به نام خدا

1 - Replay attack:

در این نوع حمله فرد مهاجم بین سرور و کلاینت قرار گرفته و باعث ایجاد اختلال در عملکرد سیستم و اطلاعات کاربران می شود. برای محدود کردن هکر اقداماتی نظیر محدود ساختن تعداد در خواست ها و استفاده از توکن برای ارتباط بین سرور و کلاینت انجام داده ایم.

```
    } catch (SocketException e) {
        System.err.println(e.getMessage());
        clientHandlers.remove( o: this);
    } catch (IOException e) {
        System.err.println(e.getMessage());
    }

}
private void timerForBroken() {
    Timer timer = new Timer();
    timer.schedule(new TimerTask() {

        @Override
        public void run() {
            counter=0;
        }
    }, delay: 0, period: 10000);


}
```

```
        @Override
        public void run() {

            try {
                String input;
                timerForBroken();
                while (true) {
                    input = dataInputStream.readUTF();
                    if (player != null)
                        System.out.println("Client.Client with username " + player.getLastName() + " sent : " +
                    else System.out.println("A client sent : " + input);
                    String answer = answerClient(input);
                    dataOutputStream.writeUTF(answer);
                    dataOutputStream.flush();
                    System.out.println("server answered : " + answer);
                    System.err.println(OnlineUsers.getOnlineUsers());
                    if (input.equals("end")) {
                        System.out.println("Connection closed!!!");
                        break;
                    }
                    if (counter==4){
                       clientSocket.close();
                       break;
                    }
                    counter++;
                }

            } catch (SocketException e) {
                System.err.println(e.getMessage());
                clientHandlers.remove( o: this);
            } catch (IOException e) {
                System.err.println(e.getMessage());
```

همان طور که در دو تصویر بالا مشخص است تعداد درخواست ها را به چهار عدد در هر ده ثانیه کاهش داده ایم.

```
[]
A client sent : login eyJhbGciOiJIUzI1NiJ9.eyJqdGkiOiJQbGF5ZXIxMjMiLCJzdWIiOiJwbGF5ZXIyIiwiaXNzIjoiYXRhIiwiaWF0IjoxNjEyMzgyNTU3LCJleHAiOjE2MTIzODMxNTd9.FWNFTmF0tFreXBrVVlKU
server answered : Success Player login
```

```java
public String generateToken(String id , String subject , String key){
    return Jwts.builder()
            .setId(id)
            .setSubject(subject)
            .setIssuer("ata")
            .setIssuedAt(new Date(System.currentTimeMillis()))
            .setExpiration(new Date(System.currentTimeMillis() + TimeUnit.MINUTES.toMillis( duration: 10)))
            .signWith(SignatureAlgorithm.HS256, Base64.getEncoder().encode(key.getBytes()))
            .compact();
}

public Claims getClaims (String key , String token){
    return Jwts.parser()
            .setSigningKey(Base64.getEncoder().encode(key.getBytes()))
            .parseClaimsJws(token)
            .getBody();
}
```

در دو تصویر بالا منیز واضح است که رابط بین سرور و کلاینت توکن های یونیکی می باشد که عمر کوتاهی نیز دارند و به صورت همیشگی استفاده کرد.

2 - Improper inputs :

برای جلوگیری از ورودی های نا مربوط در چند مرحله اقداماتی انجام میدهیم. در وهله ی اول هر ورودی نظیر شماره یا ایمیل یا رمز با رجکس بسیار قوی مطابقت داده می شود که در صورت مغایرت پیام خطا ی مناسب نمایش داده می شود.

```java
public class Validation {
    public static boolean emailIsValid(String email) throws InvalidEmailException {
        Pattern emailPattern = Pattern.compile("(^([a-zA-Z0-9._%-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,4})*$)");
        Matcher matcher = emailPattern.matcher(email);
        if (matcher.matches())
            return true;
        else
            throw new InvalidEmailException("Email is invalid");
    }
    public static boolean phoneNumberIsValid(String phoneNumber) throws InvalidPhoneNumberException {
        Pattern phonePattern = Pattern.compile("^(0)?9\\d{9}$");
        Matcher matcher = phonePattern.matcher(phoneNumber);
        if (matcher.matches())
            return true;
        else
            throw new InvalidPhoneNumberException("Phone number is invalid");
```

