Computing and Informatics, Vol. 31, 2012, 759–778

DISTRIBUTED DETECTION OF DDOS ATTACKS DURING THE INTERMEDIATE PHASE THROUGH MOBILE AGENTS

Ugur Akyazi

Turkish Air War College
Yenilevent, 34330, Istanbul, Turkey
e-mail: uakyazi99@yahoo.com

A. Sima Uyar

Computer Engineering Department
Istanbul Technical University
Maslak, 34469, Istanbul, Turkey
e-mail: etaner@itu.edu.tr

Communicated by Patrick Brézillon

Abstract. A Distributed Denial of Service attack is a large-scale, coordinated attack on the availability of services of a victim system, launched indirectly through many compromised computers on the Internet. Intrusion detection systems are network security tools that process local audit data or monitor network traffic to search for specific patterns or certain deviations from expected behavior, which indicate malicious activities against the protected network. In this study, we propose distributed intrusion detection methods to detect Distributed Denial of Service attacks in a special dataset and test these methods in a simulated-real time environment, in which the mobile agents are synchronized with the timestamp stated in the dataset. All of our methods use the alarms generated by SNORT, a signature-based network intrusion detection system. We use mobile agents in our methods on the Jade platform in order to reduce network bandwidth usage and to decrease the dependency on the central unit for a higher reliability. The methods are compared based on reliability, network load and mean detection time values.

Keywords: Intrusion detection, DDoS, DARPA dataset, mobile agents

1 INTRODUCTION

An intrusion detection system (IDS) is used to detect intrusions, which are the actions that attempt to compromise the integrity, confidentiality or availability of a resource [1]. Usually, an intruder first gains access to a single host by exploiting the software flaws, then tries to break into other hosts in the network via the formerly compromised host, like in the Denial of Service (DoS) attacks [2].

The objective of a DoS attack is to cause the target system to fail the services it normally provides. In a Distributed Denial of Service (DDoS) attack, one target is attacked simultaneously from a large number of sources. DDoS attacks often use the computers that have been previously exploited, so that an outsider can use them to launch an attack [3]. These zombie computers play their roles in the intermediate phase of the attack.

In this paper, we propose distributed intrusion detection methods to detect DDoS attacks in their intermediate phases on the MIT DARPA LLDOS 1.0 dataset. We use mobile agents in all of the methods except the first one. All of these methods use the alarms generated by SNORT, which is a signature-based network IDS. We use a novel simulated-real time test environment in order to get more realistic results.

The paper is organized as follows: Some background information about IDSs, distributed attack types, mobile agents and datasets are given in Section 2 with some related work. Section 3 gives detailed description of the method architectures and test implementations of our study. Section 4 reports the experimental evaluation of these IDS methods using the MIT DARPA dataset. Section 5 describes the conclusions reached after evaluating the methods and provides some future work.

2 LITERATURE REVIEW

2.1 Background

2.1.1 Distributed Intrusion Detection Systems (DIDS)

A single activity, which is a part of an attack when considered with other host activities, may look innocent when considered alone. Therefore, distributed IDS systems correlate the intrusion data collected from several hosts in a network. Most of the current distributed IDSs use a centralized controller component that performs analysis of the information it receives from each monitored hosts and the network. This causes an excessive load on the centralized controller and network [3, 4].

2.1.2 SNORT

SNORT is an open source, free Network IDS developed by Marty Roesch in 1990 [5]. It is a signature-based IDS that uses a combination of rules and preprocessors to

analyze network traffic. It supports both header and payload inspection methods. SNORT can be run in various modes from simply dumping sniffed traffic to the screen, to the IDS mode which is mostly preferred [6]. It produces alarms using misuse rules defined previously and has a language to define new rules. SNORT uses binary tcpdump-formatted files or plain text files to capture network packets.

We chose SNORT as the signature-based IDS to generate the alarms in our hosts as it is commonly used in academic research projects. SNORT, on its own, was able to detect 27 of 201 attack instances in the 1999 DARPA IDS dataset, which means 14 % true detection rate, in the study of [7] where PHAD [8] and NETAD [9] IDSs are experimentally improved and added as a preprocessor to SNORT. In [10], two open-source network intrusion detection systems (SNORT and Pakemon [11]) are combined with Cisco IOS Firewall [12] intrusion detection features to increase detection of attacks. SNORT had 44 % true detection rate when tested on the fourth week of the above stated dataset. Although it had low detection rates in 1999 DARPA IDS dataset experiments, it was able to detect the first three phases of the DDoS attack of the 2000 DARPA LLDOS 1.0 dataset which we used in our experiments. Since our methods give the necessary alarms to the security manager after detection of the first three DDoS phases, SNORT is a useful misuse detection tool for our study.

2.1.3 Distributed Denial of Service Attack (DDoS)

A DDoS attacker uses a large number of hosts to launch DoS attacks of SYN flooding, UDP flooding or ICMP flooding against any target system. DDoS tools, like TFN, Trinoo, Stacheldraht, and Mstream install daemon programs on all of the compromised hosts which are controlled by a master program [2]. DDoS attacks can cause serious damages to Internet services. Tools to gain root access to other machines are freely available on the Internet [13].

DDoS defense systems can be placed at the victim, at the intermediate level or at the source network [14]. It is very hard for victim systems to get rid of DDoS attacks. The attack is launched from a group of "zombie" computers, whose weak security systems allowed their abuse. These abused computer owners do not invest in strengthening their security, because they do not directly experience risks of the DDoS attacks and they expect the victims to take the necessary precautions. According to Canadian laws, the intermediate users who do not take the necessary precautions in these attacks are also accused of being guilty. It is possible that the same approach may be adopted in other countries too [3].

