Stu Henderson's Clear Explanation of Effective z/OS Security Auditing

(A Brief Description of the Steps to a Proven Practical Audit Program, Without Much Technical Detail)

Stu Henderson
The Henderson Group
5702 Newington Road
Bethesda, MD 20816
(301) 229-7187
www.stuhenderson.com

Abstract:

IS (Information System) audits are sometimes not well received or are thought to be irrelevant. In this session, Stu shows you a practical audit program for MVS (z/OS) security that produces meaningful findings and recommendations. The approach described here will result in more effective audits and can be used to good advantage by security administrators and system programmers as well. Stu shows you how to change the focus of an audit from a judgmental, checklist-based approach to one that emphasizes impartial evaluation of the tools available to systems management.

I Introduction

This paper shows you how to conduct z/OS mainframe audits, specifically security audits of IBM's MVS operating system software for mainframe computers. (Note that z/OS is a package of software programs which includes the MVS operating system. MVS is comparable to UNIX or Windows. It is the program that starts up when the computer is turned on. It controls all users, programs, and resources on the computer.)

This paper does not address security software (such as **CA ACF2™ for z/OS** (**CA ACF2**), **CA Top Secret® for z/OS** (**CA Top Secret**), or IBM's **RACF**) except as they affect operating system security.

Please note that a weakness in MVS security will undermine the reliability of the security software. In the other direction, weak security software implementation will undermine MVS security.

The approach we use here avoids rote checklists and criticisms of how a given installation functions. We start instead with the hardware controls that form the basis of MVS security. IBM gives us written assurance that MVS reliably uses these hardware controls to prevent users from interfering with each other and with MVS itself. IBM also gives us several standard techniques (or "**backdoors**") for system programmers to add programs to the system with privileges which bypass the hardware controls. (This paper will not explain the details of the hardware controls, nor the details of the techniques such as **APF** (Authorized Program Facility) authorization used to give programs these privileges. For more details on this, please see the *Further Information* resources at the end of this paper.)

We will illustrate our approach with examples from two software products: **CA Auditor for z/OS** (**CA Auditor**, which many people still call by its old name: **CA Examine**) and **CA Compliance Manager for z/OS** (**CA Compliance Manager),** both from **CA Technologies**, which used to be called **Computer Associates**.

Who This Is For and What You Should Expect

This paper is designed for IS (Information Systems) auditors who will be conducting mainframe audits, as well as system programmers and security administrators, and those responsible for self-assessment reviews in advance of scheduled audits. It assumes some basic knowledge of mainframes and auditing.

Readers will find that there is no in-depth explanation here of how the hardware controls and backdoors work. That would require a much larger document. Instead, readers will learn the basic steps to an effective MVS security audit, with the understanding that any needed technical knowledge can be acquired elsewhere and fit neatly into the framework presented here.

The Hardware Controls and the Integrity Statement

IBM mainframe computers (the "z series") have three hardware controls which form the basis of all MVS security. We list them here with a brief description of what each one does:

- The **Supervisor State Switch** (restricts when a program can execute privileged hardware instructions [such as the instruction to change the date and the instruction which writes directly to a disk drive])
- **Protect Keys** (restrict what memory a program can update or read)
- **Address Spaces** (restrict what memory a program can touch)

The MVS operating system uses these three controls to build a **virtual cage** around each program running on the computer. This virtual cage prevents each program from interfering with other programs executing at the same time, and also from interfering with MVS itself.

These hardware controls and the virtual cages MVS constructs from them are the basis of MVS security. This architecture is so solid that IBM provides us with written assurance that no program can break out of its virtual cage unless you modify the system to permit this. This assurance is in the form of ***IBM's Integrity Statement for MVS***, details of which may be found at the end of this document.

IBM also provides several standard methods for you to modify the system to give a specified program privileges which permit it to break out of its virtual cage. Such programs (called "**privileged programs**" or "**back doors**") can bypass all security on the system, including that provided by **CA ACF2**, **CA Top Secret**, and IBM's **RACF**.

Such modifications to the system are not covered by ***IBM's Integrity Statement for MVS***. IBM rightly considers data center management responsible for ensuring that these modifications do not introduce security exposures to your system.

The standard ways IBM gives us to permit a program to break out of its virtual cage are:

- User Supervisor Calls
- APF Authorization and TSO APF Authorization
- I/O Appendages
- Functional Sub-Systems
- Exits (assembler or REXX language programs which can modify the logic of standard software)
- The Program Properties Table
- Various methods (such as SRB scheduling) to cross address space boundaries. This audit program will not center on address space backdoors, since they are effectively controlled by automatic means.

A typical mainframe installation may have hundreds of such backdoors. The concept of a backdoor is not good or bad. It is practical. (You may have a backdoor on your house. Your insurance agent expects you to lock it at night.) We need to be able to know however that they don't introduce security exposures to our systems.

Our audit approach assumes that ***IBM's Integrity Statement*** provides us with sufficient assurance that the MVS security architecture can be relied upon. So we start by identifying all the instances where the system programmers have added privileged programs to the system, since these are not covered by the Integrity Statement.

What We Are Auditing and How

Our job as auditors is not to evaluate these privileged programs. Rather it is to **evaluate the tools available to system programming management** for them to know that the backdoors:

- Have All Been Approved (that is, should be there)

- Are Safe (which means "doesn't permit unauthorized users or programs to break out of their virtual cages)

- Cannot Be Modified Without Authorization and Detection

Answers to these questions will help the financial auditors to address their own questions regarding numbers processed on the mainframe:

- Are the numbers reliable?
- Are assets protected (including information assets and information processing assets)?
- Is the organization in compliance?
- Is it a going concern (likely to stay in business in the future)?

