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/

1.1 小组成员及分工

学号	姓名	分工
[redacted]		组长 软件的开发，文档撰写，CSRF
[redacted]		xss 入侵，CSRF
[redacted]		软件 ui 美化，文档撰写
[redacted]		抓包测试分析，软件测试
[redacted]		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

⑴硬件环境

- windows服务器：
- linuxs服务器：

⑵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.jps	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

⑴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)。

⑵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。

⑶色系

- 主题色调：蓝、白、黑。
- 嵌入色调：灰、蓝。

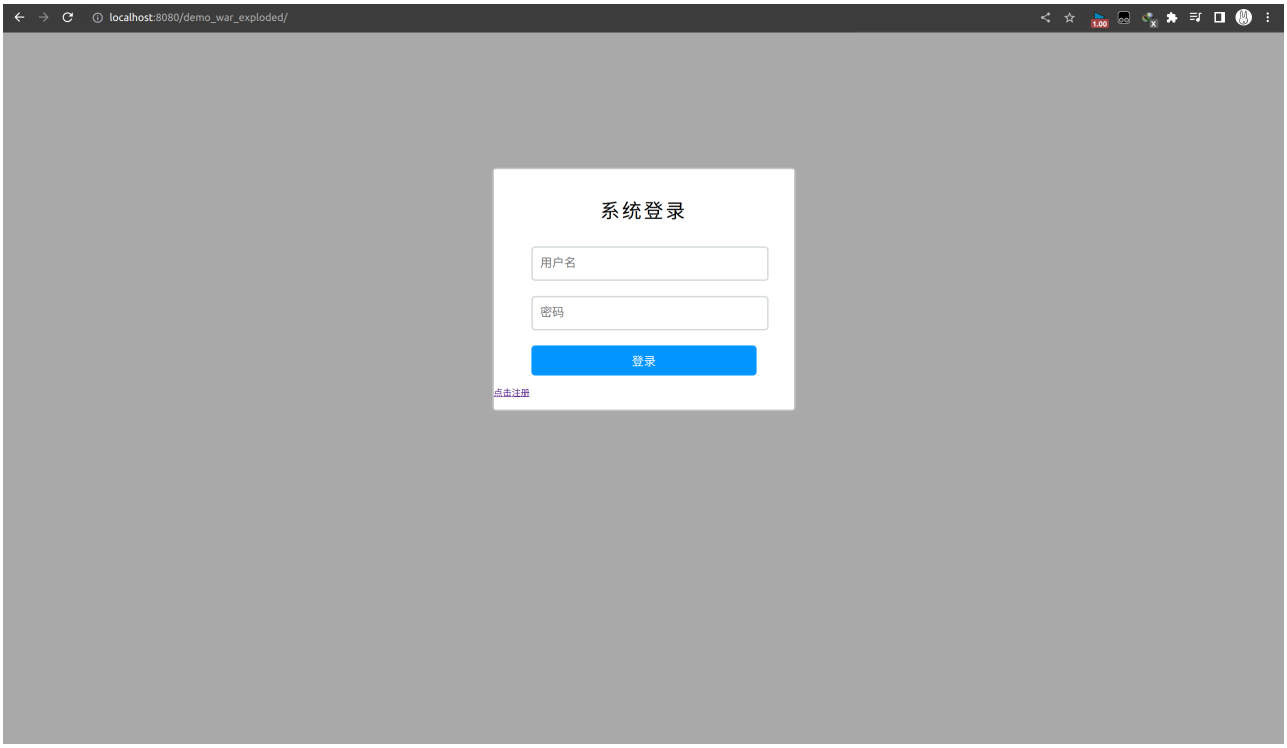
⑷控件

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(commons

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)
```

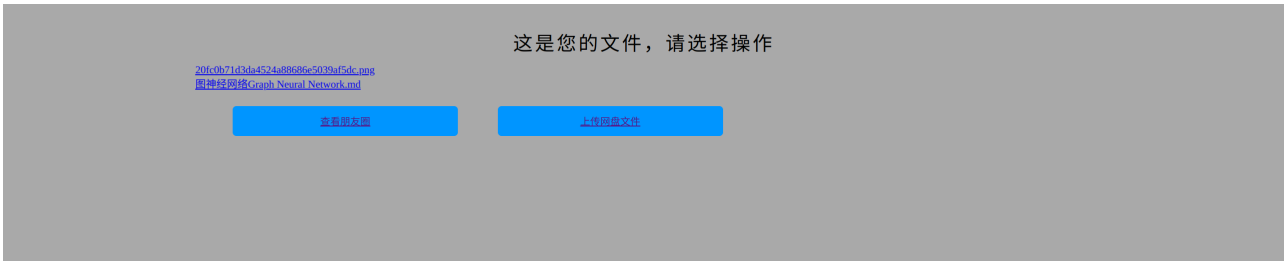
5.2 业务功能及使用方法图解

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。

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。

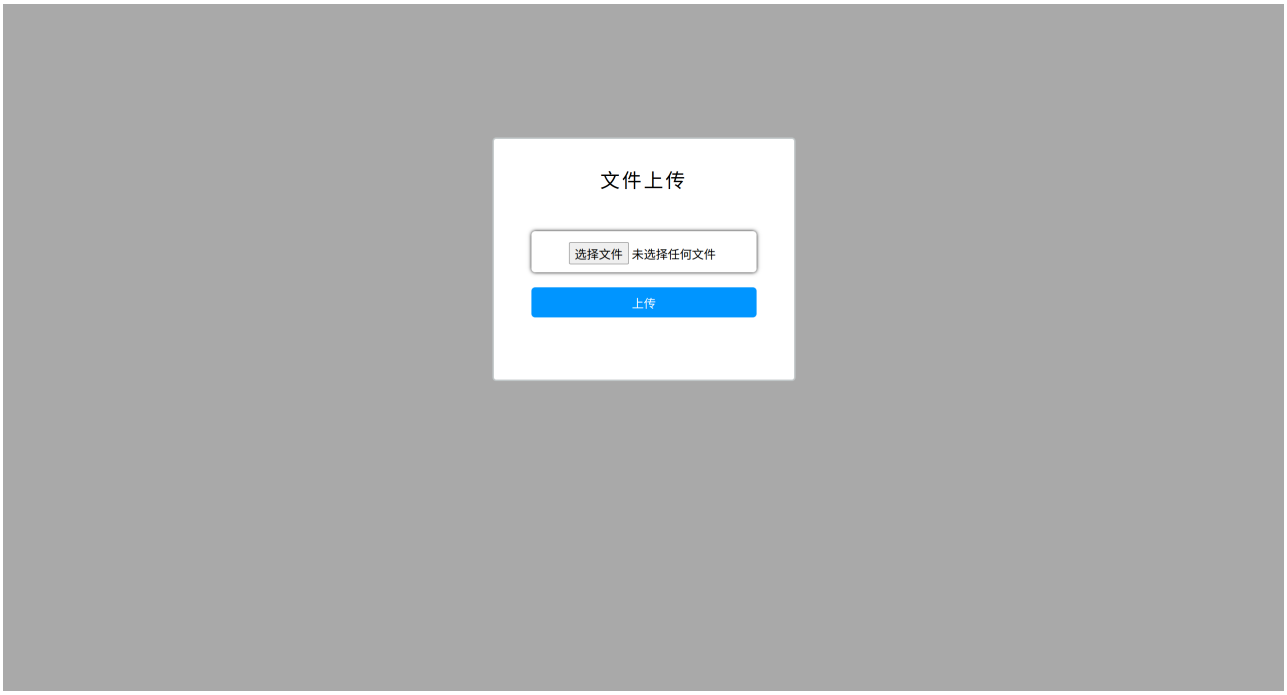


3.查看朋友圈

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
   response) throws IOException {
19         response.setContentType("text/html");
20         String name = request.getParameter("name");
21         String password = request.getParameter("password");
```

```
        HttpSession session = request.getSession();
        // Hello
        PrintWriter out = response.getWriter();
        String USER = "rytter";
        String PASS = "rytter";
        Connection conn = null;
        Statement stmt = null;
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            conn = DriverManager.getConnection(DB_URL, USER, PASS);
            // 执行查询
            stmt = conn.createStatement();
            String sql;
            sql = "SELECT name,password FROM users";
            ResultSet rs = stmt.executeQuery(sql);
            boolean exits = false;
            // 展开结果集数据库
            while (rs.next()) {
                // 通过字段检索
                String name_sql = rs.getString("name");
                String password_sql = rs.getString("password");
                if (Objects.equals(name_sql, name) &
Objects.equals(password_sql, password)) {
                    exits = true;
                    break;
                }
            }
            if (!exits) {
                out.println("<h1>没有看到你的名字，请进行注册</h1>");
                out.println("<a href=\"SignUp.jsp\">点击注册</a>");
                return;
            }
            else {
                out.println("<h1>欢迎登录小废物做的网盘</h1>");
                // 添加session属性
                session.setAttribute("username", name);
                response.sendRedirect("/demo_war_exploded/main");
            }
            // 完成后关闭
            rs.close();
            stmt.close();
            conn.close();
        }
        catch (ClassNotFoundException e) {
            out.println("<h1>" + e + "</h1>");
            throw new RuntimeException(e);
        }
```

```
        catch (SQLException e) {
            out.println("<h1>" + e + "</h1>");
            throw new RuntimeException(e);
        }
    }
}
```

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

