DEVELOPING MOBILE AGENT FOR INTRUSION DETECTION

Bambang Sugiantoro[*1] , Retantyo Wardoyo[#2], Sri Hartati[#3], Jazi Eko Istiyanto[#4]

[*]Dept. of Informatics Engineering, State Islamic University Sunan Kalijaga, Yogyakarta, Indonesia.
[#]Dept.of Computer Sciences and Electronics, Gadjah Mada University,Yogyakarta, Indonesia.
[1]bambang.sugiantoro@uin-suka.ac.id, 2rw@ugm.ac.id, 3shartati@ugm.ac.id, [4]jazi@ugm.ac.id

ABSTRACT

Mobile agent is a certain agent actively able to move from one computer to another or even travel across network to perform the task assigned. This research aimed to develop mobile agent model to detect distributed intrusion (Distributed Intrusion Detection System).

It was expected that the produced model was able to perform its task in nearly real time manner, to be immune against attack, and also fairly small overhead at the host, either in memory consumption or CPU usage. In this research, the Aglets concept was also developed to aid intrusion detection. This approach was also maintained to solve certain issues such as scalability, reliability, and configurability. Hence, the advantages and limitations of agent-based approaches will be when it is applied in real world.

KEYWORDS

Developing Mobile agent model, Aglets and Intrusion Detection

1. INTRODUCTION

Computer network is currently developing from time to time, either on its scalability or number of nodes or even the technology adopted. It needs such a good network management to ensure the high level of its network availability. The tasks that the network administrator performs has numerous problems, among others, those dealing with computer network security.

Intrusion is an action that an individual performs to destroy or abuse system, or any other measurement to compromise the integrity, reliability or availability of computer resources [2]. Such a definition does not depend on the success or failure level of the action; hence, it is related with any attack to the computer system.

Intrusion detection (ID), in short, is effort to identify the existence of intruder who access the system with no authorization (e.g.: cracker) or an authorized user who abuses his privilege on the system resources (e.g.: insider threat) [4]. Intrusion Detection System (IDS) is a computer system (may be a combination of software and hardware) aiming to conduct intrusion detection [7]. IDS will present such notification when there is a suspicious act, or illegal action, detected; yet, IDS will not prevent the intrusion. Observations toward the notification n depends on how well the IDS was configured.

DOI : 10.5121/ijcsit.2013.5211

Software Agent (hereinafter referred to as 'Agent') is an software entity dedicated for certain purposes [3]. Agent can be loaded with its own idea to do some certain jobs. A number of research in this field had created some applied function of agent such as distributed meeting scheduler, network mapping, auction, and database searching.

Mobile Agent is a breakthrough in software development. Once again, agent is a software entity dedicated to certain purposes, therefore, its advantages has attracted attention from a lot of researcher. One of them, IBM Japan, developed Aglet SDK (Software Development Kit) to facilitate programmer to create Java based Agent.

2. THEORY

IBM Aglets Workbench (Aglets) was developed by Danny B. Lange and Mitsuru Oshima from IBM Tokyo Research laboratory in 1996. Aglets is a Java object that is able to migrate from a host to another host in a network. Aglets working in a host could stop its execution, migrate to other host, and continue its execution. When aglet migrate, it carries the program code and each state from every single object carrying it. A built in security mechanism is able to secure a host from an untrusted aglet.

The term aglets itself is a combination of two words: agent and applet. The difference with an applet is that aglet also carries state and itinerary (migrating plan). ASDK (Aglets Software Development Kit) is a software packet which can be used to write a mobile agent application. ASDK is based on Java language and can be obtained freely from the internet either in program code form or byte code from compiled binary file.

Tahiti is an application program serving the works as aglets server. Tahiti is bundled inside one package with ASDK. A number of server (Tahiti) can be run in a single computer by labeling each one of them with different port number. Tahiti also provide user interface for monitoring, creating, sending, and extermination of an aglet and also determining privilege access for an agent.

