

Sistem Keamanan pada CORBA

EL-695 Keamanan Sistem Informasi

Nama : Yosi Y
NIM : 232 00 077

Program Magister Teknologi Informasi
Jurusan Teknik Elektro
Fakultas Teknologi Industri
Institut Teknologi Bandung

1. Pendahuluan

CORBA (*Common Object Request Broker Architecture*) adalah suatu standard untuk sistem *objek oriented* terdistribusi yang dikembangkan oleh OMG. CORBA memungkinkan kita menggunakan aplikasi tanpa adanya batasan platform, teknologi jaringan, bahasa pemrograman, maupun letak objek pemberi service yang dituju.

CORBA sebagai system yang terdistribusi, memiliki potensi yang besar untuk ditembus dari berbagai sisi. Karena itu diperlukan suatu sistem pengamanan yang memadai pada CORBA. Makalah ini akan membahas mengenai Arsitektur CORBA dan system keamanan pada CORBA.

2. Arsitektur CORBA

CORBA (*Common Object Request Broker Architecture*) merupakan suatu spesifikasi yang dikembangkan oleh OMG (*Object Management Group*), sebuah konsorsium yang terdiri lebih dari 800 perusahaan.

Tujuan CORBA adalah untuk pengembangan pemrograman objek terdistribusi. CORBA bukanlah bahasa pemrograman, tapi merupakan spesifikasi untuk mengembangkan objek-objek terdistribusi.

Beberapa software yang mengimplementasikan CORBA misalnya ORBIX (oleh IONA Technologies), VisiBroker (oleh Inprise), dan JavaIDL (oleh JavaSoft).

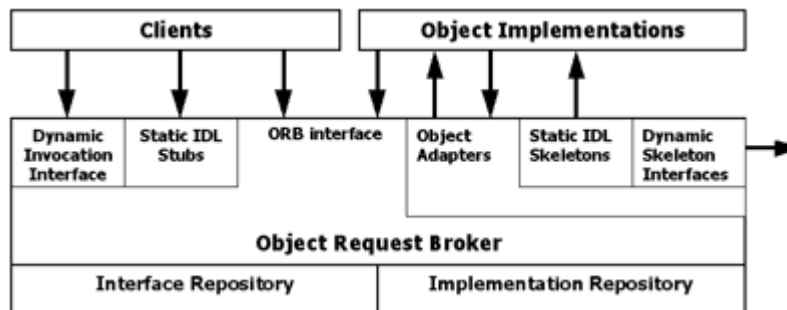
CORBA memiliki arsitektur yang berbasiskan model objek. Model ini diturunkan dari abstrak *Core Object Model* yang didefinisikan oleh OMG di dalam OMA (*Object Management Architecture*). Model ini merupakan gambaran abstrak yang tidak dapat diimplementasikan tanpa menggunakan teknologi tertentu. Dengan model tersebut, suatu aplikasi dibangun dengan standard yang telah ditentukan.

Sistem CORBA terdiri dari objek-objek yang mengisolasi suatu client dari suatu server dengan menggunakan interface enkapsulasi yang didefinisikan secara ketat. Objek-objek CORBA dapat berjalan di atas berbagai *platform*, dapat terletak dimana saja dalam suatu *network*, dan dapat dikodekan dengan bahasa pemrograman apapun asal memiliki *IDL mapping*.

Object Management Architecture (OMA) mendefinisikan berbagai fasilitas *high-level* yang diperlukan untuk komputasi berorientasi objek. Bagian utama dari OMA adalah *Object Request Broker* (ORB). ORB merupakan suatu mekanisme yang memberikan transparansi lokasi, komunikasi, dan aktivasi. Suatu objek. ORB adalah semacam *software bus* untuk objek-objek dalam CORBA. Berdasarkan OMA, spesifikasi CORBA harus dipatuhi oleh ORB.

CORBA disusun oleh komponen-komponen utama :

1. ORB (*Object Request Broker*)
2. IDL (*Interface Definition Language*)
3. DII (*Dynamic Invocation Interface*)
4. IR (*Interface Repositories*)
5. OA (*Object Adapter*)



Gambar 1. CORBA

Komponen CORBA pada sisi Client:

1. *Client Application*
2. *Client IDL Stubs*
3. *Dynamic Invocation Interface*
4. *Interface Repository*
5. *Client Side ORB Interface*
6. *ORB Core*

Komponen CORBA yang terletak di sisi Server

1. *Server Side ORB Interface*
2. *Static IDL Skeleton*
3. *Dynamic Skeleton Interface*

