FIELD NOTE / ANDROID

Threat Modeling Android App

Petruknisme · 7 Apr, 2018 · 6 min to read

Aplikasi yang baik sudah pasti datang dari perancangan sistem yang baik pula selama masa development sampai aplikasi tersebut terpublikasi, hal tersebut dilakukan agar mengoptimalkan resource serta efisiensi dari sebuah aplikasi. Biasanya hal tersebut dilakukan ketika memasuki fase Design sebuah aplikasi. Apa yang akan saya bahas kali ini masih berhubungan dengan fase yang dibicarakan, hanya saja ini mencakup bahasan untuk keamanan yang bisa disebut dengan Threat Modeling.

Bagi yang sudah berkecimpung dan sering melakukan *Mobile App Penetration Testing* ataupun *Reverse Engineering*, semestinya sudah memahami hal ini. Tapi, tulisan yang saya buat kali ini ditujukan untuk siapapun, terutama bagi pengembang aplikasi yang belum mempunyai pandangan terhadap threat modeling apa yang seharusnya dipikirkan. Oke, jadi apa itu Threat Modeling?

> *"**Threat Modeling** adalah suatu konsep pengidentifikasian sebuah ancaman(threat) terhadap aplikasi dengan cara melakukan pendefinisian aset, nilai yang diberikan, dan sudut pandang dari threat actor yang mungkin tertarik untuk menyerang aset yang anda sudah buat. Dan sudah seharusnya Threat Modeling ini dilakukan pada fase perancangan(Design) sebuah aplikasi."*

Dari sudut pandang Keamanan dan Privasi, aplikasi diharuskan:

- Mencegah pengguna yang tidak ter-autentikasi dalam menggunakan fitur seperti API dan hal-hal lain yang memerlukan autentikasi pada dasarnya.
- Mencegah pengaksesan terhadap informasi ataupun pengendalian akun pengguna.
- Mencegah agar pihak ketiga tidak mampu untuk mendapatkan rincian otentikasi ataupun identifikasi
- Mengurangi peluang terhadap pengaksesan informasi sensitif oleh orang yang tidak diinginkan

Threat sendiri terbagi menjadi dua bagian, yaitu:

- Threat pada Client Side
- Threat pada Server Side

Kita akan bahas satu per satu untuk mengetahui apa saja yang mencakup kedua bagian di atas.

Threat pada Client Side

- Data dalam Aplikasi

Dengan adanya penyimpanan data pada sisi klien, hal itu memudahkan untuk menyimpan data-data yang diperlukan oleh aplikasi. Banyak aplikasi yang menyimpan data sensitif pada perangkat tanpa

Licensed under CC BY-SA 4.0

adanya enkripsi. Nah, hal ini yang menjadi masalah besar, karena bisa saja data tersebut bersifat confidental, sensitif ataupun pribadi. Data yang tersimpan di perangkat bisa dieksploitasi dengan berbagai banyak cara. Misal, seorang attacker yang mempunyai akses terhadap perangkat secara langsung bisa mendapatkan data tersebut tanpa perlu melakukan exploitasi.

Atau bisa juga dengan menggunakan aplikasi berbahaya seperti backdoor dan lain sebagainya yang memungkinkan aplikasi tersebut mendapatkan data ketika perangkat dalam keadaan root/jailbreak. Dalam hal ini kita penting untuk memastikan bahwa tidak ada data sensitif yang disimpan di perangkat seperti username, password, token, nomor rekening dan hal-hal lain. Bila memang diharuskan untuk menyimpan data pada perangkat seperti berkas *shared_preference* file, disarankan melakukan enkripsi terhadap file yang memang mengandung data sensitif.

- Data aplikasi ketika bertukar

Bertukar yang dimaksud di sini adalah ketika sebuah aplikasi mengirim dan menerima data ke server(backend). Bayangkan bila data yang dikirimkan ternyata tidak melalui protokol https, sudah pasti data-data sensitif yang dikirimkan ke server akan dengan mudah dilakukan *eavesdrop*. Atau sekalipun menggunakan https masih dapat dilakukan MiTM dengan burp proxy, mitmproxy, bettercap dan beberapa proxy dan mitm lain. Untuk kasus ini, disarankan untuk menerapkan certificate pinning pada aplikasi ataupun implementasi HSTS agar tidak bisa dilakukan MiTM.

- Kerentanan di kode

Ketika sebuah aplikasi dikembangkan tanpa adanya peninjauan terhadap keamanan bisa saja rentan terhadap banyak macam serangan. Kecerobohan ataupun kesalahan yang dilakukan saat coding bisa menyebabkan kerentanan serius yang bisa jadi berdampak pada data user maupun aplikasi. Misalnya aplikasi memperbolehkan content providers untuk diexport, activity yang juga bisa diexport tanpa harus melalui activity utama atau autentikasi. Atau mungkin bisa jadi aplikasi berbahaya bisa membaca konten dari aplikasi lain dikarenakan kesalahan saat coding. Atau bisa jadi dengan melakukan proses dekompilasi terhadap aplikasi yang di mana mengandung *hardcoded credential* dalam source code.

