AGEN PENGAMBIL INFORMASI PADA JARINGAN KOMPUTER MENGGUNAKAN *AGLETS SOFTWARE*

Bambang Sugiantoro

Program Studi Teknik Informatika
Fakultas Sains dan Teknologi UIN Sunan Kalijaga

bambang05@gmail.com

Ahmad Ashari

Jurusan Ilmu Komputer dan Elektronika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Gadjah Mada

ABSTRAK

Aglet Software Development Kit (ASDK) merupakan perangkat lunak bantu pengembangan mobile agent berbasis Java yang pertama kali diperkenalkan oleh IBM Jepang pada tahun 1997. Aglet sangat ideal dikembangkan menjadi sistem deteksi penyusupan karena ukuran file yang kecil dan mampu berjalan pada semua host yang telah dipasang Java dan ASDK versi 2.0.2. Aktifitas deteksi dilakukan pada host sehingga tidak membebani jaringan.

Metodologi pengembangan sistem yang digunakan adalah GRAPPLE (Guidelines for Rapid APPLication Engineering). Diagram UML digunakan pada tahap analisis dan perancangan. ASDK dijalankan dengan Java 2 Standard Development Kit versi 1.5.0_01 pada sistem operasi Windows XP Professional Service Pack 1 dan Linux Fedora Core 3. Tahap pengujian menggunakan 5 komputer yang terhubung pada jaringan lokal dengan skenario 1 komputer Windows sebagai server, 2 komputer Linux sebagai client, dan sisa 2 komputer windows sebagai client.

Aplikasi ini menerima input berupa alamat IP (Internet Protocol) dari host yang akan dikunjungi serta jenis pengiriman yaitu sekali, clone di host atau berulang. Objek Aglet mengirimkan status yang dilakukan selama proses deteksi di setiap host. Data yang diambil akan dikirim ke AgletMaster yang selanjutnya diolah menjadi laporan berupa file teks dan tabel. Laporan yang ditampilkan meliputi nama host, sistem operasi, port-port aktif, ketersediaan memory, waktu uptime dan ukuran directory sementara. Aplikasi ini dapat membantu administrator dalam melakukan aktifitas pengamanan jaringan.

Kata Kunci: IDS, *Mobile Agent*, *Aglet*

A. PENDAHULUAN

Jaringan komputer terus mengalami perkembangan, baik dari skalabilitas maupun teknologi yang digunakan. Hal ini memerlukan pengelolaan jaringan yang baik agar ketersediaan jaringan selalu tinggi. Tugas administrator sebagai pengelola jaringan memiliki banyak permasalahan, khususnya dalam bidang keamanan. Salah satunya adalah penyusupan pada jaringan komputer. Penyusupan (*intrusion*) adalah tindakan seseorang yang berusaha merusak atau menyalahgunakan sistem, atau setiap usaha yang melakukan *compromise integritas*, kepercayaan atau ketersediaan suatu sumber daya komputer [1].

Definisi ini tidak bergantung pada sukses atau gagalnya aksi tersebut, namun berkaitan dengan suatu serangan pada sistem komputer.

Agen merupakan entitas perangkat lunak yang didedikasikan untuk tujuan tertentu. Keunggulan Agen telah menarik perhatian banyak pihak. Salah satunya adalah IBM Jepang yang mengembangkan Aglet SDK (*Software Development Kit*) untuk mempermudah pemrogram dalam membuat Agen berbasis Java. Alamat komputer yang akan dideteksi, ditentukan oleh administrator. Deteksi hanya dilakukan pada komputer tujuan secara lokal. Perpaduan antara Agen dan Java akan menghasilkan *software* jaringan yang tangguh dengan konsumsi *bandwidth* rendah.

B. LANDASAN TEORI

Jaringan Komputer

Jaringan komputer adalah sistem komunikasi data yang melibatkan lebih dari sebuah sistem komputer yang dihubungkan dengan jalur transmisi alat komunikasi membentuk satu sistem. Komputer yang satu dapat menggunakan data di komputer yang lain, dapat mencetak laporan di *printer* komputer yang lain, dapat memberi berita ke komputer yang lain walaupun berlainan area [3].

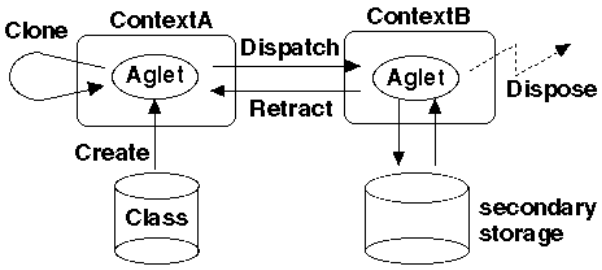
Mobile Agent

Mobile Agent merupakan entitas perangkat lunak yang memiliki kemampuan untuk bergerak dari suatu tempat ke tempat lain, dan secara mandiri melakukan tugas di tempat barunya tersebut, dalam lingkungan jaringan komputer [9]. *Mobile Agent* sering digunakan untuk mengumpulkan data atau suatu perubahan. *Mobile Agent* tidak terikat pada sistem tempat dieksekusi pertama kali. *Mobile Agent* mempunyai kemampuan unik untuk berpindah dari satu sistem ke sistem yang lain dalam suatu jaringan.

