

# Comprehensive Systems Architecture and Security Audit Report: The Born2beRoot Project

## 1. Executive Summary: The Transition to Infrastructure Competence

The Born2beRoot project represents a pivotal juncture in the 42 School curriculum, marking the transition from application-layer programming to systems administration and infrastructure architecture. Unlike previous algorithmic challenges, this project mandates the rigorous deployment of a hardened, virtualized Linux server, adhering to a "Strict Rules" paradigm that simulates enterprise-grade security baselines. The objective is not merely to install an operating system but to construct a secure, monitored, and resilient environment using Infrastructure as Code (IaC) logic manually applied.

The core pedagogical goal is the cultivation of "Defense in Depth." By enforcing encrypted storage (LVM/LUKS), strict access controls (Sudo/PAM), mandatory access control (AppArmor/SELinux), and packet filtering (UFW), the project forces a confrontation with the layers of abstraction that govern modern computing. The successful implementation of Born2beRoot requires a synthesis of virtualization theory, kernel-level storage management, and cryptographic protocol configuration. This report provides an exhaustive analysis of these domains, deconstructing the theoretical underpinnings of the Linux boot process, the mechanics of logical volume management, and the intricate syntax of bash scripting required for system monitoring.<sup>1</sup>

## 2. Virtualization Architecture and Hypervisor Technology

### 2.1 Hypervisor Mechanics: Type 2 Implementation

The project mandates the use of a Type 2 hypervisor, specifically VirtualBox (or UTM for Apple Silicon architectures). To understand the constraints and performance characteristics of the Born2beRoot VM, one must distinguish between bare-metal and hosted virtualization.

Type 2 hypervisors operate as a software application running atop a host operating system (Windows, macOS, or Linux). Unlike Type 1 hypervisors (e.g., Xen, ESXi), which have direct access to hardware resources, VirtualBox must translate guest instructions for the host kernel. When the Born2beRoot VM is initialized, the hypervisor allocates a **Virtual CPU (vCPU)**. This vCPU is a software construct—a thread scheduled by the host's kernel on a

physical core. The project requirements often specify a limited number of vCPUs and RAM (e.g., 1GB), forcing the administrator to optimize the guest OS for resource constraints.<sup>3</sup>

The mechanism of execution involves **Binary Translation** and **Hardware-Assisted Virtualization (VT-x/AMD-V)**. In modern contexts, VirtualBox leverages CPU extensions (VT-x) to allow the guest kernel to execute privileged instructions directly on the processor in "VMX non-root operation." This reduces the overhead previously associated with trapping and emulating every privileged call. However, I/O operations (disk access, network packets) still require context switching between the guest and the host, introducing latency that the system administrator must account for.<sup>4</sup>

## 2.2 Virtual Storage and Networking

The virtual disk created for the project (VDI or QCOW2 format) acts as a container file on the host. To the guest OS, this appears as a block device (e.g., /dev/sda). The hypervisor intercepts SCSI/SATA commands sent by the guest driver and translates them into file I/O operations on the host's filesystem.

Networking presents a critical architectural decision. The project generally defaults to **NAT (Network Address Translation)** or **Bridged Adapter**.

- **NAT:** The VM sits behind a virtual router created by the hypervisor. The VM can access external networks, but external machines (including the host) cannot initiate connections to the VM without **Port Forwarding**. This is why the project often requires forwarding Host Port 4242 to Guest Port 4242 to facilitate SSH access.<sup>6</sup>
- **Bridged:** The VM attaches directly to the host's physical network adapter, obtaining an IP address from the local router. This makes the VM a distinct citizen on the LAN, accessible by other devices, but exposes it to local network threats.

## 2.3 The Ban on Snapshots: State Integrity

A strict requirement of Born2beRoot is the prohibition of snapshots in the final submission. Technically, a snapshot is a "frozen" state of the disk image and memory. When a snapshot is taken, the hypervisor creates a differencing disk; subsequent writes are recorded there, preserving the original image. The ban forces students to maintain a linear, stable system state, preventing the "save-scumming" of configuration errors. It ensures that the signature.txt (a SHA1 hash of the virtual disk) matches the state of the machine at the exact moment of submission, guaranteeing integrity during the defense.<sup>1</sup>

## 3. The Linux Boot Process: From BIOS to Login

Troubleshooting the Born2beRoot project—specifically issues with encrypted partitions not mounting—requires a granular understanding of the Linux boot sequence.

