به نام او...

مستند امنیت در پروژه AP

تیم ۴۵ : امیرحسین باقری – امیرحسین ندایی پور – سید محمد مهدی حاتمی

- Replay attack :
  - سناریو : مهاجم با دریافت پیام های کلاینت و تکرار آن برای سرور سعی میکند در روند فعالیت سرور اخلال ایجاد کند.
  - راه حل : در سمت کلاینت و سرور شمارنده ای قرار می دهیم که هنگام اتصال اولیه با یکدیگر برابر می شوند و با هر پیام ارسالی از کلاینت به سرور ، هر دو شمارنده خود را زیاد می کنند. با این کار سرور در هر پیام انتظار دارد مقدار شمارنده مقداری متفاوت از پیام قبلی داشته باشد و با تکرار یک پیام، متوجه جعلی بودن آن می شود.

حال کافیست شمارنده از دست مهاجم پنهان شود تا نتواند با تغییر آن مجددا سرور را فریب بدهد. برای این کار دو روش به نظر می رسد :

۱. الگوی افزایش شمارنده پیچیده باشد و مهاجم توانایی حدس زدن دنباله آن را نداشته باشد. در این صورت مهاجم نمی تواند شمارنده ی معتبر برای پیام بعدی را پیدا کند و در نتیجه ناکام می شود. به عنوان دنباله می توان تصاعدی حسابی در نظر گرفت که قدر نسبت و جمله ی اول آن در اولین اتصال و در یک پیام رمز شده بین کلاینت و سرور رد و بدل شود.

۲. تمام پیام ، یا تنها قسمت شمارنده به گونه ای رمز شود که مهاجم توانایی دسترسی به مقدار فعلی و در نتیجه یافتن مقدار بعدی شمارنده معتبر را نداشته باشد.

  - نحوه استفاده :

در پروژه ما شمارنده روی کلاینت و سرور به صورت عددی که از صفر شروع شده و یکی یکی زیاد میشود قرار داده شده است. به عنوان message counter و بر روی کلاس های clientSocket و clientHandler و در ابتدای پیام های ارسالی از کلاینت به سرور درج می شود.

در ادامه می توان با رمز کردن کل پیام با رمز نگاری متقارن یا نامتقارن ، از دستیابی مهاجم به شمارنده جلوگیری کرد.

این رمز نگاری در قسمت فرستادن فایل پیاده سازی شده است ولی متاسفانه در پیام های معمولی فرصت نشد که پیاده سازی بشود.

```
29      private int messageCounter;
30
31      private ClientSocket() {...}
35
36      public static ClientSocket getInstance() { return instance; }
39
40      public String sendAction(String action) {
41          String Json = null;
42          try {
43              // action -> Server
44              String messageToServer = getOutPutReady(action);
45              outputStream.writeUTF(messageToServer);
46              outputStream.flush();
47              messageCounter++;
48
49              // Server -> Json
50              synchronized (inputLock) {
51                  inputLock.wait();
52                  Json = getInputReady(LastMessage);
53              }
54              // check if json is an exception
55              // TODO
56          } catch (IOException | InterruptedException exception) {
57              exception.printStackTrace();
58          }
59          return Json;
60      }
```

```
66      private String getOutPutReady(String action) {
67          return messageCounter + "T" + token + "T" + action;
68      }
```

تعریف و استفاده از شمارنده پیام در کلاس ClientSocket

- Improper input :
  - سناریو : مهاجم سعی می کند با فرستادن ورودی های نامعتبر برای سرور در روند کار آن اخلال ایجاد کند. این ورودی نامعتبر می تواند یکی از حالات زیر باشد :

۱. پیام فرستاده شده به طور کلی برای سرور نامعتبر باشد. یعنی پیام ارسالی کاملا برای سرور بی معنی باشد.

راه حل : سرور برای پیام های خودش فرمت خاصی در نظر بگیرد و در صورت صدق نکردن پیام ارسالی در فرمت معتبر، آن را نادیده بگیرد.

```
private String getInputReady(String clientMessage) throws WrongTokenException {
    Matcher tokenMatcher = Pattern.compile("^(\\d+)T(\\d+)T").matcher(clientMessage);
    if (!tokenMatcher.find()) {
        throw new WrongTokenException();
    }
    int receivedToken = Integer.parseInt(tokenMatcher.group(2));
    int receivedMC = Integer.parseInt(tokenMatcher.group(1));
    if (this.token != receivedToken || receivedMC != messageCounter) {
        throw new WrongTokenException();
    }
    messageCounter++;
    return clientMessage.substring(tokenMatcher.end());
}
```

بررسی فرمت پیام ارسالی توسط کلاینت در کلاس ClientHandler در سمت سرور

۲. کلاینت با دادن ورودی های نامعتبر به فیلد های برنامه، سعی در ثبت اطلاعات نامعتبر در سرور یا ایجاد کرش در روند کاری آن می کند.

