AN ADAPTIVE INTRUSION DETECTION AND DEFENSE SYSTEM BASED ON MOBILE AGENTS

AN ADAPTIVE INTRUSION DETECTION AND DEFENSE SYSTEM BASED ON MOBILE AGENTS

M. Eid, H. Artail, A. Kayssi, A. Chehab

Department of Electrical and Computer Engineering,
American University of Beirut
P.O.Box 11-0236 Beirut 1107 2020 Lebanon.
E-mail: {mae33, hartail, ayman, chehab}@aub.edu.lb
Correspondence Email: hartail@aub.edu.lb

ABSTRACT

This paper presents a distributed intrusion detection system (IDS) based on mobile agents that detect intrusion from outside the network segment as well as from inside. A main machine, being a typical intrusion detection system residing at a secure location, creates agents and dispatches them into the network. On each hop, the agents sniff the network traffic and look for abnormal activities by matching against a limited rule set supplied by the main machine. The agents are programmed with enough intelligence to decide whether to send the logged data (captured packets) to the main machine for further analysis. The proposed model comprises three major components: the Network Intrusion Detection Component, the Mobile Agent Platform, and distributed intelligent mobile agents called mobile IDS agents. Finally, we present partial results obtained from an early prototype and a discussion of design and implementation issues, and directions for future work.

Keywords: Mobile agents, intrusion detection, distributed systems.

1. INTRODUCTION

Computer networks, including the world wide Internet, have grown in both size and complexity. The services they offer rendered them the main means to exchange data and an optimal environment for e-businesses. Unfortunately, they have also become the means to attack hosts and legitimate users. The growing importance of network security is shifting security concerns towards the network itself rather than being host-based. Security systems will soon evolve into network-based and distributed approaches to deal with heterogeneous platform technologies and support scalable solutions.

Among all security issues, intrusion is the most critical and widespread. Intrusion can be defined as an attempt to compromise, or otherwise cause harm, to a network or a host. An Intrusion Detection System (IDS) is responsible for handling the detection tasks using footprints or signatures of malicious activities. In addition to identifying attacks, the IDS can be used to identify security vulnerabilities and weaknesses, enforce security policies, and provide further system auditing by exploiting the logs/alerts from the output component of the IDS.

Of a particular interest, mobile agents are intelligent programs that function continuously and are able to learn, communicate and migrate by themselves from host to host to gather information and perform specific tasks on behalf of a user [1]. Mobile agent technology offers the potential to overcome a number of limitations intrinsic to existing IDSs that employ only static components [2]. The advantages of using mobile agents for intrusion detection include overcoming network latency, reducing network load, performing autonomous and asynchronous execution, and adapting to dynamic environments. Moreover, implementation of mobile agents in languages such as JAVA provides mobile agents with system and platform independence and considerable security features [3].

Current commercial IDSs such as Cisco Secure IDS [4] and NetDetector [5] deploy one monitoring station that snorts on one link in the network so it misses much of the traffic exchanged between local

hosts that may be critical for the identification of invasions initiated by hosts from within the network segment. Other systems such as the one described in [6] comprise static sensors distributed over the network and a centralized management station. Transmission of Log data to a main station causes many bottlenecks and inefficient use of the network bandwidth. In addition, there is a latency associated between the intrusion event and the detection time. Moreover, if the main processing unit fails, intruders gain considerable access to the whole network.

The presented system in this paper addresses many problems facing current IDSs. First, the approach provides a highly-distributed IDS with a minimum amount of traffic generated over the network. There are mobile processing units to capture and analyze relevant data asynchronously and independently from the main machine. Second, mobility makes the IDS highly secure against attacks targeting the IDS itself. Third, roaming the internal network, agents are capable of detecting attacks from within the network. Fourth, agents may be programmed to have sufficient decision making intelligence. They decide how to respond to the detected attacks based on the severity level of the attack. Fifth, the set of signatures that the agent uses in the detection process is dynamic, meaning that it evolves in real-time depending on the feedback from the main machine. Sixth, the system is shown to be highly adaptive since population of IDS agents increases during attack times and decreases during peaceful states.

