerChatHack, “localhost” peut être remplacé par l’adresse IP (ex: 192.168.x.x) du client. Et le nom “client1” est le nom d’utilisateur que vous allez avoir.

```
thomas@Homard:~/workspace/CalonneJuillardProjetJavaReseau$ java -jar ClientChatHack.jar 192.168.1.38
7777 ../ client2
Vous etes bien connecté
```

- 4: On voit bien que le client est démarré

Connexion avec mot de passe

L’authentification se fait via ServerMDP.jar qui va récupérer des informations dans une base de données (ici sous format texte) du nom de “**passwords.txt**”. Pour se connecter avec un mot de passe au ClientChatHack nous devons donc prendre les bons identifiants et mots de passe se trouvant dans ce fichier texte.

Dans un terminal ouvert à l’emplacement du fichier ClientChatHack, tapez la commande suivante :

java -jar ClientChatHack.jar localhost 7777 ../ yann sunday

```
thomas@Homard:~/workspace/CalonneJuillardProjetJavaReseau$ java -jar ClientChatHack.jar localhost 7777 ../ yann 1
Vous n'êtes pas connecté
thomas@Homard:~/workspace/CalonneJuillardProjetJavaReseau$ java -jar ClientChatHack.jar localhost 7777 ../ yann sunday 2
Vous etes bien connecté 3
```

Avec :

- 1: On essaye de se connecter avec un profile qui est présent dans password.txt mais sans mot de passe. La connexion ne fonctionne pas.
- 2: On essaye de se connecter avec le profile de “yann” et son mot de passe.
- 3: La connexion est établie correctement.

3.Communication en mode public

Pour pouvoir communiquer avec l’ensemble des clients connecté, vous allez pouvoir écrire un message et appuyer sur la touche “Entrée” pour envoyer ce message.

Exemple: La personne qui s’appelle “**client1**” souhaite dire bonjour aux personnes de connectées. Il va écrire “Bonjour !” et appuyer sur “Entrée”

```
Bonjour !
client1: Bonjour !
```

Le nom de client1 s'affichera avec à la suite le message qu'il a envoyé. L'ensemble des utilisateurs connectés pourront voir ce message ainsi que la personne qui l'a envoyé.

4.Communication en mode privé

Il est possible de parler avec un utilisateur de façon privée, cette section permet d'expliquer le procédé.

Initialisation d'une conversation privée avec un utilisateur

Il existe plusieurs façons de demander une connexion privée avec un autre utilisateur. Vous pouvez faire **@nom_de_la_personne** ou **/nom_de_la_personne** puis appuyer sur "entrée".

```
/client2 1  
client2 n'a pas accepté ton invitation ! Tes messages en attente pour lui vont être supprimés.  
@client2 2  
client2 a accepté votre demande de connexion
```

Le client2 peut alors accepter ou refuser la demande de connexion privée.

Il est également possible de faire sa demande et d'envoyer en même temps un message.

Exemple : La personne "client1" envoie un message privée à la personne "client2" avec "Salut Client2 " et appuis sur "entrée"

```
thomas@Homard:~/workspace/CalonneJuillardProjetJavaReseau$ java -jar ClientChatHack.jar localhost 777  
7 ./ client1  
Vous etes bien connecté  
@client2 Salut client2  
Une demande d'échanges privés a été faite à : client2.
```

La personne "client2" va alors recevoir un message pour savoir s'il souhaite communiquer ou non avec la personne "client1"

```
Voulez-vous accepter la demande de connexion privée de: client1 ? ['yes'/'no']
```

Deux réponses pour le client2 seront alors possibles :

a. La réponse est "yes"

Dans le cas où la personne "client2" répond "yes". La **connexion** sera alors **établie** et la personne "client2" recevra le message que la personne "client1" aura envoyé.

```
thomas@Homard:~/workspace/CalonneJuillardProjetJavaReseau$ java -jar ClientChatHack.jar 192.168.1.38  
7777 ./ client2  
Vous etes bien connecté  
Voulez-vous accepter la demande de connexion privée de: client1 ? ['yes'/'no']  
yes  
[client1: Salut client2]
```

La personne "client2" pourra alors répondre à client1 en mettant **@client1** et le contenu de son message.

```
Voulez-vous accepter la demande de connexion privée de: client1 ? ['yes'/'no']
yes
[client1: Salut client2]
@client1 Hello
```

Et “**client1**” recevra la réponse directement car la connexion privée sera déjà établie.

b. La réponse est “no”

Dans le cas où Si la personne “**client2**” répond “**no**” donc n'a pas voulu communiquer avec “**client1**”, il ne recevra jamais le message.

```
Voulez-vous accepter la demande de connexion privée de: client1 ? ['yes'/'no']
no
```

Et la personne qui a demandé une connexion sera alors notifié que son message ne sera pas lu.

```
client2 n'a pas accepté ton invitation ! Tes messages en attente pour lui vont être supprimés.
```

Une demande de connexion privée pourra être de nouveau demandé même en cas de refus.

