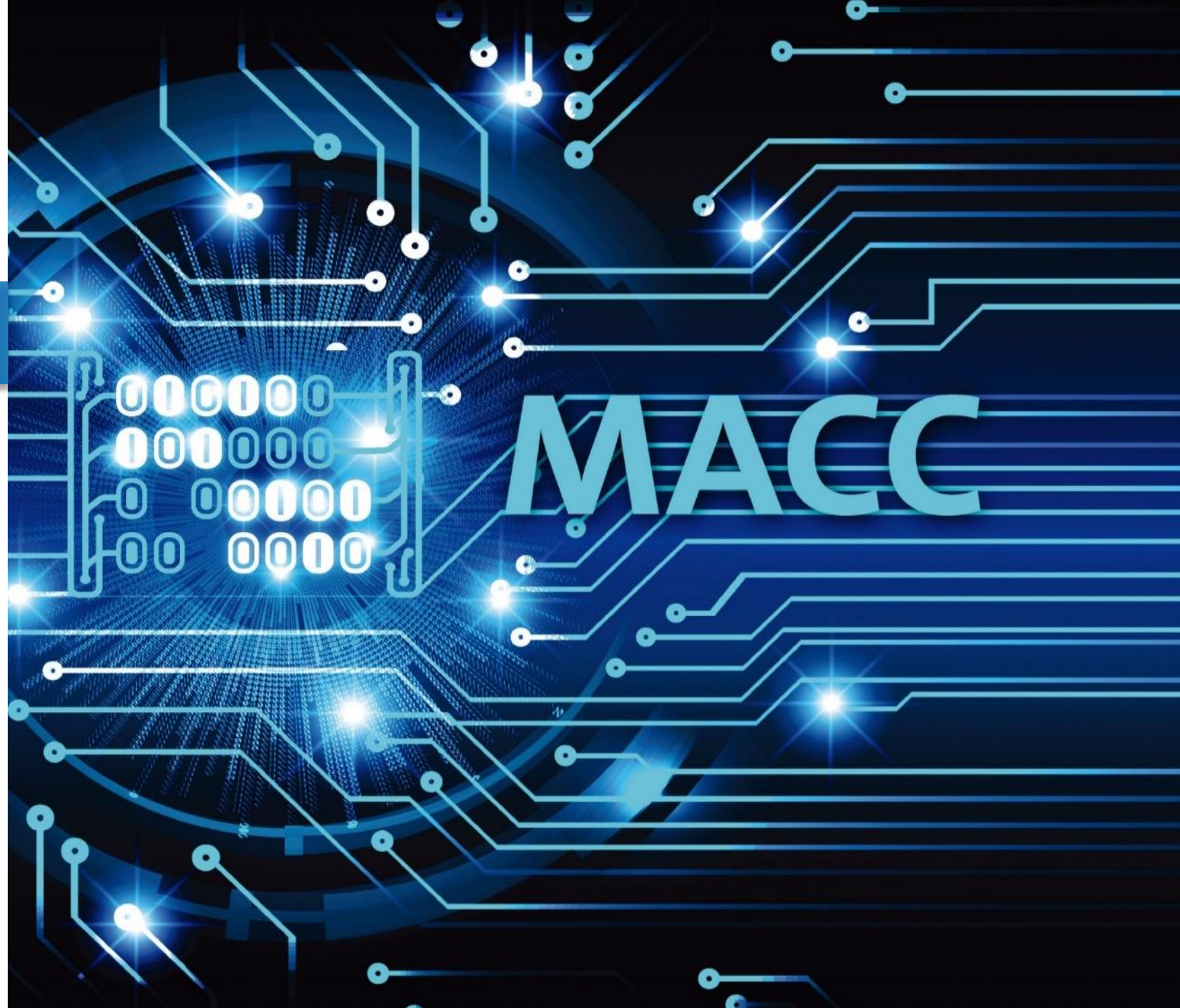
Universidad del Rosario

MACC
Matemáticas Aplicadas y Ciencias de la Computación

IoT Security

Hacking Ético



Daniel Orlando Díaz López, PhD

Profesor principal
Departamento MACC
Universidad del Rosario
danielo.diaz@urosario.edu.co

- DBI es inyectar código extraño en una aplicación existente para alterar su comportamiento
- Hacer DBI es diferente a hacer “explotación” o “Debugging”

- Por medio de DBI se puede:
  - Acceder a memoria
  - Sobrescribir funciones
  - Invocar funciones
  - Usar objetos
  - Interceptar funciones

- Usaremos la herramienta **FRIDA** para hacer DBI sobre aplicaciones Android
- Usaremos **Anbox** para emular un dispositivo móvil Android y evitar dañar un dispositivo real

Laboratorio

1. Instalar un emulador de dispositivos móviles Android: Anbox
2. Instalar la herramienta de hacking FRIDA

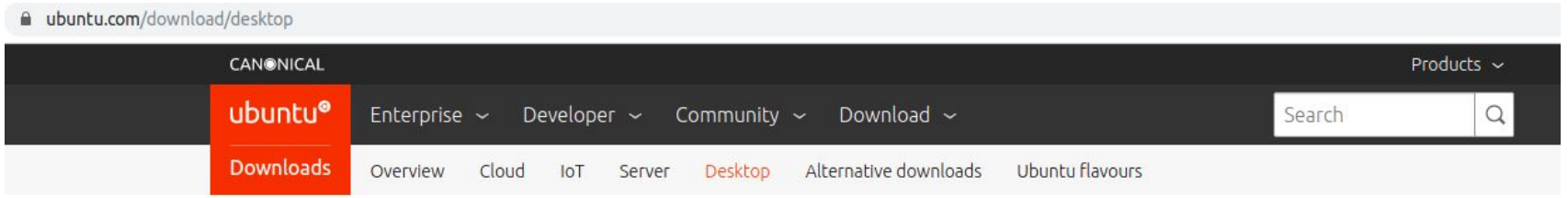
3. Instalar una aplicación móvil para Android y explotarla por medio de DBI

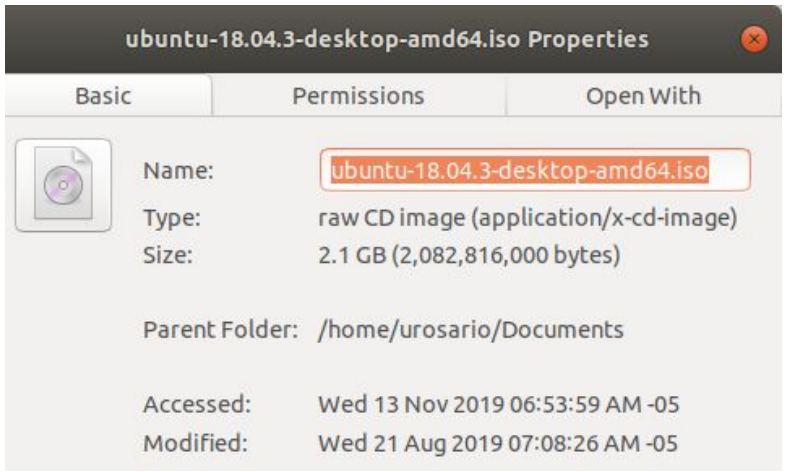
4. Documentar con capturas de pantalla cada uno de los pasos realizados
5. Responder las preguntas planteadas que se encuentran al final

