

# Web网盘产品文档

## 1 引言

本网盘第一版本已经部署到服务器上，可以直接访问，测试账号为 rytter，测试密码为 rytter

注意，这是软件的第一版，如要部署美化后第二版，请使用 war 包直接部署。

服务器网站为 [http://106.12.116.250/demo\\_war\\_exploded/](http://106.12.116.250/demo_war_exploded/)

### 1.1 小组成员及分工

学号	姓名	分工
		组长 软件的开发，文档撰写，CSRF
		xss 入侵，CSRF
		软件 ui 美化，文档撰写
		抓包测试分析，软件测试
		sql 注入,命令注入攻击

### 1.2 编写目的

本文档描述 web 应用采用的开发技术、部署方法、业务功能和每种功能的使用方法图解以及接口设计（包括接口功能描述、uri、入口参数和输出参数）。

## 2 项目描述

### 2.1 任务内容

#### 2.1.1 目标

设计并开发实现一个简单的 web 应用，例如外卖订单管理、会议室使用登记、机动车停车管理、景区在线订票管理、医院预约挂号管理等等。该应用需要有简单的页面，除了实现简单的核心业务特性外，还需要包含用户管理模块（包括用户注册/删除、用户登录/登出、用户信息维护、用户操作记录等特性）作为最基本的支撑模块。

#### 2.1.2 使用技术推荐

- (1) web 前端编程技术，例如当前流行的 JS 框架等进行开发；
- (2) web 后端编程技术，例如 Java 框架、PHP、Go、Python 框架以及 Nodejs 等进行开发）；
- (3) web 服务器和应用服务器，例如 ginx/IIS/Apache/Lighttpd/Tomcat/Jboss；
- (4) 数据库软件，例如 Mysql、PGSQL、Oracle、sqlserver 等；
- (5) 数据持久化框架（MyBatis、Hibernate、JDBCTemplate 等）；
- (6) 其他最新技术，例如 redis 内存数据库、消息中间件（RocketMQ 等）；
- (7) 部署系统尽量选择 linux，当然 windows 也可以作为部署系统使用。

### 2.1.3 要求

该 web 应用部署在本地（localhost）并且能够使用 chrome/火狐/edge 浏览器通过本机或者局域网主机完成网站页面浏览，并进行用户登录、用户管理及核心业务的运行和操作。另外，该 web 应用的访问协议需要使用 http 和 https 两种方式。

部署和运行 web 应用后，使用浏览器自带调试工具进行 request 和 response 抓取和数据分析，并且通过部署 burpsuite、tcpdump、sniffer、wireshark 等工具进行抓包，对比分析 http 和 https 协议。

## 2.2 设计思路

先写后端，其次进行前端美化，最后进行调试抓包分析。

## 2.3 环境

(1)硬件环境

- windows服务器：
- linuxs服务器：

(2)软件环境

- 开发系统：ubuntu 22.04
- 操作系统：win10及以上；
- 数据库：MySQL；
- 网络协议：
- 开发环境：
- 使用软件：

## 3 项目设计

### 3.1 设计原则

在开发本项目的过程中，主要遵循了以下几个原则：

- 项目基本上完整实现任务内容，支持后续实验测试；
- 项目需稳定支持用户长时间使用；
- 用户使用方便、界面友好、配置管理简单灵活。

### 3.2 程序文件

以下为主要程序文件

序号	名称	说明
01	DownloadServlet.class	接受到 main 当中传递的文件名，然后启动传输进行文件的下载
02	Login.class	接受输入的参数并进行数据库查询，查看有无此用户
03	Main.class	进行简单的界面展示（如当前网盘内的文件），并提供朋友圈的跳转
04	PublicServlet.class	负责对朋友圈发表内容进行储存，并将数据传输为 public.jsp，进行朋友圈内容的显示
05	SignUp.class	负责对用户进行注册，并将注册的结果放入数据库中
06	UploadServlet.class	负责对上传文件进行处理，并将上传的文件与用户放入到数据库中

以下为主要界面文件

序号	文件名称	说明
01	index.jsp	登录/注册页面，可以选择进行登录或注册操作
02	Login.jsp	登录页面
03	main.jsp	网盘主页面，可以查看当前用户之前上传至网盘的文件
04	Public.jsp	朋友圈页面，可以查看所有用户发表的朋友圈内容
05	SignUp.jsp	注册页面，用于用户注册
06	Upload.jsp	上传文件页面，可以将文件上传至网盘
07	style.css	前端美化文件

## 4 界面设计要求

用户界面采用简洁的设计风格，美观的同时便于用户进行操作，也增加了整个系统运行的稳定性、加载能力。

### 4.1 用户界面设计

#### (1) 字体

sans-serif, Sonsie One, ZCOOL KuaiLe, Open Sans Condensed, 微软雅黑, 正常体/微粗体, (12至20)px, 黑色/白色 (打印文字不在此限)。

#### (2) 风格

采用全屏网页设计，化繁为简的设计思维，让整个网站的整体性、统一性、灵活性、自适应性、流畅性得到了相对的提高，也使得平台的功能处理和管理能力在这些特点的加持之下得到综合性的展示。

