

前言

在Linux上面搞事情很多东西都可能需要我们利用源码来自己编译,相对于Windows来讲可能会比较麻烦和耗时,这个过程中肯定会遇到很多报错,所以一定要有耐心.....

方法步骤

编译内核

首先到linux内核的[官网](#)下载一份内核源代码并解压:

Protocol	Location
HTTP	https://www.kernel.org/pub/
GIT	https://git.kernel.org/
RSYNC	rsync://rsync.kernel.org/pub/

Latest Stable Kernel:



5.2.2

mainline:	5.2	2019-07-07	[tarball]	[pgp]	[patch]		[view diff]	[browse]	
stable:	5.2.2	2019-07-21	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
stable:	5.1.19	2019-07-21	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	4.19.60	2019-07-21	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	4.14.134	2019-07-21	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	4.9.186	2019-07-21	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	4.4.186	2019-07-21	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
longterm:	3.16.70	2019-07-09	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse]	[changelog]
linux-next:	next-20190719	2019-07-19						[browse]	

至于需要下载的版本,随意就好,我下载的是5.2.1的....

然后先安装有些依赖:

```
sudo apt-get install git fakeroot build-essential ncurses-dev xz-utils libssl-dev bc
```

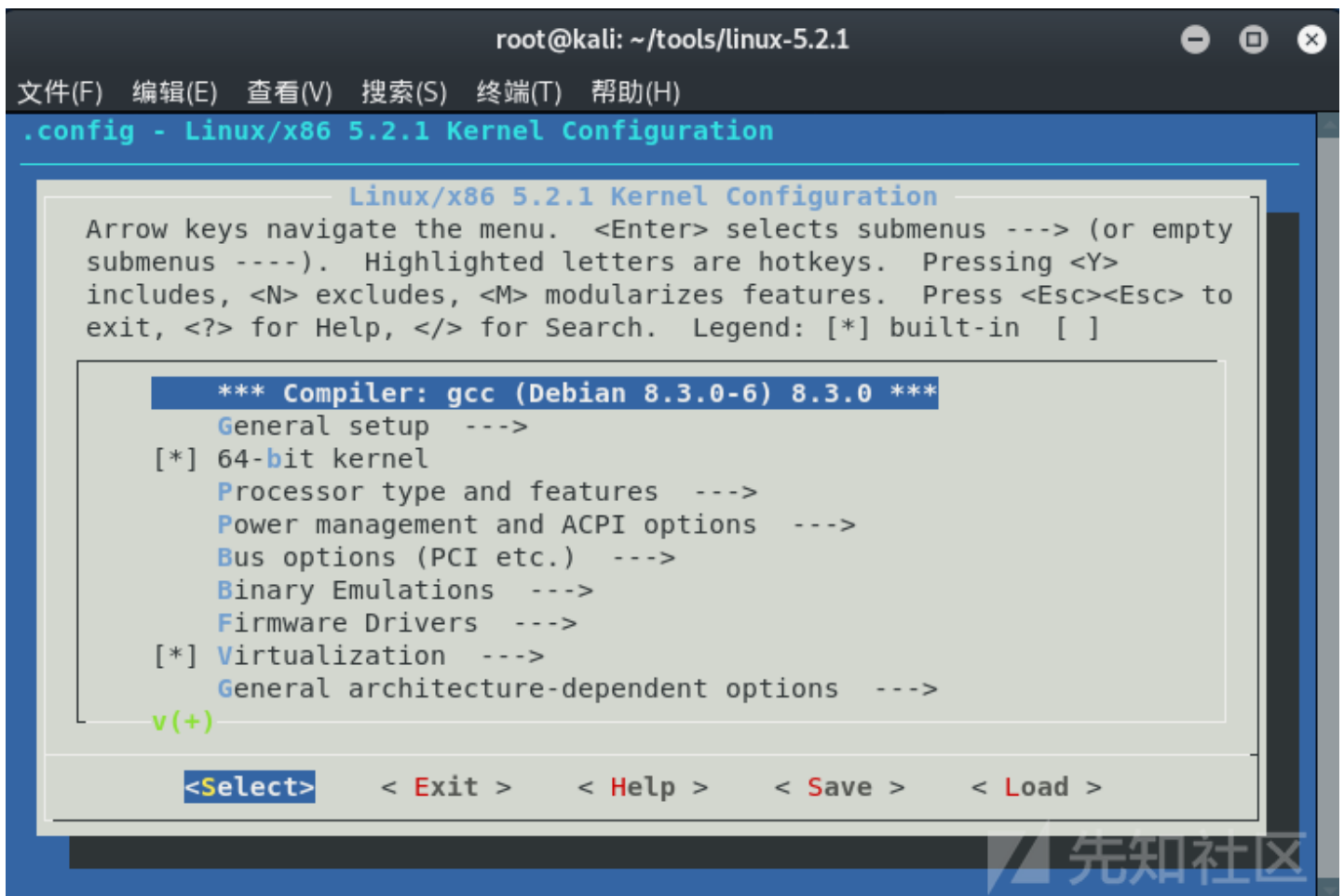
这些依赖并不一定全部概况完了,在编译的过程中可能在报错信息中还要提示你安装一些依赖,具体根据报错提示再进行安装就可以...

