Implementasi *Certificate Pinning* pada Aplikasi berbasis Android

Muhammad Abdullah Munir
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
munirabdullahm@gmail.com

Abstrak—Seiring dengan berkembangnya teknologi, semakin banyak pula pengguna teknologi tersebut. Salah satunya adalah meningkatnya penggunaan *smartphone*. Telah banyak aplikasi-aplikasi yang sebelumnya berbasiskan website, membuat *cross-platform* ke perangkat *mobile*. Salah satunya adalah perangkat *mobile* berbasiskan sistem operasi *Android*. Aplikasi pada perangkat *mobile* pada umumnya tetap melakukan komunikasi dengan *website* dengan menggunakan protokol yang telah disediakan. Dapatkah pengguna aplikasi mengetahui bahwa dirinya tidak sedang disadap. Salah satu cara untuk meningkatkan keamanan dari sisi aplikasi ialah dengan menerapkan *certificate pinning*.

Kata kunci—*Public Key Cryptography; Man-in-the-middle; certificate*

I. Latar Belakang

Teknologi pada saat ini telah berkembang dengan pesat. Hampir seluruh hal dapat dilakukan melalui *smartphone*. Seperti berbelanja, transfer dana, melihat konten hiburan, berkomunikasi, dsb. Seakan-akan kita tidak memerlukan alat bantuan lain seperti komputer.

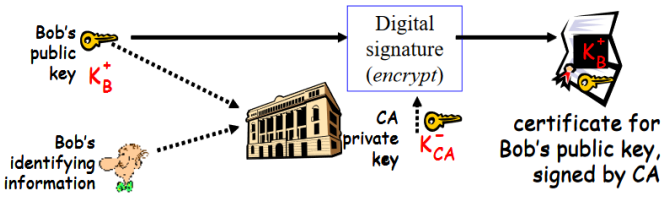
Akan tetapi dibalik kemudahan tersebut, ternyata masih banyak yang perlu diperhatikan dari sisi keamanan. Salah satunya ialah proses komunikasi dari *client* ke *server*. Proses komunikasi ini sangat penting dijaga kerahasiaannya, terlebih lagi apabila perangkat *mobile* sering digunakan untuk transaksi keuangan. Cara mengamankan yang wajib diimplementasi pada era ini adalah komunikasi dengan melalui protokol TLS, atau yang sering digunakan adalah HTTPS (HTTPSecure).

Apabila pada *web browser* kita dapat melihat transparansi dari sertifikat yang digunakan pada *server* dengan melihat *option* di dekat *navigation bar*. Namun bagaimana jika aplikasi yang kita gunakan berinteraksi dengan *server* secara internal, seperti *invoke web api* ketika suatu *button* di-klik. Masih dapatkah pengguna melihat sertifikat server?. Oleh karena itu, pada pengembangan aplikasi pada perangkat *mobile* diperlukan pengamanan yang lebih. Salah satu caranya ialah dengan menerapkan *certificate pinning* yang dapat diimplementasi dengan mudah yang nantinya akan dibahas lebih lanjut.

II. Dasar Teori

A. *Digital Certificate*

Sertifikat digital adalah suatu dokumen digital yang berisi kunci publik berserta informasi pemiliknya. Dokumen ini diterbitkan oleh *Certification Authority* (CA) yang berupa institusi yang terpercaya. Pada sertifikat terdapat pula tanda tangan dari CA untuk memastikan keasliannya.



Gambar 1. Proses pembuatan sertifikat digital (Rackenee Rhule et al, *Digital Certificates*)

Proses pembuatan sertifikat digital mirip dengan pembuatan SIM atau paspor. Pada awalnya pemohon mengirimkan kunci publik beserta informasi identitas kepada CA. Kemudian CA akan membuat sertifikat digital dan menandatangani dengan kunci privat milik CA. Tanda tangan dari CA berupa hasil enkripsi nilai hash dari kunci publik dan informasi identitas milik pemohon.