```
package com.example.demo;

import javax.servlet.annotation.*;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.*;
import java.util.Objects;

@WebServlet(name = "signup", value = "/signup")
public class SignUp extends HttpServlet {
    static final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://localhost:3306/mysql_java";

    public void doGet(HttpServletRequest request, HttpServletResponse
response) throws IOException {
        String name = request.getParameter("name");
        String password = request.getParameter("password");
        String confirm_code = request.getParameter("confirm_code");
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        Connection conn = null;
        Statement stmt = null;
        String USER = "rytter";
        String PASS = "rytter";
        String confirm_code_judge = "rytter";
        if (Objects.equals(confirm_code, confirm_code_judge)) {
            try {
                Class.forName(JDBC_DRIVER);
                System.out.println("连接数据库...");
                conn = DriverManager.getConnection(DB_URL, USER, PASS);
                // 执行查询
                System.out.println(" 实例化Statement对象...");
                stmt = conn.createStatement();
                String sql;
```

```
                sql = "SELECT name FROM users";
                ResultSet rs = stmt.executeQuery(sql);

                // 展开结果集数据库
                while (rs.next()) {
                    // 通过字段检索
                    String name_sql = rs.getString("name");
                    if (Objects.equals(name_sql, name)) {
                        //查询有没有这个名字
                        out.println("<h1>" + "该用戶名存在，请更改用戶名" + "</h1>");
                        return;
                    }
                }
                //如果没有这个名字就添加上
                String sql2 = "INSERT users (name,password) VALUES ('";
                sql2=sql2+name;
                sql2=sql2+ "','";
                sql2=sql2+password;
                sql2=sql2+"')";
                int counts = stmt.executeUpdate(sql2);
                System.out.println("正在查询");
                out.println("<h1>"+counts+"</h1>");
                out.println("<html><body>");
                out.println("<h1>" + "你的名称是：" + name + "<br/>" + "你的密码是：" + password + "</h1>");
                out.println("</body></html>");
                out.println("<a href=\"Login.jsp\">点击登录</a>");
                // 完成后关闭
                rs.close();
                stmt.close();
                conn.close();
            }
            catch (ClassNotFoundException e) {
                out.println("<h1>" + e.getMessage() + "</h1>");
                System.out.println(e.getMessage());
                throw new RuntimeException(e);
            }
            catch (SQLException e) {
                out.println("<h1>" + e.getMessage() + "</h1>");
                throw new RuntimeException(e);
            }
        }
        else {
            out.println("<h1>" + "邀请码错误，请联系管理员" + "</h1>");
        }
    }
```

```
82  }
```

