Hymmnos-Rootkit 说明文档

软件安全原理第三小组：何俊霖 简鲲鹏 李海鸣 李畅 姜楠 郭娟娟 高茜 姜昭雪 冷斯远 贾梓健

什么是 Hymmnos-Rootkit

Hymmnos-Rootkit 是一个以 C 语言写成的，可以被加载进 Linux 内核的 Rootkit 模块。通过 Hymmnos-Rootkit 可以实现如进程隐藏、权限获取、数据包监控等功能。

构建环境

GNU Make 4.1

Linux 4.15.0-38-generic

功能列表

- 隐藏 Rootkit 自身
  - 隐藏 Rootkit 文件夹
  - 从模块加载列表中隐藏
- 隐藏特定进程
  - 通过开关字符串隐藏包含改字符串的命令
  - 通过 Kill 命令给进程发送隐藏信号
  - 隐藏该被隐藏进程的TCP连接信息
- 隐藏特定文件和内容
  - 通过开关字符串隐藏以该字符串作为后缀的文件
  - 隐藏文件内容中特定标签文本
- 监控数据
  - 记录http请求
  - 记录可能包含密码的内容
  - 搜集包括特定内容的数据包并记录
- 监控模块加载信息
- 获取 root 权限

编译及安装

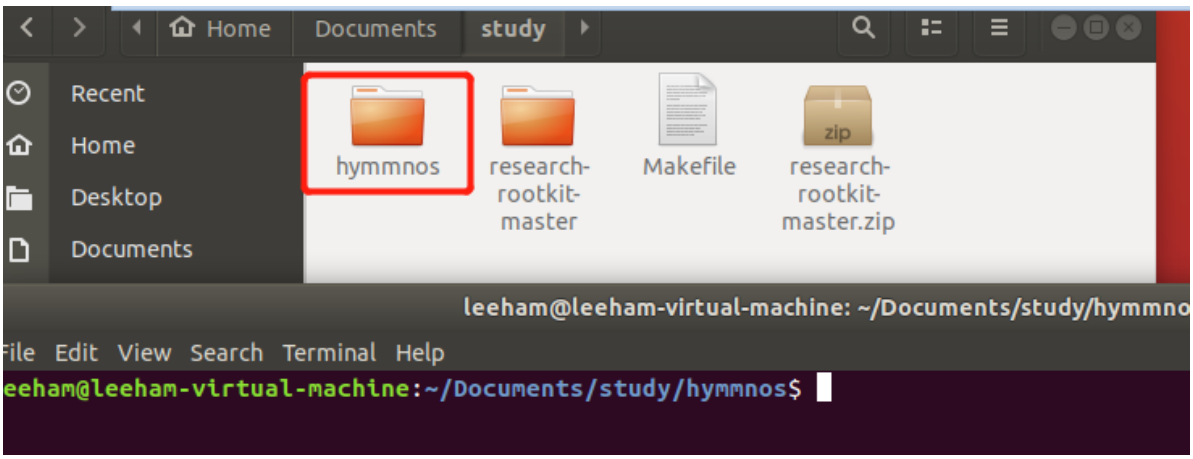
```
$ cd ~
$ git clone https://github.com/flysoar/Hymmnos-rootkit.git
$ cd Hymmnos-rootkit
$ make
$ sudo insmod hymmnos.ko
```

