关于 APP 预埋检查策略优化

背景

在 APP 海量时代，很多 APP 都进行过证书预埋检查功能设置。当预埋的 APP 更新服务器端证书时，发现部分客户出现证书更新之后，APP 无法正常通信等情况出现。经过技术排查发现由于 APP 中进行了证书预埋检查，并预埋了服务器证书（公钥）文件等信息，导致证书更新后新证书无法通过 APP 预埋检查验证，提示网络中止连接。

如果您的 APP 做了以下的预埋，都会存在隐患：

1.在 APP 中预埋了服务器证书（公钥）做证书检查

2.在 APP 中预埋了服务器证书的指纹值做证书检查

3.在 APP 中预埋了中级证书文件或者根证书文件做证书检查

4.在 APP 中预埋了中级证书指纹值或者根证书指纹值做证书检查

证书预埋检查隐患

1. 服务器证书（公钥）存在有效期限制，目前最高 3 年有效期，未来 CA/B 组织会逐渐缩短证书最长生命周期。如果将服务器证书进行 APP 预埋证书检查，当服务器证书因为生命周期终止导致证书更新时，经过预埋的 APP 会出现无法正常通信等情况。

2. 根证书和中级证书也会存在失效、策略变更或者过期，如果将根证书或者中级证书进行 APP 预埋，一旦根证书和中级证书发生了更新，经过预埋的 APP 会出现无法正常通信等情况。

虽然在大多数情况下，检查机制能够防御中间人攻击。因为当黑客窃听通信时，他提供的拦截证书多为自签名证书，TrustManager 不能识别这个证书，于是拒绝 HTTPS 连接。但是，一旦预埋的这张证书由于证书生命周期的终止，触发服务器端进行证书更新替换机制时，APP 应用端将无法验证请求证书的有效性，从而导致业务中断。

天威诚信建议

为了 APP 业务灵活性和可续性，天威建议使用系统默认的系统信任库来做证书认证，并且开启域名强验证来增强 APP 安全性。

对服务器证书域名进行强校验（仅供开发人员参考）：

```
httpsURLConnection.setHostnameVerifier(SSLSocketFactory.STRICT_HOSTNAME_VERIFIER);
```

实现 HostnameVerifier 的 verify()方法：

```
final HostnameVerifier hostnameVerifier = new HostnameVerifier() {
    @Override
    public boolean verify(String s, SSLSession sslSession) {
        HostnameVerifier hv = HttpsURLConnection.getDefaultHostnameVerifier();
        Boolean result = hv.verify("*.washington.edu", sslSession);
        return result;
    }
};
httpsURLConnection.setHostnameVerifier(hostnameVerifier);
```

由于 SSL 证书本身存在生命周期基本属性，加上中级 CA 和根证书存在可变更的因素，一旦应用中添加了证书检查功能之后，在后期业务发展过程中势必会带来不可预估的影响。在此，天威诚信技术团队建议您，在安全策略允许的情况下**取消证书检查功能**，改为使用系统自带信任库方式验证，并配合使用其他比如域名强验证,使用 Proguard 混淆代码,使用编译库等手段来确保您的 APP 安全。

如果您有任何相关疑问，可以通过以下方式联系我们技术团队——

邮箱：support@itrus.cn　　技术热线：4006-365-010

相关知识

TLS/SSL 证书链

保证通信安全至少要使用 HTTPS 协议，也就是说使用安全传输层协议（TLS）或是它的前身安全套接层协议（SSL）加密的通信。

1.SSL 证书链结构关系

公网可信 SSL 证书（至少）包括三个证书：

根证书：这是由证书认证机构（CA）颁发的，也就是一个可以确保整个通信时安全的值得

信任的组织。

中级证书：一个根证书下有多个中级证书。它们建立服务器证书和根证书的信任桥梁，是连接服务器证书和根证书的证书链，由根证书签名的证书。

服务器证书：服务器证书是绑定最终请求域名的证书，为最终的加解密证书文件。

2.证书检查

（1）Certificate Pinning

其实 Certificate Pinning 是 OkHttp 实现的一个类似于 HPKP 的技术，目的是为了使客户端可以有主动的信任 CA 的权利，它的工作原理就是使用预先设置的证书指纹和服务器传过来的证书链中的证书指纹进行匹配，只要有任何一对指纹匹配成功，则认为是一次合法的连接，否则禁止本次链接。

（2）预埋证书

把数字证书以文件或者字符串的形式写在本地，在 SSL 握手的时候用本地预埋的证书和服务器传过来的证书进行匹配，如果匹配不成功则禁止本次通信。

（3）Certificate Pinning 和预埋证书做法的比较（以 OkHttp 为例）

预埋证书的写法

首先定义一个类实现类实现 X509TrustManager 接口，实现对客户端证书链的校验方法和服务端证书链的校验方法，不写的话表示不做任何校验，默认信任所有证书链中的证书。

```
public class MyX509TrustManager implements X509TrustManager {
    @Override
    public void checkClientTrusted(X509Certificate[] chain, String authType) throws CertificateException {
        // 校验客户端证书
    }

    @Override
    public void checkServerTrusted(X509Certificate[] chain, String authType) throws CertificateException {
        // 校验服务端证书
    }

    @Override
    public X509Certificate[] getAcceptedIssuers() {
        return new X509Certificate[0];
    }
}
```

由于证书存在生命周期和可变性，证书更新和证书链变更都会导致这种预埋失效，从而带来业务中断风险。

Certificate Pinning 的写法

范例代码：

```
public void check(String hostname, List<Certificate> peerCertificates)
      throws SSLPeerUnverifiedException {
    List<Pin> pins = findMatchingPins(hostname);
    if (pins.isEmpty()) return;

    if (certificateChainCleaner != null) {
      peerCertificates = certificateChainCleaner.clean(peerCertificates, hostname);
    }

    for (int c = 0, certsSize = peerCertificates.size(); c < certsSize; c++) {
      X509Certificate x509Certificate = (X509Certificate) peerCertificates.get(c);

      // Lazily compute the hashes for each certificate.
      ByteString sha1 = null;
      ByteString sha256 = null;

      for (int p = 0, pinsSize = pins.size(); p < pinsSize; p++) {
        Pin pin = pins.get(p);
        if (pin.hashAlgorithm.equals("sha256/")) {
          if (sha256 == null) sha256 = sha256(x509Certificate);
          if (pin.hash.equals(sha256)) return; // Success!
        } else if (pin.hashAlgorithm.equals("sha1/")) {
          if (sha1 == null) sha1 = sha1(x509Certificate);
          if (pin.hash.equals(sha1)) return; // Success!
        } else {
          throw new AssertionError();
        }
      }
    }
```

pinner 就是证书指纹，sha256 表示的是哈希值得一种算法，pinner 也可以是 sha1/* ，具体 pinner 要根据购买证书所支持的配置，证书指纹可以在证书详细信息中点击查看。在 SSL 握手的时候，会检查配置中的指纹和服务端传过来的证书证的指纹是否相匹配，只要有一个指纹匹配，则进行下一步，否则直接抛出异常，禁止本次连接。

证书在使用过程中会存在不可避免的证书续期和替换更新，每次操作都会导致证书的指纹发现不可逆的变化，一旦使用了 Certificate Pinning 方式，都会在证书更新之后带来业务中断等情况。