Certificate Pinning

Overview

When communicating over public networks it is essential to send and receive information securely. The protocol widely used to secure these communications is SSL/TLS. (SSL/TLS refers to Secure Sockets Layer or to its successor, TLS, or Transport Layer Security). SSL/TLS uses digital certificates to provide authentication and encryption. To trust that a certificate is genuine and valid, it is digitally signed by a root certificate belonging to a trusted certificate authority (CA). Operating systems and browsers maintain lists of trusted CA root certificates so that they can easily verify certificates that the CAs have issued and signed.

Protocols that rely on certificate chain verification, such as SSL/TLS, are vulnerable to a number of dangerous attacks, including man-in-the-middle attacks, which occur when an unauthorized party is able to view and modify all traffic passing between the mobile device and the backend systems.

IBM MobileFirst Foundation provides an API to enable **certificate pinning**. It is supported in native iOS, native Android, and cross-platform Cordova MobileFirst applications.

Certificate pinning process

Certificate pinning is the process of associating a host with its expected public key. Because you own both the server-side code and the client-side code, you can configure your client code to accept only a specific certificate for your domain name, instead of any certificate that corresponds to a trusted CA root certificate recognized by the operating system or browser.

A copy of the certificate is placed in the application and is used during the SSL handshake (the first request to the server). The MobileFirst client SDK verifies that the public key of the server certificate matches the public key of the certificate that is stored in the app.

Important

- Some mobile operating systems might cache the certificate validation check result. Therefore, your code should call the certificate pinning API method **before** making a secured request. Otherwise, any subsequent request might skip the certificate validation and pinning check.
- Make sure to use only MobileFirst Foundation APIs for all communications with the related host, even after the certificate pinning. Using third-party APIs to interact with the same host might lead to unexpected behavior, such as caching of a non-verified certificate by the mobile operating system.
- Calling the certificate pinning API method a second time overrides the previous pinning operation.

If the pinning process is successful, the public key inside the provided certificate is used to verify the integrity of the MobileFirst Server certificate during the secured request SSL/TLS handshake. If the pinning process fails, all SSL/TLS requests to the server are rejected by the client application.

Certificate setup

You must use a certificate purchased from a certificate authority. Self-signed certificates are **not supported**. For compatibility with the supported environments, make sure to use a certificate that is encoded in **DER** (Distinguished Encoding Rules, as defined in the International Telecommunications Union X.690 standard) format.

The certificate must be placed in both the MobileFirst Server and in your application. Place the certificate as follows:

- In the MobileFirst Server (WebSphere Application Server, WebSphere Application Server Liberty, or Apache Tomcat): Consult the documentation for your specific application server for information about how to configure SSL/TLS and certificates.
- In your application:
  - Native iOS: add the certificate to the application **bundle**
  - Native Android: place the certificate in the **assets** folder
  - Cordova: place the certificate in the **app-name\www\certificates** folder (if the folder is not already there, create it)

Certificate pinning API

Certificate pinning consists of a single API method, that has a parameter `certificateFilename`, where `certificateFilename` is the name of the certificate file.

Android

```
WLClient.getInstance().pinTrustedCertificatePublicKey("myCertificate.cer");
```

The certificate pinning method will throw an exception in two cases:

- The file does not exist
- The file is in the wrong format

iOS

In Objective-C:

```
[[WLClient sharedInstance]pinTrustedCertificatePublicKeyFromFile:@"myCertificate.cer"];
```

In Swift:

```
WLClient.sharedInstance().pinTrustedCertificatePublicKeyFromFile("myCertificate.cer")
```

The certificate pinning method will raise an exception in two cases:

- The file does not exist
- The file is in the wrong format

Cordova

```
WL.Client.pinTrustedCertificatePublicKey('myCertificate.cer').then(onSuccess,onFailure);
```

The certificate pinning method returns a promise:

- The certificate pinning method will call the onSuccess method in case of successful pinning.
- The certificate pinning method will trigger the onFailure callback in two cases:
- The file does not exist
- The file is in the wrong format

Later, if a secured request is made to a server whose certificate is not pinned, the `onFailure` callback of the specific request (for example, `obtainAccessToken` or `WLResourceRequest`) is called.

> Learn more about the certificate pinning API method in the API Reference (http://www.ibm.com/support/knowledgecenter/SSHS8R_8.0.0/com.ibm.worklight.apiref.doc/apiref/c_client_api.html)

Last modified on