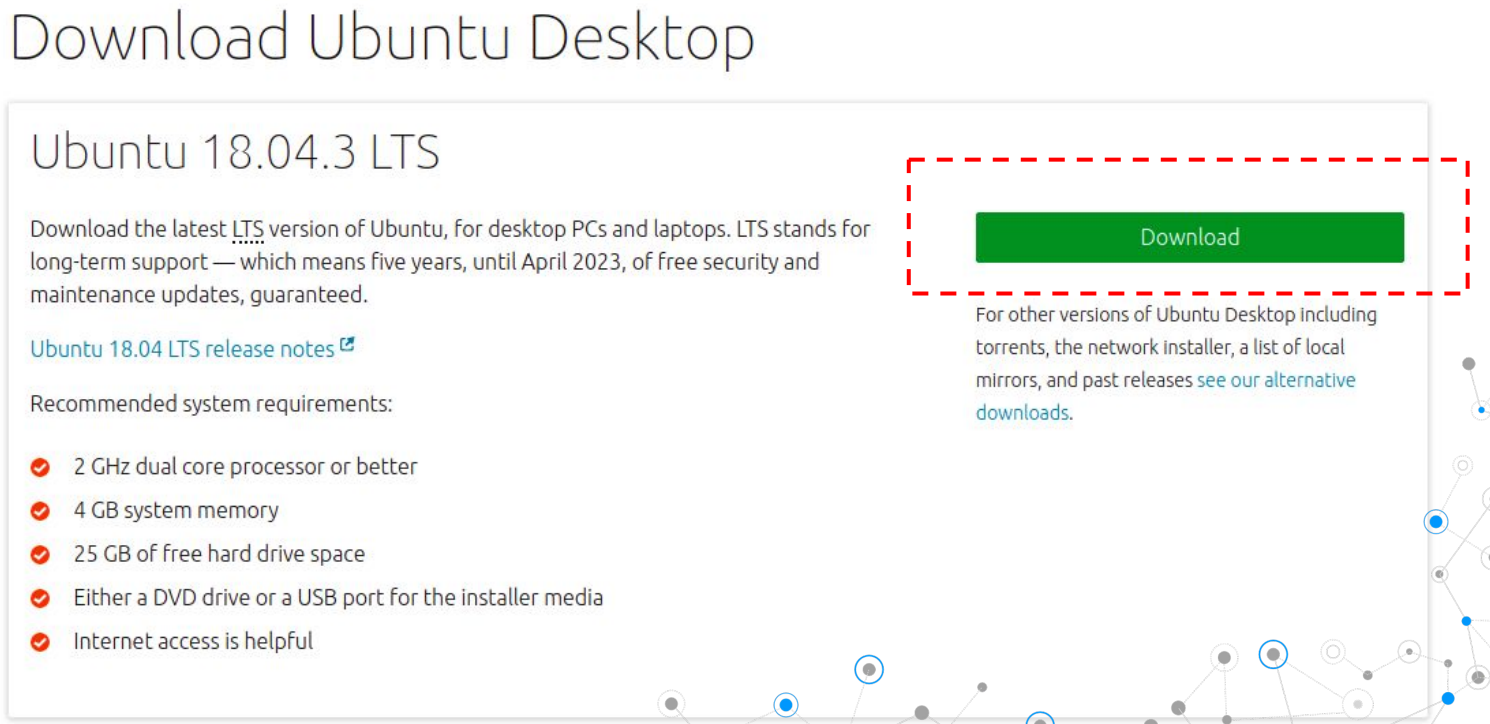
Iniciar Ubuntu 18.04 en modo live

Realizar diferentes tipos de ataques a una red inalámbrica

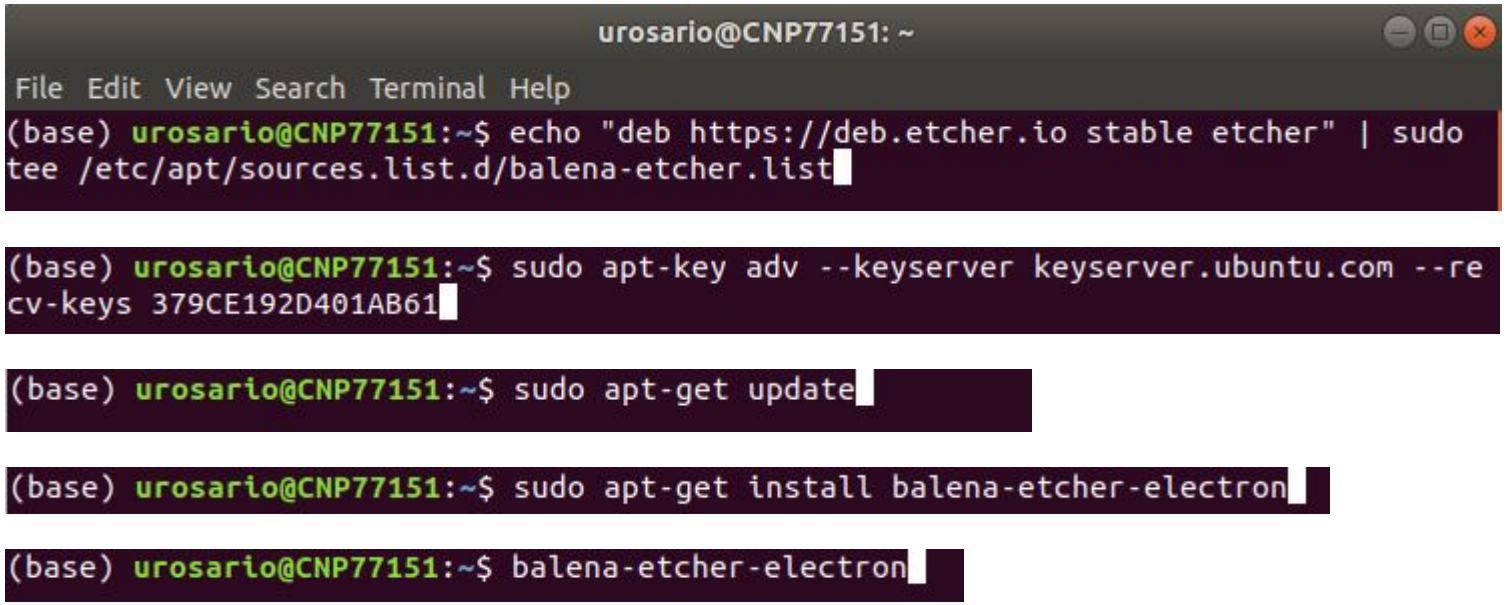
1. Descargar la última versión estable de Ubuntu en formato ISO







