2021/04/09_shellcode_第4课_metasploit 工具的简单使用

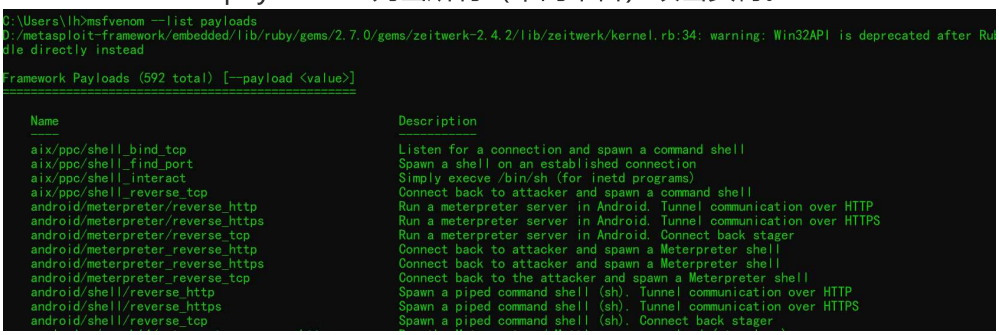
笔记本:	shellcode
创建时间:	2021/4/9 星期五 10:14
作者:	ileemi

metasploit 工具的简单使用

生成shellcode，支持生成多平台的shellcode，同时可以指定其编码格式以及可以排除坏字节（bad byte）。

命令使用示例：

- msfvenom --list payloads：列出所有（不同平台）攻击负荷。

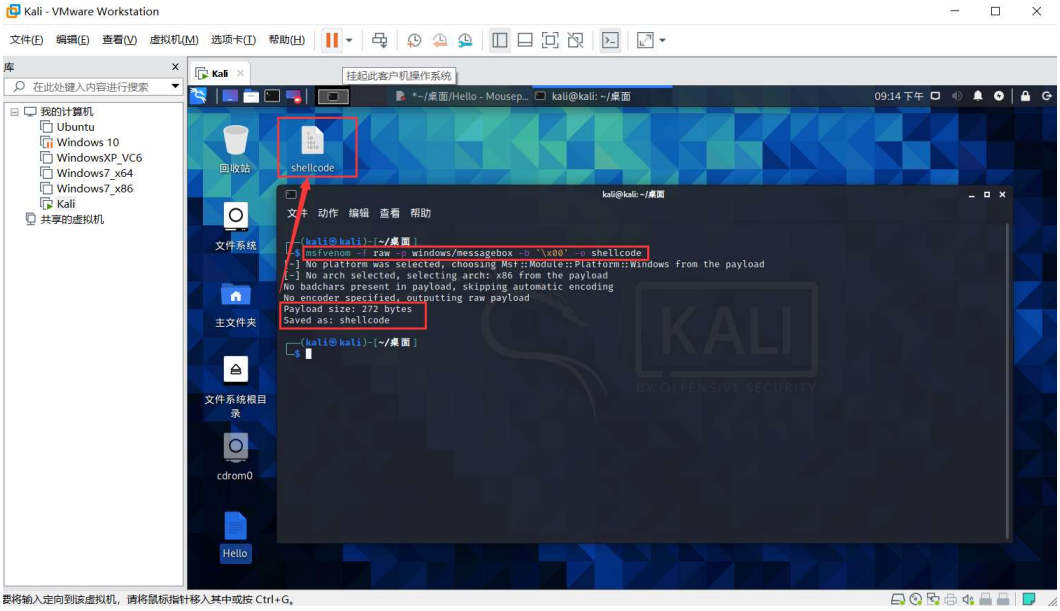


- msfvenom --list encoders：列出所有的编码器。
- msfvenom --list formats：列出所有可用的输出格式，如果没有指定格式,可以使用raw来代替。

