4. Tool setup/installation

The workshop provides detailed guidelines for three main devices, which is the proxmark3, Flipper Zero and ACR122U. If you do not have an RFID device, you can use an ACR122U from the instructor provided during the workshop. The challenges can be done with other RFID reader/writers as well.

Keep in mind that these instructions are NOT tested within Virtual Machine's (VM's) and VM's are not suggested since almost always issues occur when working inside VM's to perform the exercises.

4.1. Proxmark3

The Proxmark3 can be called the Swiss-army tool for RFID Research. The code base is mostly written in C and is fully open source and relatively easy to use. The tool is easy to use for people who have some experience with embedded programming, electronics, and Linux. Without these it might not be the tool for you.

One of key advantages of the Proxmark is its community support and active maintenance with frequent updates, which is awesome for being an open source tool. It's easy to extend and learn from the work that others have done. I used it to create fuzzing libraries and exploits for EMV and the Proxmark3 is just the tool to use for this.

Please read the installation instructions on how to setup the device here below depending if you are on Windows or Linux.

4.2. Flipper

Introduced in 2020 and received a lot of attention as it's a very useful pocketsize device. It has a Low Frequency and High Frequency reader built-in and great community support. It might not have the same specification and capabilities as the Proxmark, but it can also do much more than RFID protocols alone.

A little less user friendly in some activities such as manipulating data on RFID cards but perfect for other simpler tasks.

4.3. ACR 122U

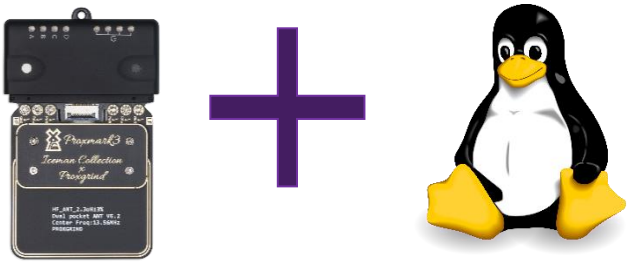
It's a cheap NFC reader/writer of around 20 - 50 euro, works well with several NFC libraries. It does not have the same capabilities as the Proxmark and Flipper, but you can still compromise most of the RFID systems in the world with just this tool.

For this reader I created a custom Python tool you can use to do the courses. Obviously, you are free to use any other tools, but the walkthroughs will be written for that tool.

https://github.com/VinnieV/crid

There are a couple more tools that are required, such as mfoc, mfcuk and milazycracker.

For more instructions on how to setup the tools and device please refer to the installation page for ACR122U.



Proxmark3 Linux Installation

The documentation to flash and build the Proxmark client for Linux is documented here:

https://github.com/RfidResearchGroup/proxmark3/blob/master/doc/md/Installation_Instructions/Linux-Installation-Instructions.md

Installing the software on Linux should be the easiest way to interact with the Proxmark. Follow these steps below:

1. Perform package update:

```
sudo apt-get update
```

2. Install following dependencies:

```
sudo apt-get install --no-install-recommends git ca-certificates build-essential pkg-config \
libreadline-dev gcc-arm-none-eabi libnewlib-dev qtbase5-dev \
libbz2-dev liblz4-dev libbluetooth-dev libpython3-dev libssl-dev
```

3. Clone the repository:

```
cd
git clone https://github.com/RfidResearchGroup/proxmark3.git
cd proxmark
```

4. Build the code:

```
make clean && make all
```

Building might take some time and hopefully no errors occurred that required additional troubleshooting. Once successful, it created the firmware binaries, ready to be flashed on the Proxmark, by following these steps:

⚠ Debugging Tip: Check if the USB cable is not charge-only type cable.

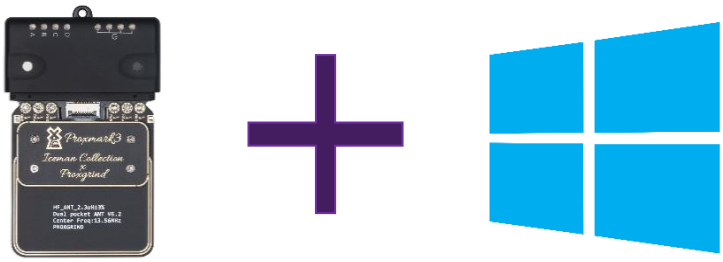
1. Connect your Proxmark device to your Windows computer.
2. Once connected, you can run the flash binary to install the bootloader and firmware on the Proxmark:

```
./pm3-flash-all
```

3. Once that's completed you should be good to go to enter the Proxmark shell by typing in:

```
./pm3
```

This Proxmark3 shell will provide all the power to perform the exercises.



