**Mitigation Write-up**

**Why the Original Implementation Was Insecure**

* Used simple concatenation MAC (MD5(key || message))
* MD5 is vulnerable to length extension attacks
* No protection against timing attacks in MAC comparison

**Secure Implementation Features**

* **HMAC Construction**: Prevents length extension attacks by design
* **Stronger Hash Function**: Uses SHA-256 instead of MD5
* **Constant-Time Comparison**: Uses hmac.compare\_digest()

**Additional Security Recommendations**

**Key Management:**

* Use cryptographically random key generation
* Implement regular key rotation

**Algorithm Selection:**

* Consider SHA-3 (resistant to length extension)
* Follow NIST cryptographic standards

**Protocol Design:**

* Include message length in MAC calculation
* Use nonces to prevent replay attacks

**Defense in Depth:**

* Combine MAC with encryption when needed
* Use rate limiting to prevent brute force attacks

**Why HMAC is Secure Against Length Extension**

HMAC’s structure is:

HMAC(K, m) = H((K ⊕ opad) || H((K ⊕ ipad) || m))

* The **inner hash** destroys predictable structure
* The **outer hash** prevents extending the inner hash
* Even with MD5, HMAC construction prevents length extension attacks (though SHA-256 is preferred)