MVS: Mainframe Virtual Security[1]

Brian Currah
BDC Computer Services Inc.
RR # 1
Picton, Ontario
CANADA K0K2T0
613-476-6670
currah@ibm.net

Abstract: Enterprises using IBM's **premier** operating system for System/390 mainframes, MVS/ESA, may have a false sense of confidence regarding the vulnerability of corporate assets. The protection offered by widely used security products can often be circumvented as a result of loopholes in other **add-on** software. Many such systems are frequently not nearly as resistant to security penetration as corporate management have been led to believe.

New inspection, certification and verification techniques must be implemented before MVS-based systems can be employed as a **safe haven** for the conduct of electronic commerce and a secure repository for corporate and national assets.

1 Introduction

The nature of computer security concerns has changed significantly over the last several decades. From the mid '70s until the mid '80s, there was a significant focus on mainframe security related issues. The major MVS-based security products (IBM's RACF and Computer Associates ACF2 and TOP SECRET) were introduced during this period and eventually almost all MVS systems utilized one of these or a similar product for centralized protection of mainframe resources. As telecommunications usage increased, concerns over electronic wiretapping or eavesdropping were addressed by cryptography-based solutions. Then, along came personal computers and for many years the focus of so called hackers (or more appropriately — crackers) has been in the areas of viruses, worms, Trojan horses, etc. The majority of recent concerns have focused on the increasing use of the Internet, the need for firewalls and secure methods of transacting business in this environment.

However, in the last several years, there seems to be increasing concern regarding areas many of us may consider too futuristic to be taken seriously. Seminar discussions or research papers on topics such as electronic economic espionage or the interests of organized crime in computer related theft may sound appropriate for scholarly research or speculation by criminologists but of little concern for everyday corporate computing.

If you think these topics are a little far fetched, you will undoubtedly be inclined to rank the topic of electronic warfare as totally science-fiction. However, it appears that a number of high profile organizations (primarily government and military, but far from exclusively these) are seriously considering such possibilities. In early September 1996, the Fifth International Information Warfare Conference was held in Washington, DC. The conference theme was *Electronic Civil Defense for the 21st Century*. If you visit the web site for the National Computer Security Association at ***www.ncsa.com*** and select the InfoWarCon topic, the list of session topics and speakers should help convince you that a number of agencies and corporations are treating the subject as more than purely academic. In spite of very impressive (and sometimes scary) titles for a number of sessions, it appeared that none addressed the subject of attacks against mainframes or network servers by insiders. Does everyone just assume that these systems are secure? Or is the assumption being made that they are uninteresting to those who may pose a threat. Maybe the lack of interest is because those concerned think some other security oriented organizations or agencies are addressing this area. Pinning your security expectations on any of these assumptions may inevitably lead to unanticipated risk and exposure.

You might be inclined to think that the title of this paper is a gross exaggeration intended purely to capture your attention. If it caught your notice and brought you this far, that's what was intended. However, the adjective describing the state of MVS security is not an attempt at witticism. The extent to which the term **virtual** overstates the security risk is largely a function of how **vanilla** your MVS system is. If your own trusted systems programming staff installed an IBM-supplied MVS integrated package (such as an IPO or PDO) and ***no*** other **outside** system or application software products have been installed and ***no*** modifications or enhancements have been made to the system, then you are probably pretty safe — not guaranteed 100% secure, but who cares — you can't do anything very productive, or risky, with such a system. Otherwise, the degree of safety or risk is a function of the particular software (yours or others) which has been added to the base level system provided by IBM. The simple truth is that some products provide no added security risks at all while others contain varying degrees of **hacker opportunities**, some of which will give your auditors and security officers nightmares and ulcers.

2 But I Thought MVS Was Supposed to be Secure?

Many people including users, programmers, system managers and especially management who are responsible for corporate asset protection may be (and frequently are) under the mistaken impression that MVS, with the addition of security controls, is an inherently safe platform. In all too many cases, this perceived protection is illusional. It's true that IBM has stated that MVS has **System Integrity** and it's also true that some levels of MVS were evaluated and found to comply with U.S. Department of Defense guidelines and requirements for **Trusted Computer Systems** at both C2 and later B1 levels. However, several important caveats apply to both situations:

(1) IBM never said MVS was bullet-proof. They said that previously known exposures had been addressed, that new development would follow strict guidelines, and that if additional exposures were reported, they would perform the same corrective action as is used for all other types of programming errors.

(2) The DoD evaluation included only a limited number of IBM Program Products in addition to the MVS/ESA base (Version 3 level), and is not necessarily representative of what a typical customer would use in today's environments.

(3) Very few commercial MVS-based add-on products (other than security resource control products) have been submitted for formal TCSEC evaluation even though many widely used products contain sensitive interfaces which must be resistant to tampering or misuse.