In the next section, we present a literature review of previous work in the domain of mobile agent-based intrusion detection systems. In section 3, we describe the approach, partial results obtained from an early prototype. Section 4 provides a discussion of design and implementation issues. Finally, section 5 presents a conclusion and directions for future work.

2. LITERATURE REVIEW OF PREVIOUS WORK

Historically, the intrusion detection technology dates back to 1980 [7]. It became a well-established research area following the introduction of the model in [8] and the prototypes presented in [9] and [10]. These systems were centralized. A single machine monitors data flow at a strategic point in the network and collects and analyzes data from the log files. Once an attacker destabilizes this host, he or she is able to gain considerable access to the whole network. This limitation, we believe, is the main vulnerability of currently implemented centralized IDSs.

Distributed IDSs were introduced to overcome this susceptibility. The approach in [11] proposes an architecture for a distributed intrusion detection system based on multiple independent entities called Autonomous Agent for Intrusion Detection (AAFID) framework. The proposed system allows data to be collected from multiple sources, thus combining traditional host-based and network-based IDSs. Several problems face this framework including scalability, performance, security, and user interface. In a similar approach, a mobile agent-based architecture and model consists of a large number of small mobile agents that perform the tasks of monitoring, decision-making, notification and reaction to attempted intrusions [12]. New specialized agents can be added whenever a new form of attack is identified or removed dynamically from the system.

Subsequent work like that portrayed in [13], [14], or [15] presents a fully distributed architecture where data collection and information analysis are performed locally without referring to the central management unit. For instance, the designed architecture in [15] comprises two components: IDS agents and a stationary secure database (SSD). IDS agents are stationary and participate in cooperative algorithms to decide if the network is being attacked. The SSD acts as a trusted database for the agents to obtain latest misuse signatures.

Another architecture for an entirely distributed IDS based on multiple independent entities working collectively is discussed in [16]. These entities are called Autonomous Agents. The approach was claimed to solve some of the problems in existing commercial IDSs associated with centralization,

configurability, and scalability. Computation is performed (and thus intrusion detection) at any point where sufficient information is available. These systems use network resources inefficiently.

3. SYSTEM ARCHITECTURE

This section presents the architecture of our distributed IDS. We detail the inner components, and then illustrate the role of each with an example. The architecture is comprised of the following components: (1) a main intrusion detection processor, (2) a mobile agent platform, and (3) distributed IDS mobile agents. A high level view of the architecture is given in Figure 1.