#### (3) 色系

- 主题色调：蓝、白、黑。
- 嵌入色调：灰、蓝。

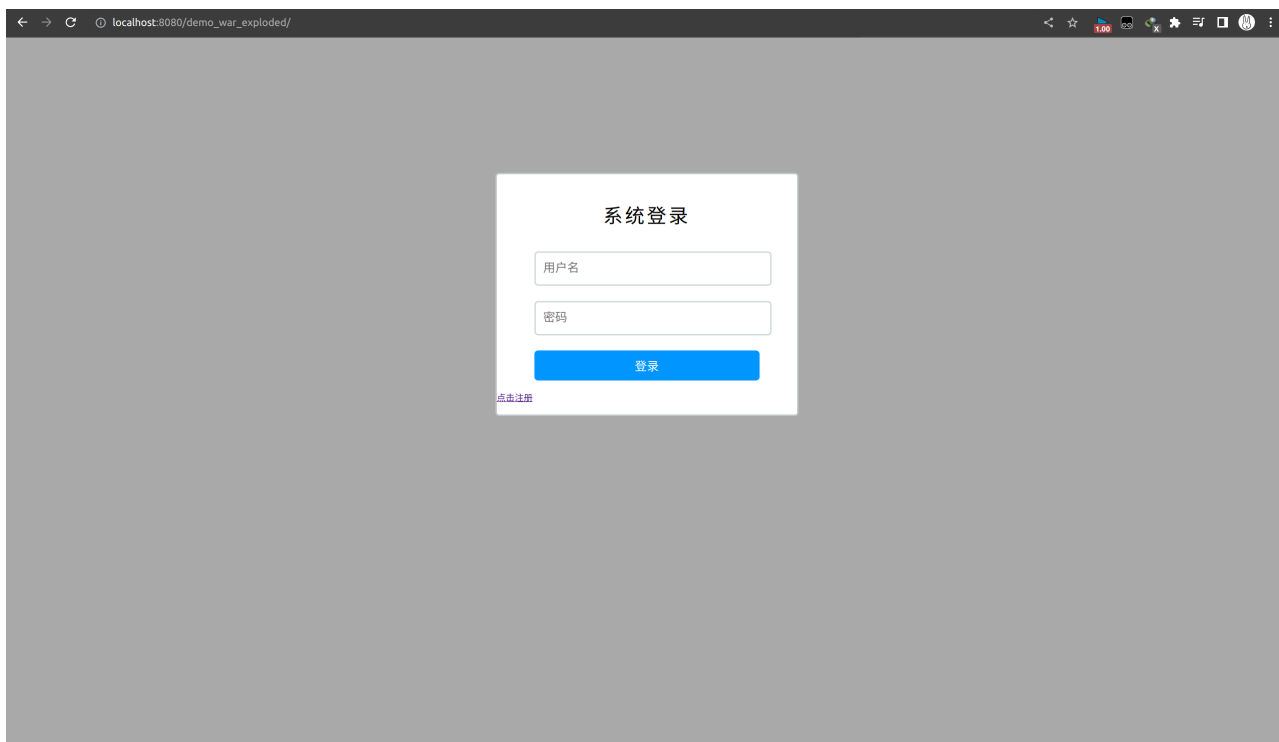
#### (4) 控件

##### 1) 尺寸

在合理的布局下尽可能多的显示页面内容。

##### 2) 布局

按照操作流程或浏览顺序自左至右、由上而下的排放各种控件，使界面整体协调、简洁、美观大方。



## 5 架构设计

### 5.1 部署方法

本程序拥有一个 war 包，只需要将这个 war 包放入到 tomcat 的 webapps 目录下并启动，tomcat 将自动部署这个包

同时需要在 tomcat 的目录下放置 mysql-connector-java-8.0.30.jar 包和两个 commons 包(common

fileupload-1.4.jar 和 commons-io-2.11.0.jar)，这样才可以正常访问本地的 MySQL

关于sql数据库的配置

首先需要有一个 rytter 用户密码为 rytter，rytter 用户下面需要有一个数据库 mysql\_java，数据库中应有两个表 files 和 users。以下为两个表的主要配置信息。

```
mysql> desc users;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| name       | char(45)  | YES  |     | NULL    |       |
| password   | char(20)  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.01 sec)
```

```
mysql> desc file
-> ;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| username   | varchar(20)   | NO   |     | NULL    |       |
| filename   | varchar(50)   | NO   |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

## 5.2 业务功能及使用方法图解

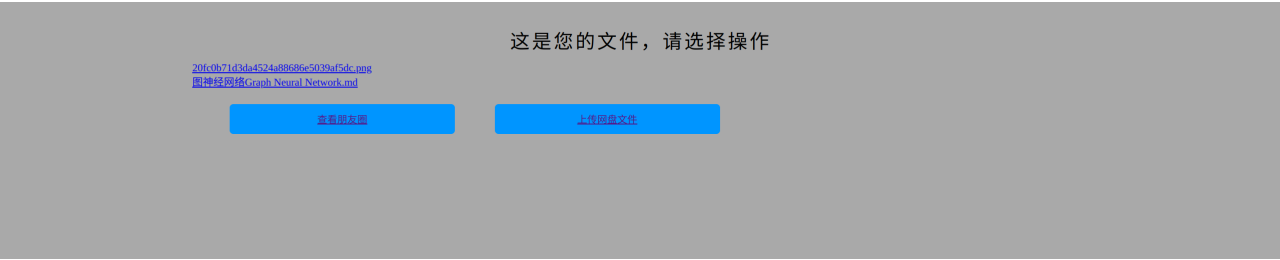
1.首先进行注册操作，输入想要注册的用户名，及密码，邀请码为 `rytter`，添加邀请码的原因是防止大量的散人用户登录网盘消耗服务器的资源。

### 用户注册

注册

2.注册成功后可以直接登录

这是网站的主界面，点击链接可以直接下载自己的网盘文件，点击查看朋友圈可以查看自己的朋友圈，点击上传网盘文件可以开始向自己的网盘添加文件。



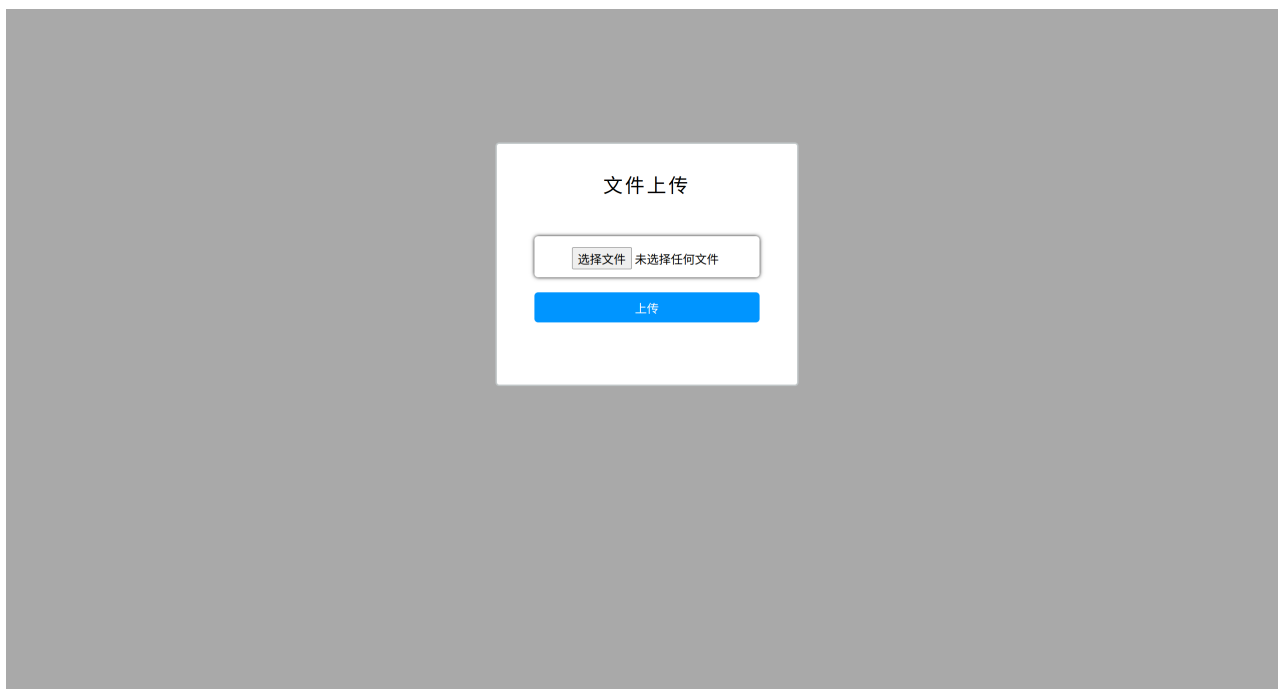
### 3.查看朋友圈

朋友圈可以以自己的账号名发送，所有人都可以看到朋友圈的消息



### 4.上传文件的操作

首先选择自己要上传的文件，然后点击上传按钮就可以看到自己上传的信息了



## 5.3 接口设计

此处通过基本的文件功能来描述各个文件之间的关系

### 5.3.1 用户登录的操作

首先获得用户提交的密码，然后通过查询本地的数据库查看当前用户的密码是否正确，如果正确的话就定位到 `main` 这个 `servlet`，并且在 `session` 中添加用户名这个信息，如果不正确就会 `out.println` 显示没有这个信息并弹出注册网站的链接

