25

Extending Distributed Audit to Heterogeneous Audit Subsystems

Chii-Ren Tsai

Citicorp International Communications, Inc.
Citicorp Global Information Network
1900 Campus Commons Drive, Reston, VA 22091, U.S.A.

ABSTRACT

We have designed and implemented a prototype of a distributed audit mechanism [1] for AIX* audit subsystems of AIX 2.2.1 and AIX Version 3 [2] [3] [4]. In the prototype, each host is running a distributed audit daemon. A central auditor can instruct each daemon to turn on/off auditing, perform audit system management, or trace audit trails. To provide distributed audit in a heterogeneous environment, we have extended the prototype to support the mechanism in VM* RACF systems. The efforts include modifying the central auditor interface and porting the distributed audit daemon to VM. Since AIX audit subsystems are totally different from the VM audit subsystem, the work has involved the integration of heterogeneous audit subsystems to support a central auditor interface.

1. INTRODUCTION

We have combined the services of the MIT Kerberos$ authentication protocol [5] with vanilla RPCs of the Network Computing System# (NCS) [6] to develop an experimental secure RPC mechanism [2]. Based on the secure RPCs, we have implemented a prototype of a distributed audit mechanism for AIX [2]. In this mechanism, we specify a central *auditor* role that (1) can invoke and revoke auditing for remote hosts, (2) can locate the *audit trail server* to which audit trails are transferred, (3) can perform audit system management, such as dynamically adding or deleting audit events on a per–user, per–group, or per–system basis and querying the audit status of remote hosts, and (4) can retrieve and analyze audit trails. In addition, we have implemented a user friendly interface for distributed audit with OSF@ Motif@ widgets on the X Window system$ [7].

AIX 2.2.1 for RT*, AIX Version 3 for RISC System/6000* and VM/SP* are designed to satisfy the C2 security requirements of the Trusted Computer Systems Evaluation Criteria [8]. Consequently, they all contain an audit subsystem. Even though the AIX 2.2.1 and Version 3 audit subsystems are slightly different, both subsystems are implemented with the same audit mechanism. In the following text, we will use the AIX audit subsystem to represent both. The VM audit subsystem is totally different from the AIX audit subsystem. To extend the prototype of the distributed audit mechanism for AIX to the VM audit subsystem, discrepancies between them must be explored and unified.

In this paper, we highlight and compare the differences between the AIX and VM RACF audit subsystems. We also review the distributed audit mechanism and the frame work of the

---

The work described herein was performed under contract to IBM when the author was with VDG, Inc., and only contains the author's perspectives. It does not represent, imply or describe any IBM products. Also, it does not represent any opinion of Citicorp International Communications, Inc.

* AIX, VM/SP, RT and RISC System/6000 are trademarks of International Business Machines Corporation.
$ Kerberos and the X Window system are trademarks of the Massachusetts Institute of Technology.
# NCS is a trademark of the Hewlett–Packard Corporation.
@ Motif and OSF are trademarks of the Open Software Foundation.
% NFS is a trademark of SUN Microsystems, Inc.

J. H. P. Eloff et al. (eds.), *Information Security — the Next Decade*

prototype including the experimental secure RPC mechanism and its Motif–based auditor interface. Furthermore, we present the design and implementation of the distributed audit daemon in VM. Finally, we discuss the interoperability and limitations of the distributed audit daemon for VM.

2. OVERVIEW OF AIX AND VM RACF AUDIT SUBSYSTEMS

The AIX and VM audit subsystems are implemented with different audit mechanisms. For instance, AIX provides a single command to turn on/off auditing for all system activities, while VM provides several subcommands for finer granularities of audit control. In terms of audit records collection and retrievement, AIX audit records can be retrieved through a system device, or compressed and stored in a file called *audit trail* that is configurable, while VM audit records must be stored in the SMF (System Management Facilities) database. To select and print audit records, AIX provides several commands and VM provides a tool.