Aglet enable automation of a lot of processes -which is commonly done manually by users- which transform user's interaction with the internet. Aglet is based on these terms structures described as following: *Aglet* is a Java object which is able to migrate from an Aglet host to another host. *Proxy* is the representation of Aglet that protects it from direct access to the public method. It provides location transparency that the real location of an Aglet's execution is usually unknown. *Context*, the local location where the Aglet runs, is a stationary object which is able to process and adjust Aglet's execution. *Message* is object passing between Aglet. *Future reply* is asynchronous messaging. *Identifier* is the globally unique identity for every single Aglet.

Basic operations that an Agent can perform is described as following: *Creation*, the creation of Aglet, happens in a *context*. The newly created Aglet is then labeled with an *identifier*, inserted into context and then initialized. Aglet execution then starts as soon as the initialization succeed. *Cloning*, duplication of Aglet, creates a new copy of the same aglet with near identical aspects inside the same context. The only differences are the initial identifier and the execution of cloned Aglet restarted from its initiation point. Note that execution thread is not cloned. *Dispatching*, transferring Aglet from one context to another, migrates Aglet from its current context into destination context and then restart its execution. *Retraction* is a process to "pull" an Aglet from current context into another context that request retraction. *Activation* is a capability to return Aglet into a context. *Deactivation* is the capability to halt current execution of the Aglet and save its execution state into secondary storage. *Disposal* is a process to stop current Aglet execution and eject the Aglet from its current context. *Messaging* between Aglet consists of sending, receiving, and message handling either by *synchronous* or *asynchronous* manner.

3. DESIGN

This system analysis process is accomplished in three phases. The first phase is deciding every consideration related to the computer network intrusion detection system that will be made. Next phase is to determine data source and every single information that will be gathered, analyzed, and how it will be presented. The last phase is to determine the requirements or minimal model specification that must be satisfied to ensure the intrusion detection process running smoothly.

Mobile agent is a very promising technology to implemented in a distributed data source. If the administrator wish to monitor every host condition, mobile agent is then sent into every host that needed monitoring. IDS is designed as external sensor which gather gathering from a number of host (multi host based). Expected advantages from this system is minimal initial installation requirements where it only requires JVM and Aglet SDK. Another advantages are less complicated and easier management (no manual configuration for each unit of computer in the network), faster response to detect distributed intrusion at several hosts, and adapt to network scalability and network traffic change.

Mobile agent is the distributed HIDS (Host Intrusion Detection System) since most of intrusion can be detected at the host, say instruction execution, service access, etc. The attacks will actually arrive at host even though they were meant to cripple the network, for example, take network flooding that also detectable in host unit. Another reason to do so is to enable data collecting that accurately represent current situation rather than guessing every single contain of packet that pass through the network. This method can also determine the level and the activity type that specifically need to be monitored.

Basically, mobile agent gather the condition at the host on the network. Data from the source is taken by utilizing system and network functions provided by Java and operational instructions from operating system such as Windows or Linux. after deciding which host become the data source, mobile agent will be sent to gather information according to adjusted configuration.
Whether a number of detected suspicious activity will be considered as intrusion or not depends on network policy that is regulated by network administrator. This policy is not rigidly applied and can be altered to meet network requirements. Some policies are:

Login originated from a machine to N different host is considered as distributed intrusion
Any attack targeting *disk space*
Any attack to temporary directory /tmp for Linux. For Windows, c:\windows\temp is assumed as temporary *directory*
Detect active port from known trojan
Any possibility of engine rebooting according to uptime length

Gathered information of intrusion detection is taken from active host while gathering processes take places. This information will be shown in the object from ReportGenerator class. Then, administrator could save this information into a text or table form file.

This model utilizes database server to save the information of intrusion detection result. Report of detection result will contains detection result summary along with whole information that is obtained from target host. A special class is needed to handle the making of the report so that presented information will match our expectation.

Intrusion detection is carried on by send an agent through a network from a host to target host inside a computer network. To ensure this agent sending does not cost a lot of time and bandwidth, the agent must be rather small in size.

Requirements for a host to be target destination are: known IP addresses or host name (and the identities are accessible from the network), installed with JDK and SDK or Tahiti server (which run at determined port) and use Windows or Linux operating system.

On every host Tahiti Server is running and ready to receive aglet entering the appropriate port.

Detection and creation steps of slave are:

Tahiti Server creates *Aglet Master* and then shows *user interface*
Administrator decide which hosts that will be monitored for detection

Mobile agents will only be sent to active hosts. Configuration to the agent including sending mode, detected port and warning message that will be shown.