```
1 package com.example.demo;
2
3 import javax.servlet.annotation.WebServlet;
4 import javax.servlet.http.HttpServlet;
5 import javax.servlet.http.HttpServletRequest;
6 import javax.servlet.http.HttpServletResponse;
7 import javax.servlet.http.HttpSession;
8
9 import java.io.IOException;
10 import java.io.PrintWriter;
11 import java.sql.*;
12 import java.util.Objects;
13
14 @WebServlet(name = "login", value = "/login")
15 public class Login extends HttpServlet {
16     static final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
17     static final String DB_URL = "jdbc:mysql://localhost:3306/mysql_java";
18     public void doGet(HttpServletRequest request, HttpServletResponse
19 response) throws IOException {
20         response.setContentType("text/html");
21         String name = request.getParameter("name");
22         String password = request.getParameter("password");
```

```

22     HttpSession session = request.getSession();
23     // Hello
24     PrintWriter out = response.getWriter();
25     String USER = "rytter";
26     String PASS = "rytter";
27     Connection conn = null;
28     Statement stmt = null;
29     try {
30         Class.forName("com.mysql.cj.jdbc.Driver");
31         conn = DriverManager.getConnection(DB_URL, USER, PASS);
32         // 执行查询
33         stmt = conn.createStatement();
34         String sql;
35         sql = "SELECT name,password FROM users";
36         ResultSet rs = stmt.executeQuery(sql);
37         boolean exits = false;
38         // 展开结果集数据库
39         while (rs.next()) {
40             // 通过字段检索
41             String name_sql = rs.getString("name");
42             String password_sql = rs.getString("password");
43             if (Objects.equals(name_sql, name) &
Objects.equals(password_sql, password)) {
44                 exits = true;
45                 break;
46             }
47         }
48         if (!exits) {
49             out.println("<h1>没有看到你的名字，请进行注册</h1>");
50             out.println("<a href=\"SignUp.jsp\">点击注册</a>");
51             return;
52         }
53         else {
54             out.println("<h1>欢迎登录小废物做的网盘</h1>");
55             // 添加session属性
56             session.setAttribute("username", name);
57             response.sendRedirect("/demo_war_exploded/main");
58         }
59         // 完成后关闭
60         rs.close();
61         stmt.close();
62         conn.close();
63     }
64     catch (ClassNotFoundException e) {
65         out.println("<h1>" + e + "</h1>");
66         throw new RuntimeException(e);
67     }

```



```

68         catch (SQLException e) {
69             out.println("<h1>" + e + "</h1>");
70             throw new RuntimeException(e);
71         }
72     }
73 }

```

### 5.3.2 用户注册的操作

用户注册首先会查询有无该用户名，如果没有这个用户名则可以进行注册，也就是写入数据库这种情况

```

1  package com.example.demo;
2
3  import javax.servlet.annotation.*;
4  import javax.servlet.annotation.WebServlet;
5  import javax.servlet.http.*;
6
7  import java.io.IOException;
8  import java.io.PrintWriter;
9  import java.sql.*;
10 import java.util.Objects;
11
12 @WebServlet(name = "signup", value = "/signup")
13 public class SignUp extends HttpServlet {
14     static final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
15     static final String DB_URL = "jdbc:mysql://localhost:3306/mysql_java";
16
17     public void doGet(HttpServletRequest request, HttpServletResponse
response) throws IOException {
18         String name = request.getParameter("name");
19         String password = request.getParameter("password");
20         String confirm_code = request.getParameter("confirm_code");
21         response.setContentType("text/html;charset=UTF-8");
22         PrintWriter out = response.getWriter();
23         Connection conn = null;
24         Statement stmt = null;
25         String USER = "rytter";
26         String PASS = "rytter";
27         String confirm_code_judge = "rytter";
28         if (Objects.equals(confirm_code, confirm_code_judge)) {
29             try {
30                 Class.forName(JDBC_DRIVER);
31                 System.out.println("连接数据库...");
32                 conn = DriverManager.getConnection(DB_URL, USER, PASS);
33                 // 执行查询
34                 System.out.println(" 实例化Statement对象...");
35                 stmt = conn.createStatement();
36                 String sql;

```

```

37         sql = "SELECT name FROM users";
38         ResultSet rs = stmt.executeQuery(sql);
39
40         // 展开结果集数据库
41         while (rs.next()) {
42             // 通过字段检索
43             String name_sql = rs.getString("name");
44             if (Objects.equals(name_sql, name)) {
45                 //查询有没有这个名字
46                 out.println("<h1>" + "该用户名存在, 请更改用户名" + "
</h1>");
47                 return;
48             }
49         }
50         //如果没有这个名字就添加上
51         String sql2 = "INSERT users (name,password) VALUES ('";
52         sql2=sql2+name;
53         sql2=sql2+ "','";
54         sql2=sql2+password;
55         sql2=sql2+"'')";
56         int counts = stmt.executeUpdate(sql2);
57         System.out.println("正在查询");
58         out.println("<h1>"+counts+"</h1>");
59         out.println("<html><body>");
60         out.println("<h1>" + "你的名称是: " + name + "<br/>" + "你的密码
是: " + password + "</h1>");
61         out.println("</body></html>");
62         out.println("<a href=\"Login.jsp\">点击登录</a>");
63         // 完成后关闭
64         rs.close();
65         stmt.close();
66         conn.close();
67     }
68     catch (ClassNotFoundException e) {
69         out.println("<h1>" + e.getMessage() + "</h1>");
70         System.out.println(e.getMessage());
71         throw new RuntimeException(e);
72     }
73     catch (SQLException e) {
74         out.println("<h1>" + e.getMessage() + "</h1>");
75         throw new RuntimeException(e);
76     }
77 }
78 else {
79     out.println("<h1>" + "邀请码错误, 请联系管理员" + "</h1>");
80 }
81 }

```

### 5.3.3 用户朋友圈的操作

这个朋友圈的方法会有点笨，我们会将用户上传的评论进行处理，就是将这些评论写成一个 txt 文件，然后等用户访问这个界面的时候将 txt 文件中的信息放到 session 中，然后在 jsp 中直接给 out.println 出来，这样就完成了基本的一个朋友圈操作