ASDK (Aglet Software Development Kit)

ASDK atau Aglet dikembangkan oleh Danny B. Lange dan Mitsuru Oshima dari IBM Tokyo Research Laboratory pada tahun 1996. Aglet adalah objek Java yang dapat bergerak dari satu *host* ke *host* lain dalam suatu jaringan [5]. Aglet yang sedang bekerja di suatu *host* dapat menghentikan eksekusinya, pergi ke *host* lain kemudian memulai eksekusinya kembali. Ketika Aglet bergerak, Aglet membawa kode program dan juga *state* dari semua objek yang membawanya. Sebuah mekanisme keamanan yang *built-in* akan mengamankan *host* dari *untrusted* Aglet. Struktur dasar Aglet adalah sebagai berikut:

a. Aglet, objek Java yang dapat berkunjung dari suatu Aglet *host* ke Aglet *host* yang lain.
b. *Proxy*, representasi dari Aglet yang melindunginya dari akses langsung ke *method public*. Menyediakan transparansi lokasi sehingga tidak diketahui lokasi sebenarnya dari suatu Aglet.
c. *Context*, tempat Aglet tersebut dijalankan, yaitu *stationary object* yang mampu melakukan pengelolaan dan pengaturan eksekusi Aglet.
d. *Message*, pertukaran pesan antar Aglet.
e. *Future Reply*, pengiriman pesan secara asinkron.
f. *Indetifier*, merupakan identitas yang dimiliki oleh setiap Aglet dan bersifat unik secara global.



Gambar 1 Siklus Hidup Aglet

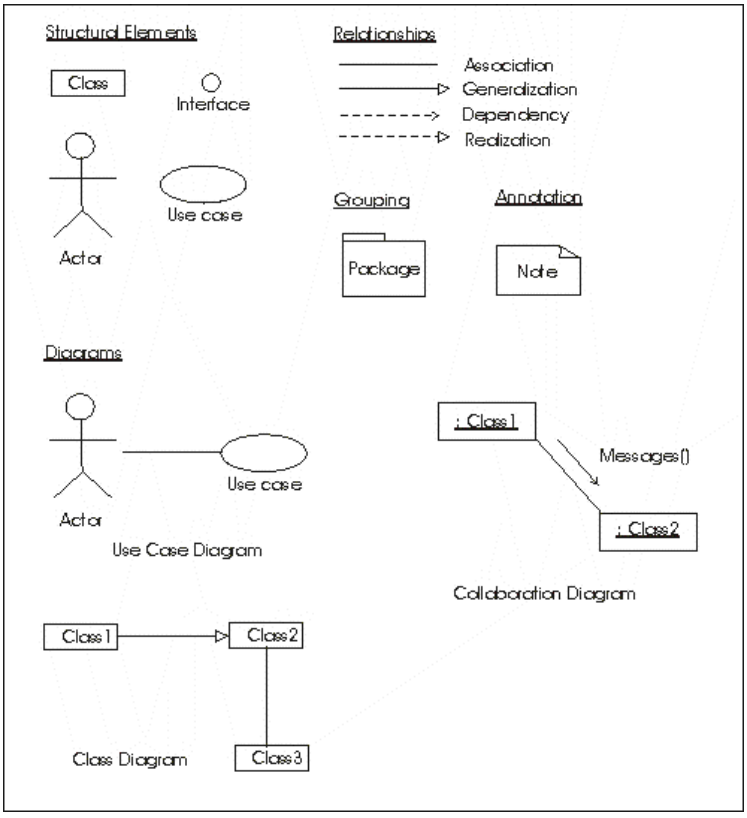
Aglet dapat melakukan operasi-operasi dasar seperti yang dijelaskan di berikut ini:

a. *Creation*, penciptaan sebuah Aglet. *Creation* terjadi di dalam *context*. Aglet yang baru diberi sebuah *identifier*, dimasukan ke dalam *context* dan diinisialisasi. Aglet mulai eksekusi segera setelah inisialisai sukses.
b. *Cloning*, proses penggandaan sebuah Aglet. Perbedaannya hanya terletak pada *identifier* yang diberikan dan eksekusi Aglet baru hasil *cloning* dimulai dari awal (*restart*). Catatan bahwa *thread* eksekusi tidak di-*clone*.
c. *Dispatching*, pemindahan sebuah Aglet dari satu *context* ke *context* yang lain. *Dispatching* akan memindahkan Aglet dari *context* yang sedang berlangsung, masuk ke *context* tujuan, kemudian memulai awal eksekusinya.
d. *Retraction*, proses untuk "menarik" Aglet dari *context* yang sedang berlangsung dan masuk ke *context* yang melakukan permintaan *retraction*.
e. *Activation*, kemampuan untuk mengembalikan Aglet ke dalam *context*.

f. *Deactivation*, kemampuan untuk menghentikan sementara jalannya eksekusi Aglet dan menyimpan *state* pada penyimpan sekunder.
g. *Disposal*, proses untuk menghentikan jalannya eksekusi Aglet dan mengeluarkannya dari *context* yang sedang berlangsung.
h. *Messaging*, komunikasi antar Aglet meliputi pengiriman, penerimaan dan penanganan *message* baik *synchronous* maupun *asynchronous*.