So What Is the Risk?

The essential issue here is control of the ability to update Key Datasets, since any programmer who can update them can add or modify backdoor programs that bypass all the security on the system, including that provided by **CA ACF2**, **CA Top Secret**, and **RACF**. The **Key Datasets** are those where the backdoors are specified and those where they reside.

Here is how the different types of backdoor relate to the Key Datasets:

- User Supervisor Calls (these programs reside in the dataset SYS1.NUCLEUS or in the LPALIST datasets
- APF Authorization and TSO APF Authorization (these programs all reside in datasets which have been flagged as APF-authorized)
- I/O Appendages (acquire their privileges through APF authorized datasets)
- Functional Sub-Systems (acquire their privileges through LINKLIST datasets)
- Exits (assembler or REXX language programs which can modify the logic of standard software) (Many exits reside in APF-authorized or other system datasets.)
- The Program Properties Table (Programs listed in this table only receive privileges if they reside in APF-authorized libraries.)
- All these backdoors are specified in the parmlib datasets

In other words, any programmer who can update (write to) such datasets can introduce or modify programs which can bypass all the security on the system. We want to know that management can provide reasonable assurance that these programs do not introduce security exposures to the system.

II The Audit Program

Our audit program examines both what is happening on the system right now and also the procedures which are meant to ensure good system security. By concentrating on established, agreed-upon facts, we avoid the subjective opinions and disparaging tone of voice which can otherwise make IS audit reports irritating to read.

Here are the steps to our audit program:

1. Purpose and Control Objectives
2. Identify the Datasets Where the Backdoors Are Specified and the Datasets Where They Reside
3. Evaluate the Security Software Rules Covering These Datasets
4. Evaluate Logging, Reporting, and Review of Updates to These Datasets
5. Evaluate the Process for Adding / Modifying Backdoor Programs
6. Determine “How Does Management Know That the Backdoors Are Safe?”
7. Determine “How Does Management Know That Each Backdoor on the System Has Been Approved?”
8. Determine “How Does Management Know That Each Backdoor Can’t Be Modified Without Approval?”
9. Summarize, Review, Report

1. Purpose and Control Objectives

Again, the purpose of our audit is to evaluate the tools available to system programming management for them to know, and for them to be able to demonstrate, that the backdoors on the system:

- Have All Been Approved
- Are Safe
- Cannot Be Modified Without Authorization and Detection

These tools can be:

- **Procedures** such as those for evaluation of how safe system software is and for controlling system software changes
- **Restrictions**, such as rules in security software restricting who can modify system datasets
- **Monitoring** such as review of reports describing system software changes

You should be able to relate these objectives to the financial audit control objectives.

2. Identify the Datasets Where the Backdoors Are Specified and the Datasets Where They Reside

The **Key Datasets** are the ones where the backdoors are specified and the ones where they reside. You need to learn their dataset names in order to learn who can change them. They include: the parmlibs, the APF authorized datasets, the LPALIST and LINKLIST libraries, and certain standard Key datasets. (Note that a **library** is just a type of dataset containing programs, lists of options, or other information.)

The backdoors are all specified by your system programmers in datasets called **the parmlibs**. Originally there was just one parmlib, always named **SYS1.PARMLIB**. IBM has since made it possible to have several datasets serve together as parmlibs, each with a different name. You will start your data gathering by identifying which datasets are the parmlibs.

When the MVS operating system starts up, it reads the parmlibs to determine which backdoor programs to add to the system. MVS then builds internal tables in memory describing the backdoors and where to find them. Each backdoor is a program with privileges that let it bypass all the security on the system.

(Any programmer can browse these tables to learn what backdoors are on the system. If any of these backdoors introduces a security weakness, it can then be abused by any programmer who knows enough to find it and analyze it.)

The backdoors reside in other datasets, including standard ones such as **SYS1.LINKLIB**, **SYS1.LPALIB**, **SYS1.NUCLEUS**, and **SYS1.SVCLIB**. They also include others such as the **APF** (Authorized Program Facility) datasets, the **LPALIST libraries**, and the **LINKLIST libraries**.

There are 3 basic ways of learning the names of these datasets:

A. Browse the Parmlibs the Same Way MVS Does to Learn the Names of the Key Datasets

B. Browse the Tables in Memory Where MVS Stores the Names

C. Use Software to Browse the Tables and Report on the Names

A. **Browse the Parmlibs the Same Way MVS Does to Learn the Names of the Key Datasets**: You can learn which backdoors have been specified, and what datasets they reside in, by reading the information from the parmlibs. You can learn the names of the parmlib datasets by issuing the operator command **DISPLAY PARMLIB**. The IBM manuals "*MVS Initialization Tuning and Reference*" and "*MVS Initialization and Tuning Guide*" describe the contents of the parmlibs and how to use them. (You start by reading the members whose names begin **IEASYS..** , which point to other members where the backdoors are specified.) A complete description of how to interpret the parmlibs is beyond the scope of this paper. This approach requires understanding of how to read the parmlibs and how the backdoors function. We will not address this approach any further in this paper.

