The Beginning of Data Security

Julie-Ann Williams
with Barry Schrager

I entered the mainframe industry in the late 1980's when I began working for NaSPA, the National Systems Programmers Association, now called Network and Systems Professionals Association. To many of you reading this, I am a relative latecomer to the mainframe world. Yet, as publisher of Technical Support magazine then and with NewEra Software now, I have worked closely with authors, columnists, speakers, competitors, colleagues – many are practitioners and all subject matter experts, which is infinitely more important. I count myself as one lucky person to brush elbows with greatness.

As the Director of Strategic Partnerships for NewEra Software, Inc., I have noticed over the years that many of the individual contributions of the thought leaders of our industry have gone unrecognized. Julie-Ann Williams correctly writes in this document that *"Mark Zuckerberg is recognized around the world but Barry Schrager and his compatriots walk unnoticed in most gatherings."*

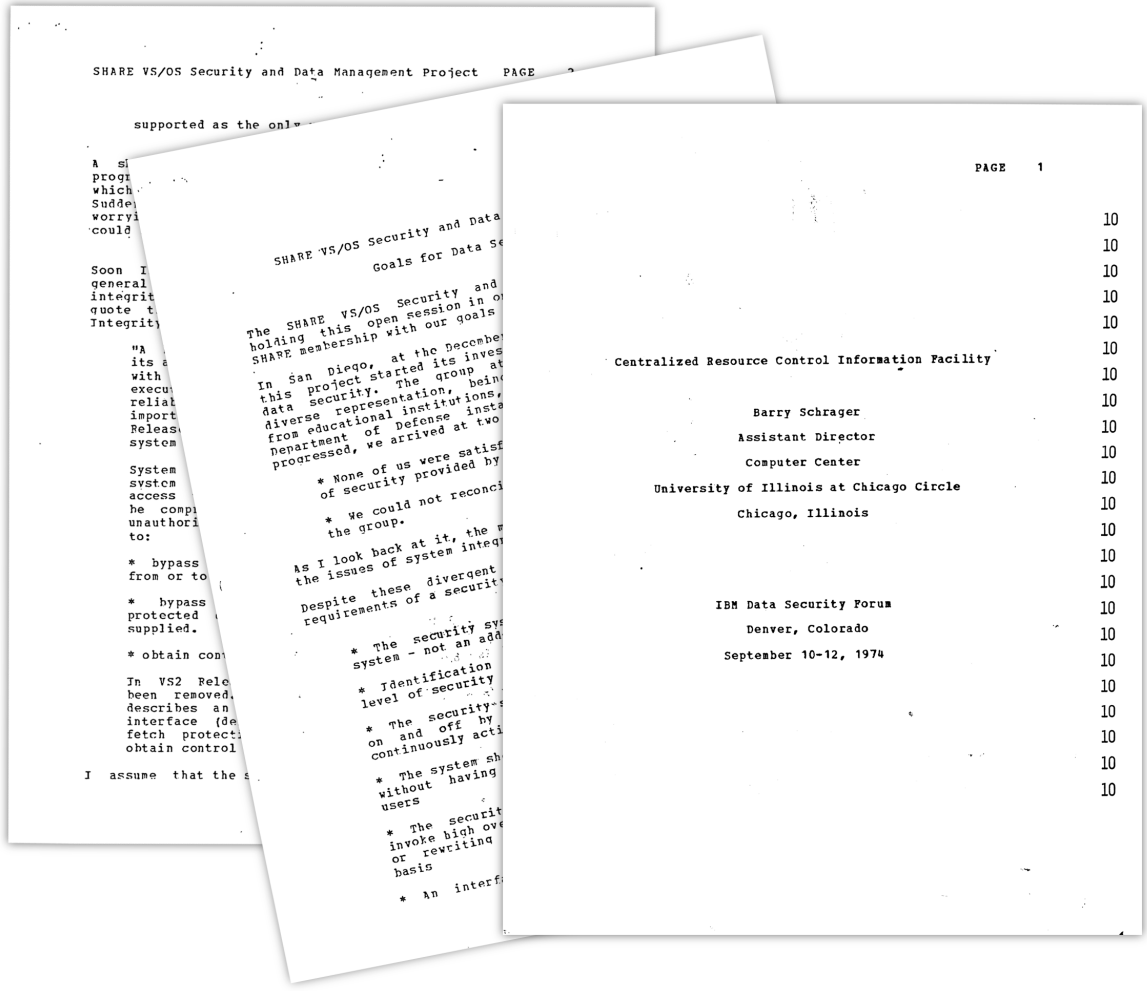
I met Barry Schrager a few years ago when I became a volunteer of the SHARE Security and Compliance Project. I did not know at the time Barry started the Security Project in 1972 and was its first Project Manager. Shame on me. We as an industry are fortunate to have people like Schrager still involved in our industry, SHARE and its Security Project and should celebrate that fact. As you read this document, you will quickly understand why Schrager is in the Mainframe Hall of Fame (www.mainframezone.com/mainframe-hall-of-fame) along with other industry giants.

When we discussed the title of this document, I suggested using ***Mainframe Security – the Beginning***. Barry countered with ***The Beginning of Data Security*** because there was no security at all at that time. My mission is to make sure more people learn about the likes of Barry Schrager and his efforts to better secure data. I also feel it is important to help z/OS practitioners stay current while honoring those who paved the way for us.

This is why I decided to use my company's resources to commission this document as well as previous books on auditing CICS (*CICS Essentials – Auditing CICS – A Beginner's Guide)* and *z/OS* (*z/Auditing Essentials – Volume 1– zEnterprise Hardware, An Introduction for Auditors*).