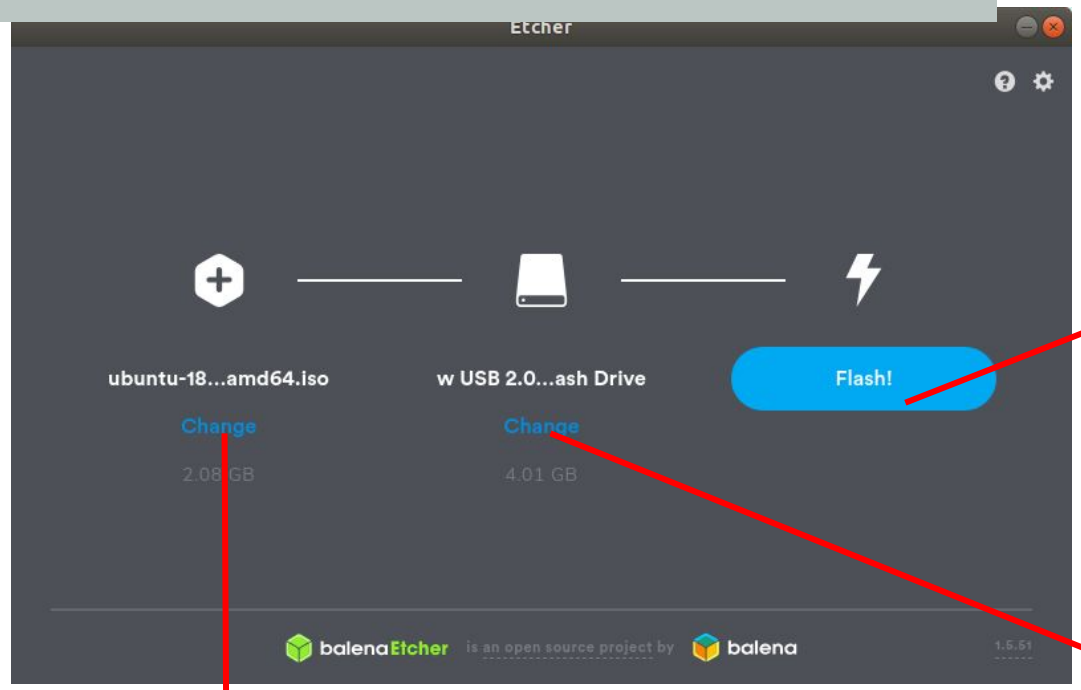
Crear una USB booteable usando como archivo .ISO, la imagen de Ubuntu que se acaba de descargar. Necesitamos una memoria USB de al menos 4 Gb!
Copiar comandos de instalación de aquí: https://github.com/balena-io/etcher



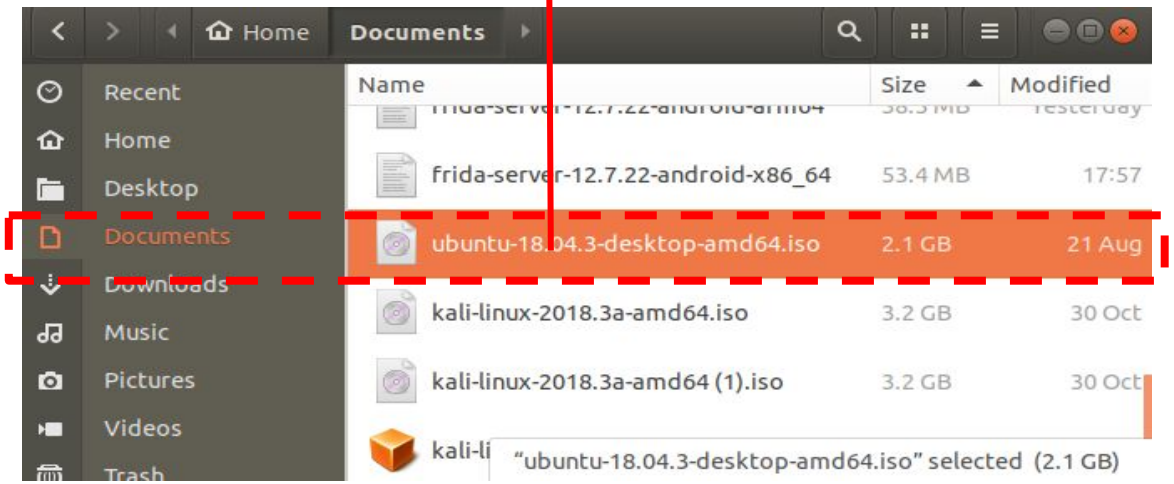
Seguir los pasos que se indican en el siguiente recurso
Desde un equipo Linux: : https://github.com/balena-io/etcher
Desde un equipo Windows: https://tutorials.ubuntu.com/tutorial/tutorial-create-a-usb-stick-on-windows#0

Dynamic Binary Instrumentation (DBI)

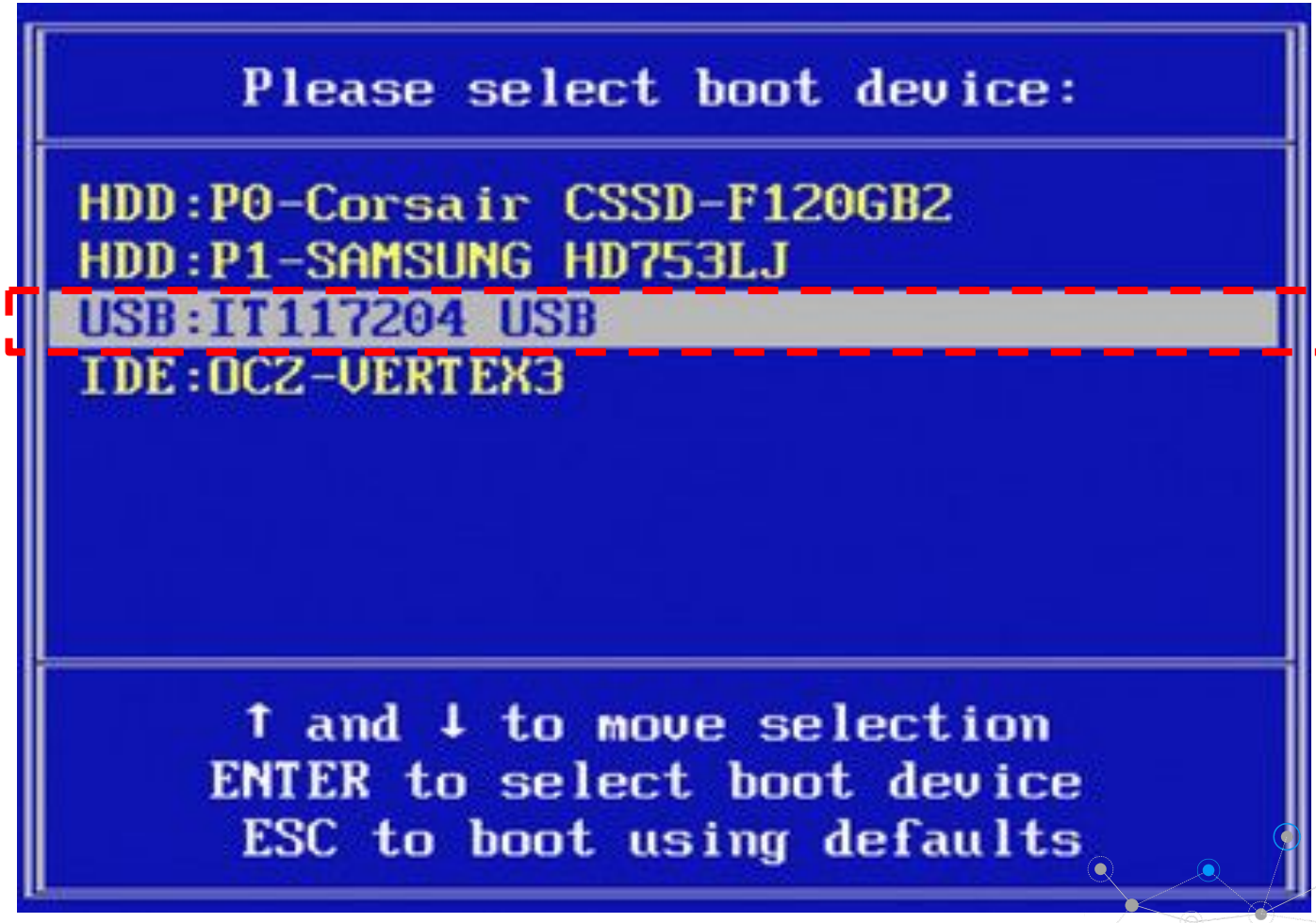
Universidad del Rosario | MACC Matemáticas Aplicadas y Ciencias de la Computación