2.1.4 Mobile Agents

An agent is an autonomous, reactive or proactive program that executes a task on behalf of a user. A mobile agent has the capability of traveling through networks, interacting with machines, collecting information and returning to its dispatcher

after it finishes its tasks [16, 17]. Mobile agents are used in order to:

- reduce the network load,
- overcome network latency,
- execute asynchronously and autonomously,
- adapt to the environmental changes [16].

Mobile agents are also useful in moving the intrusion detection code to the location of the data instead of moving the data between hosts for analysis [17]. Since mobile agents can execute their tasks even when they are disconnected from their dispatchers, a failure of the controller unit of the mobile agents does not stop the ongoing intrusion detection tasks. This makes the system more reliable [2].

We use JADE (Java Agent Development Environment) in our implementation. JADE is a framework for the development and execution of peer-to-peer agent applications, which can also operate in a wireless environment [18]. Agents are located in distributed containers which provide all of the services needed for hosting and executing agents [19].

2.1.5 Datasets

Releasing intrusion detection evaluation data is a problem because of the privacy concerns. To overcome this problem, Lincoln Laboratory (LL), under sponsorship of Defense Advanced Research Projects Agency (DARPA), created the Intrusion Detection Evaluation Dataset (IDEVAL) that serves as a benchmark [20].

In 1998, 1999 and 2000, they built a network to simulate an air force base. They gathered tcpdump, Sun Basic Security Module (BSM), process and file system information after the background activities are produced with scripts, and attacks are injected at well defined points. Figure 1 shows the 1999 evaluation test-bed network. More than 200 instances of 58 different attacks were embedded in the test data [21].

The first data set to be created for DARPA in 2000 is LLDOS 1.0 which includes a DDoS attack. The basis of the attack is that an attacker seeks to show his/her talent by using a scripted attack to break into a variety of hosts around the Internet, install the components necessary to run a Distributed Denial of Service, and then launch a DDoS at a US government site. In this scenario, the attacker uses the Solaris sadmind exploit to gain root access to three Solaris hosts of the simulated network and the Mstream DDoS tool to launch the attack. An Mstream "server" is installed on each of the three abused intermediate hosts, while an Mstream "master", which controls the "servers" is installed on one of these hosts. The DDoS attack is started by these "servers" simultaneously [22].

The attack scenario has five phases:

1. IPsweep of the network,
2. probe of active hosts to look for the sadmind tool running on Solaris hosts,

3. break-ins via the sadmind exploits,
4. installation of the trojan mstream DDoS software on three hosts,
5. launching the DDoS attack.

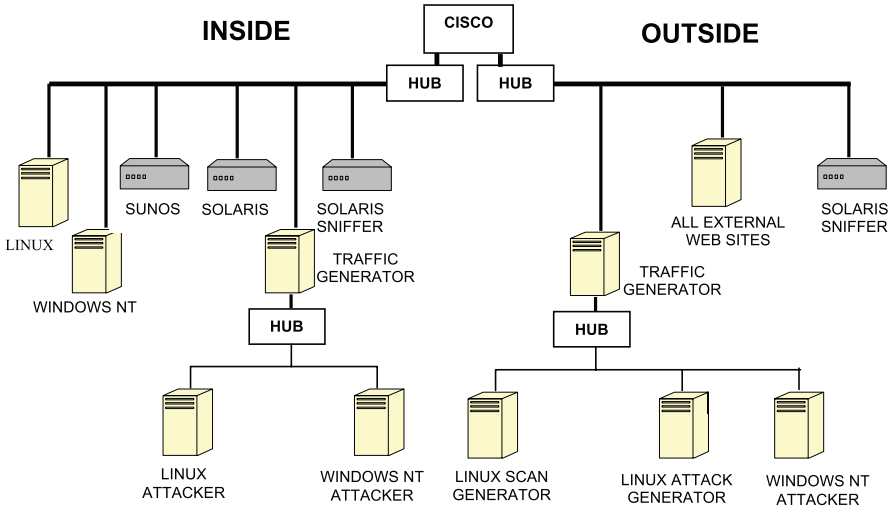

Fig. 1. Block diagram of 1999 test-bed [23]

2.2 Related Work

Intrusion detection using mobile agents is researched in several studies. In IADIDS (Independent Agents-based Distributed Intrusion Detection System) [24], static agents on the same host are in a hierarchical organization, in which the manager is at the higher level, and the detector is at the lower level. The cooperating entities among different machines are in the equity position, and there is no control center among these entities. Lack of a control center seems good but it is not explained how the system coordinates while detecting distributed attacks.

The IDS described in [25] is composed of several layers of agents. Each layer sends information to the layer above it. The mobile agents are not attack specific, and do not perform any data analysis. Analysis of data is carried out by separate decision making agents.

Manager component of [26] dispatches mobile agents which are not attack specific, and analyzes the gathered intrusion data. While a static agent is used for monitoring the hosts, the patrolling agents just collect intrusion related data from the monitored hosts. They do not correlate these data and make a decision. The GYPSY mobile agent platform [27] is used in this work.

The Intrusion Detection Agent (IDA) system [28] consists of two main components: sensors running in every monitored host that report Marks Left by Suspected Intruder (MLSI) and a central manager which is responsible for dispatching tracing agents to the host whose sensor reports an MLSI, and analyzing the information gathered by these agents.

In [17] a peer-to-peer intrusion detection system that has no central coordinator is proposed. A virtual neighborhood is created where neighbors look out for each other. Each site periodically sends mobile agents to visit and control its neighbors and report back. When inconsistent or anomalous behavior is observed, the observer neighbor initiates a voting process to take action against the compromised site.