配置项

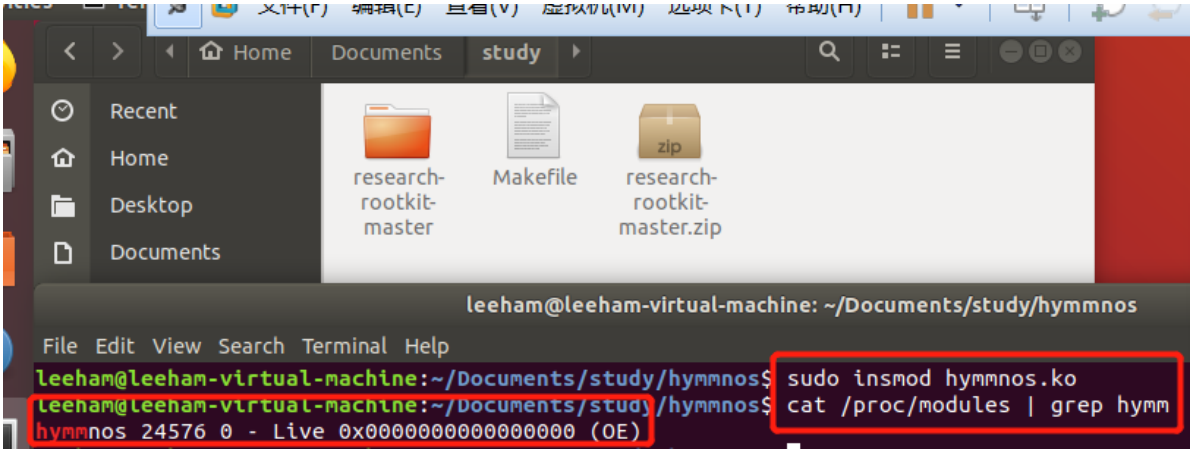
选项	默认值	说明
`SIGROOT`	48	获得 root 权限信号
`SIGHIDEPROC`	49	隐藏/显示指定进程信号
`SIGHIDEHYMMNOS`	50	隐藏/显示本模块信号
`SIGHIDECONTENT`	51	开启/关闭特定文件内容隐藏信号
`SIGBACKDOOR`	52	开启/关闭网络后门信号
`SIGKOMON`	53	阻止新模块加载
`FILE_SUFFIX`	.reyvateil	隐藏文件后缀
`COMMAND_CONTAINS`	ceil	隐藏进程开关字符串
`ROOTKIT_NAME`	hymmnos	Rootkit 名称
`HIDETAGIN`	<touwaka>	文件内容隐藏开始标签
`HIDETAGOUT`	</touwaka>	文件内容隐藏结束标签
`SHELL`	/home/flysoar/test	后门执行的程序路径
`TCPPORT`	7777	后门 TCP 端口
`UDPPORT`	7777	后门 UDP 端口
`TOKEN`	tonelico	后门 Token
`WORKNAME`	ceil	执行后门的内核工作线程名