UML (Unified Modeling Language)

UML adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak [2]. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Tiga hal penting dari proses tersebut adalah *use case diagram*, *collaboration diagram* dan *class diagram* beserta notasi yang berhubungan dengannya. *Use case diagram* merupakan artifak dari proses analisis, sementara *collaboration diagram* dan *class diagram* merupakan artifak dari proses desain. Gambar 2 merupakan gambaran secara garis besar tentang diagram UML. Tiga diagram dipilih untuk analisis dan perancangan yaitu *class diagram*, *use case diagram* dan *collaboration diagram*. Ketiga diagram tersebut telah dianggap mampu menggambarkan aplikasi ini.



Gambar 2 Gambaran Umum Diagram UML

Konsistensi antar artifak selama proses analisis dan desain sangat penting sebab perubahan yang terjadi pada satu artifak harus juga dilakukan pada artifak sebelumnya. Berikut ini akan dibahas beberapa istilah yang berkaitan dengan proses analisis dan desain berorientasi objek yaitu [4]:

1. *Actor* adalah segala sesuatu yang berinteraksi dengan sistem aplikasi komputer. Jadi *actor* ini bisa berupa orang, perangkat keras, atau mungkin juga objek lain dalam sistem yang sama.
2. *Class* merupakan pembentuk utama dari sistem berorientasi objek karena *class* menunjukkan kumpulan objek yang memiliki atribut dan operasi yang sama. *Class* digunakan untuk mengimplentasikan *interface*.
3. *Interface* merupakan kumpulan operasi tanpa implementasi dari suatu *class*. Implementasi operasi dalam *interface* dijabarkan oleh operasi dalam *class*.
4. *Use case* menjelaskan urutan kegiatan yang dilakukan *actor* dan sistem untuk mencapai suatu tujuan tertentu.
5. *Package* adalah penampung konseptual yang digunakan untuk mengelompokkan elemen-elemen dari sistem yang sedang dibangun.
6. *Note* digunakan untuk memberikan keterangan dan komentar tambahan dari suatu elemen sehingga bisa langsung terlampir dalam model.
7. *Dependency* merupakan relasi yang menunjukkan bahwa perubahan pada salah satu elemen memberi pengaruh pada elemen lain. Elemen yang ada di bagian tanda panah adalah elemen yang tergantung pada elemen yang ada di bagian tanpa tanda panah.
8. *Aggregation* mengacu pada hubungan "*has-a*", yaitu bahwa suatu *class* memiliki *class* lain, misalnya *class* Rumah memiliki *class* Kamar.
9. *Association* menggambarkan navigasi antar *class*, berapa banyak objek lain yang bisa berhubungan dengan satu objek (*Multiplicity* antar *class*), dan apakah suatu *class* menjadi bagian dari *class* lainnya (*Aggregation*).
10. *Generalization* menunjukkan hubungan antara elemen yang lebih umum ke elemen yang lebih spesifik. Dengan *generalization*, *class* yang lebih spesifik (*subclass*) akan menurunkan atribut dan operasi dari *class* yang lebih umum (*superclass*), atau "*subclass is a superclass*".
11. *Class diagram* biasanya tersusun dari elemen *Class*, *Interface*, *Dependency*, *Generalization* dan *Association*.

12. *Realization* menunjukkan hubungan bahwa elemen yang ada di bagian tanpa panah akan merealisasikan apa yang dinyatakan oleh elemen yang ada di bagian dengan panah.
13. *Collaboration diagram* menjelaskan urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari *use case*: interaksi yang terjadi antar *class*, operasi apa saja yang terlibat, urutan antar operasi, dan informasi yang diperlukan oleh masing-masing operasi.

GRAPPLE (Guidelines for Rappid APPLication Engineering)

GRAPPLE merupakan metodologi yang fleksibel dan memberikan panduan yang jelas dalam proses pengembangan sistem. Metode ini terdiri dari 5 bagian [7], yaitu:

1. *Requirement Gathering*, mengambil informasi lengkap dari *client* tentang sistem yang akan dibangun. Wawancara dilakukan dengan *client* yang memiliki hubungan langsung dengan sistem. Tahap ini menyarankan untuk mewawancarai *client* yang memiliki kemampuan teknis.
2. *Analysis*, menggali lebih dalam hasil yang diperoleh dalam tahap sebelumnya. Tahap ini mengkaji permasalahan *client* dan menganalisis solusinya.
3. *Design*, merancang solusi yang dihasilkan pada tahap analisis. Tahap *analysis* dan *design* dapat berjalan dua arah saling menyesuaikan sampai diperoleh rancangan yang tepat.
4. *Development*, tahap ini ditangani oleh pemrogram untuk membangun kode program dan *user interface*. Pengujian program dan dokumentasi sistem juga dilakukan pada tahap ini.
5. *Deployment*, mendistribusikan produk yang dihasilkan kepada *client*. Tahap ini mencakup instalasi dan perencanaan *backup* data bila diminta oleh *client* sesuai dengan perjanjian sebelumnya.

C. ANALISIS DAN PERANCANGAN

Pada bab ini akan dilakukan konversi kelima tahap dalam GRAPPLE karena tidak semua rincian langkah dikerjakan pada penelitian ini. Tahap *Requirements gathering* dan *Analysis* dibahas secara implisit pada bagian analisis. Tahap *design* akan dibahas pada bagian perancangan. Dua tahap berikutnya yaitu *Development* dan *Deployment* akan dibahas di bagian implementasi dan Pengujian.