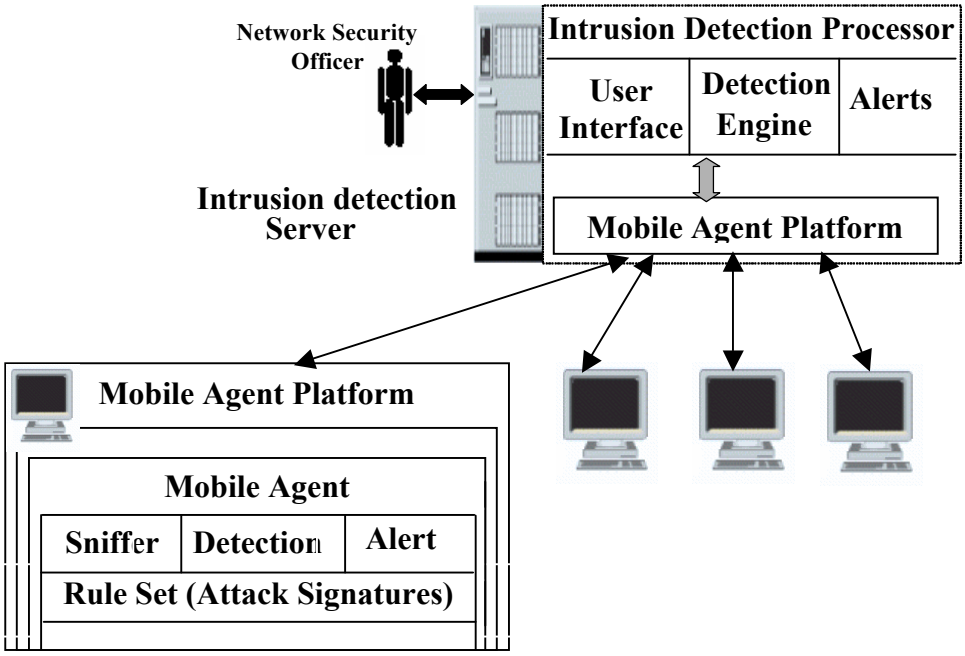

Figure 1: General Architecture for the System.

3.1 Main Intrusion Detection Processor (MIDP)

This component is the cornerstone of our distributed framework. It is responsible for monitoring network segments (subnets), and acts as central intrusion detection and processing unit. Its main capabilities are: acting as a cross-relating unit for multiple logs sent by dispatched agents, providing and updating rule sets and severity lists for each of the agents, and interfacing the IDS to the system administrator.

Basically, IDP can be considered as a typical IDS. once logs are collected, the raw data is linked to structures for analysis by the detection engine. The detection engine processes the captured packets by checking them (the header and/or the content of the packet, depending on the security level) against a set of rules. If the rules match the data in the packets, then alerts are triggered and written into the output alert files and responses are sent to both the user interface and the dispatched agents.

A major function of the IDP is the collection and correlation of IDS data from distributed IDS mobile agents. The main objective here does not lie in identifying intrusions; rather, it is in the linkage of events across a network, providing organizations with a heuristic analysis of combined data or even a static-state assessment of correlated intrusions from separate IDS mobile agents.

The IDP acts as a secure, trusted repository for the mobile agents to obtain latest information about attacks that they should look for and to update their severity lists. Attached to the IDP is a database

that contains two types of information: attack traces or signatures (rule set) and severity level associated with each attack (severity list). Severity lists defines the response mechanism that agents should use when attacks are detected. For instance, level 1 (most severe attack) means that agents should send all logged network traffic plus the alarm file, and so on. The complete list is presented in table 1.

Table 1: List of Severity levels

Severity Level	Description
1	The IDS agent should send the complete log file in one batch plus all the alerts generated by the IDS agent.
2	The IDS agent should send a summary of the log file plus all the alerts generated by the IDS agent.
3	The IDS agent should send only the alert file while saving the dump file at the current host.
4	The IDS agent should inform the main station about the attack while saving the dump file at the current host.
5	The IDS agent ignores the attack while saving the dump file at the current host.

3.2 Mobile Agent Platform

The mobile agent platform (MAP) can create, interpret, execute, transfer, and terminate/kill agents. The platform is responsible for accepting requests made by network users (in our case the IDP) and generating IDS mobile agents plus dispatching them into the network to do intrusion detection functions. The platform is a small server program that will reside in each host within the network and will be responsible for accepting and deleting mobile agents.

3.3 Mobile IDS Agent

Each subnet in the network will have a mobile IDS agent roaming among all its hosts at all times. This agent is responsible for detecting intrusions based on data gathered by sniffing on the network traffic. In general, sniffing is used for: (1) network analysis and troubleshooting, (2) performance analysis and benchmarking or, (3) eavesdropping for clear-text passwords and other interesting tidbits of data. Each IDS agent is "armed" with a light detection engine that enables it to detect most well-known attacks. Once a host receives a mobile IDS agent, the latter's first job is to create a thread and start sniffing and dumping into a log file. In some situations, the agent may accelerate or decelerate the collection rate as the IDP deems necessary. The log file is created in a share mode so that detection can proceed in parallel with sniffing to avoid detection latency. The agent starts the detection process on a separate thread so that sniffing and detection engine use the same file, one to read from while the other to dump into. Once an intrusion is caught, the agent checks its severity level and responds accordingly. The agent moves to its next hop if no severe attacks take place at the current host during a certain time interval, and the cycle starts over.