测试

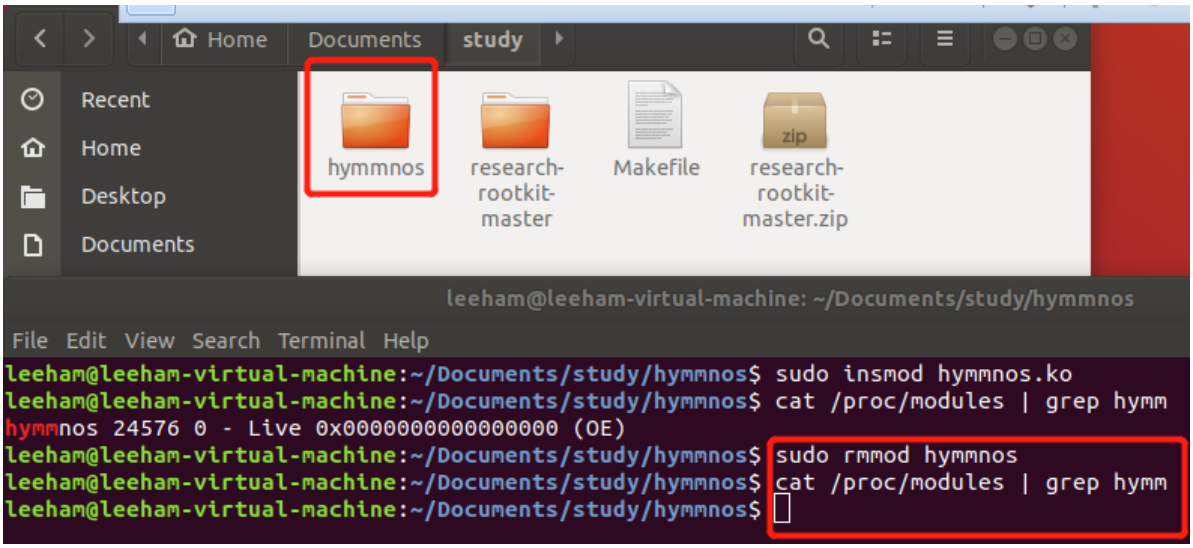
- 隐藏 Rootkit 文件夹
    1. 模块安装之前；文件夹存在



2. 模块安装之后；文件夹隐藏

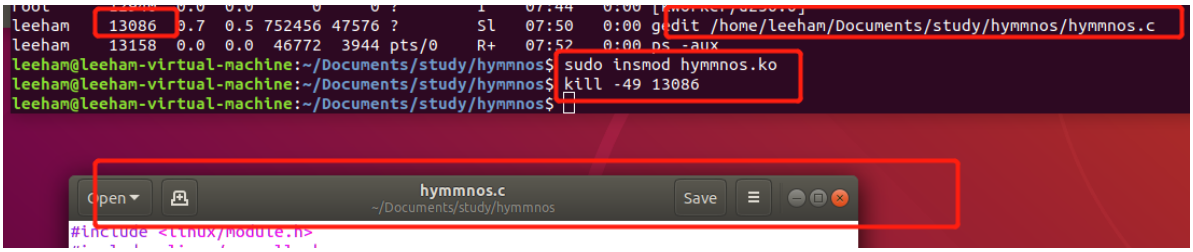


3. 模块卸载时候；文件夹存在

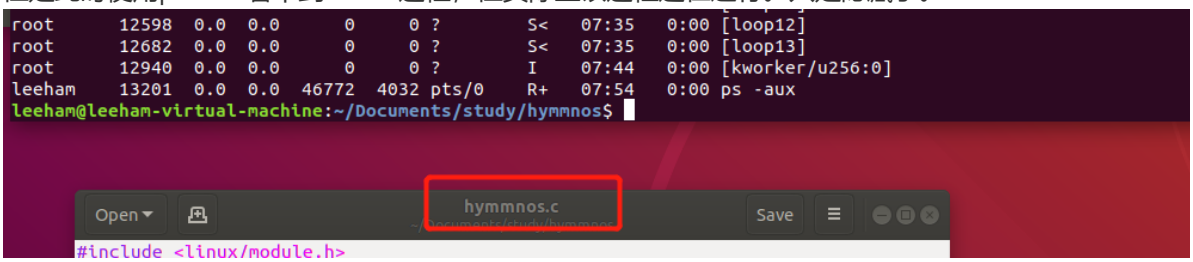


- 隐藏进程

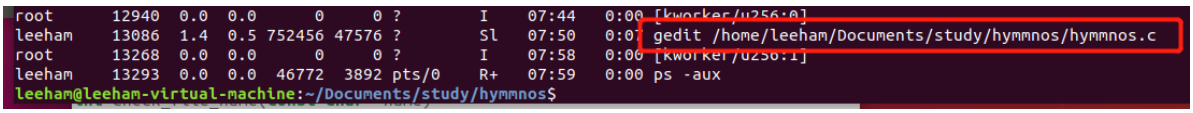
1. 安装模块之前ps -aux；可以看到13086进程（这个是gedit hymmnos.c进程，图中可以看到）。安装模块之后可以看到该进程还存在



2. 但是此时使用ps -aux看不到13086进程；但实际上该进程还在运行。只是隐藏了。



3. 卸载模块之后。



- 隐藏模块

测试kill -50的功能；将模块隐藏或显示；隐藏时显示；显示时隐藏

```
File  Edit  View  Search  Terminal  Help
leeham@leeham-virtual-machine:~/Documents/study/hymmnos$ sudo insmod hymmnos.ko
leeham@leeham-virtual-machine:~/Documents/study/hymmnos$ cat /proc/modules | grep hy
hymmnos 24576 0 - Live 0x0000000000000000 (OE)
leeham@leeham-virtual-machine:~/Documents/study/hymmnos$ kill -50 7
leeham@leeham-virtual-machine:~/Documents/study/hymmnos$ cat /proc/modules | grep hy
leeham@leeham-virtual-machine:~/Documents/study/hymmnos$ sudo rmmod hymmnos.ko
rmmod: ERROR: ../libkmod/libkmod-module.c:793 kmod_module_remove_module() could not remove 'hymmnos': No such file or dir
ory
rmmod: ERROR: could not remove module hymmnos.ko: No such file or directory
leeham@leeham-virtual-machine:~/Documents/study/hymmnos$ kill -50 7
leeham@leeham-virtual-machine:~/Documents/study/hymmnos$ cat /proc/modules | grep hy
hymmnos 24576 0 - Live 0x0000000000000000 (OE)
leeham@leeham-virtual-machine:~/Documents/study/hymmnos$ sudo rmmod hymmnos.ko
leeham@leeham-virtual-machine:~/Documents/study/hymmnos$
```

- 获取权限

  测试kill -48 ;获取root权限；使用whoami查看是否测试成功

```
leeham@leeham-virtual-machine: ~/Documents/study/hymmnos
File  Edit  View  Search  Terminal  Help
leeham@leeham-virtual-machine:~/Documents/study/hymmnos$ sudo insmod hymmnos.ko
leeham@leeham-virtual-machine:~/Documents/study/hymmnos$ cat /proc/modules | gre
p hy
hymmnos 24576 0 - Live 0xffffffffc0503000 (OE)
leeham@leeham-virtual-machine:~/Documents/study/hymmnos$ kill -48 -3
leeham@leeham-virtual-machine:~/Documents/study/hymmnos$ whoami
root
leeham@leeham-virtual-machine:~/Documents/study/hymmnos$ sudo rmmod hymmnos
leeham@leeham-virtual-machine:~/Documents/study/hymmnos$ kill -48 -3
bash: kill: (-3) - No such process
leeham@leeham-virtual-machine:~/Documents/study/hymmnos$ kill -48 -3
```

