**CSE 508 Term Project Report**

**Spring, 2014**

**Nftables on Android**

**Team Members**

**Subhasis Das , 108996931**

**Gaurav Piyush , 108996990**

**Chaitanya Saxena , 108346525**

**Introduction**

This course project aims to bring nftables, one of the latest advancements in the Linux

kernel, to the most popular Linux distribution - Android

**Definition**

Nftables[1] is a project providing packet filtering and packet classification on [Linux](http://en.wikipedia.org/wiki/Linux), and it is intended to replace existing [*iptables*](http://en.wikipedia.org/wiki/Iptables),*ip6tables*, [*arptables*](http://en.wikipedia.org/wiki/Arptables) and *ebtables* frameworks. nftables is a combination of a [Linux kernel](http://en.wikipedia.org/wiki/Linux_kernel) engine, and a [userspace](http://en.wikipedia.org/wiki/Userspace) utility.

It utilizes the building blocks of the [Netfilter](http://en.wikipedia.org/wiki/Netfilter) infrastructure, such as the existing hooks, connection tracking system, userspace queueing component, and logging subsystem. Also, there is a planned compatibility layer for the translation of already existing iptables firewall rules into their nftables equivalents

This report lists down a brief summary of the tasks we have attempted and the step by step details of how we have implemented them. We have also listed the tasks that work and the ones which we could not implement.

This report is divided into following parts:

Task 1:

* Section 1: Setting up and running the emulator
* Section 2: Building LIBMNL for android and testing it
* Section 3: Building LIBNFTNL for android and testing it
* Section 4: Building LIBGMP for android and testing
* Section 5: Building libreadline for android and testing
* Section 6: Building libncurses for Android and testing
* Section 7: Building nftables for Android and testing

Task 2:

Create a new action, which replaces the destination (i.e., redirection) and/or source fields (i.e., source spoofing) of matched packets with given addresses

**Tasks partially finished**

Task 3:

Create a new packet selector that matches "sender" and "receiver" of a packet with a given app name (or fully qualified ID)

**Task 1: Port nftables (both kernel-level modules and user-space utilities) and its required kernel support to Android  
  
  
Section 1: Setting up and running the emulator :**

Please refer to Appendix 1: Setting up emulator for details

**Section 2:** **Building LIBMNL for android and testing it:**

1. Set up Android NDK: wget -c http://dl.google.com/android/ndk/android-ndk-r9b-linux-x86\_64.tar.bz2 and extract the files.
2. Next we need to create the toolchain for X86 :

./android-ndk-r9b/build/tools/make-standalone-toolchain.sh --platform=android-19 --install-dir=/home/Desktop/Net-Sec/ndk\_standalone --ndk-dir=./android-ndk-r9b --toolchain=x86-4.6

1. Set environment variables to ensure the new compilers are known :

export PATH=/home/gaurav/Desktop/Net-Sec/ndk\_standalone/bin:$PATH

1. git clone git://git.netfilter.org/libmnl
2. Run ./autogen.sh : this generates the executable file “configure”
3. Executing configure creates the make file

Run ./configure --host=i686-linux-android --prefix=/home/gaurav/Desktop/Net-Sec/ndk\_standalone/

1. Run make file

make CC=/home/gaurav/Desktop/Net-Sec/ndk\_standalone/bin/i686-linux-android-gcc

1. Run make install

make CC=/home/gaurav/Desktop/Net-Sec/ndk\_standalone/bin/i686-linux-android-gcc install

Errors Encountered and fixed :

: ../include/linux/netlink.h:35:2: error: unknown type name '\_\_kernel\_sa\_family\_t' make[2]: \*\*\* [socket.lo] Error 1 make[2]: Leaving directory

Resolution: Replace\_\_kernel\_sa\_family\_t to 'sa\_family\_t' instead

These steps results in libmnl being installed. Now we need to test it:

Testing step:

1. Create a test user function libmnl\_test.c to test libmnl libraries
2. Make :

make CC=/home/gaurav/Desktop/Net-Sec/ndk\_standalone/bin/i686-linux-android-gcc CFLAGS=--sysroot=/home/gaurav/Desktop/Net-Sec/ndk\_standalone/sysroot LD\_LIBRARY\_PATH=-I/home/gaurav/Desktop/Net-Sec/libmnl/include/ LDFLAGS="-L/home/gaurav/Desktop/Net-Sec/ndk\_standalone/lib -lmnl" libmnl\_test

1. Make the file system writable : mount -o rw,remount /system
2. Push all the \*.so files from ndk\_standalone/lib to /system/lib in android file system
3. Now run the test command