```
1 package com.example.demo;
2
3 import com.mysql.cj.Session;
4 import org.apache.commons.fileupload.FileItem;
5 import org.apache.commons.fileupload.FileItemFactory;
6 import org.apache.commons.fileupload.FileUploadException;
7 import org.apache.commons.fileupload.disk.DiskFileItemFactory;
8 import org.apache.commons.fileupload.servlet.ServletFileUpload;
9
10 import javax.servlet.ServletException;
11 import javax.servlet.annotation.WebServlet;
12 import javax.servlet.http.HttpServlet;
13 import javax.servlet.http.HttpServletRequest;
14 import javax.servlet.http.HttpServletResponse;
15 import javax.servlet.http.HttpSession;
16 import java.io.*;
17 import java.io.IOException;
18 import java.util.ArrayList;
19 import java.util.List;
20
21 @WebServlet(name = "publicservlet", value = "/publicservlet")
22 public class PublicServlet extends HttpServlet {
23     @Override
24     protected void doGet(HttpServletRequest req, HttpServletResponse resp)
25     throws ServletException, IOException {
26         File file = new File("public_text.txt");
27         List<String> list = new ArrayList();
28         if (file.exists()) {
29             BufferedReader reader = null;
30             try {
31                 reader = new BufferedReader(new FileReader(file));
32                 String readStr;
33                 while ((readStr = reader.readLine()) != null) {
34                     list.add(readStr);
35                 }
36                 reader.close();
37             } catch (IOException e) {
38                 e.printStackTrace();
39             }
```

```

40         finally {
41             if (reader != null) {
42                 try {
43                     reader.close();
44                 }
45                 catch (IOException e1) {
46                     e1.printStackTrace();
47                 }
48             }
49         }
50     }
51     else {
52         boolean create = file.createNewFile();
53         if (create) {
54             System.out.println("创建成功");
55         }
56         else {
57             System.out.println("创建失败");
58         }
59     }
60     //到这个地方我们就已经把结果放到list中了
61     HttpSession session = req.getSession();
62     session.setAttribute("text", list);
63     resp.sendRedirect("/demo_war_exploded/Public.jsp");
64 }
65
66 @Override
67 protected void doPost(HttpServletRequest req, HttpServletResponse resp)
68 throws ServletException, IOException {
69     String public_text="";
70     req.setCharacterEncoding("utf-8");
71     resp.setCharacterEncoding("utf-8");
72     resp.setContentType("text/html;charset=utf-8");
73     req.setCharacterEncoding("UTF-8");
74     resp.setContentType("text/html;charset=utf-8");
75     FileItemFactory factory=new DiskFileItemFactory();
76     ServletFileUpload upload =new ServletFileUpload(factory);
77     try {
78         List<FileItem> items=upload.parseRequest(req);//这个地方可能会有问
79         for (FileItem item:items){
80             if(item.getFieldName().equals("public_text")){
81                 if(!item.getString().equals("")){
82                     public_text= item.getString("UTF-8");
83                 }
84             }
85         }
86     }

```

题

```

85         }
86         catch (FileUploadException e) {
87             throw new RuntimeException(e);
88         }
89
90         HttpSession session = req.getSession();
91         String username = (String) session.getAttribute("username");
92         File file = new File("public_text.txt");
93         BufferedWriter output = new BufferedWriter(new FileWriter(file,
true));
94         username = "<h2>" + username + "</h2><br>";
95         public_text = "<p>" + public_text + "</p><br>";
96         output.write(username);
97         output.write("\n");
98         output.write(public_text);
99         output.write("\n");
100        output.flush();
101        output.close();
102        resp.sendRedirect("/demo_war_exploded/main.jsp");
103    }
104    ;
105 }

```

#### 5.3.4 用户上传文件的操作

用户上传文件就使用到了 commons 的 jar 包中的一些数据，通过网上给的例子我们能很轻松的写出这些数据，我们会将上传的文件和上传的文件名放到 mysql\_java 中的 files 表中，这样可以在 main 中显示的时候显示出用户的文件

```

1  package com.example.demo;
2
3  import javax.servlet.annotation.WebServlet;
4  import javax.servlet.http.HttpServlet;
5  import javax.servlet.http.HttpServletRequest;
6  import javax.servlet.http.HttpServletResponse;
7  import javax.servlet.http.HttpSession;
8
9  import org.apache.commons.fileupload.*;
10 import org.apache.commons.fileupload.disk.DiskFileItemFactory;
11 import org.apache.commons.fileupload.servlet.ServletFileUpload;
12 import org.apache.commons.io.*;
13
14 import java.io.File;
15 import java.io.FileOutputStream;
16 import java.io.InputStream;
17 import java.io.PrintWriter;
18 import java.util.List;
19
20 import java.sql.*;

```

```

21
22 //@WebServlet(name = "uploadservlet", value = "/uploadservlet")
23 public class UploadServlet extends HttpServlet{
24     static final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
25     static final String DB_URL = "jdbc:mysql://localhost:3306/mysql_java";
26     public void doPost(HttpServletRequest request, HttpServletResponse
response){
27         //一些数据库初始化的代码
28         Connection conn = null;
29         Statement stmt = null;
30         String USER = "rytter";
31         String PASS = "rytter";
32         try{
33             HttpSession session=request.getSession();
34             String username=(String)session.getAttribute("username");
35
36             request.setCharacterEncoding("UTF-8");
37             response.setContentType("text/html;charset=utf-8");
38             PrintWriter out =response.getWriter();
39             FileItemFactory factory=new DiskFileItemFactory();
40             ServletFileUpload upload =new ServletFileUpload(factory);
41             List<FileItem> items=upload.parseRequest(request);//这个地方可能会
有问题
42             String filename="";
43             InputStream is = null;
44             for (FileItem item:items){
45                 //处理普通的表单
46                 if(item.isFormField()){
47                     if(item.getFieldName().equals("filename")){
48                         if(!item.getString().equals("")){
49                             filename= item.getString("UTF-8");
50                         }
51                     }
52                 }
53                 else if(item.getName()!=null && !item.getName().equals("")){
54                     //将客户端发出的文件路径名字进行截取，获得文件名字
55
56                     filename=item.getName().substring(item.getName().lastIndexOf("\\")+1);
57                     is = item.getInputStream();//获得上传文件的InputStream对象
58                 }
59                 File file=new File(filename);
60                 if(file.exists()){
61                     out.println("文件已经存在,请更换文件名后重新上传");
62                     System.out.println(file.getAbsolutePath());
63                 }
64                 else if(!filename.equals("")){
65                     //使用fileoutputstream打开服务器端的上传文件

```