2.1. AIX Audit Subsystem

The AIX audit subsystem provides two mechanisms to collect audit records, namely bin collection and stream collection. For bin collection, two files, **/audit/bin1** and **/audit/bin2**, alternately store audit records; for stream collection, audit records are stored in a circular buffer inside the kernel and can be retrieved through a system device, **/dev/audit**. If one of these mechanisms is configured through the audit configuration file, **/etc/security/audit/config**, the audit subsystem invokes the corresponding audit daemon, **/etc/auditbin** for bin collection or **/etc/auditstream** for stream collection, whenever the system's auditing is turned on. In addition, one command file for each mechanism can be used to specify commands needed to transfer newly generated audit records to an audit trail which file is **/etc/security/audit/bincmds** for bin collection or **/etc/security/audit/streamcmds** for stream collection.

AIX provides commands, **auditselect** and **auditpr**, to select and print audit records from the system's audit trail, which contains the compressed audit records. All audit records have a common header that includes audit event, login user name, login user ID, real user name, real user ID, process ID, parent process ID, command that generated the audit event, the result of the command execution, date and time that the audit event is generated. Any combinations of the fields in the header can be used as a key to select audit records.

2.2. VM RACF Audit Subsystem

Resource Access Control Facility (RACF) in VM is a reference monitor that implements system's security policy, authenticates users, performs access controls, and audits events relevant to system's security. The VM audit subsystem is not a single module as that in AIX. Audit is controlled by RACF and the directory maintenance module, DIRMAINT, that manages system directory and audits every security–relevant task it performs. RACF performs general audit control and manages auditable events. General audit includes the following activities:

(1) modifications to profiles (AUDIT),
(2) the activities of SPECIAL and group–SPECIAL users (SAUDIT),
(3) the activities of OPERATIONS and group–OPERATIONS users (OPERAUDIT),
(4) command violations (CMDVIOL), and
(5) access attempts to RACF–protected resources (SECLEVELAUDIT),

where profiles are data that describes the significant characteristics of users, or computer resources. Note that RACF privilege classes consist of SPECIAL, OPERATIONS and AUDITOR. Each class of RACF privilege is discrete and not predicated on others. The SPECIAL user has full control over all profiles in the RACF database, while the group–SPECIAL user has same control as the SPECIAL user but limited to the scope of the group. The OPERATIONS user has full control of all RACF–protected resources that allow OPERATIONS access. The AUDITOR class gives the user authorization to specify auditing and logging options within RACF profiles.

The auditor in VM must also have the AUDITOR attribute and the VMEVENT class authorization to create the system or user VMEVENT profiles such that auditable VM events can be turned on/off on a per–system or per–user basis. The SETROPTS and SETEVENT commands are used to conduct general audit control and turn on/off auditable VM events, respectively. The VM audit subsystem starts collecting audit records as soon as audit events are set and refreshed in the system or user VMEVENT profiles. Note that event configuration and turning on/off auditing are two distinct actions in AIX so that the AIX audit subsystem does not start collecting audit records until the auditing of the system is turned on.

VM provides a tool called the *RACF report writer*, which can retrieve audit records in the SMF database and generate audit reports. The processing of the RACF report writer consists of three phases: (1) command and subcommand processing, (2) record selection, and (3) report generation. The RACF report writer cannot be invoked directly from the command line. To invoke the report writer, the report writer command, **RACFRW**, and its subcommands, namely **SELECT, EVENT, LIST, SUMMARY** and **END**, must be written in a file, **RACFRW.CONTROL** and then the auditor invokes the **RACRPORT** command to execute the file. Therefore, an audit report is generated accordingly.

3. AN EXPERIMENTAL SECURE RPC AND DISTRIBUTED AUDIT MECHANISM

The structure of our experimental secure RPC mechanism is shown in Figure 1. In this structure, all security enhancements, including authentication, data encryption and decryption, and authorization, are highlighted and implemented at the client and server stubs, so that interfaces of secure RPCs are the same as those of vanilla RPCs and RPC runtime is left unchanged. The secure RPC mechanism uses Kerberos authentication protocol, encrypts input and returned data, provides data encryption standard (DES) cipher–block–chaining (CBC) checksums for data, and performs access checks against the caller's identity. The use of Kerberos provides the capability of authentication, data secrecy and integrity, and allows the implementation of various identity–based authentication mechanism. The secure RPC protocol is described in [2] and reviewed in [3].

