Hackathon Assignment: Comprehensive Penetration Test of Ubuntu Touch

---

Title: *"Securing the Future: Penetration Testing Ubuntu Touch"*

Objective: Conduct a thorough penetration test of **Ubuntu Touch** (running on a device like the Volla X23 or PinePhone) to identify vulnerabilities, exploits, and security weaknesses. Participants will simulate real-world attack scenarios, document their methodology, and provide a detailed report with proof of findings. The goal is to improve the security posture of Ubuntu Touch and contribute actionable insights to the open-source community.

---

1. Scope and Target

- **Target System:** Ubuntu Touch (latest stable version) running on a supported device (e.g., Volla X23, PinePhone, or emulator).
- **Scope:**
    - System-level vulnerabilities (kernel, permissions, sandboxing).
    - Application security (pre-installed apps, Open Store apps).
    - Authentication mechanisms (sudo pincode, lock screen, 2FA if available).
    - Network services and communication protocols.
    - Physical security (e.g., USB debugging, recovery mode).
    - Social engineering risks (e.g., phishing, user education gaps).
- **Out of Scope:**
    - Hardware-level exploits (unless directly related to Ubuntu Touch).
    - Denial-of-Service (DoS) attacks that could permanently damage the device.
    - Attacks requiring physical theft of the device (e.g., chip-off analysis).

---

2. Deliverables

Each team must submit the following by the end of the hackathon:

A. Penetration Test Report

A structured report (PDF or Markdown) including:

1. **Executive Summary:**
    - Overview of the target system and objectives.
    - Summary of critical findings and risk levels (e.g., CVSS scores).
    - High-level recommendations for mitigation.
2. **Methodology:**
    - Tools used (e.g., `adb`, `Linux Exploit Suggester`, `AppArmor` analysis tools, custom scripts).
    - Steps taken to enumerate, exploit, and post-exploit the system.
    - Ethical considerations and compliance with responsible disclosure.
3. **Findings:**

    - Detailed description of each vulnerability/exploit discovered.
    - Proof of Concept (PoC): Screenshots, logs, or code snippets demonstrating the exploit.
    - Impact assessment (e.g., privilege escalation, data leakage, persistence).
    - References to CVEs or existing research (if applicable).
4. **Mitigation Strategies:**
    - Technical fixes (e.g., patches, configuration changes).
    - Policy or user behavior recommendations (e.g., pincode complexity, 2FA).
5. **Appendices:**
    - Raw logs, command outputs, or scripts used during testing.
    - Timeline of activities (e.g., "Day 1: Reverse engineering; Day 2: Exploit testing").

B. Proof of Exploitation

- **Screenshots/Videos:** Show successful exploitation (e.g., gaining root access, bypassing sandboxing).
- **Code/Scripts:** Share custom tools or exploits (e.g., brute-force scripts, modified payloads).
- **Log Files:** Include relevant system logs (e.g., `dmesg`, `journalctl`, `AppArmor` violations).

C. Presentation

- A 10-minute live demo or recorded walkthrough of the most critical finding.
- Slides (optional) to summarize the approach and results.

---

3. Methodology Guidelines

Teams are encouraged to follow a structured approach:

Phase 1: Reconnaissance and Information Gathering

- **System Enumeration:**
    - Identify the Ubuntu Touch version, kernel, and installed apps.
    - Use `uname -a`, `dpkg -l`, `aa-status`, and `mount` to map the attack surface.
- **Documentation Review:**
    - Study Ubuntu Touch's official documentation and AppArmor profiles.
    - Research known vulnerabilities (e.g., CVE-2024-28085, `wall` exploit, `sudo` pincode weaknesses).

Phase 2: Vulnerability Assessment

- **Automated Scanning:**
    - Run tools like `lynis`, `linux-exploit-suggester`, or `mobsf` (for mobile apps).
    - Check for misconfigured permissions (e.g., `find / -perm -4000 -o -perm -2000 2>/dev/null`).
- **Manual Analysis:**

    - Review `/usr/bin/`, `/etc/sudoers`, and `/var/log/` for weaknesses.
    - Test default credentials and weak authentication (e.g., 4-digit pincode brute-forcing).

Phase 3: Exploitation

- **Privilege Escalation:**
    - Attempt to escalate from `phablet` user to `root` (e.g., via `sudo`, SUID/SGID binaries).
    - Test known exploits (e.g., `Wall-Escape`, `Dirty Pipe`).
- **Application Sandbox Escape:**
    - Bypass `AppArmor` or `Lomiri` confinement policies.
    - Test inter-app communication for data leakage.
- **Network Attacks:**
    - Sniff traffic (e.g., `tcpdump`) or test for MITM vulnerabilities in updates.
- **Physical Access:**
    - Exploit `adb` or recovery mode to extract data or flash malicious images.

Phase 4: Post-Exploitation

- Demonstrate impact (e.g., exfiltrate data, persist across reboots).
- Test detection/evasion (e.g., disable logging, hide processes).

Phase 5: Reporting

- Use the template provided below.
- Classify findings by severity (Critical/High/Medium/Low).

---

4. Rules and Ethics

- **Do Not Harm:** Avoid actions that could brick devices or violate privacy.
- **Responsible Disclosure:** Report critical findings to UBports Security Team after the hackathon.
- **Original Work:** Cite sources for borrowed exploits or tools.
- **Collaboration:** Teams of 2–4 members. Sharing tools/techniques between teams is encouraged.

---

5. Tools and Resources

Provided:

- Ubuntu Touch device (or emulator image).
- Pre-installed tools: `adb`, `gcc-aarch64-linux-gnu`, `binutils`, `linux-exploit-suggester`.
- Access to a private GitLab repo for submitting reports.

Recommended:

- **Reverse Engineering:** `Ghidra`, `radare2`, `objdump`.

- **Exploitation:** `Metasploit`, `sqlmap`, custom Bash/Python scripts.
- **Monitoring:** `strace`, `ltrace`, `AppArmor` logs.

---

4. Conclusion

- [Summary of risks and recommendations.]

5. Appendices

- **Logs:** [Attach log files.]
- **Scripts:** [Attach custom code.]