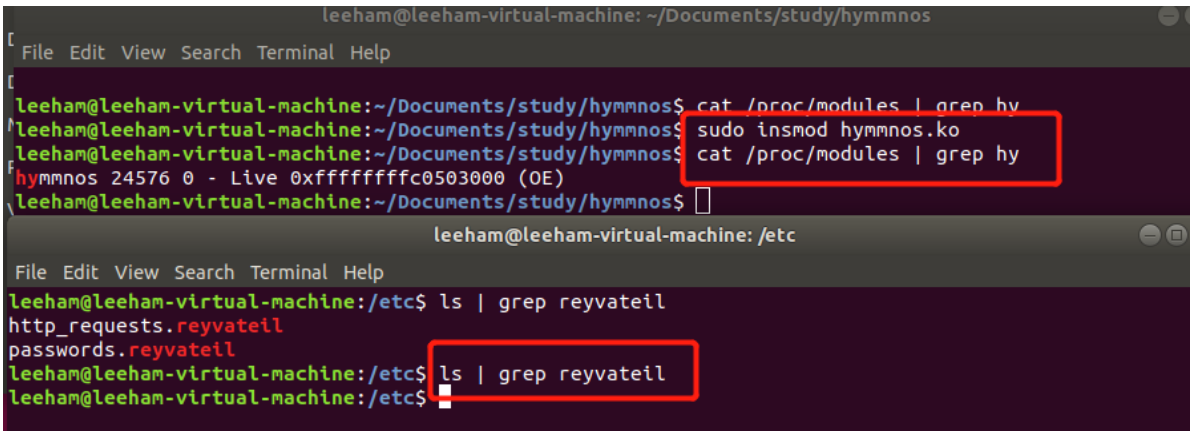
- 隐藏特定文件

  1. 根据配置我们隐藏的是后缀名为 reyvateil 的文件
  2. 未加载模块时的文件存在

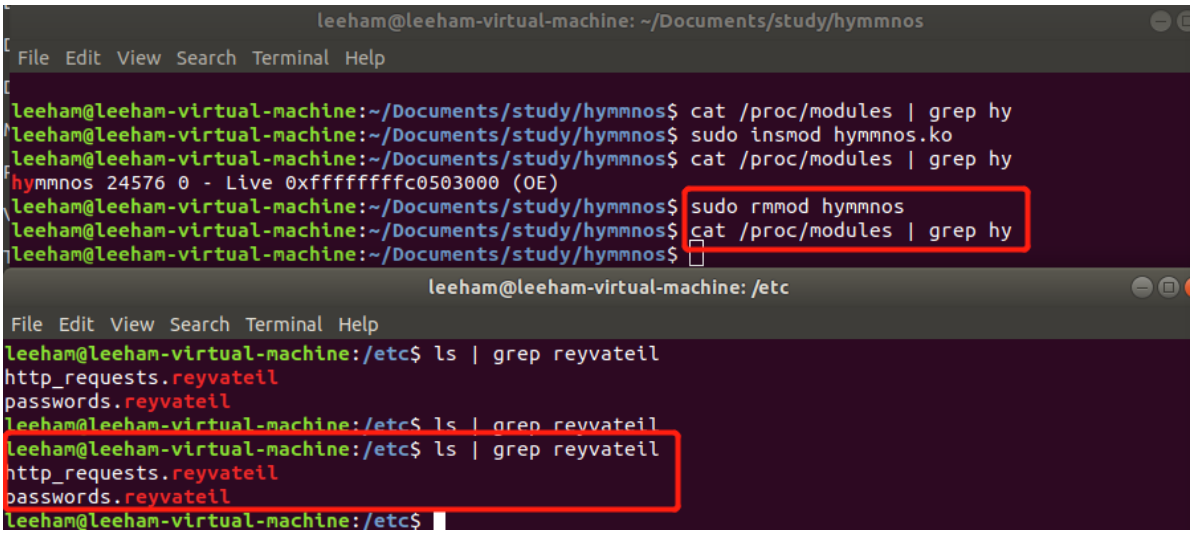
```
leeham@leeham-virtual-machine: ~/Documents/study/hymmnos
File  Edit  View  Search  Terminal  Help
leeham@leeham-virtual-machine:~/Documents/study/hymmnos$ cat /proc/modules | grep hy
leeham@leeham-virtual-machine:~/Documents/study/hymmnos$

leeham@leeham-virtual-machine: /etc
File  Edit  View  Search  Terminal  Help
leeham@leeham-virtual-machine:/etc$ ls | grep reyvateil
http_requests.reyvateil
passwords.reyvateil
leeham@leeham-virtual-machine:/etc$
```