The distributed audit mechanism discussed herein supports a central auditor role that can perform audit system management, invoke/revoke auditing for each host, instruct each host to transfer its audit trail to a specific site called an *audit trail server*, and retrieve and analyze audit trails. All audit trail transfers are done through Network File System% (NFS) [9]. As illustrated in Figure 2, the auditor can invoke/revoke auditing by using the secure RPC. Using NFS each host mounts the audit trail filesystem from the audit trail server over a local directory. Then, the host's audit records are compressed and stored in a file, which is called the audit trail of the host. As a result, audit trails are collected in the audit trail filesystem of the audit trail server, and the auditor can centrally manage these audit trails, trace user activities, or monitor security violations. An audit trail server must be equipped with a large memory space to store audit trails because audit trails may grow faster than 20 Kbytes per second on RT [1] or 500 Kbytes per second on RISC System/6000 if all audit events are turned on. The auditor has to determine what the audit trail retention period is before the audit trail filesystem is exhausted.

By using the secure RPC protocol, the auditor and each audit RPC server authenticate each other for each RPC issued by the auditor. To ensure that audit trails are transferred with NFS to the specified audit trail server, appropriate authentication between each host and the audit trail server is necessary. Therefore, NFS must be configured with a secure option which uses the DES encryption mechanism and public key cryptography to authenticate users and machines in the network [10].

4. IMPLEMENTATION

AIX provides a system call, audit, that allows the auditor to query the audit status. Unlike AIX, VM does not formally support the concept of the audit status since the auditing is always on

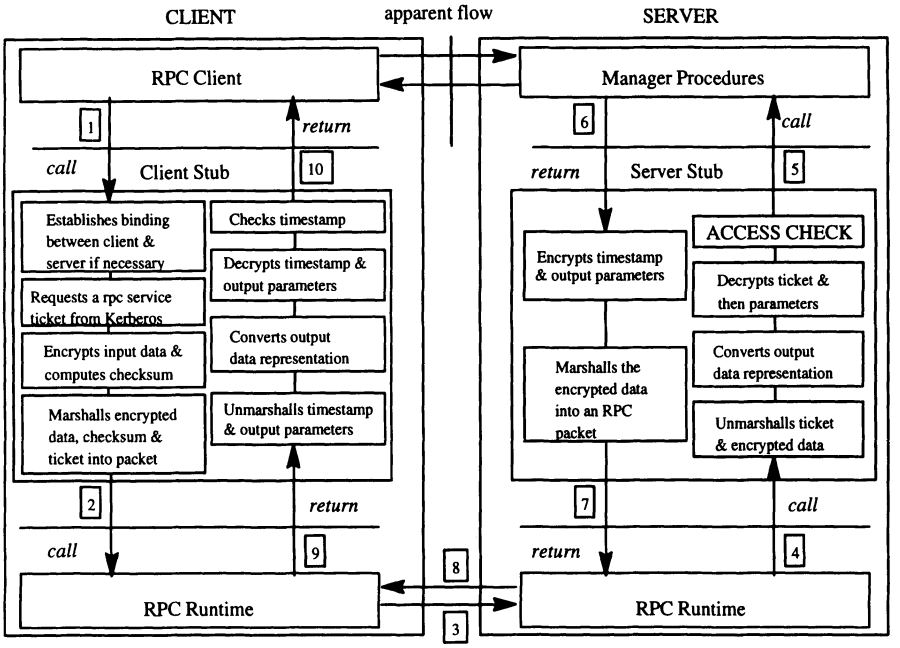

Figure 1. A Secure RPC Paradigm [2]