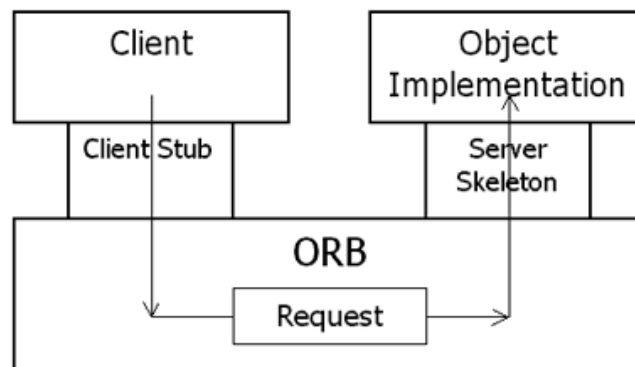
4. *Object Adapter*
5. *Server Side Implementation*

2.1 *Object Request Broker (ORB)*

ORB merupakan inti dari CORBA dan bertanggung jawab untuk menjalankan semua mekanisme yang dibutuhkan, yaitu:

1. Menemukan implementasi objek untuk memenuhi suatu *request*
2. Menyiapkan implementasi objek untuk menerima suatu *request*
3. Melakukan komunikasi data untuk memenuhi suatu *request*

Sebuah permintaan (*request*) yang dikirimkan suatu *client* ke suatu *object implementation* akan melewati ORB. Dengan ORB, yang terdiri dari *interface*, suatu *client* dapat berkomunikasi dengan *object implementation* tanpa adanya batasan *platform*, teknologi jaringan, bahasa pemrograman, dan letak objek.



Gambar 2. Request

Dengan menggunakan ORB, objek *client* bias meminta sebuah *method* pada sebuah *object server* yang bisa saja terdapat dalam satu mesin maupun jaringan yang berbeda. ORB menerima panggilan dan menemukan objek yang bisa mengimplementasikan permintaan, mengirim parameter, *invoke method*, dan mengembalikan hasil yang diperoleh.

2.2 Client

Secara umum, *client* adalah suatu program/proses yang melakukan *request* pada suatu objek. Terdapat pula *client relative*, yaitu suatu objek yang menjadi *client* dari objek lainnya.

Client suatu objek harus mengakses OR (*Object Reference*) suatu objek tertentu untuk melakukan operasi pada suatu objek. *Client* hanya mengetahui struktur logika suatu objek melalui *interface* yang dimiliki objek tersebut dan *behaviour* yang dimiliki objek tersebut saat dipanggil.

Secara umum, *client* mengakses objek dan ORB melalui *language mapping*. *Client* dapat bersifat *portable* dan seharusnya dapat berjalan tanpa harus mengubah kode pada ORB yang mendukung *language mapping* berbeda dengan objek *instance* yang mengimplementasikan *interface* berbeda.

Untuk membuat suatu *request*, *client* dapat menggunakan

1. DII (*Dynamic Invocation Interface*) yaitu suatu *interface* yang tidak tergantung pada *inteface* objek yang dituju
2. IDL *Stub*, yang tergantung pada *interface* objek yang dituju.

(ctt. Untuk fungsi-fungsi tertentu, *client* dapat berinteraksi secara langsung dengan ORB)

2.3 Object Implementation (OI)

Suatu *Object Implementation* (OI) menyediakan semantik dari objek, yang umumnya dilakukan dengan mendefinisikan data untuk *object instance* dan kode untuk method-method objek tersebut. Seringkali kita menggunakan objek lain atau menggunakan *software* tambahan untuk mengimplementasikan sifat suatu objek.

Berbagai *Object Implementation* (OI) dapat didukung oleh server yang terpisah, *librarys*, sebuah program setiap method, aplikasi terenkapsulasi, *object-oriented database*, dan lain-lain. Dengan menggunakan *object adapters* (OA) tambahan, dimungkinkan untuk mendukung suatu *Object Implementation* (OI) secara virtual.

Secara umum, *Object Implementation* (OI) tidak tergantung pada ORB atau bagaimana suatu *client* memanggil suatu objek. *Object Implementation* (OI) dapat memilih *interface*-nya ke *ORB-dependent service* yang dipilih dengan memilih *Object Adapter* (OA).

Object Implementation (OI) menerima suatu *request* melalui

1. *IDL Skeleton*
2. *Dynamic Skeleton Interface*(DSI)

Object Implementation (OI) dapat memanggil *Object Adapter* (OA) dan ORB pada saat memproses sebuah request.