- *Slave* and/or agent is created accordingly with proper configuration and 'traveling plan' that have already been decided.
- Data gathering will be occurred in every single host visited by agent if there are any useful data to aid intrusion detection. In case of clone mode activation, no detection effort will be carried on until AgletMaster command to do so.
- Agent carries an itinerary and migrates to next host according to the traveling plan.
- After visiting every planned host, the agent will return to its sender host along with report objects that have been gathered.

The model utilize 3 types of agent: AgletMaster, AgletSlave, and AgletCloned which are the further development of aglets model. AgletMaster's task is to act as stationary agent and it will be run in the server computer by the administrator. Class diagram, generally, is shown in Figure 1.

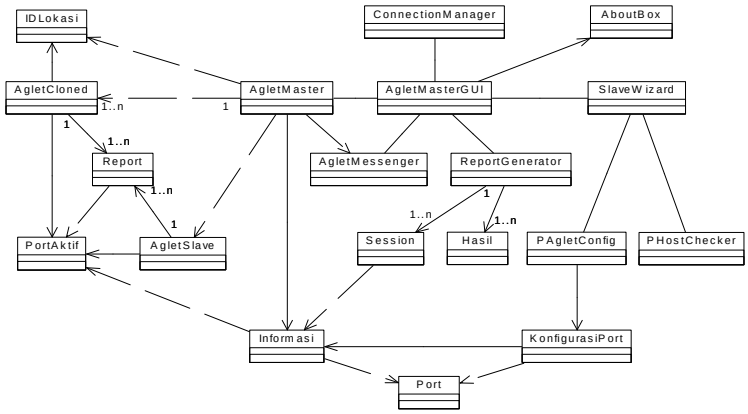

Figure 1 Class Diagram

AgletMaster as the main class in the server instantiate user interface from AgletMasterGUI class. This class related with a number of classes with each specific function. Data exchange between class is handled through Information class' object.

AgletSlave acts as the mobile agent which is able to dispatch through network into destination hosts. Gathered information from each network will be stored in Report class' objects. AgletCloned is the aglet that is sent to every hosts simultaneously. On the remote host, AgletCloned is able to communicate with AgletMaster by utilizing AgletMessenger. The later communication is done to monitor host's location where AgletCloned perform the detection.

Object from AgletSlave's class then migrate from one host to another host according to the traveling plan that has been decided before. Detection phase of the agent is illustrated with agent sending collaboration diagram in Figure 2. Agent does the *dispatch()* to every destination hosts running Tahiti Server.

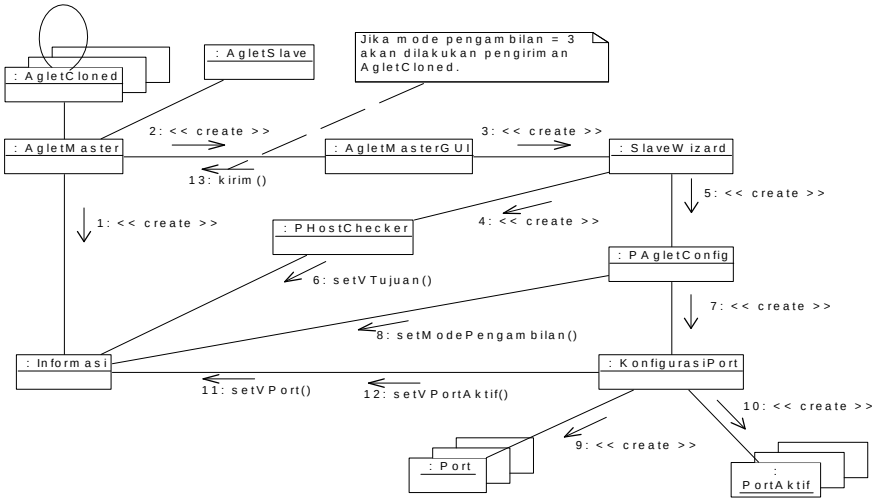

Figure 2 Create Agent

When the agent arrive at the destination host, Tahiti Server will call the method run() from the agent containing doJob() method to gather useful data. The gathered data will be stored in an object of Report class as main source to create report. Unexpected things is commonly occurred, such as the next host to be detected will be passed if--at the moment it should be detected--the host itself is inactive.