(4) Nobody's perfect!

The DoD TCSEC program is being extended through a joint NSA/NIST effort called Trust Technology Assessment Program. Detailed information on this program can be obtained from the NIST web site (refer to the references section at the end of this paper for the URL). One important aspect of this program is that it will use accredited commercial laboratories to perform C2 level evaluations on a wider range of products and in a more timely fashion.

3 MVS Security and Integrity Interdependence

The requirements for achieving a fully secure IBM MVS-based system are frequently misunderstood and in most cases, corporate management and security administration staff consider only those aspects that relate directly to the security control product being used such as IBM's RACF or CA's ACF2 or TOP SECRET.

The situation can be thought of as being similar to a bank which houses a vault containing cash and securities. Most of the security and theft prevention effort is concentrated on items such as:

- the thickness of the steel doors of the vault,
- secure front and rear doors on the building,
- steel bars on the windows,
- surveillance cameras on the entrances, or
- electronic access control devices for after-hours entry.

Such controls give the appearance of tight security and will likely deter inexperienced robbers. However, an experienced professional criminal who knows the value of the vault contents and is determined to break in will likely ignore most of the above items and use whatever resources he can muster to find the most vulnerable weak spot. Construction notes or blueprints may reveal that a utility tunnel or sewer runs under the vault. Electric wiring diagrams may reveal a means of short circuiting the vault's lock system.

It is obvious that the primary theft prevention mechanisms are only effective when constructed upon a solid, tamper-proof foundation with supporting services such as electrical and telephone wiring which are also secure.

An MVS system can be pictured in a similar manner. The security product is similar to the doors, bars, locks, cameras and access control devices. However, all of these are effective only when the underlying hardware and software are equally tamper resistant.

4 Security Design

The MVS operating system, related systems software products, and user applications can be thought of as being implemented by different levels or layers of programs are shown in Figure 1.

The basic hardware protection mechanisms related to security are:

privileged instructions many machine codes which are used to instruct the computer on certain basic operations are restricted so that they can be used only by designated **supervisory** programs, and

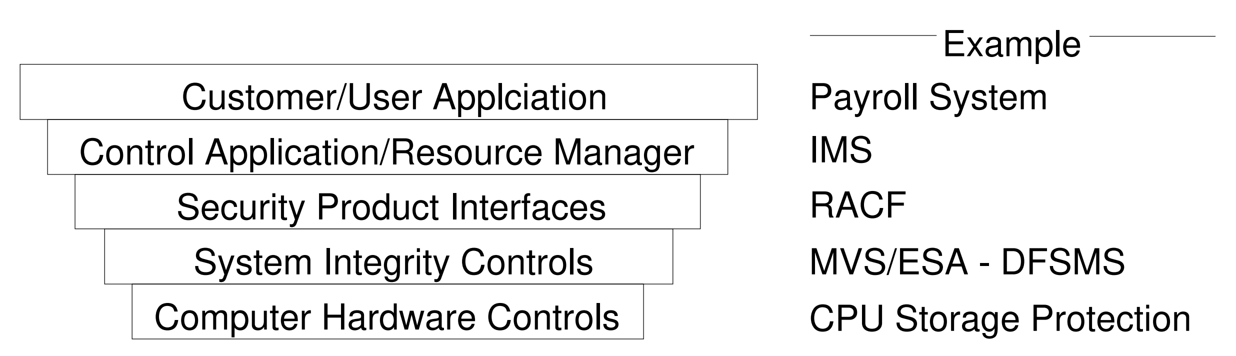

Figure 1: MVS Security Levels of Abstraction

storage protection — the computer's internal **memory** is divided into separate sections which are assigned unique identifiers or **storage keys**. Programs are assigned one or more **access keys**. Storage can only be altered by programs which have been assigned the capability to use the associated key(s). Read access to individual 4K blocks of storage can be either unrestricted or limited to programs with a matching access key. In addition, the architecture provides a **master** key of zero which allows both read and write access to all areas of storage (except for some system areas which are protected from modification for reliability purposes).

The responsibility for ensuring that these basic system controls can not be circumvented or corrupted lies within the **foundation** layer of the operating system called the **control program**. This attribute of MVS is termed **System Integrity** and IBM generally defines it as **the property of an operating system that prevents the circumvention, alteration or subversion of its security mechanisms**. The IBM publication *MVS Security* (GC28-1439) contains a more detailed definition of System Integrity and descriptions of associated controls and procedures.

The most important aspect of the MVS security design is that System Integrity provides the strong foundation which must first be in place and be properly maintained in order to have confidence that all other security control mechanisms are safe from tampering.