然后进入解压目录:

```
make menuconfig
```

```
root@kali:~/tools# cd linux-5.2.1/
root@kali:~/tools/linux-5.2.1# ls
arch          fs            LICENSES     net           tools
block         include      MAINTAINERS  README       usr
certs         init         Makefile     samples      virt
COPYING       ipc          mm           scripts      vmlinux
CREDITS       Kbuild      modules.builtin  security     vmlinux.o
crypto        Kconfig     modules.builtin.modinfo  sound
Documentation kernel      modules.order   System.map
drivers        lib         Module.symvers  test
root@kali:~/tools/linux-5.2.1# make menuconfig
```

这里会跳出一个设置框:



注意下面的配置就好:

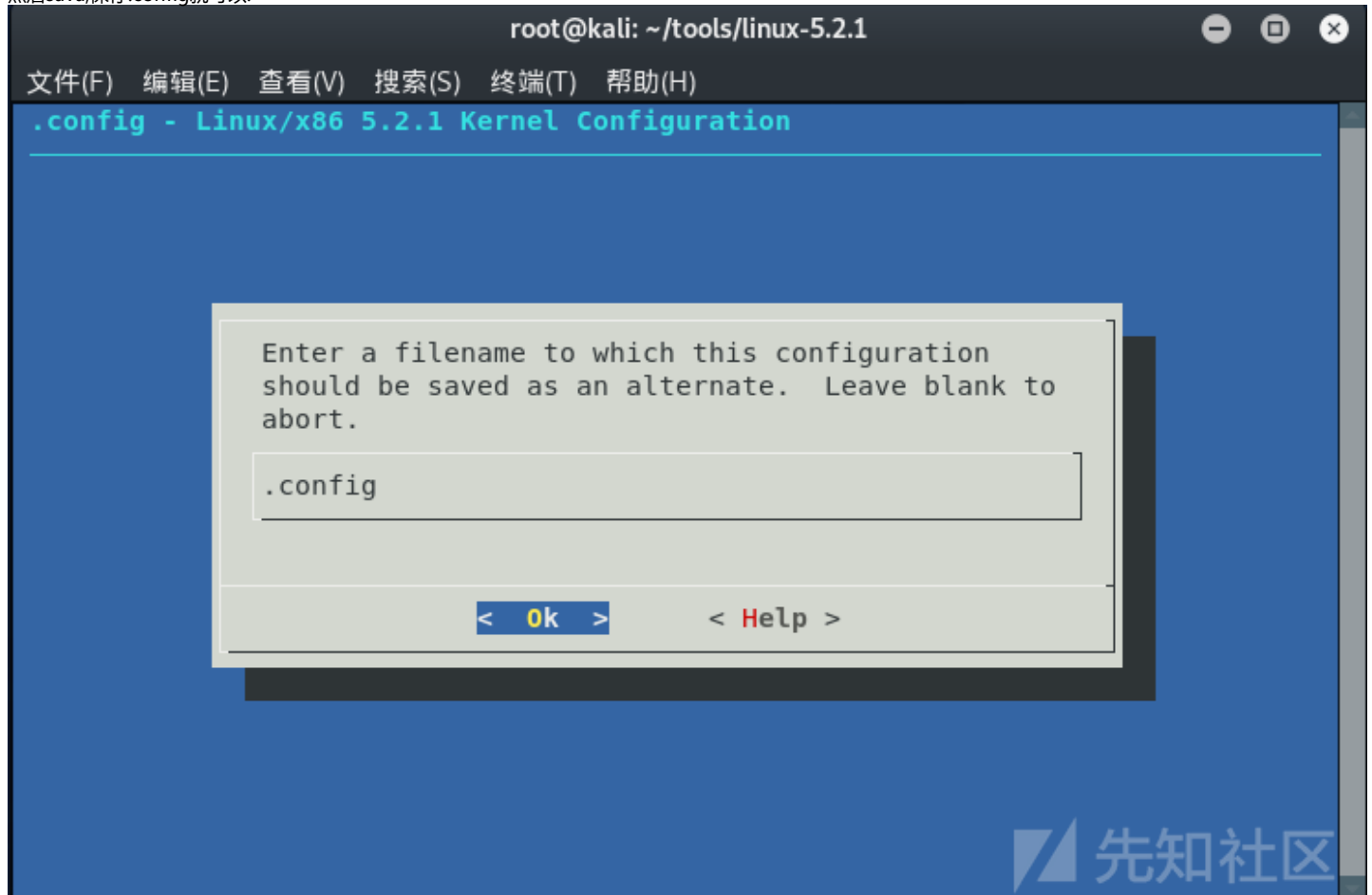
KernelHacking -->

- Compile the kernel with debug info
- Compile the kernel with frame pointers

Processor type and features-->

- Paravirtualized guest support

然后save,保存.config就可以:



