Sistem keamanan dan deteksi serangan pada Java Card

Makalah ini disusun sebagai tugas akhir

Pada Mata Kuliah

EL6115 – Operasi Keamanan dan Insiden Respon

Oleh

Yundi Supriadani

23214334

Dosen:

Dr.Ir. Budi Rahardjo

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2016

Abstrak

Sistem keamanan dan deteksi serangan pada Java Card

oleh

Yundi supriadani – 23214334

Sekolah Teknik Elektro dan Informatika ITB

Smart card merupakan divais kartu yang memiliki kecerdasan dengan ditanamkan prosesor didalamnya. Agar pmrograman Smartcard menjadi fleksibel dan mudah,digunakan bahasa java sebagai bahasa yang bersifat *object oriented programming*, namun sebuah pogram berbahasa java tidak dapat langsung berjalan sebuah *embedded system.* Maka dari itu agar program java dengan *library* java card dapat berjalan dalam sebuah memori dan prosesor perlu dikembangkan *java virtual machine* yang berjalan diatas sistem operasi. Program java yang menghasilkan *bytecode* untuk dieksekusi masih memiliki kelemahan dalam sistem keamanannya. Ketika beroperasi, sebuah javacard dapat mengalami serangan, antara lain *physical attack, observation attack* dan *fault attack.* Pada makalah ini dipaparkan mengenai teknologi java card, java card virtual machine, jenis-jenis serangan pada java card dan deteksi serangan pada java card .

Kata Kunci: *attack of java card, attack detection, Java Card virtual machine* .

Daftar Isi

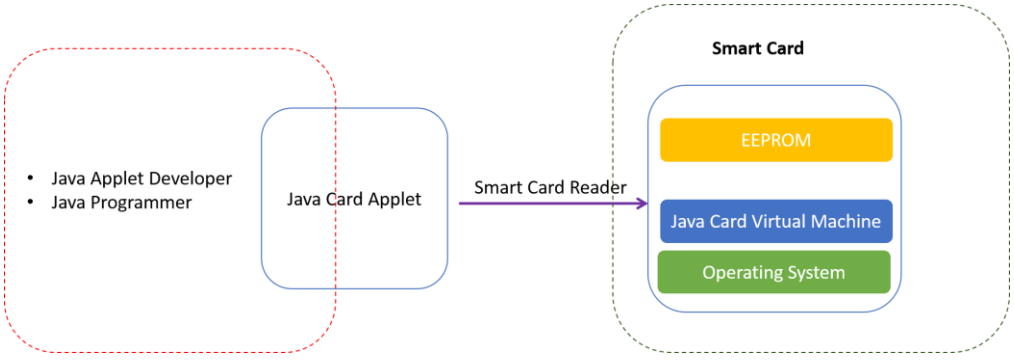
Daftar Gambar

Daftar Tabel

1. Pendahuluan

Smart card merupakan divais kartu yang memiliki kecerdasan dengan ditanamkan prosesor didalamnya. Dalam penggunaannya, smart card diprogram menggunakan bahasa pemrograman tingkat tengah, antara lain bahasa C. Perkembangan pemrograman smart card membutuhkan bahasa yang fleksibel dan bersifat *object oriented programming*. Agar pemrograman Smartcard menjadi fleksibel dan mudah,digunakan bahasa java sebagai bahasa yang bersifat *object oriented programming*. Library java yang disediakan oleh oracle adalah Java Card yang terbaru adalah Java Card 3.0.

Teknologi java card memungkinkan program ditulis dalam bahasa java dan berjalan dalam smart card atau divais kecil lainnya. Pengembang dapat membangun dan menguji program menggunakan perangkat lunak standar dan melakukan pengujian hingga menanamkan pada teknologi ini pada divais smart card. Teknologi Java Card digunakan secara luas dalam berbagai aplikasi, antara lain perbankan, aplikasi identitas atau GSM [1], berdasarkan aplikasi tersebut, java card memiliki peranan penting dalam smart card. Agar sebuah aplikasi java dapat berjalan dalam sebuah smart card, dibutuhkan java card virtual machine. Java Card virtual machine merupakan sistem yang berjalan diatas sistem operasi yang bertugas untuk menginterpretasikan byte code applet java card. Bytecode tersebut dihasilkan dari kompilasi applet java card yang disimpan pada memori EEPROM smart card. Dalam hal ini adalah byte code yang disimpan dalam EEPROM kemudian dibaca oleh *java card virtual machine*



