Security Audit Report

Project: Python Role-Based Access Control (RBAC) System
Language/Framework: Python 3.x (terminal application)
Date: 19 Sept 2025
Auditor: Irondi Ugochukwu

Executive Summary

The Python RBAC project was subjected to a security audit focusing on dependency vulnerabilities, static code analysis, and manual inspection of role-based access enforcement.

The purpose of this audit is to evaluate whether the application is resilient to common attacks such as privilege escalation, OS command injection, and dependency-based vulnerabilities.

Key Results:

- No **high-severity vulnerabilities** were found.
- **Low-severity issues** were identified, mainly involving the use *of os.system("cls").*
- **Dependency mismanagement** was noted, specifically unpinned versions *(pillow>=0)*.

Although the current risks are low, these issues could escalate into serious vulnerabilities if the application grows in complexity or if unsafe user input is introduced.

Scope of the Audit

This audit reviewed the following:

- **Source code:** *dashboard.py, help_desk.py, main.py* (945 lines total)
- **Dependencies:** Listed in *requirements.txt*
- **Environment:** Python 3.13, Windows OS
- **Threat Focus:**
    - Broken Access Control
    - Vulnerable Dependencies
    - OS Command Injection (CWE-78)
    - Least Privilege Violations

Methodology

Tools Used

- **Safety** – Dependency scanner comparing against *PyUp* vulnerability database.
- **pip-audit** – Official Python Packaging Authority tool for CVE detection.
- **Bandit** – Static code analyzer targeting common Python security issues.
- **Manual Review** – Line-by-line inspection of RBAC logic.

Process

1. Set up a virtual environment *(python -m venv .venv).*
2. Installed dependencies *(pip install -r requirements.txt).*
3. Ran dependency audits with *safety* and *pip-audit.*
4. Conducted static analysis with *bandit -r .*
5. Manually inspected role-checking functions, logging behavior, and password handling.

Threat Model for RBAC Systems

Role-Based Access Control systems are particularly sensitive to:

- **Privilege Escalation:** Attackers gaining higher-level roles (e.g., *Admin*).
- **Horizontal Privilege Bypass:** One user accessing another user's resources.
- **Weak Dependency Hygiene:** Outdated libraries enabling remote exploitation.
- **Audit Log Tampering:** Failure to record or protect critical actions.

By reviewing both dependencies and code, this audit ensures that the RBAC project resists these threats.

Findings

Dependency Findings

Finding: Unpinned dependency *(pillow>=0)*

- **Severity:** Medium
- **Description:** The requirements file permits installation of **any version** of Pillow. Safety reported **60 known vulnerabilities** across versions that meet this specifier.

- **Impact:** A developer installing dependencies later might unknowingly install a vulnerable version, exposing the project to exploits (e.g., CVE-2023-44271 – buffer overflow in Pillow).
- **CWE ID:** CWE-829 – Inclusion of functionality from untrusted sources.
- **Remediation:** Pin dependency to a secure version:

Code Findings

Finding: Insecure process execution *(os.system("cls"))*

- **Severity:** Low
- **Files Affected:**
  - *dashboard.py:5*
  - *help_desk.py:71*
  - *main.py:214, 349, 376, 415, 472, 600*
- **Description:** Multiple functions use *os.system("cls")* to clear the console. While harmless in current form (static string), it introduces two risks:
  - **OS Command Injection (CWE-78):** If user input is ever interpolated.
  - **PATH Hijacking:** Since "cls" is not an absolute path, an attacker could place a malicious executable earlier in the system PATH.
- **Representative Code Snippet:**

```
4  def it_admin_dashboard(user_dict):
5      os.system("cls")
```

- **Recommendation:** Replace with safer alternatives:

```
1  import subprocess, platform
2
3  def clear_screen():
4      if platform.system() == "Windows":
5          subprocess.run(["cmd", "/c", "cls"], check=True)
6      else:
7          subprocess.run(["clear"], check=True)
8
```

Manual Review Observations

- **Authentication:** Password hashing not visible in scanned code (must confirm bcrypt/argon2 is used).
- **Role Enforcement:** Role checks scattered across functions, increasing risk of inconsistent enforcement.

- **Logging:** Log function present but needs review to ensure no sensitive information (passwords, tokens) is written.
- **Session Handling:** Since this is a terminal app, session management is simpler, but transitions to web/mobile must include token validation.

Remediation Plan

Immediate Actions

1. **Pin dependencies** in *requirements.txt*
2. **Refactor** *os.system* **calls** to use *subprocess.*
3. **Re-run audits** to verify fixes.

Medium-Term Actions

1. **Centralize RBAC checks** in a middleware function, ensuring uniform enforcement.
2. **Introduce unit tests** for access control (e.g., normal user cannot access admin dashboard).
3. **Enable logging review** – redact sensitive data before writing to logs.

Long-Term Improvements

1. **Automated CI/CD Security Pipeline:** Integrate Bandit and Safety into GitHub Actions/GitLab CI.
2. **Regular Security Training:** Ensure developers are familiar with OWASP Top 10.
3. **Threat Modeling:** Perform structured modeling before adding new features.

Conclusion

The RBAC project demonstrates good coding practices overall, with no critical vulnerabilities identified. However, reliance on **unpinned dependencies** and repeated use of **os.system** weaken its security posture. By addressing these issues, the project can achieve higher resilience against dependency-based attacks and OS-level exploits.

Final Rating: Secure with **Low-Level Risks**, requiring immediate attention to dependency management.

```
Code scanned:
        Total lines of code: 945
        Total lines skipped (#nosec): 0

Run metrics:
        Total issues (by severity):
                Undefined: 0
                Low: 16
                Medium: 0
                High: 0
        Total issues (by confidence):
                Undefined: 0
                Low: 0
                Medium: 0
                High: 16
Files skipped (0):
```