2.4 Interface

Interface suatu objek dapat didefinisikan dengan cara statis, yaitu menggunakan IDL (*Interface Definition Language*). IDL mendefinisikan tipe suatu objek berdasarkan operasi-operasi (yang mungkin dijalankan pada objek tersebut) dan parameter operasi tersebut.

Interface dapat pula ditambahkan ke dalam suatu IRS (*Interface Repository Service*) yang menggambarkan komponen-komponen dari interface suatu objek. *Client* dapat mengakses komponen-komponen ini saat *runtime*.

Client meminta suatu request dengan melakukan akses ke OR (Object Reference) suatu objek yang dituju dan mengetahui tipe dari objek dan operasi-operasi yang dapat dilakukan pada objek tersebut. Client menginisialisasi request dengan memanggil rutin-rutin suatu stub yang sesuai dengan objek atau membangun request secara dinamik. Interface dinamik dan interface stub harus memiliki semantic request yang msa dalam pemanggilan suatu request.

ORB mencari implementation code yang dituju, mengirimkan parameter-parameter dan mentransfer kontrol pada Object Implementation melalui IDL Skeleton atau Dynamic Skeleton. Secara spesifik, skeleton berupa interface dan OA (Object Adapter).

Dalam mengolah suatu request, Object Implementation memberikan service pada ORB melalui OA (Object Adapter). Saat suatu request selesai dijalankan, kontrol dan nilai keluaran dikembalikan ke client. OI dapat memilih OA yang akan digunakan. Keputusan pemilihan OA ditentukan oleh jenis service yang dibutuhkan oleh OI tersebut. Informasi tentang OI diberikan pada saat instalasi dan disimpan dalam IR (Implementation Repository) yang digunakan selama pengiriman hasil request.

Dalam arsitekturnya, ORB tidak perlu diimplementasikan dalam sebuah komponen tunggal; namun, ORB didefinisikan menggunakan interface-interface yang dimilikinya. Interface-interface tersebut dikelompokkan menjadi:

1. operasi yang sama untuk semua implementasi ORB
2. operasi khusus untuk tipe objek tertentu
3. operasi khusus untuk style OI tertentu

2.5 Object Reference (OR)

Object Reference (OR) merupakan informasi yang dibutuhkan untuk menentukan sebuah objek dalam ORB. *Client* dan *Object Implementation* (OI) memiliki bagaian yang tertutup dari OR dengan language mapping, yang kemudian disekat dari representasi aktualnya. Dua implementasi ORB dapat memiliki representasi OR yang berbeda. Representasi OR pada sisi client hanya valid selama masa hidup client tersebut.

Semua ORB harus menyediakan *language mapping* yang sama untuk sebuah OR (umumnya disebut objek) untuk sebuah bahasa pemrograman tertentu. Hal ini memungkinkan sebuah program ditulis dalam bahasa apapun untuk mengakses OR secara independen terhadap ORB tertentu.

2.6 Interface Definition Language (IDL)

Objek-objek CORBA dispesifikasikan menggunakan *interface*, yang merupakan penghubung antara *client* dan server. *Interface Definition Language* (IDL) digunakan untuk mendefinisikan *interface* tersebut.

IDL menentukan tipe-tipe suatu objek dengan mendefinisikan *interface-interface* objek tersebut. Sebuah *interface* terdiri dari kumpulan operasi dan parameter operasi tersebut. IDL hanya mendeskripsikan *interface*, tidak mengimplementasikannya. Meskipun sintaks yang dimiliki oleh IDL menyerupai sintaks bahasa pemrograman C++ dan Java., perlu diingat, IDL bukan bahasa pemrograman.

Melalui IDL, *Object Implementation* (OI) akan memberitahu *client* --yang akan mengaksesnya—operasi apa saja dan *method* apa saja yang harus dipanggil *client* tersebut.

Dari definisi IDL, objek-objek CORBA dipetakan ke bahasa pemrograman –C, C++, Java, dan lain-lain—yang memiliki IDL *mapping*.

Bahasa Pemrograman yang berbeda dapat mengakses objek-objek CORBA dalam berbagai cara yang berbeda. Pemetaan dari IDL ke bahasa pemrograman tertentu harus sama untuk semua implementasi ORB. *Language Mapping* ini menyertakan definisi tipe data untuk bahasa pemrograman tertentu dan *procedure interface* untuk mengakses objek melalui ORB. Ini meliputi:

1. Struktur dari *client stub interface* (tidak dibutuhkan untuk bahasa OOP)
2. *Dynamic Invocation Interface*
3. *Implementation Skeleton*
4. *Object Adapters*
5. *Direct ORB Interface*

Language Mapping juga mendefinisikan interaksi antara pemanggilan objek dan langkah kontrol pada *client* dan implementasi. Pemetaan yang paling umum menyediakan *synchronous call*, dimana rutin mengembalikan nilai pada saat operasi suatu objek selesai dilakukan. Pemetaan tambahan memungkinkan sebuah *call* diisi dan kontrol dikembalikan kepada program.

