Background Study

Demonstrating and Mitigating a Message Integrity Attack (MAC Forgery)

Names:
Sarah Ibrahim – 2205223
Rewan Khaled – 2205185
Karen Alfred – 2205236

1) What is a MAC and Its Purpose in Data Integrity and Authentication?

A Message Authentication Code (MAC) is a short cryptographic tag generated using a secret key and the message. It ensures message integrity and authenticity by allowing the recipient to verify that:

- The message has not been tampered with during transmission.
- The sender possessed the shared secret key, confirming their identity.

Purpose of MACs:

- **Data Integrity**: If the message is altered, the MAC verification will fail.
- **Authentication**: Only someone with the shared key can generate the correct MAC.

How MACs Are Generated and Verified:

- The sender computes `MAC = f(secret, message)` using a secure function (e.g., a keyed hash or encryption).
- The receiver uses the same secret key to recompute the MAC and compares it to the received one.
- A match confirms both integrity and authenticity.

2) How Does a Length Extension Attack Work in Hash Functions Like MD5/SHA1?

A length extension attack exploits how some hash functions (like MD5 and SHA1) are constructed using the Merkle–Damgård design.

How the Attack Works:

- These hash functions process messages in fixed-size blocks and apply automatic padding.
- When computing MAC = hash(secret || message), the internal state of the hash is exposed in the output.
- An attacker who knows `MAC` and `message` can:
  - Guess the length of `secret`.
  - Simulate the padding MD5/SHA1 would apply.
  - Append malicious data like `&admin=true`.
  - Continue the hashing process using the exposed internal state.

All this can be done without knowing the secret key.

Why It Breaks Security:

- The attacker can forge a valid `MAC` for `message || padding || extra_data`.
- This compromises both message integrity and authentication.

3) Why Is `MAC = hash(secret || message)` Insecure?

*1. **Vulnerability to Length Extension:***

- Hash functions like MD5, SHA-1 allow the attacker to extend the original message and forge a valid MAC by continuing the hash process with `additional_data`.

*2. **Breaks Authentication:***

- Anyone can generate a new valid MAC without knowing the secret, defeating the purpose of authentication.

Secure Alternatives:

To prevent length extension attacks, use HMAC, which is designed to be resistant:
HMAC(secret, message) = H((secret ⊕ opad) || H((secret ⊕ ipad) || message))
References:

- RFC 2104 – HMAC Spec
- OWASP – Message Authentication Code (MAC)