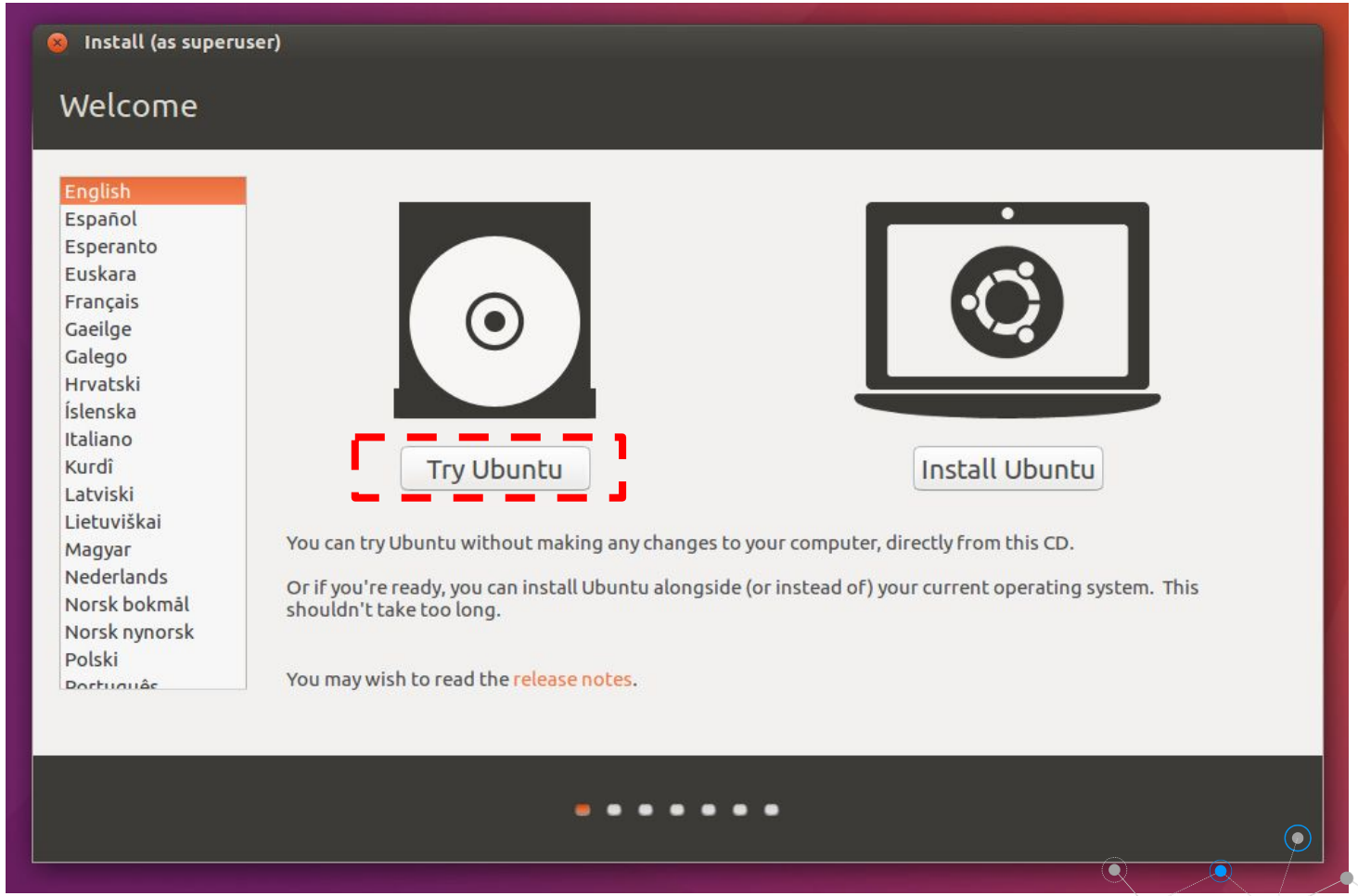
Este botón nos permite iniciar la creación de la memoria USB Bootable!



Reiniciar su computador (la memoria con Ubuntu debe estar puesta) y cuando el PC esté encendiendo presionar la tecla **F12** (dependiendo del modelo de su computador puede ser otra otra, sino funciona con F12 intentar con F2, F8, F10) para entrar la “Selección del dispositivo de arranque” y seleccionar la memoria USB en la que está instalado Ubuntu.



Iniciar Ubuntu en modo **LIVE CD** -> ¡Esto es como si tuvieramos Ubuntu instalado en el computador, pero tan pronto reiniciemos volveremos a tener nuestra instalación anterior y perderemos cualquier archivo guardado en Ubuntu!