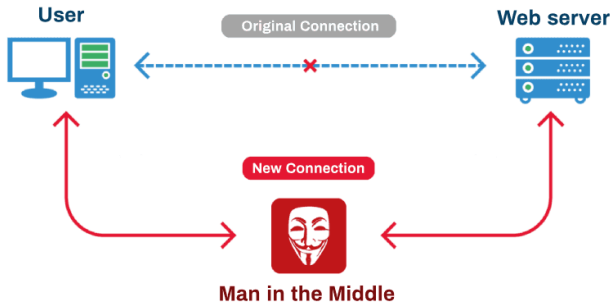
Untuk melakukan verifikasi sertifikat digital diperlukan pengecekan secara beruntun. Yang pertama adalah memastikan bahwa CA yang memberikan tanda tangan dapat dipercaya. Tahap selanjutnya setelah melakukan pengecekan pada CA adalah dengan memastikan keaslian sertifikat, dengan cara mengecek tanda tangan. Setelah itu pastikan bahwa pemilik dari sertifikat tersebut asli. Yang terakhir lakukan *nonce challenge*, yaitu dengan mengirimkan sebuah *nonce* dan minta *server* untuk melakukan enkripsi pada *nonce* tersebut. Setelah didapatkan *nonce* yang terenkripsi lakukan validasi dengan kunci publik yang terdapat pada sertifikat. Tentunya server akan memiliki kunci privat yang sesuai dengan kunci publik pada sertifikat.

Pada sertifikat digital terdapat pula masa berlakunya. Apabila telah lewat maka sertifikat tersebut akan kadaluarsa. Hal ini ditujukan agar pemiliknya mengubah kunci miliknya

secara berkala. Serta apabila kunci privat diketahui oleh orang lain sebelum sertifikat kadaluarsa, CA dapat melakukan *revoke* terhadap sertifikat tersebut. CA secara berkala akan mengeluarkan *Certificate Revocation List* yang berisi sertifikat digital yang ditarik.

B. *Man in the Middle Attack*

Pada serangan ini terdapat pihak ketiga yang melakukan penyamaran. *Attacker* akan mengaku sebagai *server* pada suatu *client*. Kemudian akan berkomunikasi dengan *server* seakan-akan merupakan *client* yang sebelumnya. Pada hal ini semua isi komunikasi bisa didapatkan oleh penyerang.



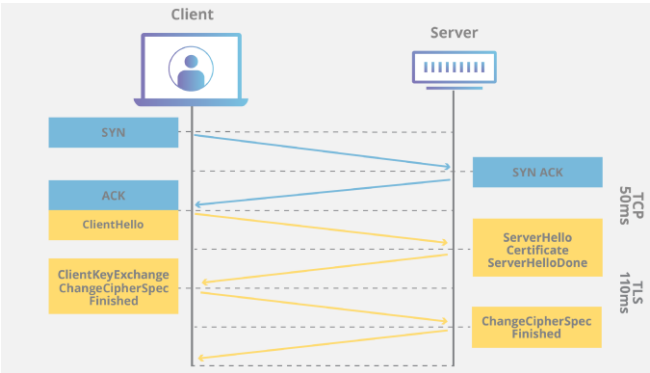
Gambar 2. *Man in the Middle Attack*

Terdapat banyak cara untuk bisa melakukan serangan ini, beberapa diantaranya: *IP Spoofing*, *DNS Spoofing*, dan *SSL Hijacking*

C. *HTTP over TLS*

HTTP adalah protokol komunikasi internet yang paling umum digunakan, termasuk pada aplikasi berbasis android. Terdapat versi yang lebih aman pada protokol ini, yaitu HTTPS. Pada protokol HTTPS semua data yang dikomunikasikan akan dienkripsi terlebih dahulu untuk menghindari adanya penyadapan.

