十五、管理linux内核和可加载内核模块

Linux黑客基础-142-管理LINUX内核和可加载内核模块-什么是内核

Linux黑客基础-143-管理LINUX内核和可加载内核模块-内核文件解析

Linux黑客基础-144-管理LINUX内核和可加载内核模块-内核模块介绍

Linux黑客基础-145-管理LINUX内核和可加载内核模块-sysctl工具使用和内核优化

Linux黑客基础-146-管理LINUX内核和可加载内核模块-模块的加载和删除化

Linux黑客基础–142–管理LINUX内核和可加载内核模块–什么是内核

Linux 其实指的是核心！ 这个“核心 （ kernel） ”是整个操作系统的最底层， 他负责了整个硬件的驱动， 以及提供各种系统所需的核心功能， 包括防火墙机制、 是否支持 LVM 或 Quota 等文件系统等等， 这些都是核心所负责的！

计算机真正在工作的东西其实是“硬件”， 例如数值运算要使用到 CPU、 数据储存要使用到硬盘、 图形显示会用到显卡、 音乐发声要有音效芯片、 连接 Internet 可能需要网卡等等。 那么如何控制这些硬件呢？ 那就是核心的工作了！ 也就是说， 你所希望计算机帮你达成的各项工作， 都需要通过“核心”的帮助才行！ 当然啰， 如
果你想要达成的工作是核心所没有提供的， 那么你自然就没有办法通过核心来控制计算机使他工作啰！

操作系统的组成=内核空间+用户空间

内核是操作系统的中枢神经系统，控制着它所做的一切，包括管理硬件组件之间的交互和启动必要的服务。内核在你看到的用户应用程序和运行所有东西的硬件（如 CPU，内存和硬盘驱动器）之间运行。

Linux黑客基础–143–管理LINUX内核和可加载内核模块–内核文件解析

1、什么是内核

操作系统的组成=内核空间+用户空间

那么核心到底是什么啊？ 其实核心就是系统上面的一个文件而已， 这个文件包含了驱动主机各项硬件的侦测程序与驱动模块。

2、查看内核版本

```
┌──(root💀kali)-[~/桌面/work]
└─# uname -a
Linux kali 5.14.0-kali2-amd64 #1 SMP Debian 5.14.9-2kali1 (2021-10-04) x86_64 GNU/Linux
```

```
┌──(root💀kali)-[~/桌面/work]
└─# uname -r
5.14.0-kali2-amd64
```

```
┌──(root💀kali)-[~/桌面/work]
└─# cat /proc/version
Linux version 5.14.0-kali2-amd64 (devel@kali.org) (gcc-10 (Debian 10.3.0-11) 10.3.0, GNU ld (GNU Binutils for Debian)
2.37) #1 SMP Debian 5.14.9-2kali1 (2021-10-04)
```

3、查看内核文件

ls /boot/vmlinuz–`uname –r`*

```
┌──(root💀kali)-[/boot]
└─# ls /boot/vmlinuz-`uname -r`*
/boot/vmlinuz-5.14.0-kali2-amd64
```

ls /boot/vmlinuz–$(uname –r)*

```
┌──(root💀kali)-[/boot]
└─# ls /boot/vmlinuz-$(uname -r)*
/boot/vmlinuz-5.14.0-kali2-amd64
```

file vmlinuz–5.14.0–kali2–amd64

```
┌──(root💀kali)-[/boot]
└─# file vmlinuz-5.14.0-kali2-amd64
vmlinuz-5.14.0-kali2-amd64: Linux kernel x86 boot executable bzImage, version 5.14.0-kali2-amd64 (devel@kali.org) #1 S
MP Debian 5.14.9-2kali1 (2021-10-04), RO-rootFS, swap_dev 0x6, Normal VGA
```

4、系统的基本启动流程

BIOS（开机自检）–––读取MBR内的操作系统引导程序（GRUB）–––加载系统内核到内存–––启动系统的第一个服务systemd（前身位init）–––由systemd加载系统的服务

Linux黑客基础–144–管理LINUX内核和可加载内核模块–内核模块介绍

核心模块(kernel module)