3.4 How does it work?

First, let us describe the initial state of the system as it is shown in figure 2. On every device in the local network, the MAP is installed to host mobile agents. The MAP that resides on these devices has the necessary information about the directories at which log and alert files should be saved.

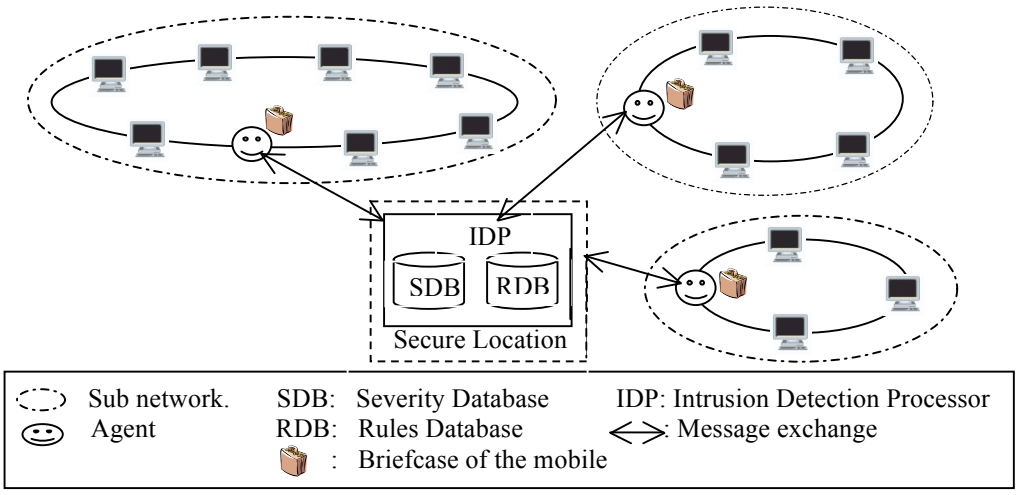

Figure 2: System at the Idle State

Scenario 1: This is the idle state of the system where no intrusions are detected. When the system is initially started, the IDP sends a START request to the MAP. The message specifies the number of agents to be launched and the corresponding IP address sets that each agent is expected to visit. This implies that the IDP has a registry containing all IP addresses in the local network. The MAP, in turn, creates the agents and dispatches them into the network.

Once an agent is received by the first host on its list, it starts sniffing and detecting processes. After a while, the agent informs the MAP on the host where it is running about its willingness to move to another host. This MAP copies the agent's code to the destination host via the MAP residing there (which will take care of running it on the new host) and then deletes the code after shutting down its process. This, in effect, moves the agent to the destination host. In this scenario, the agent population remains constant.

Scenario 2: Now, assume that an agent on its trip catches an attack that triggers an alarm. The agent checks for the severity level of the attack and responds accordingly. If the attack is severe, the agent sends the logged data as well as the alert file to the main machine and creates a clone for itself. The agent assigns half its visiting list to its clone and keeps the second half for it. Thus the agent population starts to increase when continuous attacks are launched against the network. The agent clones roam the assigned segments and may clone themselves when they detect new attacks till we reach the all-Snort state. The all-Snort state is reached when every host has a mobile IDS agent running all the time (equivalent to running snort on every host).

After a while where no attacks are detected, the clones start to dispose. Every clone should send a message to the main machine to request disposal. The main machine determines the parent agent of the clone and sends it a "Visiting_list_update" message to handover the child segment. After the parent sends an acknowledgment to the main machine that it's visiting list is successfully updated, the main machine sends the child agent an acknowledgment message for disposal.

4. IMPLEMENTATION

In this section, the initial implementation of the prototype distributed IDS is discussed. Then results about the performance of the system are presented.

4.1 IDS Implementation

The prototype IDS has been implemented on top of Snort [17] and the well known mobile agent system, Aglets. Each agent carries with it a lightweight snort engine [18] that detects local intrusions in "semi-real time", while a full fledged snort engine is installed at the main station that performs in depth analysis of log files and controls the behaviors of the agents plus their routes accordingly.

