MVS-SAT : A Security Administration Tool to support SMF Protocol Data Evaluation

C. Eckert and Th. Stoesslein

*Munich University of Technology, Department of Computer Science
D-80290 Munich, Germany, eckertc@informatik.tu-muenchen.de*

Abstract

The paper presents a tool called MVS-SAT which supports extraction, off-line analysis and evaluation of security-related data recorded by the System Management Facility (SMF) running under MVS on IBM mainframe computers.

MVS-SAT consists of a PC and a MVS-host part. The PC part offers a simple user interface providing services to specify data sets which should be extracted from the SMF database by the host part of the tool. Furthermore, the PC-part offers services and databases to analyze and evaluate the extracted data via SQL-queries. Decoupling data extraction from its evaluation enables to implement the PC-part completely independent of any MVS version and to keep the tool extensible and adaptive.

Keywords

Security Management, Security Evaluation, Audit, Control, SMF

1 INTRODUCTION

Auditing of user activity usually serves as an effective preventive measure if a potential intruder is convinced that a security administrator will actually examine and analyze the audit data. Unfortunately, most audit logs maintain the recorded information in a format that is largely unreadable by a human being.

Automated intrusion detection tools (e.g. (Denning, 1986)) have been developed earlier to process and interpret the information that is recorded in audit logs. Intrusion detection is intended as means to identify potential malicious or undesirable activity. To accomplish this task intrusion detection tools need some kind of profiling describing typical and expected behavior of each user of the system. The main task of intrusion detection is to compare expected behavior with observed behavior. Hence, auditing mechanisms are required that record all information needed to perform this comparison.

Several utilities already exist to support audit and control of mainframe

computers. SMF (System Management Facility) (cf SMF manual SMF, 1995) running on IBM mainframe computers under MVS (cf Johnson, 1991) is an example of such a utility. With SMF a huge amount of information is recorded. Usually, this data is used for tuning the operating system and for performing accounting. But the data gathered can be exploited with respect to security investigations as well.

Though a huge amount of security-relevant information is available intrusion detection tools just exploit a fraction of this information leaving worthy information uninspected. Hence, within many computing environments a great amount of a security administrator's daily work is still spent with off-line analysis and inspection of system log and audit files to detect potential security problems.

Faced with this situation in a computing environment running IBM/MVS we developed a simple tool offering some support for the security administrator to inspect SMF data logs with respect to security relevant entries.

In this paper we will present MVS-SAT (the acronym stands for Security Administration Tool). MVS-SAT supports extraction, off-line analysis and inspection of security-related data recorded by SMF. The tool has been developed in cooperation with an organization being responsible for checking the security of the computing environment of banking companies.

Note, that our tool has been neither developed to detect security holes within the underlying operating system nor is it a tool to evaluate a system according to security evaluation criteria like the TCSEC criteria (DoD, 1985). In contrast, our tool is designed and implemented to support off-line investigations of a given system configuration, to analyze how the available security services are used and whether there is any evidence that security services can be circumvented by malicious users.

The rest of the paper is organized as follows. The next section gives a brief overview over MVS and SMF. Section 3 presents the main features of MVS-SAT and gives some examples to clarify how the tool can be used. Section 4 concludes the paper.

2 BACKGROUND

2.1 MVS

MVS (Johnson, 1991) is a mainframe operating system developed by IBM since 1974. Due to this long period of time MVS has undergone considerable changes while trying to keep compatible with existing versions of the operating system. As a consequence, MVS is by now a very complex software system. New system services have been continuously added to the existing kernel. These new features can be activated and configured via specific parameter

values at system startup time or dynamically during the lifetime of the system. Hence, system services can be activated and configured through thousands of system parameters where each parameter value can influence other system components in an unforeseen way leading to a completely uncontrollable system. Paans (Paans, 1989) provides detailed insight in the internal structures of MVS.

Due to the complexity of the system (more than 10 million lines of code) no verified source code can be provided. To give a little impression of what we are talking about: 9 times a year about 1500 patches for MVS and IBM supported products running under MVS are delivered.

As verification is not possible we must at least test and analyze the system and its main components like the MVS-kernel comprising for instance SMF, the Job Entry Subsystem (JES), and system library programs APF (authorized program facility) running in privileged mode.