```

65         FileOutputStream fos=new FileOutputStream(filename);
66         byte[] buffer=new byte[8192];
67         int count=0;
68         while((count=is.read(buffer))>0){
69             fos.write(buffer,0,count);
70         }
71         fos.close();
72         is.close();
73         out.println("文件上传成功");
74         //开始进行数据库操作
75         Class.forName(JDBC_DRIVER);
76         System.out.println("连接数据库...");
77         conn = DriverManager.getConnection(DB_URL, USER, PASS);
78         // 执行查询
79         System.out.println(" 实例化Statement对象...");
80         stmt = conn.createStatement();
81         String sql2 = "INSERT file (username,filename) VALUES
('";
82         sql2=sql2+username;
83         sql2=sql2+ "','";
84         sql2=sql2+filename;
85         sql2=sql2+"')";
86         int counts = stmt.executeUpdate(sql2);
87         System.out.println(counts);
88         System.out.println(filename);
89         System.out.println(username);
90         System.out.println();
91         response.sendRedirect("/demo_war_exploded/main");
92     }
93 }
94 }
95 catch (Exception e){
96     System.out.println(e.getMessage());
97 }
98 }
99 }

```

### 5.3.5 用户下载文件的操作

我们通过文件流的操作可以将这些网上的文件进行传递,就是 FileInputStream

```

1 package com.example.demo;
2
3 import javax.servlet.ServletContext;
4 import javax.servlet.ServletException;
5 import javax.servlet.annotation.WebServlet;
6 import javax.servlet.http.HttpServlet;
7 import javax.servlet.http.HttpServletRequest;

```

```

8  import javax.servlet.http.HttpServletResponse;
9  import javax.servlet.http.HttpSession;
10 import java.io.*;
11 import java.net.URLEncoder;
12
13 @WebServlet(name = "downloadservlet", value = "/downloadservlet")
14 public class DownloadServlet extends HttpServlet {
15     @Override
16     protected void doPost(HttpServletRequest req, HttpServletResponse resp)
17     throws ServletException, IOException {
18         super.doPost(req, resp);
19     }
20     @Override
21     protected void doGet(HttpServletRequest request, HttpServletResponse
22     response) throws ServletException, IOException {
23         //super.doGet(req, resp);
24         {
25             //检验是不是第一次登录
26             HttpSession session=request.getSession();
27             String username=(String)session.getAttribute("username");
28             if (username==null){
29                 response.sendRedirect("/demo_war_exploded/Login.jsp");
30                 return;
31             }
32         }
33         String filename = request.getParameter("filename");
34         try{
35             File file = new File(filename);
36             boolean exists = file.exists();
37             if (!exists){
38                 throw new Exception("文件不存在! ");
39             }
40             String ext = filename.substring(filename.lastIndexOf(".") +
41 1).toLowerCase();
42             FileInputStream fileInputStream = new FileInputStream(file);
43             InputStream fis = new BufferedInputStream(fileInputStream);
44             byte[] buffer = new byte[fis.available()];
45             fis.read(buffer);
46             fis.close();
47             response.reset();
48             response.setCharacterEncoding("UTF-8");
49             response.addHeader("Content-Disposition", "attachment;filename="
+ URLEncoder.encode(filename, "UTF-8"));
50             response.addHeader("Content-Length", "" + file.length());
51             OutputStream outputStream = new
52             BufferedOutputStream(response.getOutputStream());
53             response.setContentType("application/octet-stream");

```



```
51         outputStream.write(buffer);
52         outputStream.flush();
53     }
54     catch (Exception e) {
55         e.printStackTrace();
56     }
57 }
58 }
```

## 6 测试

### 6.1 概述

部署和运行 web 应用后，使用浏览器自带调试工具进行 request 和 response 抓取和数据分析，并且通过部署 burpsuite、tcpdump、sniffer、wireshark 等工具进行抓包，对比分析 http 和 https 协议。

访问 http

利用浏览器自带进行抓取数据

查看朋友圈的 response，request 为空

```
1
2
3
4
5 <html>
6
7 <head>
8   <link rel="stylesheet" href="style.css">
9   <title>web安全作业 (网盘) </title>
10 </head>
11
12 <body>
13   <header>
14     <h1>网盘朋友圈</h1>
15   </header>
16   <main>
17     <div id="public_text"></div>
18     <article>
19       <h2>ysheng</h2></br><p>Hello world</p></br>
20     </article>
21   </main>
22   <footer>
23     <form action="publicservlet" method="post" enctype="multipart/form-d
24       <div class="public_page">
25         <p>发朋友圈: <input type="text" name="public_text" /></p>
26         <input type="submit" value="上传" style="width:50px">
27       </div>
28     </form>
29   </footer>
30 </body>
31
32 </html>
```

上传网盘文件 request

所有

HTML

CSS

JS

XHR

字体

图像

媒体

WS

其他

禁用缓存

不节流

⚙️

消息头

Cookie

请求

响应

耗时

过滤消息头

拦截

重发

GET http://127.0.0.1:8080/demo\_war\_exploded/publicservlet

状态

302

版本

HTTP/1.1

传输

882 字节 (大小 720 字节)

Referrer 策略

strict-origin-when-cross-origin

请求优先级

Highest

响应头 (162 字节)

原始

HTTP/1.1 302

Location: /demo\_war\_exploded/Public.jsp

Content-Length: 0

Date: Wed, 28 Dec 2022 12:20:42 GMT

Keep-Alive: timeout=20

Connection: keep-alive

请求头 (639 字节)

原始

GET /demo\_war\_exploded/publicservlet HTTP/1.1

Host: 127.0.0.1:8080

User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86\_64; rv:108.0) Gecko/20100101 Firefox/108.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,\*/\*;q=0.8

Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2

Accept-Encoding: gzip, deflate, br

Connection: keep-alive

Referer: http://127.0.0.1:8080/demo\_war\_exploded/main.jsp

Cookie: JSESSIONID=6439ED65633A2031C1DE2965E2A415D5

Upgrade-Insecure-Requests: 1

Sec-Fetch-Dest: document

Sec-Fetch-Mode: navigate

Sec-Fetch-Site: same-origin

Sec-Fetch-User: ?1

↑ 网络

样式编辑器

性能

内存

存储

无障碍环境

应用程序

发起者

类型

大小

消息头

Cookie

请求

响应

耗时

docu... htm 858 ... 571

docu... htm 756 ... 571

docu... htm 733 ... 571

Favic... x-ico 已缓存 21.6

请求有效载荷 (payload)

-----312890235124831130630020378035

Content-Disposition: form-data; name="file"; filename="tomcat.keystore"

Content-Type: application/octet-stream

bipbip

+

V

0P0

g

j

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

response

