Mitigating MAC Forgery Using HMAC

Mohamed Ahmed Ali Mubarak

Mariam Waleed

Amr Khaled

May 16, 2025

Abstract

Message Authentication Codes (MACs) play a critical role in ensuring data integrity and authenticity. However, naive implementations of MACs such as `hash(secret || message)` are vulnerable to length extension attacks. This report presents a secure mitigation strategy using HMAC (Hash-based Message Authentication Code), discusses its cryptographic advantages, and demonstrates its effectiveness against forgery attempts.

Introduction

Message Authentication Codes (MACs) are used to verify both the integrity and the authenticity of a message. They are particularly important in communication protocols and data exchange where security is a priority. However, not all MAC constructions are secure. For example, using a MAC like `hash(secret || message)` can be exploited through length extension attacks, potentially compromising the system.

Vulnerability in Traditional MAC Construction

The traditional approach of computing a MAC as `hash(secret || message)` suffers from a well-known weakness: length extension attacks. In this scenario, an attacker who knows the MAC of a message can append additional data to the message and compute a new valid MAC without knowing the secret key.

This occurs because many common hash functions, such as SHA-1 and MD5, are vulnerable to such attacks due to the way they process input in blocks and maintain internal state.

HMAC: A Secure Solution

To overcome this vulnerability, HMAC (Hash-based Message Authentication Code) was developed. HMAC combines a cryptographic hash function with a secret key using a well-designed construction that separates key handling from the message content.

HMAC Construction

The HMAC formula is as follows:

$$\text{HMAC}(key, message) = H((key \oplus \text{opad})||H((key \oplus \text{ipad})||message))$$

Here:

- H is a cryptographic hash function.
- $\oplus$ denotes bitwise XOR.
- `opad` and `ipad` are constant padding values.

This double hashing mechanism and use of key padding protect against the weaknesses present in simpler MAC constructions.

Why HMAC Prevents the Attack

HMAC effectively prevents length extension attacks because:

- The key is padded and hashed separately before being combined with the message.
- The internal hash state cannot be extended or forged without knowing the secret key.
- The structure of HMAC ensures that even if the attacker knows the message and the corresponding MAC, they cannot generate a valid MAC for an extended message.

Demonstration of Defense

To confirm the effectiveness of HMAC, we replaced the server-side MAC generation from `hash(secret || message)` to HMAC. When the same length extension attack script was re-executed, the forgery attempt failed.

The modified server correctly rejected the forged message and its MAC, thereby demonstrating that HMAC is a robust defense against this type of attack.

Conclusion

Length extension attacks represent a serious threat to naive MAC constructions. By implementing HMAC, which incorporates sound cryptographic principles and separates key handling from message hashing, we achieve secure message authentication resistant to such attacks. Our demonstration clearly shows that HMAC is a practical and effective solution for mitigating MAC forgery vulnerabilities.

Team Members

- Mohamed Ahmed Ali Mubarak
- Mariam Waleed
- Amr Khaled