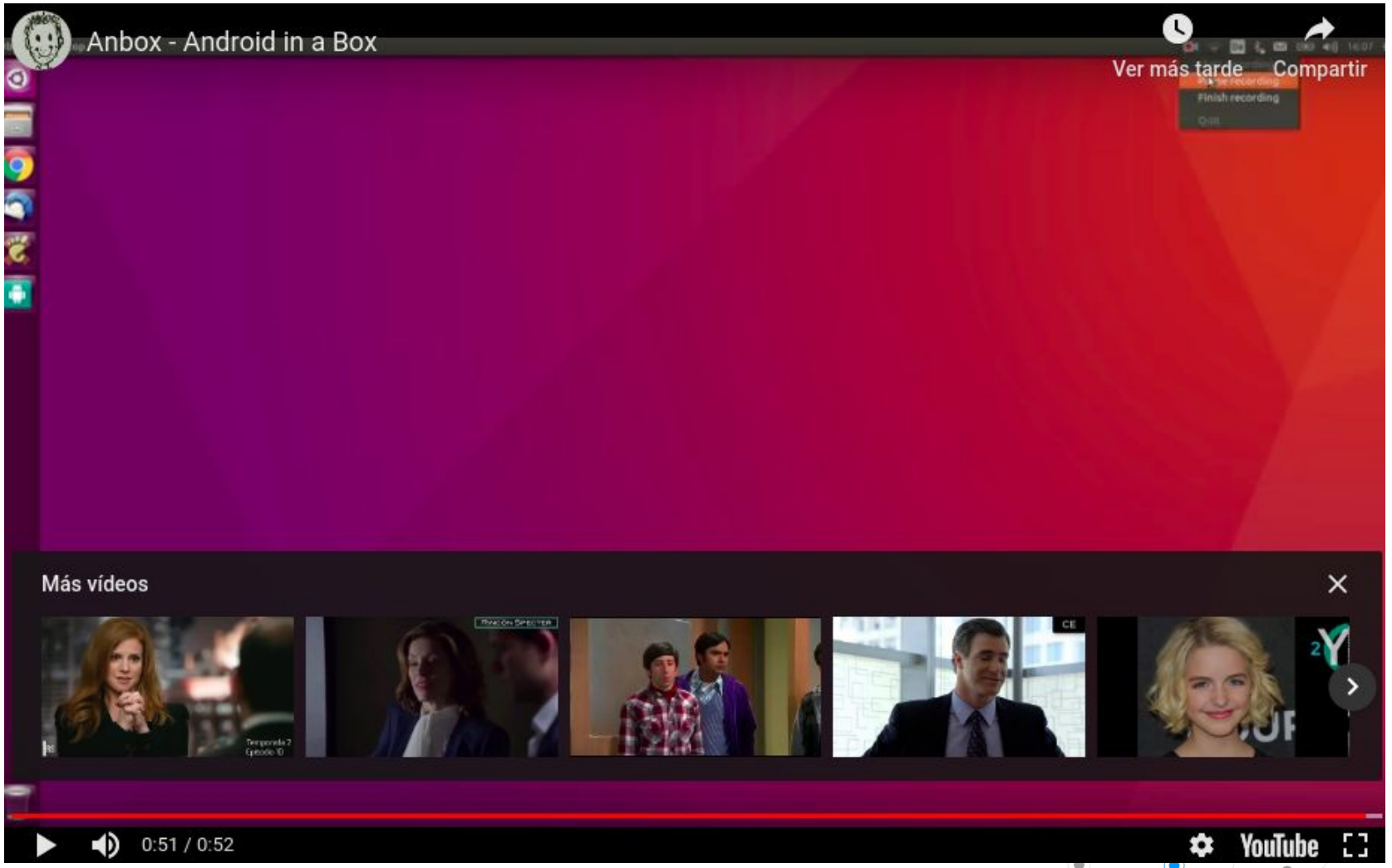


Quien desee realizar el laboratorio en los equipos de la universidad puede tener dificultades para instalar ETCHER (la herramienta que usamos para crear la USB booteable), por ello pueden venir a mi oficina y con gusto le puedo prestar alguna de las USB booteables que ya tengo preparadas con Ubuntu 18.04.

Anbox Emulator

ANBOX es la herramienta que utilizaremos para emular un dispositivo Android a atacar, así evitamos dañar un dispositivo real

- Introducción a Anbox



https://youtu.be/MbmiHnasGWg

- Instalar módulos necesarios por el kernel: https://docs.anbox.io/userguide/install_kernel_modules.html

```
$ sudo add-apt-repository ppa:morphis/anbox-support
$ sudo apt update
$ sudo apt install linux-headers-generic anbox-modules-dkms
```

- Iniciar los módulos del kernel requeridos para la virtualización

```
$ sudo modprobe ashmem_linux
$ sudo modprobe binder_linux
```

- Validar que los módulos del kernel hayan quedado cargados

```
(base) urosario@CNP77151:~$ ls -1 /dev/{ashmem,binder}
/dev/ashmem
/dev/binder
(base) urosario@CNP77151:~$
```

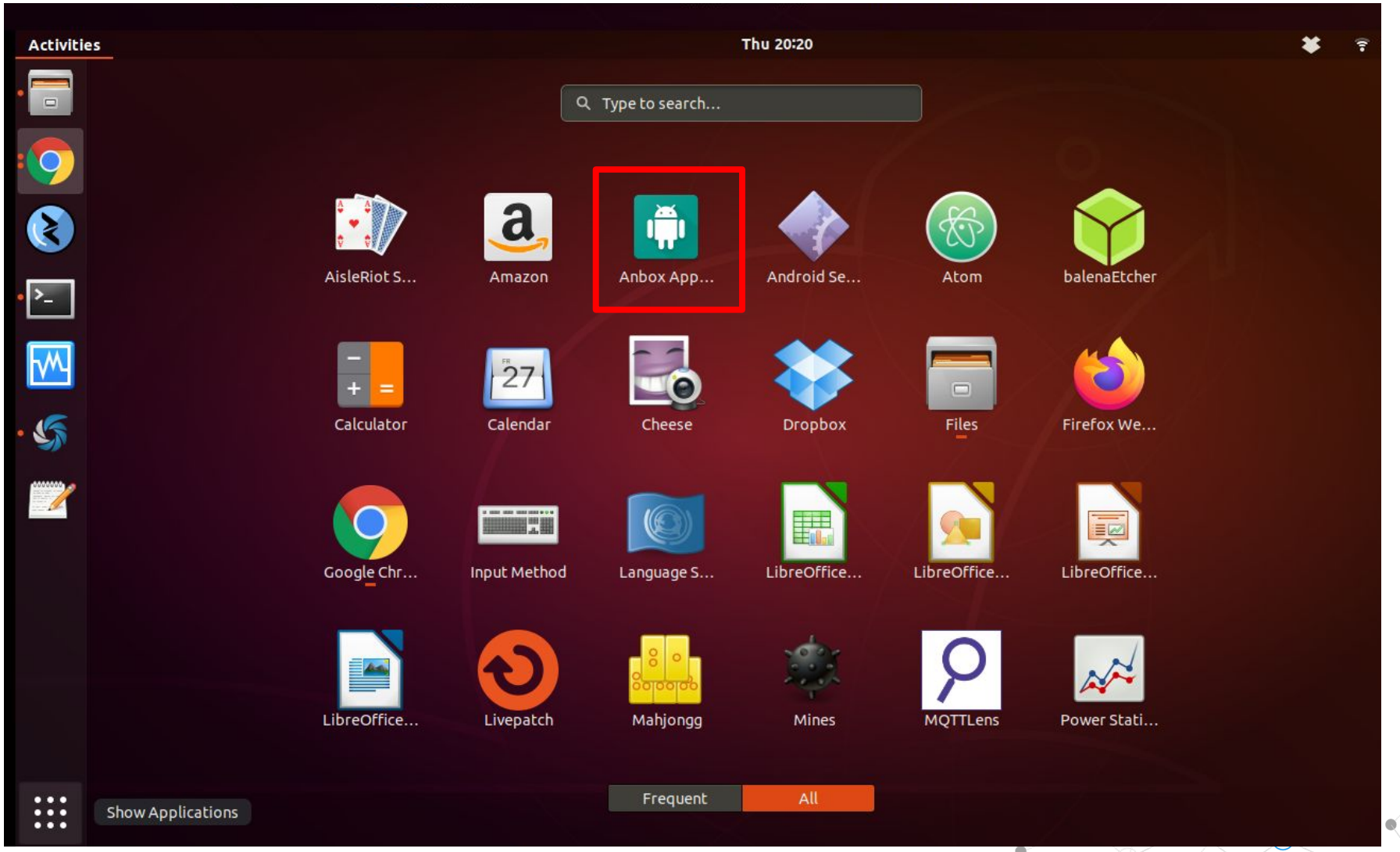
- Con los prerequisitos cumplidos en el anterior slide, ya podemos proceder a instalar Anbox. https://docs.anbox.io/userguide/install.html

```
$ snap install --devmode --beta anbox
```

