MAC Forgery: Background on Length Extension Attacks

Message Authentication Codes (MACs) are cryptographic tools designed to ensure the integrity and authenticity of a message. They operate by combining a secret key with the message, then applying a cryptographic hash function to produce a fixed-size tag (the MAC). This tag enables the receiver to verify that the message has not been altered and originates from a legitimate sender who shares the secret key.

Insecure MAC Construction

A common but insecure approach to constructing a MAC is by simply concatenating the secret key and the message before hashing, such as:
MAC = MD5(key || message)
where || denotes concatenation. This construction appears straightforward but suffers from a critical vulnerability known as a **length extension attack**.

Length Extension Attack Explained
Hash functions like MD5 and SHA-1 use a design known as the Merkle-Damgård construction. This design processes input in fixed-size blocks and maintains an internal state that is updated sequentially with each block.
The length extension attack exploits this property by allowing an attacker who knows:

- The hash output (MAC) of the original message,
- An estimate of the secret key length,

to compute a valid hash for a longer message that appends additional data, **without knowing the secret key**. This means the attacker can forge a MAC for message || malicious_data, tricking the receiver into accepting tampered data as authentic.

Why Does This Happen?

Because the internal state of the hash function after processing key || message can be inferred from the MAC, an attacker can continue hashing as if they had the secret key. The predictable padding and block processing allow this continuation.

Security Implications
Length extension attacks pose serious threats to systems that use insecure MAC constructions.
Attackers can:

- Modify messages without detection,
- Escalate privileges by injecting unauthorized commands or flags,
- Bypass authentication checks relying on the MAC.

This is especially dangerous in protocols that use simple hash(key || message) MACs for session validation, API authentication, or access control.

Mitigation Strategies
To defend against length extension attacks:

- Use **HMAC** (Hash-based Message Authentication Code), which applies a nested hash construction that hides internal states and breaks the length extension property.
- Prefer modern hash functions like **SHA-256** or **SHA-3**, where SHA-3's design resists length extension inherently.

- Always perform **constant-time MAC comparisons** to prevent timing side-channel attacks during verification.
- Include additional context such as message lengths or unique nonces in the MAC calculation to ensure message uniqueness.

Conclusion

Length extension attacks exploit structural weaknesses in common hash functions when used improperly as MACs. Employing HMAC and modern cryptographic practices is essential to ensure message integrity and authentication in secure systems.