3. 加载模块之后，文件被隐藏

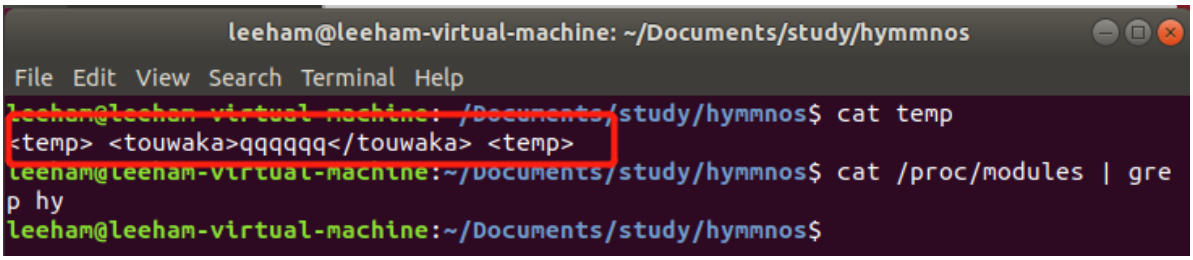


4. 卸载模块之后，隐藏的文件被显示

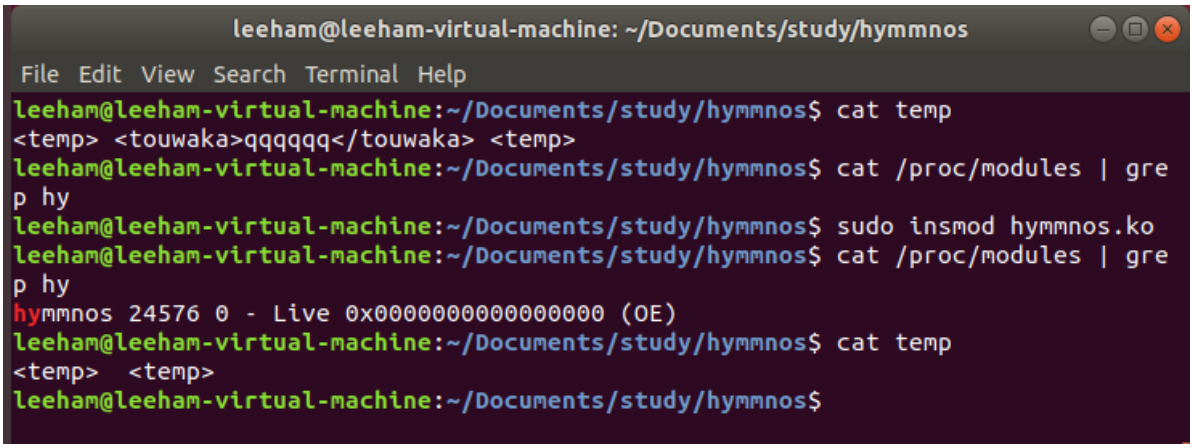


- 使用kill -51命令可以隐藏或显示文件内容

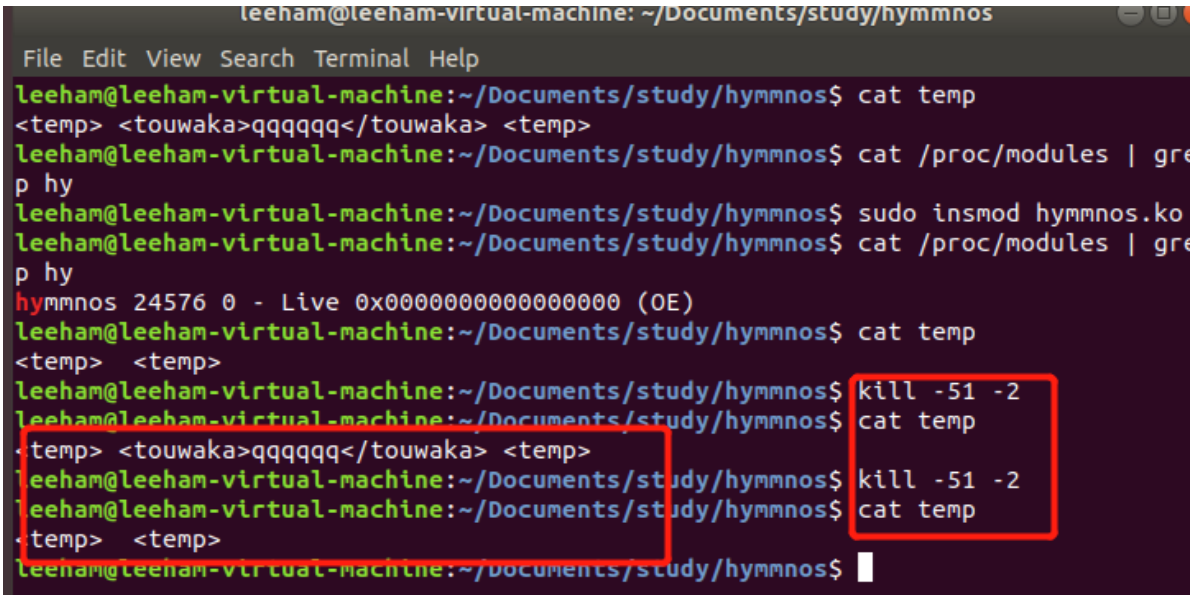
1. 首先在当前目录下新建一个文件，内容可定义如下（实际上只要包含标签就可以）；此时模块还没有有加载；并且内容是显示的。