Analisis

Proses analisis sistem ini ditempuh dalam 3 tahap. Tahap pertama adalah menentukan pertimbangan sistem deteksi penyusupan jaringan komputer yang akan dibuat. Tahap kedua adalah menentukan sumber data dan informasi apa saja yang akan dikumpulkan, dianalisa, dan dalam bentuk apa hasilnya akan ditampilkan. Tahap ketiga adalah menentukan syarat atau batas minimal yang dipenuhi agar proses deteksi penyusupan berjalan lancar.

a. Pertimbangan Sistem

Mobile agent merupakan teknologi yang menjanjikan untuk implementasi pada sumber data terdistribusi. Jika administrator ingin melihat kondisi di setiap *host*, maka *mobile agent* dikirimkan ke setiap *host* yang diperlukan. IDS dirancang sebagai *external sensor* dengan pengumpulan data dari sejumlah *host* atau *multi host based*. Kelebihan yang diinginkan dari sistem ini adalah keperluan instalasi yang minimal, karena cukup menginstall JVM dan Aglet SDK. Keuntungan lain adalah :

a. Lebih sederhana dan mudah dipelihara daripada harus melakukan konfigurasi pada tiap-tiap komputer pada jaringan.
b. Respon yang lebih cepat untuk mendeteksi penyusupan terdistribusi pada beberapa *host*
c. Mampu menangani perubahan skalabilitas jaringan dan kesibukan *traffic*

Mobile agent di sini merupakan HIDS yang terdistribusi karena sebagian besar penyusupan bisa dideteksi pada *host*, misalnya eksekusi perintah, akses ke *service*, dan lain-lain. Serangan akan berakhir pada *host* meski melalui jaringan, sebagai contoh *flooding* jaringan juga akan terdeteksi di *host*. Alasan lain adalah memungkinkan pengumpulan data yang merefleksikan secara akurat apa yang terjadi, daripada menebak paket yang melewati jaringan.

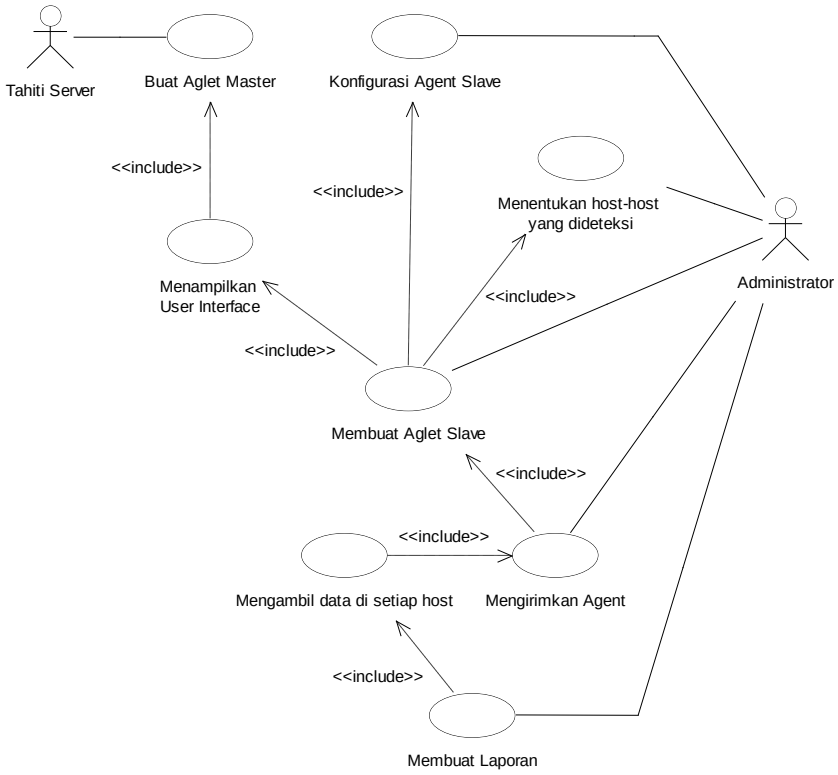
b. Informasi Deteksi Penyusupan

Sejumlah kegiatan yang akan berusaha dideteksi dan dicurigai sebagai penyusupan bergantung pada kebijakan jaringan yang ditentukan administrator. Kebijakan ini bersifat relatif dan dapat berubah sesuai dengan kebutuhan. Sebagai contoh :

a. *Login* berasal dari satu mesin ke N *host* berbeda dicurigai sebagai penyusupan terdistribusi
b. Serangan terhadap *disk space* terjadi bila media penyimpanan memiliki lebih dari 50 nama *file* yang sama yang tersebar di setiap *directory*.

Contoh kasus adalah virus HTML redlof A yang membanjiri setiap *directory* dengan *file* folder.htt.

c. Serangan terhadap direktori sementara /tmp pada Linux. Pada sistem operasi Windows diasumsikan c:\windows\temp sebagai direktori sementara.
d. Mendeteksi *port* aktif dari *trojan* yang sudah diketahui
e. Kemungkinan mesin pernah mengalami *rebooting*, berdasarkan waktu *uptime*.