5.Echange de fichiers

Il est également possible d'échanger des fichiers avec une autre personne. Pour cela il faut écrire dans l'invite de commande **/nom_du_destinataire** puis le fichier que vous souhaitez envoyer et enfin appuyer sur “**entrée**”.

Exemple : La personne “**client1**” envoie le fichier “**passwords.txt**” à la personne “**client2**”. Voici la ligne de commande que devra entrer le “**client1**”:

```
7777 ../ client1
Vous etes bien connecté
/client2 passwords.txt
../passwords.txt
```

Si il y a eu déjà une demande de connexion privée et que le client2 avait répondu favorablement alors le fichier sera envoyé.

Sinon une nouvelle demande de connexion privée sera renvoyé. Le destinataire pourra alors accepter ou refuser cette demande. Dans le cas où il accepte il recevra le fichier sinon le fichier ne sera pas envoyé.

6. Erreurs connues

Erreurs prévues au développement de la version 1.0

Connexion

Impossible de se connecter avec un mot de passe : Il faut vérifier qu'on ne se trompe pas dans le nom et le mot de passe à la saisie.

```
C:\Users\Tototaka\eclipse-workspace\CalonneJuillardProjetJavaReseau\bin>java fr.upem.projet.ClientChatHack localhost
7777 ../ yann toto
Vous n'êtes pas connecté
```

Mauvais MDP

```
C:\Users\Tototaka\eclipse-workspace\CalonneJuillardProjetJavaReseau\bin>java fr.upem.projet.ClientChatHack localhost
7777 ../ yann sunday
Vous êtes bien connecté
```

MDP OK

Il est obligatoire de préciser l'adresse Ip du client si deux machines veulent communiquer depuis deux supports physiques différents.

Envoi de messages privées

Au moment où l'on souhaite écrire un message à une personne en privée mais que ça dit erreur le nom n'est pas correct, il faut vérifier que le nom saisi correspond bien à un utilisateur connecté.

```
@client3 Hello
Aucune personne connectée ne possède ce nom: client3
```

Au moment où l'on souhaite écrire un message à une personne en privée mais que ça dit ne peux pas communiquer avec vous même, il ne faut pas utiliser son identifiant pour demander une connexion privée.

```
thomas@Homard:~/workspace/CalonneJuillardProjetJavaReseau$ java -jar ClientChatHack.jar localhost 777
7 ../ client2
Vous êtes bien connecté
/client2
Vous ne pouvez pas communiquer avec vous-même.
```

La personne qui a initié la demande de connexion privée n'est plus connectée :

```
thomas@Homard:~/workspace/CalonneJuillardProjetJavaReseau$ java -jar ClientChatHack.jar 192.168.1.38
7777 ../ client2
Vous êtes bien connecté
Voulez-vous accepter la demande de connexion privée de: client1 ? ['yes'/'no']
yes
client1 n'est plus connecté. Vous ne pouvez donc plus converser avec lui.
```


Echange de fichiers

Le destinataire du fichier n'est pas connecté:

```
thomas@Homard:~/workspace/CalonneJuillardProjetJavaReseau$ java -jar ClientChatHack.jar localhost 7777 ./ client1
Vous etes bien connecté
/client2 explication.pdf
Envoi du fichier 'explication.pdf' en cours.
Une demande d'échanges privés a été faite à : client2.
Aucune personne connectée ne possède ce nom : client2
```

Le nom du fichier que vous avez voulu envoyé n'est pas correct:

```
/client1 fichierQuiNexistePas
Il y a eu un problème avec l'envoi de votre fichier. Vérifiez qu'il se trouve bien dans pathFile que vous avez donné en argument.
```

Environnement windows

Si client1 communique avec client2, et que client1 décide de fermer sa connexion, alors client2 sera lui aussi fermé.

On arrive pas à recevoir de pdf sur windows. Erreur sur l'encodage. De Windows vers linux ça fonctionne mais pas l'inverse. L'envoi fonctionne bien.

Environnement linux

Il n'y a pas d'erreurs connues pour l'instant. A compléter lors de la découverte d'une erreur dans un environnement linux.