A Self-Extension Monitoring for Security Management

Heejin Jang and Sangwook Kim
Department of Computer Science, Kyungpook National University
1370, Sankyukdong, Bukgu, Daegu, 702-010, KOREA
{janghj, swkim}@cs.knu.ac.kr

Abstract

In the coming age of information warfare, information security patterns take on a more offensive than defensive stance [1]. However, most existing security systems remain passive and do not provide an active form of security protection. It is necessary to develop an active form of offensive approach to security protection in order to guard vital information infrastructures and thwart hackers. This paper presents a Self-Extension Monitoring, a new approach in monitoring intruders, securing evidence against hackers and identifying them. It also proposes an Intruder Identification System (IIS), which is designed and implemented based on the proposed technique. The Self-Extension Monitoring approach minimizes temporal and spatial limitations, making it possible to collect enough information for disclosure of the intruder's identity. A system security administrator can prevent any unwanted intrusion and re-attack the intruder by creating and maintaining information regarding the intruder's identity through the Self-Extension Monitoring.

Keywords*: Monitoring, Security, Intruder Identification, Replication, Shadowing*

1. Introduction

Considerable research efforts have been exerted to develop a better intrusion detection method. However, detection alone is not sufficient to prevent the current threat of abuse amid the rapidly growing network. Therefore, an intruder identification system technique is required to identify the hacker and prevent him or her from making any effective intrusion. A difficult aspect of the intruder identification is that it is nearly impossible to promptly identify the intruder because the act is carried out from a remote host over which we have no control. Leaving the intruder on-line causes security problems. Hence, an effective intruder identification technique is required to examine the behavior of the intruder and collect the pertinent information for an extended time without affecting the real system.

An intruder can be identified through retracing by means of analyzing the log file as well as monitoring the intruder at the host/network level. The log-based retracing technique analyzes log files provided by the system and certifies the name and/or address of the intruder's system. It gives intruders enough time to cover up their traces because the system security administrator has to analyze the log files manually. It is also difficult to retrace intruders and analyze the damage on a system because the analysis normally depends on the experience of a system security administrator. The Caller Identification approach is one where a Caller Identification client sends information regarding a certain user's network movement path to a Caller Identification server when logging into the system via the network[2]. Since each Caller Identification server is installed in its respective host, this technique is in fact impracticable. In the case of the system monitoring technique, it is impossible to collect further information when the intruder moves beyond the monitoring range. Thus, the network monitoring technique is at a disadvantage since it cannot provide information through real-time analysis.

To identify the intruder accurately, the following problems must be solved beforehand. A specified period needed to trace the intruder must first be ensured. That is, it requires the means to inspect the intruder's activities without an intruder recognizing the monitoring. It likewise needs to minimize restriction of the traceable domain. Even if the intruder moves to another host, chasing after the intruder continues.

This paper presents the Self-Extension Monitoring approach as a solution for meeting the two requirements described above. The Self-Extension Monitoring observes the intruder's activities at the host level. If the intruder moves into another host, network level monitoring is carried out through program replication into the host as needed. This broadens the traceable domain and also secures time for investigating the intruder by protecting the host/network itself. That is, it is possible to trace the intruder, increasing the time and space using the Self-

Extension Monitoring based on the shadowing and replication mechanism.

Preservation of evidence of intrusion paths and malicious activities prohibits possible further intrusion. Analyzing the habits and paths of an intruder for a long time makes it possible to generate a hacking scenario database more easily. System security administrators on the intruder's path can then investigate together and eventually identify the intruder.

The remainder of this paper is structured as follows. Section 2 describes the existing approaches to monitoring intruders. Section 3 presents the Self-Extension Monitoring for security management using a shadowing mechanism. Section 4 describes the architecture and working model of the Intruder Identification System designed and implemented in the approach proposed in this paper. It presents an example of this system wherein the intruder's shifting path and his/her real-time activities are obtained. Finally, section 5 draws some conclusions and outlines directions for future research.

2. Monitoring Approaches for Security Management

There are several monitoring approaches that have been developed for security management. Their primary objectives include system/network management or protection, intrusion detection and intruder identification. They are largely classified into two groups namely, host-level monitoring and network-level monitoring[3].