Gambar 3 Diagram *use case* Deteksi dan Pembuatan *Slave*

c. Penentuan Batasan Sistem

Deteksi penyusupan dilakukan dengan mengirimkan agen melalui jaringan dari satu *host* ke *host* tujuan yang berada dalam jaringan komputer. Agar proses pengiriman agen tidak menghabiskan banyak waktu dan *bandwidth* jaringan, agen tersebut harus berukuran kecil. Syarat *host* yang akan menjadi tujuan pengiriman adalah:

a. Telah dikenal alamat IP atau nama *host*-nya dan identitas tersebut dapat diakses melalui jaringan.
b. Telah dipasang JDK serta Aglet SDK atau *Tahiti Server* yang berjalan pada *port* yang telah ditentukan
c. Menggunakan sistem operasi Windows 9x/NT/2000/XP atau Linux

Urutan langkah deteksi penyusupan suatu jaringan komputer dapat dilihat pada gambar 3 yang ditampilkan dalam diagram *use case*.

Tabel 4 Daftar Class

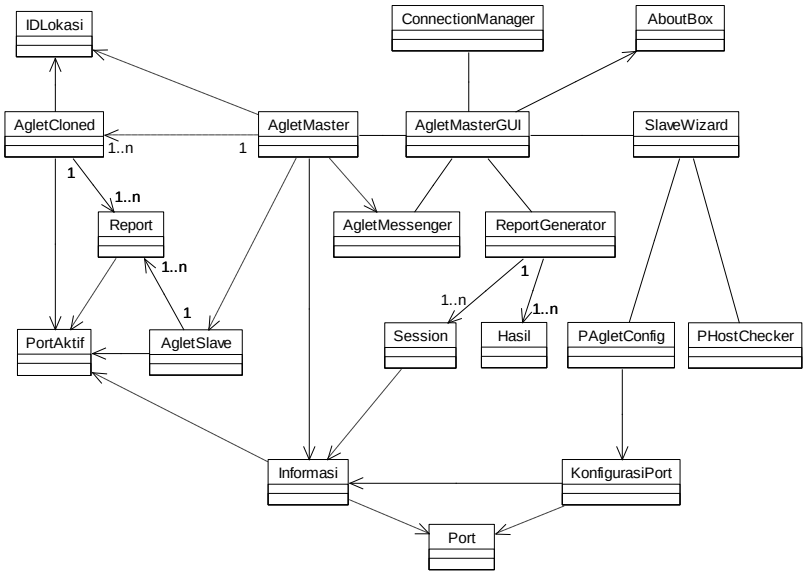
Nama Class	Keterangan
AboutBox	menampilkan informasi tentang program ini
AgletCloned	Aglet yang dikirim pada setiap *host* tujuan.
AgletMaster	merupakan *stationary aglet* yang mengendalikan *slave*.
AgletMasterGUI	merupakan antar muka utama yang berinteraksi dengan AgletMaster.
AgletMessanger	menangani aktivitas pengiriman pesan dengan aglet Slave pada *remote host* pada mode pengiriman *clone* di setiap *host*.
AgletSlave	*aglet slave* yang mengunjungi *host-host* pada *itinerary*-nya.
ConnectionManager	menampilkan tabel Hasil dan Session serta melakukan konfigurasi untuk melakukan koneksi ke *database*.
Hasil	*Class* yang menampung data yang akan disimpan pada tabel Hasil.
IDLokasi	menampung AgletID dan Lokasi *host* dari suatu *Remote aglet slave*.
Informasi	*Class* yang menampung objek-objek yang diperlukan oleh aplikasi ini untuk melakukan konfigurasi aglet dan pertukaran data.
KonfigurasiPort	menangani aktivitas yang berkitan dengan penambahan dan pengurangan *Port*.
PagletConfig	menangani aktivitas yang berhubungan dengan konfigurasi *Slave*.
PhostChecker	turunan JPanel yang menyediakan *user interface* untuk menentukan alamat *host-host* yang akan dikunjungi.
Port	Menyimpan informasi tentang *port* meliputi: mode, *alert*, nomor, deskipsi lengkap serta deskripsi singkatnya.
PortAktif	*Class* yang manampung nomor *port*, status serta *alert* yang akan dikirimkan ke *master* bila *port* tersebut dalam keadaan aktif.
Report	objek dari *class* ini akan dibawa oleh AgletSlave untuk menyimpan laporan pada setiap *host* yang dikunjungi.
ReportGenerator	*Class* yang menghasilkan laporan dan menampilkannya.
Session	*Class* yang menampung data yang akan disimpan pada tabel Session.
SlaveWizard	*Class* yang menampilkan *Wizard* untuk membuat *Slave*.