然后:

```
make -j4
```

虚拟机分配了4个核,使用-j4可以快一点...

```
root@kali:~/tools/linux-5.2.1# make -j4
scripts/kconfig/conf --syncconfig Kconfig
DESCEND objtool
CALL scripts/atomic/check-atomics.sh
CALL scripts/checksyscalls.sh
CC init/main.o
CC kernel/fork.o
CC mm/highmem.o
AR mm/built-in.a
CC arch/x86/kernel/traps.o
CHK include/generated/compile.h
CC init/do_mounts.o
CC kernel/panic.o
CC arch/x86/kernel/acpi/boot.o
```

这个过程是比较漫长的,如果你的kernel内核比较低或依赖不够的话,就会报比较多的错误,这需要根据你具体情况百度了,耐心...
当make结束了就可以:

```
make all
```

```
root@kali:~/tools/linux-5.2.1# make all
CALL scripts/checksyscalls.sh
CALL scripts/atomic/check-atomics.sh
DESCEND objtool
CHK include/generated/compile.h
```

还是可能会报错,不过都可以百度到,最多改一下Makefile文件的,不紧张...
最后就可以:

```
make modules
```

```
root@kali:~/tools/linux-5.2.1# make modules
CALL scripts/checksyscalls.sh
CALL scripts/atomic/check-atomics.sh
DESCEND objtool
```