2.1 Host-level Monitoring

It is the host-level monitoring that observes the specified user on a single host and records the log. Movement to an unspecified host makes monitoring impossible. The tty hijacking method is used to monitor the user at the host level. Examples of host-level monitoring tools are ttywatcher[4] and ttymon[5] for the UNIX system and linspy[6] for the Linux.

The ttywatcher is a utility that serves to monitor and control users on a single system. It allows the system security administrator to monitor every tty on the system, as well as interact with them. When the user logs into a computer system, the system authenticates him or her, provides a tty and then runs a shell process. While the system runs ttywatcher, it is then placed on the higher level of the shell. Thus, the user loses control over his/her own terminal as well as over all input streams from the terminal. Communication between the user and the Unix system is established via tty, and the ttywatcher creates a kernel driver. It eavesdrops in the communication between all login ttys in the system and the kernel using the kernel driver. An administrator can see all the input from *stdin* and output to *stdout* through the kernel driver. The driver communicates with the kernel by creating a c-type tty file on the */dev* directory. Since *other* or *group* is not given a read/write permission above the tty file, the ttywatcher eavesdrops on the other tty with root privilege. The ttywatcher is useful as a hacker trap for a user with an illegal account or for intruders. Aside from monitoring and controlling ttys, individual connections can be logged into either a raw log file or to a text file. These files can then be adopted as evidence. Because host-level monitoring does not have any means to protect the monitoring activity, an experienced intruder becomes aware that his/her activities have been exposed.

The ttymon is a stream-based tty port monitor. It monitors ports, sets terminal modes and line disciplines for the ports. It also connects users or applications to services associated with the ports. The ttymon cannot control the intruder but can monitor the intruder's behavior.

Host-level monitoring, as mentioned above, just records logs to watch users and secure evidence of the intrusion. It is nearly impossible for a system security administrator to access each host on a large network to collect data. Although it is possible, it is difficult to decide on which host a target resides and which activity can be logged by the host-level monitoring system itself. Therefore, host-level monitoring cannot effectively protect against intrusion via several hosts. Since it does not shield the monitoring activity itself, the intruder easily notices the fact that he/she is being watched. It also cannot obtain information any more if the intruder does not access the host with a monitoring server.

2.2 Network-level Monitoring

Because the user monitoring at network level can examine all logons, it is not necessary to access another host.

When the intruder sets up a new connection, the system security administrator just monitors the intruder's connection without accessing the system where the intruder logs in. Network-level monitoring is passive and the system operates as though nothing happened. Monitoring programs such as TCP dump[7], Netlog[8] and SNIF[9] support part of these functions mentioned above. However, all of the packet information on the network is input because they do not specify the logon connection to be inspected. Therefore, it is impossible to filter related data and draw out useful information. It cannot offer data through real-time analysis. It just watches the intruder but cannot take any active measures.

There are several network-level monitoring tools with more functions, and these include IP-watcher[10] on the UNIX system, hunt[11] on the Linux and T-sight[12] on Windows NT. These tools use connection hijacking[13] to monitor and control the user's activities. Connection

hijacking is an offensive attack for monitoring the specified user's activities. It intrudes into the connection between server and client and makes packets pass through the connection hijacking server. It is based on the premise that the system executing connection hijacking is near the target host. Network monitoring, however, has the drawback of having a limited data domain. Sniffing on the broadcast ethernet network is restricted within the traffic traveling on the subnet being monitored. Therefore, the network-level monitoring tool is located on the router nearly outside the network. An encrypted packet among the collected data through network monitoring cannot be read. It is overburdened to sniff the entire network for user identification.

3. Self-Extension Monitoring

This paper presents the Self-Extension Monitoring approach based on the Shadowing Mechanism for monitoring hacking activities. The Shadowing Mechanism is a process for securing enough time for reverse tracing by protecting a monitoring activity as well as acquiring a traceable domain, which in turn is obtained by reproducing itself in the host where the hacker resides. It is possible to gather data for intruder identification through this process.