IDReAM [15] combines mobile agents with nature-inspired learning systems. The Intrusion Detection System uses the artificial immune system approach while the Intrusion Response System uses the mechanism of ant colonies. Every dispatched mobile agent walks randomly between the different nodes and the correlation between the collected information is established by the help of deposits made by previous mobile agents.

DIDMA [2] uses mobile agents which perform the task of aggregation and correlation of the intrusion related data that it receives from static agents. DIDMA is the basic architecture for the second method proposed in our study. Mobile agents reduce network bandwidth usage by moving data analysis computation to the location of the intrusion data. The Voyager mobile agent platform [29] is used in this work.

In [17], there is no central coordinator responsible for dispatching and controlling the mobile agents; however, at each site the analysis of the gathered information is accomplished by a static agent that coordinates the mobile agents and the different reports received previously. In [28], mobile agents autonomously migrate to target systems to collect only the information related to intrusions, eliminating the need to transfer useless system logs to the analyzer server, but the decision about a potential intrusion is centralized which constitutes a major weakness. The tracing mobile agent migrates to another site to trace the path of the intrusion and identify its point of origin; however, this will only be useful if the intrusion to a host comes from another host in the network as a chain, which is not always true. In IDReAM [15], every dispatched mobile agent moves randomly between the hosts and the correlation between the collected information is done using the deposits made by previous mobile agents in a distributed way. The travel destination of the mobile agents are chosen dynamically in every host. However, there is not sufficient information in the paper about when and how many mobile agents are being created and who the creators are. The mobile agents of [25] are not attack specific, and do not perform any data analysis. Analysis of data is carried out by separate decision making agents. Patrolling agents of [26] just collect intrusion related data from the monitored hosts, while mobile agents used in our work aggregate and correlate the data received from previous hosts. In DIDMA [2], mobile agents are created centrally and take the route specified in their lists. In three of our methods mobile agents are dispatched distributedly, and travel all of the hosts if necessary. Both DIDMA and our system

complete the currently running intrusion detection tasks even in case of failure of the mobile agent dispatchers. A static agent of DIDMA generates an intrusion event on the detection of any DoS activity such as SYN, UDP or ICMP flooding from its host; whereas, intrusion events in our work are generated before a DoS activity happens in a host. The help of SNORT alarms is another main difference of our study from others.

3 METHODOLOGY

Six different distributed intrusion detection methods are proposed and tested in order to detect DDoS attacks of the MIT DARPA LLDOS 1.0 dataset through making some changes on the architecture of DIDMA [2] given in Figure 2. Mobile agents are used in all of these signature-based network IDS methods except the first one. The first four methods were initially proposed in our previous study [30] and very preliminary results were obtained. In our current study, based on our observations and the shortcomings of the previous methods, we propose two new methods, introduce a more realistic test environment and perform a detailed experimental analysis. The components of our IDS methods are as follows:

- MainAgent,
- Static Agents,
- Mobile Agents (MA),
- AlarmAgent.

Each host in the local network has a Static Agent which always monitors the host and informs the MainAgent when it detects suspicious events on its host. The MainAgent and the AlarmAgent are placed on separate protected locations in the network; so that, if the host of the MainAgent becomes unavailable due to a failure, the AlarmAgent can continue to send messages to the IDS console about intrusion threats to inform the security manager. MainAgent creates attack-specific Mobile Agents which travel through the network to gather data about suspicious events. At the end, these mobile agents decide whether there is an intrusion or not. This detection process shows some variations in each of our methods, as explained below in detail.

3.1 Method-1

The MainAgent does all the work alone in this mostly used and criticized centralized system. The Mobile Agent and the AlarmAgent are not used. The Static Agents on the hosts send intrusion-suspect messages to the MainAgent. These messages contain information about the suspicious event. If there are at least two intrusion messages of the same type, having the same attacker IP but sent by different hosts, in a given time limit, the MainAgent sends a message to the Security Console about the related distributed intrusion and saves the alarm message in a log file.

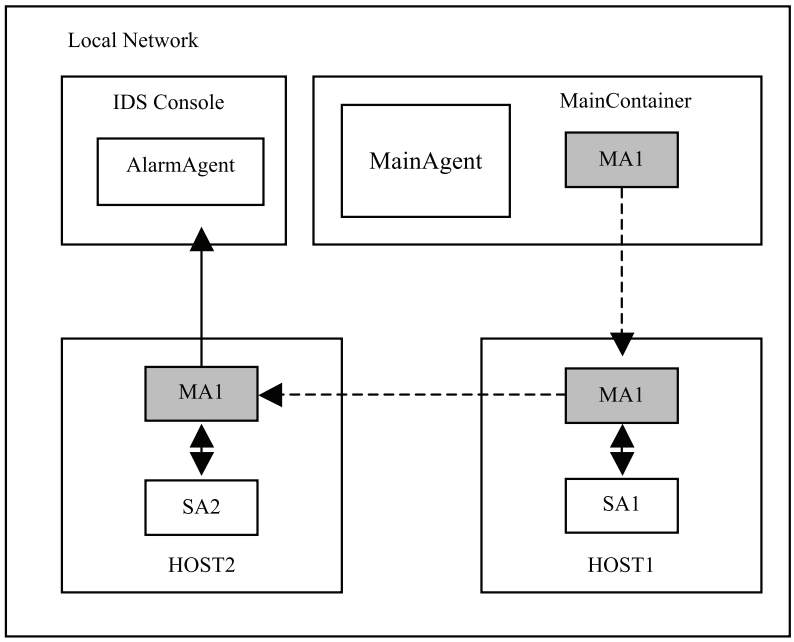

Fig. 2. General system architecture

The overall system work-time will be short due to the MainAgent doing all the work centrally. On the other hand, the intrusion-suspect messages sent to the central unit need to contain a large amount of data, since the responsibility of gathering data about suspicious events, analyzing them and making a decision about the existence of a distributed intrusion is on the MainAgent. The message data we used in our tests for rule-based detection is just the IP address of attacker; however this data may have to contain affected system files, registry records, network logs and even more for real anomaly-based detection systems. This will cause a higher network traffic load. In addition, the system will fail to work in case the MainAgent becomes unavailable either voluntarily or as a result of an attack from adversaries.