2. 接着加载模块，此时文件内容中的标签中的内容被隐藏。

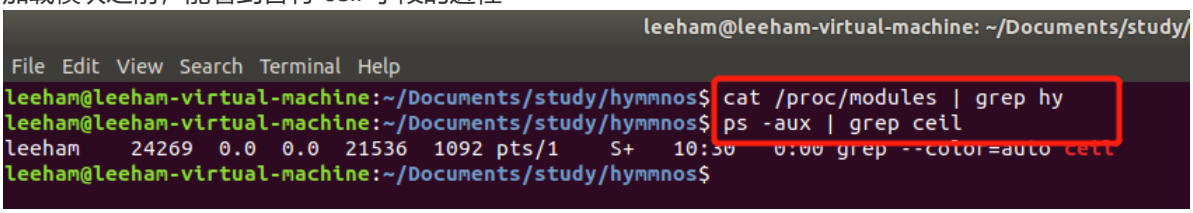


3. 分别使用kill -51命令可以隐藏或显示文件内容

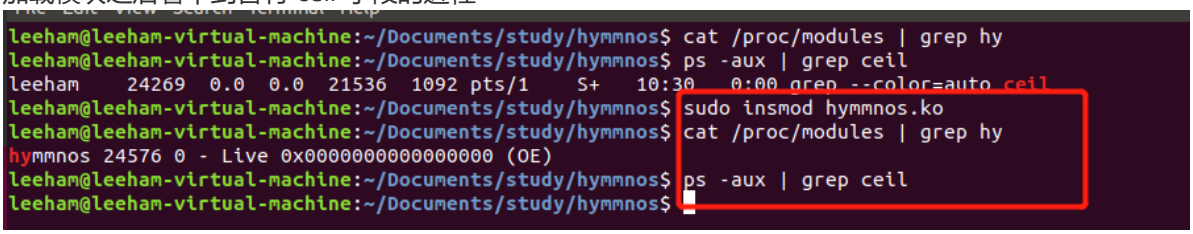


- 对包含指定字段的进程隐藏

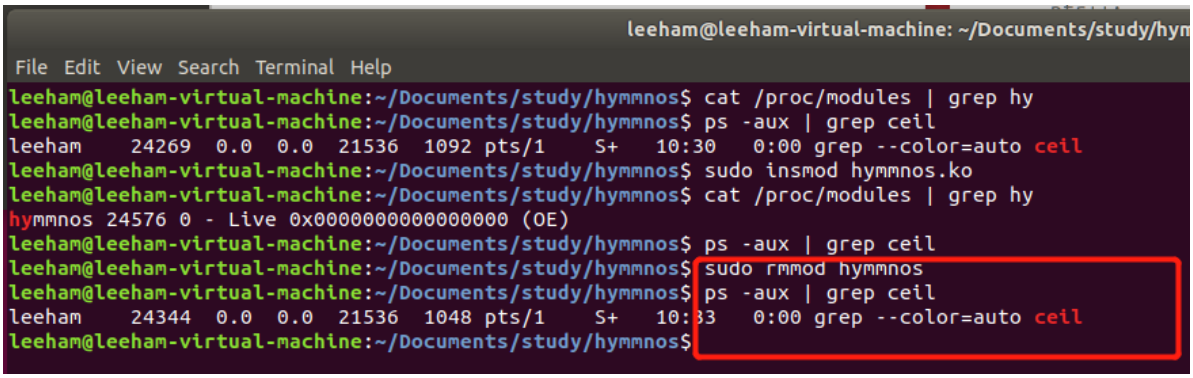
1. 加载模块之前，能看到含有“ceil”字段的进程



2. 加载模块之后看不到含有“ceil”字段的进程



3. 卸载之后又可以看到含有“ceil”字段进程



- 隐藏被隐藏进程的TCP连接

1. 加载一个不打在这个功能的rootkit模块
2. 使用sudo netstat -antup|head -10命令可以查询到一个tcp连接的进程号为1125，我们使用隐藏进城的功能
3. 再次调用命令，如下图所示，其中最上面的tcp连接就是进程号为1125的连接，可以看到他的进程号被隐藏了，但是整个tcp并没有被隐藏。

```
make[1]: Leaving directory '/usr/src/linux-headers-4.10.0-28-generic'
jiazijian@jiazijian-virtual-machine:~/jzj/test_tcp$ sudo insmod hymmnos.ko
jiazijian@jiazijian-virtual-machine:~/jzj/test_tcp$ lsmod|grep hymmnos
hymmnos                24576  0
jiazijian@jiazijian-virtual-machine:~/jzj/test_tcp$ sudo kill -49 1125
jiazijian@jiazijian-virtual-machine:~/jzj/test_tcp$ sudo netstat -antup|head -10
激活Internet连接（服务器和已建立连接的）
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.1.1:53            0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      2758/cupsd
tcp6       0      0 ::1:631                 :::*                    LISTEN      2758/cupsd
udp        0      0 0.0.0.0:631             0.0.0.0:*                           2760/cups-browsed
udp        0      0 0.0.0.0:49829           0.0.0.0:*                           -
udp        0      0 0.0.0.0:54998           0.0.0.0:*                           856/avahi-daemon: r
udp        0      0 127.0.1.1:53            0.0.0.0:*                           -
udp        0      0 0.0.0.0:68              0.0.0.0:*                           1098/dhclient
```

4. 于是我们加入新的功能，再次加载模块得到实验结果。

```
jiazijian@jiazijian-virtual-machine:~/jzj/Hymnos-rootkit$ sudo insmod hymnos.ko
jiazijian@jiazijian-virtual-machine:~/jzj/Hymnos-rootkit$ lsmod|grep hymnos
hymnos                 24576  0
jiazijian@jiazijian-virtual-machine:~/jzj/Hymnos-rootkit$ sudo kill -49 1125
jiazijian@jiazijian-virtual-machine:~/jzj/Hymnos-rootkit$ sudo netstat -antup|head -10
激活Internet连接（服务器和已建立连接的）
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      2758/cupsd
tcp6       0      0 ::1:631                 :::*                    LISTEN      2758/cupsd