Untuk mengimplementasi HTTPS pada *server* diperlukan sertifikat digital untuk melakukan enkripsi saat komunikasi. Sebelum dilakukan komunikasi akan terdapat *handshake* untuk inisialisasi koneksi. Pada proses *handshake* akan dilakukan autentikasi terhadap keaslian *server*. Hal ini dilakukan dengan validasi terhadap sertifikat yang digunakan. Setelah itu dilakukan pertukaran kunci beserta algoritma yang nantinya akan digunakan untuk berkomunikasi.



Gambar 3. Proses *TLS Handshake*

Dengan menggunakan protokol HTTPS walaupun terjadi penyadapan, pihak penyadap tidak akan mengetahui isi dari percakapan. Konten yang dikirim antar *client-server* telah dienkripsi. Penyadap juga tidak dapat melakukan pengubahan isi pesan dengan adanya *Message Authentication Code*.

III. Implementasi dan Percobaan

Pada aplikasi android, umumnya komunikasi ke *server* dengan menggunakan protokol HTTPS. Validasi *default* yang dilakukan adalah dengan melakukan pengecekan terhadap keaslian dari sertifikat digital. Selama sertifikat asli maka komunikasi akan dilakukan. Namun, pada proses pengembangan aplikasi tersebut *developer* tahu bahwa aplikasi tersebut hanya akan mengakses ke *website* apa saja. Sehingga dengan memanfaatkan kondisi ini, dapat ditambahkan validasi lebih dalam. Yaitu dengan melakukan pencocokan sertifikat digital milik *server*. Spesifikasi untuk proses *pinning* ini telah diatur dalam RFC 7469. Untuk implementasi pada aplikasi *android* terdapat banyak cara, yaitu:

1. Android versi > 7

   Pada android telah terdapat *certificate pinning*. Dengan cara melakukan konfigurasi berkas *XML* dan menambahkan *domain website* serta *pin* berupa hash dari sertifikat.

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config>
        <domain includeSubdomains="true">example.com</domain>
        <pin-set expiration="2018-01-01">
            <pin digest="SHA-
256">7HIpactkIAq2Y49orFOOQKurWxmmSFZhBCoQYcRhJ3Y=</pin>
            <!-- backup pin -->
            <pin digest="SHA-
256">fwza0LRMXouZHRC8Ei+4PyuldPDcf3UKgO/04cDM1oE=</pin>
        </pin-set>
    </domain-config>
</network-security-config>
```

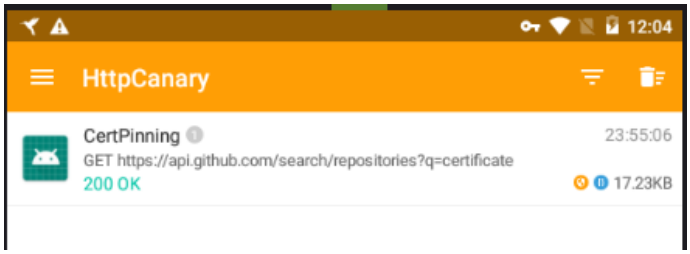
2. Library Okhttp

   Pada *library Okhttp* dapat dilakukan *certificate pinning* pada saat melakukan *build OkHttpClient.*

```
CertificatePinner certificatePinner = new
CertificatePinner.Builder()
    .add("example.com",
"7HIpactkIAq2Y49orFOOQKurWxmmSFZhBCoQYcRhJ3Y=")
    .build();

okHttpClient = new OkHttpClient.Builder()
    .certificatePinner(certificatePinner)
    .build();
```

Setelah dilakukan implementasi *Certificate Pinning* dilakukan percobaan *network capture* dengan menggunakan *HttpCanary*. Pengujian dilakukan dengan melakukan *request HTTPS* ke API github.



Gambar 4. Hasil *Intercept* pada *HttpCanary*

Pada percobaan tanpa *certificate pinning* komunikasi masih dapat disadap oleh pihak ketiga. Percobaan selanjutnya dilakukan dengan *certificate pinning*.