B. **Browse the Tables in Memory Where MVS Stores the Names**: The starting point for this is address 16 in memory which has the address of the **CVT** or Communications Vector Table, which is considered the "mother of all MVS control blocks". This requires use of a tool to browse memory. It also requires knowledge of the control blocks MVS uses, and the ability to add hexadecimal numbers. This approach will not be addressed in this paper.

C. **Use Software to Browse the Tables and Report on the Names**: Your security software (**CA Top Secret** or **RACF**, but to my knowledge not **CA ACF2**) may report a few of the dataset names. For a complete list, you need a software tool designed to report on all the backdoors, all the datasets where they are specified, and all the datasets where they reside. We will use the best-known of these, **CA Auditor**, to illustrate our approach.

To get a partial list of key datasets from your security software in a **CA TopSecret** installation, you can use the **TSSAUDIT** utility program to produce a list of many of the backdoors, including User Supervisor Calls, APF Authorized datasets, and the Program Properties Table.

To get a partial list of key datasets in a **RACF** installation you can use the **SELECTED DATASETS REPORT** and the **PROGRAM PROPERTIES TABLE** report from the **DSMON** utility. This will give you the Program Properties Table, as well as the names of APF-authorized and other key datasets.

An easier, and more comprehensive method is to use the program **CA Auditor**, (which originally was named **CA EXAMINE**). This program browses all the control blocks in memory where MVS has saved the specifications for backdoors. After this, **CA Auditor** prints a report listing the names of the parmlibs, all the backdoors, what datasets they reside in, and much more information about the system.

To help you see how these reports work, here are edited examples of **CA Auditor** reports showing: the names of the parmlibs and various backdoors, including: the User Supervisor Calls, APF Authorized datasets, and the Program Properties Table:

```
   CA AUDITOR PARMLIB DATASET INFORMATION
               SYSTEM AND PARMLIB DATASET INFORMATION
 --------------------------------------------------------------------------------
 STATIC SYSTEM INFORMATION
 ================================================================================
  …
  MASTER CATALOG = ICF.MASTER.STU06
        VOLUME  = STUA06  ADDRESS = 1A6B
  MASTER JCL SOURCE IS SYS1.LINKLIB(MSTJCL00) ON VOLUME STU002
  …
CURRENT PARMLIB CONCATENATION
 ================================================================================
…
  LOGICAL PARMLIB DATASETS SPECIFIED IN LOAD06 MEMBER INCLUDE:

    1. SYSSTU2.MYSTUFF.PARMLIB                  ON VOLUME STU003
    2. SYS1.PARMLIB                        ON VOLUME STU007
```

Abridged Illustration of CA Auditor Report Listing the Parmlib Datasets. Large type indicates the dataset names and volumes.

```
         CA AUDITOR    USER SVC ANALYSIS

  THERE ARE 13  ACTIVE AND 43  INACTIVE USER SVCS DEFINED ON THIS SYSTEM.

  ENTER S BESIDE SVC NUMBER FOR MORE INFORMATION, OR D TO UPDATE DESCRIPTION.

  ENTRIES RECOMMENDED FOR REVIEW ARE MARKED "*".
    NUMBER                                      AR
    DEC HEX  TYPE  ACTIVE  APF   LOCKS      MODE       DESCRIPTION
 -   --- ---  ----  ------    ---     -------      ----       ------       --------------------------
    204   CC      2        NO       NO                        NO        INSTALLATION DEFINED SVC
    205   CD      3        YES      NO     NO LOCK   NO        INSTALLATION DEFINED SVC
    206   CE      2        NO       NO                        NO        INSTALLATION DEFINED SVC
    209   D1      2        NO       NO                        NO        INSTALLATION DEFINED SVC
    210   D2      3        YES     NO     NO LOCK    NO         INSTALLATION DEFINED SVC
    227   E3      3        YES     NO     NO LOCK    NO        IBM'S NETVIEW PROGRAM PRODUCT
    241   F1      2        NO       NO                        NO        INSTALLATION DEFINED SVC
  * 242   F2      3        YES     NO     NO LOCK    NO        INSTALLATION DEFINED SVC
    246   F6      3        YES     NO     NO LOCK    NO        INSTALLATION DEFINED SVC
    247   F7      2        NO       NO                        NO        INSTALLATION DEFINED SVC
    248   F8      2        NO       NO                        NO        INSTALLATION DEFINED SVC
    249   F9      2        NO       NO                        NO        INSTALLATION DEFINED SVC
    250   FA      4        YES      NO    NO LOCK    NO        CA-ASM2 ALLOCATION MANAGER
  * 251  FB       3        YES      NO    NO LOCK    NO        INSTALLATION DEFINED SVC
```

Abridged Illustration of CA Auditor Report Listing User Supervisor Calls
(Only entries with YES in ACTIVE column are active. The programs which are the User Supervisor Calls usually reside in either the dataset SYS1.NUCLEUS or in one of the LPA libraries.)

```
            CA AUDITOR    APF LIBRARY STATISTICS SUMMARY

 ----------------------------------------------------
  PRESS ENTER FOR DETAILED DISPLAY.

 +------- APF LIST INFORMATION ---------+-------  LINK LIST INFORMATION -------+
 |                                      |                                      |
 |  LIBRARY NAMES SPECIFIED:  771       |  APF LIBRARIES SPECIFIED:  85        |


        CA AUDITOR   APF LIBRARY STATISTICS

  ENTER B NEXT TO NAME TO BROWSE, F TO FREEZE, OR A FOR SECURITY ACCESS.

                                          ---APFLIST---                -MEMBERS--
  NAME                                    --AUTH WITH--    AUTH VIA    TOTAL
  VOLUME COMMENTS                         SMS   VOLUME    LINKLIST      AC=1      PCT
 - ------ -----------------------------   -----  ------    --------      -----     -----
  CICSCA06.CTS220.CICS.SDFHLOAD                                          799
  STU009                                   NO     YES       NO             0      0.00
 ---------------------------------------------------------------------------
  CICSCA06.CTS320.CICS.SDFHAUTH                                          70
  STU008                                   NO     YES       NO             6      8.57
 ---------------------------------------------------------------------------
  REXX.SFANLMD                                                            2
  STU007                                   NO     YES       YES            0      0.00
```