3.2 Method-2

In this method, when two messages including an attacker with the same IP number are sent from different hosts in a given time limit, the MainAgent creates a Mobile Agent and sends it to the two hosts which sent the messages. This Mobile Agent inspects the records of the related intrusion on each host it is sent to and gets the required information like the IP number of the attacker. In the end, it determines whether there is a distributed intrusion being done from the same source or not and

then it terminates itself. If there is an intrusion, it sends a message to the AlarmAgent before terminating, and the AlarmAgent displays the intrusion information on the console and saves it in a log file.

The structure resembles the one in the DIDMA [3] study, since it is taken as an example base model. In DIDMA, Static Agents report the SYN or UDP Floods originating from their hosts to the Mobile Agent Dispatcher while detecting a DDoS attack. The Mobile Agent Dispatcher creates and sends the Mobile Agent to the reporter hosts to determine whether the victim IP address is same or not in order to make the distributed attack decision. The DDoS dataset and detection phases used in our method is much more detailed than the ones used in DIDMA. The attack is detected in early stages so that the security manager will have the chance of preventing it from being completed. Jade is used in this study instead of DIDMA's Voyager mobile agent platform. One other difference is the help of SNORT in forming the intrusion messages of Static Agents as stated before.

In this method, data about the suspicious events is not moved to the central unit as in Method_1; the Mobile Agents visit the hosts and inspect the events in their places, making the required analyses and even the decision of whether there is a distributed intrusion in the system or not. This causes a much lower network load than the first method. Furthermore, the system will still work fine in case of the MainAgent being made unavailable after creating the Mobile Agent.

3.3 Method-3

In this method, unlike in Method_2, the MainAgent does not wait for two messages from the hosts to create the Mobile Agents. A Mobile Agent is created as soon as an intrusion-suspect message for an intrusion type different than those received before comes from a different host in a given time limit. The Mobile Agent visits the host which sent the message first and learns the IP number of the attacker. Later, it visits the other hosts in the network in a random order. While looking in the host records, if it finds out that there is another recording of the same attack from the same IP numbered attacker in the same time limit, it determines that there is a distributed intrusion from that attacker. It then sends a message to the AlarmAgent about this situation, which displays the intrusion information on the console and records it in a log file. Otherwise, the Mobile Agent visits all the hosts in the network and terminates itself.

The system will still work fine in case of the MainAgent being made unavailable after accepting the first intrusion-suspect message or in case of the second message that is needed to create the Mobile Agent of Method_2 is unable to reach the MainAgent. In the worst case, the overall system working time will be long, since the Mobile Agents may need to visit all of the hosts in the network when the other hosts exposed to the distributed intrusion may be close to the end of the random visiting order of the Mobile Agents.

3.4 Method-4

In this method, unlike in the previous ones, the Mobile Agents are created by the Static Agents of each host instead of the MainAgent. In order to prevent redundancy, when the StaticAgent on a host detects a suspicious event, it first asks the MainAgent if a Mobile Agent for the same type of intrusion is already created within the same time period. If there are no Mobile Agents of the same type already created in the same timeframe or if there is no reply from the MainAgent, the Static Agent creates the related Mobile Agent and sends it on. Again the Mobile Agent travels in the network in a random order as in Method_3.

In this semi-distributed method, the system will still work fine in case of the MainAgent being made unavailable any time. But, this method will also have the disadvantage of having a long work-time as Method_3.

3.5 Method-5

In this method, unlike in Method_4, the StaticAgent of the host which detects an intrusion-suspect, dispatches a Mobile Agent in order to investigate the related intrusion on other hosts, without any coordination with the MainAgent. This method is fully-distributed since the MainAgent is never used. MA makes the necessary questioning by visiting all other hosts in a random order as in Method_4. It stops its scanning when it finds a host which holds an intrusion-suspect matching the same criteria; causes other MAs, which investigate the same intrusion-suspect, to halt via a broadcast message and terminates itself after sending an information message to the Alarm Agent. Otherwise, it visits all of the hosts for questioning and terminates itself in the end.

The communication with the MainAgent while creating a MA in Method_4 is removed in this method at the expense of increasing more MA load on the network. This allows the distributed intrusion to be detected/prevented earlier through creating the MA instantly without coordination and the system becomes fully-distributed without a central coordinator.

3.6 Method-6

In this method, a system with high reliability and short detection time is created by combining the short mean detection time advantage of Method_2 with the fully-distributed structure of Method_5. The system works in mode-1 (Method_2) normally but shifts to mode-2 (Method_5) when the central unit becomes unavailable. This allows the intrusions to be detected even in the case of a MainAgent failure.

In mode-1, the MainAgent sends a message which contains its availability, as a reply to each incoming intrusion-suspect message from the Static Agents of the hosts. If this reply message does not reach the related Static Agent in two seconds, the system automatically shifts to mode-2 and continues to work as Method_5. This

means that the MainAgent is not available any more and the system needs to work fully-distributedly. At the same time, a message is sent to other Static Agents in order to warn them about shifting to mode-2. A Static Agent which gets this warning shifts to mode-2, if it already has not shifted automatically.