Gambar 1.1 Java Applet dan Java Card Virtual Machine

Java card sebagai library yang berjalan dalam java card virtual machine dapat mengalami serangan, baik secara fisik ataupun dalam bentuk perangkat lunak, antara lain *combining fault* dan *logical attacks*[1]. Kini, serangan dengan ancaman paling kuat adalah *fault attack. Fault attack* bekerja dengan cara menghasilkan kegagalan ketika eksekusi kode dan kemudian menghasilkan kesalahan komputasi. Kesalahan tersebut dimanfaatkan untuk meperoleh informasi yang tersimpan di dalam smart card. Adapun salah satu metode untuk menjaga keamanan perangkat lunak melalui verifikasi dan validasi pada layer komunikasi yang ditanamkan dalam smart card[2].

Dalam makalah ini dipaparkan jenis-jenis serangan pada teknologi java card dan bagaimana menanggulangi juga mendeteksi serangan pada java card.

2. Tinjauan Pustaka

2.1 Java Sandbox Model

Java sandbox model merupakan lingkungan yang menerima segala macam kode dari berbagai sumber, namun jika kode berasal dari sumber yang tidak terpercaya, Sandbox akan menolak kode sehingga tidak menghasilkan bahaya dalam sistem.[3].

2.2 *Java virtual machine*

Java virtual machine merupakan sistem yang bertugas menerjemahkan program java (applets) menjadi intruksi-intruksi yang harus dieksekusi oleh sistem. *Java virtual machine* sendiri seperti mesin di dalam mesin (fisik). Dalam hal ini, program java berjalan di dalam *java virtual machine* dan secara fisik, sistem ini hanya ada di dalam memori dari sebuah sistem, baik komputer ataupun embedded system.

Karena setiap program java berjalan dalam *java virtual machine, java virtual machine* ini memiliki peranan yang sangat penting sehingga membutuhkan sistem keamanan di dalamnya. Adapun berbagai jenis fitur keamanan dalam *java virtual machine* antara lain:

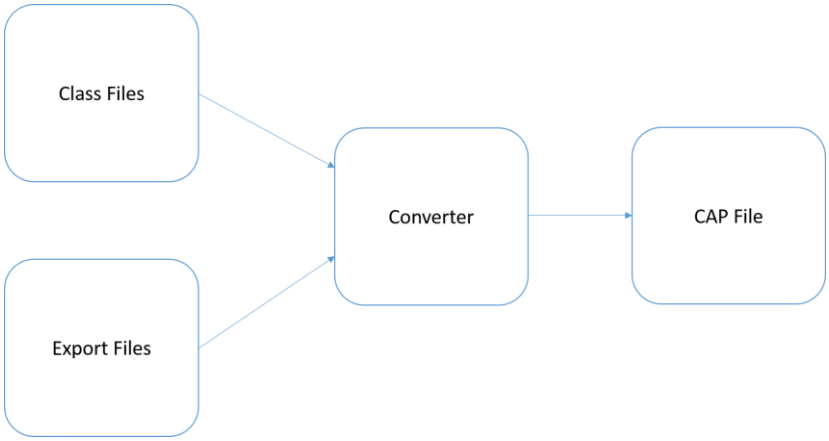
1. Byte Code Verification
2. Dynamic Class Loading
3. Pengecekan keamanan Runtime
4. Kontrol melalui akses ke sumber sistem krusial
5. Akses Kontroler dan permissions

2.3 *Java card virtual machine* dan Sistem Java Card

Java card virtual machine merupakan subset dari *java virtual machine*. Fungsi *java card virtual machine* sendiri adalah untuk menjalankan teknologi java card dalam sebuah smart card. Untuk menghasilkan sebuah sistem java card, dibutuhkan beberapa komponen antara lain:

- *Java card virtual machine*
- *Java Card Converter*
- *Terminal installation tool*
- *Installation Program.*

Sebuah program java card yang telah ditulis dan di-*compile* dalam bentuk class diterjemahkan menjadi byte code melalui java card converter sehingga diperoleh file dengan ekstensi CAP. Setiap program java card yang dibuat akan menghasilkan file dalam bentuk CAP (Converted Applet) untuk disimpan ke dalam memori dari divais smart card. Penyimpanan file CAP tersebut digandakan pada *card* terminal menggunakan perangkat *card reader*. Dengan *installation tool* pada terminal, CAP file dipindahkan ke dalam memori pada smart card. Installation prgoram pada divais menerima file CAP dan mempersiapkan aplet agar berjalan dalam *Java card virtual machine*.



