

به نام خدا

گزارش فاز دوم پروژه درس سیستم های عامل

اعضای گروه:

امینه احمدی نژاد 94102647
صبا زرباف 94105311

گام اول:

در این گام ابتدا باید کرنل لینوکس را build کرده و سپس مازولی می نوشتیم و به هسته لینوکس اضافه می کردیم.
برای build کردن ابتدا قبل شروع کردن هرکاری باید چند package را install کرده با دستورات زیر:

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

سپس ورژن کرنل فعلی خود را مشخص می کنیم با دستور زیر:

```
name -r
```

به سایت kernel.org رفته و آخرین ورژن را دانلود می کنیم و چون قراره تغییراتی انجام دهیم ابتدا با دستور cp یک نمونه از آن را در documents نگه می داریم و تغییرات لازمه رو آنجا انجام می دهیم.
برای extract کردن فایل مذکور از دستور زیر استفاده می کنیم:

```
tar xf linux-4.7.1.tar.xz
```

سپس به درون فایل extract شده می رویم (cd linux-4.7.1)

قبل از کامپایل کردن باید مشخص کنیم که کدام مازول ها باید باشند و کدامیک نباید! ساده ترین راه برای انجام این کار دستورات زیر می باشند:

```
cp /boot/config-$(uname -r) .config  
make menuconfig
```

بعد دستورات بالا باید تغییرات لازم را انجام بدهیم و اگر به نظر می آید تغییری لازم نیست کافی است exit, save رو می زنیم.

(ما تعداد core cpu رو به چهار تغییر می دهیم)

سپس دستور آخر برای ساختن کرنل لینوکس را وارد می کنیم و برای اینکه از چهار تا core استفاده کند از 4 -j استفاده می کنیم :

```
sudo make -j 4 > ../build.txt
```

(برای اینکه در فایلی نگه داریم دستور ../build.txt را وارد می کنیم)

اگر بخواهیم تغییرات در فایل بالا رو ببینیم هنگام ساخت کرنل باید دستور زیر را در ترمینال دیگری که باز کردیم بزنیم:

```
tail -f build.txt
```

برای ساخت مازول ابتدا در ترمینال ubuntu یک فایل به نام hello.c می سازیم و کد زیر را در آن می نویسیم (برای نوشتن کد ابتدا باید i را کلیک کرده تا بتوان در فایل insert کرد)

```
#include<linux/module.h> //for all of modules this library is needed
#include<linux/kernel.h> // this library is needed for kernel's info
```

```
int init_module(void){
    printk(KERN_INFO "Hello OS!\n"); // This line print Hello World in dmesg
    return 0;
}
void cleanup_module(void){
    printk(KERN_INFO "GoodBye OS!\n");//This line print Goodbye World in dmesg
}
```

ماژول های کرنل مقداری با ماژول های دیگر در کامپایل شدن فرق دارند و Makefile برای نگهداری اطلاعات ماژول ساخته می شود که کد آن در زیر آمده است:

```
obj-m +=hello.o
all:
```

```
make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
```

```
clean:
```

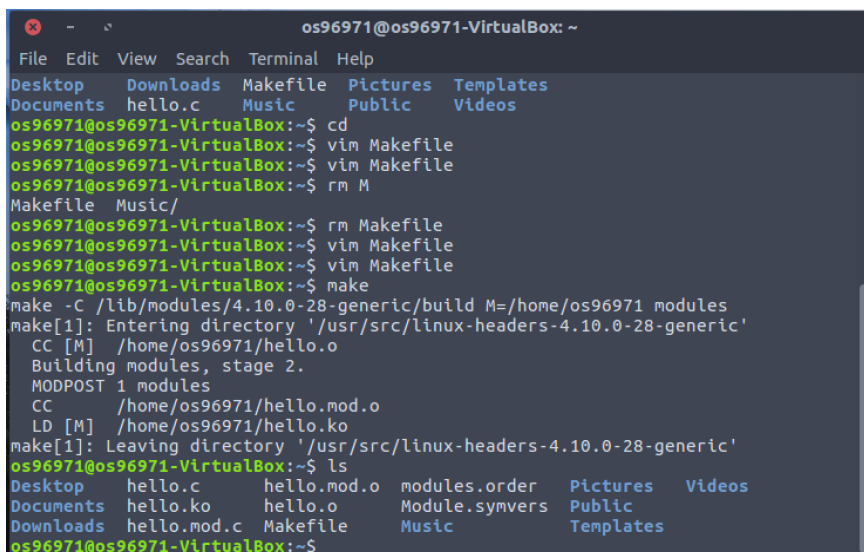
```
make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

با دستور make ماژول را کامپایل می کنیم و با کامپایل کردن ماژول فایل های hello.ko ,hello.mod.c ,hello.mod.o , hello.o , modules.order,Module.symvers

هم ساخته می شوند که در تصویر زیر نشان داده شده است.

(خروجی حاصل از کامپایل کردن ماژول در زیر

نمایش داده می شود)