Illustration of Abridged CA Auditor Report Listing the APF Authorized Datasets. Top of page shows counts of APF Authorized Datasets. Bottom Part Shows Listing of Their Names.

```
   CA AUDITOR   PROGRAM PROPERTIES TABLE ANALYSIS

 THERE ARE 403   PROGRAMS DEFINED IN THE PPT. ITS VERSION ID IS: 0
 ENTRIES MARKED "*" ARE RECOMMENDED FOR REVIEW.
 ENTER I OR S NEXT TO PROGRAM NAME FOR ADDITIONAL INFORMATION OR LIBRARY SEARCH.

                                DATASET                                SMF          PREF
   PROGRAM            WHERE     INTEG        SECURITY  NON-            TIMING  CPU  STOR
    NAME      SOURCE  FOUND     BYPASS  KEY  BYPASS  CANCEL SWAP  BYPASS AFFN  FLAG
 -  --------  ------  --------  ------- ---  -------- ------ ----  ------  ----  ----
   AHLGTF     IBM    IEFSDPPT   NO      0      NO      YES    NO    YES     ALL   001
   AKPCSIEP   IBM    IEFSDPPT   YES     1      NO      NO     NO    YES     ALL   001
   BPXBATA2   IBM    IEFSDPPT   NO      2      NO      NO     YES   NO      ALL
 * SMFDUMP    USER   SCHED13    NO      8      YES     NO     YES   NO      ALL



        CA AUDITOR     PPT LIBRARY SEARCH

 403  MEMBER(S) WERE SELECTED FOR THIS SEARCH OF ALL ELIGIBLE PPT LIBRARIES.
   ENTER B NEXT TO PROGRAM NAME TO BROWSE, F TO FREEZE.

   PROGRAM      LINK DATE       SIZE     VOLUME         LIBRARY NAME
 - --------      ---------      ------    ----         ---------------------------------
   DFHSIP        10/11/02       146620   STU06A        CICSSTU3.CTS220.CICS.SDFHAUTH
   DSNYASCP      04/19/01       004E90   CICST9        SYS2.DB2510.SDSNLOAD
   EZBTCPIP      09/11/09       00BAC8   STUMVS        TCPIP.SEZALOAD
```

Illustration of CA Auditor Reports Listing Program Properties Table
(Only programs which reside in APF authorized datasets are affected. Top of this listing shows programs described in the table. Bottom part lists datasets or libraries where they reside.)

3. Evaluate the Security Software Rules Covering These Datasets

Whether your security software is **CA ACF2**, **CA Top Secret** or IBM's **RACF**, list the dataset rules for the Key Datasets identified in the previous step. From the rules, determine who can update the datasets. These datasets should include:

- The parmlibs
- The APF (Authorized Program Facility) Datasets LPA (Link Pack Area) Libraries and often LINKLST libraries
- SYS1.LINKLIB, SYS1.NUCLEUS, SYS1.LPALIB, and SYS1.SVCLIB

Using **CA ACF2** as an example, here is the command to list the rules for all datasets whose names begin with SYS1., along with an extract of the resulting report. The first line starting **PARMLIB** describes who can update (that is, WRITE to) the dataset SYS1.PARMLIB, that is everyone whose UID string matches the pattern **xxx**. (A **UID string** is an ACF2 construct made up of various fields from the user record in the ACF2 database, for example, Department, Jobcode, and Employee Number. The second PARMLIB line permits everyone whose UID string is any three characters followed by **YYY** to READ and WRITE this dataset.

```
ACF
SET RULE
LIST SYS1
*ACCESS RULE SYS1 STORED BY STU ON 12/28/10-15:32
$KEY(SYS1)
* -------------------------- *
*    SYS1.PARMLIB ACCESS
* -------------------------- *
 PARMLIB  UID(XXX)                      WRITE(L)
 PARMLIB  UID(***YYY)                   READ(A) WRITE(A)
* -------------------------- *
```

Illustration of CA ACF2 commands to list dataset rule for SYS1 with abridged listing of output. (Listing begins with the line *ACCESS RULE SYS1 ….
Any line beginning with an * is a comment. Enlarged type added to highlight the PARMLIB lines.