This document is the first article in *z/Auditing Essentials – Volume 2*, which will contain white papers from subject matter experts. We offer this first article ahead of the rest of the book because 2012 marks the 40th anniversary of the SHARE Security Project.

Jerry Seefeldt – December 2011



SHARE VS/OS Security and Data Management Project PAGE

supported as the onl

I assume that the s

SHARE VS/OS Security and Data

Goals for Data Se

The SHARE VS/OS Security and
holding this open session in o
SHARE membership with our goals

In San Diego, at the Decembe
this project started its inves
data security. The group at
diverse representation, being
from educational institutions,
Department of Defense insta
progressed, we arrived at two

* None of us were satisf
of security provided by

* We could not reconci
the group.

As I look back at it, the w
the issues of system integ

Despite these divergent
requirements of a securit

* The security sys
system - not an add

* Identification
level of security

* The security-
on and off by
continuously acti

* The system sh
without having
users

* The securit
invoke high ove
or rewriting
basis

* An interf

PAGE 1

Centralized Resource Control Information Facility

Barry Schrager

Assistant Director

Computer Center

University of Illinois at Chicago Circle

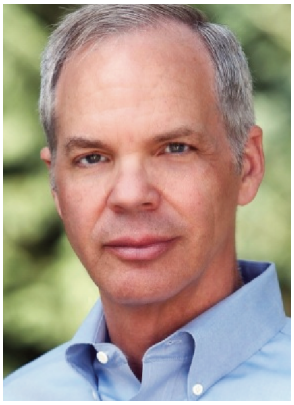
Chicago, Illinois

IBM Data Security Forum

Denver, Colorado

September 10-12, 1974

It was 1974. The year that American racing driver Dale Earnhardt Jr, French rally driver Sebastien Loeb and British super model Kate Moss were all born but some of the "grown-ups" had other things on their minds...



Barry Schrager

A group of IBM mainframe specialists gathered together to do one of the things they do best – think up new problems! It was time for Barry Schrager and his SHARE Team to report back on their research into this new problem they'd been looking into called data security.

It always sounds like a fabulous time to have been involved in this emerging technology "club" of mainframe computing. Educational establishments were leaders in that technology exploitation. Governments took an active interest in IT security and the Military (not to mention certain 3 lettered organizations) were involved in the same committees as us "normal folk". The stories that Schrager and others tell of the time put me in mind of the recent blockbuster movie, "The Social Network" except with more polyester and mainframes. Oh and all this was some 10 years before Mark Zuckerberg was even born.

Whilst Barry Schrager and some of his peers had an enormous impact on the way that we Mainframe Users do business today, their stories are not well known. Mark Zuckerberg is recognized around the world but Barry Schrager and his compatriots walk unnoticed in most gatherings. I hope to be able to do something to redress this imbalance!

This will be the first of a number of papers introducing a "Who's Who" of some of the brightest, under-sung stars of our mainframe generation. Following on from the acclaimed "z/Auditing Essentials VOLUME 1 - zEnterprise Hardware - An Introduction for Auditors" the papers are designed as a teaser for VOLUME 2 – Auditing z/OS. These folks will be providing their input to the new book.

I am blessed with being able to work with many of my heroes and you can look forward to a quick succession of interviews with the guys who weren't afraid to challenge the status quo. But, for now, come with me back to the 1960s for a bit of background on Barry Schrager's introduction to the cut and thrust of mainframe computing.

IBM 1620

After spending the Fall of 1964 at the University of Illinois at Navy Pier on the shores of Lake Michigan, the Campus moved to become the full 4 year University of Illinois at Chicago Circle. The new campus contained a Computer Center which supported the academic and research needs of the students and faculty.

Barry started his career working on the IBM 1620 (the scientific version of the 1401) and rapidly became the "Go to Guy" for many programming requirements – helping both his fellow students and the Faculty.

Barry's unique way of looking at IT problems came to light pretty quickly. As did his not insignificant skills in making a mainframe behave better. During one of our chats Barry shared the following story which I think demonstrates why he was able to be so influential.

> "I remember one case where a Faculty Chemist was working on molecular attraction and had a program that processed one piece of data every 14 HOURS! They used to start it on Friday

> evening and stop it on Monday morning. No one normally used the computer on weekends so this was not a problem, but everyone wanted a larger volume of data processed.
>
> So, I started working on making this task more efficient – first by doing approximations instead of calculating exact numbers from formulas, but that didn't help much. Then I just worked through the calculus and simplified the process, combining equations, etc.
>
> The outcome ran for just 45 minutes per set of data with the same results!
>
> I remember the Professor not really believing me so he sat me down and made me go through every equation, what I did, why, etc. After hours, he finally believed!!!"

This ability of Barry's to patiently explain things, sometimes over and over, until his audience understands, is something I have learned to lean heavily on. When even the most even tempered amongst us would be losing our collective cool, Barry maintains the same patience he started the process with!

He once found himself lent out to the Microbiology Department of Presbyterian St. Luke's Hospital which was close to the University campus for a project many of us will have benefitted from. It was a joint project with Argonne National Laboratories.

Argonne was working on a project "Automatic Chromosome Analysis By Computer" and their objective was to be able to develop a program that would pick out cells with damaged chromosomes from film shot through a microscope.

Argonne was interested in radiation damage to chromosomes and the Hospital in microbiological damage to chromosomes, but the process for detecting them would be the same. The geniuses behind this were a couple, Jim and Margaret Butler, researchers who worked at Argonne. The project was able to match chromosome pairs, and finding and pointing out any damaged ones, but, at that point in time they did not have a clue as to what the results meant. Barry takes up the story:

> "I worked on this and then left to go back to working at the Computer Center and never thought anything of it. Until about ten years ago when I first saw an Amniocentesis report...
>
> Amniocentesis is the test given to a pregnant mother-to-be which draws ammonic fluid from her and determines if the fetus has any genetic diseases.
>
> I couldn't believe it because the results I was looking at were the same as those we created at Argonne!"