4.2 Mobile Agent System (Aglets)

The Aglets Software Developer Kit (ASDK) is developed at the IBM Research Laboratory in Japan [19]. It is not a commercial product and its latest version is 2.0.2 (also called Aglets (Java 2) now). It is entirely written in Java and has been released in Feb 2002 and can be downloaded for free from [20] for noncommercial purposes. Aglets was chosen as the mobile agent platform because of its availability, ease of running, reliable messaging, dynamic routing on agent itinerary, and support for mobile agents.

The ASDK runtime consists only of one aglets server that has a daemon process to handle the incoming and outgoing aglets over the network. The elements of an Aglets system include context, proxy, and aglets [21]. The aglet context is the workspace for aglets. It allows maintenance and management of running aglets and also provides the security management. An aglet is a mobile agent that can migrate between aglet-enabled hosts and runs on its own thread. An aglet interacts with other aglets or objects through its proxy, which on the programming level is a public interface for the aglet. An aglet is protected from direct access to its public methods by communicating through a proxy. Moreover, proxy is a means by which location transparency is achieved.

4.3 Light Weight Snort

Snort is a full-fledged; open-source network based IDS (NIDS) that has many capabilities such as packet sniffing and packet logging in addition to intrusion detection. Snort is a signature-based IDS that uses rule sets to check for errant packets crossing a node in the network. A rule is a set of requirements that would trigger an alert. Light-weight snort [18] runs against a limited rule set. It was selected in the prototype because of its lightweight, popularity, portability, support of multiple operating systems, configurability, and the availability of multiple output options (alerting to syslog, file, or win popup).

4.4 Discussion and Results

To study the feasibility of our security architecture, we have conducted a series of experiments to evaluate its effectiveness. Our experiments have shown that the system can rapidly reach the all-Snort state when continuous attacks are launched against the home network, and return to the idle state when the attacks are terminated.

Figure 3 presents the prototype network that we used to proof-concept our work. The network comprises a main station and forty Windows hosts. Aglet system runs at every host. The forty computers are connected via 2 switches, simulating switched network. The system is evaluated using a mix of normal traffic and four attacks. The array of attack types is described in Table 2. The normal traffic is generated using the WINJET packet generator [22] and attacks are simulated using the BLADE Software [23]. Miss rates and false positive rates were measured against the residence time of the agent. The attacks were launched manually and randomly and the averages of 10 runs were recorded.

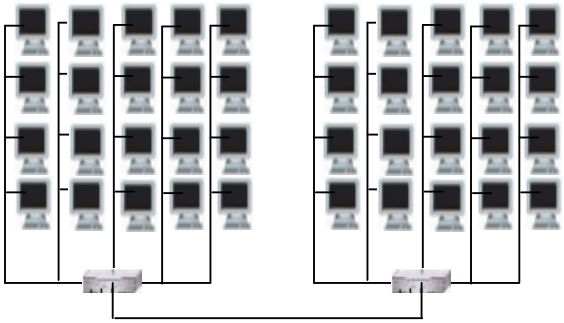

Figure 3: The Sample Network

Table 2: Simulated attacks and their description.

Attack Name	Description
Backdoor Back Orifice	A remote administration tool that allows almost complete control over a computer by the remote attacker.
Finger User	Allows an attacker to disrupt your network using the redirection capability in the finger daemon.
RPC Linux Statd Overflow	Buffer overflow vulnerability exists making it possible for malformed requests by an attacker to be devised giving root privileges.
DNS Zone Transfer	DNS server provides information for all DNS resource records registered with DNS server that can be used by attackers to better understand a network.

An experiment was conducted to determine the number of start-up agents (N) relative to the tested network. The results are plotted in figures 4, 5, and 6 where the number of start-up agents is plotted against the false positive rates, false negative rates, and CPU times consumed by agent's execution respectively. It is obvious that as N increases the false positive and false negative rates decreases while the CPU time increases. Therefore, there is a tradeoff between network resources and security level where highest security level corresponds to all-Snort state, while least security level corresponds to one agent serving the whole network.