2.7 Dynamic Invocation/Skeleton Interface

IDL *interface* yang digunakan oleh sebuah *client* ditentukan pada saat *client* dikompilasi. Hal tersebut mengakibatkan seorang programmer hanya dapat menggunakan server-server yang terdiri dari objek-objek yang mengimplementasikan *interface-interface* tersebut.

Bila suatu aplikasi membutuhkan *interface-interface* yang tak didefinisikan saat kompilasi, maka diperlukan DII (*Dynamic Invocation Interface*) atau pun DSI (*Dynamic Skeleton Interface*).

DII memungkinkan suatu aplikasi/*client* memanggil operasi-operasi dari sembarang *interface*. DSI menyediakan suatu cara untuk mengirim *request* dari sebuah ORB ke sebuah *Object Implementation* (OI) tanpa harus mengetahui tipe dari objek pada saat kompilasi.

2.7.1 *Dynamic Invocation Interface (DII)*

CORBA mendukung DII dan SII. Operasi *invocation* dapat dilakukan menggunakan *static interface* ataupun *dynamic interface*. *Static Invocation Interface* (SII) ditentukan pada saat kompilasi dan dihubungkan dengan client menggunakan stub. Sedangkan *Dynamic Invocation Interface* (DII) memungkinkan aplikasi di sisi client untuk menggunakan *server object* tanpa perlu mengetahui tipe objek-objek tersebut saat kompilasi.

DII memungkinkan *client* untuk mendapatkan sebuah *instance* dari objek CORBA dan membuat *invocation* pada objek tersebut dengan menciptakan request yang sifatnya dinamis. DII menggunakan *Interface Repository* (IR) untuk memvalidasi dan mengambil *identifier* operasi pada suatu *request* yang dibuat.

Client menggunakan *Interface Repository* (IR) untuk mempelajari tentang interface-objek yang tidak diketahui dan client menggunakan DII untuk memanggil methods suatu objek.

Empat tahap yang diperlukan saat penggunaan *Dynamic Invocation Interface* (DII):

1. Mengidentifikasi target objek yang akan dipanggil
2. Mendapatkan target interface dari objek tersebut
3. Membangun *invocation*
4. Mengirim *request* dan mendapatkan respon

Aplikasi-aplikasi *client* yang menggunakan *Dynamic Invocation Interface* (DII) tidak lebih efisien dari yang menggunakan SII, tetapi ada dua keuntungan menggunakan DII, yaitu:

- Aplikasi client dapat melakukan permintaan kepada setiap operasi meskipun tersebut tidak diketahui pada saat aplikasi dikompilasi
- Aplikasi *client* tidak harus dikompilasi ulang untuk mengakses OI yang diaktivasi ulang

2.7.2 *Dynamic Skeleton Interface (DSI)*

Dynamic Skeleton Interface (DSI) menyerupai DII, namun terletak di sisi server. DSI memungkinkan server ditulis tanpa harus mempunyai skeleton-skeleton atau informasi

tentang waktu kompilasi, dan untuk objek mana server ini diimplementasikan. Fungsi utama *Dynamic Skeleton Interface (DSI)* adalah mendukung implementasi *gateway* antara ORB yang memiliki protocol komunikasi berbeda.

2.9 Object Adapter (OA)

Object Adapter (OA) merupakan cara utama bagi sebuah *Object Implementation (OI)* untuk mengakses service yang disediakan oleh ORB. Tugas utamanya adalah melakukan masking (menutupi) perbedaan dalam implementasi objek untuk memperoleh *portability* yang lebih tinggi.

2.10 ORB Interface

ORB Interface Merupakan *interface* yang berhubungan langsung dengan ORB yang sama untuk semua ORB dan tidak tergantung pada *interface* suatu objek atau *Objek Adapter (OA)*. Karena banyak fungsionalitas ORB yang disediakan melalui OA, stub, skeleton, maupun dynamic invocation; maka ada sedikit operasi yang umum bagi semua objek.

2.11 Interface Repository (IR)

Interface Repository (IR) merupakan *online database* yang berisi tentang meta informasi tentang tipe dari objek ORB. Meta informasi yang disimpan meliputi informasi tentang modul, interface, operasi, atribut, dan eksepsi dari objek.