/home/gaurav/Desktop/Net-Sec/ndk\_standalone/bin/i686-linux-android-gcc --sysroot=/home/gaurav/Desktop/Net-Sec/ndk\_standalone/sysroot -I/home/gaurav/Desktop/Net-Sec/libmnl/include/ -L/home/gaurav/Desktop/Net-Sec/ndk\_standalone/lib -lmnl  libmnl\_test.c -o libmnl\_test

1. Push the test files after successful compilation
2. Next, in adb shell : /data # ./libmnl\_test

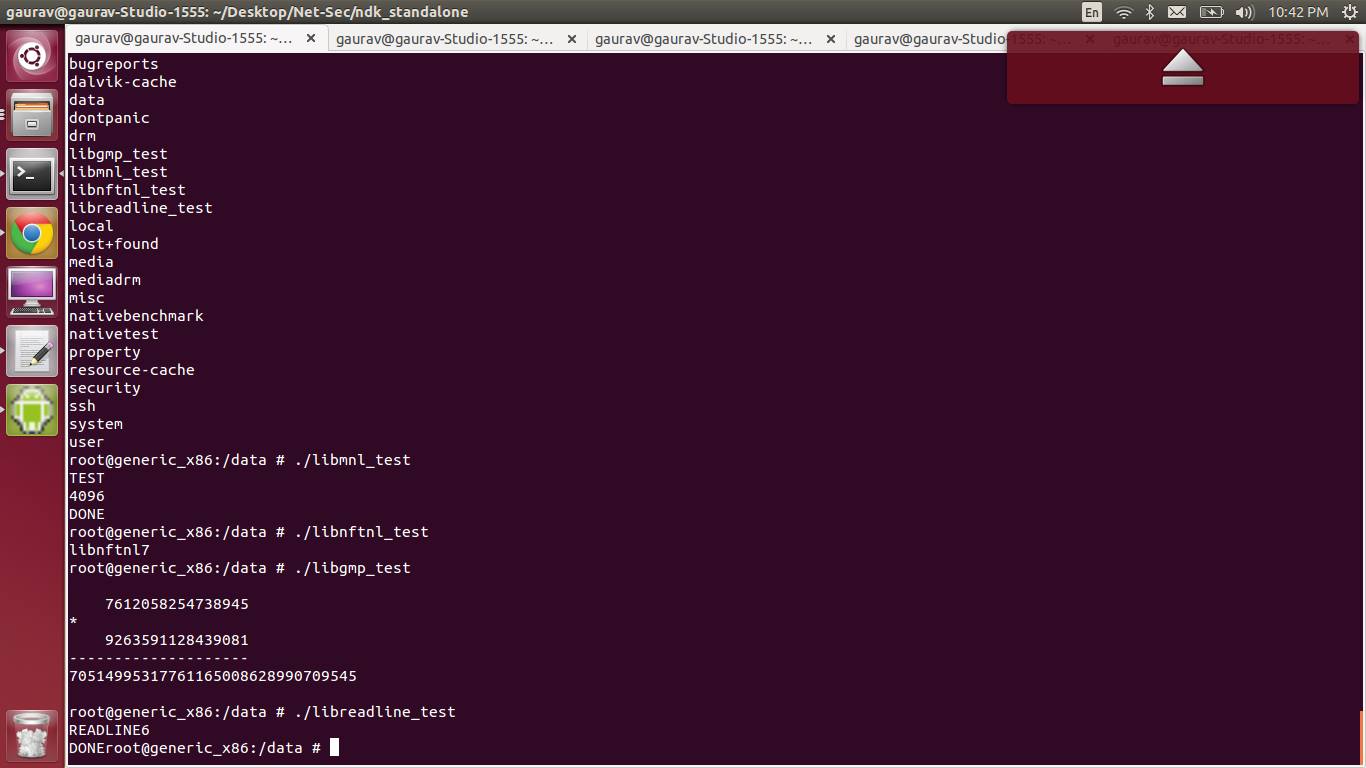


Figure 1 : LIBNFTL , LIBMNL , READLINE , LIBGMP test success

**Section 3:** **Building LIBNFTL for android and testing it:**

1. Git clone : <http://git.netfilter.org/libnftnl/>
2. Run ./autogen.sh : this generates the executable file “configure”
3. Set the following environment variables

export LIBMNL\_CFLAGS=-I/home/gaurav/Desktop/Net-Sec/libmnl/include

export LIBMNL\_LIBS=-L/home/gaurav/Desktop/Net-Sec/ndk\_standalone/lib

1. Executing configure creates the make file

CC=/home/gaurav/Desktop/Net-Sec/ndk\_standalone/bin/i686-linux-android-gcc ./configure --host=i686-linux-android --prefix=/home/gaurav/Desktop/Net-Sec/ndk\_standalone/