So far, we have viewed this situation considering only the IBM supplied **operating system** (MVS/ESA and DFSMS) and a security product such as RACF or ACF2. However, almost all commercial customers and many military and government users of MVS have installed add-on products from IBM and other mainframe software vendors to perform functions not provided by the base MVS operating system. Examples of such products are compilers, telecommunications support, data base systems, sort/merge applications, file transfer utilities, etc. These types of products are typically referred to as either **systems software** (which can be thought of as extensions to the operating system like IMS in the illustration above) or **applications software** — like the payroll system above.

However, from a system integrity and security standpoint, what really distinguishes one type of software from another is whether or not it contains **authorized** code. The term authorized has a very different meaning with respect to MVS integrity than its usual connotation used in conjunction with a security product. From the view of a security product, it is users who are authorized or not to perform a function. By creating security definitions (or access rules), users can be permitted by the security administrator to:

- access a specific computer system,
- use one or more specific terminals,
- read or update specific computer files (**data sets** in MVS parlance), and/or
- execute a specific program.

In the case of MVS and its system integrity controls, it is programs and not individuals that may be authorized or not. MVS provides several forms or subsets of authorization, but before covering these we need to review some IBM System/390 architecture fundamentals.

5 System/390 Program States and Storage Protection

As mentioned above, the basic S/390 instruction set is divided into two major groups — ordinary (non-privileged) instructions which can be executed by any program and restricted (privileged) instructions which can only be used by certain programs. S/390 provides two types of program execution environments — problem state and supervisor state. Programs operating in problem state may only execute non-privileged instructions while those running in supervisor state are allowed to execute both non-privileged and privileged instructions. In addition, S/390 provides a mechanism for subdividing its internal storage and allowing only certain programs to read or update some areas of storage. S/390 provides 16 levels of storage isolation which are not hierarchical; they are termed **keys** — storage keys and execution (program) keys. The sixteen available keys are numbered 0 through 15. Programs running with an execution key of 1 to 15 may only read or update areas of storage which have a matching key. Programs running with an execution key of 0 may read or update storage in any key. MVS uses the S/390 storage protection facility for both system integrity and reliability purposes. The sixteen available storage keys are divided into two groups — keys 0 to 7 are reserved for use by the control program and related extensions and keys 8 to 15 are for use by application programs. The system keys (0-7) are allocated for use by various elements of the operating system and this provides a degree of isolation which prevents one component from inadvertently damaging storage allocated to another one.

6 SVCs: A Primary Source of Integrity Exposures

The S/390 architecture provides one non-privileged instruction which allows an ordinary application program (one which executes in problem state) to pass control to another program which is intended to execute in supervisor state. This specialized instruction is called Supervisor Call or SVC. MVS itself contains many (100+) SVCs which are defined to pass control to parts of the operating system to perform specialized and/or restricted functions which require the use of privileged instructions and also involve use of system key (0-7) storage areas. A few examples of MVS's use of SVC functions are:

SVC 0 start an operation to an external input/output device

SVC 1 wait for a predefined event to occur

SVC 4 allocate a block of internal (virtual) storage

SVC 5 release a block of internal (virtual) storage

SVC 8 load a program into main storage from external disk

SVC 9 delete a program from main storage

MVS also allows a customer to define new SVC's to perform their own specialized functions; such extensions to MVS are termed **User SVCs**. Many vendor supplied products (usually those in the **systems software** category) also include one or more programs which must be installed as a User SVC.

MVS provides a specialized SVC function which is used to allow certain **controlled** programs to change their execution status from problem state to supervisor state and/or to change its execution key to any desired value. This SVC is called MODESET and to be eligible to use this sensitive function, a program must meet two requirements:

(1) the program must reside in a special category of external disk storage file called an APF library (APF stands for Authorized Program Facility)

(2) the program must have been explicitly designated as requiring access to restricted functions when it was created (by means of an **authorization code** set by the MVS linkage editor utility).

Any program which is permitted to use the MODESET SVC can potentially switch its execution mode to supervisor state and key zero. When it does this, the program is subsequently able to issue any privileged instruction and is also able to read or modify any area of the system's internal storage.

7 System Integrity Vulnerability

One of the most technically challenging areas affecting system integrity is the design and implementation of program code for operating system functions which provide interfaces between applications (unauthorized programs) and the low level operating system components which, due to their functional requirements, must execute in supervisor state and one or more of the **system** keys of 0 to 7.

These special **programming interfaces** are most frequently implemented using an SVC instruction (SVC) and an associated service routine which performs the low level function. A similar facility is also available using another instruction called Program Call (PC). Hence, the integrity exposures discussed in this section are equally applicable to service routines invoked through a program call instruction.

Figure 2 provides an overview of an application program to SVC service routine interface.

(1) The application program executes in problem state and key 8. It can only execute non-privileged instructions and can read or update key 8 storage. It can also read (but not alter) storage in keys 0-7 and 9-15 if such areas are not **fetch protected**, and

(2) The SVC routine executes in supervisor state and key 0. It can execute all instructions (privileged and non-privileged) and can read or update any area of storage (by intent or accident).