RACF (Resource Access Control Facility) (cf RACF manual (RACF, 1995)), the software to control accesses to critical resources, is not part of the MVS kernel and can be configured and installed separately. Hence, an evaluation of MVS security must comprise an analysis of the correct installation and integration of RACF (cf Conyers, 1991).

The prerequisite to perform security tests is an appropriate audit facility that records relevant system activities. Audit and control of MVS are for example addressed in (Conyers, 1991) and (Soper, 1989). MVS contains three major audit facilities: (1) EREP, (2) SYSLOG and (3) SMF. With the Environment Recording and Editing Program (EREP) faults of the underlying hardware are recorded. SYSLOG gathers all activities concerning the operator terminal. Examples of such activities are start and termination of jobs or messages from RACF concerning for instance access denials. SMF will be addressed in more detail in the next subsection.

2.2 System Management Facility (SMF)

System Management Facility is the most important audit facility of MVS. SMF offers services to record data about resource consumption and about access behavior of single jobs (users), i.e. which files have been accessed and so on. At system startup-time the operator can determine which data should be logged by SMF but this configuration can be changed dynamically during system execution. The data recorded is first stored in data logs maintained in main memory and these logs are then periodically written to disk.

SMF knows 256 different record types (cf SMF, 1995) which do not have a common structure. To access a specific information within a data record the start address of the required section is needed. The computation of this address requires loading of the whole data record. The heterogeneity and lack of uniformity of the SMF record types is very cumbersome, as a tool that

tries to extract specific data from these records can not rely on any a priori information about position and length of the needed data being stored within a specific record.

Among other system components SMF audits RACF, RMF (Resource Measurement Facility), and JES. Information about SMF itself, that is about the selected values of the parameters to configure the facility are recorded in specific SMF record types as well. Depending on the selected parameter values different features of SMF can be activated or deactivated.

By now it should be evident that SMF records a lot of security related data like information about (1) all activities concerning system components being involved in performing access control. Especially, activities activating security related components or altering the configuration of security related components can be recorded. Recording data about which security services with which parameter values have been active during specific periods of time is very useful to detect potential security holes. (2) SMF gathers information about all actions concerning altering and management of access rights as well as (3) information about all unexpected accesses and access attempts.

Handling the amount of recorded data

A SMF log file stores in average 200.000 – 1.500.000 data records per 24 hours. Based on an average record length of 1Kbytes and an average amount of 500.000 records stored a day this results in memory requirements of 15GByte a month. The issue of system audit and control involves the task of inspecting and analyzing this huge amount of data.

To support the data evaluation only insufficient tools and facilities are available in MVS environments. One of the available facilities is the Dump Utility which allows to filter specific records and data sets. Major drawbacks of this facilities are that accessing an individual data field within a record is not possible and in addition, the output produced by the Dump Utility is nearly unreadable because it consist of raw hex representations.

Due to the huge amount of data the filtered data still contains an enormous amount of information. Even worse, most of this information is of no use for security evaluation, as it just records data concerning expected, that is, average behavior of users of the system. Using available filtering facilities a security administrator is still flooded with data which must be inspected 'by hand'. This is a time-consuming and error-prone task. Hence, the available utilities do not provide a pragmatic aid for the security administrator.

With respect to the evaluation of RACF data-sets the RACF Report Writer or other languages to query and evaluate RACF data sets can be used. Major problems with these facilities are that they are primarily directed towards tuning and that they are restricted to predefined record types and predefined evaluation criteria lacking flexibility needed with respect to more general security evaluations.

To summarize, depending on the given system configuration a SMF log may contain data concerning nearly all relevant system activities. The information is stored in an extremely complicated format, which aggravates evaluation considerably. SMF records enormous amounts of data per day which requires a proper filtering of the relevant from the irrelevant information where the amount of the latter preponderates the former considerably. Existing utilities provide just insufficient support to filter data according to criteria that can be specified dynamically and adapted to specific requirements of security audit and control. As security problems involve different components of a system it is necessary to combine information stemming from different data records, but as far as we know such a facility is not offered yet. In addition, existing utilities lack facilities to present the filtered data in a comprehensive and readable way.

3 MVS SECURITY ADMINISTRATION TOOL

3.1 Overview

We have designed and implemented a tool called MVS-SAT which supports the extraction and evaluation of security-related SMF-data. MVS-SAT offers a user interface which can dynamically be adapted to individual user requirements concerning the data to be extracted and analyzed.