The Shadowing Mechanism is a module equipped with a self-protection facility, which watches the intruder's tty and monitors them at the host level. Only when a hacker invades into a host, does the monitoring at host level convert to a network level. It is possible to trace the intruder by extending the monitoring domain. We call it the Self-Extension Monitoring based on the Shadowing Mechanism.

The Self-Extension Monitoring approach supports monitoring at host level or network level as needed with the Shadowing Mechanism as the basis. It normally watches the intruder at host level, while shifting of the intruder starts monitoring at the network level. While the intruder behaves illegally on another host, some modules are copied to the target host for host-level monitoring using the acquired identity information. The collected information from each target host is then transferred and analyzed. It is possible to determine the target host of the attack, which is made via several hosts. Executing host-level monitoring allows recording of logs for observing intruders and securing evidence. Once a target host is set, restriction on the data domain diminishes and encrypted data need not be read.

Figure 1 shows the Self-Extension Monitoring. It shows that chasing the intruder's movement path enlarges the monitoring domain.

The Shadowing Mechanism logically consists of monitoring, self-protection and replication.

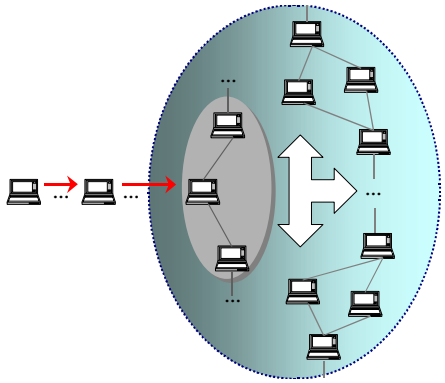

Figure 1 Self-Extension Monitoring

3. 1 Monitoring

While a hacker stays in the host, a Shadowing process begins by monitoring the intruder's terminal discreetly. It observes a user's login tty to monitor that user's activity. The tty is watched via pty, which is dynamically allocated whenever the user logs into the host. The intruder is monitored by copying his/her terminal into the dynamically allocated pty in the Shadowing Mechanism.

3.2 Self Protection

Self-protection in the Shadowing Mechanism means sheltering the monitoring activity from detection of the intruder. This plays an important role in ensuring time to observe the intruder. Trojan horse applications use the execution path mechanism wherein they create the executable file with the same name as the existing program and execute the former instead of the latter. A virus protects itself by changing the execution code.

The Shadowing Mechanism uses a number of techniques to evade detection. It attempts both to cover its tracks and to blend into the normal Linux environment using camouflage. These techniques have various aspects of effectiveness.

The Shadowing Mechanism carries out a number of functions to cover its trail. It erases its argument list after processing the arguments, so that the process status command would not reveal how it is invoked. It also deletes the executing binary, which would leave the data intact but unnamed, and only referenced by the execution of the Shadowing Mechanism. It uses resource limit functions to prevent a core dump. Thus, it prevents any bugs in the program from leaving telltale traces behind. The file /var/adm/utmp(x) shows users who are logging into a computer system. Since the intruder can recognize the fact that he/she is under observation, the mechanism deletes the records. In case of remote login, the user who is executing a program on a pseudo terminal is recorded in

the log file. However, it can record user information or not when the user executes a shell on the pseudo terminal. The Shadowing Mechanism does not necessarily need to protect itself from detection by the normal user or administrator because shadowing is performed through the intruder's process. There are several methods such as erasing a log, system and network, console or shell history, which mainly covers the ordinary user's activities and traces, but not those of the intruder. They can be added if necessary.

In addition to covering tracks, camouflage is used to hide the shadowing. It is compiled under the name sh, the same name used by the Bourne Shell, a command interpreter which is often used in shell scripts and automatic commands. Even a diligent system manager would probably not notice a large number of shells running for short periods of time. It forks, splitting into a parent and child. The parent would then exit, leaving the child to continue from the exact same place. This has the effect of refreshing the process.

It likewise shields itself by replacing an original application program with a modified one. It sends the fake program along with the modules for replication. It can conceal processes using ps, top or pidof and hide files using find, ls or du.

3.3 Replication

The replication protocol in the Shadowing Mechanism operates as depicted in figure 2.

