An Intruder Tracing System based on a Shadowing Mechanism

H. Jang and S. Kim
Dept. of Computer Science, Kyungpook National University, Korea
{janghj, swkim@cs.knu.ac.kr}

Abstract

Since current internet intruders conceal their real identity by distributed or disguised attacks, it is not easy to deal with intruders properly only with an ex post facto chase. Therefore, it needs to trace the intruder in real time. Existing real-time intruder tracing systems has a spatial restriction. It is impossible to respond to an attack which is done out of the security domain. This paper proposes Shadowing Mechanism, a new approach to real-time intruder tracing, minimizing a spatial limitation of traceable domain. The real-time tracing supports prompt response to the intrusion, detection of target host and laundering hosts. It also enhances the possibility of intruder identification.

1. Introduction

An identification service is a service which identifies which person is responsible for a particular activity on a computer or network [1]. Currently, most internet attackers disguise their locations by attacking their targets indirectly via previously-compromised intermediary hosts [2,3]. They also erase their marks on previous hosts where they have passed. These techniques make it virtually impossible for the system security officer of the final target system to trace back an intruder in order to disclose intruder's identity post factum. Chasing after the intruder in real time can be an alternative, which supports prompt response to the intrusion, detection of target host and laundering hosts. It also enhances the possibility of intruder identification.

There are several approaches that have been developed to trace an intruder. They fall into two groups such as an ex post facto tracing facility and a real-time identification service [1]. The first type of the intruder tracing approach contains reactive tracing mechanisms. Caller Identification System (CIS)[4] is along this approach. The second type, the real-time identification service, attempts to trace all individuals in a network by the user ID's. The Distributed Intrusion Detection System (DIDS)[5] developed at UC Davis is an example of such system. The biggest problem in the existing intruder tracing approaches is a restriction on the traceable domain.

As a solution, this paper presents the Shadowing Mechanism that meets the aforementioned requirements. It also introduces the HUNTER which is a real-time intruder tracing system based on the Shadowing Mechanism. The Shadowing Mechanism keeps track of a new connection caused by the intruder and replicates the security scheme to the target host through the connection. It broadens the security domain dynamically following the intruder's shifting path. It means that the traceable domain is extended.

2. Shadowing Mechanism

We here define a trusted domain. The trusted domain is composed of several domains including single

Proceedings of the Seventh International Symposium on Computers and Communications (ISCC'02)
1530-1346/02 $17.00 © 2002 IEEE

administrative domains or cooperative domains. In the trusted domain, each administrator of constituent domains also has the administrative privilege in other domains. The Shadowing Mechanism supports dynamic extension of the security domain. The Shadowing Mechanism observes behavior of the user who is presumed as an intruder and acquires activity and identity information of the user. Using these data, it replicates itself or any other security scheme automatically in the hosts where an intruder has passed. Consequently, it broadens the security domain for data collection used for security management and intruder tracing. The Shadowing Mechanism consists of monitoring and filtering, replication and self-protection.

Monitoring in the Shadowing Mechanism is a data-collecting phase for replication. It is done for specific users or all users who enter the trusted domain. The output of monitoring activity is a log. For replication, the Shadowing Mechanism filters some useful states or events among logs, which is related to establishing of new connections. Interesting states are aspects of important objects which can affect system security such as *setuid* and *setgid* files, users with root privilege, or integrity of important files. Events under the close observation include the action which relates to the creation of a user session, to the change in an important account, or to the intrusive behavior. The user session creation event contains commands such as *telnet, ftp, rlogin*, *rexec* and *rsh*. The account change event includes activity of other user's privilege gain using *su* command. The intrusive behavior event comprises actions such as vulnerability scanning, backdoor installation, creation of malicious process or new trap, and illegal acquisition of root privilege.

After filtering, the Shadowing Mechanism decides the target host and starts to replicate itself to the host when an intruder succeeds in connecting with another host. We just take the connection through a medium object into consideration in this paper. The Shadowing Mechanism provides all users with medium objects which work normally but are controllable by this mechanism. It delivers modules and events for replication to the target host via the medium object, especially the medium object with a command processing function such as a pseudo terminal with a working shell.