4 EXPERIMENTAL RESULTS

4.1 Dataset Used in the Experiments

As stated before, we used the Lincoln Laboratory Scenario (DDoS) 1.0 of DARPA Intrusion Detection Evaluation in our experiments [22]. These data files were collected over a span of approximately 3 hours on Tuesday, 7 March 2000, from 9:25 AM to 12:35 PM, Eastern Standard Time. The detailed descriptions of the five phases of the attack scenario are:

Phase 1: The attacker applies an IPsweep attack to multiple class C subnets (e.g., 172.16.115.0/24), which contain twenty hosts in total, on the air force base by sending ICMP echo-requests and listening for ICMP echo-replies to determine the active hosts.

Phase 2: Each active host is probed using the "ping" option of the sadmind exploit program to determine whether they are running the sadmind remote administration tool.

Phase 3: The active hosts running the sadmind tool are attacked in order to get root access and break-in. After these attacks, the attacker attempts a login via telnet to test whether the break-ins were successful or not.

Phase 4: Up to this phase, the attacker successfully obtains the root access of three hosts whose names are: mill (172.16.115.20), pascal (172.16.112.50), and locke (172.16.112.10). After the attacker sets up these hosts as "servers" and one of them as "master", they perform the required messaging between themselves.

Phase 5: The DDoS attack is launched in this final phase. A DDoS attack of 5 second duration against the victim with IP number of 131.84.1.31 is started simultaneously by the "servers" with the "mstream 131.84.1.31 5" command of the attacker [22].

We downloaded the offline data of the inside sensor as a tcpdump file. This inside sniffer is placed just behind a firewall, so that it gathers all network traffic in one point before it is sent to other inside hosts or the outside. This one-file tcpdump data is resolved into twenty different "alarm.ids" files which belong to each inside host by using SNORT. This way, we obtained SNORT alarms of each potential intrusion event of our dataset. The intrusion types of the DDoS attack phases written in the SNORT alarms are as follows:

- Phase 1: ICMP PING
- Phase 2: RPC portmap sadmind request UDP, RPC sadmind UDP Ping
- Phase 3: RPC sadmind query with root credentials attempt UDP, RPC sadmind UDP NETMGT_PROC_SERVICE CLIENT_DOMAIN overflow attack
- Phase 4: RSERVICES rsh root
- Phase 5: telnet to master (in order to start DDoS).

Attack phases	Attack names	Number of hosts exposed to this attack	Time when the first host is exposed to this attack	Time when the second host is exposed to this attack	Time when the last host is exposed to this attack
1	ICMP Ping	20	17:51:36.14	17:51:36.52	17:52:00.82
2a	RPC portmap sadmind request UDP	11	18:08:07.35	18:08:07.50	18:34:36.43
2b	RPC sadmind UDP ping	3	18:08:07.36	18:15:10.03	18:15:10.10
3a	RPC portmap query with root credentials attempt	3	18:33:10.62	18:34:36.45	18:34:49.00
3b	RPC sadmind UDP NETMGT PROC SERVICE CLIENT DOMAIN overflow attack	3	18:33:10.62	18:34:36.45	18:34:49.00
MIN. ATTACK DETECTION TIME = 42 MIN 59.93 SEC = 2 579 930 MILLISECONDS					
4	RSERVICES rsh root		18:50:02		18:50:38

Table 1. Execution times of attack phases

Execution times of attack phases (Turkish local time zone) and the number of hosts exposed to these attacks are given in detail in Table 1. To detect the presence of a distributed attack, we aim to detect the first three phases of this scenario and inform the security manager before the last two phases occur. In this case, at least 42 minutes and 59.93 seconds have to pass after the first host is exposed to the first attack phase, to be able to decide about the presence of a DDoS attack. This value is obtained by subtracting "*the time when the* 1^{st} *host exposed to the* 1^{st} *phase*" from "*the time when the* 2^{nd} *host exposed to the* 3^{rd} *phase*".

4.2 Test Designs

The tests are implemented on a Pentium M 1.6 GHz notebook computer, which runs on Windows XP SP2 operating system. Twenty-two Jade containers of Static Agents, the MainAgent and the AlarmAgent are created on separate DOS Command Prompts in one computer. By using this simulated setup, we made sure that there is no outside network traffic which would affect our measurements.

For an intrusion to be distributed, it has to be done on several hosts from the same source or from a group of organized sources working together in a limited time period. Therefore, in this study, two "intrusion-suspected" messages having the

above stated properties are considered sufficient for the intrusion being distributed. The time-out value for the distributed intrusions is chosen to be one hour as an average time period. This type of intrusions may even last a few seconds or a day.

In our former study [30] tests, it was assumed that the records of the attacks detected by SNORT were recorded on the related hosts in advance, so that the timestamps stated in the dataset were not taken into account. MAs continued their search without synchronizing with these timestamps. It is apparent that measurements in these conditions are not realistic. Instead of that, a technique is implemented in which the starting time of the tests is synchronized with the starting time of dataset traffic and all of the static/mobile agents consider the timestamp information of the attacks while analyzing the records on the hosts. In this technique, agents compare the timestamp of the recorded attack with its own timer while scanning the logs and move on to the next host without making any process on that record and the following ones if the attack has not been executed yet. Therefore, attacks are able to be detected at their real occurring times and are reported to the related units. Also, allowing the MAs to travel for one hour in the network becomes more meaningful in this scenario. Tests are performed in a one-hour period, since the DDoS attack phases occur in less than a one-hour period, although the dataset spans approximately three hours.

In two of 20 test runs (10 %) of Method_4 and Method_6, the system is made to work in mode-2 because of a MainAgent failure. Thus, we were able to get the detection time results of these two methods when it worked in fully-distributed mode for 10 % of the time.