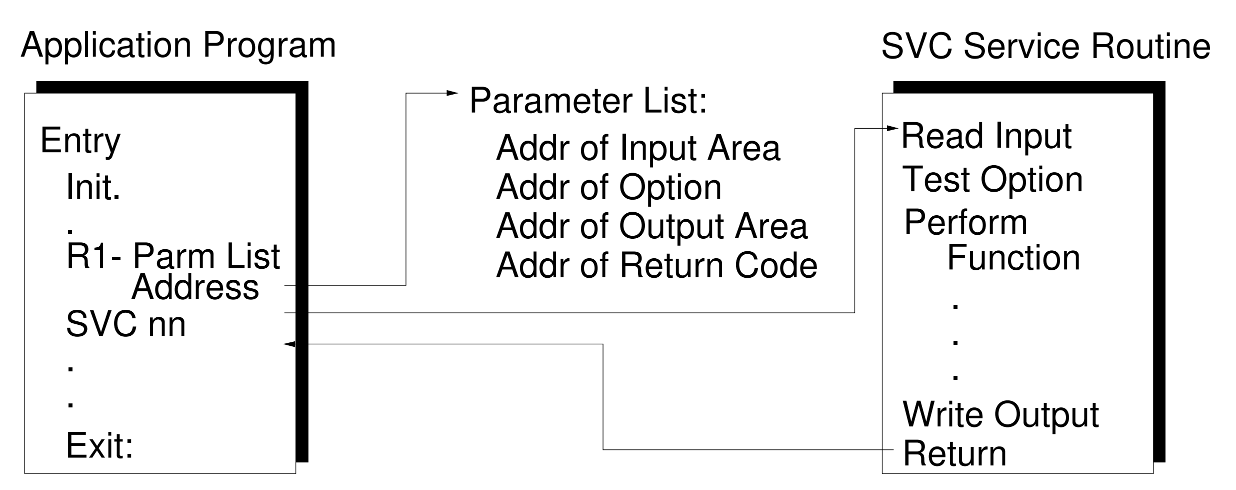

Figure 2: Application Program to SVC Routine Interface

The input and output areas should be within the application program or in dynamically acquired storage owned by the application, either of which should be key 8 storage. If the SVC routine **trusts** the application to provide valid input and output addresses, potentially serious damage may occur if the application program accidentally provides an incorrect value. If the incorrect address happens to point to a valid storage area but one which has a storage key of 0 to 7 or 9 to 15, the code in the SVC routine will still be successful when it attempts to access or update the designated storage area since it is executing in key zero and is thereby allowed to access storage in any key. In such a case, there would likely be two results:

(1) the application program would fail or produce incorrect results because the wrong input data was used or the results were not placed in the expected area

(2) some other application or product or operating system component may fail or produce incorrect results because an area of storage belonging to it was inadvertently modified by the SVC routine.

If a scenario such as this actually occurred (and they do — frequently), the most likely action to be taken would be to correct the application program to ensure that it provides correct input addresses to the SVC. However, what we really have in this simple example is a dangerous security exposure **waiting to happen** or more likely **waiting to be exploited**.

The critical factor is that the implementation of the SVC routine did not anticipate (or protect itself against) invalid input information from application programs. If such an error can happen by accident, then it presents the opportunity for the flaw in the SVC to be intentionally exploited with the intent of causing damage to or modification of areas of storage that the application program itself is not allowed to update.

8 Exploiting an Integrity Exposure

Here is an example of how such a problem can result in serious security problems. Typically, the customer's security administrator has a security classification which allows access to any disk or tape file without needing explicit authorization to the named data set (this is sometimes called **Superuser Authority** on non-MVS systems). All users of the system have a user identification (USERID), which is not secret. When accessing the system (LOGON or LOGIN), some form of USERID authentication is done, most often in the form of a confidential password. Now, assume that a programmer is aware of a problem in an SVC routine similar to the one described above, perhaps from accidental discovery while developing and testing a new application. Also assume that this person knows or can readily obtain the USERID of the security administrator. The only thing the programmer needs in order to masquerade as the security administrator is the required password. The programmer also knows from prior experience or from examining IBM MVS error diagnosis books that MVS retains user passwords in internal storage, but in a storage area in key 0-7 so that they are not accessible to ordinary applications. With this knowledge, the programmer then creates a new application program in which the **Address of Input Area** is intentionally set to point to the area containing the security administrator's password. It then invokes the desired SVC routine which, in turn, moves the password value back to the application program's output area. The programmer can code the program so that the returned password is displayed on a terminal or printed in a hardcopy listing. Having the required password, the programmer can now enter the system at a convenient time and pose as the security administrator, now with the ability to access any data desired.