udp        0      0 0.0.0.0:631             0.0.0.0:*                           2760/cups-browsed
udp        0      0 0.0.0.0:49829           0.0.0.0:*                           -
udp        0      0 0.0.0.0:54998           0.0.0.0:*                           856/avahi-daemon: r
udp        0      0 127.0.1.1:53            0.0.0.0:*                           -
udp        0      0 0.0.0.0:68              0.0.0.0:*                           1098/dhclient
udp        0      0 0.0.0.0:5353            0.0.0.0:*                           856/avahi-daemon: r
jiazijian@jiazijian-virtual-machine:~/jzj/Hymnos-rootkit$ sudo rmmod hymnos
jiazijian@jiazijian-virtual-machine:~/jzj/Hymnos-rootkit$ sudo netstat -antup|head -10
激活Internet连接（服务器和已建立连接的）
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.1.1:53            0.0.0.0:*               LISTEN      1125/dnsmasq
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      2758/cupsd
tcp6       0      0 ::1:631                 :::*                    LISTEN      2758/cupsd
udp        0      0 0.0.0.0:631             0.0.0.0:*                           2760/cups-browsed
udp        0      0 0.0.0.0:49829           0.0.0.0:*                           1125/dnsmasq
udp        0      0 0.0.0.0:54998           0.0.0.0:*                           856/avahi-daemon: r
udp        0      0 127.0.1.1:53            0.0.0.0:*                           1125/dnsmasq
udp        0      0 0.0.0.0:68              0.0.0.0:*                           1098/dhclient
```

- 控制（阻止）新的内核模块加载
  1. 先把当前模块 `hymmnos` 加载到内核，此时会将真实的 `init()` 和 `exit()` 替换为 `fake_init()` 和 `fake_exit()` ；由于将初始化函数替换，导致新模块不能加载。
  2. 加载一个新的模块；加载之后 `dmesg` 可以看到 `fake_init()` 的 `printk` 内容；
  3. 卸载这个简单的模块，能从 `dmesg` 中看到 `fake_exit()` 的 `printk` 内容；
  4. 如果成功看到这两个 `printk` 的信息，则监听成功

函数说明

基础辅助函数

函数	说明
`file_open`	打开一个文件返回 file 结构，`open` 函数的内核实现
`file_close`	`close` 函数的内核实现
`file_read`	`read` 函数的内核实现
`file_write`	`write` 函数的内核实现
`file_sync`	`sync` 函数的内核实现
`make_rw`	通过查找指定地址的页表，并设置权限为可写
`make_ro`	通过查找指定地址的页表，并设置权限为只读
`read_whole_file`	直接读写整个文件的内容，只适合读写文件内容不大的文件，在使用完成后，需要手动销毁缓冲区
`read_n_bytes_of_file`	读取 n 字节文件内容，需要手动销毁缓冲区