### 3.1 BIOS/UEFI and the Bootloader

The process begins with the **Basic Input/Output System (BIOS)** or **Unified Extensible Firmware Interface (UEFI)**. The firmware performs the Power-On Self-Test (POST) and scans for bootable devices. In a VirtualBox environment, this hands control to the Master Boot Record (MBR) or the EFI System Partition (ESP).<sup>9</sup>

**GRUB2 (Grand Unified Bootloader version 2)** is the standard bootloader for Debian.

1. **Stage 1:** A tiny binary in the MBR (first 512 bytes) points to the next stage.
2. **Stage 1.5:** Located in the gap between the MBR and the first partition, containing filesystem drivers (e.g., ext4).
3. **Stage 2:** Loads the menu interface from `/boot/grub/grub.cfg`.  
Crucially, GRUB must support LUKS to some extent if the `/boot` partition is encrypted, though standard practice (and the project default) leaves `/boot` unencrypted to simplify the loading of the kernel (`vmlinuz`) and the initial RAM disk (`initramfs`).<sup>9</sup>

### 3.2 Kernel Initialization and Initramfs

Once GRUB loads the kernel into memory, the kernel initializes hardware drivers. However, because the root filesystem (`/`) is encrypted, the kernel cannot mount it immediately. This presents a "chicken and egg" problem: the tools to decrypt the disk are *on* the encrypted disk.

The solution is the **initramfs (Initial RAM Filesystem)**. This acts as a temporary, small root filesystem loaded into memory. It contains:

1. The `dm-crypt` kernel module.
2. The `cryptsetup` binaries.
3. A script (`init`) that pauses the boot process and prompts the user for the LUKS passphrase.

Upon successful entry of the passphrase, the `init` script unlocks the partition, maps it to `/dev/mapper/cryptroot`, and mounts the real root filesystem. The kernel then uses `switch_root` to discard the `initramfs` and pivot to the real OS.<sup>11</sup>

### 3.3 Systemd and Service Management

The kernel launches the first user-space process, **PID 1**, which on Debian is **systemd**. Systemd is an initialization system that parallelizes service startups using "units."

- **Targets:** Instead of runlevels, systemd uses targets. `default.target` typically links to `graphical.target` or `multi-user.target`. Since Born2beRoot forbids a GUI, the system must boot to `multi-user.target` (CLI mode).
- **Service Dependencies:** Systemd reads unit files (e.g., `ssh.service`, `ufw.service`) to determine startup order. It ensures networking is up before starting SSH, and mounts filesystems (defined in `/etc/fstab`) before starting logging services.<sup>13</sup>

## 4. Storage Architecture: LVM and Encryption

### 4.1 Logical Volume Management (LVM) Internals

LVM provides a layer of abstraction between the physical storage and the filesystem, allowing for dynamic resizing and flexible management.

Component	Description	Project Context
Physical Volume (PV)	The raw block device (e.g., /dev/sda5) initialized for LVM use.	The encrypted LUKS container is formatted as a PV.
Volume Group (VG)	A pool of storage aggregated from one or more PVs.	A single VG (e.g., LVMGroup) is created to span the encrypted space.
Logical Volume (LV)	Virtual partitions carved from the VG.	root, home, swap, var are LVs formatted with ext4.
Physical Extent (PE)	The smallest chunk of data (usually 4MB) moved by LVM.	LVs are chains of PEs.

The advantage of LVM in this project is the decoupling of the filesystem from physical cylinder boundaries. If the /var partition (containing logs) fills up, the administrator can add a new virtual disk, extend the VG, and extend the var LV without unmounting or rebooting—a critical capability in enterprise environments.<sup>15</sup>

### 4.2 LUKS: Cryptography at Rest

**Linux Unified Key Setup (LUKS)** manages the encryption of the PV. It uses the kernel's dm-crypt subsystem.

- **Header:** The LUKS header contains the metadata necessary to decrypt the drive, including the cipher used (e.g., aes-xts-plain64) and the salt.
- **Keyslots:** LUKS supports multiple keyslots (passphrases). The user's passphrase decrypts a Master Key stored in the keyslot. This Master Key is what actually decrypts the data. This architecture allows changing the password without re-encrypting the entire drive (by re-encrypting the Master Key in the keyslot).<sup>11</sup>
- **Cipher Selection:** AES (Advanced Encryption Standard) in XTS mode is the industry standard for disk encryption. XTS prevents "watermarking" attacks where identical data

blocks produce identical ciphertext at different locations on the disk.