```
os96971@os96971-VirtualBox: ~
File Edit View Search Terminal Help
Desktop Downloads Makefile Pictures Templates
Documents hello.c Music Public Videos
os96971@os96971-VirtualBox:~$ cd
os96971@os96971-VirtualBox:~$ vim Makefile
os96971@os96971-VirtualBox:~$ vim Makefile
os96971@os96971-VirtualBox:~$ rm M
Makefile Music/
os96971@os96971-VirtualBox:~$ rm Makefile
os96971@os96971-VirtualBox:~$ vim Makefile
os96971@os96971-VirtualBox:~$ vim Makefile
os96971@os96971-VirtualBox:~$ make
make -C /lib/modules/4.10.0-28-generic/build M=/home/os96971/modules
make[1]: Entering directory '/usr/src/linux-headers-4.10.0-28-generic'
CC [M] /home/os96971/hello.o
Building modules, stage 2.
MODPOST 1 modules
CC /home/os96971/hello.mod.o
LD [M] /home/os96971/hello.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.10.0-28-generic'
os96971@os96971-VirtualBox:~$ ls
Desktop hello.c hello.mod.o modules.order Pictures Videos
Documents hello.ko hello.o Module.symvers Public
Downloads hello.mod.c Makefile Music Templates
os96971@os96971-VirtualBox:~$
```

سپس دستور

```
insmod hello.ko
```

را می زنیم تا در dmesg خروجی را چاپ کند و

سپس دستور dmesg را می زنیم تا خروجی را مشاهده کنیم مطابق شکل زیر:

```
os96971@os96971-VirtualBox: ~
File Edit View Search Terminal Help
6.243996 RAPL PMU: hw unit of domain package 2^0 Joules
6.243997 RAPL PMU: hw unit of domain dram 2^0 Joules
6.243997 RAPL PMU: hw unit of domain ppi-gpu 2^0 Joules
6.262468 [drm] Initialized vboxvideo 1.0.0 20130823 for 0000:00:02.0 on mi
hor 0
6.343477 AVX version of gcm_enc/dec engaged.
6.343481 AES CTR mode by8 optimization enabled
7.377803 Adding 1046524k swap on /dev/sda5. Priority:-1 extents:1 across:
1046524k FS
7.434496 snd_intel8x0 0000:00:05.0: white list rate for 1028:0177 is 48000
8.575269 floppy0: no floppy controllers found
8.575305 work still pending
8.644688 IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not ready
8.646315 IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not ready
8.652280 e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control:
RX
8.652695 IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link becomes ready
1246.869002 hello: loading out-of-tree module taints kernel.
1246.869003 hello: module license 'unspecified' taints kernel.
1246.869004 Disabling lock debugging due to kernel taint
1246.869020 hello: module verification failed: signature and/or required key
missing - tainting kernel
1246.870833 Hello OS!
os96971@os96971-VirtualBox:~$
```

گام دوم:

در بخش اول گام دوم از فایل پیوست شده در فایل های ارسالی به نام `kconfig.txt` می بینیم که اول کد نوشته شده است

default n

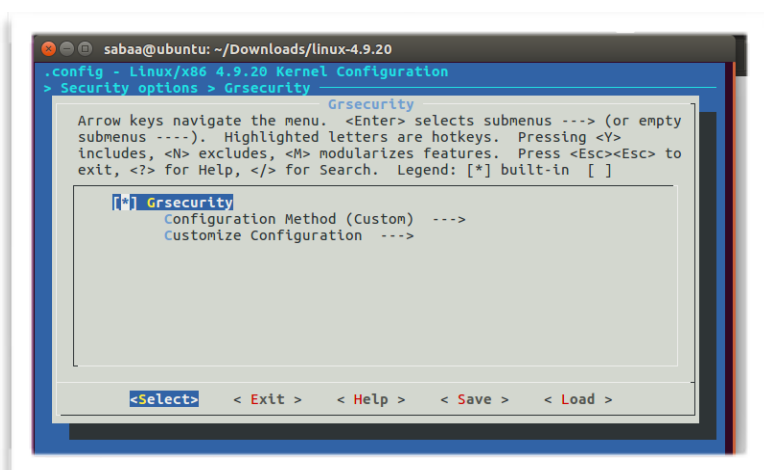
این خط بدان معنی است که طور پیش فرض این ماژول غیر فعال می باشد و نیازی نیست که ما آنرا غیرفعال کنیم.

در بخش دوم گام دوم ابتدا کرنل را دانلود کرده و همچنین ماژول `grsecurity` موردنیاز آن را دانلود کرده و سپس با استفاده از دستور

