BUSINESS PROBLEM SUMMARY

During a security review in my role as Database Administrator for a CMMS company supporting over 1,000 tenant SaaS databases and thousands of on-premises client installations, I uncovered a significant vulnerability in the way user credentials were stored.

Passwords were stored in cleartext in the user registration database. While there was an option for encryption, it was little more than superficial. Passwords were technically obfuscated—but not actually secured.

Key Issues Identified:

- The existing encryption implementation had several vulnerabilities, including encryption and decryption functions
    - The decryption function would return the cleartext password using only the encrypted value as input.
- Both functions were unencrypted, exposing the passphrase and logic to anyone with schema access, although this was partly moot, given the existence of the decryption function.

The most troubling part was some of the client base:

- Municipal water departments, public utilities, and critical infrastructure – organizations which frequently mapped their full infrastructure within the GIS module
- Healthcare organizations
- Public schools and universities
- The U.S. Department of Defense
- Plum Island Animal Disease Center (DHS National Laboratory)

This introduced two additional vulnerabilities:

- Most users reuse credentials across systems. Compromise of one password could provide access to other environments.
- The application granted unrestricted SQL access through built-in query tools and report builders—creating lateral access risks if credentials were compromised.

MY SOLUTION

Working within legacy system constraints, I designed and implemented a reusable T-SQL encryption framework to eliminate cleartext password storage entirely.

Although other concerns existed—such as insufficient write-level controls and injection risks—I prioritized addressing the most critical issue: eliminating credential exposure. Additional risks were documented and shelved for a future release, which would require coordination with the application development team, who were facing a rapidly approaching deadline for a major version upgrade.

Key goals included:

- Protecting user credentials from exposure, even to internal users with full query access.
- Designing a solution that integrated seamlessly with existing schema and business logic.
- Reducing the attack surface for database-level exploitation.

I selected AES-256 encryption, implemented via a wrapper that incorporated a passphrase salted with multiple user-specific data fields. This approach made reverse engineering impractical, even if the encrypted column was exposed.

MY SCHEMA UPDATES

While I cannot share specific T-SQL objects or encryption logic for ethical and legal reasons, I can outline the structural changes:

- Added a tenant-level flag to track encryption status for each container (multi-tenant environment).
- Created an encrypted password column in the user registration table.
- Implemented a trigger to manage encryption of newly created and modified passwords.
    - Encrypted the trigger to prevent reverse engineering of logic.
- Updated the user authentication, password check, and password reset stored procedures to support on-the-fly decryption.
    - Encrypted the stored procedures to prevent reverse engineering of logic.
- Developed deployment scripts to migrate and re-encrypt all existing passwords using the new method.
- Deprecated and removed the original encryption and decryption functions.

Third-party validation

While I developed this solution on my own initiative, it occurred at a very opportune time.

At the time, an implementation consultant was working to secure Apple as a client, and their compliance team required detailed documentation of our credential security practices. Management asked me to describe our current security practices. I informed them of the challenges that would arise if they presented the current security implementation, explained the existing vulnerabilities, described what I had built just a week prior, and provided a detailed walkthrough.

Apple's compliance engineers reviewed the full solution and raised no objections. They proceeded with the purchase and integration, indicating confidence in the system's design and intent.

Takeaways

This project reflects my ability to design pragmatic, security-conscious solutions within constrained environments—addressing critical vulnerabilities while respecting organizational boundaries and long-term architectural strategy.

It demonstrates:

- Initiative and ownership in the absence of direction
- A strong understanding of real-world security principles
- The ability to deliver high-impact solutions in complex systems