Link

Video: Lab 3_ThomasDuong.mp4

Code: https://github.com/DThomas230/FSCT-8561_Security-Applications

Part 6 – Reflection Questions

1. Why is hashing required for password storage?
    a. Hashing is required for password storage because it protects the privacy of the user. It transforms the password into a unique string of characters so that even if a database is breached, the actual password remains hidden and cannot be reversed by an attacker.
2. How does OTP mitigate replay attacks?
    a. OTP mitigates replay attacks because each code is valid for only one login session or a very short amount of time. If an attacker captures a used code, they cannot use it again because the server will recognize it as expired or already processed.
3. What happens if the client and server clocks are not synchronized?
    a. If the client and server clocks are not synchronized, the authentication will fail. Since TOTP rely on both parties having the exact same time to generate the matching code, a time gap will cause the server to reject a code as invalid or expired
4. What are the limitations of OTP-based authentication?
    a. The limitations of OTP-based authentication include a reliance on delivery methods (like SMS or email) which can be delayed or intercepted. It is also vulnerable to phishing sites that trick users into entering their OTP in real-time, and it can be difficult for users if they lose access to their physical device or generator.

Part 7 – Security Analysis

In 300–400 words, analyze the security implications of your authentication system. Your analysis must reference specific implementation details and address:

- Threats mitigated by OTP
- Remaining attack vectors
- Why authentication alone does not provide full security

- Suggested improvements for real-world deployment

My System implementation utilizes TOTP with the pyotp library which offers multi-factor authentication which would go a long way in limiting credential theft attacks. The two-factor authentication (password and OTP) is used so that even in case, the attacker hacked the password database, they will not be able to log-in without the secret of the OTP. Exposure of plaintext passwords is discouraged by the hash-based password algorithm (SHA-256) in password-hashing function, hash password. Even more so, the rate limiting feature that restricts failed logins to three in every five minutes is an effective measure against brute-force attacks on password and OTP authentication.

Irrespective of these safeguards, there are still essential weaknesses. The system sends all of the credentials: passwords, OTP codes, secrets, in plaintext via TCP through the encoding of the data in the form of a JSON, and it is susceptible to man-in-the-middle attacks and network sniffing. The users database dictionary is in-memory and does not offer data persistence or encryption, and so all user data is lost on server reboot and available in case of memory compromise. During registration, the OTP secret is sent to the client in plaintext and this gives an interception opportunity. Moreover, SHA-256 without salting is not sufficient to use in password hashing, the rainbow table attacks might break even hashed passwords. The implementation has no session handling- successful authentication does not present any token or method of dealing with successfully authenticated state between requests.

Authentication provides verification of identity, which happens during the initial authentication but not to maintain security of later communications and information. In the absence of transport encryption (TLS/SSL), the traffic after authentication is susceptible to interception, man in the middle attacks, and replay attacks. The system does not have any authorization controls to restrict what can be accessed by authenticated users, or audit logs to monitor security or prevent session hijacking.

To deploy production, TLS encryption should be used on all communications, password hashing should be done with bcrypt or Argon2 but use unique salts, credentials should be stored in an encrypted database with appropriate backup procedures, JWT or session tokens should be used to maintain authenticated state, extensive audit logs need to be implemented, and established OTP authenticator applications should be used rather than displaying secrets. It is possible to consider the use of certificate pinning, input validation against injection attacks, and periodic security audits..