```
patch -p1 < ../grsecurity-3.1-4.9.20-201703310823.patch
```

سپس `make menuconfig` را می زنیم تا تغییرات لازم را انجام دهیم و تصویر زیر قسمت مربوط به `grsecurity` را نشان می دهد در آخر برای ساخت هسته دستور زیر را زده و نتایج را در فایل ذخیره می کنیم.

```
sudo make -j 4 > ../buildgr.txt
```



گام سوم:

ابتدا ubuntu با ویرچوال باکس نصب شد و با اضافه کردن shared files بین کامپیوتر اصلی (host) و سیستم عامل ubuntu موفق به کپی کردن کرنل اندروید در ubuntu شدیم. سپس با دستور زیر فایل فشرده شده اکسترکت شده و به دایرکتوری مورد نظر می‌رود.

```
tar -xzf android.tar.gz -C ~/
```

سپس ، دستورات زیر به ترتیب در terminal اجرا شدند تا کتابخانه‌های لازم نصب شوند.

```
sudo apt install bison build-essential curl flex git gnupg gperf libbsd0-dev  
sudo apt install liblz4-tool libncurses5-dev libsdl1.2-dev libwxgtk3.0-dev libxml2 libxml2-utils  
lzop maven pngcrush schedtool squashfs-tools  
sudo apt install xsltproc zip lib32readline-gplv2-dev zlib1g-dev g++-multilib gcc-multilib  
lib32ncurses5-dev lib32z1-dev
```

در دستور آخر به مشکل خوردم و به جای نصب lib32readline-gplv2-dev از libreadline6-dev استفاده کردم.

سپس دستورات فایل صورت پروژه به ترتیب اجرا شدند ولی به دلیل فیلتر بودن سایت‌های مورد نیاز، قبل از اجرای دستور breakfast bullhead فایل repo به صورت دستی به دایرکتوری bin اضافه شد. (همچنین با دستورات زیر سعی کردم از فیلترشکن استفاده کنم ولی موفق نشدم، و گزینه‌ی اضافه کردن دستی را انتخاب کردم.