A major issue in logging and auditing is to guarantee that all relevant data is actually gathered, i.e. that the log information is complete. As this requires detailed knowledge and deep insight in the complex relationships between different operating system components as well as knowledge about the features offered by audit facilities configuring the system appropriately can not be performed automatically by a tool. But an appropriate tool can offer means to facilitate this task in order to reduce the likelihood of overseeing important things due to the complexity of the task. A tool comprising certain security tests checking the security of a given configuration would be helpful. Such tools have recently been developed for network environments, e.g. SATAN. Unfortunately, those tools are not appropriate for mainframe configurations because they only offer a static set of test scenarios analyzing security holes stemming from well-known flaws within standard network services. Within our tool we decided to maintain a library of test scenarios which can be dynamically adapted and extended with user-defined security scenarios.

To check whether the selected configuration matches his security requirements the security administrator must investigate and analyze the audit data. This can hardly be performed automatically but the task can considerably be facilitated. In an average MVS installation the security administrator is typically faced with 10 - 30 million SMF-data sets per month filling hundreds of pages of a paper listings. The task of inspecting this huge amount of data

is very time consuming and error prone. MVS-SAT offers means to reduce this huge amount of protocol data by extracting data according to conditions specified by the security administrator. The extracted data is made available for the administrator via a graphical interface. In addition, the tool aims to serve as online manual presenting all information needed concerning SMF data record's structures in an easy to use and in a comprehensive manner.

MVS-SAT comprises two main components: a MVS host part and a PC part. Figure 1 depicts an overview over the components of MVS-SAT. Below the dotted line the components of the host part and above the line the ones of the PC-part are depicted. The grey boxes describe the software routines being implemented on the host and the PC respectively. The various databases maintained by the tool are given by ovals.

Figure 1 Overview over MVS-SAT

The main task of the host component of the tool is to extract desired data

records out of the SMF database. The filter program reads the extraction data file, containing the information needed by the program to extract the required data items from the SMF-database. The extracted data is written into the extracted data file and then transferred to the PC.

The main task of the PC component is to generate the extraction data file based on user-defined extraction conditions and to send this file to the host. After receiving the extracted data file from the host it is the task of the PC part to convert the extracted data into appropriate database formats enabling a SQL-based evaluation where SQL-queries are specified by the user. Finally, the PC part offers means to support the evaluation of the extracted data. Decoupling data extraction from its evaluation enables to implement the PC-part completely independent of any MVS version and to keep it extensible and adaptive. That is, the PC-part can easily keep path with for instance changing SMF versions.

3.2 PC-Part

The PC part of MVS-SAT provides a user interface offering a simple menu which can be used to handle the tool. Figure 2 sketches the main structure of the PC interface menus.

Figure 2 Menu structure of the PC interface

Usually, an evaluation of SMF audit data starts with the specification of the

so called extraction conditions. To edit user-defined extraction conditions the security administrator selects the `extraction condition` button of the menu and then the `edit` button of the sub-menu. After the user has finished the specification of extraction conditions he clicks the `start generator` button. This activates the filter generator program which compiles the specified conditions into the extraction data file and automatically transfers this file to the host. The host executes the filter program and sends back the extracted data.

Via the menu button `read transfer file` in the menu `evaluation` the user can now select the option to convert the transferred data into database formats. As a result a conversion routine is executed which transforms each SMF record type contained in the transferred file into a type-specific database. The extracted data is now prepared to be used.

The user can evaluate the extracted data by submitting specified SQL queries to the PC evaluation routine. Within the menu `SQL query` several sub-menus are offered to edit queries, to browse through an existing query database, to select one of these queries and to modify it in a syntactically correct manner. SQL queries can be stored in the query database for later reuse by using user-defined names. Each such database entry describes the security problem being addressed with the specific query. Hence, the database contains a library of predefined security scenarios to evaluate a specific configuration. A user of MVS-SAT can browse through this library of existing queries taking advantage of the accumulated knowledge of previously performed security investigations. As mentioned above, the database entries can be altered and enhanced via the `SQL query` menu as well.

User-defined Extraction Conditions

As already mentioned, a menu is offered to specify extraction conditions in a syntactical correct way. Each extraction condition may comprise several conditions. Each condition may concern a different SMF record type. Hence, the first step in specifying extraction conditions is to specify these conditions. In a subsequent step the user has to combine these conditions applying logical operators AND or OR. As our tool offers a quite comfortable user-interface the user must not cope with translating its conditions into the format that can be understood by the filter program. During the execution of the filter program each SMF database entry is checked whether it meets the condition.