The host, to which the Shadowing Mechanism is initially applied, is called an intruder identification server. If the intruder invades an intrusion identification server, the server starts to monitor an intruder's terminal and gathers the pertinent authentication information and hacking activities. When the intruder moves into another host and begins to get the administrator's privilege in the system, the intruder identification server establishes the connection with the target host and obtains the shell with the administrator's authority by using the acquired authentication information or following the intruder's commands. After the intruder identification server and target host authenticate each other, the server sends some modules to be copied to the target host. The target host informs the intruder identification server that it receives every module to be duplicated perfectly by sending a *resynch* message. The *resynch* message is used to synchronize the server with the target host. If each process is successful, the intruder identification server manages the remote shell directly. It then sends the commands for installing, compiling and running the copied modules to the target host. The target host executes the commands from the server by virtue of initializing replication. The Shadowing Mechanism is set up and starts inspecting the intruder in the target system. The target system performs *resynch* again to notify the intrusion identification server when replication is completed. After executing *resynch*, the intrusion identification server transmits the command to eliminate the compiled program from the target system. It then selects the next target system according to the intruder's movement. Replication in the Shadowing Mechanism is achieved through the above process.

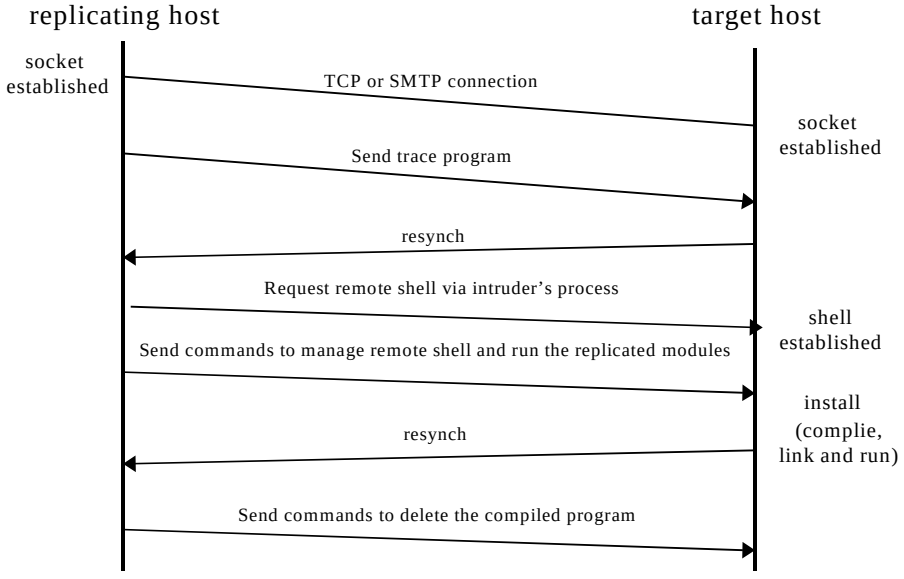

Figure 2 Replication Protocol

4. Intruder Identification System (IIS)

The Intruder Identification System (IIS) [14] is developed on the basis of the Self-Extension Monitoring using the Shadowing and Replication Mechanisms. This system is implemented in the C language for Linux 2.2.12-20.

The IIS obtains the user's authentication or activity information when the Intrusion Detection System defines him as an intruder. This system aims at disclosing the intruder's identity accurately, and is composed of a single server (Intruder Identification Server) and unspecified several clients (Intruder Identification Client). It observes the intruder's activities at the host or network level as needed. All hosts that the intruder goes through become the Intruder Identification Clients. The Intruder Identification Server shadows the user presumed as the intruder and generates the client for identification on the host where the intruder goes. It periodically reports the information gathered about the specified user to the Intruder Identification Server. This section explains the working model and the general architecture of the IIS.

4.1 Working Model

The functions of the IIS include virtual concealment, monitoring, replication mode and investigation mode. The monitoring mode and the replication mode are jointly referred to as a shadowing mode. When the Intrusion Detection System identifies the user as an intruder, it changes into virtual concealment mode [15]. IIS provides a real system to the regular user, and a faked virtual system to the intruder. Therefore, it shields the system from an intruder and confines him or her in a virtual hacking space for an extended period of time for observation and monitoring purposes. When the intruder shifts to another host that is called a target host, basic authentication information and the movement path of the intruder are acquired in the monitoring mode. The IIS starts running the replication and investigation mode using this information. It then duplicates itself to the target host to secure another stronghold in the replication mode. In the investigation mode, it gathers the identity and activity information of the intruder and inspects security risks such as a backdoor. Figure 3 presents an overview of an IIS.