Proxmark3 Windows Installation

The documentation to flash and build the Proxmark client for Windows is documented here:

https://github.com/RfidResearchGroup/proxmark3/blob/master/doc/md/Installation_Instructions/Windows-Installation-Instructions.md

Feel free to follow that documentation but I have broken down the generic steps here. The first step would be to download and install the ProxSpace Tool. This is a Linux subsystem within windows where we will have all the tools and dependencies installed to build and execute the Proxmark project. To install ProxSpace perform the following:

1. Download the ProxSpace ZIP file, available here: https://github.com/Gator96100/ProxSpace/releases
2. Extract The ProxSpace ZIP file to a location path without spaces.
3. Execute the bat file within the ProxSpace folder:

```
.\runme64.bat
```

It might take some time the first time to install everything but that should give you a new command prompt starting with pm3:

```
Initial setup complete. MSYS2 is now ready to use.
pm3 ~$
```

This shell provides you with the code building environment to build the Proxmark repository for Windows. To perform this, please follow these steps:

1. Clone the Proxmark Repository:

```
cd
git clone https://github.com/RfidResearchGroup/proxmark3.git
cd proxmark3
```

2. Then build the code:

```
make clean && make all
```

Building might take some time and hopefully no errors occurred that required additional troubleshooting. Once successful, it created the firmware binaries, ready to be flashed on the Proxmark, by following these steps:

⚠ Debugging Tip: Check if the USB cable is not charge-only type cable.

1. Connect your Proxmark device to your Windows computer.

2. Once connected, you can run the flash binary to install the bootloader and firmware on the Proxmark:

```
./pm3-flash-all
```

```
pm3 ~/proxmark3$ ./pm3-flash-all
[=] Session log C:\Users\kernelpanic\Documents\Tools\ProxSpace\pm3/.proxmark3/logs/log_20231208202849.txt
[+] About to use the following files:
[+]    C:\Users\kernelpanic\Documents\Tools\ProxSpace\pm3\proxmark3\client\../bootrom/obj/bootrom.elf
[+]    C:\Users\kernelpanic\Documents\Tools\ProxSpace\pm3\proxmark3\client\../armsrc/obj/fullimage.elf
[+] Loading ELF file C:\Users\kernelpanic\Documents\Tools\ProxSpace\pm3\proxmark3\client\../bootrom/obj/bootrom.elf
[+] ELF file version Iceman/master/v4.17511-96-g8419b9c69-suspect 2023-12-08 21:23:22 255a01757

[+] Loading ELF file C:\Users\kernelpanic\Documents\Tools\ProxSpace\pm3\proxmark3\client\../armsrc/obj/fullimage.elf
[+] ELF file version Iceman/master/v4.17511-96-g8419b9c69-suspect 2023-12-08 21:23:50 255a01757

[+] Waiting for Proxmark3 to appear on COM7
[|] 59 found
[+] Entering bootloader...
[+] (Press and release the button only to abort)
[+] Waiting for Proxmark3 to appear on COM7
[/] 48 found
[=] Available memory on this board: 512K bytes

[=] Permitted flash range: 0x00100000-0x00180000
[+] Loading usable ELF segments:
[+]    0: V 0x00100000 P 0x00100000 (0x00000200->0x00000200) [R X] @0x94
[+]    1: V 0x00200000 P 0x00100200 (0x00001260->0x00001260) [R X] @0x298

[+] Loading usable ELF segments:
[+]    0: V 0x00102000 P 0x00102000 (0x00053cac->0x00053cac) [R X] @0x98
[+]    1: V 0x00200000 P 0x00155cac (0x00001b9e->0x00001b9e) [R X] @0x53d48
[=] Note: Extending previous segment from 0x53cac to 0x5584a bytes

[+] Flashing...
[+] Writing segments for file: C:\Users\kernelpanic\Documents\Tools\ProxSpace\pm3\proxmark3\client\../bootrom/obj/bootrom.elf
[+]  0x00100000..0x001001ff [0x200 / 1 blocks]
. ok
[+]  0x00100200..0x0010145f [0x1260 / 10 blocks]
.......... ok

[+] Writing segments for file: C:\Users\kernelpanic\Documents\Tools\ProxSpace\pm3\proxmark3\client\../armsrc/obj/fullimage.elf
[+]  0x00102000..0x00157849 [0x5584a / 685 blocks]
.............................................................
        @@@  @@@@@@@ @@@@@@@@ @@@@@@@@@@   @@@@@@  @@@  @@@
        @@! !@@      @@!      @@! @@! @@!  @@@ @@!@!@@@
        !!@ !@!      @!!!:!   @!! !!@ @!@ @!@!@!@! @!@@!!@!
        !!: :!!      !!:      !!:     !!: !!:  !!! !!:  !!!
        :    :: :: : : :: :::  :      :    :   : : ::    :
        .    .. .. . . .. ...  .      .    .   . . ..    .
.............................................................
.............................................................
.............................................................
..................................... ok

[+] All done

[=] Have a nice day!
```

3. Once that's completed you should be good to go to enter the Proxmark shell by typing in:

```
./pm3
```

```
[=] No previous history could be loaded
[usb] pm3 -->
```

This Proxmark3 shell will provide all the power to perform the exercises.