- Validamos que la instalación haya sido correcta

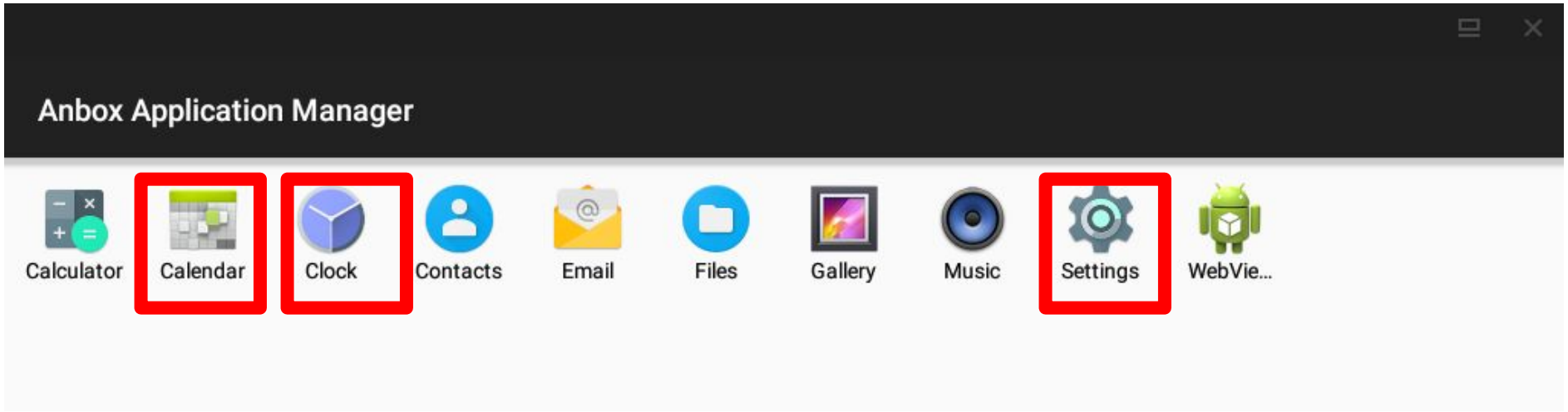
```
(base) urosario@CNP77151:~$ snap info anbox
name:      anbox
summary:   Android in a Box
publisher: morphis
contact:   https://anbox.io
license:   unset
description: |
  Runtime for Android applications which runs a full Android system
  in a container using Linux namespaces (user, ipc, net, mount) to
  separate the Android system fully from the host.
```