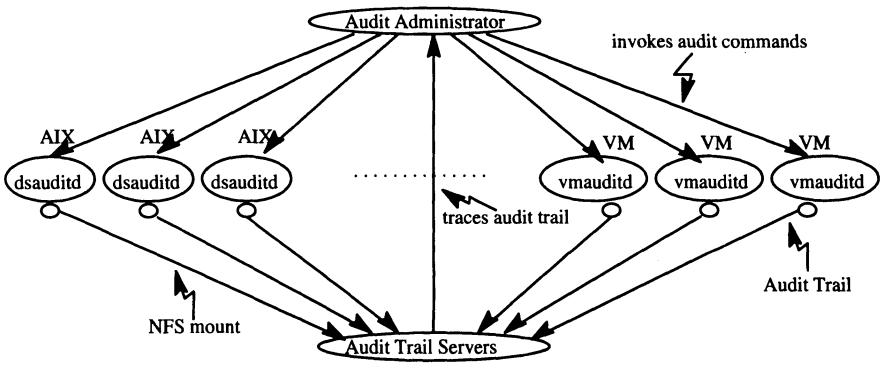

Figure 2. A Distributed Audit Mechanism for Heterogeneous Audit Subsystems

as long as an audit event is set and refreshed. Alternatively, the auditor can query the system or each user's vmevent profile to figure out if an audit event has been turned on for the system or a specific user. Nevertheless, it is time–consuming to go through each user's vmevent profile to find out any audit event has been turned on since each VM system may be shared by hundreds or thousands of users. Consequently, we create a VM audit configuration file, **VMCONFIG**, to

reduce the efforts for discovering the audit status. A **VMCONFIG** example is shown in Figure 3.

```
setropts:
        AUDIT = NO
        SAUDIT = NO
        OPERAUDIT = NO
        CMDVIOL = NO
        SECLEVELAUDIT = NO

classes:
        general = LOGON,LOGOFF,HALT,HOLD,SHUTDOWN,COMMANDS,DISCONN,PER
        appent = APPCCON,APPCSVR,IUCVCON,IUCVSVR
        spool = SFOPEN,SFCREATE,SFDEL,SPOOL,CHANGE.G,CHANGE.S,CLOSE,DUMP
        spool1 = LOADBUF,LOADVFCB,ORDER.G,ORDER.S,PURGE.G,PURGE.S,TAG
        device = SPTAPE,TERMINAL,SCREEN,REWIND,SOUPLE,ECHO,NOTREADY,READY,RESET
        comm = NETWORK.O,NETWORK.R,SEND,DIAL,MESSAGE.G,MESSAGE.O,SMSG
        query = QUERY.A,QUERY.C,QUERY.G,QUERY.O,QUERY.P,QUERY.R,QUERY.S,QVM
        set = SET.A,SET.C,SET.G,SET.R
        cp = ATTN,CP,CPTRAP,DMCP,DRAIN,FLUSH,FREE,LOCATE,MIGRATE,MSGNOH,REPEAT
        cp1 = SAVESYS,SPACE,SPMODE,STAR,START,STCP,VARY,WARNING
        process = FORCE,IPL,REQUEST
        file = LOCK,UNLOCK,TRANSFER.G,TRANSFER.S
        vmops = ADSTOP,BEGIN,DISCONN,DEFINE.G,DEFINE.R,DISPLAY,EXTERNAL
        vmop1 = INDICATE.A,INDICATE.G,INDICATE.O,SLEEP,STORE,SYSTEM,TRACE,VMDUMP
        disk = ATTACH,DETACH.G,DETACH.R,LINK
        adm = ACNT,ATTACH,AUTOLOG,BACKSPAC,DCP,DISABLE,ENABLE,MONITOR
        hpo = CACHE,XLINK,XSPOOL
        diag01s = DIAG00,DIAG04,DIAG08,DIAG0C,DIAG10,DIAG14,DIAG18,DIAG1C
        diag23s = DIAG20,DIAG24,DIAG28,DIAG2C,DIAG30,DIAG34,DIAG38,DIAG3C
        diag45s = DIAG40,DIAG44,DIAG48,DIAG4C,DIAG50,DIAG54,DIAG58,DIAG5C
        diag67s = DIAG60,DIAG64,DIAG68,DIAG6C,DIAG70,DIAG74,DIAG78,DIAG7C
        diag89s = DIAG80,DIAG84,DIAG88,DIAG8C,DIAG90,DIAG94,DIAG98,DIAG9C
        diagABs = DIAGA0,DIAGA4,DIAGA8,DIAGAC,DIAGB0,DIAGB4,DIAGB8,DIAGBC
        diagCDs = DIAGC0,DIAGC4,DIAGC8,DIAGCC,DIAGD0,DIAGD4,DIAGD8,DIAGDC
        diagEFs = DIAGE0,DIAGE4,DIAGE8,DIAGEC,DIAGF0,DIAGF4,DIAGF8,DIAGFC
users:
        PERSYS = general
```