The project requires at least two encrypted partitions. This is typically achieved by creating a partition scheme where the LVM Volume Group resides inside a single large LUKS container. Thus, every Logical Volume created (root, swap, home) is inherently encrypted.<sup>1</sup>

## 5. Operating System Internals: Debian vs. CentOS

The project requires a choice between Debian and Rocky Linux (a CentOS fork). This analysis strongly recommends Debian for its alignment with the project's complexity curve.

### 5.1 Package Management: APT vs. DNF

- **Debian (APT):** Uses .deb packages. apt handles dependency resolution by querying local caches of repositories defined in /etc/apt/sources.list. dpkg is the backend tool that actually installs the files.
  - **Aptitude:** A high-level interface for apt. It features a text-based UI (TUI) and sophisticated heuristics for resolving dependency conflicts (e.g., suggesting downgrades to satisfy version requirements). The project requires understanding the difference: apt is for automation/scripts; aptitude is for interactive management.<sup>19</sup>
- **Rocky (DNF):** Uses .rpm packages. dnf (Dandified YUM) resolves dependencies using a SAT solver, which is mathematically rigorous but can be slower.

### 5.2 Mandatory Access Control (MAC): AppArmor

Debian uses **AppArmor** by default. AppArmor binds security profiles to *programs* (path-based).

- **Mechanism:** When a process (e.g., /usr/sbin/tcpdump) attempts a syscall (open file, bind port), the kernel queries the AppArmor profile. If the action isn't whitelisted, it is denied (Enforce mode) or logged (Complain mode).
- **Project Requirement:** AppArmor must be active at boot. Verification is performed via aa-status.
- **Comparison:** Rocky Linux uses **SELinux**, which uses a labeling system (inodes are tagged with contexts). SELinux is more granular but significantly more complex to configure, making Debian the preferred choice for this level of training.<sup>21</sup>

## 6. Security Engineering: Identity, Access, and Firewalling

### 6.1 Password Policy Implementation (PAM)

Linux authentication is modular, handled by **PAM (Pluggable Authentication Modules)**. The

project requires a stringent password policy, enforced via libpam-pwquality.

### Configuration Breakdown:

- **Aging (/etc/login.defs):**
  - PASS\_MAX\_DAYS 30: Forces rotation every 30 days.
  - PASS\_MIN\_DAYS 2: Prevents users from immediately changing a password back to an old one.
  - PASS\_WARN\_AGE 7: UX requirement to warn users.
- **Complexity (/etc/security/pwquality.conf):**
  - minlen=10: Minimum character length.
  - dcredit=-1, ucredit=-1, lcredit=-1: The negative sign makes these *mandatory* requirements (must have digit, uppercase, lowercase) rather than just adding to a complexity score.
  - maxrepeat=3: Prevents patterns like AAA.
  - difok=7: Requires 7 characters to be different from the previous password during a change.
  - enforce\_for\_root: This is the critical "gotcha" of the project. By default, root is immune to policy checks. This flag must be explicitly enabled in common-password or pwquality.conf to meet the "Strict Rules" criteria.<sup>23</sup>

## 6.2 Sudo Auditing and Configuration

The sudo command allows privilege escalation. The project transforms sudo from a convenience tool into an auditing engine via /etc/sudoers (edited via visudo).