先知社区

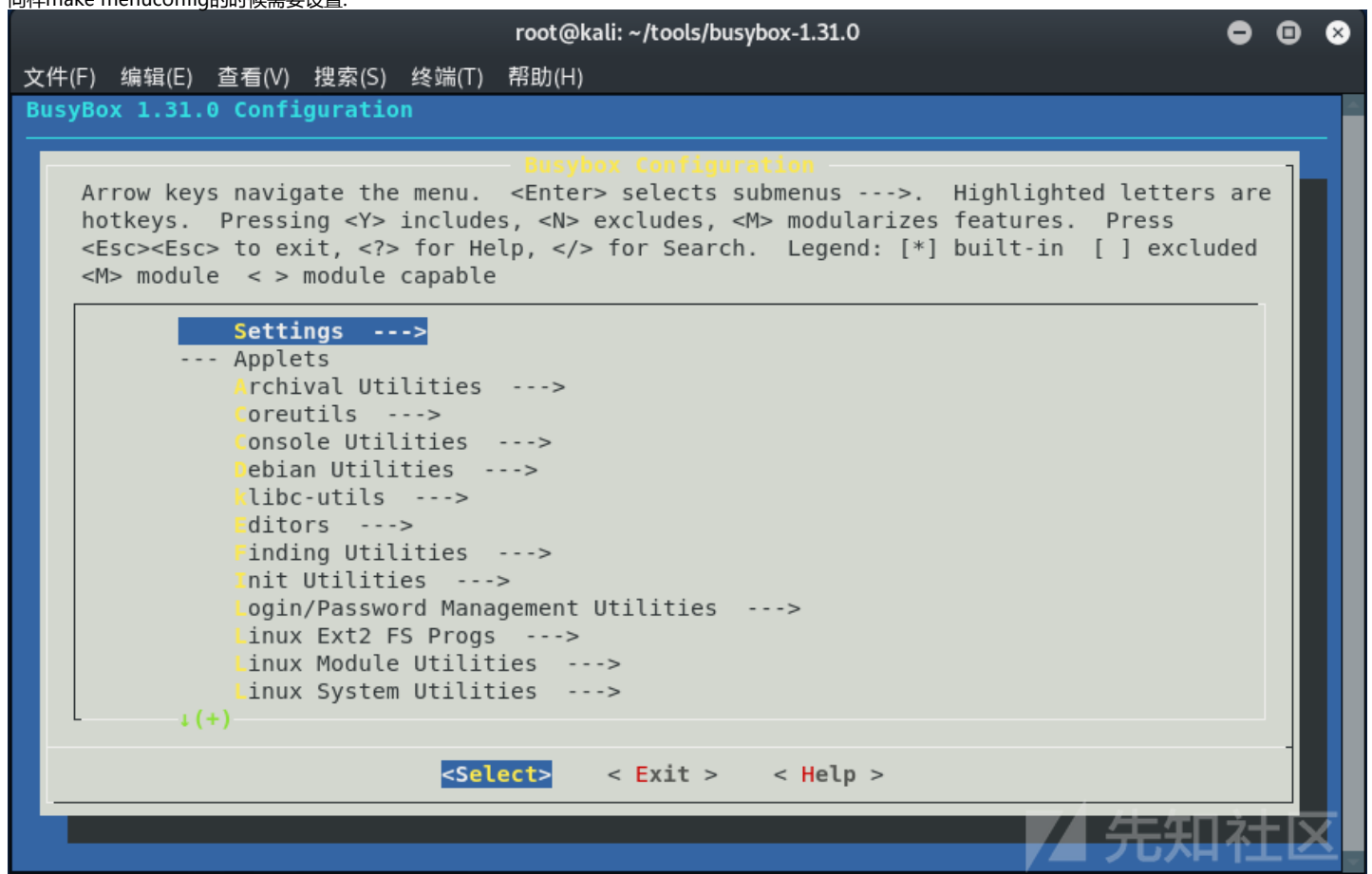
我们可以从./arch/x86/boot/拿到bzImage，从源码根目录拿到vmlinuz....

编译busybox && 构建文件系统

```
cd ..
wget https://busybox.net/downloads/busybox-1.31.0.tar.bz2
tar -jxvf busybox-1.19.4.tar.bz2
cd busybox-1.19.4
make menuconfig
make install
```

其中busybox-1.31.0.tar.bz2建议下载最新版的...

同样make menuconfig的时候需要设置:



Busybox Settings -> Build Options -> Build Busybox as a static binary ☒

☒:

Linux System Utilities -> [] Support mounting NFS file system ☒

Networking Utilities -> [] inetd (Internet ☒)