- Kebocoran data di Aplikasi

Tidak bisa dipungkiri bahwa kebocoran informasi bisa berasal dari aplikasi itu sendiri yang memungkinkan aplikasi lain untuk leak data sensitif. Misal kode yang dipakai oleh pengembang masih melakukan *logging* selama masa development dan hal itu masih berjalan setelah aplikasi terpublikasi. Seorang pengembang harus memastikan bahwa tidak ada informasi yang bisa dimanfaatkan oleh attacker nantinya. Atau bila seorang melakukan copy data sensitif berupa credit card, password dan yang lain melalui clipboard maka aplikasi lain pun bisa membaca data tersebut tanpa sepengetahuan aplikasi utama.

- Permasalahan platform tertentu

Disaat kita melakukan threat modeling untuk aplikasi mobile, kita wajib menyesuaikan dengan platform yang akan kita bangun atau gunakan. Misal kalau aplikasi yang akan kita bangun adalah native android app. Native android app untuk android biasanya lebih mudah dalam melakukan reverse engineering dengan proses dekompilasi apk dan kemudian java source code akan dengan mudah dilihat dan

dibaca. Hal tersebut memungkinkan seorang attacker untuk memahami alur dari program atau bahkan mendapatkan data sensitif/kredensial yang disimpan pada source code. Dan memungkinkan juga untuk dilakukan modifikasi source code serta direcompile kembali agar aplikasi tersebut menjadi valid seperti aplikasi sebelumnya lalu disebarkan seakan-akan aplikasi yang asli.

Threat pada Server Side

Biasanya server side/backend yang digunakan untuk komunikasi oleh klien menggunakan web service seperti REST API ataupun jenis web service lain. Yang di mana web service ini masih bisa kita katakan sebagai web app dan mempunyai kesamaan dalam kerentanan secara umum. Ini berarti bahwa dalam melakukan pengembangan API juga wajib memikirkan beberapa kerentanan umum, berikut akan saya jelaskan beberapa gambaran kerentanan umum yang terjadi.

- Otentikasi/otorisasi

Hal yang paling sering ditemukan dalam melakukan pengujian terhadap aplikasi di mana terdapat endpoint yang seharusnya membutuhkan otentikasi atau otorisasi tapi bisa diakses begitu saja. Atau bisa juga berupa kerentanan IDOR[1] yang melakukan akses terhadap data user lain dengan otentikasi/otorisasi sah untuk diri sendiri.

- Session management

Biasanya, aplikasi mobile ini menggunakan token atau semacam nya untuk setiap user yang sudah login dan akan menghapus session tersebut bila user telah logout. Kerentanan yang terjadi bisa berupa session fixation[2] yang memanfaatkan insecure session handling[3] pada server.

- Input validasi

Ini merupakan masalah umum yang sering kita temui, banyak sumber kerentanan berasal dari validasi input. Misal input yang tidak tervalidasi dengan baik pada server/client bisa menimbulkan kemungkinan SQL Injection, Command Injection, dan lain-lain bila tidak diterapkan validasi.

- Penanganan error

Pesan error dari aplikasi bisa menjadi pesan penting bagi seorang attacker yang sedang melakukan recon terhadap service atau fitur apa saja yang diimplementasikan. Bila error tidak ditangani dengan baik, seperti error yang terjadi karena ada permasalahan database, maka hal tersebut bisa dimanfaatkan untuk melakukan penyerangan nantinya. Percayalah, ini bukan hanya soal pesan error saja.

- Kelemahan Kriptografi

Setiap platform memiliki dukungan untuk implementasi yang tepat dalam mengamankan data menggunakan kriptografi. Hal yang biasanya terjadi adalah bagaimana aplikasi menangani manajemen kunci untuk membaca dan menulis ulang. Dipastikan bahwa pengembang tidak menyimpan manajemen kunci ditempat yang tidak aman. Untuk android terbaru sudah mendukung manajemen

Licensed under CC BY-SA 4.0

kunci secara acak[4], jadi setiap aplikasi tersebut berjalan maka android keychain akan memberikan kunci untuk membaca dan menulis ulang source code/pesan.

- Penyerangan pada Database

Hal ini terjadi ketika seorang pengembang membiarkan pengaksesan terhadap database server tanpa otentikasi ataupun menggunakan kredensial yang lemah. Seperti MongoDB yang secara default tidak membutuhkan kredensial ataupun Postgre.

Sekian tulisan kali ini, semoga penjelasan yang sederhana ini mudah untuk dipahami. Bila ada hal-hal yang tidak dipahami bisa cantumkan pada kolom komentar atau bisa hubungi saya langsung. Akhir kata, semoga tulisan kali ini bermanfaat.

Referensi

1. https://www.owasp.org/index.php/Testing_for_Insecure_Direct_Object_References_(OTG-AUTHZ-004) (https://www.owasp.org/index.php/Testing_for_Insecure_Direct_Object_References_(OTG-AUTHZ-004))
2. https://www.owasp.org/index.php/Session_fixation (https://www.owasp.org/index.php/Session_fixation)
3. https://www.owasp.org/index.php/Session_Management_Cheat_Sheet (https://www.owasp.org/index.php/Session_Management_Cheat_Sheet)
4. https://medium.com/@ericfu/securely-storing-secrets-in-an-android-application-501f030ae5a3 (https://medium.com/@ericfu/securely-storing-secrets-in-an-android-application-501f030ae5a3)

Android # Security # Tutorial

Licensed under CC BY-SA 4.0