Figure 1 illustrates the event transmission through the pseudo terminal object. A hexahedron denoted as US_X is a user space in each host. It includes all actions by a specific user and every resource related to those actions. The front rectangle of the US_X is a perceptible part to the user such as standard input or standard output. As it goes back, it gets closer to the operating system.

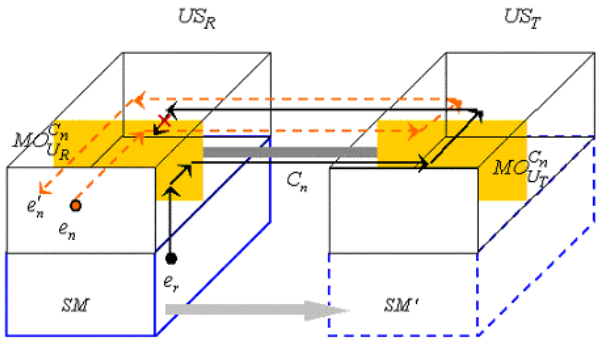

Figure 1. Event transmission

A solid arrow shows transfer of specific event $e_r \in E_R$ where E_R is a subset of E and a set of events such as copying, compiling or execution command for replication. A dotted arrow indicates forwarding of normal events $e_n \in (E - E_R)$. For example, a user, U_R in a host, H_R which already has the Shadowing Mechanism *SM* attacks a host, H_T and U_R becomes U_T who has super-user privilege in H_T.

A connection C_n is set up between two pseudo terminal objects, $MO_{U_R}^{C_n}$ which is allocated for U_R in the H_R by *SM* and $MO_{U_T}^{C_n}$ for U_T in the H_T by an operating system. Therefore, it is possible to send an event to $MO_{U_T}^{C_n}$ via $MO_{U_R}^{C_n}$ in order to remotely execute any command in H_T

from H_R. A normal event e_n is delivered to H_T via $MO_{U_T}^{C_n}$ and $MO_{U_R}^{C_n}$ so that the user U_R can accomplish the event e_n with the privilege of U_T in the H_T.

The event, e_n, is carried out in the H_T normally and the result, e_n' is showed to the user, U_R.

The *SM* makes the replication event e_r, which is delivered to $MO_{U_T}^{C_n}$ via $MO_{U_R}^{C_n}$ and applied to H_T.

The result at H_T comes back to US_R but is not showed to $MO_{U_R}^{C_n}$ to keep any intruder from watching the whole process.

It protects the replication process and *SM* itself from detection by U_R. As a result, *SM* of the host H_R replicates itself to the host H_T and the security scheme operates at H_T.

