ChatGPT

From Broken to Stealth: Comprehensive Raspberry Pi 5 Recovery and Hardening Guide

This guide walks you through recovering a Raspberry Pi 5 from a bricked state and transforming it into a secure, stealthy platform. It integrates official documentation, community-tested practices, and advanced operational security (OPSEC) techniques. Follow each section carefully to avoid data loss or permanent damage.

Table of Contents

---

1 Understanding the Problem

A Raspberry Pi can become unbootable for many reasons, such as misconfiguring the boot order for an NVMe drive or corrupting the bootloader. In your case, the EEPROM was set to boot from NVMe, and now the Pi won't boot even from a new SD card. Recovery requires resetting the bootloader and restoring the operating system.

Key symptoms include:

- Power LEDs are on, but the Pi does not complete the boot process.
- Green LED flashes but no display output; fans may not spin when powered via the usual supply.
- The Pi does not boot even with a fresh SD card.

Before assuming hardware failure, follow the recovery steps below.

2 Pre-requisites and Tools

Gather the following before beginning:

- A separate computer (Linux, macOS or Windows) with internet access.
- A spare microSD card (4 GB or larger) and a card reader.
- An official Raspberry Pi power supply (27 W recommended for Pi 5).
- USB–NVMe adapter (if flashing NVMe on another computer).
- The `rpi-eeprom` recovery files and Raspberry Pi Imager.
- A network connection (Ethernet preferred for security) and optional Ethernet cable.

You will also install several software packages on the Pi for hardening, including `ufw`, `fail2ban`, `aide`, `lynis`, `rfkill`, `tor`, `binwalk`, and others. These tools are available in standard repositories.

3 Bootloader Recovery

3.1 Why Recovery Is Needed

The Raspberry Pi 5 uses an EEPROM chip to store the bootloader. If the EEPROM becomes misconfigured or corrupted—such as setting an invalid boot order—the Pi may ignore any microSD or NVMe boot devices. Fortunately, the SoC ROM has a built-in mechanism to load a recovery image, `recovery.bin`, from a microSD card and restore the EEPROM.

The SARPi Project documentation explains that at power-on, the ROM on BCM2712 (Pi 5) looks for `recovery.bin` in the root of the microSD boot partition and executes it instead of the EEPROM [1]. This mechanism allows you to reset the bootloader to factory defaults [2].

3.2 Downloading the Recovery Image

1. On your separate computer, navigate to the Raspberry Pi EEPROM firmware repository:
2. Raspberry Pi 5 recovery file: `firmware-2712/latest/recovery.bin` at https://github.com/raspberrypi/rpi-eeprom [3].
3. Download `recovery.bin` for the Pi 5 and save it locally.

3.3 Preparing the Recovery microSD Card

1. Format a spare microSD card as **FAT32**. This card should contain **only** `recovery.bin` in its root directory.
2. Copy the downloaded `recovery.bin` onto the card. Ensure no other files are present.

3.4 Performing the Recovery

1. With the Pi powered off, insert the microSD card containing `recovery.bin`.

2. Apply power using the official 27 W supply. The green ACT LED should blink rapidly, indicating the recovery file is executing.

3. Wait at least **30 seconds**. The ROM loads `recovery.bin`, which restores a known-good bootloader image [2].

4. Power off the Pi and remove the recovery microSD card.

3.5 Confirming Recovery

- Upon powering up with a normal boot medium, the Pi should resume booting. During recovery, watch the green LED; rapid flashing followed by normal patterns indicates a successful EEPROM reset.
- If recovery fails, double-check that the correct `recovery.bin` for Pi 5 was used and that the card was formatted correctly.

4 Re-Flashing Raspberry Pi OS

After the EEPROM is reset, you need a working operating system on either a microSD card or NVMe drive.

4.1 Install Raspberry Pi Imager

On your separate computer, download and install **Raspberry Pi Imager** from https://www.raspberrypi.com/software/.

4.2 Flash a Fresh microSD Card

1. Insert a microSD card into your computer's card reader.
2. Open Raspberry Pi Imager, choose **Raspberry Pi OS (64-bit)** (or another supported OS).
3. Select the microSD card as the target and write the image.
4. Optional: Use advanced settings (the gear icon) to set a hostname, enable SSH, and preload your public SSH key.
5. When complete, insert the microSD card into your Pi and power on. The Pi should boot into the OS.

