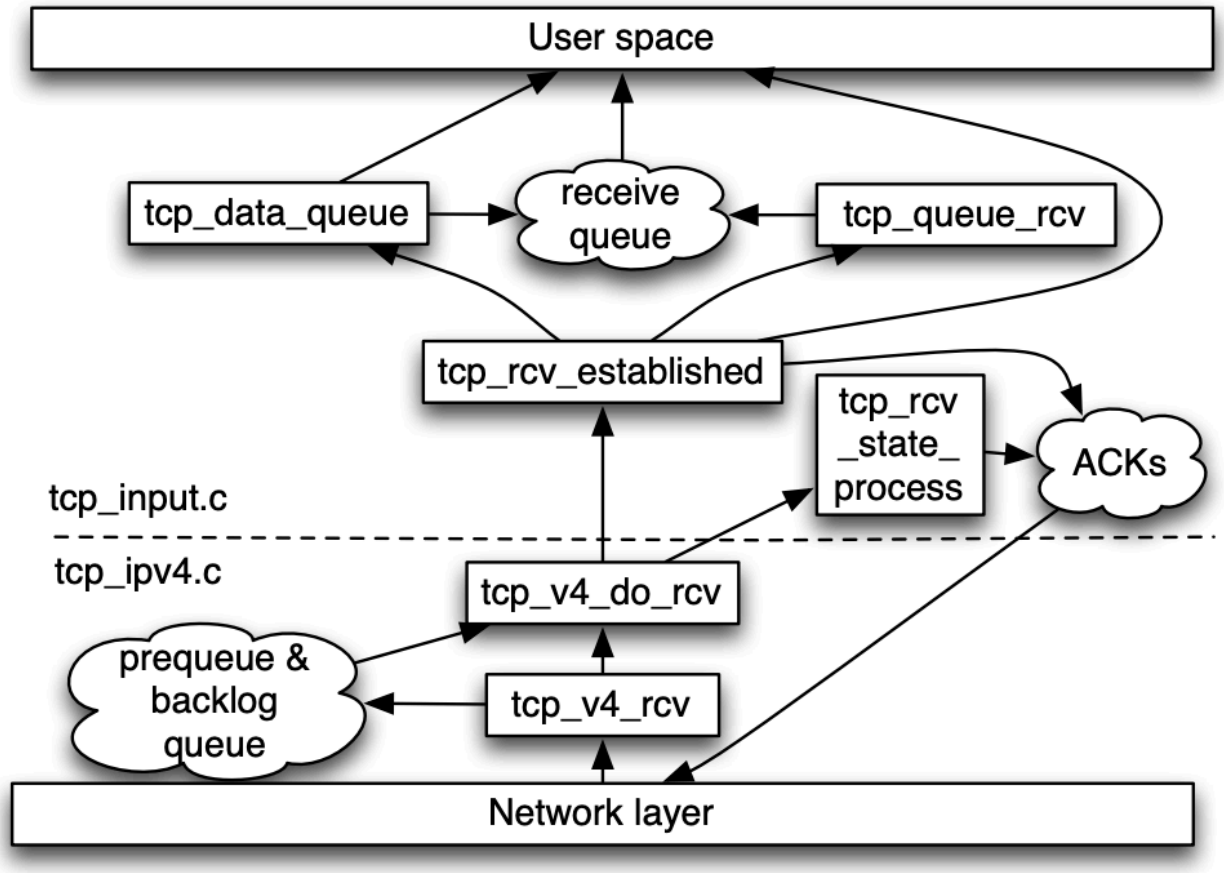
Linux Guideline for integrating TCP-AAD in network module

Introduction

The main purpose of the guideline to meet people with challenges during integrating TCP-AAD. inside the Linux Kernel and testing approaches of TCP stack in the Operating System.

Main functions during packet receive in TCP

In the beginning we should understand which functions responsible for receivement in TCP.