A variation of this scenario could be the development of a similar program but one which instead uses a false **Address of output area** which is intentionally set to point to an area of storage containing critical MVS control information. When this MVS control data is damaged, it will result in a **crash** of the entire system.

At the time of discovering the **integrity exposure** in the SVC, the programmer may not have the inclination or motivation to actually obtain any confidential data or cause system damage, but may very well go ahead and develop the **exploiting** application anyway as a harmless prototype just for the satisfaction of proving the theory or demonstrating superior technical skill.

9 Spreading the Risk

This person is now in possession of a dangerous **time bomb** which can be resurrected from their utility program toolkit at any point in the future if a financial or beneficial opportunity arises or if the individual wants to inflict damage to the system or corporation (due to employee firing, company takeover, political or ideological differences, etc.) Consider the situation if the programmer who discovers this exposure subsequently reveals (or more likely brags about) the technique to associates or friends within or outside of the company. Further, what if the newly discovered subversive technique is posted on a hacker bulletin board or Internet web site or presented in a trade journal?

With respect to a specific customer's MVS systems, the scope of vulnerability is directly dependent on the origin and distribution of the **exposure-prone** SVC routine. If it is a **User SVC** which was created by a systems programmer at the subject installation and is only installed on MVS systems owned and operated by this client at one or more sites, only those particular systems are at risk.

However, if the original systems programmer who created it thought that the SVC routine contained significant utilitarian value to other users, it is entirely possible or likely that it may have been passed on to others, with or without customer management's knowledge and consent. Such redistribution of non-proprietary programs is frequently done through IBM User Groups like Guide, Share or Common, or through common interest groups like CMG or NaSPA. Once this occurs, it is unpredictable as to which MVS systems may be exposed since such exchanging of system programmer **neat tools** is frequently done and is often not tightly controlled by security administrators or auditors, except in the most security conscious installations. Even in the later case, the system programmer at the receiving site is likely unaware of the presence of the exposure and can convince the security staff of the benefits of installing it for its utility value and/or cost savings in not having to develop from scratch something that already exists and is free.

In other cases, the **exposure-prone** SVC routine may be part of a commercial product such as a database or telecommunications system. The level of code containing the exposure may be present in all versions of the product or might be limited to specific versions or releases created on or after a particular date. In these cases, the vulnerability is present in all MVS systems with this particular product or the affected versions installed. There have also been cases where the client decided at some point in time to switch from **Product X** to **Product Y** but no one ever bothered to remove or disable the SVC routine(s) that were originally installed as part of Product X. *After all, it isn't used, so what's the harm?* the system programmer might respond, if asked why the obsolete SVC is still present.

10 Types of Integrity Exposures

System integrity errors caused by failures to adequately validate user supplied storage addresses are the ones which most frequently occur but this class of errors is just one of many generic types which can lead to integrity and security exposures. Within those related to lack of storage address verification, there are many possible variations. In the example above, we considered the case of an SVC providing access to a protected password (typically eight bytes); this type of error is called a fetch protection violation. The other type which involves modification by the SVC of storage not owned by the application is called a store protection violation and need not involve an area as large as eight bytes. Modification of one byte at a location which can be specified directly or indirectly by the application program can have serious consequences. In fact, the modification of a single bit may be all that is necessary to result in the application program subsequently being able to do absolutely anything it desires! A one bit change, if directed to the proper place could potentially allow the following:

- change the application's mode from problem state to supervisor state,
- change the application's execution key from 8 to 0,
- give the program APF authorization (so that is then able to use the MODESET SVC or any other SVC whose use is restricted by APF), or
- give the user a level of security authorization normally reserved for use by a security officer or computer operator.

There is another class of integrity and security problems which is not related to implementation errors in the code of SVC routines. This is the commonly encountered **Authorization SVC**. Such a routine may be developed by a customer or vendor when creating an application or product which does not have a requirement for the whole application to be APF authorized for the entire execution time period but at the same time does need the temporary ability (often just during the startup or shutdown of the program) to use a restricted MVS function. In such cases, the developers often create a **User SVC** whose purpose it is to modify MVS internal control information to allow such access. Many SVC routines which do this have no other function except to enable or disable APF authorization for the subject application/product and can usually be easily recognized as **potential suspects** due to their small size and limited function. In some cases of such Authorization SVCs, the developers place no restrictions in the routine on which specific application(s) may invoke the SVC, and (naively) rely on the fact that since its name and function is not documented in product publications, its existence (and security/integrity potential) will be secret. For other implementations, the developer requires the requesting application program to supply some (supposedly) secret password or authorization key in a specific register or parameter list field before it will activate APF authority for the task. However, the password or key must also exist in the executable code of the SVC routine itself for comparative purposes and can usually be easily discovered by even a moderately skilled programmer. In some slightly more sophisticated implementations, some form of masking or encoding is used to hide the **secret key**, but such attempts just present an additional level of challenge to a more determined or experienced attacker.