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)
   throws ServletException, IOException {
25         File file = new File("public_text.txt");
26         List<String> list = new ArrayList();
27         if (file.exists()) {
28             BufferedReader reader = null;
29             try {
30                 reader = new BufferedReader(new FileReader(file));
31                 String readStr;
32                 while ((readStr = reader.readLine()) != null) {
33                     list.add(readStr);
34                 }
35                 reader.close();
36             }
37             catch (IOException e) {
38                 e.printStackTrace();
39             }
```

```
            finally {
                if (reader != null) {
                    try {
                        reader.close();
                    }
                    catch (IOException e1) {
                        e1.printStackTrace();
                    }
                }
            }
        }
        else {
            boolean create = file.createNewFile();
            if (create) {
                System.out.println("创建成功");
            }
            else {
                System.out.println("创建失败");
            }
        }
        //到这个地方我们就已经把结果放到list中了
        HttpSession session = req.getSession();
        session.setAttribute("text", list);
        resp.sendRedirect("/demo_war_exploded/Public.jsp");
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        String public_text="";
        req.setCharacterEncoding("utf-8");
        resp.setCharacterEncoding("utf-8");
        resp.setContentType("text/html;charset=utf-8");
        req.setCharacterEncoding("UTF-8");
        resp.setContentType("text/html;charset=utf-8");
        FileItemFactory factory=new DiskFileItemFactory();
        ServletFileUpload upload =new ServletFileUpload(factory);
        try {
            List<FileItem> items=upload.parseRequest(req);//这个地方可能会有问
题
            for (FileItem item:items){
                if(item.getFieldName().equals("public_text")){
                    if(!item.getString().equals("")){
                        public_text= item.getString("UTF-8");
                    }
                }
            }
```

```
        }
        catch (FileUploadException e) {
            throw new RuntimeException(e);
        }

        HttpSession session = req.getSession();
        String username = (String) session.getAttribute("username");
        File file = new File("public_text.txt");
        BufferedWriter output = new BufferedWriter(new FileWriter(file,
true));
        username = "<h2>" + username + "</h2></br>";
        public_text = "<p>" + public_text + "</p></br>";
        output.write(username);
        output.write("\n");
        output.write(public_text);
        output.write("\n");
        output.flush();
        output.close();
        resp.sendRedirect("/demo_war_exploded/main.jsp");
    }
    ;
}
```

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

```
package com.example.demo;

import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.apache.commons.fileupload.*;
import org.apache.commons.fileupload.disk.DiskFileItemFactory;
import org.apache.commons.fileupload.servlet.ServletFileUpload;
import org.apache.commons.io.*;

import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.PrintWriter;
import java.util.List;

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
    filename=item.getName().substring(item.getName().lastIndexOf("\\")+1);
56                      is = item.getInputStream();//获得上传文件的InputStream对象
57                  }
58                  File file=new File(filename);
59                  if(file.exists()){
60                      out.println("文件已经存在,请更换文件名后重新上传");
61                      System.out.println(file.getAbsolutePath());
62                  }
63                  else if(!filename.equals("")){
64                      //使用fileoutputstream打开服务器端的上传文件
```

```
                    FileOutputStream fos=new FileOutputStream(filename);
                    byte[] buffer=new byte[8192];
                    int count=0;
                    while((count=is.read(buffer))>0){
                        fos.write(buffer,0,count);
                    }
                    fos.close();
                    is.close();
                    out.println("文件上传成功");
                    //开始进行数据库操作
                    Class.forName(JDBC_DRIVER);
                    System.out.println("连接数据库...");
                    conn = DriverManager.getConnection(DB_URL, USER, PASS);
                    // 执行查询
                    System.out.println(" 实例化Statement对象...");
                    stmt = conn.createStatement();
                    String sql2 = "INSERT file (username,filename) VALUES
('";
                    sql2=sql2+username;
                    sql2=sql2+ "','";
                    sql2=sql2+filename;
                    sql2=sql2+"')";
                    int counts = stmt.executeUpdate(sql2);
                    System.out.println(counts);
                    System.out.println(filename);
                    System.out.println(username);
                    System.out.println();
                    response.sendRedirect("/demo_war_exploded/main");
                }
            }
        }
        catch (Exception e){
            System.out.println(e.getMessage());
        }
    }
}
```

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

```
package com.example.demo;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.*;
import java.net.URLEncoder;

@WebServlet(name = "downloadservlet", value = "/downloadservlet")
public class DownloadServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        super.doPost(req, resp);
    }
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        //super.doGet(req, resp);
        {
            //检验是不是第一次登录
            HttpSession session=request.getSession();
            String username=(String)session.getAttribute("username");
            if (username==null){
                response.sendRedirect("/demo_war_exploded/Login.jsp");
                return;
            }
        }
        String filename = request.getParameter("filename");
        try{
            File file = new File(filename);
            boolean exists = file.exists();
            if (!exists){
                throw new Exception("文件不存在! ");
            }
            String ext = filename.substring(filename.lastIndexOf(".") +
1).toLowerCase();
            FileInputStream fileInputStream = new FileInputStream(file);
            InputStream fis = new BufferedInputStream(fileInputStream);
            byte[] buffer = new byte[fis.available()];
            fis.read(buffer);
            fis.close();
            response.reset();
            response.setCharacterEncoding("UTF-8");
            response.addHeader("Content-Disposition", "attachment;filename="
+ URLEncoder.encode(filename, "UTF-8"));
            response.addHeader("Content-Length", "" + file.length());
            OutputStream outputStream = new
            BufferedOutputStream(response.getOutputStream());
            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