4.3 Results

Detection time results taken over twenty runs are shown in Figure 3. It can be seen that the mean intrusion detection times of the methods are in the [+1 375, −359] ms range of the minimum detection time of 42 min 59.93 sec (2 579 930 ms). When mean detection times are evaluated with their standard deviations, it can be seen that methods do not differ so much in their intrusion detection times and have close values to the minimum intrusion detection time.

MAs of the last four methods in the offline tests [30] were scanning entirely, from beginning to end, the intrusion records of the "alarm.ids" files on the host that they visit, comparing the scanned record information with "intrusion_type and source_ip" data of the intrusion that caused their creation. A more effective comparison method is applied in the simulated-real time tests. In this approach, the test time is compared with the "intrusion_time" data of the scanned "alarm.ids" file, and the file is left without examining the current record and the following ones if the related attack has not occurred yet. Therefore, all of the records up to end of the file do not have to be examined unnecessarily.

In Table 2, network load measures of each method and their ranks according to these measures are given. Network load is created by the messages sent between the hosts and the Mobile Agents traveling in the network. A message is represented

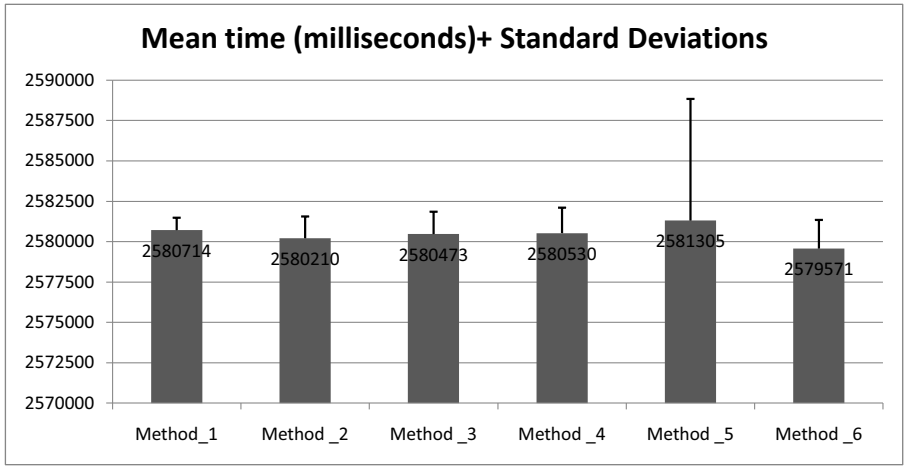

Fig. 3. Detection time results of simulated-real time tests

in the table as Mes3_SAtoMainA where this message is sent from Static Agents to MainAgent in Method_3 or Mes5_MAtoALL where this message is sent from Mobile Agents to all of the hosts in Method_5. Mobile Agents of the methods are represented as MA_4. Method_2 causes 1 375.37 KB network load as the sum of 236 Mes2_SAtoMainA messages each of which have 3 250 bit size, and 10 MA_2 mobile agents each of which have 1 050 000 bit size. The sizes of the messages and Mobile Agents which are given in parentheses change with the loads they carry. The message sizes are small in our tests since they include less but enough information about the *intrusion-suspects*.

In 10 % of the tests, MainAgent of method_4 and method_6 was made to fail and the system worked without MainAgent. The number of messages and Mobile Agents belonging to a method is stated in the column of that method where the total size is given in the end. Ranks of the methods are directly proportional to their network load sizes since the best method should have the least network load. Method_1 is not classified in this and following rankings because of its full dependency to the central unit, which is out of the scope of our study.

As stated before, MainAgent of method_3 needs one intrusion-suspect message to create a Mobile Agent while MainAgent of method_2 needs two messages. Mobile Agents of these two methods will work independently after being created. Mobile Agents of the last three methods work independently from the MainAgent in the case of its failure anytime. These "independence from the central unit" cases can be seen in Table 3, where the value "1" means that the system in the method still can work in the failure of the MainAgent in stated conditions. Ranks of the methods are inversely proportional to the sum values which are calculated according to the "$sum_i = value_{i1} + value_{i2} + value_{i3}$" formula in which "$i$" represents the method number, since the best method should be independent in more cases.

	Method_1	Method_2	Method_3	Method_4 (10 fail)	Method_5	Method_6 (10 fail)
Mes1_SAtoMainA (3 080 bit)	236					
Mes2_SAtoMainA (3 250 bit)		236				
Mes3_SAtoMainA (3 250 bit)			236			
Mes4_SAtoMainA (3 080 bit)				56		
Mes4_MainAtoSA (850 bit)				50		
Mes5_MAtoALL (490 bit)					210	
Mes6_SAtoMainA (3 250 bit)						215
Mes6_MainAtoSA (1 150 bit)						212
Mes6_MAtoALL (490 bit)						23
Mes6_SAtoALL (550 bit)						2
MA_2 (1 050 000 bit)		10				
MA_3 (1 500 000 bit)			16			
MA_4 (750 000 bit)				18		
MA_5 (750 000 bit)					2	
MA_6 (1 020 000 bit)						14
TOTAL (KB)	88.73	1 375.37	3 023.32	1 674.19	4 773.3	1 888.12
RANKS	Not classified	1	4	2	5	3

Table 2. Network load measures

In simulated-real time test results of this study, Method_6 has the disadvantage of more network load relative to Method_4, because the MainAgent of Method_6 has to send a reply message to each incoming message in order to control the mode-1 to mode-2 switches.