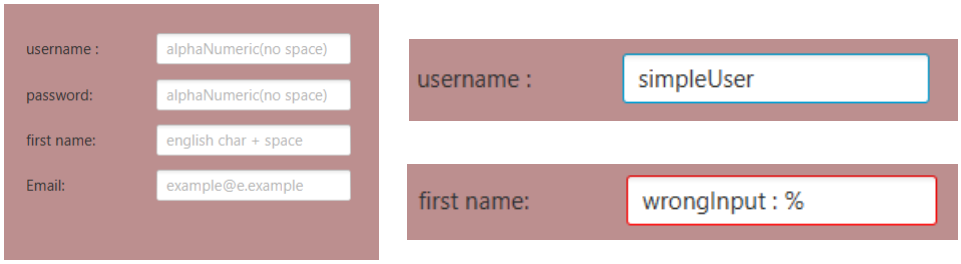
راه حل : در ابتدا لازم است کلاینت داده های ورودی خود را اعتبار سنجی کند و در صورت نامعتبر بودن از ارسال آن ها جلوگیری کند. این راهبرد در پروژه ما استفاده شده است و کلاس ValidatorField تمام عملیات های اعتبار سنجی را در لحظه ی ورودی گرفتن از کاربر بررسی می کند.

اما در ادامه ممکن است مهاجم با دسترسی به کد کلاینت از این محدودیت آن گذر کند و اطلاعات نامعتبر برای سرور ارسال شود. در این جا لازم است سرور قبل از استفاده از اطلاعات دریافتی از کلاینت، معتبر بودن آن ها را بررسی کند. متاسفانه این راهبرد در پروژه ما استفاده نشده است.

```
@FXML
public boolean validate() {
    if (getText().equals("")) {
        if (isNecessary.get()) {
            this.setStyle("-fx-text-box-border: #fc0606;");
            return false;
        }
        else {
            this.setStyle("-fx-text-box-border: #039ed3;");
            return true;
        }
    }
    if (validatorRegex == null) {
        setValidatorRegex();
    }
    String value = getText();
    if (value.matches(validatorRegex)) {
        this.setStyle("-fx-text-box-border: #039ed3;");
        return true;
    }

    this.setStyle("-fx-text-box-border: #fc0606;");
    return false;
}
```

تابع validate در کلاس ValidatorField



نمونه ای از ValidatorField به همراه یک ورودی معتبر و یک ورودی نامعتبر در صفحه ی ثبت نام کاربران

- Broken authentication :
  - سناریو : مهاجم سعی می کند بدون دسترسی به یوزرنیم و پسوورد (یا هرگونه اطلاعات احراز هویت) یک کاربر، به اکانت او دسترسی پیدا کند و خود را به عنوان آن کاربر به سرور معرفی کند. این کار ممکن است با به دست آوردن اطلاعات کاربر از پیام های ارسالی انجام شود. همچنین مهاجم می تواند با ارسال پیام های خود از طریق درگاه ارتباطی سرور با کلاینت مزبور خور را به جای وی جا بزند.
  - راه حل : با رمز کردن پیام ها می توانیم از دسترسی مهاجم به اطلاعات هویتی کاربر در پیام ها جلوگیری کنیم. همچنین برای جلوگیری از حمله ی نوع دوم، کافیست کلید واژه ای میان کلاینت و سرور قرار داد کنیم که در پیام ها ظاهر شوند تا مشخص شود که پیام قرار گرفته بر روی درگاه ارتباطی سرور با کلاینت از طرف خود کلاینت معتبر باشد.
  - استفاده در پروژه : کلاینت و سرور در ابتدای اتصال به یکدیگر یک توکن را با یکدیگر قرار داد میکنند و از آن پس کلاینت در تمام پیام های خود این توکن را به گونه ای می آورد و سرور با بررسی آمدن توکن مرتبط با کلاینت معتبر در پیام ارسالی، از عدم حمله ی مهاجم اطمینان حاصل می کند.

```java
private String getInputReady(String clientMessage) throws WrongTokenException {
    Matcher tokenMatcher = Pattern.compile("^(\\d+)T(\\d+)T").matcher(clientMessage);
    if (!tokenMatcher.find()) {
        throw new WrongTokenException();
    }
    int receivedToken = Integer.parseInt(tokenMatcher.group(2));
    int receivedMC = Integer.parseInt(tokenMatcher.group(1));
    if (this.token != receivedToken || receivedMC != messageCounter) {
        throw new WrongTokenException();
    }
    messageCounter++;
    return clientMessage.substring(tokenMatcher.end());
}
```

اعتبار سنجی توکن در پیام قرار گرفته بر درگاه کلاینت ، توسط سرور در کلاس ClientHandler