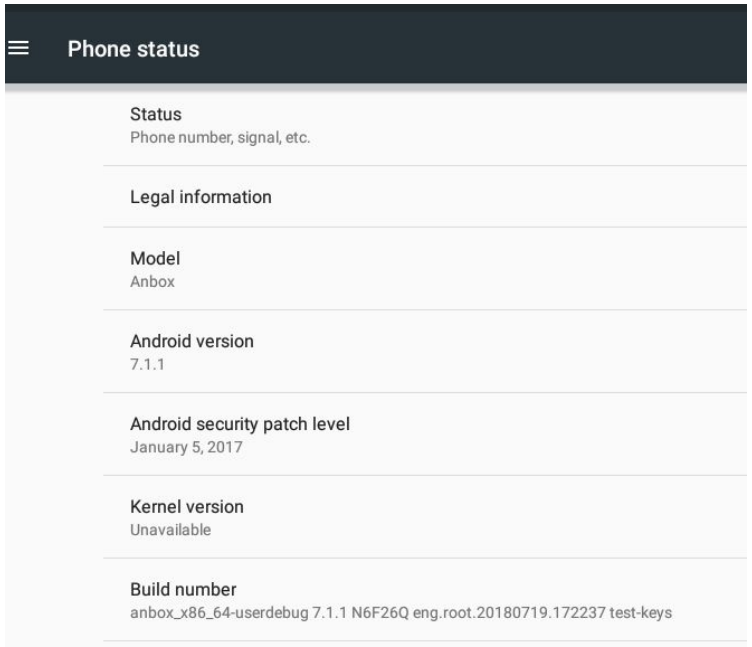
- Ahora ya podemos ejecutar Anbox

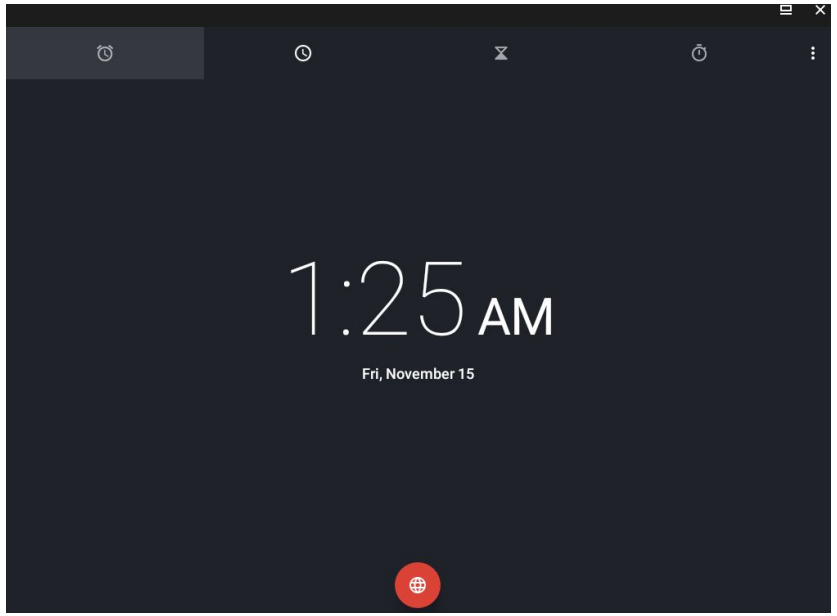


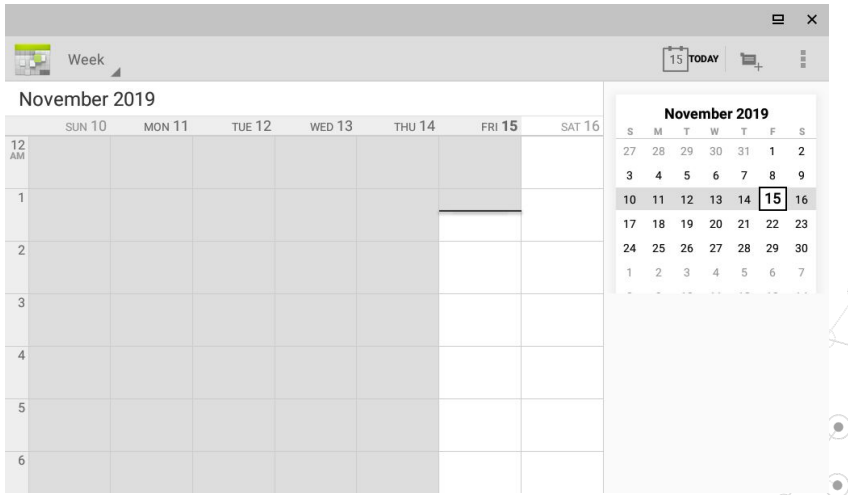
Dynamic Binary Instrumentation (DBI)

Universidad del Rosario | MACC Matemáticas Aplicadas y Ciencias de la Computación









- Ahora debemos instalar ADB (Android Debug Bridge), el cual nos permitirá comunicarnos desde la máquina física con el dispositivo emulado

```
(base) urosario@CNP77151:~$ sudo apt install adb
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libllvm7
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  android-libadb android-libbase android-libboringssl android-libcrypto-utils
  android-libcutils android-liblog android-sdk-platform-tools-common
```

- Para ver los dispositivos móviles “reales” o emulados mediante ANBOX que están conectados al computador físico lo hacemos por medio del siguiente comando:

```
(base) urosario@CNP77151:~$ adb devices
List of devices attached
* daemon not running; starting now at tcp:5037
* daemon started successfully
emulator-5558   device
```

Este es el dispositivo emulado!!

https://developer.android.com/studio/command-line/adb

FRIDA

FRIDA es la herramienta que utilizaremos para inyectar código a una aplicación
móvil
vulnerable

- FRIDA es la herramienta que utilizaremos para inyectar código a una aplicación móvil
  vulnerable

```
(base) urosario@CNP77151:~/Documents$ pip install frida-tools
Requirement already satisfied: frida-tools in /home/urosario/anaconda3/lib/python3.7/site-p
ackages (5.2.0)
Requirement already satisfied: frida<13.0.0,>=12.7.3 in /home/urosario/anaconda3/lib/python
3.7/site-packages (from frida-tools) (12.7.22)
```

```
(base) urosario@CNP77151:~$ pip install frida
Collecting frida
  Downloading https://files.pythonhosted.org/packages/7d/27/69bc3f09b52241b2d890
c8fdbb54bf8cf85817f3c23146cadd841a876fde/frida-12.7.22.tar.gz
Building wheels for collected packages: frida
  Building wheel for frida (setup.py) ... \
```

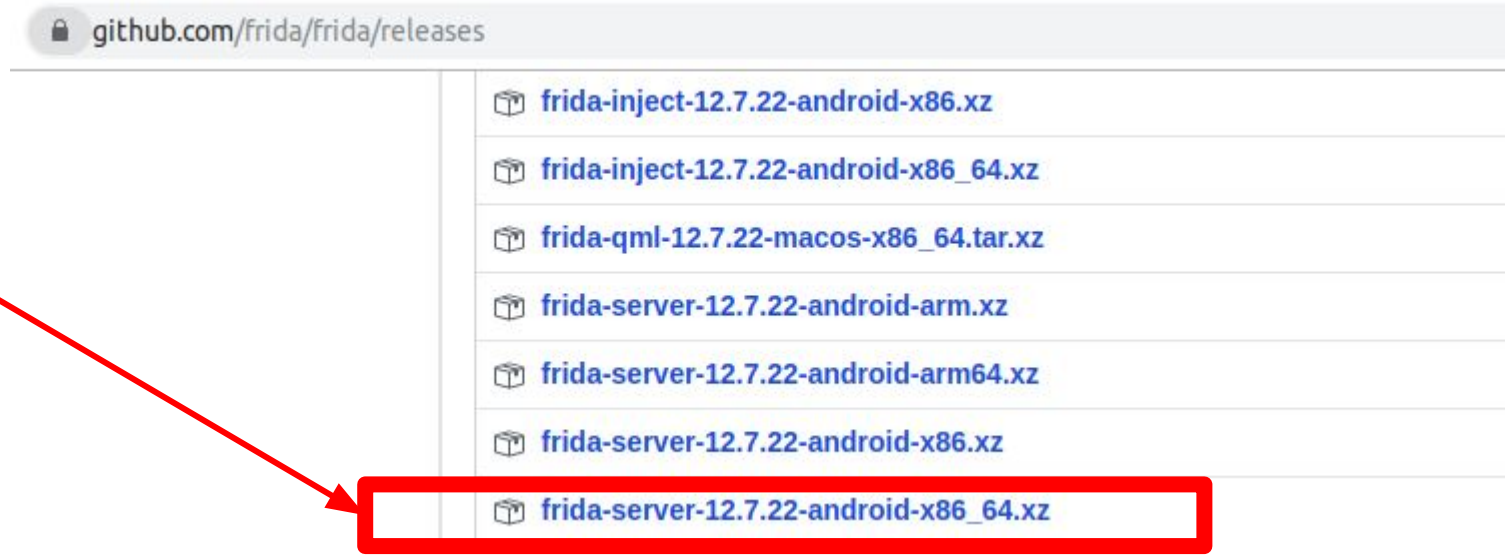
```
(base) urosario@CNP77151:~$ sudo apt install npm
[sudo] password for urosario:
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

```
(base) urosario@CNP77151:~$ npm install frida
loadDep:strip-json-commen
```

Comandos aqui: https://github.com/frida/frida

- Descargar y descomprimir FRIDA server - https://github.com/frida/frida/releases

Escoger la versión adecuada según la arquitectura del dispositivo móvil (X86, X86_64, arm), en nuestro caso será la x86_64



Esta es la ruta de destino de la transferencia en el dispositivo emulado

- Instalar FRIDA server en el dispositivo emulado utilizando la herramienta “adb” previamente instalada y el argumento “push”

