Take-Home Assignment

Secure, Low-Friction Access for Sensitive Users

Context

We build software used by law enforcement agencies and private-sector partners to handle highly sensitive data. Account takeover is a critical risk for us.

Today we have:

- Username/password login
- Optional 2-factor authentication (2FA)

We want something **meaningfully more secure** than this, **without adding heavy friction** for legitimate users.

Task

Design and implement a system that makes access to our platform **significantly harder to compromise** than basic login + optional 2FA, while keeping the day-to-day experience **as smooth as possible** for normal users.

You can approach this however you like:

- System design only
- Rules-based security logic
- Machine learning or anomaly detection
- Any combination of the above

Using real or synthetic data is **optional**. If you want to demonstrate or simulate behavior with data, that's great but not required.

Requirements

Your solution should:

1. **Improve Security Beyond Basic Login**

- Show how your design makes account takeover harder than simple username/password (+ optional 2FA).
- Consider both login and active sessions (not just the first login step).

2. **Minimize User Friction**
   - Clearly explain when a user has a smooth, no-extra-steps experience.
   - Explain when and why additional checks or friction are introduced.
3. **Support Risk-Aware Behavior (Design-Choice)**
   - You may choose to:
     - Use rules (e.g., certain conditions trigger extra checks), and/or
     - Use ML / scoring / anomaly detection.
   - We're interested in how you'd structure this, not in any specific algorithm.
4. **Be Practical to Implement**
   - Think about how this would fit into a real product:
     - Where does this logic live?
     - How does it talk to the existing auth system?
     - How would it evolve over time?

What to Deliver

1. **Short Design Document (2–3 pages max)**
   - A **high-level architecture diagram** showing:
     - Existing auth / login
     - Your additional security layer
     - Any data you would store or process
   - A brief description of:
     - The main components

        - How they interact
        - The key ideas behind your approach
    - A short “security vs friction” section:
        - How your design improves security
        - Where users might see friction and why that trade-off is acceptable

2. **Code (minimal / optional)**
    - A small backend service (language/framework of your choice) that:
        - Exposes an endpoint (or set of endpoints) that demonstrates your idea, e.g.:
            - Simulated login or session events
            - A decision about whether to allow, block, or add extra checks
    - You can mock or stub anything that isn’t central to your design.
3. **Optional: Demo or Data**
    - This is **not required**, but you may optionally include:
        - A minimal UI (web page, CLI, or notebook) that shows the flow.
        - Any simple empirical or simulated demonstration that:
            - Compares your approach to a basic login + optional 2FA baseline, and/or
            - Illustrates how your system behaves in a few example scenarios.

Evaluation

We will look for:

- **Security thinking**: Does your design meaningfully improve resistance to account takeover?
- **Friction awareness**: Are the user experience trade-offs thoughtfully handled and justified?
- **System design**: Is the architecture clear, modular, and realistic?

- **Implementation quality**: Is the code organized, readable, and aligned with your design?
- **(Optional) Use of data/ML**: If you choose to use data or ML, is it used thoughtfully and explained clearly?