To edit a condition a specific editing-mask is provided via the menu button `conditions` and sub-menu button `edit`. In addition to edit a new condition a user can browse through a pool of previously defined conditions, can select one of these conditions and can modify it if necessary. New conditions or modified ones can be saved under a new user-defined name for later re-use.

Example:

If you want to analyze whether RACF has been active during Initial Program Load (IPL) you can specify a condition that extracts all record types number 81 being logged at a time that lies within a specific time interval

because the information needed is held in type 81 records. $\triangle$

As mentioned above, each record type of a SMF data set contains different entries with a completely different structure. Hence, each condition must contain sufficient information to enable the selection of the required data items. To fulfil this requirement each condition contains the (1) **number of the record type** which should be analyzed, (2) **a list of record type entries** which should be checked against the condition, the specification of the condition itself consisting of an (3) **operator** (e.g. $=$, $\leq$) and of (4) a **comparison value** which specifies the criteria to be fulfilled.

Having specified the individual conditions, the whole extraction condition is created by composing conditions using logical operators. Again, condition composition is guided by a specific editing mask.

Example:

Imagine that a system administrator wants to know, who had initiated a specific job X in 1996. The required information can be retrieved from record type number 20. Job activation time is recorded in entry SMF20RSD of record type 20 and the name of the job is recorded in entry SMF20JBN.

The whole extraction condition specification consists of at least three single conditions:

$C1 = \text{SMF20RSD} \geq 0096001F$; $C2 = \text{SMF20RSD} \leq 0096365F$;

$C3 = \text{SMF20JBN} = X$;

The extraction condition is given by $EC = C1 \text{ AND } C2 \text{ AND } C3$. $\triangle$

Besides the conditions just explained each extraction condition specifies a list of record type entries which should be extracted from the SMF database. Whenever a recorded data set of the given record type fulfils the specified condition, the entries addressed with the entry list are extracted from the SMF database and stored in the extracted data file for later evaluation.

Example: (cont.)

Consider again the above example. The list of extracted entries comprises an entry SMF20UID recording the user identification UID of the user who had started the specific job X in 1996. $\triangle$

As it is very difficult for a security administrator to remember the meaning of the 256 different record types each comprising a lot of different entries our tool includes an online manual support describing every record type and its entry fields.

3.3 Host-Part

The MVS-host part of MVS-SAT comprises the filter program and a process which starts the program and creates the extracted data file. As our primary goal was to keep our tool compatible with as many MVS installations as

possible we implemented the filter program in 370/Assembler. In addition, an assembler implementation seemed necessary to perform extractions which involves browsing through huge amounts of recorded data with acceptable performance.

The functionality of the filter program is very simple. First, the header information of the extraction data file is evaluated containing for instance information about the used SMF version. As the filter generator generates extraction data tailored to the structure of a given SMF version, this version information is necessary to verify that the SMF version used on the MVS host is the right one.

Second, the filter program reads every recorded data set from the SMF-database into main memory and checks the specified extraction conditions. The requested entries are written into the extracted data file converting all data into ASCII format. As the host part just uses standard MVS routines it can be ported to different MVS configurations and it does not require any special host support. The data transfer between PC and MVS host is performed using sequential files. Using simple and uniform formats (raw ASCII data) instead of a specific database format keeps the tool as compatible with heterogeneous MVS installations as possible and enables great flexibility with respect to evaluating and analyzing the data on the PC.

3.4 Examples

In this subsection we want to sketch how MVS-SAT can be used to support a security administrator in accomplishing his daily work. We give two simple but pragmatic examples of security scenarios by briefly describing the underlying security problem and the SMF record types required to investigate the problem.

Example 1: Time intervals with SMF being deactivated

We are interested in answering the question whether the SMF audit facility was deactivated or activated during a specific period of time, for instance during 6. p.m and 7 a.m. on Friday the 6th of September 1996. Moreover, if there was indeed a deactivation we are interested to learn how long this deactivation occurred. As we have learned in the previous section we first must specify appropriate extraction conditions to gather the required data from the SMF database.