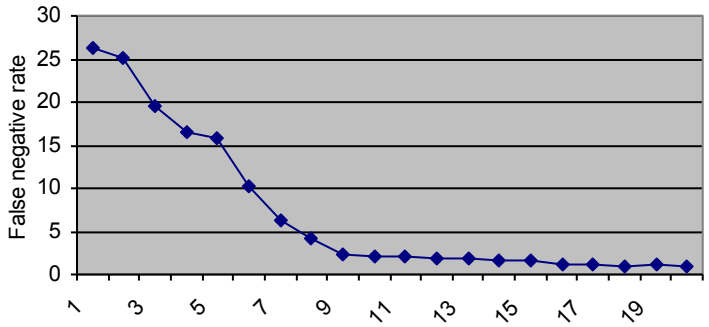

Figure 4: Number of Start-up Agents

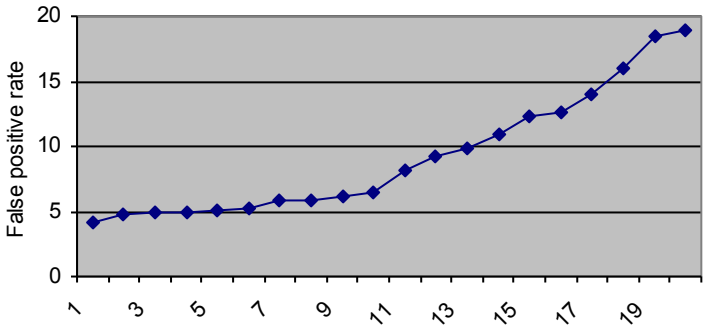

Figure 5: Number of Start-up Agents

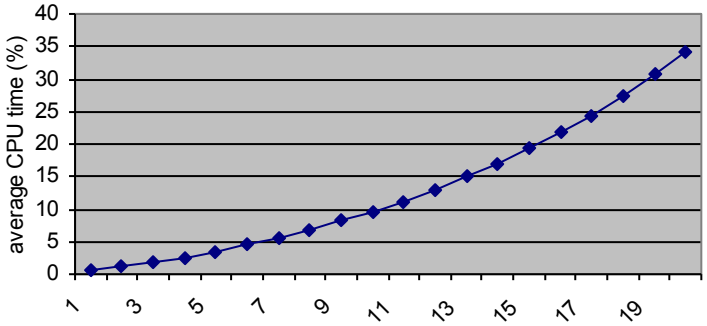

Figure 6: Number of Start-up Agents

Figure 7 shows the false positive rates and missed attacks rates versus the agent residence period (T) for the four types of attacks. The figure shows that as the residence time increases the number of missed attacks decreases since agents will start to spend enough time at a host to capture attacks. When the residence time increases more that necessary, the agent will be wasting time on other hosts while attacks are taking place on others. For this reason, the number of missed attacks starts to increase again.