Barry went back to work at the Computer Center. Partly because he was still a student but partly because they were getting "new toys"! An IBM/360 Model 50 for the geeks out there. He took a course in IBM 360 Assembler (taught by Jack Stoller – another of the unsung greats in this industry) and the die was cast. Despite his intent to go to Graduate School full time, Barry only stayed away for one semester. And it was then that our story starts to take shape.

One of the first tasks that Barry tackled on the new IBM/360 mainframe was the implementation of a product called Conversational Programming System (CPS). This was an interactive programming environment that compiled PL/1 into an intermediate code which was then interpreted when the student "ran" the program.

IBM/360 Model 50

Fans of ACF2 may recognise the concept of a compile into a more efficiently interpreted state and then an interpreter which was inspired by this project.

Barry seems to have found it very hard to ignore a problem. As a result, he was a prolific developer of special code to deal with difficult situations. A lot of this code still exists today!

For example the problem that the University had with a huge backlog of student jobs which were prioritized but never ran because higher priority jobs always came into the system. Barry's response was to develop the HASP Priority Ager which moved the jobs up in the queue at intervals. This code is still in JES2 today.

Barry got involved with SHARE at this time too. His initial interest was the TSO Project.

In the early 1970's, Barry and Scott Krueger, who became one of the K's in SKK[1], needed something better than what TSO supplied for Job Submission and Output Retrieval. So, they created a product, which they describe as somewhat of a kludge, called the HASP/TSO Interface Package (H/TIP).

When it became time for the Computer Center to migrate from MVT to MVS, Barry, who was then Assistant Director of the Computer Center, had already assigned Scott to convert many of the HASP modifications to JES2. So Barry worked alone to write J/TIP (JES2/TSO Interface Package) which provided sophisticated job submission and output retrieval. It counted General Electric Timesharing, Mutual Life of Canada and even the IBM Development Labs amongst its Customers. The product is dead now but its spirit can be said to live on in SDSF.

Then there was this little problem MVS had with paging. It stemmed from there not being enough real memory to back up the virtual memory. One of the big contributors to this problem was a badly organized Link Pack Area. Barry had an idea:

> "I had an idea to collect a sample of execution addresses and optimize the placement of modules in the Link Pack Area. I asked Eb Klemens (the other K in SKK) to write some code to sample and collect the PSW addresses and then I used his data as input, with a map of the Link Pack Area, to determine where the high density execution was and, for modules less than 4K in length, pack the high usage ones on a small set of pages. The result was a packing list that significantly reduced paging due to the Link Pack Area."

During this whole time, security wasn't even a consideration. Computers were still, for the most part, independent devices not connected out over a wide area network. Access was primarily physical and no one had even thought about it. Sarbanes Oxley was a very long way off!

But, students started deleting and modifying Graduate Student and Faculty research data. So guess what happened? You got it – Super Barry to the rescue! Working with Eb Klemens again, the birth of the Hacker was soon being dealt

with (albeit at a very superficial level on the mainframe to begin with). It was enough to see Barry being asked to form and become the Manager of the SHARE Security Project in 1972.

Key to the start of this project were both Bill Griffin from Mitre Corporation and Mary Lasky of the Johns Hopkins University Applied Physics Lab. They were both quite senior in the SHARE hierarchy.

The other members of the group, including representatives from Boeing and the Defense Department (the mists of time have dulled the memory and both Barry and I would like to apologize for being unable to either remember or find out who these strong hearted individuals were).

Barry presented back the results from the SHARE Security Project in a large SHARE Open Session in 1974. The representatives from the IBM Labs were in attendance. Bill Murray was the IBM representative to the project and he championed it within IBM.

When RACF was introduced in 1976, Barry and his Team at the SHARE Security Project initially thought that they had won a great victory. However, after closer examination, Barry realized that very little of the requirement, as defined in his 1974 paper, had been addressed. When he asked what had happened, he was informed that the IBM Development Team had decided that some of the requirements were not achievable! This included “protection by default” and “algorithmic grouping of users and resources”. Think ACF2 pattern masking or RACF generic profiles to get the idea of the second requirement.

We know enough of Barry by now to realize that this kind of statement is simply a red rag to a bull! The result was that he went on to develop the Access Control Facility at the Computer Center which did satisfy the SHARE Security Project’s requirements. After the University of Illinois declined to back the creation and funding of a commercial version of ACF, the London Life Insurance Company of London, Ontario, Canada funded it and ACF2 was born. It’s a personal rule of mine never to say “No” to Barry!

Barry even tried to use this opportunity to complete his Doctoral Thesis. He had already picked Data Security as a topic because he was most interested in this new field. Barry picks up the story:

> “One of the members on my Committee told me that he wanted me to prove my theory in real life. That set me back because I felt there was no way I could prove this in “real life”. And, IBM had just announced a $40 million program in data security so how was I going to compete with that?
>
> So, I worked on my concepts at the University of Illinois and turned them into the prototype for ACF2. London Life Insurance funded the development of the commercial version and then things moved very fast – General Motors Audit wanted ACF2 installed at one of their sites.
>
> This was in a world where, to take one example, GM’s Delco Division had RACF installed for 18 months and had only 3% of their data protected. Their Auditors felt that they did not know exactly what percentage of data should be protected, but they knew that 3% was not it!
>
> In January 1978, the month before we went to London Ontario to develop the commercial version of ACF2, I gave a presentation at General Motors. The result of the presentation was interest and

a statement by Corporate Audit – if you ever get this working, let us know.