```
sudo apt install openconnect  
sudo openconnect test.cisadd.com
```

Username: vpnmakers
Password: vpnmakers

نصب Tor browser هم بی‌فایده بود.)

نکته‌ی خیلی مهم: هنگام اجرای دستور breakfast bullhead، علی‌رغم دانلود ۱ گیگ فایل، به هیچ عنوان نباید اینترنت قطع شود چون مجبور می‌شویم همه‌ی مراحل را از اکسترکت کردن فایل از اول انجام دهیم!

بعد از دانلود شدن کامل فایل‌ها، قبل از اینکه build پروژه را شروع کنیم، با دستورات زیر برای آن cache در نظر می‌گیریم تا سرعت اجرا چند برابر شود.

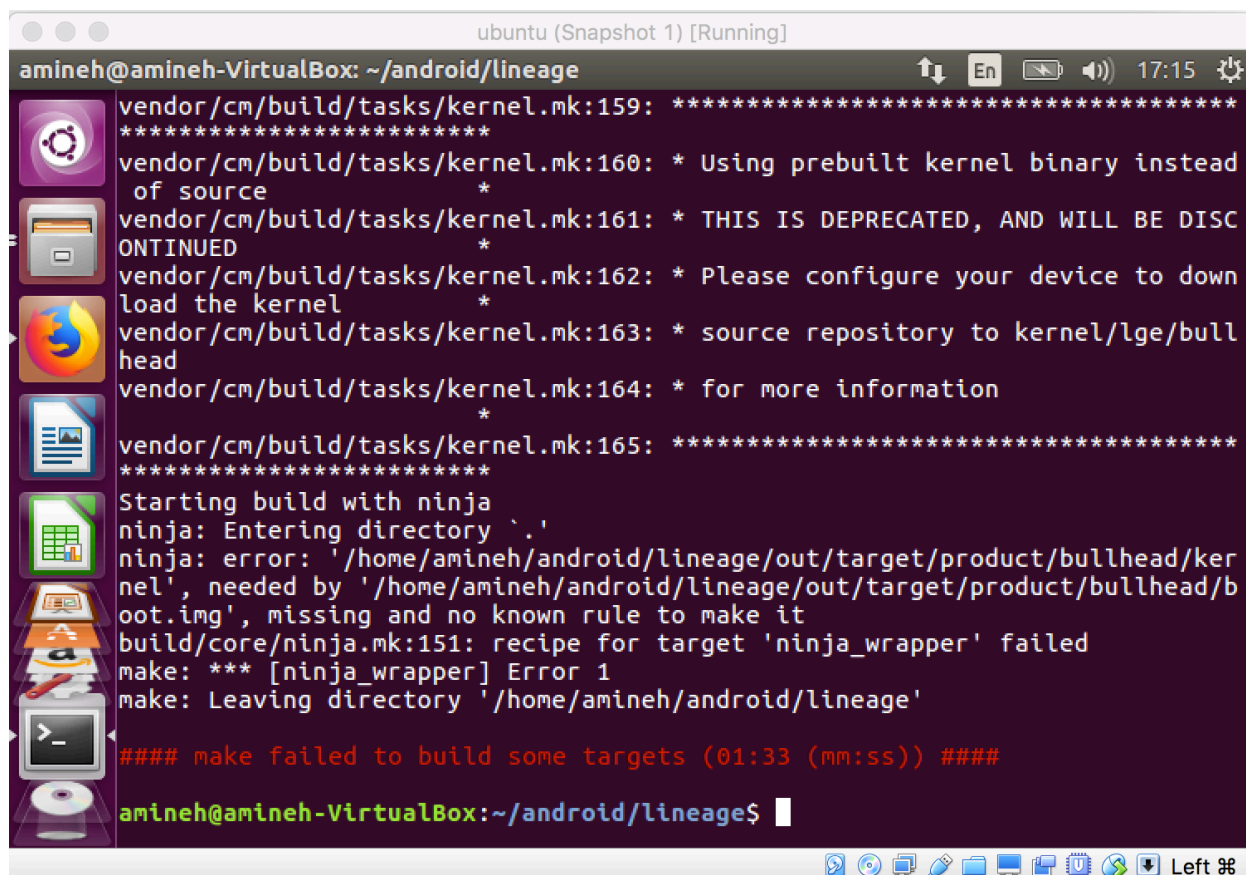
```
export USE_CCACHE=1  
prebuilts/misc/linux-x86/ccache/ccache -M 50G  
export CCACHE_COMPRESS=1  
export ANDROID_JACK_VM_ARGS="-Dfile.encoding=UTF-8 -XX:+TieredCompilation -Xmx4G"
```

توجه کنید که باید در دایرکتوری android/lineage باشیم در غیر این صورت به ارور no such file or directory می‌خوریم. همچنین در دستورات بالا، در دومین دستور 50G فرضی بوده و به جای آن باید از هر مقدار که می‌خواهیم برای کش استفاده کنیم قرار دهیم.

سپس دستورات آخر را اجرا می‌کنیم تا سیستم build شود.