IIS follows the intruder through his/her login tty. If the intruder does not use the tty on the host with the IIS, chasing the intruder becomes nearly impossible. For this reason, it needs to secure another point for identifying the intruder. Therefore, the Intruder Identification Server installs the Intruder Identification Client into the target host when the intruder gets the root privilege by attacking another host. The administrator can configure which modules need to be replicated. Normally, the monitoring, replication and investigation modules are copied to the target host. The Intruder Identification Server continuously

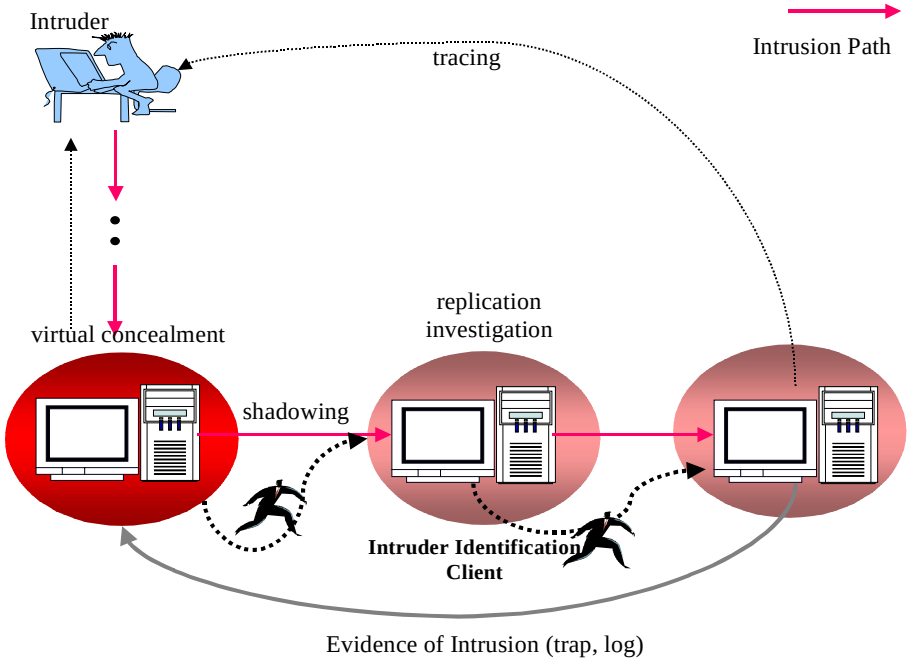

Figure 3 Overview of IIS

manages the replicated Intruder Identification Client.

4.2 Architecture and Implementation

The IIS consists of components for shadowing and investigation modes. On the basis of the Environment Communicator, the upper portion in figure 4 illustrates the shadowing component while the lower portion presents the investigation component.

The Shadowing component is composed of monitoring and replication mechanisms. The replication and investigation components deliver an execution configuration through the Environment Communicator. The Activity Analyzer and Identification Analyzer examine a collection of the intruder's activities as well as identity information and pass them on to the administrator. The Remote Monitor watches the intruder's terminal to collect the activity information. The Activity Filter extracts the useful piece among the information transferred as a result of the monitoring process. The Activity Analyzer then resolves this information. Information about the user's identity is delivered to the Self Replication Module Manager and is used to replicate the module. If it is related to activities, it is saved to the database to generate the hacking scenario database. When the IIS receives information regarding the intruder's identity, the Connection Manager attempts to connect with the target host. The Remote Protocol Manager commands the file transmission and executes it in order to reproduce the investigation module. If the intruder acquires the root privilege, the Connection Manager can start up a root shell through the intruder's new privilege. The Remote Protocol Manager orders the compilation and execution of the investigation module using the shell. The Remote Shell Manager administers the acquired root shell. It then takes actions so that the intruder cannot notice the investigation. The Self-Replication Module Manager receives the start-replication message from the Activity Analyzer and controls the replicated modules of each host.