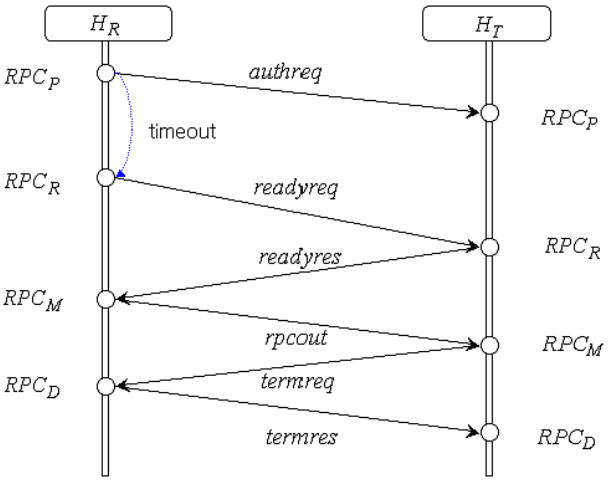

Figure 2. Replication protocol

Figure 2 depicts the message exchange for replication when an intruder attacks the target host H_T from the host H_R through the medium object. RPC_X shows the replication status in each host. When a connection is established by the specific event *e* in the state of RPC_P, host H_R sends a *authreq* message to request the authentication for replication. It is a verification process that examines the existence of the same security scheme in the target host. If there is the same security scheme, H_T delivers a response message *authres*. H_R certifies legitimacy of the scheme and terminates the replication process. Otherwise the host H_R enters a replication ready state of RPC_R and sends a *readyreq* message to check the intruder's environment in the H_T. The target host enters the state RPC_R and transfers a *readyres* message which is the information about the intruder's execution environment in the target. After recognizing the intruder's environment, H_R enters a replication execution state of RPC_M and transmits modules and events for replication with *rpcout* message to H_T. H_T in the state of RPC_M executes the commands from the host H_R. The Shadowing Mechanism is set up in the host H_T and starts inspecting the host and the specified intruder. And chasing an intruder continues. H_T sends *termreq* message to inform H_R of completion of the replication process. Then H_R enters a replication completion mode RPC_D and puts H_T into the state RPC_D by transmitting the *termres* message. Since the replication process is hidden from an intruder and the intruder's execution environment is maintained in the target host, the intruder cannot recognize the process.

3. HUNTER : Real-Time Intruder Tracing System

3.1. System Overview

The HUNTER is composed of a single master system and several agent systems. This is initialized by installing a master system in the host which is the unique entrance to the trusted domain (for example, routers). Initially, there is a master system only in the trusted domain. If an intruder moves into another host via master system in the trusted domain, agent system is automatically placed into the target host through the replication process. As the replication process goes on following the intruder's movement path, the number of agent systems increase dynamically. This system is implemented in the GNU C/C++ 2.7.x.x for core modules and Java 2 SDK for the

user interface on Linux 2.4.6 and Solaris 2.8.

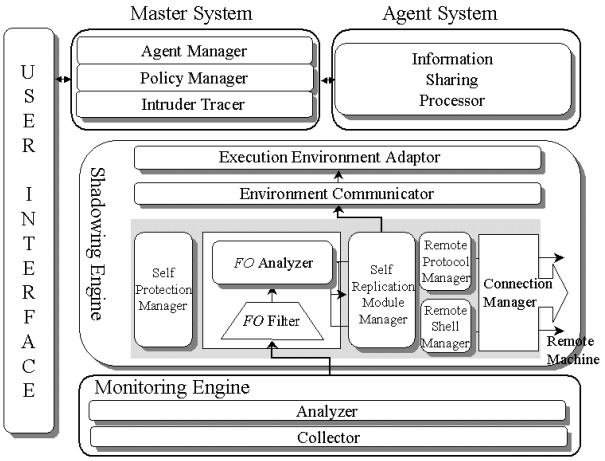

Figure 3. System architecture

Figure 3 describes the architecture of HUNTER. The master system and agent systems share a Monitoring Engine and a Shadowing Engine. The Monitoring Engine gathers activities of suspicious users, examines each collected activity and produces the formalized object *FO*. Certain critical *FOs* are always transmitted to the master system in real-time; others are processed locally by the agent system and only summary reports are sent to the master system. The master system manages the predefined rules to trace an intruder. The Intruder Tracer of master system extracts the useful pieces among *FOs* and constructs a connection chain which will be explained in subsequent sections. The Agent Manager controls all the distributed agent systems in the trusted domain.

A Shadowing Engine replicates the security scheme following intruder's migration. A *FO* Filter extracts useful pieces among data sent from the Monitoring Engine and a *FO* Analyzer decides the point of time and the target host for replication. The Connection Manager attempts to establish the TCP connection with the target host. The Self Replication Module Manager checks the existence of the same security scheme in the target host. If there is same security scheme, the Self Replication Module Manager verifies that the installed scheme is the legal one through authentication and terminates the replication into the target host. Otherwise it lets the Remote Protocol Manager and the Remote Shell Manager send the security scheme to be copied and commands for installation, compiling and running of the duplicated modules to the target host. If above process is successful, the security scheme is set up in the target host. Since the security scheme is replicated using the pseudo terminal as a medium object, it is necessary to maintain an intruder's environment so that the intruder cannot recognize the replication. The Environment Communicator and Execution Environment Adaptor support this maintenance. Replication protocol in the Shadowing Mechanism works as explained in section 3. This plays an important role in earning some time to observe an intruder. The Self-Protection Manager attempts both to erase shadowing tracks and to blend into the normal Unix/Linux environment using camouflage in order to protect the monitoring activity itself.