Figure 3. An Example of the vmconfig File

The first stanza in **VMCONFIG** is setropts. The VM audit includes three distinct paths: audit through the SETROPTS command, the SETEVENT command, and the DIRMAINT module; while SETROPTS has five audit operands, namely AUDIT, SAUDIT, OPERAUDIT, CMDVIOL, and SECLEVELAUDIT. The meaning of each operand has been described in section 2.2. These operands are configurable because if there is a local audit policy the policy cannot be interfered. For instance, a local audit policy states that the activities of SPECIAL users and command violations must be audited all the time. Consequently, SAUDIT and CMDVIOL are set to NO, and the rest of the operands are set to YES. Whenever the central auditor turns on/off the auditing, only AUDIT, OPERAUDIT and SECLEVELAUDIT are affected, while SAUDIT and CMDVIOL are left unchanged.

The second stanza in **VMCONFIG** lists the definitions of audit classes. Each audit class represents a set of audit events. The central auditor turns on/off audit classes in the sense of audit events instead of audit events directly, which simplify audit event configuration. The simplicity shown in the Figure 3 is 25 audit classes against 184 audit events for VM/SP.

The last stanza in **VMCONFIG** records audit classes configured for users. Since VM provides the auditing on a per–system or per–user basis, the first entry in this stanza is PERSYS,

which stands for the per–system basis. Whichever audit classes are configured for PERSYS or any user, the corresponding audit events are set into the system vmevent profile or the user's vmevent profile whenever the auditing is turned on. If the auditing is turned off, the stanza still keeps the information of audit classes configured for the system and users. This separates audit event configuration and the activation of the auditing into two steps.

The AIX audit subsystems provides the audit capability on a per–user basis. We have extended the capability to a per–user, per–group or per–system basis. Similarly, we also extend the audit capability of VM to a per–user, per–group or per–system basis. To support these features, we parse the output of RACF commands LISTUSER and LISTGRP to create the ETC.USERS and ETC.GROUPS files, which files contain user lists and the definitions of groups, respectively. By these two files, the auditor is able to perform audit event configuration.

To invoke the RACF report writer, it is necessary to ipl (Initial Program Load) the RACFVM 490 minidisk before executing **RACRPORT** and to ipl CMS after the execution. Those two subsequent ipl actions cannot be invoked from the distributed audit daemon because it involves resetting the nucleus in VM. Consequently, we cannot use **RACRPORT** to perform **RACFRW.CONTROL**. Fortunately, we obtain another program, **RACRPORX**, that can run in CMS and provide the same functionality as that of **RACRPORT** without ipling the minidisk 490. Whenever **RACRPORX** is invoked, an audit report is generated and stored in a file, **RACFRW.REPORT**, so that the VM distributed audit daemon can call a Motif–based application function to open a window at the central auditor's display and show the audit report.

5. INTEROPERABILITY AND LIMITATIONS

The distributed audit prototype is based on TCP/IP, Kerberos, NCS and the X window system, so that TCP/IP, the Kerberos server, and the NCS global location broker must be properly configured. As shown in Figure 4, the Kerberos server is configured in an AIX system, *allosaur*. The VM key tables can be generated by AIX since AIX and VM Kerberos key tables are the same.