The Execution Environment Adaptor receives information about the environment from the Self-Replication Module Manager and establishes the environment for the investigation of the intruder. The Attribute Collector, Port Scanner, Pattern Matcher and Common Collector gather information about the identity. The information collected is set against the integrity of the guaranteed data by comparison. The Identification Analyzer processes the collected information on identity and sends it to the administrator.

The IIS supports the web-based user interface. Figure 5 presents the user interface of the IIS. The bottom window in the figure displays the intruder's moving path, the hosts' information on the path and the intruder's activities in real time. The intruder's identification information on each host is shown in the bottom frame of the top window. It is possible to retrieve the intruder's intrusive path, intrusion

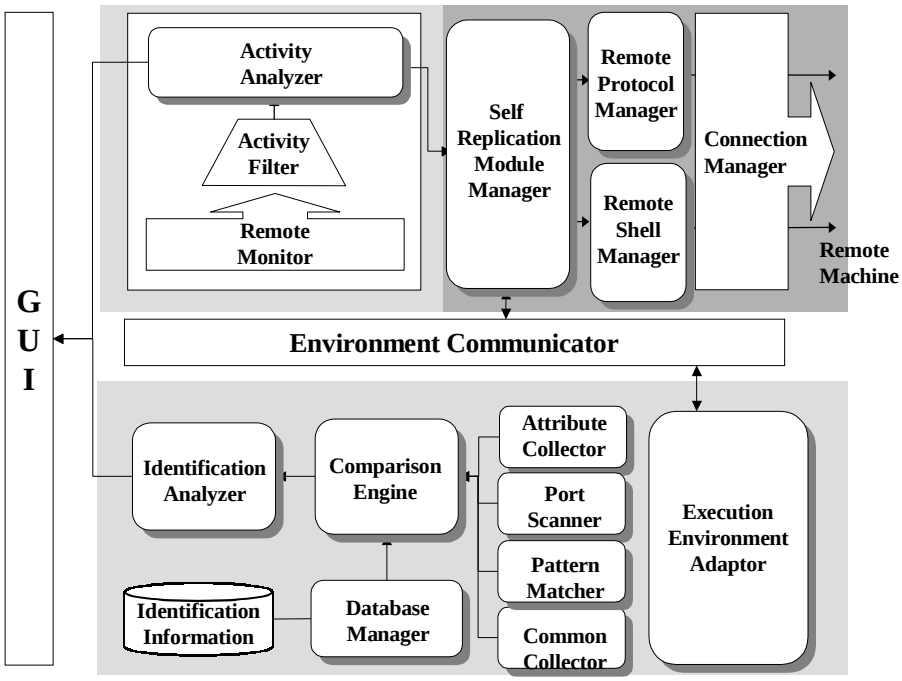

Figure 4 System Architecture

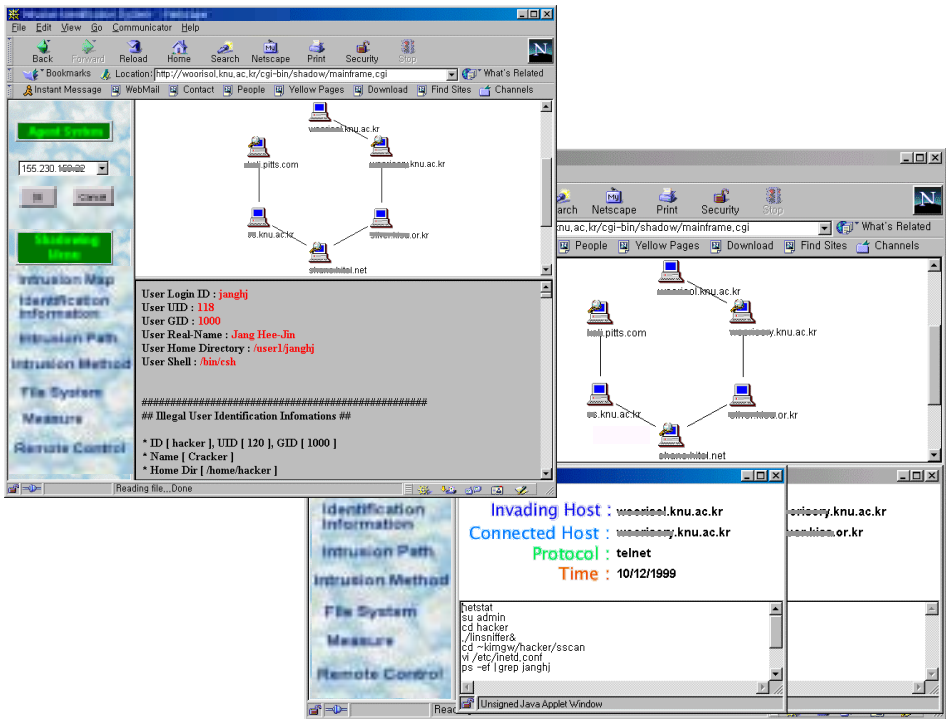

Figure 5 User Interface

methods and the intruder's own file system. In case of a security threat, the administrator can respond, take actions and control the intruder's process remotely.

5. Conclusions and Future Works

It is impossible to cope with the future era of information warfare armed with only the existing system/ network monitoring tools. Protection of information infrastructure and reaction to attack, specifically a defensive attack, are both required. For this reason, this paper proposed the Self-Extension Monitoring using Shadowing Mechanism for Security Management. It monitors intruders by following their movements and hiding the monitoring process itself from the intruders. It enables system security administrators to secure time to investigate various malicious activities.

Currently, the Intruder Identification System based on the Shadowing Mechanism using Replication is developed on Linux 2.2.12-20. A current prototype of the system identifies intruders according to the general attack scenario drawn from previous research.