现在的Linux内核是基于模块化设计的，可以通过内核模块实现功能的扩展和识别一些新的硬件，而不需要重更新编译

我们把种模块称为LKM（可加载内核模块）

一种称为 rootkit 的特殊类型的恶意软件通常通过这些 LKM 嵌入到操作系统的内核中。如果恶意软件嵌入内核， 黑客就可以完全控制操作系统

Linux黑客基础–145–管理LINUX内核和可加载内核模块–sysctl工具使用和内核优化

目的：

提升性能

扩展功能

系统的加固、安全防范

工具：sysctl

配置文件：/etc/sysctl.conf

查看帮助 man sysctl.conf

–p, ––load[=<file>］read values from file　读取文件中的参数值使其生效

–a　显示所有生效的值

–w, ––write　　enable writing a value to variable　临时更改内核参数的值

sysctl –w net.ipv4.ip_forward=1　临时直接写入

案例1、禁止其他主机ping

忽略去他主机的echo请求

ls /proc/sys/net/ipv4/icmp_echo_ignore_all

echo 1 > icmp_echo_ignore_all　立即生效，只生效一次

永久生效

vi /etc/sysctl.conf　编辑配置文件

net.ipv4.icmp_echo_ignore_all = 1　在配置文件下写入

sysctl –p　使其生效

案例2、打开IP转发功能（实施中间人攻击，劫持流量时）

场景：实施中间人攻击，劫持流量（攻击现象如ARP欺骗）

vi /etc/sysctl.conf　编辑配置文件

net.ipv4.ip_forward = 1　在配置文件下写入

sysctl –p　使其生效

内核的优化

内核的优化 Plain Text

```
#-内核优化开始——

# 内核panic时，1秒后自动重启

kernel.panic = 1

# 允许更多的PIDs（减少滚动翻转问题）; may break some programs 32768

kernel.pid_max = 32768

# 内核所允许的最大共享内存段的大小（bytes）

kernel.shmmax = 4294967296

# 在任何给定时刻，系统上可以使用的共享内存的总量（pages）

kernel.shmall = 1073741824

# 设定程序core时生成的文件名格式

kernel.core_pattern = core_%e

# 当发生oom时，自动转换为panic

vm.panic_on_oom = 1

# 表示强制Linux VM最低保留多少空闲内存（Kbytes）

vm.min_free_kbytes = 1048576

# 该值高于100，则将导致内核倾向于回收directory和inode cache

vm.vfs_cache_pressure = 250

# 表示系统进行交换行为的程度，数值（0-100）越高，越可能发生磁盘交换

vm.swappiness = 20

# 仅用10%做为系统cache

vm.dirty_ratio = 10

# 增加系统文件描述符限制 2^20-1

```

```
fs.file-max = 1048575

# 网络层优化

# listen()的默认参数,挂起请求的最大数量，默认128

net.core.somaxconn = 1024

# 增加Linux自动调整TCP缓冲区限制

net.core.wmem\_default = 8388608
net.core.rmem\_default = 8388608
net.core.rmem\_max = 16777216
net.core.wmem\_max = 16777216

# 进入包的最大设备队列.默认是300

net.core.netdev_max_backlog = 2000

# 开启SYN洪水攻击保护

net.ipv4.tcp_syncookies = 1

# 开启并记录欺骗，源路由和重定向包

net.ipv4.conf.all.log\_martians = 1
net.ipv4.conf.default.log\_martians = 1

# 处理无源路由的包

net.ipv4.conf.all.accept\_source\_route = 0
net.ipv4.conf.default.accept\_source\_route = 0

# 开启反向路径过滤

net.ipv4.conf.all.rp\_filter = 1
net.ipv4.conf.default.rp\_filter = 1

# 确保无人能修改路由表

net.ipv4.conf.all.accept\_redirects = 0
net.ipv4.conf.default.accept\_redirects = 0
net.ipv4.conf.all.secure\_redirects = 0
net.ipv4.conf.default.secure\_redirects = 0

# 增加系统IP端口限制

```

```
net.ipv4.ip_local_port_range = 9000 65533

# TTL

net.ipv4.ip_default_ttl = 64

# 增加TCP最大缓冲区大小

net.ipv4.tcp\_rmem = 4096 87380 8388608
net.ipv4.tcp\_wmem = 4096 32768 8388608

# Tcp自动窗口

net.ipv4.tcp_window_scaling = 1

# 进入SYN包的最大请求队列.默认1024

net.ipv4.tcp_max_syn_backlog = 8192

# 打开TIME-WAIT套接字重用功能，对于存在大量连接的Web服务器非常有效。

net.ipv4.tcp\_tw\_recycle = 1
net.ipv4.tcp\_tw\_reuse = 0

# 表示是否启用以一种比超时重发更精确的方法（请参阅 RFC 1323）来启用对 RTT 的计算；为了实现更好的性能应该启用这个选项

net.ipv4.tcp_timestamps = 0

# 表示本机向外发起TCP SYN连接超时重传的次数

net.ipv4.tcp\_syn\_retries = 2
net.ipv4.tcp\_synack\_retries = 2

# 减少处于FIN-WAIT-2连接状态的时间，使系统可以处理更多的连接。

net.ipv4.tcp_fin_timeout = 10

# 减少TCP KeepAlive连接侦测的时间，使系统可以处理更多的连接。

# 如果某个TCP连接在idle 300秒后,内核才发起probe.如果probe 2次(每次2秒)不成功,内核才彻底放弃,认为该连接已失效.

net.ipv4.tcp\_keepalive\_time = 300
net.ipv4.tcp\_keepalive\_probes = 2
net.ipv4.tcp\_keepalive\_intvl = 2