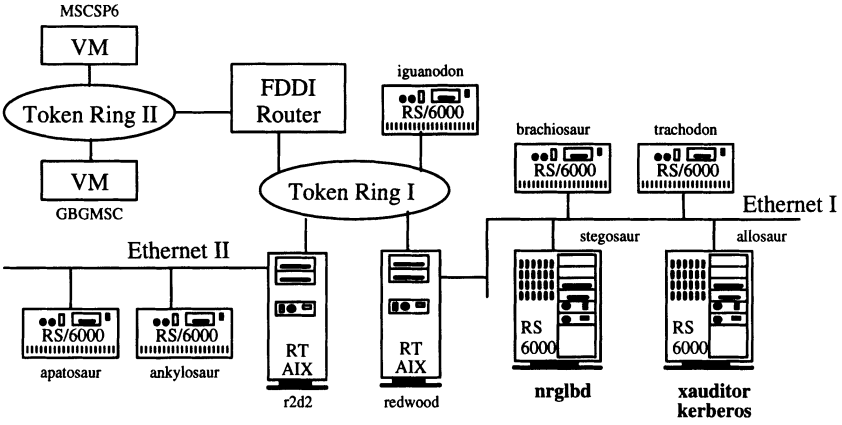

Figure 4. Network Environments for Distributed Audit

5.1. Remote NCS Global Location Broker

As shown in Figure 4, we configure the NCS global location broker, **nrglbd**, in an AIX system, *stegosaur*, which is in a network different from that of VM systems. The NCS local

location broker running in VM cannot automatically locate a global location broker outside the local network since it only broadcasts within the local network to identify the location of the global location broker and then stores the information into its database. However, the local location broker can identify the global location broker as long as its database can be changed to store the address of the global location broker. Consequently, we delete the local location broker database after the local location broker is brought up, and use a program to re–register the global location broker.

5.2. The Central Auditor

An instance of the central auditor interface is shown in Figure 5. Other than the HELP and QUIT buttons, there are four major buttons—namely, ON, OFF, CONFIG and TRACE—on the main window. Each button is associated with a pull–down menu that is used to select either a specific host or all the hosts. Also, four panels display the names of the hosts, the audit status, audit trail sizes (in bytes), and the times of turning on/off auditing. Note that the audit status and audit trail sizes are periodically updated by the program. Nevertheless, two shaded buttons on the top of the status panel and the trail sizes panel allow the auditor to instantly update the audit status and audit trail sizes, respectively. Another shaded button above the time panel provides the values of the clock deviations among the clients and the auditor systems, and the capability to synchronize the clients' clocks with the auditor's.

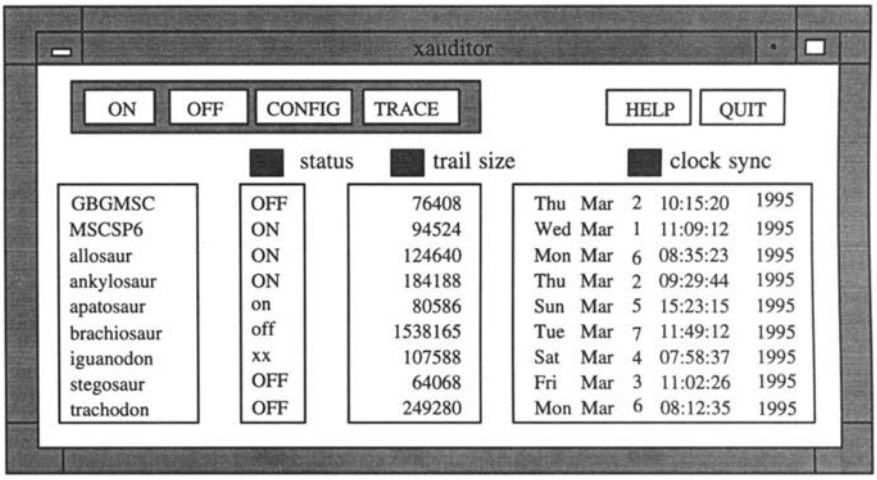

Figure 5. The Central Auditor Interface

The ON and OFF buttons are used to turn on and off auditing, respectively. The auditor can press the TRACE button to trace all the audit trails or specific audit trails with query; those functions are selected through subsequent pop–up menus. The CONFIG button allows the auditor to configure the audit trail servers, the upper–limits for audit trail size, and the fullness of audit trail filesystems, and to specify audit classes for users on a per–system, per–group, or per–user basis, where an audit class is defined as a set of audit events. Through these configurations, the audit subsystem can collect audit records of specified audit events (classes) for users, can transfer those audit records to the appropriate audit trail server, and can turn off auditing if one of those parameters exceeds its upper–limit.