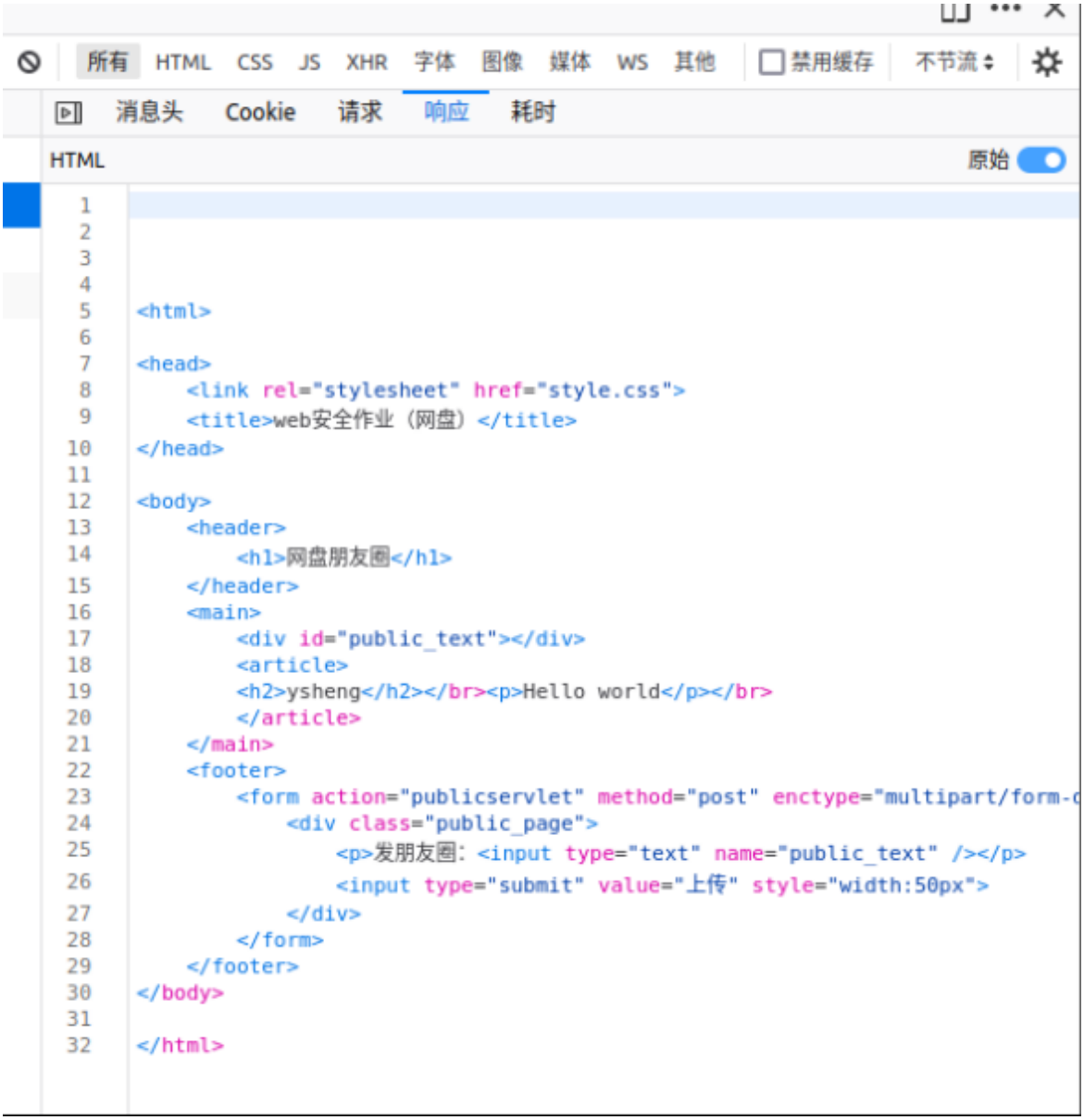
6.1 概述

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

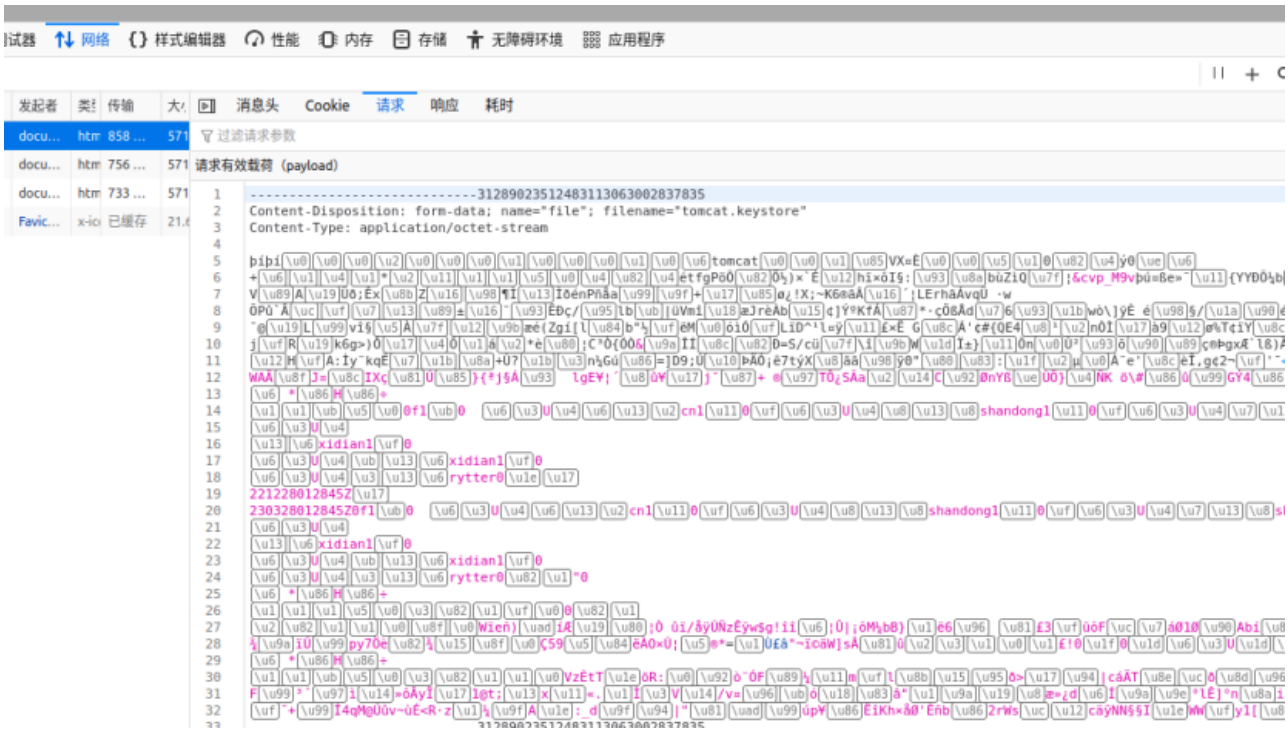
利用浏览器自带进行抓取数据

查看朋友圈的 response，request 为空



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 Fire
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/wel
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
消息头 Cookie 请求 响应 耗时
过滤请求参数
请求有效载荷 (payload)
----------------------------312890235124831130630028378 35
Content-Disposition: form-data; name="file"; filename="tomcat.keystore"