Gambar 1.2. Diagram Konversi Class File menjadi CAP File

Keamanan dalam *java virtual machine* disediakan dalam bentuk *class file verifier*. Namun, class file vierifier tersebut terlalu besar untuk diimplementasikan dalam platform java card. Untuk itu, data dalam CAP file yang dibutuhkan untuk verifikasi dikemas secara terpisah dari data yang akan dieksekusi sesuai aplet tersebut. Terdapat berbagai pilihan untuk menghasilkan sistem keamanan dalam teknologi java card. Hal yang paling sederhana adalah melakukan verifikasi konten dari CAP file, namun hal ini hanya untuk divais yang cukup besar. Sedangkan untuk divais yang kecil, dapat dilakukan verifikasi dari subset CAP file saja, dan pilihan lain adalah dengan melakukan kombinasi kemanan antara lain keamanan secara fisik dari terminal instalasi, kriptografi dan verifikasi konten CAP file sebelum dilakukan download.

3. Serangan pada Java Card

3.1 Serangan pada Java Card

Secara umum serangan pada java card dapat dikelompokan menjadi 3 grup, antara lain *physical attacks, observation attacks* dan *fault attacks*[4].

1. Physical Attacks

   Serangan secara fisik pada java card antara lain dengan menghilangkan layer kontak pada java card baik seperti melalui etsa. Serangan secara fisik dapat dilakukan melalui magnetik, mengakses blok perangkat keras baik digital ataupun analog juga mengganti unit input atau output. Dengan menempatkan probe pada pin input dan output, dapat pula dilakukan deteksi dari fungsi internal pada chips, memperoleh pesan data. Dengan ion beam, dapat pula dilakukan perusakan atau mengkontakan jalur pada chip sehingga terjadi kegagalan chip.

2. Observation attacks

   Serangan ini dilakukan secara non invasif yaitu melalui perangkat lunak. Smart card disimpan pada lingkungan yang membutuhkan energi listrik tinggi untuk melakukan komputasi, sehingga konsumsi daya dan radiasi elektromagnetik berubah-ubah berdasarkan perintah yang harus dieksekusinya. Dalam hal ini, serangan ini melakukan observasi kemudian mengambil kesimpulan dari perilaku fisik nya. Analisis yang dilakukan antara lain dengan teknik *differential power analyses*.

3. Fault Attacks

   Fault attack dilakukan dengan menjalankan chip diluar spesifikasi teknis baik seperti tegangan kerja yang dinaikan atau diturunkan,frekuensi *clock* sinyal yang dirubah atau amplitudo yang di kuatkan atau dilemahkan. Serangan pada tegangan suplai atau sinyal *clock* sesaat dinamakan *glitch attack.* Ada pula serangan yang dilakukan dengan pulsa optik sehingga merusak perilaku chip. Perilaku yang berubah ini dapat mengacaukan operasi kriptografi sehingga dapat diperoleh kunci rahasia dari kriptografi tersebut. Kini, *fault attack* mampu melakukan serangan pada *Java card virtual machine*. *Fault attack* tersebut merubah tegangan suplai selama proses akses pada memori non volatil, sehingga dapat merubah tegangan referensinya. Hal ini dapat mengarah pada

kesimpulan data yang tersimpan dalam memori apakah nol atau satu. Dengan cara ini, applet dapat mengalami perubahan sehingga dikatakan applet jahat (*malicious applets*).

3.2 Mekanisme *Share* yang mengarahkan pada serangan java card.

Mulanya, java card menggunakan file sistem yang diamankan dengan daftar akses kontrol untuk berbagi data antar applet. Daftar ini menentukan identitas akses dan *permission* yang diperbolehkan untuk setiap file. Metode ini baik digunakan untuk berbagi data, namun tidak untuk berbagi data antar applet java card [5].

Firewall pada applet bertugas untuk mengatur transaksi antara objek yang berbeda, disebut sebagai *context*. Setiap applet berbagi data dengan applet lain untuk *context* yang sama. Jika sebuah virtual machine mengeksekusi sebuah byte code dan menghasilkan *context*, *context* tersebut disimpan, dan JCRE memiliki kewenangan terhadap *context* tersebut. Agar sebuah applet dapat menerima *context* dari applet yang sebelumnya, maka applet tersebut harus membuat sebuah *shareable interface object* (SIO). Mekanisme *sharing* ini terjadi dengan cara sebuah applet server mendefinisikan *shareable interface* M jika ingin membagian sebuah data, sedangkan applet lainnya jika ingin memperoleh data tersebut harus membuat *object reference* (RO).