Perancangan

Proses perancangan aplikasi ini terdiri dari 3 bagian yaitu perancangan logis, *database* dan *user interface*.

a. Perancangan Logis

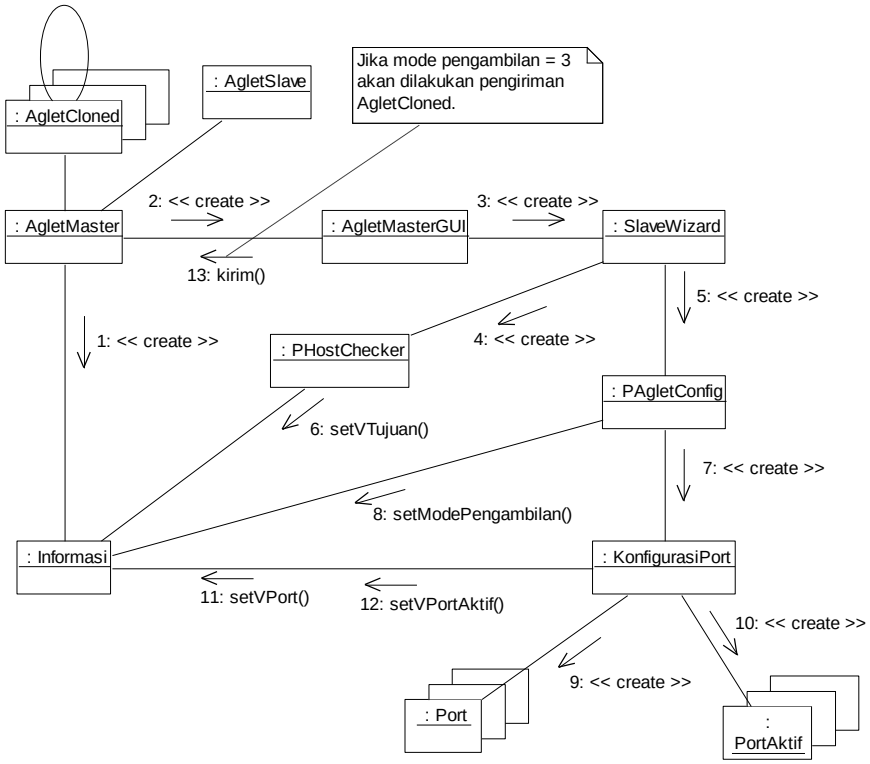
Tahap ini menggunakan 2 buah diagram UML yaitu *class diagram* dan *collaboration diagram*. Pada aplikasi ini dipergunakan 3 buah agen, yaitu: AgletMaster, AgletSlave dan AgletCloned yang merupakan turunan dari class Aglet. AgletMaster bertugas sebagai *stationary agent*, akan dijalankan di komputer *server* oleh administrator. Diagram *class* ditampilkan pada gambar 4. *Class-class* yang digunakan dalam aplikasi ini dapat dilihat pada tabel 1.



Gambar 4 Diagram *class*

Pada setiap *host* sudah dijalankan *Tahiti Server* yang siap menerima aglet yang masuk pada *port* yang sudah ditentukan. Gambar 5 merupakan *collaboration diagram* pembuatan *Slave*. Diagram tersebut menjelaskan proses pembuatan *Slave* di *host* asal sebelum dikirim ke tujuan. Langkah-langkah deteksi dan pembuatan *Slave* adalah:

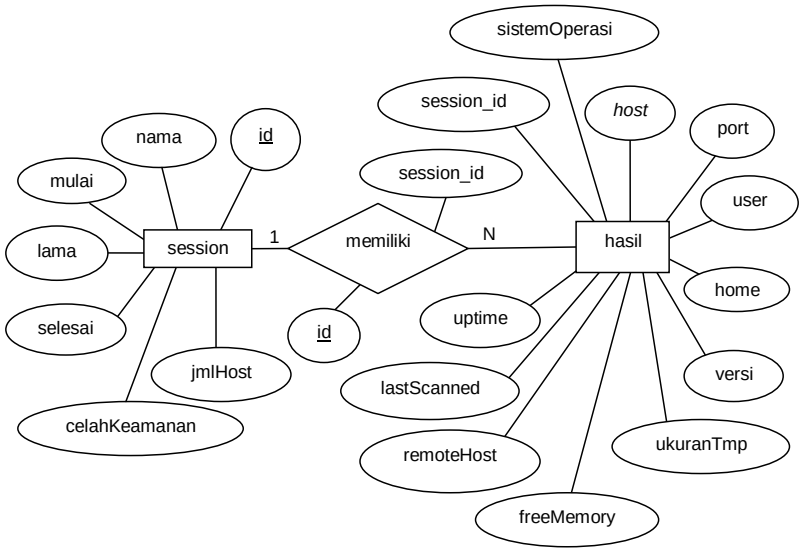
a. *Tahiti Server* membuat AgletMaster kemudian menampilkan *user interface*
b. administrator menentukan *host-host* mana yang akan dideteksi
c. Hanya terhadap *host-host* yang sedang aktif saja pengiriman *mobile agent* dilakukan.
d. konfigurasi dilakukan pada agen yang akan dikirim meliputi mode pengiriman, *port* yang dideteksi serta peringatan yang akan ditampilkan.
e. *Slave* dibuat sesuai dengan konfigurasi dan rencana perjalanan yang telah ditentukan.
f. pada setiap *host* yang disinggahi, agen akan melakukan pengambilan data yang diperlukan untuk deteksi penyusupan. Pada mode *clone* di setiap *host*, tidak akan dilakukan deteksi apapun sampai diperintahkan oleh AgletMaster.
g. agen membawa rencana perjalanannya (*itinerary*) dan akan berpindah ke *host* berikutnya sesuai dengan rencana perjalanan tersebut.
h. setelah semua *host* dalam rencana perjalanan dikunjungi, agen kembali ke *host* pengirim dengan membawa objek-objek Report yang telah dikumpulkan.



