Artemis

Artemis originates from antique Greek and means - amongst other - *unharmful* or *safe*. Exactly that, Artemis should deliver to you: safety for your data and no harm to your communications.

In detail, Artemis is a simple RaspberryPi packed up with different kinds of software packages, meshed up configurations, which results in an isolated, secure server.

Artemis' can be used in two different modes:

- **SafeHarbour** (SF) provides services locally as an isolated server on a LAN, with no connection to the Internet
- **Periscope** (P) or as acting as a secure proxy server on an unsecured network

Requirements

Hardware	Quantity
RaspberryPi + MicroUSB / Power Adapter	1
MicroSD Card	1
USB Thumbdrive	1
RJ45	1

Prerequisites (paran0id modus)

You either can proceed with the following advices or just jump to configuration and head on with encryption of the filesystem.

Creating temporary environment

Ensure that you download and install the ArchLinux ARM to a non-infected SD card on a secured computer. We recommand the following way of proceeding.

1. Download the latest ArchLinux ISO and its checksum.
2. Check against each other.
3. Copy the ISO to a clean USB thumbdrive.

<sxh bash;># dd bs=1m if=./archlinux-YYYY.MM.DD-dual.iso of=/dev/sdX</sxh>

1. Plug in a computer and boot it up.

Preparing Micro SD Cards

1. Start fdisk to partition the SD card:

```
fdisk /dev/mmcplk0
```

1. At the fdisk prompt, delete old partitions and create a new one:
    1. Type o. This will clear out any partitions on the drive.
    2. Type p to list partitions. There should be no partitions left.
    3. Type n, then p for primary, 1 for the first partition on the drive, press ENTER to accept the default first sector, then type +100M for the last sector.
    4. Type t, then c to set the first partition to type W95 FAT32 (LBA).
    5. Type n, then p for primary, 2 for the second partition on the drive, and then press ENTER twice to accept the default first and last sector.
    6. Write the partition table and exit by typing w.
2. Create and mount the FAT filesystem:

<sxh>

```
    mkfs.vfat /dev/sdX1
    mkdir boot
    mount /dev/sdX1 boot</sxh>
- Create and mount the ext4 filesystem:
    <sxh>
    mkfs.ext4 /dev/sdX2
    mkdir root
    mount /dev/sdX2 root
    </sxh>
- Download and extract the root filesystem (as root, not via sudo):
  <sxh>
  wget http://archlinuxarm.org/os/ArchLinuxARM-rpi-latest.tar.gz
http://archlinuxarm.org/os/ArchLinuxARM-rpi-latest.tar.md5
  # GNU/Linux
  md5sum ArchLinuxARM-rpi-latest.tar.gz > md5sums.md5
  cat ArchLinuxARM-rpi-latest.tar.gz.md5 >> md5sums.md5
  md5sum -c md5sums.md5
  # MacOS X
  md5 ArchLinuxARM-rpi-latest.tar.gz > ArchLinuxARM-rpi-
latest.tar.gz.original.md5
  diff ArchLinuxARM-rpi-latest.tar.gz.original.md5 ArchLinuxARM-rpi-
latest.tar.gz.md5
  # If there is no return given / 0 as exit code, the match was successful.
  tar -xf ArchLinuxARM-rpi-latest.tar.gz -C root
  sync
  </sxh>
- Move boot files to the first partition:
  <code>
  mv root/boot/* boot
  </code>
- Unmount the two partitions:
  <code>
  umount boot root</code>
- Insert the SD card into the Raspberry Pi, connect ethernet, and apply 5V
power.
- Use the serial console or SSH to the IP address given to the board by your
router. The default root password is 'root'.
```

When finished, fire up the RaspberryPi with the new system connected to a display and keyboard, but **not** to a network. Log in with username **root** and password **root**.

SSHD

Firewall

In order to network-secure Artemis tight as possible, we are just going to drop every input and output traffic by default, and whitelist whatever services we need.

Purging iptables

Safe the following commands in /etc/iptables/purge-all-rules.sh

<sxh bash;Purging all iptable rules> iptables -F iptables -X iptables -t nat -F iptables -t nat -X iptables -t mangle -F iptables -t mangle -X iptables -t raw -F iptables -t raw -X iptables -t security -F iptables -t security -X iptables -P INPUT ACCEPT iptables -P FORWARD ACCEPT iptables -P OUTPUT ACCEPT </sxh>

<sxh bash;title:Dropping all packages;> # iptables -N TCP # iptables -N UDP </sxh>

Reboot

Now, that you have sealed off your RaspberryPi, you'll have to reboot and we can finally get on the encryption and configuration parameters.

Configuration

Note: Ensure to be on a separate secured network or even better, plug yourself in a non-networked computer.

<sxh bash; title:Setting hostname> # hostnamectl set-hostname your-desired-hostname </sxh>

We'd recommend gluuing alluminium heatsinks on the GPU as CPU.

We upgraded this configuration to the medium as we are going to use some software as which needs a bit more power than usual. Pleas adapt this to your needs.

<sxh bash; title:Boosting computational power of RPi> # vim /boot/config.txt ##Medium arm_freq=900 core_freq=333 sdram_freq=450 over_voltage=2 </sxh>

Network Configuration

By default RPi receives IP address via DHCP. We'll assign it a static one.

<sxh bash;> # cp /etc/netctl/examples/ethernet-static /etc/netctl # vim /etc/netctl/ethernet-static

PUT CONFIG HERE </sxh>

<sxh bash; title:Disabling default network settings> # systemctl disable dhcpcd@eth0.service # systemctl disable dhcpcd # systemctl disable netctl-ifplugd@eth0 </sxh>

<sxh bash; title:Enabling new network config> # netctl enable ethernet-static </sxh>

System Update

<sxh bash;> # pacman -Syyu </sxh>

Additional Software

The first both programs are needed in order to complete the howto, *vim* is a comfortable choice though.

<sxh bash;> # pacman -S rsync mkinitcpio vim </sxh>

root partition encryption

Step	Command
1	n
2	p
3	ENTER
4	ENTER
5	w

References

[0] RaspberryPi With Root Partition Encryption, Unlocked Using Flash Drive
[1] Archlinux ARM encrypted root
[2] ArchLinux ARM - Raspberry Pi Installation