Hash-based Message Authentication

CMPU4023 - Enterprise Application Development

Authenticate Every Request

- An alternative to pre-authentication (and tokens) is to authenticate each request individually
- Ideally we'd like to be able do this without having to incur any additional API calls
- The context here is that the API calls are being made on behalf of some identity which needs to be authenticated
- And we assume that client and server possess identity credentials (including a shared secret) distributed in advance by some out-of-band means

Message Authentication

- A simple scheme which allows API calls to be authenticated with more extra messages is to securely authenticate each request using the shared key which works as follows:

1. The client generate a secure signature using on a HMAC (symmetric key) over the API message contents
2. This signature is sent along with the API request and a unique key to identify which user is making the request
3. On message receipt, the server, which also knows the shared secret key, generates the signature for the API call and compares it with the one being sent
4. If they match, then the server can assume that the client was in possession of the secret for the associated user (unique key) and is therefore authenticated

Security Properties

- It is assumed the the shared secret key is known only to the client and server and would be “hard” to guess (e.g. 320 bits)
- The signature is based on a known-secure hashing algorithm with a extremely low probability of collisions (e.g. SHA256)
- The hash is encrypted using a shared key symmetric-key algorithm (e.g. HMAC-SHA256)
- It is impossible for an attacker to generate the MAC of message and a secret key without knowing the secret key
- The message cannot be altered by an attacker and also pass the message integrity and signature check

Security Guidelines

- There is no standard which dictates how hash-based message authentication should be implemented but there are some good practice guidelines:

1. Use known-secure crypto algorithms (hashing and encryption for signature)
2. Include the message body (if any), the query parameters (if any), the resource path (assuming REST) and the unique key in the bytes block to be hashed
3. Randomise the request message signature to guard against replay attacks by, for example, including a variable component such as a timestamp
4. Independently encrypt the message (e.g. TLS) for transmission
5. Periodically reissue new secret keys and invalidate old ones

- In practice, enterprise solutions will provide for bespoke application-specific implementations of these guidelines

HTTP Message Authentication

- Let's consider an HTTP-based API request (e.g. REST) being authenticated using HMAC-SHA256
- Keeping with the principle of separating authentication metadata from API message data, we would see the authentication data being placed into a HTTP header field as in the following example:

```
Authorization: HMAC-SHA256 Key=7e96106333af9110bc50d4f3cbfc806b76cf1a3a
Signature=d44869e9bb60c5696dc72199388f96d89739a800891242509fbb827b9fee64
0988eafc7db540283d
```

- The encoding can be implementation specific - (e.g. could use base64 instead of hexadecimal)

Considerations

- Hash-based message authentication (when done properly) is secure and provides a high degree of confidence for authentication
- In theory, the per-request checking steps are no more expensive than pre-authentication and token verification schemes
- However, the HMAC verification may need to consult the authenticating authority each time so this can be more expensive and be less flexible and scalable in practice - (i.e. only the authority has a copy of the shared secret key)
- Generating signatures for API requests is non-trivial (a complex computation) and this adds some logical complexity to the client-side in terms of signing and solution testing

Summary

- HMAC is a per-request authentication and integrity checking scheme which is suitable for use in API security
- It is based on secure hashing and encryption building blocks and a shared secret key
- No pre-authentication or extra messages are required which makes it simpler on the one hand
- However, the client has the added responsibility to correctly compute API message signatures which requires tight agreement on how this is to be done for a wide variety of potential message types

Hash-based Message Authentication

CMPU4023 - Enterprise Application Development