1. Run make

make CC=/home/gaurav/Desktop/Net-Sec/ndk\_standalone/bin/i686-linux-android-gcc LDFLAGS="-L/home/gaurav/Desktop/Net-Sec/ndk\_standalone/lib/ -lmnl"

1. Run make install:

make CC=/home/gaurav/Desktop/Net-Sec/ndk\_standalone/bin/i686-linux-android-gcc LDFLAGS="-L/home/gaurav/Desktop/Net-Sec/ndk\_standalone/lib/ -lmnl" install

Errors/Challenges faced and resolved:

1. One header file nfnetlink\_compat.h needs to be manually added in include/linux/netfilter. (Source : [5])
2. Fix some warnings such as :

chain.c:364:3: warning: implicit declaration of function 'be64toh' [-Wimplicit-function-declaration]

Resolution : add the macro #define be64toh(x) betoh64(x) in affected files

LIBNFTL TEST:

1. Create a user test file libnftnl\_test.c to test libnftl libraries
2. Run

make CC=/home/gaurav/Desktop/Net-Sec/ndk\_standalone/bin/i686-linux-android-gcc LDFLAGS="-L/home/gaurav/Desktop/Net-Sec/ndk\_standalone/lib/ -lmnl" install

1. Make the file system writable : mount -o rw,remount /system
2. Push all the \*.so files from ndk\_standalone/lib to /system/lib in android file system
3. Run

/home/gaurav/Desktop/Net-Sec/ndk\_standalone/bin/i686-linux-android-gcc --sysroot=/home/gaurav/Desktop/Net-Sec/ndk\_standalone/sysroot -I/home/gaurav/Desktop/Net-Sec/libnftnl/include -I/home/gaurav/Desktop/Net-Sec/libmnl/include -L/home/gaurav/Desktop/Net-Sec/ndk\_standalone/lib -lnftnl -lmnl libnftnl\_test.c -o libnftnl\_test