Protected by ACF2...
Courtesy of GM Photo Store

So, after ACF2 was operating at London Life, GM, in an effort to put pressure on IBM to fix RACF, had Gerry Lyons, the Security Officer of Pontiac Motor Division come to London Life to see ACF2 and soon thereafter install it. (Sadly, Gerry Lyons has since lost his battle with cancer.) Pontiac was at 100% data protection in 3 months. With this early success at Pontiac, GM had their Assembly Division install ACF2. The Assembly Division was at 100% data protection in even less time.

GM Audit then had an awakening and wrote a letter to all their divisions demanding that they implement either ACF2 or RACF. They went on to say, that because Delco Division had RACF installed for almost two years with only 4% of their data protected and Pontiac, GMAD, Chevrolet, etc had ACF2 installed for under six months and had 100% of their data protected, they wanted justification for how much data was actually being protected!

So, by the time I got back to the Dissertation Committee at Northwestern, a good percentage of the members had moved on. I met with the new Committee and told them that I did not have to prove my concepts worked in real life – General Motors and the Central Intelligence Agency had proved that for me. The new members felt that much of what I was saying was not really new, using patterns to select resources, and therefore not worthy of a dissertation!

Finally, they asked how many sites ACF2 was in and I responded 125. Their parting shot was to say that ACF2 was too commercialized to be a dissertation topic. They gave me six months to come up with a new topic. I told them I had too much going on to spend time with their foolishness."

Barry is an expert in both developing software himself and developing Teams to produce complex software. He is currently the Chief Security Architect at Vanguard Integrity Professionals, a supplier of RACF enhancement and other products to secure the z/OS Operating System. Over the years since our story takes place, he has started and run a number of software companies (including pivotal roles at SKK, EKC and JME) and has worked in many guises from Systems Software Developer through z/OS Security Analyst/Auditor and Team Leader through CEO. Whatever role he takes on, Barry brings his incredible level of technical understanding of the operating system and applies it in business terms. Under Barry's leadership, SKK developed the first z/VM security product, ACF2/VM (working with Charlie Kao) and the first MVS Operating System Auditing Product, Examine/MVS (working with Martin King). The later is now known as CA-Auditor.

But you can check out those facts for yourself if you are interested in finding out more about Barry Schrager. For me, he will always be my hero for never accepting "No" as an answer and for forcing an entire industry to listen to reason! He also did me the great honor of writing the foreword to "z/Auditing

Essentials VOLUME 1 - zEnterprise Hardware - An Introduction for Auditors" and I am grateful to be able to return that favor!

I hope you enjoy Barry's original papers[2] which follow as much as I did. We are living in a world where much of the content is as fresh today as it was back in 1974. And I for one think that the world could only benefit from the wide-spread acceptance of lip prints as an authentication protocol, as described by Barry in his paper, – then I could start every working day by giving my lap top a quick kiss! ☺

Note 1: SKK was the company that Barry Schrager went on to start (with Scott Krueger and Eb Klemens) in order to have a vehicle through which to sell ACF2 after the product concept went over so well at the 1977 SHARE Conference when he presented there. When SKK was sold to UCCEL (who were later acquired by CA), ACF2 was installed in about 2700 sites and has now generated over $1 Billion in revenue!

SKK led the development of mainframe security from the late 1970s. They also produced the first VM Security product, ACF2/VM and the first Operating System Auditing Product, Examine/MVS, now known as CA-Auditor.

Note 2: The original papers were produced using Wylbur on MVS!

Wylbur was a great text editing, program editing, job submission and retrieval system. In fact, the 1974 papers were written using Wylbur on an IBM 2741 terminal which was really a Selectric Typewriter connected to the mainframe!

I received the papers on actual paper and had to scan and OCR them. The electronic version has been manually compared against the original and I have tried to replicate the work exactly. Any errors probably stem from that process not mistakes by Barry Schrager!

SHARE VS/OS Security and Data Management Project
Goals for Data Security

The SHARE VS/OS Security and Data Management Project is holding this open session in order to acquaint the general SHARE membership with our goals for data security.

In San Diego, at the December 1972 interim SHARE meeting, this project started its investigation into the problems of data security. The group attending that meeting had very diverse representation, being composed of representatives from educational institutions, service bureaus, industry and Department of Defense installations. As the discussions progressed, we arrived at two conclusions:

- None of us were satisfied with the non-existent level of security provided by OS/MVT.
- We could not reconcile the different requirements of the group.

As I look back at it, the main problem we had was separating the issues of system integrity from data security.

Despite these divergent viewpoints we did agree on certain requirements of a security system. These are:

- The security system should be part of the operating system - not an add-on package
- Identification and validation of users is the first level of security
- The security-system should not be able to be turned on and off by an ”authorized user” - it should be continuously active
- The system should be able to run a highly secure job without having to purge itself of all other jobs or users
- The security system should be able to selectively invoke high overhead functions, such as dataset purging or rewriting with zeros, on an individual resource basis
- An interface program, for example IMS, should be supported as the only way to access a specific dataset

A short time later, IBM announced its forty million dollar program in data security. Then they announced VS2 Release 2, which provides the basis for a secure system – integrity. Suddenly the project was freed of the constraints of worrying about both system integrity and data security, and could focus its attention solely on data security.