- **I/O Logging:** Defaults log\_input, log\_output. This captures the keystrokes (stdin) and screen output (stdout/stderr) of every sudo session, storing them in /var/log/sudo/. This allows forensic reconstruction of administrative actions using sudoreplay.
- **TTY Enforcement:** Defaults requiretty. This prevents scripts or cron jobs (which have no terminal attached) from running sudo commands, mitigating automated exploit scripts.
- **Path Restriction:** secure\_path. Limits the binaries root can execute to trusted system directories (/usr/sbin, etc.), preventing attacks where a user places a malicious binary named ls in their home folder.<sup>1</sup>

## 6.3 Network Security: UFW and SSH

### SSH Hardening:

- **Port 4242:** Changing the port from 22 is "Security through Obscurity." While sophisticated scanners will still find it, it reduces log noise from automated botnets targeting default ports.
- **Root Login:** PermitRootLogin no. This is a fundamental security control. It forces an attacker to compromise two accounts: a standard user first, and then the root account (via sudo or privilege escalation), doubling the effort required.<sup>7</sup>

UFW (Uncomplicated Firewall):

UFW manages the kernel's iptables or nftables. The policy must be `DEFAULT_INPUT_POLICY="DROP"`. Only port 4242 is whitelisted. This creates a "default deny" posture, where no traffic is allowed unless explicitly permitted—the gold standard for server security.<sup>29</sup>

## 7. System Monitoring: The Bash Script

The `monitoring.sh` script is the synthesis of the student's ability to interrogate the kernel via the command line. It must broadcast system metrics every 10 minutes using cron and wall.

### 7.1 Metric Extraction Strategy

Metric	Command/Source	Explanation
Architecture	<code>uname -a</code>	Returns kernel version and hardware platform.
Physical CPUs	<code>/proc/cpuinfo</code>	Grep for physical id, sort, and count unique entries to handle multi-core sockets.
vCPUs	<code>/proc/cpuinfo</code>	Grep for processor lines. Direct count of threads visible to the OS. <sup>31</sup>
RAM Usage	<code>free -m</code>	Requires awk to calculate $\text{used} / \text{total} * 100$ for percentage.
Disk Usage	<code>df -h --total</code>	Must parse the total line. Filters are needed to exclude tmpfs or udev.
CPU Load	<code>top / vmstat</code>	<code>vmstat 1 2</code> is preferred. <code>vmstat</code> requires two samples to calculate load; the first sample is the average since boot, the second is current. Subtracting the id (idle)

		column from 100 gives utilization. <sup>32</sup>
<b>Last Boot</b>	who -b	Extracts the boot timestamp from /var/run/utmp.
<b>LVM Status</b>	lsblk	Checks if any partition has the type lvm. `if [ \$(lsblk
<b>TCP Connect</b>	ss -ta	netstat is deprecated. ss queries kernel netlink sockets. Filter for ESTABLISHED.
<b>User Count</b>	users / who	Counts active TTY sessions.
<b>Sudo Usage</b>	journalctl	`journalctl _COMM=sudo

## 7.2 Automation via Cron

The script is scheduled in the root crontab (sudo crontab -u root -e).

- Syntax: \*/10 \* \* \* \* /usr/local/bin/monitoring.sh
- **The wall Command:** The script pipes output to wall (Write All), which broadcasts the text to the message buffer of all logged-in TTYs. This simulates a system-wide alert mechanism.<sup>36</sup>

## 8. Bonus Implementation: The LLMP Stack

The bonus part requires setting up a functional WordPress site using **Lighttpd**, **MariaDB**, and **PHP** (LLMP).

### 8.1 Lighttpd vs. Apache

Lighttpd is chosen for its event-driven architecture (asynchronous I/O), making it lighter on memory than Apache's process-driven model—ideal for the constrained VM resources of this project.

### 8.2 FastCGI and PHP

Unlike Apache, which can embed PHP directly (mod\_php), Lighttpd uses **FastCGI**. The web

server forwards .php requests to a separate process (PHP-FPM) listening on a socket. The configuration requires enabling mod\_fastcgi and defining the socket path (e.g., /run/php/php7.4-fpm.sock). This separation enhances stability; if PHP crashes, the web server remains up.<sup>37</sup>

## 8.3 MariaDB and WordPress

- **Security:** mysql\_secure\_installation must be run to remove the test database and anonymous users.
- **Deployment:** WordPress connects to MariaDB via localhost. The wp-config.php file contains the database credentials.
- **Bonus Partitioning:** The bonus mandates separating /var/log, /tmp, /srv, and /home into their own LVs. This prevents a log flood in /var/log from filling up /, ensuring the system doesn't crash during a DDoS or error loop.<sup>1</sup>

# 9. Validation Strategy and Evidence Checklists

## 9.1 The Signature.txt

The critical validation step is the SHA1 signature of the virtual disk (.vdi).

