# Mobile App https encryption guideline

The author has talked to friends about various security issues of using https encryption in mobile app programming on many occasions. I remember that many years ago, my company's information security engineer tested several online banking apps and found that each one had https encryption security issues. I don't know the security situation now and whether it has been improved. However, the author has communicated with several App development engineers in different industries and found that the problems discovered many years ago still exist. To this end, the author took the time to organize this common security issue into a collection as a guideline for readers.

As we all know, mobile apps generally have registration and login operations, and interact with the server for various user confidential data. To protect the security of passwords and confidential information, the https protocol must be used to implement encrypted communication with the server. If https is not enabled, it must not be approved by the Apple App Store. However, if just enabling https encryption is not enough, there are five common security issues as follows.

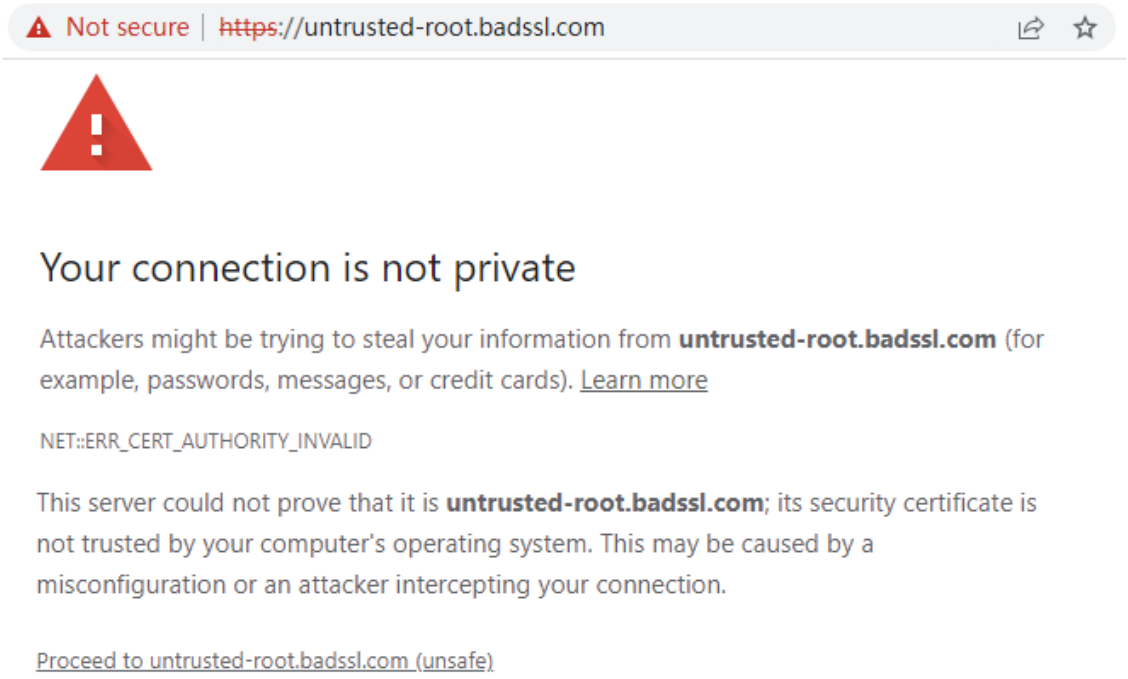
**First, not judge whether the domain name bound to the SSL certificate is correct**

If https is enabled to shake hands with the server, the server will return the domain name bound to the SSL certificate. If the domain name bound to the returned SSL certificate is inconsistent with the URL the user requests to connect, the connection should be terminated. For example, when the App initiates a connection to https://login.domaina.com, the returned SSL certificate is bound to the login.domainb.com domain name. if the App does not judge whether the domain name matches, it connects with it directly, and send the user data like bank account password to the fake online banking server! Most domain name mismatches are caused by man-in-the-middle attacks. This problem of not judging whether the domain names match allows attackers to easily obtain the user's bank account password (if the app is an online banking app). However, if the app can verify whether the domain name matches in real time, it can prevent such attacks.