Flipper Zero Installation

The documentation to flash the firmware of the is documented here:

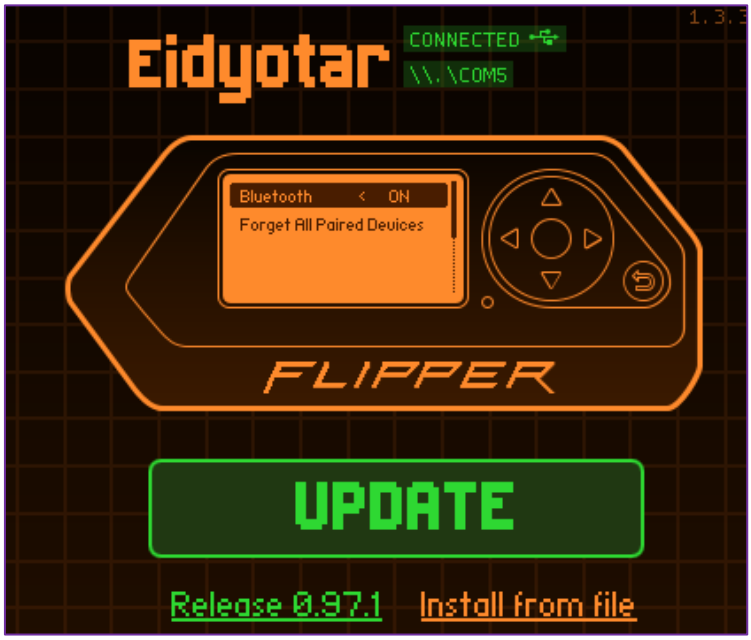
https://docs.flipper.net/basics/first-start

It is required to have a micro-SD Card available for your flipper device and it is advised to check for firmware updates. To achieve this, follow either the instructions to update through qFlipper application or the Flipper Mobile Application:

qFlipper Application

Install qFlipper tool from here: https://flipperzero.one/update

1. Connect your Flipper Zero to your computer with USB-C cable.
2. Open the qFlipper tool and Update the software

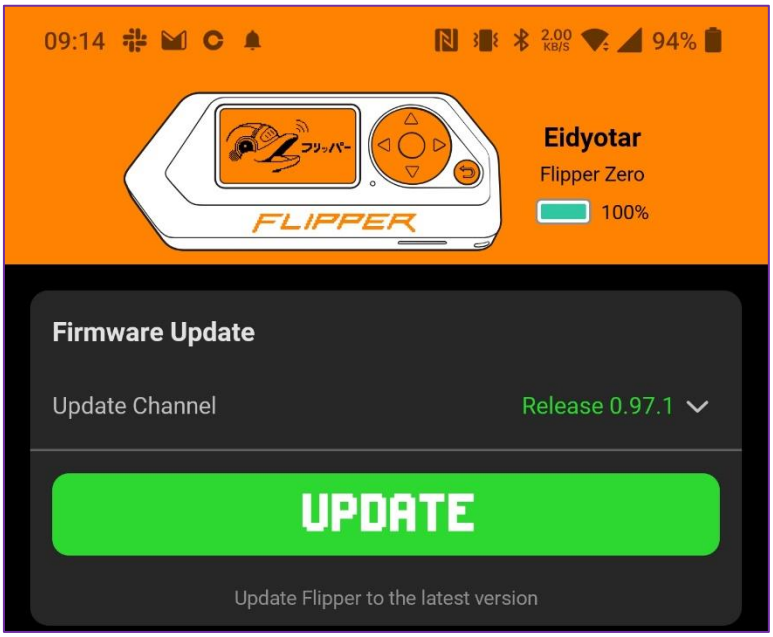


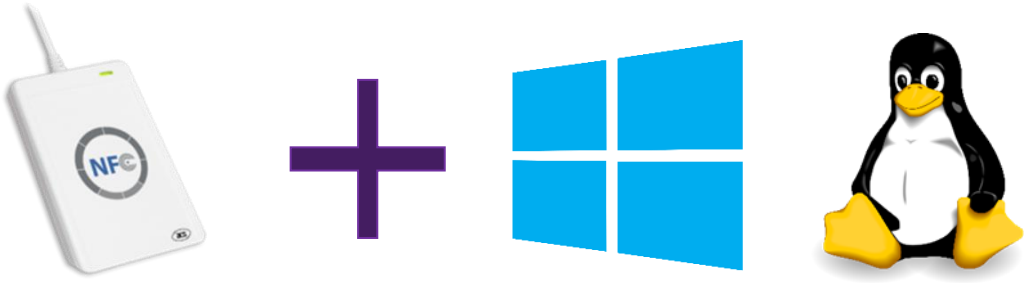
Flipper Mobile Application

iOS App Store: https://apps.apple.com/app/flipper-mobile-app/id1534655259
Android Play Store: https://play.google.com/store/apps/details?id=com.flipperdevices.app