3.3 Kode berpenyakit pada Kartu

Terdapat 4 cara untuk memperoleh tipe kode berpenyakit pada smart card, antara lain manipulasi CAP file, menyalahgunakan sharable interface objects, menyalahgunakan mekanisme transaksi dan injeksi *fault* [6].

Manipulasi CAP File

Manipulasi CAP file dapat dilakukan dengan cara merubah CAP file menjadi salah kemudian dipasangkan pada kartu. Dalam hal ini, cara ini bekerja pada divais yang tidak memiliki byte code verifier internal. Salah satu cara manipulasi adalah dengan merubah tipe array data, sebagai contoh type data byte menjadi short. Hal ini akan mengakibatkan salah interpretasi tipe sehingga terjadi kegagalan dalam mengakses memori.

Penyalahgunaan Shareable Interface Objects

Shareable interface digunakan untuk menghasilkan kesalahan tipe, yaitu dengan menjalankan dua applet berkomunikasi melaui shareable interface tersebut, namun dalam *compile* dan

menghasilkan CAP file untuk applet tersebut digunakan definisi shareable interface yang berbeda.

Penyalahgunaan mekanisme transaksi

Berdasarkan Marc Witteman, ika beberapa referensi disebarkan dapat terjadi kebingungan ketika membuat tipe, sehingga mekanisme transaksi mengalami kesulitan menggunakan referensi yang mana untuk digunakan.

Fault Injection

Fault injection merupakan jenis *fault attack* dimana manipulasi cahaya dilakukan untuk merubah bytecode yang telah terpasang pada kartu sehingga menimbukan cacat pada program. Namun, *fault injection* ini tidak menghasilkan perubahan pada memori secara terkontrol, sehingga peluang untuk memperoleh kecacatan yang dinginkan menjadi kecil.

4. Deteksi Serangan pada Java Card

4.1 Redudancy based attack detection

Metode ini merupakan pendekatan untuk memproteksi sistem pada *java virtual machine* dengan cara menambahkan kode untuk memastikan bahwa applet tidak mengalami modifikasi. Apabila modifikasi terjadi, maka eksekusi akan dihentikan.

Intruksi dalam java card virtual machine terdiri dari hampir 250 set instruksi. Dengan adanya serangan secara fisik, dapat terjadi perubahan sebuah instruksi, sebagai contoh instruksi *if opcode* menjadi *nop instruction* (0x00) [7]. Dikelompokan kemungkinan *opcode* yang dapat mengalami perubahan menjadi 3 kategori yaitu *control flow related opcodes*, *nop opcodes* dan *method incovation opcodes*.

Control flow-related opcodes merupakan opcode yang membutuhkan single operand, *branch* yang disajikan dalam tabel berikut

goto	goto_w
if_acmpeq	if_acmpeq_w
if_acmpne	if_acmpne_w
if_scmpeq	if_scmpeq_w
if_scmpne	if_scmpne_w
if_scmplt	if_scmplt_w
if_scmpge	if_scmpge_w
if_scmpgt	if_scmpgt_w
if_scmple	if_scmple_w
ifeq	ifeq_w
ifne	ifne_w
iflt	iflt_w
ifge	ifge_w
ifgt	ifgt_w
ifle	ifle_w
ifnonnull	ifnonnull_w
ifnull	ifnull_w

Tabel 4.1 Control flow-related opcodes [7]

Nop opcode merupakan instruksi dengan kode 0x00. Sedangkan *method invocation opcodes* antara lain: *invokeinterface, invokespecial, invokestatic* dan *invokevirtual.*

Pendekatan yang dilakukan dalam mendeteksi serangan adalah dengan membuat daftar posisi *opcode* yang telah dikategorikan sebelumnya. Untuk itu dibuat daftar antara lain:

- *C-list* , yaitu daftar yang mennyimpan *Control flow-related opcodes* dimana C_{op} menyimpan daftar *offset* dari *opcode* dan C_{ta} menyimpan *operand* (*target*) dari *opcode*.
- *N-list* , yaitu daftar yang menyimpan offset dari seluruh *nop opcodes*
- *I-list* , yaitu daftar yang menyimpan *invocation-realted opcodes* dimana I_{op} menyimpan *offset* dari *invocation-realted opcodes* dan I_{ta} menyimpan target *invocation-realted opcodes*.

Dalam operasinya, dengan pendekatan ini dalam java card virtual machine dilakukan verifikasi lokasi dari setiap *opcodes* apakah sesuai dengan daftar, jika tidak sesuai maka dideteksi bahwa applet tersebut mengalami kerusakan.