4.3 Flash an NVMe Drive (Optional)

If you want to boot from NVMe after recovery:

1. Attach your NVMe SSD to your computer via a USB–NVMe adapter.
2. Use Raspberry Pi Imager to flash the same OS directly to the NVMe drive. Ensure you choose the correct target drive to avoid overwriting your computer's drives.
3. Eject the NVMe drive when flashing completes and attach it to the Pi via a compatible NVMe HAT.

4.4 Cloning an Existing SD to NVMe

Alternatively, boot your Pi from microSD and clone the system to the NVMe drive. Jeff Geerling recommends using `rpi-clone` [4]:

```
# Install rpi-clone (Jeff Geerling's fork supports NVMe)
git clone https://github.com/geerlingguy/rpi-clone.git
cd rpi-clone
sudo cp rpi-clone rpi-clone-setup /usr/local/sbin

# Identify the NVMe device (usually /dev/nvme0n1)
lsblk

# Clone microSD contents to the NVMe drive
sudo rpi-clone nvme0n1
```

After cloning, shut down the Pi, remove the microSD card and boot from the NVMe drive.

5 Configuring Boot Order and NVMe (Optional)

The bootloader configuration determines the sequence in which devices are tried. To modify it:

1. Boot into Raspberry Pi OS (from microSD) and update packages:

```
sudo apt update
sudo apt full-upgrade -y
sudo reboot
```

1. Edit the EEPROM configuration:

```
sudo rpi-eeprom-config --edit
```

1. Find the `BOOT_ORDER` line and set a value:
2. `BOOT_ORDER=0xf41` → MicroSD first, then USB/NVMe [5].
3. `BOOT_ORDER=0xf416` → NVMe first, then microSD [5].
4. Add `PCIE_PROBE=1` if using non-HAT NVMe adapters [6].
5. Save and exit the editor; reboot to apply.
6. Using `raspi-config` is simpler:

```
sudo raspi-config
# Advanced Options → Boot → Boot Order
```

1. Optional: Add `dtparam=pciex1` (or `dtparam=nvme`) to `/boot/firmware/config.txt` to enable the external PCIe port [7].

6 Power and Hardware Considerations

- Use the **official 27 W USB-C power supply** for Pi 5. Insufficient power can cause NVMe boot failures.
- For NVMe boot issues, adding `max_framebuffers=2` to `config.txt` may reduce PCIe bandwidth contention (empirical workaround).
- Ensure your NVMe HAT and SSD are compatible and properly seated.

7 Base System Hardening

Before enabling advanced OPSEC, establish a secure baseline. These steps protect against common attacks and unauthorized access.

7.1 Change Default Credentials

- If you used Raspberry Pi Imager, you may have already set a username and password. Otherwise, change the default password with `passwd` and consider creating a new user.

7.2 Update the System Regularly

Run updates to keep packages and the kernel current:

```
sudo apt update && sudo apt full-upgrade -y
sudo reboot
```

7.3 Configure SSH Key Authentication

1. Generate an SSH key pair on your workstation:

```
ssh-keygen -t ed25519 -b 4096 -C "pi-key"
```

1. Copy the public key to the Pi:

```
ssh-copy-id -i ~/.ssh/pi-key.pub pi@<pi-ip-address>
```

1. On the Pi, edit `/etc/ssh/sshd_config` to harden SSH [8]:

```
PasswordAuthentication no
PermitRootLogin no
KbdInteractiveAuthentication no
AuthorizedKeysFile .ssh/authorized_keys
```

1. Restart SSH:

```
sudo systemctl restart ssh
```

Now only key-based logins will succeed [8].

7.4 Enable the Uncomplicated Firewall (UFW)

Restrict inbound connections to essential services [9]:

```
sudo apt install ufw -y
sudo ufw default deny incoming
sudo ufw default allow outgoing
sudo ufw allow ssh  # default port 22
sudo ufw enable
sudo ufw status
```

7.5 Install Fail2Ban

Fail2Ban monitors logs and bans IPs after repeated authentication failures [10]:

```
sudo apt install fail2ban -y
# Default settings protect SSH; review /etc/fail2ban/jail.conf for
customization.
```

7.6 Disable Unused Services and Radios

- Edit `/boot/firmware/config.txt` to disable Bluetooth and Wi-Fi (if using wired Ethernet) [11]:

`dtoverlay=disable-bt`
`dtoverlay=disable-wifi`

- Install `rfkill` and block radios at runtime [12]:

```
sudo apt install rfkill -y
sudo rfkill block all
sudo rfkill list
```

7.7 Verify Radio Silence

After disabling radios, verify no interfaces are active:

```
hciconfig -a | grep -i "up\|running"  # Should return nothing
iwconfig 2>/dev/null | grep -i "essid"  # Should return nothing
```

8 Advanced OPSEC: Submarine Mode

These measures go beyond basic hardening and aim to minimise the Pi's network fingerprint and provide anonymity where appropriate. Apply only those steps necessary for your use case.