After visiting every host in the traveling plan, the agent can revisit those hosts again, from the very first one, if repeated sending mode is chosen. If either there is an error or the detection is done, agent will return back to its originating host carrying Report object from each hosts. AgentCloned will not return automatically if not instructed to do so.

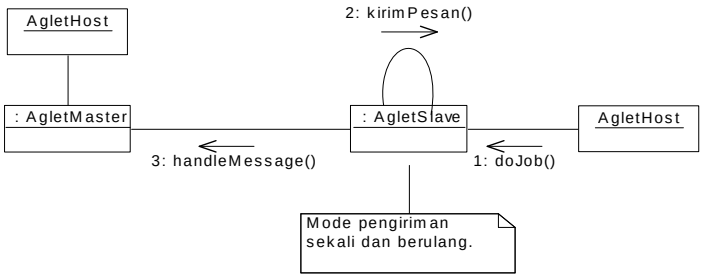

Figure 3 Aglets Slave

At the server, AgletMasterGUI will call the makeReport() method from ReportGenerator object. The Report object will be converted to Result and Session object. These two object will be stored as Vector in either database or text file form. Resulting report will consist of general information of network and specific information about each specifically visited hosts. Figure 4 describes report creation by ReportGenerator.

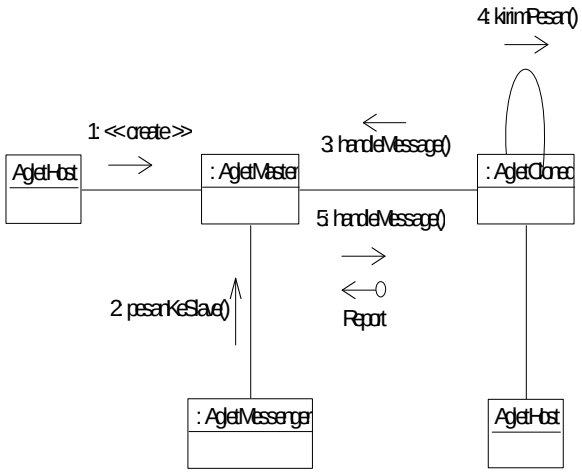

Figure 4 Aglet Cloned

Report that originated from Report object will be converted into Session and Result object. Each object represents a table, namely aglets, inside the database. Created model utilizes *Java Database Connectivity*. The *Driver* described Session table's structures. There are seven *Field* with primary key *id*. Session related to the result table by 1-to-N manner. Security hole is detected based on active ports at the destination host. Session will change automatically every single time the makeReport() method from ReportGenerator is being called.

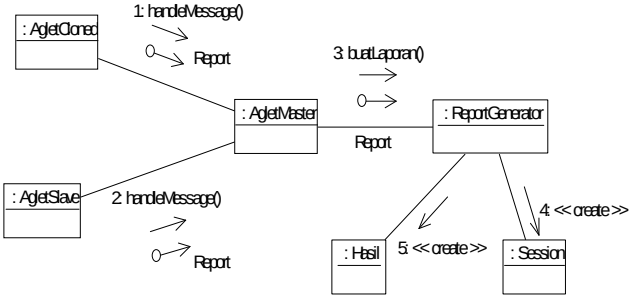

Figure 5 Report Agent

4. IMPLEMENTATION

These issues will be covered by explanation: database implementation, intrusion detection mechanism by AgletSlave, AgletCloned, message handling between Aglet and report creation by ReportGenerator. Not every single method will be discussed afterward; only those which is deemed as important. The discussion will be started by logic explanation and followed by its implementation in programming code.

```
AgletMasterGUI frame;

private void buatFrame() {
    frame = new AgletMasterGUI(this);
    info.tampilkanDiTengah(frame);
}
public void onCreation(Object obj) {
    setText("AgletMaster telah diciptakan");
    buatFrame();
}
```