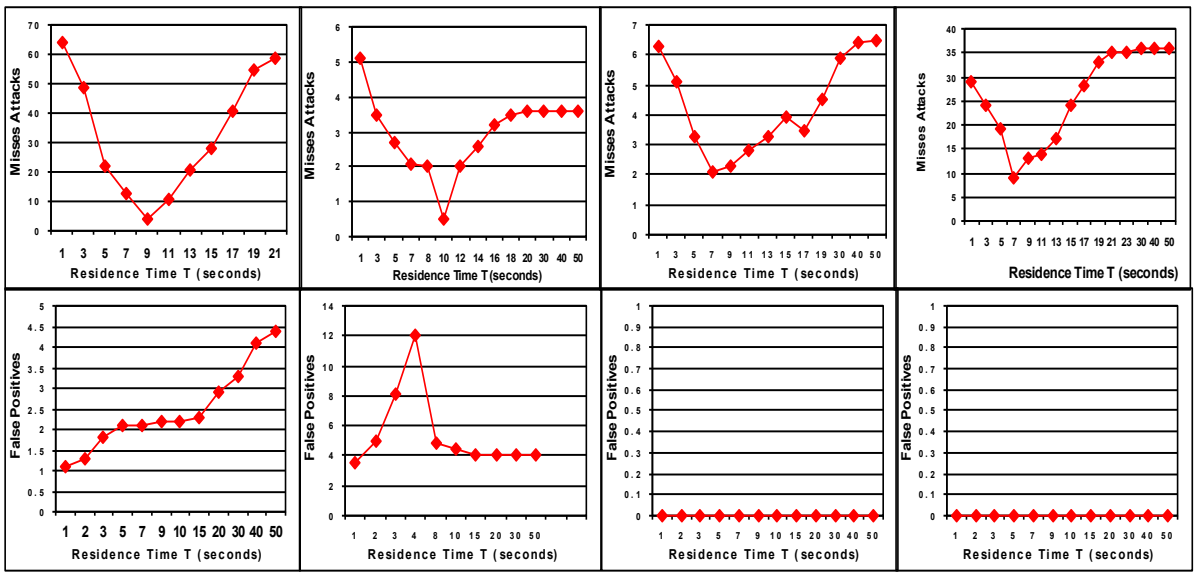

Figure 7: Missed attacks and false positives versus residence time. (a) Backdoor Back Orifice. (b) DNS Zone Transfer. (c) Finger User S. (d) RPC Linux Statd Overflow.

Finally, to measure the response time of the system caused by the cloning strategy, we performed the following test. We run the system and checked the agent population during continuous attacks as function of time. The results are shown in figure 8. We notice that as time increases the agent population increases rapidly until we reach the all-Snort state. This means that the system can rapidly adapt to its environment. After stopping the attacks, the initial state was retained in a reasonable time.

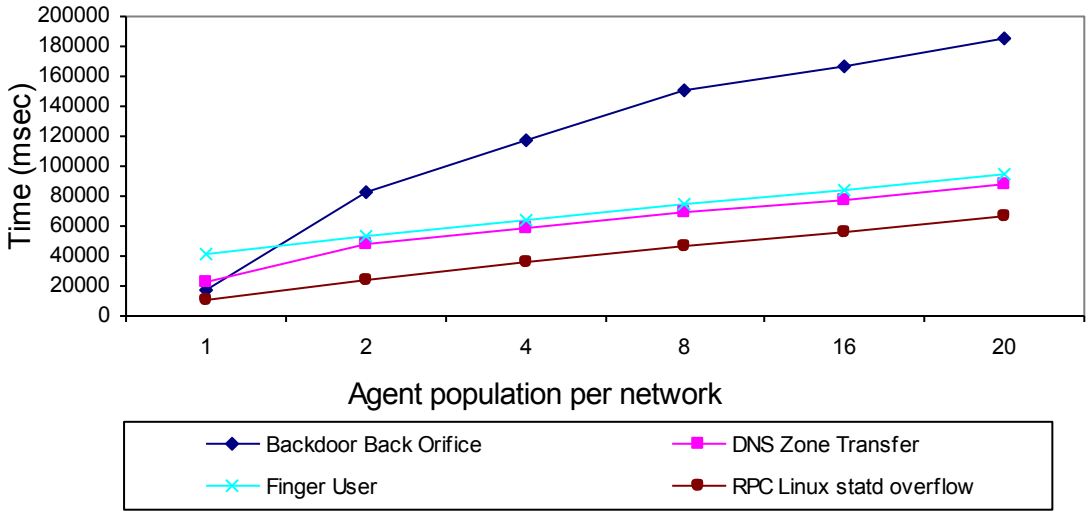

Figure 8: Agent population as function of time during continuous attack.

CONCLUSION AND FUTURE WORK

Inspired from real life situations where policemen roam city streets looking for dangerous people and when suspecting something wrong, they watch and follow more closely, we present an architecture for distributed intrusion detection based on mobile agents that follows a similar approach. The system is shown to be highly adaptive and saves network resources during "no attacks" times. An expansion of the distributed IDS seems to be possible using response and immunity components. Automating the response mechanisms decreases the time window an attacker has before being encountered by a human. The initial number of agents that the system should start with needs further investigation to determine the optimal number of hosts per agent. The security issue of the system has not yet been tackled, and thus should be part of our future work.