1. Push the test files after successful compilation
2. Next, in adb shell : /data section run

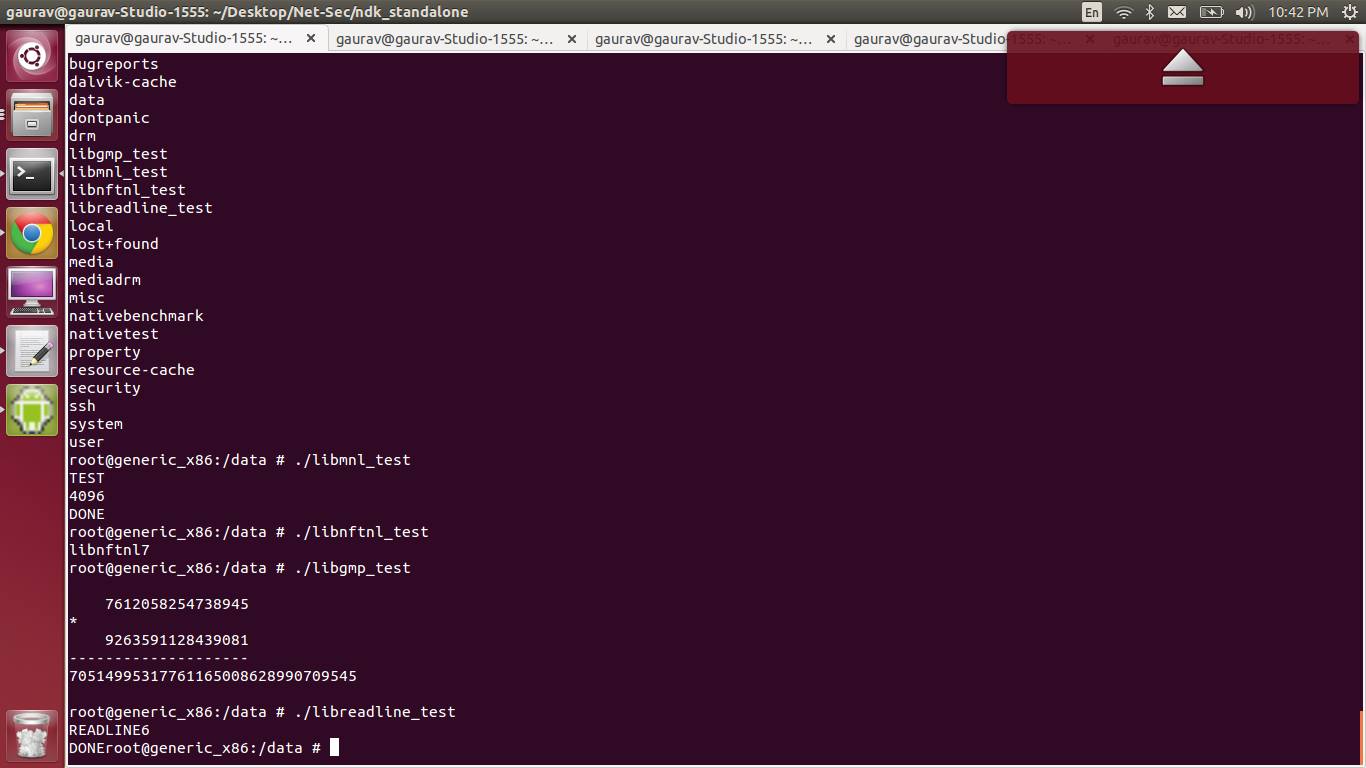


Figure 2 : LIBNFTL , LIBMNL , READLINE , LIBGMP test success

**Section 4: Building libgmp for android**

As discussed in the class forum, we got the files from  <https://github.com/Rupan/gmp>

Testing LIBGMP :

1. Create the test file libgmp\_test.c to test libgmp libraries
2. Run

/home/gaurav/Desktop/Net-Sec/ndk\_standalone/bin/i686-linux-android-gcc --sysroot=/home/gaurav/Desktop/Net-Sec/ndk\_standalone/sysroot -I/home/gaurav/Desktop/Net-Sec/libgmp/include -L/home/gaurav/Desktop/Net-Sec/ndk\_standalone/lib -lgmp libgmp\_test.c -o libgmp\_test

1. Follow the steps mentioned above to run and test the output in the adb shell

**Section 5: Building libreadline for android**

1. Download libreadline-6.2 from <http://ftp.gnu.org/gnu/readline/> { please note that we tried running libreadline- 6.3 but after failing to resolve some issues and talking to the Prof, we decided to go with the older but stable version}
2. Copy the 2 files config.sub and config.guess into support folder. Currently, libreadline's configure is pretty outdated and does not support cross-compiling
3. Configure step

CC=/home/gaurav/Desktop/Net-Sec/ndk\_standalone/bin/i686-linux-android-gcc ./configure --host=i686-linux-android --prefix=/home/gaurav/Desktop/Net-Sec/ndk\_standalone

1. Make

make CC=/home/gaurav/Desktop/Net-Sec/ndk\_standalone/bin/i686-linux-android-gcc LDFLAGS="-L/home/gaurav/Desktop/Net-Sec/ndk\_standalone/lib/ -lreadline"

1. Make install

make CC=/home/gaurav/Desktop/Net-Sec/ndk\_standalone/bin/i686-linux-android-gcc LDFLAGS="-L/home/gaurav/Desktop/Net-Sec/ndk\_standalone/lib/ -lreadline" install

Testing:

1. Create test file libbreadline\_test.c
2. Run

/home/gaurav/Desktop/Net-Sec/ndk\_standalone/bin/i686-linux-android-gcc --sysroot=/home/gaurav/Desktop/Net-Sec/ndk\_standalone/sysroot -I/home/gaurav/Desktop/Net-Sec/ndk\_standalone/include -L/home/gaurav/Desktop/Net-Sec/ndk\_standalone/lib -lreadline -lncurses  libreadline\_test.c -o libreadline\_test

1. Send the output files to adb [system/lib]
2. Test the files in adb shell : /data # libreadline\_test

**Section 6: Building libncurses for android**

1. Download link : <https://code.google.com/p/android-cruft/downloads/detail?name=ncurses-5.7-built-for-android.tar.gz&can=2&q=>
2. Some issues fixed while doing make: Error : HAVE\_LOCALE\_H  not defined

Resolution : in ncurses\_cfg.h set #define HAVE\_LOCALE\_H 0

1. Run : cp /usr/share/misc/config.{sub,guess}
2. Configure

CC=/home/gaurav/Desktop/Net-Sec/ndk\_standalone/bin/i686-linux-android-gcc ./configure --host=i686-linux-android --prefix=/home/gaurav/Desktop/Net-Sec/ndk\_standalone --without-cxx-binding

1. Make

make CC=/home/gaurav/Desktop/Net-Sec/ndk\_standalone/bin/i686-linux-android-gcc LDFLAGS="-L/home/gaurav/Desktop/Net-Sec/ndk\_standalone/lib/ -lncurses" install

Testing ncurses : libncurses libraries are used by libreadline, so the test case remains the same as libreadline