**Second, not judge whether the SSL certificate is trustworthy**

This problem is also relatively common. I checked related issues on the Internet, and there are many articles that teach users to not need to call the certificate verification function. If the user knows that the domain name must be verified to match but does not know whether the server certificate is a certificate trusted by the operating system, or whether it is a certificate trusted by the App, it is also possible to encounter a man-in-the-middle attack after DNS hijacking, and the user's confidential information is handed to the fake online banking system. Because the self-signed SSL certificate that the operating system does not trust can be arbitrarily created by using the OpenSSL command, even if the App will verify whether the connected domain name matches the certificate, it will still be tricked because the App does not verify whether the SSL certificate is trusted.

A more reliable approach is not only to verify that the SSL certificate is a certificate trusted by the operating system, but also to verify whether the certificate is issued by a designated CA's issuing CA to prevent possible malicious attack on the SSL certificate. For some important systems, such as payment systems, the author recommends customizing the company's dedicated branding issuing Sub CA to issue SSL certificates for these systems, so that the App can only trust the SSL certificate issued by the company's dedicated issuing SUB CA, this is the best solution.



**Third, not judge whether the SSL certificate has been revoked**

Under normal circumstances, if the user suspects that the private key of the certificate may be leaked (such as the departure of key personnel or the attack on the server), the user must apply to the CA to revoke the certificate and re-apply for a new SSL certificate. If the app does not verify whether the certificate is revoked when shaking hands with the server, if the SSL certificate used by an online banking has indeed been leaked, the attacker can use this certificate to successfully implement a man-in-the-middle attack. However, if the App verifies whether the certificate has been revoked in real time, it can detect and terminate the connection in time, which can effectively prevent attackers from using the revoked certificate for man-in-the-middle attacks.

**Fourth, not judge whether the SSL certificate has expired**

This is a low-level mistake, but one that most apps make regularly. All SSL certificates have expiration date, and if they expire, they will be renewed with a new valid certificate. However, if the attacker obtains the expired SSL certificate of the online banking system and deploys the expired certificate for man-in-the-middle attack, if the App does not check whether the certificate has expired when shaking hands with the server, then it is attacked! However, if the App can check whether the certificate has expired in real time, once it finds that the certificate has expired, it will immediately terminate the connection, which can effectively prevent attackers from exploiting the expired certificate.

**Fifth, not support the SM2 algorithm and the SM2 SSL certificate**

This is a new problem, because in the current very uncertain international environment, it is necessary for the important website to deploy the SM2 SSL certificate, so the mobile app must support the SM2 algorithm and the SM2 SSL certificate, then it can correctly implement the SM2 https encryption with the server. If the SM2 algorithm is not supported, the RSA algorithm can only be used to communicate with the server for https encryption, but once the RSA SSL certificate used for https encryption is revoked or the supply is cut off, the mobile app cannot achieve normal encrypted communication, and the App data communication security cannot be guaranteed.

This is especially worthy of the high attention for all commonly used mobile apps. Start upgrading and transforming the app as soon as possible to support the SM2 algorithm and the SM2 SSL certificate. Only in this way can App users still can use the mobile app normally even the RSA SSL certificate is revoked, and the normal operation of mobile business can be process. We must plan ahead and take precautionary measures.

The above-mentioned four certificate problems are judged by browsers when accessing websites with https, which is why some users will see the security warning messages from browser, but App developers are often not professionals who understand PKI technology, and they do not know that there are so many things about https connections, they may think that it is ok if the SSL certificate is deployed on the server, then the App programming enable the https connection. In fact, this is far from enough. When the app uses the https protocol to shake hands with the server, it must check that the first four case above are all correct before exchanging data with the server normally. Only in this way can we truly ensure that https encryption can protect confidential data. As for the fifth issue, many PC browsers already support the SM2 algorithm, the commonly used mobile apps must also support it as soon as possible.

The author calls on all app developers to check whether the app you developed has done the above five checks immediately after reading this article. If not, immediately improve and release an updated version. If a friend who sees this article is not an app developer but has an app in his own organization or a friend's organization, please inform the app developer to check the app program immediately and fix these security problems as soon as possible. I also call on friends who do App security testing to

pay special attention to checking whether the App has these problems when testing the user's App, to help users fix it in time. In this way, the security level of mobile apps in China can be greatly improved, the data of app users can be made more secure, and the Internet can be made more secure.

Richard Wang

**Sept. 22, 2022**
**In Shenzhen, China**