When implementing database server, a database--namely aglets--is created with two tables: Session and Result. Session table has the role as *Master* while Result table will be filled with *detailf* of *master-detail* relationship. A single record of Session table has one or more detail in Result table if Session.id = Result.Session.id. This relationship is created using QueryDataSet object from com.borland.dataset package inside JBuilder 9 Enterprise.

```
public void buatSlave(Vector tujuan) {
    try {
      AgletSlave.vPortAktif = info.getVPortAktif();
      AgletSlave.BACA_SEMUA_PORT = info.isBacaSemuaPort();
      Slave.create(null, namaSlave, getAgletContext(),
                                this, tujuan, new Vector());
    }catch(Exception ex) {
      frame.kirimStatus(ex.getMessage());
    }
  }
  public void kirim(Vector vtujuan) {
    setText("Buat Slave dan kirimkan");
    buatSlave(vtujuan);
  }
```

Intrusion detection is carried on by AgletSlave and AgletCloned while AgletMaster acts as the mediator of administrator and both Slave at the remote host. AgletMaster is able to call every methods in AgletMasterGUI and vice versa. This is done to refer AgletMaster object as input parameter for AgletMasterGUI constructor. References of AgletMaster object are also exist in AgletMasterGUI.

```
public void kirimClone(Vector vTujuan) {
   jmlClonedHost = 0;
   vCloned = new Vector();
   vClonedReport = new Vector();
   Enumeration e = vTujuan.elements();
    while(e.hasMoreElements()) {
      try {
        URL tujuan = (URL) e.nextElement();
        frame.kirimStatus("[v] Kirim ke : " + tujuan);
        AgletProxy p = AgletCloned.create(namaClone,
getAgletContext(),getProxy(),tujuan,info.getVPortAktif());
      }catch (Exception ex) {
         frame.kirimStatus("[x] Gagal mengirimkan clone ke [" +
                           e.nextElement() + "]");
      }
   }
 }
```

Being the inheritance of Aglet class, Tahiti Server will call onCreation(Object obj) method while creating AgletMaster. This method is similar with constructor in Java classes. Object creation from AgletMasterGUI is done by calling buatFrame() method which will automatically call its constructor from program module. The creation and dispatch of AgletSlave to the destination host is done by calling kirim(Vector vTujuan) method which will automatically call buatSlave() method from AgletMaster. AgletSlave is created by calling *Slave.create* method with six parameters--URL of code's origin, class name of Slave, context where aglet is created, slave aglet, destination Vector, and the object that will be sent as result. Warning message will be shown at JTextAre if *Slave* creation is a failure.

AgletCloned is then created if the returned value from initiating getModePengambil() method--from Information class--showing the value of 3. The creation and dispatch of AgentCloned is carried on in kirimClone(Vector vtujuan) method from AgletMaster class. Implementation result, in code form, is shown at program module AgletCloned.create method is modified to five parameters: class name from AgletCloned, context where aglet run, proxy of the slave, destination URL and the ports that will be monitored for detection.

A host will be considered 'infiltrated' if there are any active certain ports. List of ports that considered as security hole, or suspected as intruder's way in, is stored in a file, namely infoport.dat, with these rules:

- Every row will be stored in a *Port* object
- The character '#' is used as token separator
- Every row format of writing is: access mode#port number#alert#short description#complete description
- Access mode define whether the port is being examined or not. Alert shows whether server will be notified if the port is in active state. For example, of port 21 is suspected as the troubled port, the writing format for the database is +#21#+#FTP#*File Transfer Protocol*.

File reading, using StringTokenizer class, is done by tokenPort(String tokens) method from Information class. The program code is shown by Figure 10. This method returns Port objects as the token result from the input String in the parameter. This method calling is done as many time as the row number in infoport.dat file.

Every returned Port object will be stored in vPort object from Vector class. This object is the result of calling loadFilePort() method in the Information class. The code can be seen in program module. This method is called while the Information class' constructor is being called by AgletMaster. JFileChooser will appear if the infoport.dat file cannot found to help user locate the file.

In the target host, Slave will do the port detection according to the prior configuration in KonfigurasiPort class. Port analysis in Java is accomplished by creating an object from Socket class to establish connection with the local host at the detected port number. If the connection is successful, the port is considered active. The state of the port is then stored. A message of warning will be sent to slave if alert state value in that port is true.