	MainAgent becomes unavailable anytime	**MainAgent becomes unavailable after accepting <u>one</u> intrusion-suspect message and creating a Mobile Agent**	**MainAgent becomes unavailable after accepting <u>two</u> intrusion-suspect message and creating a Mobile Agent**	**Sum**	**Ranks**
Method_1	0	0	0	0	Not classified
Method_2	0	0	1	1	3
Method_3	0	1	1	2	2
Method_4	1	1	1	3	1
Method_5	1	1	1	3	1
Method_6	1	1	1	3	1

Table 3. Independence from the central unit

	"Network load" ranks	"Independence from the central unit" ranks	Mean of ranks	Normalized ranks
Method_2	1	3	2	2
Method_3	4	2	3	3
Method_4	2	1	1.5	1
Method_5	5	1	3	3
Method_6	3	1	2	2

Table 4. Ranks of the methods of simulated-real time tests

The overall ranks of the methods according to the two criteria are given in Table 4. "Intrusion detection time" is not used as the third criterion since all of the methods have similar values. Ranks are normalized after their mean values are calculated. According to these ranks, Method_4 turns out to be the best one.

5 CONCLUSION

We designed and tested six distributed intrusion detection methods to detect DDoS attacks in the MIT DARPA LLDOS 1.0 dataset, considering the timestamp information of attacks in a simulated-real time test environment. We used mobile agents in all of the methods, except the first one, which are all signature-based network IDS. In Method_1, there is a centralized DIDS without mobile agents; the security manager is informed of a distributed intrusion in case of getting two intrusion-suspect messages of the same type and same attacker IP from different hosts in one hour. Mobile Agents, which are created by the MainAgent under suitable conditions and which visit only the two listed hosts to inspect related data in their places, give the distributed intrusion decision in Method_2. In Method_3, the MainAgent does not wait for two messages from the hosts for creating the Mobile Agent. In Method_4, Mobile Agents are not created by the central unit, but by the Static Agents on the hosts through coordination with the central unit. In Method_5, Mobile Agents are dispatched by the StaticAgent of the hosts, which detect *intrusion-suspect* without coordinating with the MainAgent and other hosts. This method is fully-distributed, since the MainAgent is never used. The Mobile Agent travels through the whole network in a random order to make the distributed intrusion decision. In Method_6, a system with high reliability and short detection time is created by combining the short mean detection time advantage of Method_2 with the fully-distributed structure of Method_5. This system works in mode-1(Method_2) normally but shifts to mode-2 (Method_5) when the central unit becomes unavailable.

The objective of the proposed methods is alerting a security manager to take necessary precautions before the last two phases of a DDoS attack can be completed, by detecting the first three phases quickly and correctly. Mean detection times, network loads and reliability properties of each method are compared over twenty simulation runs of the system. It is observed that usage of mobile agents increases the network load, but distributed creation of these mobile agents also increases the independency of the system from the central unit. Messages have smaller sizes

than mobile agents in our current study; however this may be the opposite in an anomaly-based IDS. Method_4 is determined as the best method because of its full-independence from the central unit and its low network load. All of the methods have too close "intrusion detection time" values, so it is not used as a third criterion.

In most of the studies that we cited in the beginning, mobile agents are just used to collect information from the hosts, but the decision about potential intrusion is centralized which constitutes a major weakness. Even in DIDMA [2] which makes the analysis distributedly, mobile agents are created centrally and take the route specified in their lists. In the methods that we propose in this study, mobile agents are created distributedly, move through all of the nodes if necessary, gather the related information, correlate it and make the decision autonomously. The mobile agents can still do their job even in the failure of their dispatchers which makes our system more reliable. Although there are studies about "Distributed Detection of DDoS Attacks" or "Distributed Detection of the Intrusions using Mobile Agents", this is the only study that we know, trying to make the detection of DDoS attacks during the Intermediate Phase. In our work, intrusion alarms are generated before a DoS activity happens in a host that belongs to this phase.

In order to detect DDoS attacks of various types, under various circumstances, signature-based and anomaly-based, network-based and host-based DIDS should be hybridized. We are currently working on developing adaptive and intelligent anomaly-based distributed intrusion detection methods which adapt themselves to new types of attacks to be used with SNORT. Security of the agents and messages should also be considered in future studies.

REFERENCES

[1] Abraham, A.—Grosan, C.—Chen, Y.: Cyber Security and the Evolution of Intrusion Detection Systems. Journal of Educational Technology, Special Issue in Knowledge Management, ISSN 0973-0559, I-Manager Publications, 2005.

[2] Kannadiga, P.—Zulkernine, M.: DIDMA A Distributed Intrusion Detection System Using Mobile Agents. Proceeding of the ACIS 6th International Conference on Software Engineering, Networking and Parallel/Distributed Computing (SNPD/SAWN), 2005, pp. 238–245.

[3] Chandler, J. A.: Security in Cyberspace Combatting Distributed Denial of Service Attack. University of Ottawa Law & Technology Journal, Vol. 1, 2003–2004, p. 231.

[4] Kruegel, C.—Toth, T.: Distributed Pattern Detection for Intrusion Detection. Proceedings of Network and Distributed System Security Symposium Conference (NDSS) 2002.

[5] SNORT – The de facto standard for intrusion detection/prevention. `http://www.snort.org`, accessed in 2008.

[6] Northcutt, S.—Novak, J.: Network Intrusion Detection: An Analyst's Handbook. USA, New Riders Publishing 2002, pp. 235–240.

[7] Aydin, M. A.—Zaim, A. H.—Ceylan, K. G.: A Hybrid Intrusion Detection System Design for Computer Network Security. Computers and Electrical Eng. Journal, Vol. 35, 2009, No. 3, pp. 517–526.

[8] Mahoney, M. V.—Chan, P. K.: PHAD: Packet Header Anomaly Detection for Identifying Hostile Network Traffc. Florida Tech., Technical Report, 2001-04, 2001.

[9] Mahoney, M. V.: Network Traffic Anomaly Detection Based on Packet Bytes. ACM Symposium on Applied Computing (SAC), 2003.