Of course, in ACF2 you need to issue follow-on commands to learn all the users whose UID strings match **XXX**:

```
ACF
SET LID
SET VERBOSE
LIST UID(XXX)
```

CA ACF2 commands to list all LIDs (LOGONIDs, that is, userids) whose UID string matches XXX

Using **CA Top Secret** as an example, here is the command to list the rules for all datasets whose names begin with SYS1., along with an extract of the resulting report. Each pair of lines starting with XAUTH describes one permission. The second pair for example states that the ACID (user) STU001 has total (ALL) access to the dataset named **SYS1.SVCLIB**.

```
TSS WHOHAS DSN(SYS1)

DATASET    = SYS1                              OWNER(STUDEPT)

 XAUTH     = SYS1.                             ACID(*ALL*   )
   ACCESS  = READ


 XAUTH     = SYS1.SVCLIB                       ACID(STU001 )
   ACCESS  = ALL


 XAUTH     = SYS1.SVCLIB                       ACID(STUSBUDS )
   ACCESS  = ALL


 XAUTH     = SYS1.SVCLIB                       ACID(STUSPALS )
   ACCESS  = ALL
```

Illustration of CA Top Secret Command to List Dataset Rule for SYS1, with abridged listing of output. (Output begins with line DATASET = SYS1. Blank lines and enlarged type added for readability.)

Using IBM's RACF as an example, here is the command to list the rules for all datasets whose names begin with SYS1., along with an extract of the resulting report. The UNIVERSAL ACCESS is the default access. The line starting STU002 says that STU002 has update access. The line below that says that STUSPALS have read access to the dataset.

```
LISTDSD PREFIX(SYS1) ALL

INFORMATION FOR DATASET SYS1.SVCLIB (G)

LEVEL  OWNER     UNIVERSAL ACCESS      WARNING    ERASE
-----  --------  ----------------      -------    -----
00     STU001    NONE                  NO         NO



AUDITING
--------
SUCCESS(UPDATE)



ID                ACCESS
-------           -------
STU002            UPDATE
STUSPALS          READ
```

Edited and Abridged Illustration of RACF Command to List Dataset Rule for SYS1, with abridged listing of output. (Output begins with line INFORMATION FOR. Blank lines and enlarged type added for readability.)

Listing the dataset rules from the security software tells you who can update these datasets.

If you used CA Auditor to identify the Key Datasets, you can use its **ACCESS** command (or type an **A** next to the name of the dataset of interest) to list the contents of the dataset rule protecting the dataset. This works whether you have **CA ACF2**, **CA Top Secre**t, or IBM's **RACF**. An example follows on the next page.

```
                CA AUDITOR    APF LIBRARY STATISTICS SUMMARY

 ------------------------------------------------------

          CA AUDITOR   APF LIBRARY STATISTICS

  ENTER B NEXT TO NAME TO BROWSE, F TO FREEZE, OR A FOR SECURITY ACCESS.

                                    ---APFLIST---                  -MEMBERS--
  NAME                             --AUTH WITH--   AUTH VIA      TOTAL
  VOLUME COMMENTS                   SMS   VOLUME   LINKLIST      AC=1        PCT
 - ------ ---------------------------- -----  ------    --------      -----       -----
 A  CICSCA06.CTS220.CICS.SDFHLOAD                                   799
    STU009                           NO     YES       NO               0        0.00



                DATASET ANALYSIS AND ACCESS INFORMATION

 ------------------------------------------------------------------------------

 Data Set Security Analysis For ACF2
 ACCESS Subcommand Results as of 01/01/11-14:10 for: CICSCA06.CTS220.CICS.SDFHLOAD

 Key: CICSA06

 Ruleline: - VOL(STU***) UID(SALES***) READ(A) WRITE(L) EXEC(A)
 Lids: STU007   STU27A   PAYROLL   SYSPROG1
```

Illustration of Abridged CA Auditor Report with A Typed in Requesting ACCESS, that is,   Dataset Rule Listing, Followed by Listing of the Matching Dataset Rule in ACF2.. Top of Page Shows Listing of APF Authorized Dataset with A on extreme left. Bottom Part Shows Resulting Listing of the Dataset Rule, Combining Two Audit Steps into One.

4. <u>Evaluate Logging, Reporting, and Review of Updates to These Datasets</u>

You want to know whether management is informed of every update to these datasets, since any such update can be the addition or modification of a privileged program. (Privileged programs can bypass all security on the system.) For management to be informed of every such update, the update must be logged to SMF or detected in some other fashion. The security software rules can specify that updates get logged or not.

In the **CA ACF2** example, look at the two lines starting PARMLIB. One of them gives the permission as W(A) and the other as W(L). The W stands for WRITE. The **A for "Allow, but don't bother logging**". The **L for "Allow, but log**".

In the **CA Top Secret** example, the word AUDIT on an XA line would indicate that accesses by that permission get logged. Other ways to cause logging in **CA Top Secret** include the AUDIT record and attributes on user ACIDs.

In the **RACF** example, options to cause logging include options in SETR LIST (the settting of RACF options), the AUDIT operand on the dataset rule (as in the example), and the GLOBALAUDIT operand on the dataset rule (not illustrated), and options on user records..

You should be able to conclude that either all updates to these datasets get logged, or not. If they are all logged, then ask to see the resulting report. Ask the manager of system programming to describe how she reviews the report. Ask specifically how she knows that each update described in the report has been approved. (For example, are there change control tickets or similar documents that could be compared to the list of updates.)

Review any standards or procedures which require all such updates to be logged, reported, and reviewed.

5. <u>Evaluate the Processes for Adding / Modifying Backdoor Programs and for Monitoring Changes to Them</u>

You will want to evaluate use of:

- Formal Procedures
- Security Software Restrictions on Updates
- Monitoring of Updates
- Automated Tools to Alert Management of Changes
- Program Signing

Formal Procedures

Review any written procedures or standards describing the processes for control of changes to Key Datasets, and interview system programmers to see how they work. Summarize whether unauthorized backdoors could be on the system and whether they could be added to the system without detection.