Content-Type: application/octet-stream

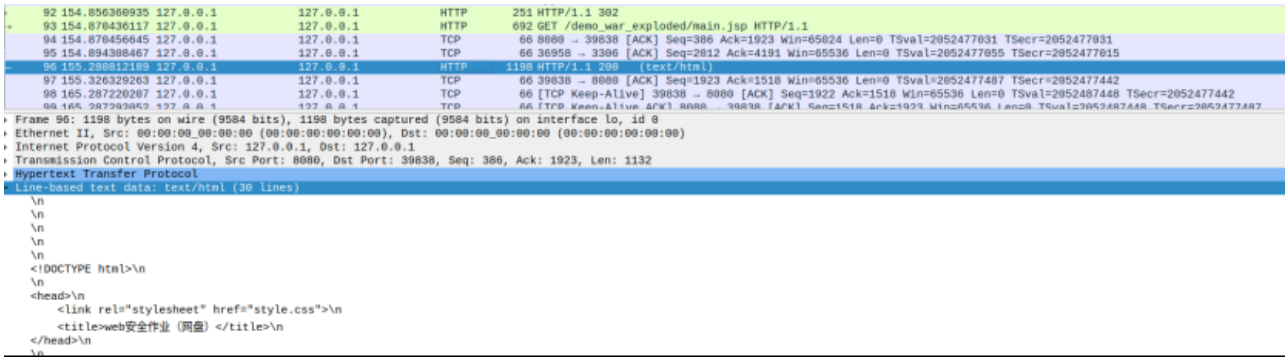
response

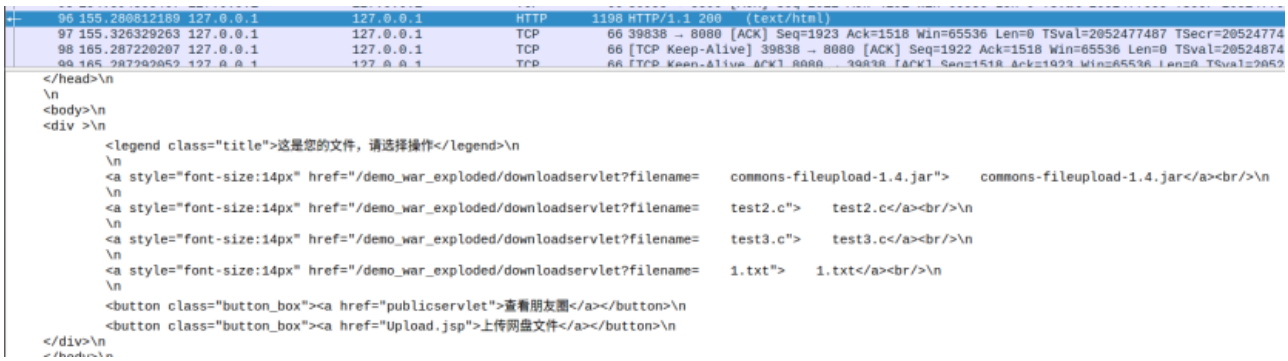
消息头 Cookie 请求 响应 耗时

HTML 原始

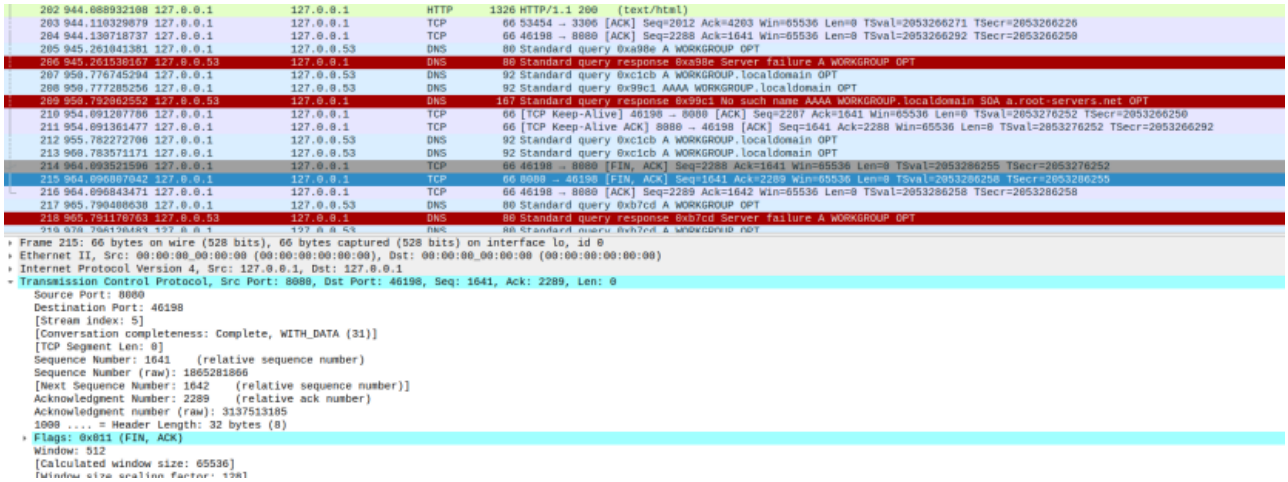
```
1
2
3   <!DOCTYPE html>
4
5   <head>
6       <link rel="stylesheet" href="style.css">
7       <title>web安全作业（网盘）</title>
8   </head>
9
10  <body>
11      <form action="uploadservlet" method="post" enctype="multipart/form-data"
12          <div class="upload-box">
13              <legend class="title">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>
```