```
(base) urosario@CNP77151:~/Documents$ adb push frida-server-12.7.22-android-x86_
64 /data/local/tmp/frida-server
frida-server-12.7.22-android-x86_64: 1... 97.4 MB/s (53368072 bytes in 0.523s)
```

- Iniciar una consola en el dispositivo emulado usando el comando “adb shell”
- Después de iniciar la consola ejecutar el comando “su” para cambiar a superusuario

```
(base) urosario@CNP77151:~/Documents$ adb shell
generic_x86:/ $ su
generic_x86:/ # ls
acct                    init.environ.rc          sdcard
bugreports              init.goldfish.rc         seapp_contexts
cache                   init.ranchu-encrypt.rc   selinux_version
charger                 init.ranchu-noencrypt.rc sepolicy
config                  init.ranchu.rc           service_contexts
d                       init.rc                  storage
data                    init.usb.configfs.rc     sys
default.prop            init.usb.rc              system
dev                     init.zygote32.rc         ueventd.goldfish.rc
etc                     mnt                      ueventd.ranchu.rc
file_contexts.bin       oem                      ueventd.rc
fstab.goldfish          proc                     var
fstab.ranchu-encrypt    property_contexts        vendor
fstab.ranchu-noencrypt  root
init                    sbin
generic_x86:/ #
```

Cambio a superusuario!

- En este punto ya estamos dentro del dispositivo móvil y acabamos de rootear el dispositivo (acceder como usuario root)

- Revisemos que el archivo de FRIDA server que transferimos hace 2 slides esta en la ruta hacia donde lo transferimos

```
x86_64:/ # cd /data/local/tmp/
x86_64:/data/local/tmp # ls
frida-server re.frida.server
```

- Ahora asignarle permisos de ejecución al archivo y ejecutarlo

```
x86_64:/data/local/tmp # chmod 755 frida-server
x86_64:/data/local/tmp # ./frida-server
```

- En este punto ya podemos ejecutar diferentes comandos desde la consola del equipos físico para acceder a los datos del equipos emulado, por ejemplo: en una consola del equipo físico (no del dispositivo móvil) ejecutar el siguiente comando: **frida-ps -U**

```
(base) urosario@CNP77151:~/Documents$ frida-ps -U
 PID  Name
----  ------------------------------------------------
1322  adbd
2823  android.process.acore
1406  audioserver
1407  cameraserver
1856  com.android.inputmethod.latin
2021  com.android.phone
2438  com.android.printspooler
3024  com.android.providers.calendar
1904  com.android.systemui
```

Primer ataque con FRIDA

- Abrir un programa (por ejemplo gedit) y obtener los módulos en ejecución utilizando FRIDA.
  Link de referencia: https://www.frida.re/docs/installation/

```
import frida

def on_message(message, data):
    print("[on_message] message:", message, "data:", data)

session = frida.attach("gedit")

script = session.create_script("""
rpc.exports.enumerateModules = function () {
  return Process.enumerateModules();
};
""")
script.on("message", on_message)
script.load()

print([m["name"] for m in script.exports.enumerate_modules()])
```

Código de explotación

```
daniel@Latitude-E5470:~/Dropbox/UR/Ethical Hacking/LabIoT$ python example.py
[u'gedit', u'linux-vdso.so.1', u'libgedit.so', u'libgio-2.0.so.0.5600.4', u'libg
object-2.0.so.0.5600.4', u'libc-2.27.so', u'libm-2.27.so', u'libxml2.so.2.9.4',
u'libgtksourceview-3.0.so.1.8.0', u'libpeas-gtk-1.0.so.0.2200.0', u'libgtk-3.so.
0.2200.30', u'libgdk-3.so.0.2200.30', u'libpango-1.0.so.0.4000.14', u'libatk-1.0
.so.0.22810.1', u'libcairo.so.2.11510.0', u'libgdk_pixbuf-2.0.so.0.3611.0', u'li
bpeas-1.0.so.0.2200.0', u'libgirepository-1.0.so.1.0.0', u'libglib-2.0.so.0.5600
.4', u'libX11.so.6.3.0', u'libpthread-2.27.so', u'libgmodule-2.0.so.0.5600.4', u
'libz.so.1.2.11', u'libselinux.so.1', u'libresolv-2.27.so', u'libmount.so.1.1.0'
, u'libffi.so.6.0.4', u'ld-2.27.so', u'libdl-2.27.so', u'libicuuc.so.60.2', u'li
blzma.so.5.2.2', u'libpangocairo-1.0.so.0.4000.14', u'libXi.so.6.1.0', u'libXfix
es.so.3.1.0', u'libcairo-gobject.so.2.11510.0', u'libatk-bridge-2.0.so.0.0.0', u
'libepoxy.so.0.0.0', u'libpangoft2-1.0.so.0.4000.14', u'libfontconfig.so.1.10.1'
, u'libXinerama.so.1.0.0', u'libXrandr.so.2.2.0', u'libXcursor.so.1.0.2', u'libX
```

Resultado de la explotación

- Explicar la lógica del código de explotación de la “Primera ejecución de FRIDA”. ¿Que hace el método attach, create_script, Process.enumerateModules?
- Documentar cada uno de los pasos con capturas de pantalla

Segundo ataque con FRIDA

Seguir las instrucciones del siguiente ejemplo en el cual se instala una aplicación APK en el emulador y posteriormente por medio de un script en Python se logra inyectar un script que monitoriza la aplicación:
https://www.frida.re/docs/examples/android/

Comando de instalación de la APK:
adb install rps.apk

- ¿En que consiste la aplicación APK instalada?
- Explicar la lógica del código de explotación. ¿Cuales son los métodos principales del script?
- Documentar cada uno de los pasos con capturas de pantalla
-

```
import frida, sys

def on_message(message, data):
    if message['type'] == 'send':
        print("[*] {0}".format(message['payload']))
    else:
        print(message)

jscode = """
Java.perform(function () {
  // Function to hook is defined here
  var MainActivity = Java.use('com.example.seccon2015.rock_paper_s
```

Segundo ataque con FRIDA

Seguir las instrucciones del siguiente ejemplo en el cual se instala una aplicación APK en el emulador y posteriormente por medio de un script en Python se logra inyectar un script que monitoriza la aplicación:
https://www.frida.re/docs/examples/android/

Explicar otro tipo de ataque con alguno de los tools de FRIDA::

- frida-ps
- frida-trace
- frida-discover
- frida-ls-devices
- frida-kill

```
import frida, sys

def on_message(message, data):
    if message['type'] == 'send':
        print("[*] {0}".format(message['payload']))
    else:
        print(message)

jscode = """
Java.perform(function () {
  // Function to hook is defined here
  var MainActivity = Java.use('com.example.seccon2015.rock_paper_s
```

Universidad del Rosario | MACC | HINNT

¡Gracias!