Security Software Restrictions on Updates

These procedures will have to rely on the security software rules restricting who can update Key Datasets, as discussed in section 3 above. Summarize who can update them, and whether updates get logged.

Monitoring of Updates

Monitoring doesn’t prevent unauthorized changes, but can make it possible for management to be aware of unauthorized changes shortly after they happen. For monitoring to be effective, the following will all be necessary:

- Each update to one of these Key Datasets is logged: to SMF, to the SYSLOG, to the security software log, or to some other reliable log file

- The log records are processed every day to produce a report of who made changes to key datasets

- The report is read, and properly processed, which means that each such change in the report is compared to some standard, such as a change control ticket or a written approval

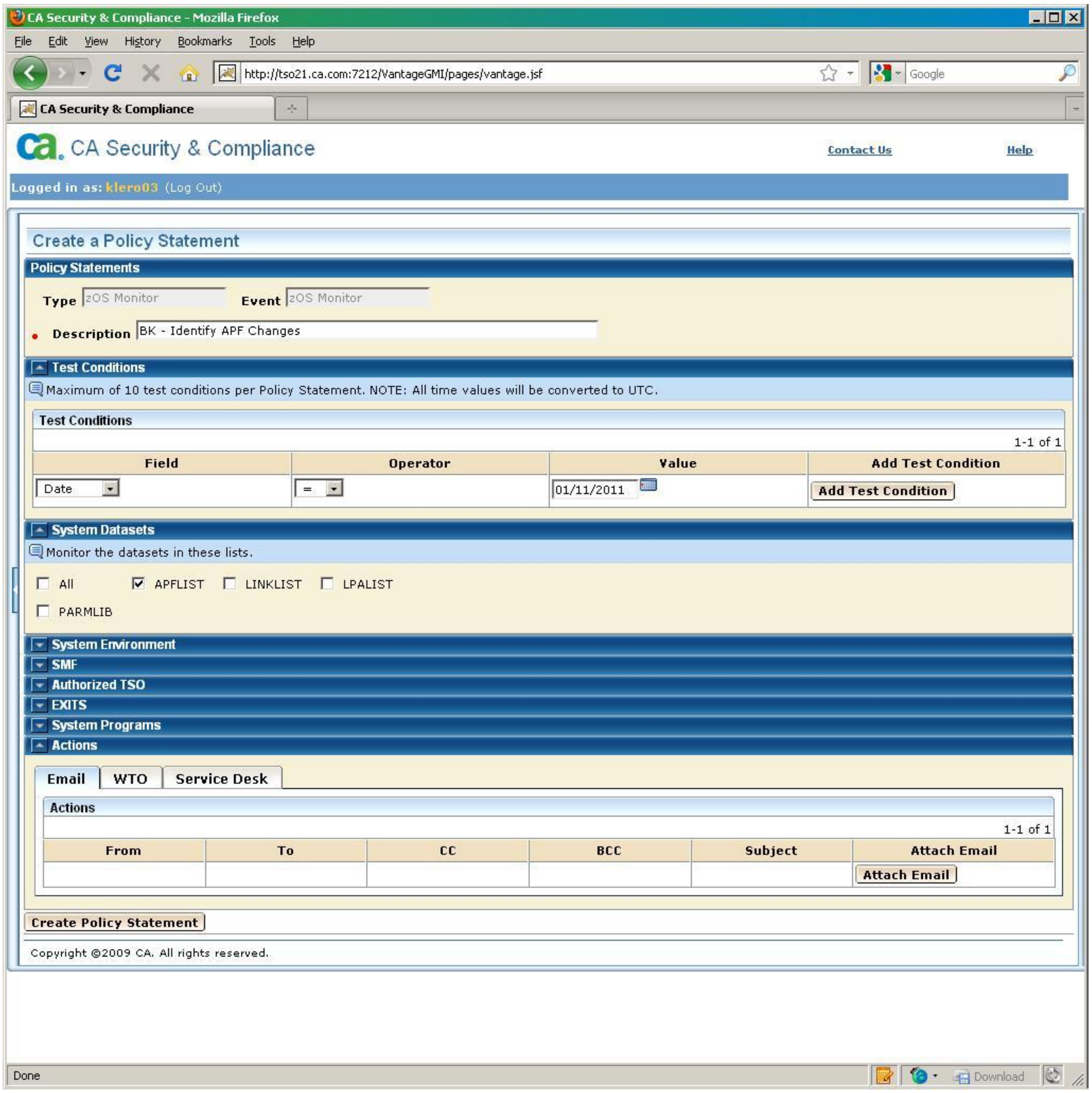
Automated Tools to Alert Management of Changes

In addition, the processes to control changes to Key Datasets may make use of software tools to detect and report (often in realtime) on any changes to these Key Datasets. For such tools to be effective, they need to:

- Detect all changes to Key Datasets
- Provide alerts or realtime notifications to management

CA Auditor provides one approach to detect changes: it collects and records a baseline listing of various backdoors on the system. Subsequent checks compare the current backdoors to this baseline, reporting any changes.