Interface Repository (IR) menyediakan cara lain untuk menentukan *interface* ke suatu objek. *Interface* ini dapat ditambahkan ke layanan IR. Dengan menggunakan IR, sebuah *client* akan mencari objek yang tidak diketahui pada saat kompilasi, menemukan informasi tentang *interface* objek tersebut dan implementasi suatu aktivasi dan deaktivasi.

ORB biasa menggunakan IR untuk:

1. menyediakan *interoperability* antar implementasi ORB yang berbeda
2. menyediakan *type checking* dari *signature* sebuah *request* yang melalui SII dan DII
3. Mengecek kebenaran grafik *inheritance*

4. Mengelola instalasi dan distribusi *interface definition* alam sebuah jaringan
5. Mengeizinkan *designer* aplikasi untuk memodifikasi *interface definition*
6. Mengizinkan *language compiler* untuk mengcompile *stub* dan *skeleton* dari IR bahkan langsung dari file IDL.

2.12 Implementation Repository

Implementation Repository terdiri dari informasi yang memperbolehkan ORB untuk mencari dan mengaktifasi implementasi suatu objek. Meskipun untuk suatu ORB atau lingkungan operasi, *Implementation Repository* merupakan tempat yang konvensional untuk menyimpan suatu informasi.

2.13 Internet Inter-ORB Protocol (IIOP)

CORBA mendefinisikan IIOP (*Internet Inter-ORB Protocol*) untuk mengatur bagaimana objek berkomunikasi melalui jaringan. IIOP merupakan *open protocol* yang berjalan diatas TCP/IP.

3. Sistem Keamanan pada CORBA

Selain mendefinisikan arsitektur CORBA, OMG (*Object Management Group*) juga mengembangkan definisi formal untuk servis keamanan pada CORBA.

Keamanan (*security*) merupakan hal sangat vital pada sistem komputer modern, terutama untuk sistem terdistribusi yang lebih mudah diserang dari pada sistem tradisional biasa. Oleh karena itu servis yang menunjang keamanan sangat diperlukan dalam system terdistribusi seperti CORBA.

3.1 Sistem Keamanan

Sistem keamanan (*sekuriti*) adalah proteksi umum suatu sistem informasi dari orang-orang yang akan melakukan akses yang tak diizinkan maupun interferensi dalam pengiriman informasi. Secara umum, keamanan berkenaan dengan masalah:

- *Confidentiality* (informasi hanya diberikan pada user yang berhak mengaksesnya)
- *Integrity* (informasi hanya boleh diubah oleh user yang berhak mengubahnya)

- *Accountability* (aksi-aksi user yang berhubungan dengan keamanan selalu dicatat)
- *Availability* (sistem selalu tersedia bagi user yang berhak mengaksesnya)

Sebagai tambahan dari hal-hal mendasar tersebut, spesifikasi OMG juga menyebutkan sejumlah ancaman (*threat*) dan sejumlah fitur (*feature*) penting untuk mencapai tujuan (*goal*) dari sistem keamanan.

3.1.1 Ancaman

Banyak ancaman yang terdapat pada suatu sistem informasi karena selalu saja ada orang yang mencoba untuk menjebol sistem tersebut dan berusaha mendapatkan informasi yang seharusnya tidak boleh diakses mereka.

Sistem objek terdistribusi, semisal CORBA, lebih mudah diserang dari pada sistem-sistem tradisional. Hal ini dikarenakan pada sistem terdistribusi terdapat banyak komunikasi antar komponen software yang beraneka macam sehingga menjadi peluang bagi para penjebol sistem.

Beberapa jenis ancaman yang dideskripsikan dalam spesifikasi OMG adalah:

- Kontrol keamanan (*security control*) di-bypass oleh orang lain
- Seorang *authorised user* mendapatkan akses pada informasi yang seharusnya disembunyikan darinya
- Seorang user menyamar sebagai orang lain dan mendapatkan akses, sehingga aksinya tercatat dilakukan oleh orang lain tersebut. Pada sistem terdistribusi, *user* mungkin saja mendelegasikan proses pada objek lain, sehingga objek tersebut dapat digunakan untuk kepentingannya.
- Kurangnya *accountability*, misalnya identitas user yang tidak mencukupi
- Penyadapan untuk mendapatkan data yang seharusnya dirahasiakan
- Memodifikasi pada komunikasi antar objek (mengubah, menambah maupun menghapus item)

Tingkat keamanan suatu sistem yang merupakan sasaran bagi berbagai macam serangan, kadang merupakan hasil tawar-menawar pada desain dan implementasi, biasanya untuk mencapai tujuan yang diinginkan seperti penambahan kinerja atau fungsi tambahan.