The distributed audit system encompasses its an integration of stand–alone audit subsystems, so that both the central auditor and the local auditor can manipulate the audit subsystem. If an action of the central auditor causes an audit, the audit status is displayed in capital letters such as ON and OFF shown in Figure 5, otherwise, it is shown in small letters. Any intervention of the local auditor will immediately signal the central auditor. If the audit rpc daemon of a host was running and is out of service, its status is marked as "xx." If a daemon has never been reached, its status is left blank.

5.3. Limitations

If the separation of roles mechanism has been enforced in VM, the auditor may not have the privilege to mount a remote filesystem over a local minidisk or synchronize the system clock with a remote host's unless additional privilege attribute is added to the auditor. We assume that clocks among the central auditor host and VM systems are loosely synchronized, the central auditor, consequently, does not perform clock synchronization among VM systems. If the auditor does not have the privilege to mount a remote filesystem to the RACFVM 301 or 302 minidisk, the audit trail cannot be transferred to the audit trail server, and the growth of the audit trail cannot be shown. Furthermore, VM does not support the X server such that the Motif–based central auditor interface cannot be displayed under VM.

6. CONCLUSIONS

Interoperability among heterogeneous audit subsystems is very important in security management for distributed systems since audit is a basic security requirement for all trusted computer systems. This work shows the effort to provide a common management interface for heterogeneous audit subsystems. From this work, we learn that each system's audit trail must be separated from other system data such that audit trails can be managed by the central auditor of distributed audit. Also, a tool to trace user activities or security violations among audit trails generated by heterogeneous audit subsystems needs to be developed.

ACKNOWLEDGMENTS

The author is grateful to the following people at IBM: Gerry J. Boncavage, Janet E. Brown, Anthony J. Nadalin and Michael K. Mattheiss, without their help the work cannot be done, especially Anthony J. Nadalin helped solving several major implementation issues on VM. The author also acknowledges Professor Virgil D. Gligor of University of Maryland, College Park, for his support of this effort and suggestions on this paper.

REFERENCES

1. C.–R. Tsai, V. D. Gligor and M. S. Hecht, "Potential Pitfalls of a Distributed Audit Mechanism," in Proceedings of the 1990 EurOpen Autumn Conference, Nice, France, pp. 91–103, October 1990, also available as *IBM Gaithersburg Technical Report 85.0098*, October 1990.
2. C.–R. Tsai and V. D. Gligor, "Distributed Audit with Remote Procedure Calls," in Proceedings of the 1991 IEEE International Carnahan Conference on Security Technology, Taipei, Taiwan, pp. 154–160, October 1991.
3. C.–R. Tsai and V. D. Gligor, "Distributed System and Security Management with Centralized Control," in Proceedings of the 1992 EurOpen/USENIX Workshop, Jersey, Channel Islands, pp. 137–146, April 1992, also available as *IBM Gaithersburg Technical Report 85.0155*.
4. C.–R. Tsai, "Security Issues on Distributed System Applications," in Proceedings of the 1993 National Security Conference, Baltimore, Maryland, pp. 385–390, September 1993.
5. J. G. Steiner, C. Newman, and J. I. Schiller, "Kerberos: An Authentication Server for Open Network Systems," in Proceedings of the 1988 Winter USENIX Conference, Dallas, Texas, pp. 191–202, Feb. 1988.

6. T. H. Dineen, et al., "The Network Computing Architecture and System: An Environment for Developing Distributed Applications," in Proceedings of the 1988 Summer USENIX Conference.
7. R. W. Scheifler and J. Gettys, "The X Window System," *ACM Transactions on Graphics*, 5:2, pp. 79–109, April 1986.
8. National Computer Security Center, *Department of Defense Trusted Computer System Evaluation Criteria*, DOD 5200.28–STD, December 1985.
9. R. Sandberg, "Design and Implementation of the Sun Network Filesystem," in Proceedings of the 1985 Summer USENIX Conference, pp. 119–130, June 1985.
10. IBM Corporation, *Communication Concepts and Procedures*,1990.