8.1 Network Invisibility Enhancements

Change or Randomize MAC Address

To avoid exposing the Pi's factory MAC address, set a static, non-identifiable MAC for the Ethernet interface:

```
# Edit /etc/dhcpcd.conf
echo "interface eth0" | sudo tee -a /etc/dhcpcd.conf
echo "static mac_address=02:12:34:56:78:9a" | sudo tee -a /etc/dhcpcd.conf
```

The first octet should be `02` to indicate a locally administered address.

SSH Port Obfuscation

Changing the SSH port and restricting access to trusted subnets adds obscurity and reduces scan noise:

```
# Update UFW rules
sudo ufw delete allow ssh
sudo ufw allow from 192.168.1.0/24 to any port 2222

# Edit /etc/ssh/sshd_config
sudo sed -i 's/#Port 22/Port 2222/' /etc/ssh/sshd_config
sudo systemctl restart ssh
```

For further camouflage, consider using **sslh**, which multiplexes SSH over HTTPS (port 443).

8.2 Kernel and Network Hardening

Add sysctl tunables to reduce network visibility:

```
echo "net.ipv4.icmp_echo_ignore_all=1" | sudo tee -a /etc/sysctl.conf  # Ignore
ping
echo "net.ipv4.tcp_timestamps=0" | sudo tee -a /etc/sysctl.conf      # Hide
timestamp info
echo "kernel.dmesg_restrict=1" | sudo tee -a /etc/sysctl.conf        # Restrict
dmesg
sudo sysctl -p
```

8.3 Hostname and Identity Sanitization

Remove Pi-specific identifiers:

```
sudo hostnamectl set-hostname localhost
sudo sed -i 's/raspberrypi/localhost/g' /etc/hosts
```

Use a neutral case and hide any Pi branding if physically visible.

8.4 Tor Hidden Service (Optional and Legal Considerations)

Tor can provide anonymized remote access via `.onion` addresses if permitted in your jurisdiction. John Harrington's tutorial demonstrates installing Tor and configuring a hidden SSH service [13]:

```
sudo apt install tor -y
# Configure Tor to expose SSH as a hidden service
sudo tee -a /etc/tor/torrc <<'TORCFG'
HiddenServiceDir /var/lib/tor/ssh/
HiddenServicePort 22 127.0.0.1:2222
TORCFG
sudo systemctl restart tor
# Get the .onion hostname
sudo cat /var/lib/tor/ssh/hostname
```

Only those with the `.onion` address and your private key can access SSH over Tor. Use responsibly.

8.5 Memory-Only Logging (Extreme Anti-Forensics)

Mounting `/tmp` and `/var/log` as `tmpfs` keeps logs in volatile memory. This reduces persistent forensic traces but also means logs vanish on reboot. Use only if you understand the consequences:

```
# Append to /etc/fstab
sudo tee -a /etc/fstab <<'TMPFS'
tmpfs /tmp tmpfs defaults,nosuid,nodev,noexec 0 0
```

```
tmpfs /var/log tmpfs defaults,size=100m 0 0
TMPFS
```

Reboot to apply. Combine this with remote logging if you still require audit trails.

8.6 Process Monitoring and Integrity Checks

Install **aide** (Advanced Intrusion Detection Environment) to monitor filesystem changes:

```
sudo apt install aide -y
sudo aideinit
sudo cp /var/lib/aide/aide.db.new /var/lib/aide/aide.db
# Schedule a daily check in cron (example)
echo "0 3 * * * /usr/bin/aide.wrapper --check" | sudo tee -a /etc/crontab
```

Install **lynis** for periodic security audits [14] :

```
sudo apt install lynis -y
sudo lynis audit system
```

Review the generated report and address any warnings.

9 Forensic Analysis Environment

With your Pi hardened, you can safely analyze unknown or suspicious files in a controlled manner. Create an isolated analysis directory and never execute binaries unless necessary.