用 Wireshark 抓包





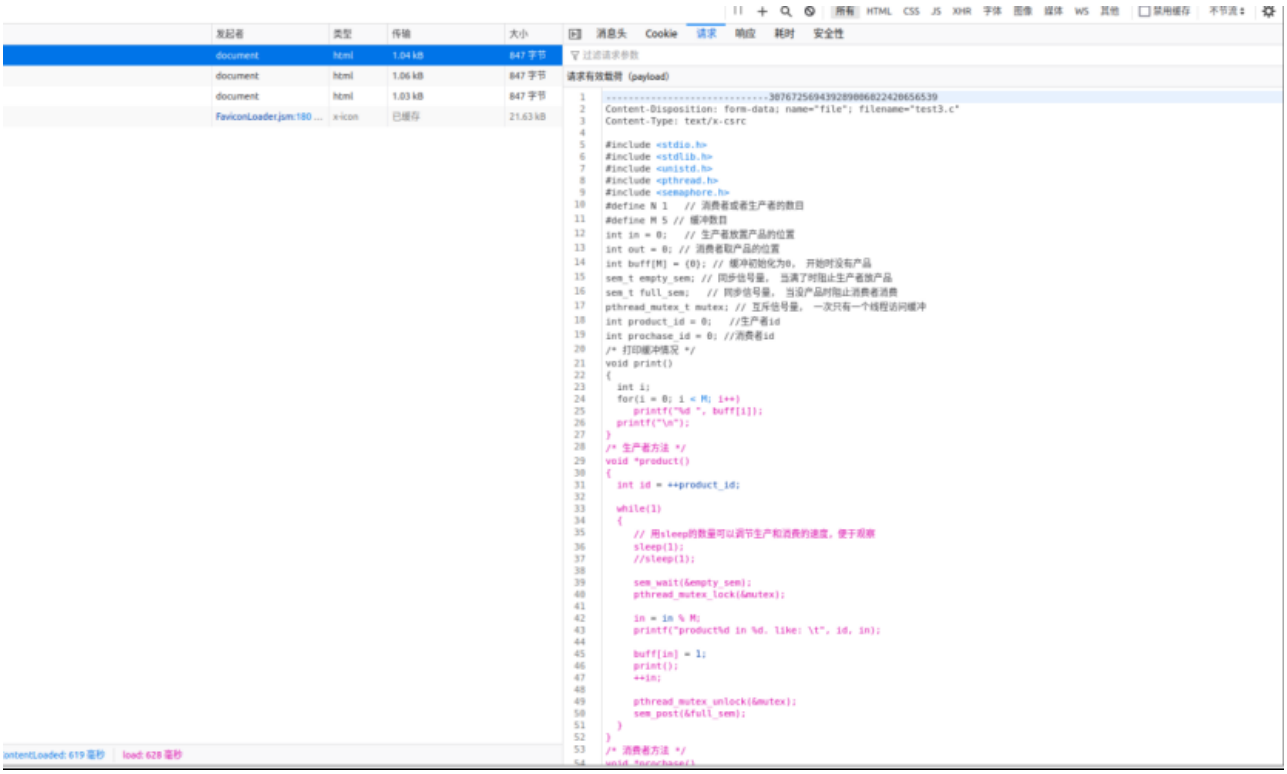
上传文件



使用 http

上传文件

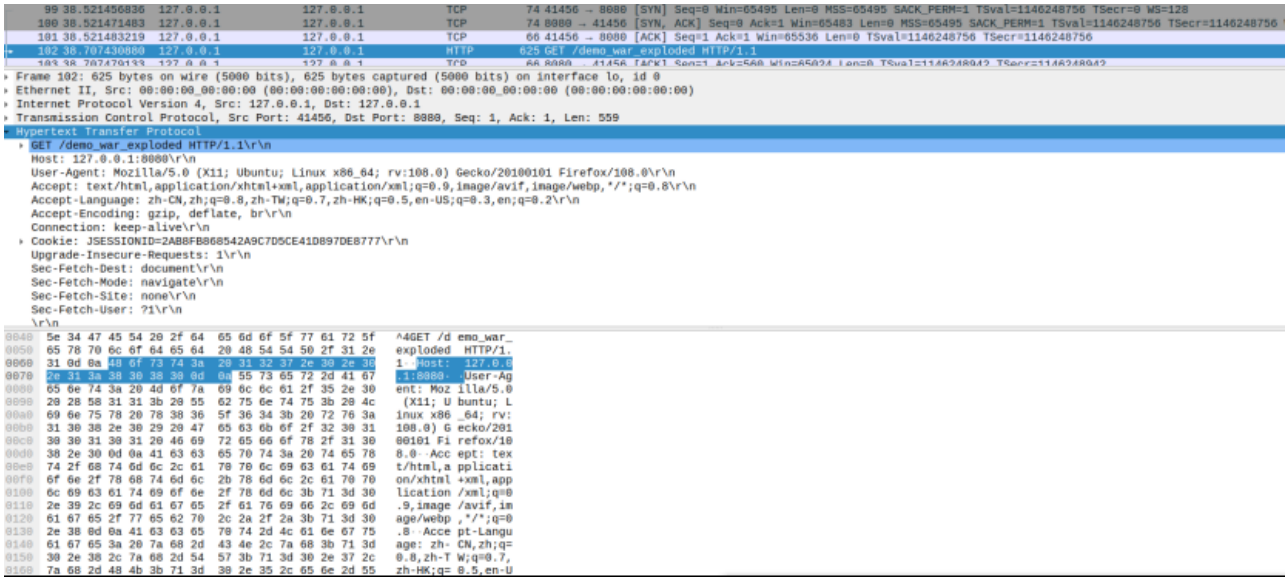
request

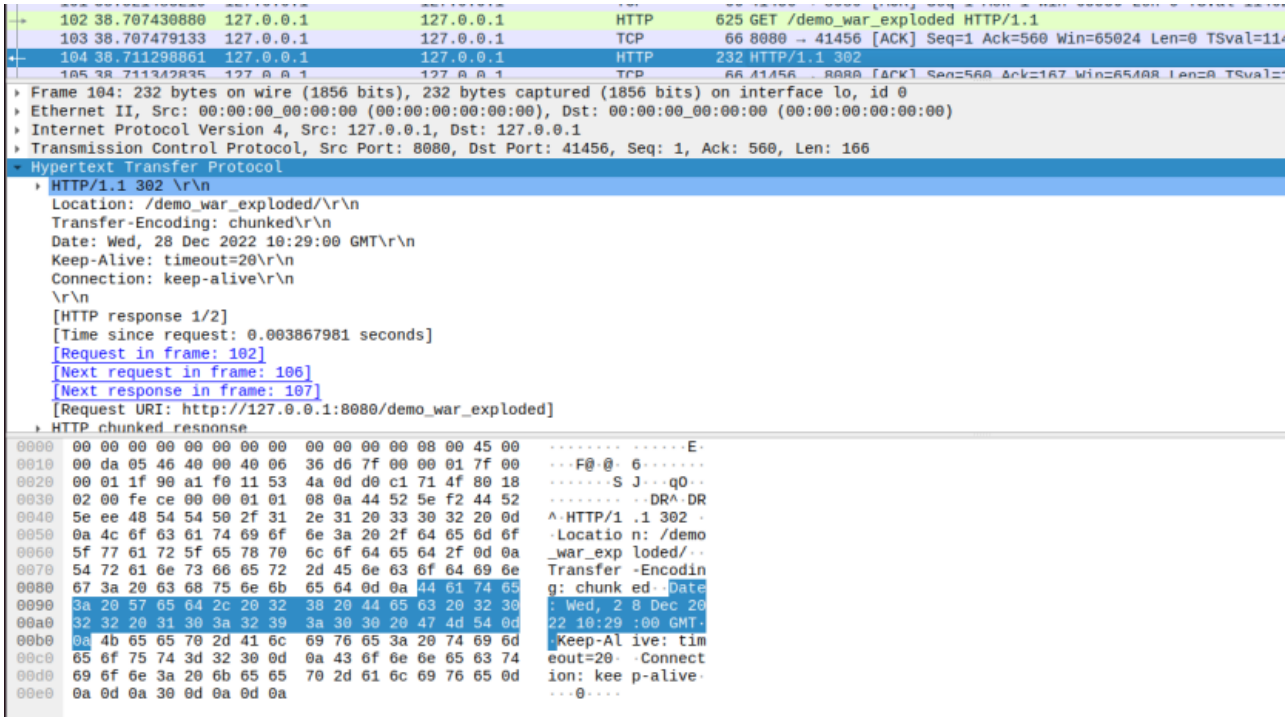


response

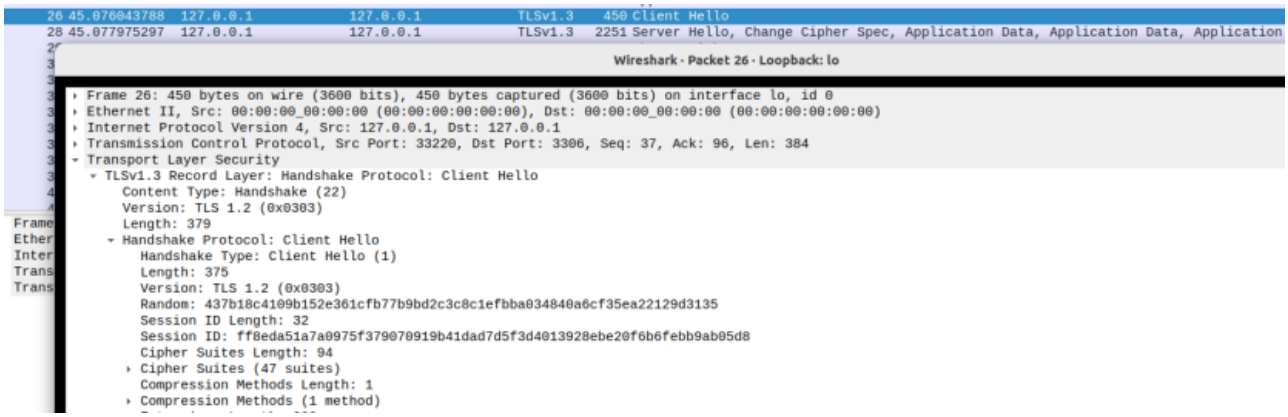
wireshark 抓包

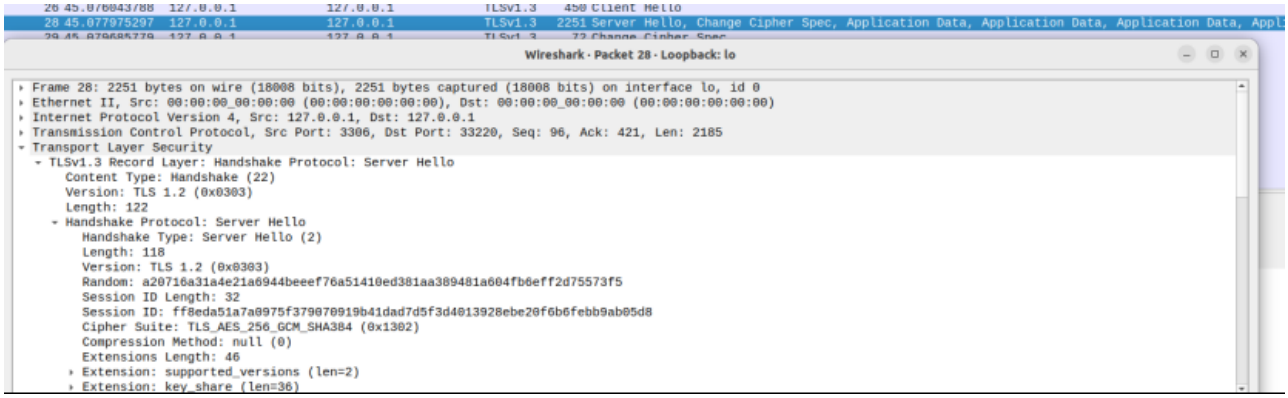
登录 http

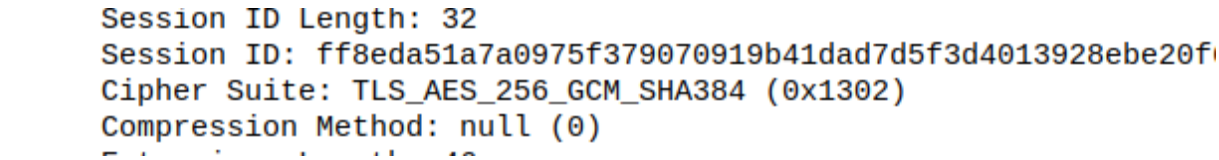

99 38.521456836 127.0.0.1 127.0.0.1 TCP 74 41456 → 8080 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=1146248756 TSecr=0 WS=128
100 38.521471483 127.0.0.1 127.0.0.1 TCP 74 8080 → 41456 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=1146248756 TSecr=1146248756
101 38.521483219 127.0.0.1 127.0.0.1 TCP 66 41456 → 8080 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=1146248756 TSecr=1146248756