In a few cases, the developer recognizes the potentially dangerous potential of the Authorization SVC routine and attempts to create more complex schemes to restrict its use. Very few of such implementations are completely successful in preventing unintended usage. It now becomes an issue of the knowledge and ingenuity of the developer versus the skills and determination of the hacker. Most often, the latter wins!

Just as in the situation described earlier, Authorization SVCs are often shared between business associates (both internal and external) or redistributed via computer user groups and online bulletin boards. As an example, there is a group called NaSPA — it originally stood for National Association of System Programmers of America; it is now named The Association for Corporate Computing Technical Professionals. This group creates a number of CDROMs containing PC and mainframe shareware code, articles, tips and techniques. One of these is for IBM System/390 users and Version 3 of this CD is currently available to association members for $75. The cover says *thousands of programs and miscellaneous mainframe software that can make support personnel's professional life easier and more productive*. Certainly true, but also consider the fact that this collection includes, by my estimate, probably a dozen varieties of authorization SVCs and numerous SVC routines for other purposes, few of which, if any, have been implemented according to the stringent requirements of IBM's System Integrity Guidelines. Please note that the above reference to NaSPA and its distribution of member supplied code is not intended to be critical of this organization or its membership. The issue is simply that what may be expedient at one point may have unintended consequences later.

As you can see from these examples, the code for an SVC routine must be implemented very carefully and in accordance with a general set of *System Integrity Guidelines* published by IBM. While the guidelines cover all of the general areas of concern and describe the common pitfalls and implementation techniques required, in actual practice the development of such routines requires a variety of technically complex methods which IBM does not describe in detail. Due to its own experience in these areas and also the level of internal education it provides to its designers and developers, IBM written SVC routines have generally been of a higher quality with respect to system integrity than many of those written by customers or other vendors of MVS-based products.

The types of integrity exposures discussed previously are the ones which occur most frequently and are also usually the easiest for a cracker to detect. Other more complex types involving multitasking, asynchronous processing or falsification of internal data structures have been detected in penetration studies but a detailed description of such techniques is beyond the scope of this paper.

11 Vendor Support for MVS Integrity

IBM has supported **system integrity** for its MVS operating system ever since it was first released in 1975, after having gained extensive experience during efforts to **harden** the security facilities of two earlier operating systems (MVT for S/360 and SVS for S/370).

IBM has made a specific commitment to its MVS customers regarding MVS **System Integrity** errors; if any system function or interface presents a means of bypassing this set of basic system controls, IBM will correct the problem in the same manner as any other systems software programming error. When a programming correction for a System Integrity error is made available to customers, the accompanying documentation simply says **Integrity Problem** since knowledge of the details of the programming error which causes a security exposure of this type might cause a risk to customers who have not yet implemented the required correction on their system.

The position of other MVS software product vendors has often been much different. Few, if any, openly cite support for and commitment to MVS System Integrity in their advertising or product publications. Of course, if questioned by a potential client on integrity support, a salesman or product manager may likely answer *Certainly!* (or maybe *What's system integrity?*). A number of security conscious customers have asked their non-IBM vendors to provide written statements of support for MVS

System Integrity. Even in cases where the vendor's response indicated genuine commitment and an apparent degree of technical competency in this subject, the results of a system integrity audit have frequently proven otherwise.

Of course, the above information is not intended to suggest that IBM always does it right the first time or that others don't. IBM has provided corrections for a significant number of integrity exposures over the lifetime of MVS and these problems were not limited to just the **control program**. Corrections for such errors have been observed for a variety of other IBM products such as IMS, CICS, DB2, RACF, JES2/3, NetView, etc.

In addition, a number of non-IBM vendors have been responsive to reports of similar problems in their products, once such exposures were identified — as a result of a security audit, penetration study or as a by-product of analysis of an associated reliability failure.

12 Future Security Concerns

Recent articles in both general public media and computer industry publications have shown a distinct rise in concern and interest in security, much of it due to increased system accessibility due to the Internet and perceived threats from terrorists, foreign hackers and military or industrial espionage. Here are a few recent examples:

- *Economic espionage is the new global warfare* — U.S. News 3/04/96
- *U.S. easy target for cyberattacks* — Computerworld 5/27/96
- *U.S. squares up to cyberterrorists* — London Financial Times 6/11/96
- *Feds turn blind eye to systems security* — Computerworld 6/27/96
- *Hackers may be soldiers of the future* — InfoCanada June 96
- *Computer crimes surge* — USA Today 7/02/96
- *Security issues lurk behind bank doors* — Computerworld 7/22/96
- *Few gains made against hackers* — Computerworld 9/16/96

While the focus on system penetration from the outside using network access does reflect a legitimate area of concern, the possibility of threats from insiders is largely ignored, especially for MVS mainframes. This is primarily due to the false sense of **complete protection** which customers believe is provided by the dominant access control products (RACF and CA-ACF2).