Whenever all data logs are filled and all data buffers are exhausted SMF raises data lost status and recording of data is no longer active. As soon as a new data log is available SMF writes a dataset of record type number 7 containing information concerning the time (SMF7STM) and the date (SMF7STD) when SMF was deactivated. Hence, SMF record type number 7 records some of the information we need.

We specify an extraction condition concerning entries of record type number

7 for which the conditions $SMF7STD = 9/6/96$ and $SMF7STM \geq 6$ p.m. and $SMF7STM \leq 7$ a.m. hold.

Type 7 records just record the beginning of a deactivation period, hence, we need some more information to accomplish our evaluation. As each SMF database entry comprises a time stamp recording its logging time, the information we need is provided by the database entry following immediately after a record type 7 entry. Hence, our extraction condition specifies that such data items should be extracted as well.

Evaluation is now very easy. We just look at the extracted data to detect the periods of time during which the audit facility was deactivated. In a subsequent step we can perform further investigations to learn about the events that caused these deactivations. $\triangle$

Example 2: Evaluating access attempts

We are interested in information about unauthorized access attempts to the system or accesses to system resources controlled by RACF. The information needed is recorded in SMF record type number 80 which gathers information about for instance access attempts, access denials, modifications of user profiles etc. The structure of record type number 80 is very complex and can not be described here. We rather explain its relevance for security evaluation by means of a simple example. To be able to distinguish between different events recorded with type number 80 we must investigate several fields. An unauthorized attempt to gain access to the system is signaled by the record entries SMF80DES and SMF80EVT by setting the values SMF80DES=1 and SMF80EVT=1 (Event Code=Job Initiation, TSO Logon). This simply means, that the user-id used to login is unknown to RACF. If an invalid password is used the record item SMF80EVQ=1 is set or if an invalid terminal is used this is recorded in record item SMF80EVQ=4 .

It should be clear how to proceed. We must specify an extraction condition concerning record type 80 data sets with conditions testing the values of the fields SMF80DES, SMF80EVT and SMF80EVQ depending on our specific requirements. $\triangle$

These simple examples should demonstrate that MVS-SAT enables to evaluate a broad range of security related problems in a comfortable and efficient.

4 CONCLUSION

We have presented a tool that supports system administrators to evaluate the security of a given MVS configuration based on recorded SMF audit data. MVS-SAT offers a simple but quite comfortable user interface to specify user-defined extraction conditions and to evaluate extracted SMF protocol data via SQL queries. The tool has been developed in cooperation with an organization

being responsible for checking the security of the computing environment of the members of the organization. The tool is in daily use since December 1995.

We tried to keep the tool as portable and independent of any specific MVS configuration as possible. This task was accomplished by our modular implementation, especially by separating the MVS-specific host part from the MVS independent PC part. Note, that our tool by no means is intended to replace a human being who is responsible to investigate security problems evaluating recorded SMF data. Our tool relies on the knowledge of its users. That is, the user must know the potential security holes of a system he wants to evaluate. He must have at least rudimentary knowledge of which information are stored within which SMF record types to investigate specific security problems. Our tool aims to support the user by offering databases and libraries maintaining as much information as possible a user needs to perform his evaluation. That is, previously acquired knowledge is kept in a quite comfortable manner for later reuse.

Though desirable, our tool does not yet support an administrator with hints how to improve his MVS installation. Future work will investigate this task. In addition, we intend to extend our tool by incorporating an on-line intrusion detection facility.

5 REFERENCES

- Conyers, V.L. (1991) *Audit and Control of MVS*. EDP Auditors Foundation.
- Denning, D.E. (1986) An Intrusion-Detection Model. *IEEE Transaction on Software Engineering*, SE-13(2):222 – 232.
- Department of Defense. (1985) *Trusted Computer System Evaluation Criteria*. Technical Report CSC-STD-001-83, Department of Defense.
- Johnson, R. (1991) *MVS Concepts and Facilities*. Intertext/McGraw-Hill.
- Paans, R. 1989) *A Close Look at MVS Systems: Mechanisms, Performance, and Security*. North-Holland
- RACF. (1995) *RACF1.9 Security Administrator's Guide*. Manual SC28-1340, IBM.
- System Management Facility (SMF). (1995) *MVS/ESA System Management Facility*. Manual GC28-1628, IBM.
- Soper, K.R. (1989) *MVS Top-to-Bottom: An Analysis of MVS and its Impacts on Audit and Security*. MIT Training Institute.