1. Make sure the Flipper Zero has an SD card inserted.
2. Connect mobile phone and Flipper Zero with Bluetooth.

3. Initiate the Update through the mobile application.





ACR122U Setup for Windows and Linux

These installation guidelines will be similar for Windows and Linux since the tools we are going to use are created and python and work on both Operating systems. If you don't have any Python3 version installed yet, please follow the respective links to install Python3 for your system:

Python3: https://www.python.org/downloads/

Pip3: https://pip.pypa.io/en/stable/installing/

Additionally, on Linux you need to install the following packages from your OS package manager (might not need all of them):

```
sudo apt-get update
sudo apt-get install libpcsclite-dev libpcsclite1 pcscd pcsc-tools
```

Once Python is installed you have a couple of options to install the tools:

PIP

You can use pip3 to install the required tools by running the following command:

```
pip3 install swig crid
```

If you don't have pip3 installed, check if the command pip is a working alias for pip3 for python3 or install Python3/pip3.

You can also do pip3 install from the source directly:

```
git clone https://github.com/VinnieV/crid.git
cd crid
pip3 install .
```

To make sure you can just run the command `crid` in your terminal do the following:

Windows:

In an Administrative Command Line shell the following oneliner will add the Python scripts folder to the PATH in the current shell:

```
for /f "delims=" %a in ('python3 -m site --user-site') do set
PATH=%PATH%;%a\..\Scripts
```

or for PowerShell:

```
$env:PATH += ";" + (python3 -m site --user-site) + "\..\Scripts"
```

Cloning and running from source

You can directly clone

```
git clone https://github.com/VinnieV/crid.git
cd crid
pip3 install -r requirements.txt
python3 main.py
```

Optionally for some challenges, the crid CLI will use some other tools to recover the keys. Install instructions through :

```
sudo apt-get install mfoc
```

OR

```
git clone https://github.com/nfc-tools/mfoc.git
cd mfoc
sudo apt-get install autoconf build-essential
autoreconf -is
./configure
make && sudo make install
```

To install the hardnested tool you can follow these instructions:

```
git clone https://github.com/VinnieV/crypto1_bs
cd crypto1_bs
make get_craptev1
make get_crapto1
make
sudo cp -a libnfc_crypto1_crack /usr/local/bin
```

Installing MFCUK for the darkside can be done as follows:

```
git clone https://github.com/nfc-tools/mfcuk.git
cd mfcuk
autoreconf -is
./configure
make && sudo make install
```

Installing MiLazyCracker can be done as follows:

```
git clone https://github.com/nfc-tools/miLazyCracker.git
cd miLazyCracker
./miLazyCrackerFreshInstall.sh
```

Debugging

- (Linux) sudo nfc-list error libnfc.driver.acr122_usb Unable to claim USB interface (Device or resource busy) nfc-list: ERROR: Unable to open NFC device: acr122_usb:001:020.
  Perform the following commands:
  `sudo modprobe -r pn533_usb && sudo modprobe -r pn533`
  But for a more permanent solution, create the following file:
  `sudo nano /etc/modprobe.d/blacklist-libnfc.conf` and then add the following lines:
  ```
  blacklist pn533
  blacklist pn533_usb
  blacklist nfc
  ```
- (Windows) error: command 'swig.exe' failed: None
  make sure to install download and unzip this file
  http://prdownloads.sourceforge.net/swig/swigwin-4.2.0.zip and copy swig.exe
  Then put the location of the of the unzipped folder in your Path environment variable.
- (Windows) error: Microsoft Visual C++ 14.0 or greater is required.
  Install and run this tool: https://visualstudio.microsoft.com/visual-cpp-build-tools/
  Make sure to install at least the package: MSVC v142 - VS 2019 C++ x64/x86 build tools (v14.0 or greater)

ACR122U installation MacOS Installation

```
brew install swig
pip3 install crid
```

Might be required for the following as well on MacOS version 14 (Sonoma)

```
pip3 install py122u smartcard pcsc-lite libusb --force
```

Or when having some issues when running the tools related to Transaction Error:

Failed to transmit with protocol T1

Then follow this to enable another driver:

https://blog.apdu.fr/posts/2023/11/apple-own-ccid-driver-in-sonoma/

Which is essentially running the following command:

```
sudo defaults write /Library/Preferences/com.apple.security.smartcard
useIFDCCID -bool yes
```

And might require reboot.