隐藏文件与进程功能

函数	说明
`check_file_suffix`	确定文件名称是否满足特定后缀
`is_int`	确定是否为数字
`is_pid_hidden`	该 PID 进程是否被隐藏，Rootkit 使用了一个双向链表保存被制定隐藏的进程
`make_pid_hidden`	将该进程隐藏，如果已经被隐藏，直接返回
`make_pid_show`	解除该 PID 的隐藏
`clean_hidden_pids`	清除pid链表，释放内存
`check_process_name`	确定该进程可执行文件是否包含特定字符串
`check_process_prefix`	确定该进程是否饱和特定字符串在可执行文件名称或命令行中，并确定该进程是否是被指定需要隐藏的进程
`check_file_name`	确定该文件是否是 Rootkit 文件
`should_be_hidden`	确定一个目录项是否需要被隐藏
`new_sys_getdents`	对 `getdents` 的 hook 函数，检查是否有需要被隐藏的的目录项目并进行隐藏
`new_sys_getdents64`	对 `getdents64` 的 hook 函数，检查是否有需要被隐藏的的目录项目并进行隐藏

packet 记录模块

函数	说明
`save_to_log`	保存内容到指定的的文件
`password_found`	检查是否可能包含密码
`http_header_found`	检查是否包含 Http Header
`new_sys_sendto`	对 `send` 函数进程 hook，并对进程进行检查，如果包含感兴趣的内容则保存

隐藏 tcp 连接信息

函数	说明
`is_inode_hidden`	对于需要被隐藏的端口，记录下他们的 inode 信息，使用一个单向链表进行记录
`make_inode_hidden`	添加新的 inode 信息到链表中
`clean_hidden_inodes`	清空 inode 信息链表
`extract_type_1_socket_inode`	从 socket:[12345] 中提取 12345，该数字即是 inode 节点
`load_inodes_of_process`	检查需要被隐藏进程的 fd，如果 fd 中存在软连接到 socket 的，软链接的目标将是类似 socket:[12345] 的形式，12345 即是 socket 的 inode 节点，将这些 inode 节点记录下来
`load_inodes_to_hide`	从需要被隐藏的进程中寻找需要被隐藏的 socket 结点
`next_column`	读取下一行，帮助函数
`new_seq_show`	对 `/proc/net/tcp` 文件对 `show` 函数的 hook 函数，该文件是特殊文件，通过他可以获得 tcp 连接信息。调用原始函数后，对内容进行过滤，删除掉需要被隐藏对 inode 对条目

隐藏特定内容模块

函数	说明
`f_check`	确定是否包含特定 tag
`hide_content`	删除特定 tag 间的内容
`e_fget_light`	轻量对获取特定 fd 的引用， 该函数的目的是降低对性能对影响
`new_sys_read`	对 `read` 调用的 hook，该函数会预先尝试获得该文件对锁，失败时不做处理，这样做的原因是，需要隐藏特定内容对文件一般是不经常被读写的，所以可以获取锁，而对于高 IO 的文件可以降低性能影响

网络后门功能

函数	说明
`s_xor`	对缓冲区每个字符串进行异或，混淆流量
`atoi`	转化为 int
`exec`	在用户态执行命令
`shell_execer`	执行 shell
`shell_exec_queue`	准备好 work queen 数据结构，将 shell 执行任务放入工作队列中
`decode_n_spawn`	对缓冲区进行 xor 解码
`magic_packet_hook`	注册为 packet 处理函数，并在最开始阶段对 packet 进行处理，如果是特定结构 packet 进行处理并 drop，否则传递给下一个阶段
`regist_backdoor`	注册后门
`unregist_backdoor`	取消对后门的注册

内核模块隐藏功能

函数	说明
`hide`	隐藏内核模块，通过将该模块从模块信息链表上删除
`show`	恢复内核模块，通过将该模块信息加入模块信息链表

内核模块监视功能

函数	说明
`fake_init, fake_exit`	替换其他模块的 `init` 和 `exit`
`module_notifier`	需要被注册的模块 `notifier` 函数，简单的替换新加入模块的 `init` 和 `exit` 函数以组织模块的添加
`regist_komon, unregist_komon`	注册与接触注册模块 notifier

控制与 root 后门功能

函数	说明
`new_sys_kill`	对 `kill` 调用进行 hook，负责控制，使用未被使用的信息号数

初始化功能

函数	说明
`acquire_sys_call_table`	查找系统调用表，通过对 `close` 函数的标记查找
`create_file`	创建文件
`create_files`	创建记录文件
`rootkit_start`	Rootkit 的 `init` 函数
`rootkit_end`	Rootkit 的 `uninit` 函数