Gambar 5 *Collaboration diagram* pembuatan *Slave*

b. Perancangan *Database*

Laporan yang berasal dari objek Report akan dikonversi menjadi objek Session dan Hasil yang selanjutnya disimpan ke dalam tabel session dan hasil pada *database* aglets. Aplikasi ini menggunakan JDBC (Java Database Connectivity) *driver* org.gjt.mm.mysql.Driver untuk koneksi antara Java dan MySQL. Sebuah *session* memiliki banyak hasil dengan relasi *master-detail*. *Field* yang digunakan sebagai penghubung adalah session.id dan hasil.session_id.



Gambar 6 Diagram hubungan antar entitas

c. Perancangan *User Interface*

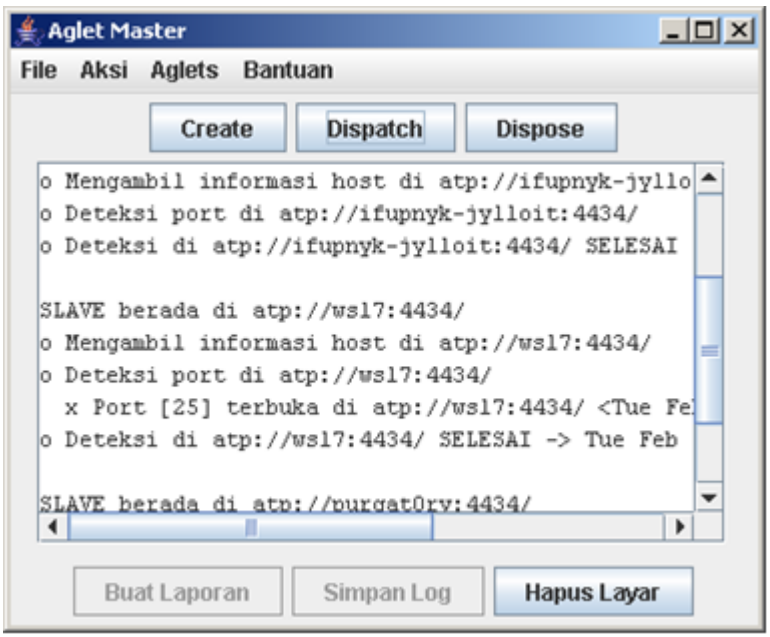
User interface atau tampilan antarmuka pengguna yang dirancang pada aplikasi ini mengacu pada *class diagram* pada gambar 4. Tidak semua *class* memiliki tampilan antar muka. Hanya *class-class* yang berinteraksi dengan *user*. Khusus untuk *class* AgletMasterGUI diturunkan dari *class* javax.swing.JFrame sedangkan *class-class* yang lain diturunkan dari *class* javax.swing.JDialog.

D. IMPLEMENTASI DAN PENGUJIAN

Tahap implementasi dan pengujian aplikasi ini dilakukan pada 2 sistem operasi yang berbeda yaitu Windows XP Professional Service Pack 1 dan Linux Fedora Core 3.

Implementasi

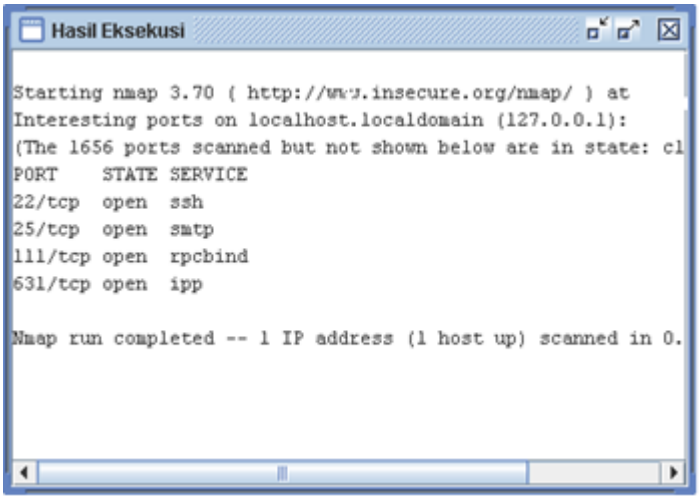
Aplikasi ini diimplementasikan dengan Borland JBuilder 9 Enterprise. Pembuatan kode program secara garis besar dilakukan di Windows. Tidak ada masalah dalam perubahan kode baik di Windows maupun Linux karena JBuilder dapat berjalan pada kedua sistem operasi ini. Khusus untuk deteksi pada Linux, *coding* dilakukan di Linux. Perancangan *database* diimplementasikan dengan mySQL Server. Database bernama aglets diciptakan yang memiliki 2 tabel yaitu session dan hasil. *Class-class* yang dihasilkan dikelompokkan dalam sebuah *package* bernama ibaglets. *Class* yang dpanggil saat eksekusi program adalah ibaglets.AgletMaster pada Aglet Server.