```

```
# 系统所能处理不属于任何进程的TCP sockets最大数量

net.ipv4.tcp_max_orphans = 262144

# 系统同时保持TIME_WAIT套接字的最大数量，如果超过这个数字，TIME_WAIT套接字将立刻被清除并打印警告信息。

net.ipv4.tcp_max_tw_buckets = 20000

# arp_table的缓存限制优化

net.ipv4.neigh.default.gc\_thresh1 = 128
net.ipv4.neigh.default.gc\_thresh2 = 512
net.ipv4.neigh.default.gc\_thresh3 = 4096
```

net.ipv4.tcp_syncookies = 1表示开启SYN Cookies。当出现SYN等待队列溢出时，启用cookies来处理，可防范少量SYN攻击，默认为0，表示关闭；

net.ipv4.tcp_tw_reuse = 1 表示开启重用。允许将TIME-WAIT sockets重新用于新的TCP连接，默认为0，表示关闭；

net.ipv4.tcp_tw_recycle = 1 表示开启TCP连接中TIME-WAIT sockets的快速回收，默认为0，表示关闭。

net.ipv4.tcp_fin_timeout = 30 表示如果套接字由本端要求关闭，这个参数决定了它保持在FIN-WAIT-2状态的时间。默认是60s。

net.ipv4.tcp_keepalive_time = 1200 表示当keepalive起用的时候，TCP发送keepalive消息的频度。缺省是2小时，改为20分钟。

net.ipv4.ip_local_port_range = 1024 65000 表示用于向外连接的端口范围。缺省情况下很小：32768到61000，改为1024到65000。

net.ipv4.tcp_max_syn_backlog = 8192 表示SYN队列的长度，默认为1024，加大队列长度为8192，可以容纳更多等待连接的网络连接数。

net.ipv4.tcp_max_tw_buckets = 5000表示系统同时保持TIME_WAIT套接字的最大数量，如果超过这个数字，TIME_WAIT套接字将立刻被清除并打印警告信息。默认为180000，改为5000。

Linux黑客基础-146-管理LINUX内核和可加载内核模块-模块的加载和删除化

内核文件一般都是压缩文件， 在加载到内核之前需要进行解压缩
核心解压时需要一个内存磁盘（RAM DISK）

文件名：/boot/initramfs---内核版本

/boot/initrd.img---内核版本

1、模块文件存放位置：/lib/modules/`uname –r`/kernel

```
┌──(root💀kali)-[/proc/sys/net/ipv4]
└─# ls /lib/modules/`uname -r`/kernel
arch  block  crypto  drivers  fs  lib  mm  net  sound  virt
```

```
arch     ：与硬件平台有关的项目，例如 CPU 的等级等等；
crypto     ：核心所支持的加密的技术，例如 md5 或者是 des 等等；
drivers     ：一些硬件的驱动程序，例如显卡、网卡、PCI 相关硬件等等；
fs     ：核心所支持的 filesystems ，例如 vfat, reiserfs, nfs 等等；
lib     ：一些函数库；
net     ：与网络有关的各项协定数据，还有防火墙模块 （net/ipv4/netfilter/*） 等等；
sound     ：与音效有关的各项模块；
```

2、模块之间由相互的依耐性

3、模块文件的扩展名*.ko

4、内核模块的管理

方法1：insmod（老的方法）

方法2：modprobe（推荐）

5、lsmod 列出核心加载的模块

```
┌──(root💀kali)-[/proc/sys/net/ipv4]
└─# lsmod
Module                  Size  Used by
```

6、modinfo 列出模块的详细信息

```
┌──(root💀kali)-[/proc/sys/net/ipv4]
└─# modinfo e1000
filename:       /lib/modules/5.14.0-kali2-amd64/kernel/drivers/net/ethernet/intel/e1000/e1000.ko
```

7、模块的加载和删除

加载一个模块

insmod /lib/modules/5.14.0–kali2–amd64/kernel/fs/fat/fat.ko

insmod /lib/modules/`uname –r`/kernel/fs/fat/fat.ko

查看模块

lsmod | grep fat

删除模块

rmmod fat

modprobe 自动解决模块之间的依赖关系

案例：CIFS

modprobe cifs　加载模块

modprobe –r cifs　删除模块

dmesg　//查看系统详细的硬件信息

腾讯哈勃分析系统 (qq.com)

悟空扫描器 https://github.com/Lee-0x00/wukong-agent