We illustrate another approach with a different product: **CA Compliance Manager**. This product can monitor changes to Key Datasets based on the direction you give it. It can monitor these types of dataset: LPALIST, LINKLIST, APFLIST, and PARMLIB. This includes monitoring changes both to the lists of Key Datasets and monitoring of changes to the datasets themselves. (**CA Compliance Manager** also monitors other types of changes, such as security software option settings. These are however outside the scope of this paper.)


CA Security & Compliance - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://tso21.ca.com:7212/VantageGMI/pages/vantage.jsf
Google
CA Security & Compliance
Contact Us
Help
Done
Download

Logged in as: klero03 (Log Out)

Create a Policy Statement

Policy Statements

Type: zOS Monitor Event: zOS Monitor

● Description: BK - Identify APF Changes

Test Conditions

Maximum of 10 test conditions per Policy Statement. NOTE: All time values will be converted to UTC.

Test Conditions

1-1 of 1

Field	Operator	Value	Add Test Condition
Date	=	01/11/2011	Add Test Condition

System Datasets

Monitor the datasets in these lists.

☐ All ☑ APFLIST ☐ LINKLIST ☐ LPALIST
☐ PARMLIB

System Environment
SMF
Authorized TSO
EXITS
System Programs
Actions

Email | WTO | Service Desk

Actions

1-1 of 1

From	To	CC	BCC	Subject	Attach Email
					Attach Email

Create Policy Statement

Copyright ©2009 CA. All rights reserved.

Illustration of CA Compliance Manager Panel. Note the Checkboxes Halfway Down the Screen Below the Line "Monitor the Datasets in These Lists". The Checkboxes Permit Setting of Policy to Monitor Any of: APFLIST, LINKLIST, LPALIST, and PARMLIBs.

Program Signing

Program signing is a way of validating that a given program has not been modified, and that it comes from the correct, trusted source. It uses Message Digests (a mathematical method of generating a unique number based on the contents of a program, and then encrypting the unique number [called a "hash total"] in a way that can only be decrypted with the vendor's public key. The details of this process are beyond the scope of this paper.)

IBM has provided within MVS automated methods to use these hash totals to ensure that critical programs have not been modified and that they actually come from the suppliers they are supposed to come from. In the future, use of these methods may become widespread in identifying unauthorized changes to privileged programs.

6. <u>Ask "How Does Management Know That the Backdoors Are Safe?"</u>

There are two basic approaches to know that a privileged program is "safe", that is, that it doesn't introduce security exposures to the system. The first is to have formal review of the source code of the privileged program, based on the principles IBM gives us for writing safe privileged programs. These principles are provided in IBM classes and also in the IBM manual, "***MVS Programming:*** **Authorized Assembler Services Guide**".

The second approach is needed when the software supplier is not willing to make source code available for review. In that case, it makes sense to ask the vendor for an integrity statement for their product comparable to ***IBM's Integrity Statement for MVS*** (Please note that CA Technologies provides a comparable Integrity Statement for all of their software products, available on their website at:
http://arcserve.com/~/media/Files/TechnicalDocuments/common-integrity-statement.pdf .)
If they are not willing or able to provide this, would you want to pay them annual fees to run their software on your system, giving their software privileges that bypass all security on your system? If the system programming manager is not asking vendors for integrity statements, the question arises "what is she then doing to know that all privileged programs are safe?".

Your evaluation will consist of reviewing procedures, documentation of formal reviews, and integrity statements from vendors. Inquire whether there are other techniques in practice for management to know the backdoors are safe.

If management does not have the tools to know and to demonstrate that the backdoors are safe, what effect does this have on the financial audit?

7. **<u>Ask "How Does Management Know That Each Backdoor on the System Has Been Approved?"</u>**

The only way to know this is to have a comparison of the actual backdoors on the system to an independent list of what has been approved, that is what should be on the system. The standard could be a written list, a change control database, or software that keeps track of changes made to the system, along with links to approval documentation.

If system programming management does not agree with this, then offer to take the list of backdoors and add one fake entry, and then to ask the manager how she would identify the fake entry.

The **CA Compliance Manager** software product illustrated above gives you the ability to take different actions when a change is detected, depending upon whether the change has been approved. This product can also provide an automated link as well to change control or other mechanisms for documentation of change approvals.

8. **<u>Ask "How Does Management Know That Each Backdoor Can't Be Modified Without Approval?"</u>**

Review the security software rules to see who can update the key datasets, and whether updates are logged and reviewed in a meaningful fashion. Evaluate any use of program signing (described above) to verify that privileged programs haven't been modified improperly.

If management uses software which tracks changes to the system, you will want to determine that the software covers all key datasets. Some such programs permit you to maintain a list of key datasets to be monitored. Unless there is a reliable method to keep the list up-to-date, this approach will be insufficient. For software to provide complete assurance that all key datasets are being monitored, the software should maintain this list automatically.

9. **<u>Summarize, Review, Report</u>**

You can now summarize your answers to these three questions, based on the information you have collected: Does management have adequate tools available to them for them to know and to be able to demonstrate, that all backdoors on the system not covered by ***IBM's Integrity Statement***:

- Have All Been Approved
- Are Safe
- Cannot Be Modified Without Authorization and Detection

Consider the effect of your findings on the financial audit and what you should be communicating to the financial auditors.

Include practical recommendations where needed. (A good test of reasonableness of each recommendation is to ask yourself if you would be happy being responsible for carrying out each recommendation.)

Review a draft of the report with the system programming manager and any other interested parties. Invite corrections if you have missed any facts. Offer to consider changes to wording which might make the report easier to receive. If you have concluded that there could be more or better procedures or tools available to management, ask management whether they would like to have such tools.