**Section 7: Building nftables for android**

1. Git clone git://git.netfilter.org/nftables
2. Set export ac\_cv\_header\_libintl\_h=yes
3. Run ./autogen.sh
4. Configure step

CC=/home/gaurav/Desktop/Net-Sec/ndk\_standalone/bin/i686-linux-android-gcc ./configure --host=i686-linux-android --prefix=/home/gaurav/Desktop/Net-Sec/ndk\_standalone/ CFLAGS="--sysroot=/home/gaurav/Desktop/Net-Sec/ndk\_standalone/sysroot -I/home/gaurav/Desktop/Net-Sec/libmnl/include/ -I/home/gaurav/Desktop/Net-Sec/libnftnl/include/ -I/home/gaurav/Desktop/Net-Sec/libgmp/x86 -I/home/gaurav/Desktop/Net-Sec -I/home/gaurav/Desktop/Net-Sec/ndk\_standalone/include" LDFLAGS="-L/home/gaurav/Desktop/Net-Sec/ndk\_standalone/sysroot/usr/lib -L/home/gaurav/Desktop/Net-Sec/ndk\_standalone/lib -lmnl -lnftnl -lreadline -lgmp -nostdlib"

1. Make step

make CC=/home/gaurav/Desktop/Net-Sec/ndk\_standalone/bin/i686-linux-android-gcc LDFLAGS="-L/home/gaurav/Desktop/Net-Sec/ndk\_standalone/sysroot/usr/lib/ -L/home/gaurav/Desktop/Net-Sec/ndk\_standalone/lib -lmnl -lnftnl -lgmp -lreadline -lncurses"

Errors/Challenges faced:

|  |  |
| --- | --- |
| **ISSUES/ERRORS** | **RESOLUTIONS** |
| C compiler cannot create executable | run configure with -nostdlib option |
| error header file not found libintl.h | set env var : export ac\_cv\_header\_libintl\_h=yes |
| error "NFT\_META\_BRI\_IIFNAME" undeclared in meta.c  error: No suitable version of flex found | apt-get remove flex |

Testing nftable

1. adb push src/nft /sbin/
2. Type : nft –i //use interactive mode

Opens the nft named set

**Task 2: Create a new action, which replaces the destination (i.e., redirection) and/or source fields**

IPV4 NAT EXAMPLES

1. Setting IPfilter

nft add rule ip filter output ip daddr 1.2.3.4 drop

1. Setting rule on a network

nft add rule ip filter output ip daddr 172.24.241.220/24 counter

1. Rule to drop packets to port 80

nft add rule ip filter input tcp dport 80 drop

1. We have tried to create a NAT dedicated chain referring the source [6]

(Prerequisite: check that NAT module is present)

Commands

* nft add table nat
* nft add chain nat post { type nat hook postrouting priority 0;}
* nft add chain nat pre { type nat hook prerouting priority 0;}

Notes:

This creates a dedicated NAT chain

1. Adding nat rules :

Commands:

* nft add rule nat post ip saddr 172.24.241.220/24 oif snat 172.24.20.261"

This command NATS all trafic from 172.24.241.220/24 outgoing to wlan0 interface to the IP 172.24.20.261

* nft add rule nat pre udp dport 53 ip saddr 172.24.241.220/24 dnat 8.8.8.8:53

This command redirects all DNS trafic from 172.24.241.220/24 to the 8.8.8.8 server(Google public DNS).

**Task 3: Create a new packet selector that matches "sender" and "receiver" of a packet with a given app name (or fully qualified ID)**

We tried the following successfully on a linux machine(yet to successfully test in out android)

Note : insert or add a rule at a specific position

# nft list table filter -n -a

table filter {

chain output {

table filter hook output priority 0;

ip protocol tcp counter packets 82 bytes 9680 # handle 8

ip saddr 127.0.0.1 ip daddr 127.0.0.6 drop # handle 7

}

}

**References**

1. <http://en.wikipedia.org/wiki/Nftables> : What is Netfilter:
2. <https://wiki.archlinux.org/index.php/Nftables> : A good basic documentation on usage and design of nftables.
3. <https://github.com/sam8dec/NetSec> : We have referred to this excellent write up by Samudra for our initial setup.
4. <http://en.wikipedia.org/wiki/GNU_Multiple_Precision_Arithmetic_Library#Example> : Excellent reference for writing test cases
5. Source for nfnetlink\_compat.h : <https://git.netfilter.org/libnetfilter_acct/tree/include/linux/netfilter>.
6. <http://kernelnewbies.org/nftables_examples> : Excellent examples of NAT rule handling