3.2 deteksi serangan dengan Fingerprint Applets

bytecode verifier merupakan komponen keamanan dalam java sandbox model, bertugas untuk mendeteksi kerusakan dari bytecode secara keseluruhan, sehingga menghindari kegagalan ketika dilakukan eksekusi oleh *java virtual machine*. Namun bytecode verification tersebut merupakan proses yang kompleks dan memakan memori yang besar[8].

Deteksi serangan melalui *fingerprint applet* merupakan pendekatan deteksi serangan pada applet dengan cara mengumpulkan data mengenai struktur statis dari applet dan kebiasaan dinamisnya. Dibangun diagram pohon yang menyatakan relasi biner dari intetas program, antara lain bagian dari kode, mtode, kelas sehingga diperoleh hubungan. Setiap diagram pohon menyatakan eksekusi yang terjadi akibat input tertentu, sehingga dapat dikatan diagram pohon ini seperti sebuah sidik jari (finger print).

Untuk melakukan deteksi ini, mula-mula dilakukan analisis statis dinamakan MarkerBuilder. Melalui *markerBuilder* ini dilakukan proses observasi secara komputasi untuk memperoleh bycode dan operasi yang mengandung operasi kritis, antara lain NOP dan branch, sehingga diperoleh tabel yang memuat konfigurasi stack yang berhubungan dengan operasi kritis tersebut. Hal ini dilakukan dalam kondisi *off card analysis*.

Ketika aplikasi dalam smart card berjalan, sebuah komponen bernama *MutantDetector* melakukan monitoring eksekusi dari applet. Setiap applet dieksekusi, jumlah operand akan

direkam, ketika operasi dan hasil observasi bertemu, *MutantDetector* melakukan verifikasi apakah opcode mengalami perubahan atau tidak, jika hasil cek cocok maka eksekusi akan dilanjutkan, sedangkan jika tidak maka akan dihentikan.

5. Penutup

5.1 Kesimpulan

1. Java card merupakan teknologi yang memungkinkan agar program dalam bahasa java dapat berjalan dalam divais smart card
2. Agar java card dapat berjalan dalam sebuah smart card dibutuhkan sistem java card yang terdiri dari java card virtual machine, java card converter, terminal installation tools, installation software
3. Terdapat berbagai jenis serangan pada java card antara lain: physical attack, observation attack dan fault attack.
4. Mekanisme share data pada java card merupakan salah satu celah terjadinya serangan pada java card.
5. Deteksi serangan dapat dilakukan dengan cara redudancy based-attack detection dan finger printing applets.

5.2 Saran

Perkembangan aplikasi java card yang luas membutuhkan sistem keamanan yang baik karena menyangkut transaksi, identitas dan komunikasi. Agar diperoleh java card yang semakin aman dapat terjadi secara signifikan jika ada perubahan pada spesifikasi java card.

7. Daftar Pustaka

[1.] Barbu, Guillaume, Hugues Thiebeauld, and Vincent Guerin. "Attacks on java card 3.0 combining fault and logical attacks." *Smart Card Research and Advanced Application*. Springer Berlin Heidelberg, 2010. 148-163.

[2.] Card, Java. "2.1. 1 Virtual Machine Specification." *SUN Microsystems Inc* (2000).

[3.] Hashemi, Mohammad Shouaib. "Security Issues of the Sandbox inside *Java virtual machine* (JVM)." (2010).

[4.] Lackner, Michael, et al. "A defensive Java Card virtual machine to thwart fault attacks by microarchitectural support." *Risks and Security of Internet and Systems (CRiSIS), 2013 International Conference on*. IEEE, 2013.

[5.] Wu, Yuchuan, and Yaqin Sun. "Analysis and research of securing from attack for Java Card." *2010 International Conference on E-Business and E-Government*. IEEE, 2010.

[6.] Mostowski, Wojciech, and Erik Poll. "Malicious code on Java Card smartcards: Attacks and countermeasures." *Smart Card Research and Advanced Applications*. Springer Berlin Heidelberg, 2008. 1-16.

[7.] Giunta, Rosario, Giuseppe Pappalardo, and Emiliano Tramontana. "A redundancy-based attack detection technique for java card bytecode." *WETICE Conference (WETICE), 2014 IEEE 23rd International*. IEEE, 2014.

[8.] Morana, Giovanni, Emiliano Tramontana, and Domenico Zito. "Detecting attacks on java cards by fingerprinting applets." *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2013 IEEE 22nd International Workshop on*. IEEE, 2013.