REFERENCES

[1] S. Fuenfrocken. "How to Integrate Mobile Agents into Web Servers", Sixth IEEE workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, Cambridge, MA USA, June 1997.

[2] W. Jansen, P. Mell, T. Karygiannis, and D. Marks. "Applying Mobile Agents to Intrusion Detection and Response", National Institute of Standards and Technology (NIST) Interim Report (IR) - 6416. ACM, October 1999.

[3] S. Fuenfrocken. "Integrating Java-based Mobile Agents into Web Servers under Security Concerns", Proc. of the Thirty-First Hawaii International Conference on System Sciences, Jan. 1998.

[4] "Technical overview of system operations for the Cisco Secure Intrusion Detection System" (formerly NetRanger), Cisco Systems, Inc. 2000.

[5] "The Most Advanced Network Security and Forensics Analysis System", NIKSUN, Inc. NetDetector White Paper, 2001.

[6] R. Gopalakrishna and E. Spafford. “A Framework for Distributed Intrusion Detection using Interest Driven Cooperating Agents”, Purdue University, 2001.
[7] J. Anderson. “Computer Security Threat Monitoring and Surveillance”, Technical report, James P Anderson Co., Fort Washington, PA, Arpil 1980.
[8] D. Denning. “An intrusion-detection model”, Proc. of the IEEE Symposium on Security and Privacy, pages 118-131, 1986.
[9] D. Bauer and M. Koblentz. “NIDX – an expert system for real-time network intrusion detection”, Proc. of the Computer Networking Symposium, pages 98-106, Washington, DC, April 1988.
[10] R. Schoonderwoerd, O. Holland, and J. Bruten. “Ant-like agents for load balancing in telecommunications networks”, Proc. of the first International Conference on Autonomous Agents, 1997.
[11] J. Balasubramaniyan, J. Garcia-Fernandez, D. Isacoff, E. Spafford, and D. Zamboni. “An Infrastructure for Intrusion Detection using Autonomous Agents”, COAST technical Report 98/05, June 11, 1998.
[12] M. Bernardes, E. Moreira. “Implementation of an Intrusion Detection System Based on Mobile Agents”, Proc. of the International Symposium on Software Engineering for Parallel and Distributed Systems, 2000.
[13] G. White, E. Fisch, and U. Pooch. “Cooperating security managers: A peer-based intrusion detection system”, Network, IEEE, Volume: 10, Issue: 1, 1996.
[14] J. Barrus and N. Rowe. “A distributed autonomous-agent network-intrusion detection and response system”, Proc. of the 1998 Command and Control Research and Technology Symposium, 1998.
[15] A. Smith. “An Examination of an Intrusion Detection Architecture for Wireless Ad Hoc Networks”, 5th National. Colloquium for Information System Security Education, May 2001.
[16] J. Balasubramaniyan, J. Garcia-Fernandez, D. Isacoff, E. Spafford, and D. Zamboniy “An Architecture for Intrusion Detection using Autonomous Agents”, Proc. of the Computer Security Applications Conference, 1998.
[17] Snort website: www.snort.org (Accessed in March 15, 2004).
[18] M. Roesch. “Snort - Lightweight Intrusion Detection for Networks”, A white paper on the design features of Snort 2.0 from: www.sourcefire.com/technology/whitepapers.html (Accessed in January 15, 2004).
[19] http://www.trl.ibm.com/aglets/index_e.htm (Accessed in April 1, 2004).
[20] http://sourceforge.net/projects/aglets/ (Accessed in April 1, 2004).
[21] D. Lange, and M. Oshima. “Programming and Deploying Java Mobile: Agents with Aglets”, Addison-Wesley, 1998.
[22] Drugs for Windows Website: http://home19.inet.tele.dk/moofz/index_o.htm. (Accessed in April 1, 2004).
[23] BLADE Software Website: http://www.bladesoftware.com/IDSInformer.htm. (Accessed in April 10, 2004).