使用 "msfvenom" 脚本生成shellcode代码，命令示例如下：

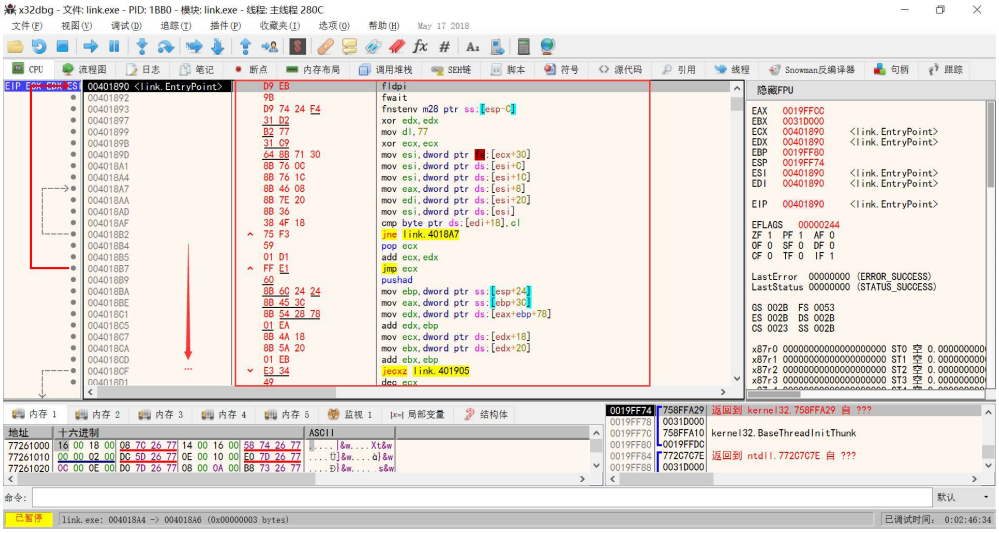
```
msfvenom -f raw -p windows/messagebox -b '\x00' -o shellcode
```

生成shellcode十六进制代码：

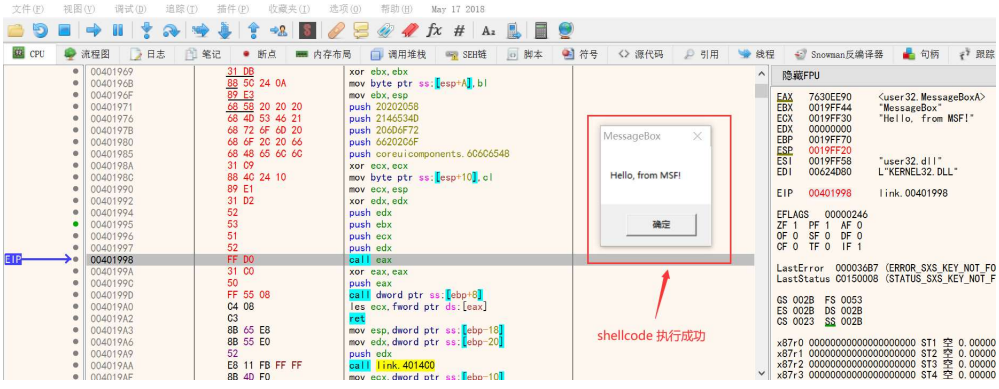


对生成的shellcode进行测试：

- 复制生成的shellcode十六进制代码到任意 ".exe" 中进行测试，如下图所示：



- 跳过单步执行汇编代码，"F9" 直接运行程序，结果如下：



"fnstenv" 指令使用了28字节的结构体，这个结构体在32位保护模式中执行时被用来保存FPU状态到内存中，详细解释在《恶意代码分析实战》的 "19.3.2" 章节，如下图

所示：

19.3.2 使用fnstenv指令

x87浮点单元（FPU）在普通x86架构中提供了一个隔离的执行环境。它包含一个单独的专用寄存器集合，当一个进程正在使用FPU执行浮点运算时，这些寄存器需要由操作系统在上下文切换时保存。代码清单19-2为被`fstenv`指令与`fnstenv`指令使用的28字节结构体，这个结构体在32位保护模式中执行时被用来保存FPU状态到内存中。

代码清单19-2 FpuSaveState结构体的定义

```
struct FpuSaveState {
    uint32_t    control_word;
    uint32_t    status_word;
    uint32_t    tag_word;
    uint32_t    fpu_instruction_pointer;
    uint16_t    fpu_instruction_selector;
    uint16_t    fpu_opcode;
    uint32_t    fpu_operand_pointer;
    uint16_t    fpu_operand_selector;
    uint16_t    reserved;
};
```

这里唯一影响使用的域是在字节偏移量12处的`fpu_instruction_pointer`。它将保留被FPU使用的最后一条CPU指令的地址，并为异常处理器标识哪条FPU指令可能导致错误上下文信息。需要这个域是因为FPU是与CPU并行运行的。如果FPU产生了一个异常，异常处理器不能简单地通过参照中断返回地址来标识导致这个错误的指令。

代码清单19-3为另外一个使用`fnstenv`获取EIP的Hello World程序反汇编代码。

代码清单19-3 使用fnstenv的Hello World例子

```
Bytes           Disassembly
83 EC 20        sub     esp, 20h
31 D2           xor     edx, edx
EB 15           jmp     short loc_1C
EA 07 45 7E     dd 7E4507EAh            ; MessageBoxA
FA CA 81 7C     dd 7C81CAFAh            ; ExitProcess
48 65 6C 6C 6F  db 'Hello World!',0
20 57 6F 72 6C
64 21 00

loc_1C:

D9 EE           fldz ❶
D9 74 24 F4     fnstenv byte ptr [esp-0Ch] ❷
5B              pop     ebx ❸           ; ebx points to fldz
```