9.1 Set Up an Analysis Workspace

```
sudo mkdir -p /opt/analysis
sudo chmod 700 /opt/analysis
cd /opt/analysis
# Copy your mystery file here when ready
```

Remove execute permissions to avoid accidental execution:

```
chmod 000 mystery_file
```

9.2 Tools to Install

```
sudo apt install binwalk foremost scalpel radare2 libimage-exiftool-perl -y
# Python packages for advanced decoding
pip3 install cyberchef-py floss
```

9.3 Phase-Based Analysis Workflow

1. **Identify file type** – Use `file mystery_file` to detect if it's an ELF, archive, or data file [15]. Avoid executing unknown binaries.
2. **Extract metadata** – Use `exiftool mystery_file` for images or documents.
3. **Extract printable strings** – Use `strings` to find embedded text and search for URLs or keys:

```
strings -a mystery_file > strings_all.txt
strings -n 8 mystery_file > strings_long.txt
grep -E '[A-Za-z0-9]{20,}' strings_all.txt
floss mystery_file  # Detect obfuscated strings
```

1. **Hex dump and structural mapping** – Use `xxd -l 256 mystery_file` to view the first bytes [16]. Use `binwalk` to scan for embedded files and signatures:

```
binwalk -e mystery_file  # Extract embedded files
binwalk -B mystery_file  # Show block structure
```

1. **Entropy and encoding detection** – Check for base64 or hex patterns. Try decoding with CyberChef or `base64 -d`.
2. **Deep binary inspection** – If it's an executable, use `radare2 mystery_file` to list imports (`ii`), strings (`iz`), and assembly.
3. **File carving** – Use `foremost` or `scalpel` for data recovery from damaged files:

```
sudo foremost -i mystery_file -o recovery_dir
sudo scalpel -o carve_out mystery_file
```

1. **Monitor system behaviour** – If you decide to execute code, do so in a virtual machine or container. Use tools like `strace`, `ltrace`, and `htop` to monitor processes and system calls.
2. **Network isolation** – During analysis, temporarily block all outgoing connections to prevent the file from contacting external servers:

```
sudo ufw default deny outgoing
sudo ufw allow out 53  # Allow DNS if necessary
```

1. **Document findings** – Keep a record of all commands run, outputs, and observations. If you need to share logs externally, scrub sensitive data.

10 Ongoing Maintenance and Monitoring

Maintaining a stealthy Raspberry Pi requires regular attention:

- **Update**: Run `sudo apt update && sudo apt full-upgrade` weekly. Check for new `rpi-eeprom` updates (use `sudo rpi-eeprom-update`).
- **Monitor logs**: If you kept persistent logs, review them for unusual activity. Fail2Ban and UFW logs reveal scan attempts.
- **Audit**: Use `lynis` monthly to audit security settings [14].
- **Change keys**: Rotate SSH keys periodically and revoke old ones.
- **Power supply**: Inspect cables and power supply for wear; replace if needed.

11 Conclusion

Recovering a bricked Raspberry Pi and elevating it to a stealth, submarine-mode device requires meticulous work. By leveraging the ROM's recovery mechanism with `recovery.bin` [2], flashing a fresh OS, and configuring boot options [5], you restore functionality. Harden the base system with SSH keys, firewalls, and intrusion prevention [8] [9] [10]. Advanced OPSEC tactics—radio silence, MAC randomization, kernel tuning, Tor hidden services—further reduce the device's footprint. Finally, establishing a robust forensic analysis workflow ensures you can safely decode unknown data without compromising security.

Use this guide as a reference whenever you need to bring a Raspberry Pi back from a broken state or deploy one in a sensitive environment. Adapt the steps to your threat model and operational requirements, and stay vigilant with updates and audits. With the right preparation, a Pi can go from bricked to a covert, resilient platform ready for any mission.

---

[1] [2] [3] SARPi Project - Slacking on a Raspberry Pi
https://sarpi.penthux.net/index.php

[4] [5] [6] [7] NVMe SSD boot with the Raspberry Pi 5 | Jeff Geerling
https://www.jeffgeerling.com/blog/2023/nvme-ssd-boot-raspberry-pi-5

[8] [9] [10] [11] Security | RaspiBolt
https://raspibolt.org/guide/raspberry-pi/security.html

[12] [13] [14] Tor on Raspberry Pi
https://jharrington.io/hidden-svc

[15] 10 ways to analyze binary files on Linux | Opensource.com
https://opensource.com/article/20/4/linux-binary-analysis

[16] Analysing Binary Files using xxd - Gigi Labs
https://gigi.nullneuron.net/gigilabs/analysing-binary-files-using-xxd/