Soon IBM will ship VS2 Release 2. For the first time a general purpose operating system that is guaranteed to have integrity will be available to the IBM user community. To quote the VS2 Release 2 Planning Guide's section on System Integrity:

> “A highly desirable property of an operating system is its ability to insure that one program cannot interfere with or modify another program’s (system or user) execution unless it is authorised to do so. For reliability and system availability, this is extremely important: for data security, it is essential. VS2 Release 2 provides this capability in the form of system integrity support.
>
> System integrity is defined as the ability of the system to protect itself against unauthorized user access to the extent that the security controls cannot be compromised. That is, there is no way for an unauthorized problem program using any system interface to:
>
> - bypass store or fetch protection, i.e. read or write from or to another user’s areas.
> - bypass password checking, i.e. access password protected data for which a password has not been supplied.
> - obtain control in an authorized state.
>
> In VS2 Release 2 all known integrity exposures have been removed. IBM will accept as valid, any APAR that describes an unauthorized program’s use of any system interface (defined or undefined) to bypass store or fetch protection, to bypass password checking, or to obtain control in an authorized state.”

I assume that the severity level of these APAR’s can range from severe to trivial, just like normal APAR’s, depending on the severity for the installation involved. However, handling of the APAR’s cannot be done normally. Placing the APAR in Early Warning Microfiche will jeopardize the security of every installation running the system. Yet to not notify the installations will leave then in a position of vulnerability. This is indeed a serious problem and we have not resolved it as yet.

Therefore, with the integrity commitment from IBM, it is now is an opportune time for us to develop our set of data security requirements for an IBM operating system.

Although physical security, protection from destruction of data through a natural or man-made disasters, disclosure of data via electronic eavesdropping, etc., are crucial issues in the total security picture, we feel that security systems from the operating system point of view are now an overriding point of concern. We are well on our way to solving the other issues, at least to the point where the risks associated with them are negligible compared to operating system security.

Pressure is also being brought to bear on us from other directions. More and more States, and the federal government as well, are enacting laws to prosecute those who disclose confidential personal data even if it is by negligence.

In addition, as we continue to consolidate and centralize our data processing centers and with the trend towards larger online databases, we increase our risks of accidental or malicious destruction and alteration of data. In many cases, we have dropped our manual backup systems. Even if we did have them, we have become highly dependent on computer based systems. For example, many companies have used the capabilities of computers to reduce their inventories to a minimum. Reverting to a manual system, even with accurate local inventories, could not be accomplished without a substantial period of chaos.

A few years ago, one could assume that someone might attack a database for personal gain. Unfortunately, nowadays, large corporations and governmental agencies may have their systems attacked solely for the purpose of disrupting service, with the attacker not really caring whether he is caught. Smaller companies are not immune either, since a revengeful employee could be similarly motivated.

We can no longer be complacent with the knowledge that we can probably apprehend all violators because even fear of apprehension is not always a deterrent. The motivation for the attack may be to interrupt service and have nothing to do with personal gain.

We must have adequate identification techniques and journaling systems that can allow us to reconstruct what happened. We must also have authorization and surveillance systems that will halt any attack at its early stages - before it can do any damage. Once the security system recognizes that an attack is in progress, it must take appropriate action to protect itself. Furthermore, the security system must be capable of tracking attacks which have failed and of initiating affirmative action to prevent further attacks from this source.

This action is obviously dependent on the installation and the data involved: but it may range from just notifying security personnel to actually shutting down the computer system itself. Whatever the case, the system must act reliably and predictably and must fail-soft, that is, if the system is to fail, it must fail in a predictable manner.

We believe that security must be a joint vendor-customer endeavour in that the operating system and all sub-systems and application programs must work together in providing defense of data. Thus installations must develop security standards for all application software and provide mechanisms to enforce those standards.

Many times the argument is encountered that some other computer vendor has produced a security system that is independent of the application programs. Where this is true, it is generally applicable only to single terminal/task relationships, introduces restrictions on sharing of data, and would require rebuilding our existing systems from the bottom-up. Multiple terminal/task systems, where the terminals have different security requirements, still require the cooperation of the application program to enforce security. Also, unfortunately, we have been developing our systems based on an operating system that was designed in the early 1960's and these designers just did not conceive of the applications and the associated security requirements of the systems that are in use today.

Most of us here have too much invested in application systems to be able to afford to start over and redesign our systems to run under a totally new operating system. Besides, many data dependent security constraints still could not be enforced without some security in the application program itself.

It is necessary to provide for auditability and administration of both applications and the system itself. Placing security controls in these areas, each independent of the other, could lead to obvious and severe inconsistencies. Consequently, in VS/OS Group Requirement #73-86, we have recommended the following:

Description:

There should be a centralized bank of resource control information and an installation replaceable operating system provided service for accessing and maintaining it. The resource control information must relate resources (such as datasets, program paths, etc.), conditions under which they can be made available

(such as level of validation), and user identifiers. New types of resources, the resources themselves, the conditions, and the user identifiers must all be installation definable.

All authorization and delegation must flow through the single operating system access and maintenance service, and this service must be invocable during normal production operation. Invocation for the purpose of validating access to a resource should return a yes or no answer and optionally a variable length byte string to be used in corrective action (e.g. an error message, a module name, or a limit on a quantative resource). Security violation recovery routines should be modular and designed for easy user replacement.

Both Algorithmic grouping and grouping by itemization for both resources and user identifiers should he possible.

Incentive:

Demonstrable consistent application of resource control has become a requirement. In addition, administration of resource control will be facilitated by its centralization.

This facility, along with security standards in all ”secure” application programs to make sure that they invoke it, solves the problems of consistency of application, ease of administration, and auditability.

Notice that in addition to physical resources such as datasets, the facility handles logical resources such as program paths and anything else the installation wishes to define. Thus an installation's security management will be able to say that an individual is able to enter a certain type of transaction from a set of designated terminals between certain hours. In this case, the transaction type can be considered the resource and the latter statements are the conditions under which the individual can access the resource.