Data exchange mechanism between slaves is accomplished by message passing. A message can be sent from Aglet A to Aglet B if Aglet A possess AgletProxy from Aglet B and vice versa. Proxy is the tool to communicate with an aglet. In the proxy, sendMessage(*Message* m) method is called to send *Message* m to the aglet pointed by the proxy. Every message received will be handled by calling handleMessage(*Message* msg) method.

Message is an object containing String, of message type that will be sent, and also the object that will be sent as an argument. Interactive message exchange between AgletMaster and AgletCloned is carried on using AgletMessenger interface. Type of message is extracted using getKind() method and passed object, as an argument, is extracted by getArg() method. Extracted object cannot directly be used, but instead, must be first passed to *casting* process to become appropriate data type.

Reports are made based on the data gathered by slave from destination host. By using any data gathering mode, either once or repeated, AgentSlave will return to its originated host after finishing its task along with carrying Vector containing Report object from each hosts. This Vector will be stored in information class to be processed into report by ReportGenerator class.

The buatLaporan() method will fill the data in Session object and then calls reportToHasil(report r) method to convert the Report object into Result.

Result object and Session will be presented in ReportGenerator and can be saved as text file or database type. If it is saved in database type, Result and Session object will be converted into

query form by calling toSql() method in every single object. This method alter the attributes of the object into query String.

AgentCloned will send a report if instructed so by AgletMaster. In the destination host, AgletCloned will send its id and location.This information is then used by AgletMaster to create AgletCloned's proxy for message sending. AgletCloned is capable of these activities: gather host's information, read port, send report and run local application following the instruction from AgletMaster. Output result, in String, from the program's execution will be sent to AgletMaster and will be shown as soon as the message arrives. Module in Figure 15 illustrate how to catch the output from an executed console program.

CONCLUSION

The conclusion obtained from analysis, design and implementation is that the model of mobile agent for intrusion detection is successfully developed. The next research will focus on the testing of this developed model.

REFERENCES

[1] Sugiantoro, B., Wardoyo, R., Hartati, S., and Istiyanto, J.E., Mobile Agent to Perform Query on Multiple Database Server for Security, Proc. of International Conference on Informatics for Development, UIN Sunan Kalijaga, Nov. 2011, pp. C4102-105.

[2] Lange, D.B, 1997, Java Aglet Application Programming Interface (J-AAPI) White Paper – Draft 2, <http://www.trl.ibm.co.jp/aglets/api/Package-com.ibm.aglets.html>,

[3] Oshima, Mitsuru, 1998, Aglet Spesification 1.1 Draft, <http://www.trl.ibm.co.jp/aglets/spec11.htm>

[4] Bursell , M., 2009, A Aglets Puppies Workshop, online pada www.ansa.co.uk/ANSATech/FollowMe/Puppies/apm/workshop/AGLETS.pdf.

[5] Bace, R., dan Mell, P., 2002, “Intrusion Detection System”: NIST Special Publication On IDS, online pada http://www.snort.org/docs/nist-ids.pdf

[6] Balasubramaniyan, J.S., Fernandes, J.O.G., Isacoff, D., Spaffoer, E,. and Zamboni, D., 1998, “An Architecture For Intrusion Detection Using Autonomous Agents” ,online pada https://www.cerias.purdue.edu/assets/pdf/bibtex_archive/98-05.pdf.

[7] Dune, C.R., 2000, “Using Mobile Agents For Network Resource Discovery In PeerToPeerNetworks”,onlinepadahttp://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.98.4772. 27 Februari 2009

[8] Farmer, D., dan Venema, W., 2009, Improving The Security of Your Site by Breaking in to it,online pada http://www.porcupine.org/satan/admin-guide-to-cracking.html.

[9] Gopalakrishna, R., dan Spafford, E., 2000, “A Framework for distributed Intrusion Detection Using Interest Driven Cooperative Agents”, online padahttp://www.raidsymposium.org/Raid2001/papers/gopalakrishna_spafford_raid2001.pdf.

[10] Hunt, C., 1992, TCP/IP Network Adminitration, O’Reilly & Associates, Inc

[11] Is,. 2009, A Strategy for a Successful IDS Evaluation, Atlanta: Internet Security Systems, online pada www.enterprisesecuritysolutions.net/files/IDS_presentation.ppt.

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