- **Warning:** Booting the VM changes the file hash (logs are written, access times update).
- **Procedure:** A snapshot or clone of the VM *in the submission state* must be kept. The signature provided in the git repo must match the disk *before* it is booted for the defense. Any mismatch results in an automatic zero.<sup>1</sup>

## 9.2 Evidence Checklist for Defense

Requirement	Command/Validation	Expected Output
OS Check	hostnamectl	Debian/Rocky Linux
User Groups	groups <user>	sudo, user42
Password Policy	chage -l <user>	Max: 30, Min: 2, Warn: 7
Sudo Logging	ls /var/log/sudo/	Log files present (00/00/01...)
UFW Rules	sudo ufw status numbered	4242/tcp ALLOW, default deny

<b>SSH Port</b>	grep Port /etc/ssh/sshd_config	Port 4242
<b>Root SSH</b>	grep PermitRootLogin	no
<b>LVM</b>	lsblk	Tree structure showing LVM type
<b>Cron</b>	sudo crontab -l	*/10... monitoring.sh

---

## Works cited

1. en.subject1.pdf
2. Born2beroot | École 42 Project Notes - GibbonTech, accessed December 3, 2025, <https://www.gibbontech.com/ecole42/born2beroot/index.html>
3. What's a virtual machine - Guide, accessed December 3, 2025, <https://42-cursus.gitbook.io/guide/1-rank-01/born2beroot/whats-a-virtual-machine>
4. What is a vCPU and How to Calculate vCPU Requirements? - Cộng Đồng Linux, accessed December 3, 2025, <https://congdonglinux.com/what-is-a-vcpu-and-how-to-calculate-vcpu-requirements/>
5. What Is vCPU and How to Calculate vCPU? | phoenixNAP KB, accessed December 3, 2025, <https://phoenixnap.com/kb/what-is-a-vcpu>
6. Born2beroot. 42 school project | by Baigalmaa Baatar - Medium, accessed December 3, 2025, <https://baigal.medium.com/born2beroot-e6e26dfb50ac>
7. Born2beRoot - Sumi Garden, accessed December 3, 2025, <https://notes.devnyxie.com/0-Notes/unix/Born2beRoot>
8. Signature.txt | Born2BeRoot Guide - GitBook, accessed December 3, 2025, <https://noreply.gitbook.io/born2beroot/virtual-machine-setup/signature.txt>
9. Boot Process in Linux: Step-by-Step Guide & Troubleshooting - CyberPanel, accessed December 3, 2025, <https://cyberpanel.net/blog/boot-process-in-linux>
10. Understanding the Linux Boot Process & System Initialization, accessed December 3, 2025, <https://iies.in/blog/understanding-the-linux-boot-process-system-initialization/>
11. Linux Unified Key Setup - Wikipedia, accessed December 3, 2025, [https://en.wikipedia.org/wiki/Linux\\_Unified\\_Key\\_Setup](https://en.wikipedia.org/wiki/Linux_Unified_Key_Setup)
12. cryptsetup - manage plain dm-crypt and LUKS encrypted volumes - Ubuntu Manpage, accessed December 3, 2025, <https://manpages.ubuntu.com/manpages/focal/man8/cryptsetup.8.html>
13. Chapter 2 – Linux Boot Process (BIOS, POST, GRUB, Kernel, initramfs, systemd) -