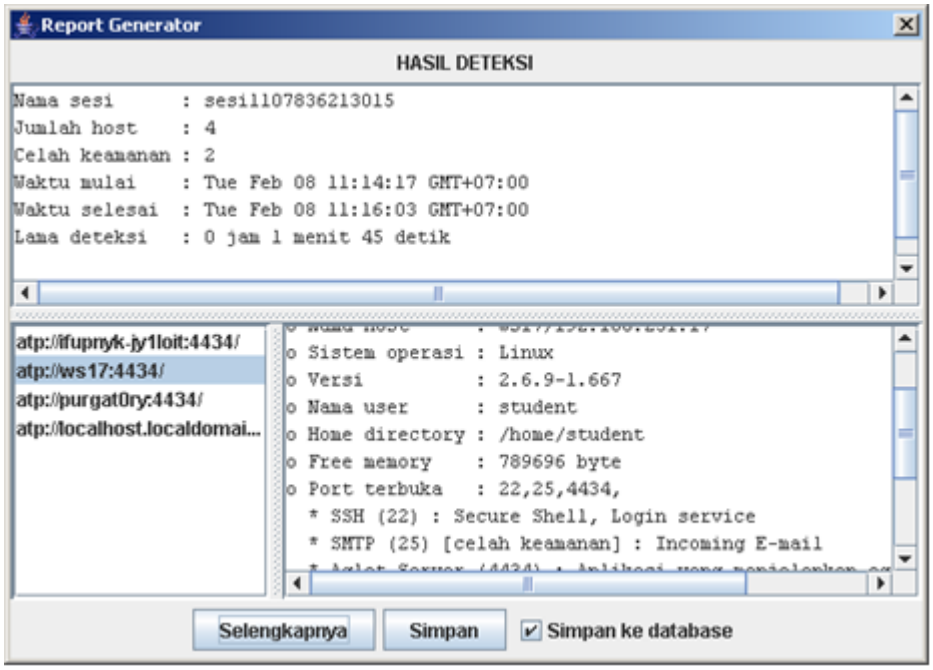
Gambar 7 Tampilan AgletMasterGUI

Pengujian

Tahap pengujian menggunakan 5 komputer yang terhubung di jaringan lokal. Sebuah komputer Windows berperan sebagai *server*, 2 komputer Linux dan 2 komputer Windows berperan sebagai *client*. Pengujian dilakukan dalam 3 tahap yaitu deteksi dengan mode pemilihan sekali jalan, *clone* di setiap *host* dan berulang 3 kali. Ketiga tahap tersebut akan menghasilkan format laporan yang sama. Khusus pada mode *clone* di setiap *host*, administrator dapat melakukan komunikasi secara interaktif dengan mengirimkan pesan dari *server*. Gambar 7 adalah tampilan utama aplikasi ini. Hal-hal yang dilakukan *Slave* pada *client* dilaporkan ke *server* dan ditampilkan pada JTextArea status.



Gambar 8 Tampilan hasil eksekusi program



Gambar 9 Tampilan ReportGenerator

Administrator dapat menjalankan perintah-perintah eksternal sistem operasi untuk mengambil informasi yang lebih lengkap untuk deteksi atau perawatan sistem. Bila terdapat kesalahan perintah, akan muncul tampilan kosong. Gambar 8 adalah tampilan eksekusi perintah nmap localhost pada Linux yang dijalankan dari server. Laporan hasil deteksi ditampilkan pada gambar 9. Tombol selengkapnya berfungsi untuk menampilkan informasi lengkap tentang *host* yang alamatnya terpilih.

E. KESIMPULAN

Kesimpulan dari hasil analisis, perancangan, implementasi dan pengujian adalah:

a. Aplikasi ini mampu berjalan pada sistem operasi Windows dan Linux untuk melakukan pengambilan informasi pada jaringan komputer
b. Aplikasi ini membantu maintenance pada jaringan lokal dengan menjalankan perintah-perintah eksternal administrasi jaringan

DAFTAR PUSTAKA

[1] Bace, R., Mell, Peter, 2009, *Intrusion Detection System*, NIST Special Publication.

[2] Dharwiyanti, Sri, 2003, *Pengantar Unified Modeling Language (UML)*, http://www.ilmukomputer.com, diakses pada 10-05-2010.

[3] Jogiyanto, 2000, *Pengenalan Komputer*, Penerbit Andi,Yogyakarta.

[4] Hermawan, Julius, 2004, *Analisa Desain dan Pemrograman Berorientasi Obyek dengan UML dan Visual Basic.net*, Penerbit Andi, Yogyakarta.

[5] Lange, D.B, 2009, *Java Aglet Application Programming Interface White Paper – Draft 2*, http://www.trl.ibm.co.jp/aglets/api/Packagecom.ibm.aglet.html, diakses pada 10-7-2010.

[6] Oshima, Mitsuru, 2008, *Aglet Spesification 1.1 Draft*, http://www.trl.ibm.co.jp/aglets/spec11.htm, diakses pada 2-11-2010.

[7] Schmuller, Joseph, 1999, *Teach Yourself UML in 24 Hours*, Sams Publishing, Indianapolis.

[8] Sugiantoro, Bambang, 2009, *Perancangan dan Implementasi Aplikasi Mobile Agent untuk Deteksi Penyusupan pada Jaringan Komputer*.

[9] Wahono, Satria, Romi, 2009, *Pengantar Software Agent: Teori dan Aplikasi*, http://www.ilmukomputer.com, diakses pada 25-02-2010.