```
croot  
brunch bullhead
```

اولین اروری که به آن برخورد کردم به صورت زیر بود:



```
ubuntu (Snapshot 1) [Running]  
amineh@amineh-VirtualBox: ~/android/lineage  
vendor/cm/build/tasks/kernel.mk:159: *****  
*****  
vendor/cm/build/tasks/kernel.mk:160: * Using prebuilt kernel binary instead  
of source *  
vendor/cm/build/tasks/kernel.mk:161: * THIS IS DEPRECATED, AND WILL BE DISC  
ONTINUED *  
vendor/cm/build/tasks/kernel.mk:162: * Please configure your device to down  
load the kernel *  
vendor/cm/build/tasks/kernel.mk:163: * source repository to kernel/lge/bull  
head *  
vendor/cm/build/tasks/kernel.mk:164: * for more information *  
vendor/cm/build/tasks/kernel.mk:165: *****  
*****  
Starting build with ninja  
ninja: Entering directory `.'  
ninja: error: '/home/amineh/android/lineage/out/target/product/bullhead/ker  
nel', needed by '/home/amineh/android/lineage/out/target/product/bullhead/b  
oot.img', missing and no known rule to make it  
build/core/ninja.mk:151: recipe for target 'ninja_wrapper' failed  
make: *** [ninja_wrapper] Error 1  
make: Leaving directory '/home/amineh/android/lineage'  
#### make failed to build some targets (01:33 (mm:ss)) ####  
amineh@amineh-VirtualBox:~/android/lineage$
```

که برای حل آن از دستور زیر استفاده شد:

```
export USE_NINJA='false'  
export USE_NINJA=false
```

پس از دومین اجرا به مشکل زیر برخوردم:

```
amineh@amineh-VirtualBox: ~/android/lineage
EXECUTABLES/acp_intermediates/export_includes
mv /home/amineh/android/lineage/out/host/linux-x86/obj/EXECUTABLES/acp_intermediates/export_includes.tmp /home/amineh/android/lineage/out/host/linux-x86/obj/EXECUTABLES/acp_intermediates/export_includes ;
host Executable: acp (/home/amineh/android/lineage/out/host/linux-x86/obj/EXECUTABLES/acp_intermediates/acp)
Notice file: external/compiler-rt/NOTICE -- /home/amineh/android/lineage/out/host/linux-x86/obj/NOTICE_FILES/src//lib64/libcompiler_rt-extras.a.txt
Notice file: external/compiler-rt/NOTICE -- /home/amineh/android/lineage/out/host/linux-x86/obj/NOTICE_FILES/src//lib/libcompiler_rt-extras.a.txt
Notice file: external/compiler-rt/NOTICE -- /home/amineh/android/lineage/out/host/windows-x86/obj/NOTICE_FILES/src//lib/libcompiler_rt-extras.a.txt
Notice file: external/compiler-rt/NOTICE -- /home/amineh/android/lineage/out/host/windows-x86/obj/NOTICE_FILES/src//lib64/libcompiler_rt-extras.a.txt
Install: /home/amineh/android/lineage/out/host/linux-x86/bin/acp
host Prebuilt: mkbootimg (/home/amineh/android/lineage/out/host/linux-x86/obj/EXECUTABLES/mkbootimg_intermediates/mkbootimg)
Install: /home/amineh/android/lineage/out/host/linux-x86/bin/mkbootimg
make: *** No rule to make target '/home/amineh/android/lineage/out/target/product/bullhead/kernel', needed by '/home/amineh/android/lineage/out/target/product/bullhead/boot.img'. Stop.
make: Leaving directory '/home/amineh/android/lineage'

#### make failed to build some targets (03:38 (mm:ss)) ####
Trash -VirtualBox:~/android/lineage$
```

و برای حل آن از دستور زیر استفاده کردم:

```
export JACK_SERVER_VM_ARGUMENTS="-Dfile.encoding=UTF-8 -XX:+TieredCompilation -Xmx4g"
./prebuilts/sdk/tools/jack-admin kill-server
./prebuilts/sdk/tools/jack-admin start-server
```

متأسفانه پس از تلاش‌های پی در پی مشکلی در این مرحله باقی‌ماند و هر بار پس از چندین ساعت build کردن دوباره پیغامی مشابه چاپ می‌شد.

- 1) https://en.wikibooks.org/wiki/Grsecurity/Configuring_and_Installing_grsecurity
- 2) <https://medium.freecodecamp.org/building-and-installing-the-latest-linux-kernel-from-source-6d8df5345980>
- 3) <http://tldp.org/LDP/lkmpg/2.6/html/lkmpg.html#AEN121>
- 4) <https://wiki.lineageos.org/devices/bullhead/build#initialize-the-lineageos-source-repository>