```
HTML 原始
1
2
3 <!DOCTYPE html>
4
5 <head>
6   <link rel="stylesheet" href="style.css">
7   <title>web安全作业（网盘）</title>
8 </head>
9
10 <body>
11   <form action="uploadServlet" method="post" enctype="multipart/form-data">
12     <div class="upload-box">
13       <legend class="title">文件上传</legend>
14       <div class="button_box2"><input type="file" name="file" /></div>
15       <input class="button_box" type="submit" value="上传">
16     </div>
17
18   </form>
19 </body>
20
21 </html>
```

```

92 154.8563669935 127.0.0.1 127.0.0.1 HTTP 251 HTTP/1.1 302
+ 93 154.8704361117 127.0.0.1 127.0.0.1 HTTP 692 GET /demo_war_exploded/main.jsp HTTP/1.1
94 154.870456645 127.0.0.1 127.0.0.1 TCP 66 8080 - 39838 [ACK] Seq=386 Ack=1923 Win=65024 Len=0 TSval=2052477031 TSecr=2052477031
95 154.8943884667 127.0.0.1 127.0.0.1 TCP 66 30958 - 3306 [ACK] Seq=2012 Ack=4191 Win=65536 Len=0 TSval=2052477055 TSecr=2052477015
96 155.280812189 127.0.0.1 127.0.0.1 HTTP 1198 HTTP/1.1 200 (text/html)
97 155.326329263 127.0.0.1 127.0.0.1 TCP 66 39838 - 8088 [ACK] Seq=1923 Ack=1518 Win=65536 Len=0 TSval=2052477447 TSecr=2052477442
98 155.2872202087 127.0.0.1 127.0.0.1 TCP 66 [TCP Keep-Alive] 39838 - 8080 [ACK] Seq=1922 Ack=1518 Win=65536 Len=0 TSval=2052487448 TSecr=2052477442
99 165.5879829652 157.2.0.0 157.2.0.0 TRD 66 [TRD Keep-Alive] 8080 - 39838 [ACK] Seq=1518 Ack=1923 Win=65536 Len=0 TSval=2052487448 TSecr=2052477447
Frame 96: 1198 bytes on wire (9584 bits), 1198 bytes captured (9584 bits) on interface lo, id 8
Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 8080, Dst Port: 39838, Seq: 386, Ack: 1923, Len: 1132
Hypertext Transfer Protocol
Line-based text data: text/html (30 lines)
\n
\n
\n
\n
\n
\n
<!DOCTYPE html>\n
\n
<head>\n
<link rel="stylesheet" href="style.css">\n
<title>web安全作业 (网盘)</title>\n
</head>\n

```

上传文件



发起者	类型	传输	大小	消息头	Cookie	请求	响应	耗时	安全性
document	html	1.04 kb	847 字节	通过请求参数					
document	html	1.06 kb	847 字节	请求有效载荷 (payload)					
document	html	1.03 kb	847 字节	<pre> 1 Content-Disposition: form-data; name="file"; filename="test3.c" 2 Content-Type: text/x-csrc 3 4 5 #include &lt;stdio.h&gt; 6 #include &lt;stdlib.h&gt; 7 #include &lt;unistd.h&gt; 8 #include &lt;pthread.h&gt; 9 #include &lt;semaphore.h&gt; 10 #define N 1 // 消费者或者生产者的数目 11 #define M 5 // 缓冲数目 12 int in = 0; // 生产者在放置产品的位置 13 int out = 0; // 消费者取产品的位置 14 int buff[M] = {0}; // 缓冲初始化为0，开始时没有产品 15 sem_t empty_sem; // 同步信号量，当满了时阻止生产者放产品 16 sem_t full_sem; // 同步信号量，当空产品时阻止消费者消费 17 pthread_mutex_t mutex; // 互斥信号量，一次只有一个线程访问缓冲 18 int product_id = 0; // 生产者id 19 int purchase_id = 0; // 消费者id 20 /* 打印缓冲情况 */ 21 void print() 22 { 23     int i; 24     for(i = 0; i &lt; M; i++) 25         printf("%d ", buff[i]); 26     printf("\n"); 27 } 28 /* 生产者方法 */ 29 void *product() 30 { 31     int id = ++product_id; 32     while(1) 33     { 34         // 用sleep的数量可以调节生产和消费的速度，便于观察 35         sleep(1); 36         pthread_mutex_lock(&amp;mutex); 37         sem_wait(&amp;empty_sem); 38         buff[in] = 1; 39         printf("productid is %d, like: %t", id, in); 40         print(); 41         in = in % M; 42         pthread_mutex_unlock(&amp;mutex); 43         sem_post(&amp;full_sem); 44     } 45 } 46 /* 消费者方法 */ 47 void *purchase() 48 { 49     int id = ++purchase_id; 50     while(1) 51     { 52         // 用sleep的数量可以调节生产和消费的速度，便于观察 53         sleep(1); 54         pthread_mutex_lock(&amp;mutex); 55         sem_wait(&amp;full_sem); 56         printf("purchaseid is %d, like: %t", id, out); 57         print(); 58         out = out % M; 59         pthread_mutex_unlock(&amp;mutex); 60         sem_post(&amp;empty_sem); 61         buff[out] = 0; 62         printf("\n"); 63     } 64 } 65 int main() 66 { 67     pthread_t producer, consumer; 68     sem_init(&amp;empty_sem, 0, M); 69     sem_init(&amp;full_sem, 0, 0); 70     pthread_mutex_init(&amp;mutex, 0); 71     pthread_create(&amp;producer, 0, product, 0); 72     pthread_create(&amp;consumer, 0, purchase, 0); 73     pthread_join(producer, 0); 74     pthread_join(consumer, 0); 75     return 0; 76 } </pre>					

intentLoaded: 619 毫秒 | load: 628 毫秒

response

消息头	Cookie	请求	响应	耗时	安全性
HTML					
<pre> 1 2 3 4 5 6 &lt;!DOCTYPE html&gt; 7 8 &lt;head&gt; 9   &lt;link rel="stylesheet" href="style.css"&gt; 10   &lt;title&gt;web安全作业（网盘）&lt;/title&gt; 11 &lt;/head&gt; 12 13 &lt;body&gt; 14 &lt;div&gt; 15   &lt;legend class="title"&gt;这是您的文件，请选择操作&lt;/legend&gt; 16 17   &lt;a style="font-size:14px" href="/demo_war_exploded/downloadervlet?filename=commons-fileupload-1.4.jar"&gt; 18   &lt;a style="font-size:14px" href="/demo_war_exploded/downloadervlet?filename=test2.c"&gt; test2.&lt;/a&gt;&lt;br&gt; 19   &lt;a style="font-size:14px" href="/demo_war_exploded/downloadervlet?filename=test3.c"&gt; test3.&lt;/a&gt;&lt;br&gt; 20 21   &lt;button class="button_box"&gt;&lt;a href="publicervlet"&gt;查看朋友圈&lt;/a&gt;&lt;/button&gt; 22   &lt;button class="button_box"&gt;&lt;a href="Upload.jsp"&gt;上传网盘文件&lt;/a&gt;&lt;/button&gt; 23 24 &lt;/div&gt; 25 &lt;/body&gt; 26 27 &lt;/html&gt; 28 </pre>					

wireshark 抓包

登录 http

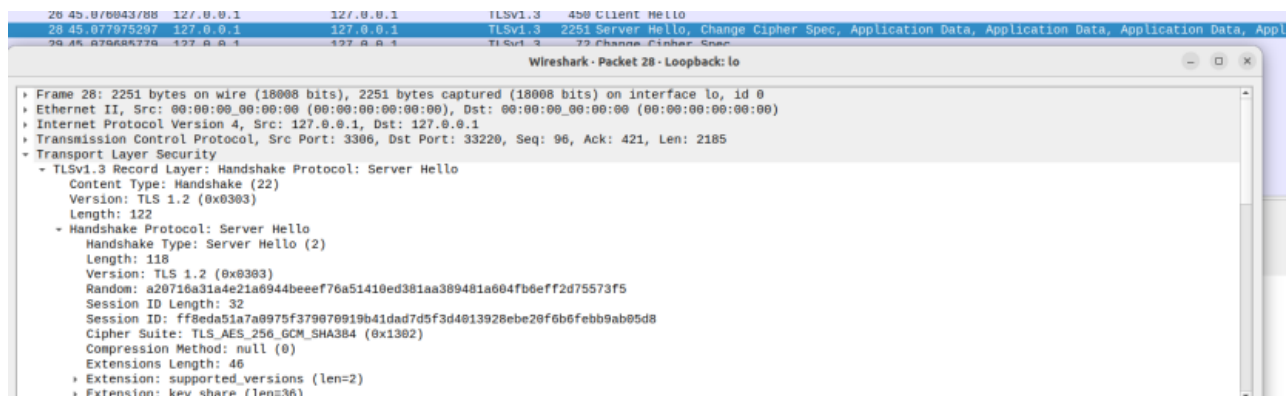