Invite comments on how your recommendations could be improved.

Edit your report on the basis of what you have learned, and submit it.

For More Technical Follow-On: If you have the technical knowledge, you might extend your audit to include several more advanced topics:

- **Dynamic changes to backdoors**. After MVS starts up, additional changes or additions to backdoors may be introduced by a variety of methods, including operator commands, APIs (Application Programming interfaces), and by other privileged programs. These are detected by software which scans control blocks (such as CA Auditor and CA Compliance Manager), but not by techniques which merely browse the parmlibs.

- The **proclibs** (which contain the **JCL** (Job Control Language) for started tasks). Any programmer who can modify this JCL can cause rogue programs to execute with the privileges of the started tasks.

- **Additional datasets** necessary for protecting system software, such as: the SMF datasets (log files) and security software files

- **APF authorization for USS** programs, which can involve file attributes or the sanctions list.

The techniques and tools describe above can easily be extended to address these advanced topics. However, since our audit approach concentrates on the tools available to management, the advanced topics may safely be left for a follow-on audit.

Examples of Two Different Audit Comments (With a Common Beginning)

Example A: One Hundred thirty nine programs and datasets (other than those provided by IBM) have been added to this computer system with privileges that permit them to bypass all security on the system. Such programs and datasets are commonly called "backdoors". In this audit, we evaluated the controls available to IT management for them to know and to be able to demonstrate that these backdoors:

- Are "safe", that this they cannot be abused to access data improperly
- Cannot be modified without authorization and without detection
- Have been approved by IT management.

We determined that 17 programmers are able to modify these backdoors and to add additional ones without authorization and without being detected. Information Systems management does not maintain an independent list of which backdoors have been authorized. We were therefore unable to determine whether the backdoors which have been added to the system are ones that should be on the system.

Information Systems management does not require software vendors to provide integrity statements (formal assurance that the software does not introduce security exposures) for these backdoors. Information Systems management does not maintain formal documentation of security reviews nor of other evidence that such programs are safe.

Because other data centers have often found examples of such privileged programs that introduce serious security exposures, Information Systems management may want to improve controls over MVS security.

Financial audits which rely on the accuracy and confidentiality of information maintained on these computers will need to consider this in their evaluation of information security.

Example B: One Hundred thirty nine programs and datasets (other than those provided by IBM) have been added to this computer system with privileges that permit them to bypass all security on the system. Such programs and datasets are commonly called "backdoors". In this audit, we evaluated the controls available to IT management for them to know and to be able to demonstrate that these backdoors:

- Are "safe", that this they cannot be abused to access data improperly
- Cannot be modified without authorization and without detection
- Have been approved by IT management.

We found that management has formal documentation of security reviews and integrity statement from vendors for all such privileged programs. Our review of a sample of twenty such documentation packages found …

We found that security software rules in this data center prevent unauthorized updates to these backdoors, and cause management to be notified of any changes to them. Management uses the following methods to verify that only approved backdoors are on the system….

We conclude that the security of the MVS system software in this data center may reasonably be relied up to support the security software. (A separate audit addresses how well the security software protects sensitive production data.)

III Summary

We have shown a practical approach to auditing MVS security, relying on evaluation of the controls management has available to them to maintain the security of the system. You should be able to relate your findings from this directly to the financial audit objectives.

For Further Information:

- **On IBM's Integrity Statement for MVS**:

First issued in 1973, IBM's MVS ™ System Integrity Statement, and subsequent statements for OS/390® and z/OS, has stood for over three decades as a symbol of IBM's confidence in and commitment to the z/OS operating system. IBM reaffirms its commitment to z/OS System Integrity.

IBM's commitment includes design and development practices intended to prevent unauthorized application programs, subsystems, and users from bypassing z/OS security – that is, to prevent them from gaining access, circumventing, disabling, altering, or obtaining control of key z/OS system processes and resources unless allowed by the installation. Specifically, z/OS "System Integrity" is defined as the inability of any program not authorized by a mechanism under the installation's control to circumvent or disable store or fetch protection, access a resource protected by the z/OS Security Server (RACF®), or obtain control in an authorized state; that is, in supervisor state, with a protection key less than eight (8), or Authorized Program Facility (APF) authorized. In the event that an IBM System Integrity problem is reported, IBM will always take action to resolve it.

- **IBM Manuals:**
  - ***MVS Programming: Authorized Assembler Services Guide*** SA22-7608-15
  - ***MVS Initialization and Tuning Reference*** SA22-7592-21
  - **IBM Principles of Operations** SA22-7832-01
  - RACF Manuals

- **CA Technologies Manuals (available at support.ca.com)**
  - CA Technologies Integrity Statement for All of Their Software Products (http://arcserve.com/~/media/Files/TechnicalDocuments/common-integrity-statement.pdf)
  - CA Auditor Manuals
  - CA Compliance Manager Manuals
  - CA ACF2 Manuals
  - CA Top Secret Manuals

- **On MVS security architecture: www.stuhenderson.com**

- **About the Author:** Stu Henderson is an experienced system programmer, auditor, trainer, and consultant, specializing in IBM mainframe computers. He is editor of the ***RACF User News*** and the online ***Mainframe Audit News***. He teaches seminars nationwide, both public sessions and in-house. His website www.stuhenderson.com contains a wealth of articles and useful links.