3.1.2 Goal

Hal-hal mendasar pada sistem keamanan, yaitu *confidentiality*, *integrity*, *accountability*, dan *availability* adalah dasar untuk membangun sistem keamanan pada CORBA. Namun, sistem CORBA bukanlah jenis sistem informasi biasa, melainkan karena sifat terdistribusinya, sistem ini memiliki potensi ancaman yang mungkin tidak terdapat pada sistem lain.

Oleh karena sifat terdistribusi tersebut, beberapa tujuan keamanan yang khusus pada CORBA adalah:

- menyediakan keamanan atas sistem heterogen dimana vendor yang berbeda mungkin mensuplai ORB yang berbeda pula
- karena sistem CORBA berorientasi objek, maka spesifikasi-nya juga harus berorientasi objek:
 - interface harus sepenuhnya *objek oriented* murni
 - model harus menggunakan enkapsulasi untuk menampilkan kesatuan sistem dan menyembunyikan kompleksitas mekanisme sekuriti dibawah interface sederhana
 - model harus mengizinkan implementasi polimorfisme pada objeknya yang berbasis pada mekanisme lapisan bawah berbeda, sehingga menyediakan apa yang disebut teknologi keamanan independen
- *Secure Object Invocation*, untuk memastikan *invocation* diproteksi oleh aturan sekuriti
- *Access Control* dan *Auditing*, untuk memastikan bahwa *access control* dan *auditing* yang diperlukan telah diterapkan pada *invocation* objek.

3.1.3 Fitur

Dalam rangka menghadapi ancaman yang telah disebutkan diatas dan mencapai tujuan yang diinginkan, spesifikasi OMG menentukan fitur-fitur kunci yang harus diproses oleh sistem keamanan pada CORBA, yaitu:

- *Identification* dan *Authentication*
- *Authorisation* dan *Access control* (memutuskan apakah suatu user dapat mengakses objek (umumnya menggunakan identitas secara normal dan/atau atribut istimewa lain) dan apakah atribut kontrol dari objek target dapat mengaksesnya)

- *Security Auditing* (untuk membuat mencatat semua kegiatan user yang berhubungan dengan sekuriti. Mekanisme *auditing* harus harus dapat mengidentifikasi user secara benar, bahkan setelah rangkaian call melalui banyak objek)
- Keamanan dari komunikasi antar objek (hal ini memerlukan koneksi yang terpercaya antara *client* dan target, yang mungkin memerlukan autentifikasi dari client untuk target, maupun autentifikasi dari target untuk client. Hal ini juga memerlukan *integrity protection* dan *confidentiality protection* untuk message yang dikirimkan antar object)
- *Non-repudiation* (menyediakan bukti nyata dari suatu aksi yang dilakukan oleh user)
- Administrasi dari informasi sekuriti.

3.2 Penerapan Sistem Keamanan

Spesifikasi keamanan CORBA menentukan bagaimana menyediakan keamanan dalam sistem CORBA. Spesifikasi tersebut mendefinisikan dua level fungsionalitas sekuriti yang utama (*main security functionality*), yaitu:

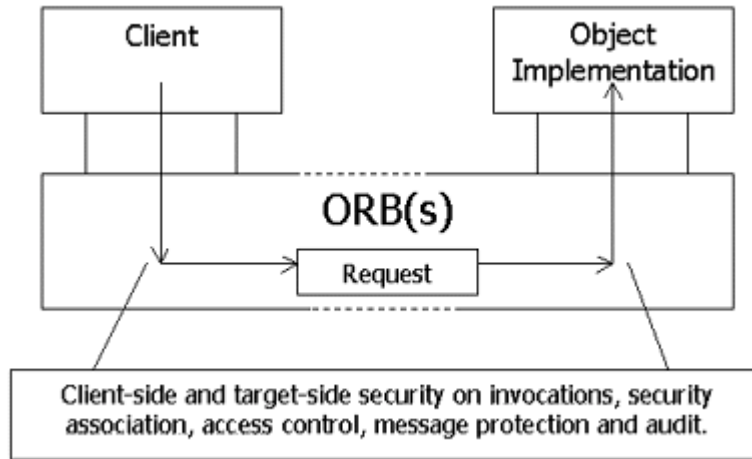
- Level 1: menyediakan keamanan pada level pertama untuk aplikasi yang tidak peduli pada keamanan dan untuk yang memiliki kemampuan terbatas dalam menangani sekuriti mereka (dalam arti *access control* dan *auditing*).
- Level 2: menyediakan fasilitas keamanan yang lebih tinggi dan mengizinkan aplikasi untuk mengontrol keamanan yang disediakan pada invokasi objek. Level ini juga termasuk administrasi dari aturan sekuriti.