Future work will focus on completion of the first prototype and research into dealing with unpredicted attacks.

References

[1] J.R. Winkler., C.J. OShea. and M.C. Stokrp, "Information Warfare, INFOSEC and Dynamic Information Defense," Proceedings of National Information Systems Security Conference, December 1996

[2] H.T.Jung, et. al.,"Caller Identification System in the Internet Environment," Proceedings of USENIX Security Symposium IV, 1993

[3] S. Garfinkel, G. Spafford, "Practical UNIX and Internet Security," 2nd Ed. O'Reilly & Associates Inc., pp.731-757, 1996

[4] Russel D. and Gangemi G., Computer Security Basics, O'Reilly & Associates, 1991

[5] E. Nemeth, G. Snyder, S. Seebass and T.R. Hein, "UNIX System Administration Handbook," 2nd Ed. Prentice Hall, Inc. pp.124, 1995

[6] Abuse of the Linux Kernel for Fun and Profit, vol. 7, issue 15, Phrack 50-05, 1997

[7] tcpdump(8) Version 2.2.1, Van Jacobson, Craig Leres, Steven Berkeley, University of California, Berkeley, CA.

[8] Netlog. Mark Gates, Alex Warshavsky and Von Welch, National Laboratory for Applied Network Research. http://dast.nlanr.net/Projects/Netlog/

[9] Alves-Ross J., "An Overview of SNIF: A Tool for Surveying Network Information Flow," Proceedings of the Internet Society Symposium on Network and Distributed System Security, February, 1995

[10] N. Michael, “Monitoring and Controlling Suspicious Activity in Real-time with IP-Watcher,” Proceedings of the Annual Computer Security Applications Conference, December 1995.
[11] Hunt. http://www.cri.cz/kra/index.html#HUNT
[12] T-sight. En Garde Systems, Inc. http://www.engarde.com/.
[13] Hyunchul Jung, “TCP connection hijacking attack and countermeasure,” CERTCC-KR-TR-99-002, Korea Information Security Agency, 1999
[14] Final research report : The Design and Implementation of Intruder Identification System, Korea Information Security Agency, December, 1999
[15] Sangwook Kim, Heejin Jang, Boseok Park, Gunwoo Kim, Junghyun Park and Chaeho Lim, "A Virtual System for Monitoring intruders," Proceedings of the 4th Annual Joint Workshop on Modern Electronic Technologies and Applications, pp.197-202, Beijing, November, 1999