```
99 38.521456836 127.0.0.1 127.0.0.1 TCP 74 41456 → 8080 [SYN] Seq=0 Win=65496 Len=0 MSS=65496 SACK_PERM=1 TSval=1146248756 TSecr=0 WS=128
100 38.521471483 127.0.0.1 127.0.0.1 TCP 74 8080 → 41456 [SYN, ACK] Seq=0 Ack=1 Min=65483 Len=0 MSS=65496 SACK_PERM=1 TSval=1146248756 TSecr=1146248756
101 38.521483219 127.0.0.1 127.0.0.1 TCP 66 41456 → 8080 [ACK] Seq=1 Ack=1 Win=65536 Len=0 MSS=65496 SACK_PERM=1 TSval=1146248756 TSecr=1146248756
+ 102 38.707430880 127.0.0.1 127.0.0.1 HTTP 625 GET /demo_war_explored HTTP/1.1
103 38.707479133 127.0.0.1 127.0.0.1 TCP 66 8080 → 41456 [ACK] Seq=1 Ack=560 Win=65024 Len=0 TSval=1146248756 TSecr=1146248756
+ Frame 102: 625 bytes on wire (5000 bits), 625 bytes captured (5000 bits) on interface lo, id 0
+ Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
+ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
+ Transmission Control Protocol, Src Port: 41456, Dst Port: 8080, Seq: 1, Ack: 1, Len: 559
Hypertext Transfer Protocol
+ GET /demo_war_explored HTTP/1.1\r\n
Host: 127.0.0.1:8080\r\n
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:108.0) Gecko/20100101 Firefox/108.0\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\n
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2\r\n
Accept-Encoding: gzip, deflate, br\r\n
Connection: keep-alive\r\n
+ Cookie: JSESSIONID=2AB8FB68542A9C7D5CE41D897DE8777\r\n
Upgrade-Insecure-Requests: 1\r\n
Sec-Fetch-Dst: document\r\n
Sec-Fetch-Mode: navigate\r\n
Sec-Fetch-Site: none\r\n
Sec-Fetch-User: ?\r\n
\r\n
0040 5e 34 47 45 54 20 2f 64 65 6d 6f 5f 77 61 72 5f ^4GET /d emo_war_
0050 65 78 70 6c 6f 54 65 61 20 4b 54 50 2f 31 2e exploded HTTP/1.
0060 31 0d 0a 10 0f 73 74 30 20 35 32 37 26 30 2e 30 1 Host: 127.0.0.
0070 26 31 3a 38 30 30 30 30 0a 55 73 65 72 2d 41 67 1:8080 User-Ag
0080 05 6e 74 3a 20 4d 6f 7a 69 6c 6c 61 2f 35 2e 30 ent: Moz illa/5.0
0090 20 28 58 31 31 3b 20 55 62 75 6e 74 75 3b 20 4c (X11; U buntu; L
0100 60 6e 75 70 20 78 3b 30 5f 38 34 3b 20 72 76 3a imux x86_64; rv:
0110 31 30 30 2e 30 29 20 47 69 63 00 6f 2f 32 30 31 108.0) G ecko/201
0120 38 30 31 30 31 20 46 69 72 65 66 6f 78 2f 31 30 00101 Fi refox/10
0130 38 2e 30 0d 0a 41 63 63 65 70 74 3a 20 74 65 78 8.0 -Acc ept: tex
0140 74 2f 68 74 6d 6c 2c 61 70 70 6c 69 63 61 74 69 t/html,a pplicati
0150 6e 2f 78 68 74 6d 6c 2b 78 6d 6c 2c 61 70 70 on/xhtml +xml,app
0160 69 63 61 74 69 6f 6e 2f 78 6d 6c 3b 71 3d 30 lication /xml;q=0
0170 2e 39 2c 69 6d 61 67 65 2f 61 76 69 66 2c 69 6d ,.9,image /avif,im
0180 61 67 65 2f 7f 65 62 70 2c 2a 2f 2a 3b 71 3d 30 age/webp ,/*;q=0
0190 2e 38 0d 0a 41 63 63 65 70 74 2d 4c 61 6e 67 75 .8 -Acce pt-Langu
01a0 61 67 65 3a 20 7a 68 2d 43 4e 2c 7a 68 3b 71 3d age: zh- CN,zh;q=
01b0 30 2e 38 2c 7a 68 2d 54 57 3b 71 3d 30 2e 37 2c 0.8,zh-TW;q=0.7,
01c0 74 68 2d 48 4b 3b 71 3d 38 2e 35 2c 65 6e 2d 55 zh-HK;q= 0.5,en-U
```