This image could be considered in depth by analysing the paper . As we focus on the behaviour after handshake occurs, we will consider `ESTABLISHED` state. The entry point will be `tcp_rcv_established` located in `net/ipv4/tcp_input.c` of Linux Kernel. Mainly the function splits the flow on `slow` and `fast` path.

The next important part of ACK is `tcp_event_data_recv` located in the same file as `tcp_rcv_established` and responsible for timer-calculation of DACK.
After time being scheduled we should notice function `__tcp_ack_snd_check` that checks either delayed ack or compressed ack, or quick ack should be sent.
The last function is `tcp_send_delayed_ack` that actually schedules the timer. When the timer expires there will be a callback function execution that could be find during timer initialisation at (initialisation of the `icsk_delack_timer` timer that could be find in `inet_csk_init_xmit_timers` function of `inet_connection_sock.c` file)

Important structure that is a key part under DACK is `inet_connection_sock`. Main responsibility is to contain timers for calculation and some extra information. For, example information about **IAT**. Also in details there should be considered initialisation part of tcp and inet sockets in `tcp.c` called `tcp_init_sock`.

Building process of Linux Kernel

Detailed overview of the installation part could be find here

Before just install important packets:

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

Package	Package description
git	Tracks and makes a record of all changes during development in the source code. It also allows reverting the changes.
fakeroot	Creates the fake root environment.
build-essential	Installs development tools such as **C**, **C++**, gcc, and g++.
ncurses-dev	Provides **API** for the text-based terminals.
xz-utils	Provides fast **file compression** and **file decompression**.
libssl-dev	Supports **SSL and TSL** that **encrypt** data and make the internet connection secure.
bc (Basic Calculator)	Supports the interactive execution of statements.
flex (Fast Lexical Analyzer Generator)	Generates lexical analyzers that convert characters into tokens.
libelf-dev	Issues a shared library for managing ELF files (executable files, core dumps and object code)
bison	Converts grammar description to a C program.

After that go to the source code of linux kernel and copy existing config on your machine:

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

If you want to change something in configuration use the command:

```
make menuconfig
```

After configuration has been completed you should build the kernel it could be performed by next instructions:

```
scripts/config --disable SYSTEM_TRUSTED_KEYS
scripts/config --disable SYSTEM_REVOCATION_KEYS

fakeroot make -j$(nproc)
```

After building is complete install the kernel by executing commands:

```
sudo make modules_install
sudo make install
```

```
marko@pnap:~/linux-6.0.7$ sudo make install
sh ./arch/x86/boot/install.sh 6.0.7 arch/x86/boot/bzImage \
        System.map "/boot"
run-parts: executing /etc/kernel/postinst.d/apt-auto-removal 6.0.7 /boot/vmlinuz-6.0.7
run-parts: executing /etc/kernel/postinst.d/dkms 6.0.7 /boot/vmlinuz-6.0.7
 * dkms: running auto installation service for kernel 6.0.7                  [ OK ]
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 6.0.7 /boot/vmlinuz-6.0.7
update-initramfs: Generating /boot/initrd.img-6.0.7
run-parts: executing /etc/kernel/postinst.d/update-notifier 6.0.7 /boot/vmlinuz-6.0.7
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 6.0.7 /boot/vmlinuz-6.0.7
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
done
marko@pnap:~/linux-6.0.7$
```

After that you will see the kernel image (in the picture is 6.0.7) and you can perform reboot step to see GRUB menu:

```
sudo reboot
```

If you do not see GRUB menu follow this

After reboot step you can by yourself define the version of the kernel by using next command:

```
uname -r
```

If there was some debug message we could find them by:

```
sudo dmesg -w
```

This command responsible for writing exploring messages.

Some short tips

- Jiffies - value for most timers in Linux Kernel that is basically 1ms for 1jiffy.
- `pr_info` - command responsible for writing message in logging page of Linux Kernel.
- Basic time resolution for hrtimer is nanoseconds.

Summary

By following instructions above you can completely meet with the workflow of modifying Linux Kernel and its subsequent building.