در وهله ی دوم یک سری توابع جهت تشخیص ورودی با فرمت مناسب (string یا int) قرار داده شده است.

```java
public boolean isInteger(String input){
    boolean isInteger = false ;
    try{
        int validInt = Integer.parseInt(input);
        isInteger = true;
    }catch (NumberFormatException numberFormatException){
        System.out.println("This is not a valid number! ");
    }
    return isInteger;
}

public boolean isString(String input){
    boolean isString = false ;
    try{
        String validString = String.valueOf(input);
        isString = true;
    }catch (Exception numberFormatException){
        System.out.println("This is not a String! ");
    }
    return isString;
}
```

: Brute fruce - 3

جهت مقابله با این اقدام اگر کاربر مدت زمانی در هر صفحه ای از پلاتو ثابت بماند و حرکتی نداشته باشد به صورت خودکار از برنامه خارج می شود و به صفحه ورود هدایت شده و مجبور به وارد کردن یوزرنیم و پسورد برای بار دیگر خواهد بود.

```java
private void broken() {
    Stage stage = new Stage();
    Object root = null;
    try {
        URL url = new File( pathname: "src/main/resources/FXML/Login.fxml").toURI().toURL();
        root = FXMLLoader.load(url);
        stage.setScene(new Scene((Parent) root));
    } catch (IOException e) {
        e.printStackTrace();
    }
    Stage window = (Stage) BtnLogOut.getScene().getWindow();
    window.setScene(stage.getScene());
    window.show();
}
@FXML
private void timerForBroken() {
    Timer timer = new Timer();
    timer.schedule(new TimerTask() {

        @Override
        public void run() {
            try {
                dataLoader.makePlayerOffline(LoginController.getUsername());
            } catch (IOException e) {
                e.printStackTrace();
            }
            LoginController.setUsername("null");
            LoginController.setPlayer(null);
            //broken();
            Platform.runLater(() -> broken());
        }
    }, delay: 30000, period: 30000);
```

4 - DOS :

برای حفظ امنیت در این مورد با توجه به اینکه برنامه در بستر اپلیکیشن قرار دارد به اقدام مشابه مورد اول یعنی محدود کردن تعداد در خواست های تبادل شده ی بین سرور و کلاینت بسنده کردیم. به این صورت که اگر تعداد درخواست ها بالا باشد برای مدتی ارتباط سرور و کلاینت قطع خواهد شد.

```
@Override
public void run() {

    try {
        String input;
        timerForBroken();
        while (true) {
            input = dataInputStream.readUTF();
            if (player != null)
                System.out.println("Client.Client with username " + player.getLastName() + " sent : " +
            else System.out.println("A client sent : " + input);
            String answer = answerClient(input);
            dataOutputStream.writeUTF(answer);
            dataOutputStream.flush();
            System.out.println("server answered : " + answer);
            System.err.println(OnlineUsers.getOnlineUsers());
            if (input.equals("end")) {
                System.out.println("Connection closed!!!");
                break;
            }
            if (counter==4){
               clientSocket.close();
               break;
            }
            counter++;
        }

    } catch (SocketException e) {
        System.err.println(e.getMessage());
        clientHandlers.remove( o: this);
    } catch (IOException e) {
```

5 - Broken authentication :

این مورد را نیز در سه مرحله بررسی می کنیم. وهله ی اول رجکس قوی را قرار دادیم که تعداد کاراکتر های رمز عبور بالا بوده و شامل عدد و حرف بزرگ و کوچک باشد.

```
public static boolean passwordIsValid(String password) throws StrongerPasswordException {
    Pattern passwordPattern = Pattern.compile("(^(?=^.{6,}$)((?=.*[A-Za-z0-9])(?=.*[A-Z])(?=.*[a-z]))^.*$)");
    Matcher matcher = passwordPattern.matcher(password);
    if (matcher.matches())
        return true;
    else
        throw new StrongerPasswordException ("Please try stronger password !");
}
```

در مرحله دوم تدبیری اندیشیده شده است که اگر کاربر تعداد معینی رمز خود را اشتباه وارد کند به صورت خودکار توسط ادمین بن شده و می بایست احراز هویت کند تا بتواند دوباره به محیط پلاتو باز گردد.

```java
public static boolean checkPassword(String username, String password) throws IOException, AlreadyBan {
    boolean result = false;
    Player userByUsername = null;
    for (Player user : Player.players) {
        if (user.getUserName().equals(username)) {
            userByUsername = user;
            break;
        }
    }
    if (userByUsername != null) {
        if (userByUsername.getPassword().equals(password)) {
            result = true;
        }
        if (!userByUsername.getPassword().equals(password)){
            new AdminGeneralController().banPlayer(userByUsername.getUserName());
        }
    }

    return result;
}
```

مرحله ی سوم همان بیرون انداختن کاربری که هیچ فعالیتی ندارد می باشد که در بالا نیز به ان اشاره شد.