The requirement also means that the information facility must be able to dynamically update its databank while the system is running. The system cannot be required to be unavailable during entry or processing of these changes.

Algorithmic grouping of resources and user identifiers is required; for example a group of datasets can be referred to as ”SYS1.*". This means that, as datasets are added or removed from the system with the names beginning with "SYS1”, they will automatically attain the protection level of the group without costly database updating for each addition and removal. This is particularly important in interactive environments where datasets are created and deleted continuously. The user and the installation must be able to specify global security requirements for these resources.

Along with this facility, an authorization system or application program is necessary to update the database. This program must be able to relate the system’s secured resources to an administrator in the administrator’s language. It must be easy to use by non-technicians, such as those responsible for defining authorization privileges.

For “open shop” installations, it is necessary to introduce the concept of the “owner” of the resource where resources may be datasets and programs. The owner is the administrator of that resource and may specify who can access it. He may also request that accesses be journaled. This is particularly important when dealing with proprietary programs and databases.

All of the five requirements submitted to the project use the phrase “installation replaceable”. We define “installation replaceable” to include “easily modifiable in its source language”. Thus, the default routines provided by IBM should be designed for ease of modification, well documented, and written in a language for which we have a compiler.

A phrase that is often used in security systems is “acceptable level of risk”. We believe that using VS2 Release 2 as a base, the above requirement provides the authorization and delegation functions which are mandatory for a secure system and that installation security standards aided by the other project requirements submitted at the same tine can allow an installation to maintain whatever it considers to be its acceptable level of risk.

We feel that an installation is never totally secure and always has some chance of an attack against it succeeding. By devoting effort in the proper areas, an installation can reduce the probability of an attack succeeding, or can lower its level of risk.

We have determined five areas where effort must be placed to reduce the level of risk. Four additional requirements were submitted to IBM to give us the proper tools within the operating system to increase our

security. One area is solely the installation's responsibility. These areas are:

1. installation security standards
2. level of identification and validation of each user
3. journaling facility and recovery procedures
4. security violation recovery
5. designated interface programs

Installation security standards:

There is no way to underestimate the importance of adequate security standards and their strict enforcement. The programmers who code the application programs, the systems programmers who maintain and customize the operating system, the security staff that maintains and customizes the security violation and validation routines, all have an opportunity to subvert the security of the system.

As programmers and managers, we should welcome strictly enforced security standards and complete journaling or all operating system and application program changes. Systems programmers are in a particularly precarious position since they can be blamed for almost any incident on the system. Being able to affix responsibility for some action to an individual also means it is easy to exonerate everyone else.

However, we do not expect that this mechanism will get its greatest use from tracking down security violators, but rather that errors in the system will be more easily determined and corrected. Since all changes to application programs and to the operating system must be journaled in detail, they can easily be backed off. Improved maintenance procedures are a significant by-product of security standards.

The minutes of the project meeting last August reflect this concern:

> "An important point was raised about the system database (e.g. LINKLIB, LPALIB, etc.). This is one of the most critical databases on any system and a full audit trail is absolutely necessary both in case of eventual security violations and standard recovery in case of system failure."

IBM, in a recent announcement, has given us a tool called the System Modification Program which is due to be shipped at the end of this month. This service aid applies superzaps, module replacements, macro replacements, and source code changes via IEBUPDTE. It keeps a detailed log of all changes and provides a back-off capability for removing modifications.

Level of identification and validation:

Another area of risk in a secure system is "How sure is the system that a user is who he says he is?" This problem is compounded by the use of different identification and validation procedures in batch, TSO, APL, IMS, etc. Group requirement #73-85 aids an installation in reducing its risk in this area:

Description:

> All subsystems must call upon a single installation replaceable subroutine for the purpose of identification and validation of all users of the system. A default routine should be provided by IBM for this purpose along with full specifications of the interface requirements.
>
> The subsystem should provide to the subroutine the origin of the entry request and full facilities for interacting with the user and related equipment.

Incentive:

> Identification and validation of the user is a prerequisite to any data access control system.

Comment:

> The purpose of this requirement is not to ensure identical logon protocols but rather to specify that the information collected and the validation procedures are identical for both batch and interactive systems.

This requirement gives the installation the ability to validate the identity of the user to whatever level it wishes and to use this level of validation in later security access decisions. For instance, with hardware now readily available, it is possible to have the following levels of validation:

- Password only
- Password/Badge reader
- Password/ 2 Badge readers - one being used by a guard who has visually identified the user.

More exotic validation techniques could use fingerprints, voice prints, lip prints, etc. Identification and validation of a user's identity is the cornerstone of security. This, combined with proper journaling, provides the ability to limit the accountability for some action to an individual. It is important to think of the initial identification and validation procedures as being the first line of defense of the system. The risk of data security violations is reduced, by limiting access to the system to only those who are authorized to use It.

Journaling facility and recovery procedures:

Application designers must be encouraged to journal everything that is necessary to reconstruct an event or recover a database. Journaling should be a service offered by the operating system just like the access methods. Depending on the installation, the journaling facility may have to handle very large volumes of information and utilize several output devices. It may have to have the strictest integrity and use special hardware such as a special tape drive and controller that buffers the output within the controller so no data can be lost by a system failure. Another possible hardware requirement is a “write once” tape that cannot be rewritten.

The main requirements of a journaling facility therefore are:

- easy to use
- able to handle large volumes of data efficiently
- be tailorable to an installation's needs
- provide an unimpeachable record of activity for auditability and recovery

VS/OS Group Requirement #73-87 requests this facility:

Description:

There should be a centralized installation replaceable journaling facility provided by the system control program which can be invoked by any program. This facility must be capable of recording on a variety of data storage devices. Where IBM provided subsystems use this facility, programs should be provided by IBM to analyze these entries.

