Access Control Compliance Review Tool

Fernando Djingga
5 August 2025

Introduction

I created a simple Access Control Compliance Review Tool that cross-checks a user access list against the official role policy catalog. My project simulates how IT audit and tech risk teams perform User Access Reviews (UAR) in real organizations.

My program takes two inputs:

1. 'access_list.csv' consists of the user to role mapping including the department, status, and last login date.

2. 'policy_roles.csv' is the catalog of valid roles and which departments are allowed to use them

It then produces one output per run: 'violations.csv' is a machine-readable CSV file that contains all the detected violations with severity (HIGH/MEDIUM/LOW) and descriptive details for audit evidence.

My project is able to turn raw user access lists into actionable compliance evidence automatically. This mirrors how access control reviews are carried outs in professional workflows, such as in banks, fintechs, SaaS providers, and IT audit firms.

High-Level Implementation

Firstly, my program loads both input files into structured data models where 'AccessRow' objects for each user record (user_id, user_name, department, role, status, last_login), and 'PolicyRole' objects for each role policy (role, allowed_departments, description). Then my tool runs multiple checks against each row.

If there is an unknown role, then it will be flagged as HIGH severity. If the role is invalid and not allowed for department, it is flagged as HIGH severity. If the user is inactive but roles are still assigned, it is flagged as HIGH severity. If then status is not active, then it is flagged as MEDIUM severity. If there are duplicate assignments of pairs, then it is flagged as LOW severity. If the active user has not logged in for longer than the threshold of 90 days, then it is flagged as LOW severity.

All the violations are written as the output, which acts as a structured evidence for auditing, where it displays the aggregate totals by severity, issue type, and shows the top users with the most violations, which simulates real-life audit reports.

Coding Implementation

My code is implemented in a single main script at 'review.py'. I implemented the data models 'AccessRow', 'PolicyRole', and 'Violation' whicha re structured representations of input and

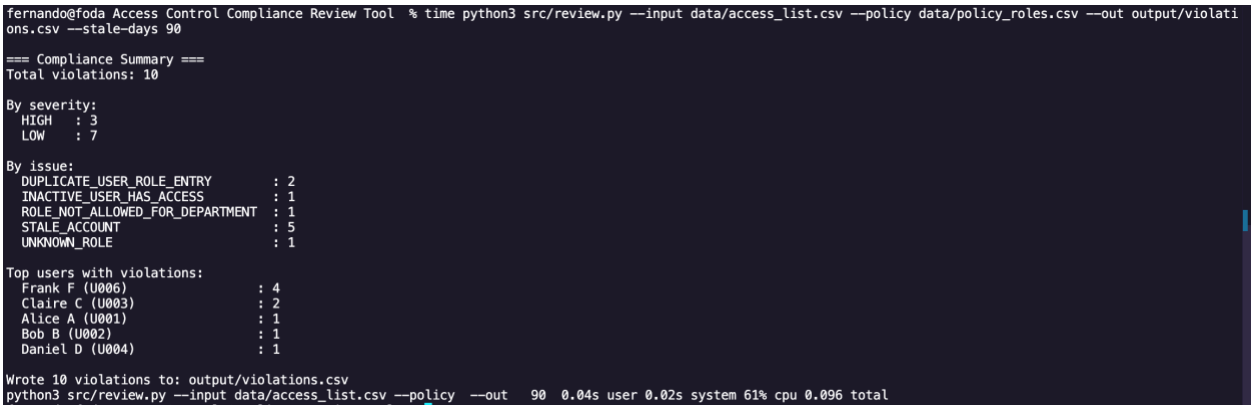
output. I also implemented the CSV readers ‘read_access_list’, ‘read_policy_roles’ to validate and parse the CSVs into objects. I used validation logic in ‘review_access’ to run all the six checks, and then use the writer ‘write_violations’ to output results into violations.csv. For the CLI interface, I used ‘argparse’ with flags ‘--input’, the path to ‘access_list.csv’; ‘--policy’, the path to ‘policy_roles.csv’; ‘--out’, the output path for ‘violations.csv’; and ‘--stale-days’, the threshold in days (which I set to 90 days).

My entire codebase uses only the Python standard library (csv, datetime, argparse, dataclasses) to ensure portability and no external dependencies.

Instructions on how to run my project are found in README.md.

Project Results

The following are the terminal results from my actual run.



The measured runtime of my program for 8 rows was 0.096 seconds.

If done manually, reviewing 8 rows would realistically take about 2 minutes (or 15 seconds per row). Extrapolating to 1000 rows, it would take my program 12 seconds, and about 250 minutes (15000 seconds) when done manually.

$$0.096 \div 8 \times 1000 = 12$$
$$15 \times 1000 = 15000$$

$$\frac{15000 - 12}{15000} \approx 99.99\%$$

The improvement is over 99.99%, which remarkably reduces hours of manual effort into just seconds, while also providing automated human-readable and machine-readable outputs.

Conclusion and Project Significance

In conclusion, my Access Control Compliance Review Tool demonstrates how audit checks that normally take hours of manual effort can be reduced to seconds with automation. The project provides significant scalability and time savings (that reduces hours of manual work into seconds), as well as ensuring accuracy where 100% of rows are checked consistently without any human oversight errors. It also generates structured CSV output as evidence that are suitable for regulators and auditors. Practically, it produces exception reports directly aligned with IT audit requirements.

As a fresh graduate, I designed my project with a focus on real-world IT risk and auditing needs. Every financial institution, SaaS provider, or regulated company must perform user access reviews, yet manual reviews are time-consuming and sometimes error-prone. By automating this process, my tool delivers measurable improvements in efficiency, accuracy, and consistency.

My project shows that I can translate theoretical knowledge of IT controls into working automation tools through Python programming, CSV processing, audit automation, and compliance reporting – all directly applicable to productive and professional workflows.