Sebagai tambahan pada dua level sekuriti dasar ini, spesifikasi OMG juga menentukan suatu arsitektur yang dapat mensupport bermacam-macam aturan keamanan untuk memenuhi kebutuhan yang berbeda-beda.

3.2.1 Model Keamanan

Model referensi sekuriti yang didefinisikan pada spesifikasi OMG adalah model yang mendeskripsikan keseluruhan framework dari sistem keamanan CORBA, dengan tujuan untuk memperlihatkan fleksibilitas yang diperlukan untuk mendefinisikan berbagai aturan sekuriti yang berbeda-beda.

Gambar berikut ini merepresentasikan ensensi dari model tersebut secara sederhana:



Gambar 3. Model Keamanan untuk Sistem Objek

Secara actual, model tersebut sebagai keseluruhan dibagi menjadi beberapa model terpisah, bagian yang paling penting adalah apa yang disebut "*Principle*" beserta atribut-atributnya, dan *Secure Object Invocations*.

Principle didefinisikan sebagai user dan objek yang perlu beroperasi sendiri. Intinya, model tersebut mendeskripsikan mengapa dan bagaimana user dan apa yang dapat dilakukan mereka dapat dimodelkan.

Membuat identifikasi dan authorisasi pada tiap invokasi akan tidak efisien, oleh karena itu model tersebut mengenalkan konsep tentang *asosiasi security*. Dengan konsep keamanan asosiasi tersebut, identifikasi dan authorisasi tetap bertahan untuk banyak interaksi, tidak hanya untuk sebuah invokasi.

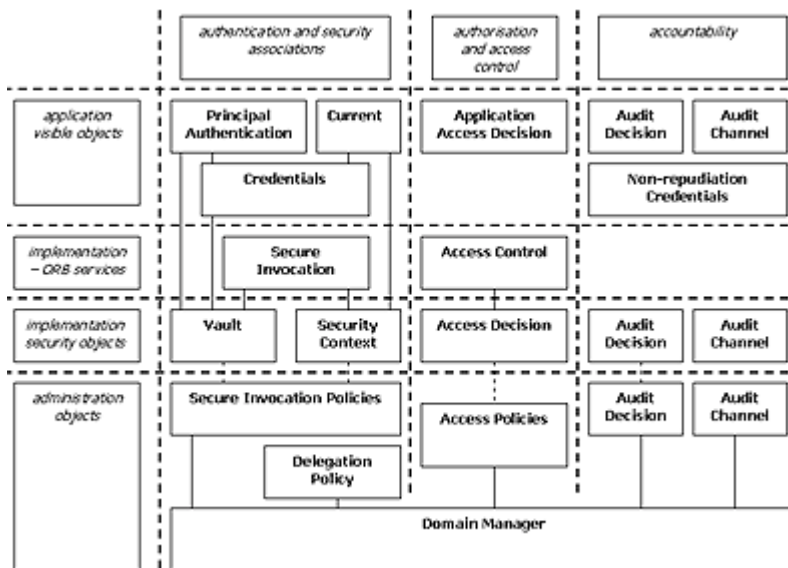
Secure Object Invocation menanggulangi masalah pada komunikasi yang tidak aman melalui *message protection*. Message dapat diproteksi dengan menggunakan algoritma enkripsi baik simetrik maupun asimetrik. Message hanya diprotek dengan kualitas proteksi yang diperlukan oleh aturan keamanan, hal ini mungkin dilakukan untuk memastikan integritas dan menyediakan konfidensialitas. Bila keamanan telah disediakan oleh layer dibawahnya (missal melalui SSL), maka komunikasi software pada ORB tidak perlu menyediakan sekuriti sendiri.