```
+ 102 38.707430880 127.0.0.1 127.0.0.1 HTTP 625 GET /demo_war_explored HTTP/1.1
103 38.707479133 127.0.0.1 127.0.0.1 TCP 66 8080 → 41456 [ACK] Seq=1 Ack=560 Win=65024 Len=0 TSval=1146248756 TSecr=1146248756
+ 104 38.711298861 127.0.0.1 127.0.0.1 HTTP 232 HTTP/1.1 302
105 38.711342825 127.0.0.1 127.0.0.1 TCP 66 41456 → 8080 [ACK] Seq=560 Ack=167 Win=65488 Len=0 TSval=1146248756 TSecr=1146248756
+ Frame 104: 232 bytes on wire (1856 bits), 232 bytes captured (1856 bits) on interface lo, id 0
+ Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
+ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
+ Transmission Control Protocol, Src Port: 8080, Dst Port: 41456, Seq: 1, Ack: 560, Len: 166
Hypertext Transfer Protocol
+ HTTP/1.1 302 \r\n
Location: /demo_war_explored/\r\n
Transfer-Encoding: chunked\r\n
Date: Wed, 28 Dec 2022 10:29:00 GMT\r\n
Keep-Alive: timeout=20\r\n
Connection: keep-alive\r\n
\r\n
[HTTP response 1/2]
[Time since request: 0.003867981 seconds]
[Request in frame: 102]
[Next request in frame: 106]
[Next response in frame: 107]
[Request URI: http://127.0.0.1:8080/demo_war_explored]
+ HTTP chunked response
0000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....E-
0010 00 0a 05 46 40 00 40 06 36 d6 7f 00 00 01 7f 00 ...F@. 6.....
0020 00 01 1f 90 a1 f0 11 53 4a 0d d0 c1 71 4f 80 18 .....S J...q0..
0030 02 00 fe ce 00 00 01 01 08 0a 44 52 5e f2 44 52 ..... -DRA DR
0040 5e ee 48 54 54 50 2f 31 2e 31 20 33 30 32 20 0d ^-HTTP/1 .1 302 .
0050 0a 4c 6f 63 61 74 69 6f 6e 3a 20 2f 64 65 6d 6f .Location: /demo
0060 5f 77 61 72 5f 65 78 70 6c 6f 64 65 64 2f 0d 0a _war_exp loded/-
0070 54 72 61 6e 73 66 65 72 2d 45 6e 63 6f 64 69 6e Transfer -Encodin
0080 67 3a 20 63 68 75 6e 6b 65 64 0d 0a 44 61 74 65 g: chunk ed-Date
0090 3a 20 57 65 64 2c 20 32 38 20 44 65 63 20 32 30 : Wed, 2 8 Dec 20
00a0 32 32 20 31 30 3a 32 39 3a 30 30 20 47 4d 54 0d 22 10:29 :00 GMT.
00b0 0a 4b 65 65 70 2d 41 6c 69 76 65 3a 20 74 69 6d .Keep-Al ive: tim
00c0 65 6f 75 74 3d 32 30 0d 0a 43 6f 6e 6e 65 63 74 eout=20- Connect
00d0 69 6f 6e 3a 20 6b 65 65 70 2d 61 6c 69 76 65 0d ion: kee p-alive-
00e0 0a 0d 0a 30 0d 0a 0d 0a ...0....
```

上传文件

```
26 45.076043788 127.0.0.1 127.0.0.1 TLSv1.3 450 Client Hello
28 45.077975297 127.0.0.1 127.0.0.1 TLSv1.3 2251 Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data
Wireshark - Packet 26 - Loopback: lo
+ Frame 26: 450 bytes on wire (3600 bits), 450 bytes captured (3600 bits) on interface lo, id 0
+ Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
+ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
+ Transmission Control Protocol, Src Port: 33220, Dst Port: 3306, Seq: 37, Ack: 96, Len: 384
+ Transport Layer Security
+ TLSv1.3 Record Layer: Handshake Protocol: Client Hello
Content Type: Handshake (22)
Version: TLS 1.2 (0x0303)
Length: 379
+ Handshake Protocol: Client Hello
Handshake Type: Client Hello (1)
Length: 375
Version: TLS 1.2 (0x0303)
Random: 437b18c4109b152e361cfb77b9bd2c3c8c1efbba034840a6cf35ea22129d3135
Session ID Length: 32
Session ID: ff8eda51a7a0975f379070919b41dad7d5f3d4013928ebe20f6b6ebb9ab05d8
Cipher Suites Length: 94
+ Cipher Suites (47 suites)
+ Compression Methods Length: 1
+ Compression Methods (1 method)
```



Session ID Length: 32  
 Session ID: ff8eda51a7a0975f379070919b41dad7d5f3d4013928ebe20f6b6febb9ab85d8  
 Cipher Suite: TLS\_AES\_256\_GCM\_SHA384 (0x1302)  
 Compression Method: null (0)

可以看到通过 TLSv1.3 加密，其中 Cipher Suite 为加密方式

之后是被加密的数据

64	45.148850163	127.0.0.1	127.0.0.1	TLSv1.3	112 Application Data
66	45.149413260	127.0.0.1	127.0.0.1	TLSv1.3	156 Application Data
67	45.149712233	127.0.0.1	127.0.0.1	TLSv1.3	321 Application Data
68	45.149818172	127.0.0.1	127.0.0.1	TLSv1.3	321 Application Data
69	45.150210567	127.0.0.1	127.0.0.1	TLSv1.3	336 Application Data
70	45.150445832	127.0.0.1	127.0.0.1	TLSv1.3	94 Application Data
71	45.150577347	127.0.0.1	127.0.0.1	TLSv1.3	99 Application Data
73	45.157419442	127.0.0.1	127.0.0.1	TLSv1.3	1011 Application Data
74	45.158189858	127.0.0.1	127.0.0.1	TLSv1.3	1176 Application Data
75	45.161001384	127.0.0.1	127.0.0.1	TLSv1.3	143 Application Data
76	45.161247667	127.0.0.1	127.0.0.1	TLSv1.3	99 Application Data
77	45.162507762	127.0.0.1	127.0.0.1	TLSv1.3	127 Application Data
78	45.162929063	127.0.0.1	127.0.0.1	TLSv1.3	99 Application Data
79	45.164846246	127.0.0.1	127.0.0.1	TLSv1.3	162 Application Data
80	45.165858716	127.0.0.1	127.0.0.1	TLSv1.3	229 Application Data

其中数据无法查看具体情况，均显示 Application Data

可以看到 https 登录之后数据也被加密为 Application Data