102 38.707430880 127.0.0.1 127.0.0.1 HTTP 625 GET /demo_war_exploded HTTP/1.1
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_exploded 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=2AB8FB868542A9C7D5CE41D897DE8777\r\n
Upgrade-Insecure-Requests: 1\r\n
Sec-Fetch-Dest: document\r\n
Sec-Fetch-Mode: navigate\r\n
Sec-Fetch-Site: none\r\n
Sec-Fetch-User: ?1\r\n
\r\n


102 38.707430880 127.0.0.1 127.0.0.1 HTTP 625 GET /demo_war_exploded 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=114
104 38.711298861 127.0.0.1 127.0.0.1 HTTP 232 HTTP/1.1 302
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_exploded/\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_exploded]
HTTP chunked response

上传文件

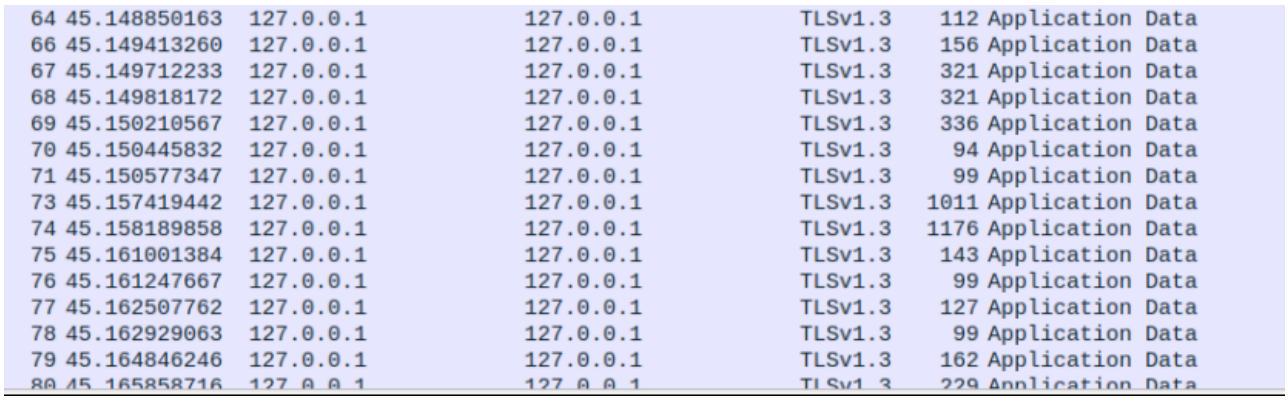

26 45.076043788 127.0.0.1 127.0.0.1 TLSv1.3 450 Client Hello
28 45.077975297 127.0.0.1 127.0.0.1 TLSv1.3 2251 Server Hello, Change Cipher Spec, Application Data, Application Data, Application
Wireshark · Packet 26 · Loopback: lo
Frame 26: 450 bytes on wire (3600 bits), 450 bytes captured (3600 bits) 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: ff8eda51a7a0975f379070919b41dad7d5f3d4013928ebe20f6b6febb9ab05d8
Cipher Suites Length: 94
Cipher Suites (47 suites)
Compression Methods Length: 1
Compression Methods (1 method)





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



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