[10] Kayacik, G. H.—Zincir-Heywood, A. N.: Using Intrusion Detection Systems with a Firewall: Evaluation on DARPA 99 Dataset. NIMS Technical Report, 062003, 2003.

[11] Takeda, K.—Takefuji, Y.: Pakemon – A Rule Based Network Intrusion Detection System. Int. Journal of Knowledge-Based Intelligent Engineering Systems, Vol. 5, 2001, No. 4, pp. 240–246.

[12] Cisco IOS Firewall Intrusion Detection System. `http://www.cisco.com/en/US/docs/ios/12_0t/12_0t5/feature/guide/ios_ids.html`.

[13] Lin, S.: A Survey on Solutions to Distributed Denial of Service Attacks. Research Profficiency Examination Report, TR-201, Experimental Computer System Lab, SUNY at Stony Brook, September 2006.

[14] Peng, T.—Leckie, C.—Ramamohanarao, K.: Detecting Distributed Denial of Service Attacks Using Source IP Address Monitoring. Draft, November 2002.

[15] Foukia, N.: IDReAM: Intrusion Detection and Response Executed with Agent Mobility Architecture and Implementation. Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '05), 2005, pp. 264–270.

[16] Xu, C.-Z.: Naplet: A Flexible Mobile Agent Framework for Network-Centric Applications. Proceedings of the 16th International Parallel and Distributed Processing Symposium (IPDPS 2002), pp. 219–222.

[17] Ramachandran, G.–Hart, D.: A P2P Intrusion Detection System Based on Mobile Agents. Proceedings of the 42nd Annual Southeast Regional Conference (ACM-SE 42), 2004, pp. 185–190.

[18] Bellifemine, F.—Caire, G.—Poggi, A.—Rimassa, G.: JADE – A White Paper – EXP in Search of Innovation – Special Issue on JADE. TILAB Journal 2003.

[19] Bellifemine, F.—Caire, G.—Greenwood, D.: Developing Multi Agent Systems with JADE. John Wiley and Sons Ltd. 2007, pp. 1–34.

[20] Mahoney, M. V.—Chan, P. K.: An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection. Proceedings of Recent Advances in Intrusion Detection, 6th International Symposium (RAID 2003), Pittsburgh, PA, USA, 8–10 September 2003, pp. 220–239.

[21] Brugger, S. T.–Chow, J.: An Assessment of the DARPA IDS Evaluation Dataset Using SNORT. UC Davis Technical Report CSE-2007-1, Davis, CA, 6 January 2007.

[22] MIT Lincoln Laboratory, Information Systems Technology, `http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/2000/LLS_DDOS_1.0.html`, accessed in 2009.

[23] Korba, J.: Windows NT Attacks for the Evaluation of Intrusion Detection Systems. Master of Engineering thesis, MIT, USA, May 22, 2000.

[24] Du, Y.—Wang, H.-Q.—Pang, Y.-G.: IADIDS – Design of a Distributed Intrusion Detection System Based on Independent Agents. Proceedings of International Conference on Intelligent Sensing and Information Processing, 2004, pp. 254–257.

[25] Bernardes, M. Y.—Moreira, E. dos S.: Implementation of an Intrusion Detection System Based on Mobile Agents. Proceedings of the International Symposium on Software Engineering for Parallel and Distributed Systems (PDSE 2000), pp. 158–164.

[26] Shao-Chun, Z.—Qingfeng, S.—Xiao-Chun, C.—Yan, W.: Safe Mobile Agent System For Distributed Intrusion Detection. Proceedings of The Second International Conference on Machine Learning and Cybernetics, November 2003, pp. 2009–2014.

[27] Jaayeri, M.—Lugmayr, W.: Gypsy: A Component-Based Mobile Agent System. Proceedings of the Eighth Euromicro Workshop on Parallel and Distributed Processing (EURO-PDP 2000), Rhodos, Greece, January 19–21, 2000.

[28] Asaka, M.—Okazawa, S.—Taguchi, A.—Goto, S.: A Method of Tracing Intruders by Use of Mobile Agents. Proceedings of the 9th Annual Conference of the Internet Society (INET '99), June 1999.

[29] Recursion Software, Products, Voyager Edge, Solutions for Intelligent Mobile Applications, Rapid Development, and Social Networking. `http://www.recursionsw.com/Products/voyager.html`, accessed in 2008.

[30] Akyazi, U.—Uyar, A. S.: Distributed Intrusion Detection Using Mobile Agents Against DDos Attacks. 23rd International Symposium on Computer and Information Sciences (ISCIS), IEEE 2008, ISBN: 978-1-4244-2880-9, DOI:10.1109/ISCIS.2008.4717920.

Ugur Akyazi received his B. Sc. degree in 1999 from Computer Engineering Department of Turkish Air Force Academy, and his M. Sc. degree in 2002 from Systems Engineering program of Air Force Institute of Technology (USA). He then received his Ph. D. degree in 2011 from Computer Engineering Program of Istanbul Technical University. He worked as a teaching assistant in Computer Engineering Department of Turkish Air Force Academy between 2002 and 2011. He is still a student of Turkish Air War College. His research interests include nature-inspired computing and evolutionary algorithms especially in computer network security.

A. Sima Uyar received her B. Sc. degree in 1990 from the Control and Computer Engineering Department of Istanbul Technical University. She then received her M. Sc. (in 1992) and Ph. D. (in 2002) degrees from the Control and Computer Engineering program of the Institute of Science and Technology in the same university. She worked as a teaching and research assistant in the Computer Engineering Department of Istanbul Technical University between 1990 and 2003. She is an Assistant Professor in the same department since 2003. Her research interests include nature-inspired computing and evolutionary algorithms especially in dynamic/noisy/uncertain environments and applications in data clustering, learning and combinatorial optimization.