3.2.2 Arsitektur Keamanan

Arsitektur Sekuriti dalam spesifikasi OMG, menjelaskan bagaimana model sekuriti diimplementasikan dan terdiri dari berbagai bagian yang berbeda. Bagian yang paling penting adalah *model structural* yang mendeskripsikan suatu arsitektur konsep utama dimana spesifikasi lainnya berdasar. Arsitektur keamanan juga mendefinisikan empat level dari object yang digunakan selama invokasi objek, yaitu:

- Komponen Application-level (yang dapat mengindahkan maupun tak mengindahkan keamanan)
- Komponen yang mengimplementasikan servis sekuriti, independen dari teknologi sekuriti pada layer dibawahnya (spesifikasi mengizinkan penggunaan interface pengisolasi antara level ini dan teknologi keamanan, mengizinkan penggunaan teknologi keamanan yang berbeda). Komponen-komponen tersebut adalah:
 - ORB core dan servis ORB yang digunakannya
 - Servis keamanan
 - Aturan objek yang digunakan untuk menjalankan aturan sekuriti
- Komponen yang mengimplementasikan teknologi sekuriti khusus
- Proteksi dan komunikasi dasar, umumnya disediakan oleh kombinasi perangkat keras dan mekanisme Sistem Operasi

Relasi antar objek digambarkan pada gambar dibawah ini.



Gambar 4. Relasi antar Objek

Pada gambar diatas, arsitekur keamanan terlihat rumit dan terdiri dari bermacam-macam objek. Yang paling penting adalah bahwa tidak ada *code* spesifik untuk teknologi keamanan yang eksplisit sebagai bagian dari arsitektur. Semua *code* spesifik tersebut disembunyikan dalam servis ORB dan objek mengenkapsulasinya hingga dapat menggunakan *underlying* teknologi keamanan yang berbeda.

3.2.3 Interface

Interface memastikan model sekuriti diimplementasikan secara standard dan dapat dipertukarkan. Spesifikasi OMG juga menentukan sejumlah *interface*, yaitu: *Application developer's interface*, *administrator's interface*, dan *the implementor's security interface*. Sebagai tambahan pada *interface-interface* tersebut, spesifikasi OMG mendefinisikan beberapa ekstensi ke IDL.

3.2.4 Penerapan pada ORB

Sistem keamanan adalah servis CORBA dan bukan bagian integral dari CORBA core. Namun, hanya menambah beberapa objek keamanan tidak menjamin operasi yang aman, maka penerapan keamanan pada ORB diperlukan.

Keamanan CORBA meliputi:

- *Main security functionality*

terdiri dari dua pilihan, yaitu fungsionalitas sekuriti level 1 dan level 2

- *Security Functionality Options*

dalam spesifikasi ini hanya berisi *non-repudiation*.

- *Security Replaceability*

tediri dari dua pilihan:

- *ORB Services replaceability* (ORB menggunakan *interceptor* untuk memanggil servis objek, termasuk keamanan)

- *Security Service replaceability* (ORB dapat menggunakan/tidak menggunakan *interceptor* , tapi semua *call* pada servis sekuriti dibuat melalui *interfase replaceability* yang sudah dispesifikasi)

- *Secure Interoperability*

terdapat dua pilihan, yaitu:

- *Secure Interoperability Standard*, ORB dapat menggunakan dan membangkitkan informasi sekuriti dalam IOR dan dapat mengirim serta menerima request menggunakan GIOP/IIOP dengan peningkatan security (SECIOP).
- *Standard plus DCE-CIOP Option*, ORB mensupport pilihan standar dan juga menyediakan interoperabilitas yang aman dengan menggunakan servis keamanan DCE dan protocol DCE-CIOP.

Spesifikasi ORB memperkenalkan *interceptor*. Suatu *interceptor* bertanggung jawab terhadap eksekusi satu atau lebih servis ORB dan secara logika diletakkan dalam jalur invokasi antara objek *client* dan objek target. Dua tipe *interceptor* didefinisikan dalam spesifikasi ini, yaitu *request level interceptor* dan *message level interceptor*. Dimana *request level interceptor* mengeksekusi *request* yang diberikan, sedangkan *message level interceptor* mengirim dan menerima *message* yang diturunkan dari *request* dan *reply*. Spesifikasi OMG mendefinisikan servis ORB mana yang harus dipanggil oleh *interceptor*. Ketika ORB perlu untuk mem-*bind* suatu Object Reference (OR), maka ORB membuat *interceptor* yang sesuai dengan karakteristik objek tersebut.

4. Penutup

Demikianlah pembahasan tentang Sistem Keamanan pada CORBA sebagai suatu sistem terdistribusi yang memerlukan pengamanan yang lebih dari sekedar sistem biasa.

Daftar Pustaka

- Yosef, Ian. *Pemrograman Terdistribusi dan CORBA*, Diktat Kuliah, 2001
- Ballegooy, A.R. van. *The CORBA security service*. Essay for the course : "Object Oriented Programming". 1998
- Lang, Ulrich. *CORBA Security on the Web: An Overview*. University of Cambridge Computer Laboratory. 1999