Incentive:

A centralized journaling facility is necessary for auditing and recovery.

Another area of concern is a misuse of authorization. For example, a person may be authorized to access certain types of services, but by combining them in an unforeseen manner, he would be able to do something that is not authorized.

Detailed journaling by the application programs and by certain operating system interfaces would allow reconstruction of the sequence of events which led to the unauthorized use.

Security Violation Recovery:

It is admitted by most security people that, given enough time, any security system can be violated. However, in the process of testing a system’s defenses and attempting to bypass them, indications of these attempts will occur. Whether the system will actually be violated, is therefore a function of how alert an installation's security staff is, how good the installation’s security violation analysis programs are, how well attuned the security system is to the installation’s needs, and what action the system takes to preserve its security.

A security violation facility should be provided by the operating system in order to institute timely and appropriate recovery action. Group requirement #73-88 addresses this need:

Description:

The system control program should provide an installation replaceable security violation exit. An IBM default routine should be provided which makes a journal entry and writes a message with a security violation routing code.

Incentive:

Timely corrective action and notification of the proper installation personnel is critical. This is an extremely installation dependent function.

A trivial example of this requirement is the case of several failures on a prompt for a password at logon. Most systems merely disconnect the line - a secure system should call upon the security violation exit which will deactivate the user-id until security personnel have investigated.

Designated interface programs:

The project's final requirement deals with designated interface programs. We do not feel that it is the responsibility of the operating system to do "record and field" level data protection although we do feel that the operating system should maintain the authorization database which is involved in making those decisions.

The possibility of an application program inadvertently modifying data incorrectly and the chance that an authorized user of a database may bypass security and journaling functions by using his own program to access a database led us to group requirement #73-89:

Description:

There should be the capability of associating with any dataset a single interface program capable of accessing that dataset. Where the interface program is a subsystem (e.g. IMS) an interface should be provided to other subsystems (e.g. TSO).

Incentive:

The need to be able to limit the path to a dataset to one interface program structures the system so as to provide increased integrity, security, and back-up capabilities.

This requirement also addresses the problem of data integrity. Data integrity is concerned with the accuracy and completeness of the data while data security is concerned with protection from unauthorized destruction, modification, or disclosure whether accidental or intentional. Misuse of authorized privileges, errors in application programs, etc. can cause a database to lose its integrity. In these cases data security, as we define it, has not been violated.

A designated interface program is in a position to perform validation of database update requests and to journal all changes. This helps to protect the integrity of the database by providing a record of database modifications for error determination and recovery. In addition, it can be much more efficient than the operating system in journaling those chances since it is able to deal with specific and meaningful information. The operating system would have to journal all data changes, thereby incurring much more overhead.

In summary, VS2 Release 2 provides a base for a data security system because of its integrity features. The VS/OS Security and Data Management Project has determined the following requirements for a data security system:

- Security system should be an integral part of the operation system and should be tailorable to an installation's needs.
- High overhead security functions should be selectively invoked on an individual resource basis.
- Users should be identified and validated in the same manner at all interfaces to the system.
- Resource control information and decision making should be centralized.
- Journaling capability should be an operating system service.
- Security violation handling should be centralized.
- Designated interface programs should be supported on a dataset basis.

The five requirements submitted by the project on this issue were:

- 73-85 Identification and Validation Exit
- 73-86 Central Resource Control Information Facility
- 73-87 Centralized Journaling Facility
- 73-88 Centralized Security Violation Exit
- 73-89 Designated Interface programs.

If anyone has any questions, opinions, or suggestions concerning these topics or any other aspect of data security, please contact me or any or the other project members.

Barry Schrager UIC
Project Manager
March 4, 1974

Centralized Resource Control Information Facility
Barry Schrager
Assistant Director
Computer Center
University of Illinois at Chicago Circle
Chicago, Illinois

IBM Data Security Forum
Denver, Colorado
September 10-12, 1974

In December 1973 the SHARE Security and Data Management Project submitted requirements to IBM in the area of future data security goals. The objective of these requirements was to define an environment within the OS/VS operating system such that a basic security system would exist and implementing a customized security system would be a feasible task for most installations.

IBM has provided a crucial prerequisite of this environment with OS/VS2 Release 2 and its system integrity support. Until this time, the only way to provide a secure system was to limit system access to some secure subsystem. An example of this type of subsystem could be IMS. Unfortunately, the security of such a system leaves much to be desired since most installations, because of economics, are forced to allow application program development and system programming activity to simultaneously occur on the same system as the hopefully secure DB/DC subsystems. These concurrent activities, all occurring on the same data processing system are an area of considerable concern.

This is not meant to indicate mistrust of the system programmers; rather, this concern is a realization that when there is a breach of system security, those with unrestricted access to the system are, unfortunately, often assumed guilty until proven innocent. We define the datasets which contain the modules of the operating system as an entity called 'the system database' and system programmers must deal with this database daily. This database is the most important database on any system, since a violation of its integrity can lead to an undetectable violation of any other database on the system. To treat the system database lightly, or with inadequate, security and journaling procedures, is a clear invitation to disaster.

It is my belief that the first objective of a security system must be to protect both the system database and the application program database from unauthorized change. This includes the automatic journaling of sufficient information such that recovery of the system to the state prior to the change is possible, as well as making a record of each change so that accountability rests with a single individual. Furthermore, the journal itself must be completely secure and inviolate. Good maintenance procedures are a by-product of good security procedures since even unintentional errors can be easily found and corrected.