3.2 Intruder Tracing by the HUNTER

This system assigns trace-id(TID) to a new user who is decided to be the intruder by any intrusion detection module and maintains a connection chain about TID. The connection chain chases the intruder's movement. The connection includes all sorts of connections which can occur through the pseudo terminal.

The master system monitors all users' activities on the host which the master system runs. The agent system just watches the users thought to be intruders by the master system. Monitoring activity records a log from which the formalized object *FO* is generated. A useful data abstracted from *FO* contains a connection object and a user object. Concerning Unix and Linux, the three ways to create a new connection are for a user to login from a terminal, console, or off-LAN source, to login locally or

remotely from an existing user object legitimately, and to gain other user's privilege by illegal method such as a buffer overflow attack. These connections make new user objects and connection objects. The master system receives those objects from agent systems. It constructs a connection chain from connection objects and tries to associate the user object with an existing TID or allow the user object a new TID. We consider a user object $uo_i \in UO$ ($i \geq 0$) to be the 4-tuple <*TID, UserID, HostID, Time*> where *UO* is a set of user objects in the trusted domain. After TID generating rule is applied to the user object, value for TID is assigned.

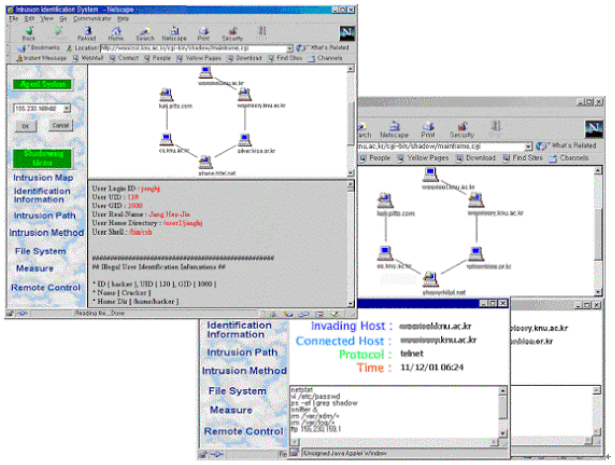

Figure 4. User interface

Whenever a new connection object is created, new user object is formed and applicable TID is assigned to the user object. Finding an applicable TID consists of several steps. If a user changes identity on a host, the new user object is assigned the same TID as the previous one. If a user establishes a new connection with another host, the new user object gets the same TID as that of the source user object. The new user object is assigned the same TID as the previous identity in the case where the intruder obtains the superuser privilege in the remote host using vulnerabilities of the remote server. Since the user who logins from a terminal, console, or off-LAN source does not have the previous identity, new TID is assigned to the user object. Each TID maintains its own single connection chain to keep track of the intruder. Whenever a user with same TID sets a new connection, the generated connection object is appended to the established connection chain. New connection chain is created if new TID is allocated for the user object. The connection chain is a sequence of more than one connection object. The connection chain makes it possible to trace an intruder and disclose the source of attack and the intruder's identity. Figure 4 presents the web-based user interface of HUNTER.

4. Experimental Results

An Intruder Tracing Rate Per Individual $ITRPI_i$ is a degree of tracing the intrusive path of the user i who establishes new connections within the trusted domain D_t. *ITR* is the mean intruder tracing rate for all users who are inferred to be intruders in the trusted domain. They are given by

$$ITRPI_i = \frac{ICC_i}{BI_i} \text{---(1)} \qquad ITR = \frac{\sum_{i=1}^{n} ITRPI_i}{n} \text{---(2)}$$

where BI_i is the number of connection objects generated by the user i in the D_t. ICC_i is the number of connection objects in the connection chain maintained for the user i uniquely by the HUNTER. n is the number of distinctive intruders in the domain D_t.