```
E/error: fail cert check
    javax.net.ssl.SSLPeerUnverifiedException:Certificate
pinning failure!
    Peer certificate chain:
      sha256/2u7WL3HpMMTz6/OPTCgkqw/Kdas0utrJp6TakByKVw4=:
OU=HttpCanary,O=HttpCanary,CN=api.github.com
      sha256/lYgYmvj0e7JxXab7wxaN/I+Pa9unvqiDpNMLK7VR4hY=:
OU=HttpCanary,O=HttpCanary,CN=HttpCanary Root CA
    Pinned certificates for api.github.com:
      sha256/ORH27mxcLwxnNpR7e0i6pdDPWLXdpeWgr5bEfFVbxW8=
    at
okhttp3.CertificatePinner.check$okhttp(CertificatePinner.kt:
200)
    at
okhttp3.internal.connection.RealConnection.connectTls(RealCo
nnection.kt:398)
    at
okhttp3.internal.connection.RealConnection.establishProtocol
(RealConnection.kt:325)
     at
okhttp3.internal.connection.RealConnection.connect(RealConne
ction.kt:197)
```

Dengan adanya *certificate pinning*, ketika *client* mencoba melakukan *connect* dan diketahui bahwa hash dari sertifikat berbeda maka aplikasi melakukan *throw error*. Sehingga komunikasi tidak akan dilakukan.

IV. Analisis Keamanan

Implementasi dari *certificate pinning* sangatlah sederhana, namun memberikan keamanan yang lebih kepada pengguna aplikasi. Akan tetapi, validasi ini masih dapat di *bypass*. Tingkat kesulitan *bypass* bergantung pada implementasi *pinning*. Beberapa caranya sebagai berikut:

1. Mengubah konfigurasi *XML*

   Apabila *certificate pinning* diatur hanya dengan menggunakan default dari android, maka *bypass* dapat dilakukan dengan mengubah *hash* pada *XML* sesuai dengan *hash* sertifikat milik *HttpCanary*.

2. Melakukan *hook* pada fungsi validasi

   Dengan melakukan *intercept* pada fungsi yang melakukan validasi dan mengembalikan nilai *true*. Hal ini dapat digunakan untuk melewati pengecekan.

Untuk meningkatkan tingkat kesulitan dalam melakukan *bypass*. Dapat dibuat *library* atau fungsi yang melakukan pengecekan secara *custom*. Dengan begitu, penyerang perlu untuk menganalisa kode terlebih dahulu sebelum dapat melakukan *patching* untuk *bypass*.

V. Kesimpulan

Certificate pinning merupakan salah satu lapisan untuk meningkatkan keamanan dalam komunikasi. Dalam hal ini yaitu mencegah terjadinya *Man-in-the-Middle*. Metode ini sangat cocok untuk digunakan karena implementasinya yang mudah. Namun, akan lebih baik lagi apabila dapat dibuat sendiri kode validasinya untuk mempersulit *bypass*.

Referensi

[1] Munir, Rinaldi. 2018. *Secure Socket Layer* (SSL) . Program Studi Teknik Informatika Institut Teknologi Bandung. Diakses pada 2 Mei 2020

[2] Munir, Rinaldi. 2018. Sertifikat Digital dan *Public Key Infrastructure* (PKI) . Program Studi Teknik Informatika Institut Teknologi Bandung. Diakses pada 2 Mei 2020

[3] Munir, Rinaldi. 2018. SSL. Program Studi Teknik Informatika Institut Teknologi Bandung. Diakses pada 2 Mei 2020

[4] Podgornyy, Sergey. 2019. Prevent MITM attacks on your website [Online]. Available at: https://blog.larapulse.com/security/prevent-mitm-attacks [Diakses 2 Mei 2020].

[5] Cloudflare. (n.d.). What Happens in a TLS Handshake? [Online]. Available at: https://www.cloudflare.com/learning/ssl/what-happens-in-a-tls-handshake/ [Diakses 2 Mei 2020].

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Jombang, 8 Mei 2020

Muhammad Abdullah Munir
13516074