编译完make install后,在busybox源代码的根目录下会有一个_install目录下会存放好编译后的文件:

```
root@kali:~/tools/busybox-1.31.0# ls
applets          examples         modutils
applets_sh       filter_log       networking
arch             findutils       NOFORK_NOEXEC.lst
archival         include         NOFORK_NOEXEC.sh
AUTHORS          init            printutils
busybox          _install        procs
busybox.links    INSTALL         qemu_multiarch_testing
busybox_unstripped klibc-utils     README
busybox_unstripped.map libbb           rootfs.img
busybox_unstripped.out libpwdgrp       runit
Config.in        LICENSE        scripts
configs          loginutils     selinux
console-tools    mailutils      shell
coreutils        Makefile       size_single_applets.sh
debianutils      Makefile.custom syslogd
dev              Makefile.flags testsuite
docs             Makefile.help  TODO
e2fsprogs        make_single_applets.sh TODO_unicode
editors          miscutils      util-linux
```

然后我们需要在里面配置一下:

```
cd _install
mkdir proc sys dev etc etc/init.d
vim etc/init.d/rcS
chmod +x etc/init.d/rcS
```

其中vim etc/init.d/rcS的内容:

```
#!/bin/sh
mount -t proc none /proc
mount -t sysfs none /sys
/sbin/mdev -s
```

然后利用命令创建文件系统:

```
find . | cpio -o --format=newc > ../rootfs.img
```

```
root@kali:~/tools/busybox-1.31.0/_install# find . | cpio -o --format=newc > ../rootfs.img
13642 块
root@kali:~/tools/busybox-1.31.0/_install#
```

最后我们就可以使用 qemu 来运行内核了:

```
qemu-system-x86_64 \
-kernel ~/tools/linux-5.2.1/arch/x86_64/boot/bzImage \
-initrd ~/tools/busybox-1.31.0/rootfs.img \
-append "console=ttyS0 root=/dev/ram rdinit=/sbin/init" \
-cpu kvm64,+smep,+smap \
-nographic \
-gdb tcp::1234
```

其中:

```
-cpu kvm64,+smep,+smap CPU
-kernel bzImage
-initrd busybox rootfs.img
-gdb tcp::1234 gdb
```

```
/ # ls
bin      etc      linuxrc  root     sys      usr
dev      init     proc     sbins    tmp
/ #
```

加载驱动

加载驱动很简单,只需要命令insmod就可以,然后rmmod可以卸载驱动,lsmod可以查看加载了的驱动....

```
/ # lsmod
/ # insmod baby.ko
/ # lsmod
baby 16384 0 - Live 0xfffffffffc021e000 (OE)
/ # rmmmod baby.ko
-/bin/sh: rmmmod: not found
/ # rmmod baby.ko
/ # lsmod
/ # |
```



gdb调试

我们用qemu运行内核的时候,加了一个-gdb tcp::1234的参数, qemu会在1234端口起一个gdb_server我们直接用gdb连上去:

```
/ # lsmod
/ # insmod baby.ko
/ # lsmod
baby 16384 0 - Live 0xfffffffffc021e000 (OE)
/ # rmmmod baby.ko
-/bin/sh: rmmmod: not found
/ # rmmod baby.ko
/ # lsmod
/ # |

root@kali:~/tools/linux-5.2.1# gdb vmlinux
GNU gdb (Debian 8.2.1-2) 8.2.1
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
pwndbg: loaded 180 commands. Type pwndbg [filter] for a list.
pwndbg: created $rebase, $ida gdb functions (can be used with print/break)
Reading symbols from vmlinux...done.
pwndbg> add-symbol-file ~/desktop/baby.ko 0xfffffffffc021e000
add symbol table from file "/root/desktop/baby.ko" at
.text_addr = 0xfffffffffc021e000
Reading symbols from /root/desktop/baby.ko...(no debugging symbols found)...done.
pwndbg> target remote :1234
Remote debugging using :1234
0xffffffff8ed99d46 in ?? ()
```



同时我们为了调试内核模块,利用add-symbol-file命令加载了驱动的符号文件,并且加上了系统里面驱动的加载基地址....

后续

之后我会主要利用kernel pwn来帮助学习Linux kernel
Exploit内核漏洞学习,掌握一些基本的内核漏洞利用技巧....另外如果文章有错误和改进之处,还请大家可以指出....

点击收藏 | 1 关注 | 1
[上一篇 : SA-CORE-2019-008 ...](#) [下一篇 : Capstone反汇编引擎数据类型...](#)

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](#)