- YouTube, accessed December 3, 2025,  
<https://www.youtube.com/watch?v=yX1VuY0xORY>
14. Introduction to the boot process | Administration Guide | SLED 15 SP7, accessed December 3, 2025,  
<https://documentation.suse.com/sled/15-SP7/html/SLED-all/cha-boot.html>
  15. Logical Volume Manager (Linux) - Wikipedia, accessed December 3, 2025,  
[https://en.wikipedia.org/wiki/Logical\\_Volume\\_Manager\\_\(Linux\)](https://en.wikipedia.org/wiki/Logical_Volume_Manager_(Linux))
  16. Logical Volume Management (LVM) in Linux | phoenixNAP KB, accessed December 3, 2025, <https://phoenixnap.com/kb/lvm-linux>
  17. cryptsetup(8) - Linux manual page - man7.org, accessed December 3, 2025,  
<https://www.man7.org/linux/man-pages/man8/cryptsetup.8.html>
  18. AGolz/Born2beRoot: :package: The goal of this project is to get acquainted with the amazing world of virtualization. - GitHub, accessed December 3, 2025,  
<https://github.com/AGolz/Born2beRoot>
  19. What is the real difference between "apt-get" and "aptitude"? (How about "wajig"?), accessed December 3, 2025,  
<https://unix.stackexchange.com/questions/767/what-is-the-real-difference-between-apt-get-and-aptitude-how-about-wajig>
  20. Debian Package Management: Aptitude vs. Apt-Get in Ubuntu | Linux Journal, accessed December 3, 2025,  
<https://www.linuxjournal.com/content/debian-package-management-aptitude-vs-apt-get-ubuntu>
  21. AppArmor vs SELinux: Compare the Differences in Linux Security - TuxCare, accessed December 3, 2025, <https://tuxcare.com/blog/selinux-vs-apparmor/>
  22. Core Differences Between SELinux and AppArmor | Baeldung on Linux, accessed December 3, 2025, <https://www.baeldung.com/linux/selinux-vs-apparmor>
  23. pwquality.conf(5) — libpwquality-common — Debian testing, accessed December 3, 2025,  
<https://manpages.debian.org/testing/libpwquality-common/pwquality.conf.5.en.html>
  24. Need to set Password complexity for root in ubuntu 20.04, accessed December 3, 2025,  
<https://askubuntu.com/questions/1479361/need-to-set-password-complexity-for-root-in-ubuntu-20-04>
  25. How to enforce password complexity policies in Linux - LabEx, accessed December 3, 2025,  
<https://labex.io/tutorials/linux-how-to-enforce-password-complexity-policies-in-linux-414805>
  26. Useful Sudoers Configuring sudo Command in Linux - GeeksforGeeks, accessed December 3, 2025,  
<https://www.geeksforgeeks.org/linux-unix/useful-sudoers-configurations-for-setting-sudo-in-linux/>
  27. sudo policies | Born2BeRoot Guide - GitBook, accessed December 3, 2025,  
<https://noreply.gitbook.io/born2beroot/virtual-machine-setup/sudo-policies>
  28. How to fix the SSH "Connection refused" error, accessed December 3, 2025,

- <https://kinsta.com/blog/ssh-connection-refused/>
29. Miami05/Born2beRoot: Born2beroot is a 42School project designed to build foundational skills in cybersecurity. It covers Linux system administration, network security, firewall configuration, user and group management, and secure service setups. Through hands-on tasks, students learn to identify vulnerabilities, implement security measures, and ensure system integrity. - GitHub, accessed December 3, 2025, <https://github.com/Miami05/Born2beRoot>
  30. 5 Ways to Fix the SSH Connection Refused Error [SOLVED] - RunCloud, accessed December 3, 2025, <https://runcloud.io/blog/fix-ssh-connection-refused>
  31. How to find the number of CPU cores including virtual? - Ask Ubuntu, accessed December 3, 2025, <https://askubuntu.com/questions/724228/how-to-find-the-number-of-cpu-cores-including-virtual>
  32. Get Overall CPU Usage on Linux. 1. Introduction | by Shalin Patel | Medium, accessed December 3, 2025, <https://medium.com/@shalinpatel/get-overall-cpu-usage-on-linux-785f2e4608bd>
  33. command line - Getting cpu usage realtime - Ask Ubuntu, accessed December 3, 2025, <https://askubuntu.com/questions/274349/getting-cpu-usage-realtime>
  34. wall command in Linux with Examples - GeeksforGeeks, accessed December 3, 2025, <https://www.geeksforgeeks.org/linux-unix/wall-command-in-linux-with-examples/>
  35. lighttpd - ArchWiki, accessed December 3, 2025, <https://wiki.archlinux.org/title/Lighttpd>
  36. How to Install Lighttpd on Debian 9 | RoseHosting, accessed December 3, 2025, <https://www.rosehosting.com/blog/how-to-install-lighttpd-on-debian-9/>