The majority of individuals with access to MVS systems are not computer professionals. It is also true that few **application** programmers or computer operators have the experience or skills to detect system integrity exposures or recognize the presence of authorization SVCs. Even for system programmers, the fact that most vendor written products are distributed without source code makes it difficult to uncover such conditions without significant effort and motivation.

However, a concentrated effort by a small group or even a single individual could present a widespread threat. Once the skills of recognizing and detecting system integrity exposures have been acquired (or taught), a number of commercial MVS remote computing services could be studied with the objective of collecting a wide variety of such exposures and the development of associated system penetration test cases. What results is a **tool bag** developed by a professional hacker which can be used by that individual or others to attack a system according to the specific product suite it contains. Keep in mind that system integrity, like most security mechanisms, is a matter of the weakest link in the chain. For a given system, all that is usually required to bypass normal security controls is a single system integrity exposure or authorization SVC.

13 Summary

The most significant issue here is the fact is that some/many/most (take your pick) MVS systems contain customer written or vendor written SVC routines (and/or similar PC routines) which either ignore or fail to measure up to the requirements of IBM's System Integrity Guidelines and present the opportunity for circumventing the customer's intended security controls. The author has not yet encountered an MVS system that could not be penetrated, one way or another!

What can customers do? Should they continue to rely on the good faith efforts of vendors? Should they just wait for the forecasted chaos, thefts or malicious acts to occur (if they ever do) and then try to determine how their systems were subverted?

Several of the steps considered for helping to resolve such problems present the same dilemma - balancing the computing users need-to-know about already discovered exposures in certain products against the issue of aiding and abetting potential hackers. One of those considered is the establishment of a national or international bulletin board or registry of at least the most flagrant easy-to-find/easy-to-exploit exposures. Even if the information is limited to vendor, product, version and problem reference number, it still serves to alert hackers to the best places to attempt penetration.

Another option considered is that of using the skills and experience of some individuals proficient in this area to produce a comprehensive treatise which describes what to do and what not to do when developing products and applications subject to IBM's System Integrity guidelines, along with examples of good and bad implementation techniques. While such information would be beneficial to system designers and developers as well as professional security auditors, it would be a powerful tool for anyone with malicious intent.

14 Recommendations

Three areas are proposed for consideration by security organizations, MVS customers and MVS-based product developers:

- education of security criteria evaluation groups,
- development of both static and runtime code analysis tools similar to current PC virus detection facilities, and
- provision of additional program authorization controls and auditing facilities by IBM.

Each of these recommendations is discussed in detail in the following sections.

14.1 Security Evaluation Group Education

It has been the author's experience that several MVS-based products from different vendors which were evaluated under the current Trusted Product Evaluation Program were later found to contain system integrity exposures which could be exploited to not only avoid the security controls in that specific product but also those of the entire system. It can only be assumed that the evaluation team lacked the experience, education or directives to determine that the programming interfaces provided were flawed. If the new Trust Technology Assessment Program is to be successful and gain the respect and confidence of both government and commercial enterprises, the evaluation criteria must include such tests as are required to ensure a high degree of resistance to penetration. The same need for additional education and compliance verification may apply to the evaluation programs in Canada and Europe.

14.2 MVS Code Analysis Tools

This suggestion is directed more towards MVS-based security product and auditing tool vendors or computer science research facilities. It seems that if some of the more common problems such as the presence of authorization SVCs or first level storage protection violations can be easily detected by the trained eye, it should not be too complex a matter to develop automated tools to do the same. Static analysis refers to the scanning of program object code on disk storage, somewhat similar to virus detection. Over time, more complex analysis methods which are able to keep track of multiple logic paths and utilize environmental history (such as the current status of locks or potential for asynchronous processing) may be employed. Runtime analysis refers to tools which monitor such programming services in real time by use of breakpoints or traps or by use of some form of overseer facility such as IBM's VM operating system provides. The development of such tools would obviously be of great benefit to the security evaluation programs, even if they did no more than highlight suspect areas for more detailed inspection.

14.3 IBM Development of Extended Authorization and Auditing

In general terms, the MVS Authorized Program Facility is an all or nothing mechanism. The initial design, it appears, was intended to allow multiple levels of program authorization capabilities, presumably in some type of hierarchical fashion. Perhaps the implementation of multiple levels turned out to be too complex for practical purposes. However, it may be that the time has come to revisit this concept. A vendor product or customer application may currently require APF authorization only because it needs to use a relatively harmless but nevertheless restricted function or portion thereof. Once provided such authorization, the program can potentially modify or extend the most elemental and critical MVS facilities. It could, for example, dynamically add a new SVC interface which could be utilized by any program in the system even though it was intended for use only by other programs which are part of the subject product. The ability to provide a more granular mechanism would allow the most sensitive restricted functions to be more tightly controlled and monitored.