The target network includes 48 hosts and based on the Ethernet. It is in a single trusted domain. In this experiment, we confine the operating system to Solaris 2.4 or Linux 2.4.6 or above, and offered service to *telnet, ftp, www*, and e-mail service. In order to lower a complexity, we assume that an IMAP vulnerability[6] is implicit in every target and intermediary hosts in the trusted domain. We also presume that the only attack is buffer overflow using vulnerability of the IMAP server and the success rate of the attack is 100%.

We assess the intruder tracing rate by the variation of an initial security domain location. We assume an initial security domain covering only one host and a specific intrusive path within the trusted domain. The path includes 12 hosts which are distributed in four subnets. There is only one host X with the master system in the trusted domain. A condition for this experiment is as follows. In case A, X is out of the intrusive path. In case B, X is the fifth host on the intrusive path. In case C, the intruder attacks the host X first to penetrate the trusted domain.

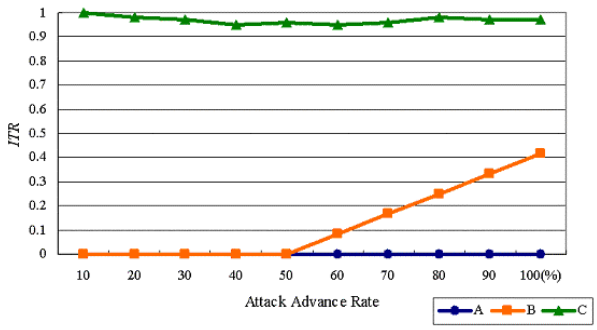

Figure 5. ITR by the attack advance rate

Figure 5 shows the result of the experiment. The x-axis presents the degree of an attack advance through the intrusive path. The ITR indicated by y-axis is 1 if every connection caused by the intruder is noticed within the trusted domain. In case A, the attack is advanced out of the security domain. That's why the intruder-tracing rate is 0. In case B, not all intrusion paths can be traced. It is possible to trace the path when it begins to pass the master system. In case C, when the intruder passes through the host X first to penetrate the trusted domain, the extended security domain covers the total intrusive path within the trusted domain, making it possible to trace the intruder. This experimental result shows that the effect of the Shadowing Mechanism can be maximized if the master system is in the host which is an unique entrance to the trusted domain.

5. Conclusions

Existing security management systems including intruder tracing systems have the restriction on the security domain. As an intruder continues to attack across several hosts, it is impossible to respond to the attack properly. For this reason, this paper proposed the Shadowing Mechanism and HUNTER which is an intruder tracing system based on the mechanism.

Future work will focus on the completion of the prototype. Since the proposed approach in this paper traces the user who is assumed to be the intruder by any intrusion detection system, it is necessary to consult intrusion detection system. Response mechanism is also needed to recover the system after intrusion.

6. References

[1] S. S. Chen and L. T. Heberlein, "Holding Intruders Accountable on the Internet," Proceedings of the 1995 IEEE Symposium on Security and Privacy, pp. 39-49, May 1995.

[2] G. Eschelbeck, "Active Security-A proactive approach for computer security systems," Journal of Network and Computer Applications, 23, pp.109-130, 2000.

[3] D. Schnackenberg, K. Djahandari and D. Sterne "Infrastructure for Intrusion Detection and Response," Advanced Security Research Journal, 3, pp. 17-26, 2001.

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

[5] S.Snapp et al., "DIDS(Distributed Intrusion Detection System) – Motivation, Architecture, and an early prototype," Proceedings of National Computer Security Conference, pp.167-176, Oct 1991.

[6] S.J. Kim, IMAP Vulnerability, CERTCC-KR-TR-98-009, Korea Information Security Agency, Aug 1998.