An alternative but equally effective technique might be to utilize the generic resource protection facilities of RACF or equivalent security products to limit access to such sensitive functions to particular registered and controlled programs.

Auditing extensions would involve new facilities to externally record the occurrence of system events which have a potential security impact. For example, MVS currently provides an in-storage log called the SVC update recording table. The current data is inadequate for effective audit usage. To be effective the information needs to be more like SMF data, containing for example, user name, job or task name, date/time, requesting program, etc. and it needs to be recorded externally. Providing a similar facility for external recording of dynamic activation and deactivation of new Program Call interfaces would provide additional audit data which would also be beneficial for security evaluation programs.

Finally, the ability to have MVS, in conjunction with the customer's particular security product, positively verify the protected status of all libraries containing authorized programs would be a valuable enhancement, particularly in view of the dynamic authorized library activation that newer MVS levels provide. This facility is pictured as a customer selectable option which, if activated, would verify that every APF library is:

- covered by an explicit or generic profile,
- ***not*** defined with universal update access,
- ***not*** operating in **warning mode** only, and
- physically present on the designated volume.

Security officers and auditors would likely welcome the ability to have the system perform this function for them, rather than rely on periodic scanning of selected data set profiles looking for exceptions. Imagine the frustration (and embarrassment) of discovering, after the fact, that an inadequately protected authorized library was the cause of a costly intrusion or theft.

15 Conclusion

In spite of assertions that *The mainframe is dead*, MVS-based systems continue to be the critical focal point for enterprise computing in many of the largest civilian, government and military installations. This paper has attempted to demonstrate that these systems may be more vulnerable than generally perceived and that existing government programs for evaluation and certification of Trusted Systems may need improvement before consumers can place a higher level of confidence in the real trustworthiness of evaluated products operating on MVS.

16 References

IBM Publications on MVS Security and Integrity

G221-3528, *IBM Security Architecture: A Model for Securing Information Systems*

GC28-1439, *MVS Planning: Security (Chapter 5 — System Integrity)*

GC28-1467, *MVS/ESA Authorized Assembler Services Guide (Chapter 21 — Protecting the System — System Integrity and Authorized Programs)*

GC28-1440, *MVS/ESA SP V5 Planning: B1 Security*

SC28-1459, *MVS/ESA SP V5 Installation Exits*

MVS Auditing for System Integrity

Conyers, V.L., ***Audit and Control of MVS***, published and distributed by the **Information Systems Audit and Control Association**, order from http://www.isaca.org/pubs1.htm

Standards and Evaluation Program Materials Available on the Internet

Trusted Product Evaluation Program, ***http://www.radium.ncsc.mil/tpep/***

Trust Technology Assessment Program, ***http://csrc.nist.gov/ttap/***

Minimum Security Requirements for Multi-User Operating Systems (NIST "3/93), ***http://csrc.ncsl.nist.gov/nistir/ir5153.txt***

Threats to Computer Systems: An Overview, ***http://bilbo.isu.edu/security/csl/csl03-94.html***

Information Systems Trustworthiness, ***http://www2.nas.edu/cstbweb/240e.html***

FIRST Security Papers (Evaluation Criteria), ***http://www.alw.nih.gov/Security/first-papers.html***

Generally Accepted System Security Principles, ***http://nsi.org/library/compsec/gssp.txt***

A Guide to Understanding Audit in Trusted Systems (NCSC 7/87), ***http://www.army.cz/secupage/orig/tg001.txt***

Computer Security Evaluation FAQ, ***http://www.ja.net/newsfiles/janinfo/cert/FAQ/Comp_Security_Evaluation***

Security Related Internet Sites

National Institute of Standards and Technology (NIST) — Computer Security Resource Clearinghouse, ***http://csrc.ncsl.nist.gov/***

NIST Computer Systems Laboratory, ***http://www.nist.gov/itl/csl***

NIST Information Technology Laboratory, ***http://www.nist.gov/itl***

Computer Operations, Audit, and Security (COAST), Purdue University, ***http://www.cs.purdue.edu/coast/***

National Computer Security Association, ***http://www.ncsa.com***

National Security Institute, ***http://nsi.org***

The Information Systems Security Association , Inc. (ISSA), ***http://www.uhsa.uh.edu/issa/***

Center for Secure Information Systems, George Mason University, ***http://www.isse.gmu.edu/~csis/***

Center for High Assurance Computer Systems, ***http://www.itd.nrl.navy.mil/itd/5540/***

Many of the above contain links to a wide variety of other security oriented web sites.