The ability to identify the responsible party for any modification to any system, application, or working database on the data processing system is a requirement. To be able to do this implies that the system must be able to identify its users with a reasonable degree of confidence. The degree of confidence should be related to the importance of the database and the level of risk that the installation wishes to assume. This requirement is currently complicated by the fact that today's delivery subsystems such as JES, CICS, IMS and TSO, have no common method of identification and validation of that identification. In fact, some do not provide for any identification at all.

It is for the above reasons that one of the SHARE Security and Data Management Project's requirements called for a common user identifier across all interfaces to the system, and for the ability of the installation to provide its own validation techniques in order to assure the desired degree of confidence in the identification necessary for their environment.

An installation's validation techniques may include the use of passwords, badge readers, or the placement of terminals and other input devices in supervised areas where a secondary visual identification is required by a guard before access is permitted. The required hardware for this support is currently available and the software required is not difficult to produce; yet almost no delivery systems support it. The point is that we have no system-wide user identifier and no software support of the hardware security devices.

It should also be noted that an unsuccessful attempt to access the system may be an indication that the system is under attack. Unsuccessful attempts should be journaled with all available supporting information, including the geographical location of the attempt if it is available. Computer analysis of the journal may provide an installation with information that can help to prevent successful unauthorized access to the system.

Likewise, since the user identifier is the same for all delivery systems and is unique to a single individual, simultaneous access to the system with the same identifier from two terminals is an absolute signal that the security of the system has been breached and security personnel should be immediately dispatched to investigate. Simple computer analysis of all access attempts may also uncover violations of the system security if the accesses from a single user occur from different geographical locations, and these accesses are physically impossible to accomplish within a short period of time.

Thus, common user identification, sufficient validation techniques, and some method of restricting users to only those subsystems that they need to access are the first lines of defense for any system. This, along with proper journaling techniques, which can limit the responsibility for any action to the level of an individual, which, if well publicized, can act as a deterrent to misuse of authorized privileges, and which can reconstruct the system to the state prior to an undesirable modification, may be all that an installation requires to protect its mission capability.

These procedures are sufficient protection for an installation only under limited circumstances. First, data processing management, either directly or indirectly via corporate management, must have absolute control over all users of the system, such that it would be extremely disadvantageous for someone to misuse his authorized privileges. The second condition is that any unauthorized alteration of data could be quickly found and corrected, and responsibility for this alteration could be easily assigned. The third condition is that the mission capability of the system would not be seriously jeopardized during the interval of time between the alteration and the eventual correction of the data.

The procedures discussed are not adequate if there is no direct control over the users such as in a service bureau environment, if the confidentiality of the data is such that serious consequences would occur if disclosed, or if the correctness of the data at all times is essential to the mission capability of the data processing center.

An additional step towards protecting data would be a program such as IMS which would be supported by the operating system as a single designated interface for accessing a database. This implies that the “designated interface program” must be able to analyze the access request to determine its validity both in terms of the individual making the request and the content of the request. An interface program should also be capable of preventing accidental destruction of the data and be in a good position to efficiently journal changes to the database for later recovery.

However, although designated interface programs are required for system security, to place the authorization decision making within each interface program may be difficult and lead to inconsistencies between various programs. In addition, as was stated previously, if this is to be considered a solution, then the system and application program databases must also be treated securely. The only program and process that is able to protect these databases is the operating system itself, and no preconceived security scheme provided by the vendor will be able to enforce the wide variety of requirements found in various installations.

A good example of this is the question of whether to have centralized or decentralized administration of the security or the data on the system. If the system only services one corporate function, if the corporation believes it is economically and practically feasible to have a central security staff, or if the system handles highly classified information, then a centralized staff for administration of the security function provided by the data processing system is mandatory.

In a service bureau environment, a university, or anywhere that the responsibility for the security and integrity of the data lies outside of the data processing center, then the people who own or are responsible for the data must have administrative control over it. In these kinds of environments decentralized administrative control places security where it is most meaningful.

Any security system must be able to function regardless of whether its administration is centralized or decentralized. It must be a part of both the interface programs and of the operating system if it is to be effective.

In addition, the security system must be fairly efficient. This is not to say that it must induce no overhead, but

rather that the overhead induced must not be so great that it is not economically feasible to use the system. Overhead can be reduced by decreasing the number of discrete security decisions that the system must take. It is better to control more global resources such as transactions, rather than to control local resources such as the fields within a database.

The case against controlling local resources can also be strengthened by what can be loosely described as a cross-coupling of resources. An example of resource coupling would be that, while it may be permissible for a user to modify a field in a database, a condition imposed may be that some other field is also modified in a predetermined manner. This would require a sophisticated designated interface program or simple transaction level authorization. In this case, control at the transaction level would reduce overhead, simplify the analysis of the system and not jeopardize the security of the data.

Unfortunately, as the magnitude of the resources being controlled increases, the proportion of responsibility for the implementation of the security system shifts from the vendor to the customer. For example, while the responsibility for the limitations of access to specific transactions may lie with the vendor, what the transaction does and what rules it follows are the customer's responsibility.

If the data and programs residing in a data processing system are considered the resources of that system, then the function of the security system is to control the interaction between the users of the system and these resources in a manner predetermined by the administrators of the resources.

Whatever the implementation of this system, it must be easy to use and easy to understand. For many systems, a high requirement is that it be easy to audit. Complexity and confusion within the system may make the auditing process difficult and therefore lead to unknown irregularities.

However, an uncoordinated approach, with a multiplicity of resource control routines in the various interface programs, in the operating system, and in other miscellaneous application programs, is neither easy to use nor easy to understand. Understanding the complex interaction for several systems, each with its own unique database and algorithmic processes, will make comprehension of the system as a whole difficult at best. Finally, demonstrating consistent application of resource control will be time consuming and difficult.

On the other hand, a single process within the system, making resource control decisions, must treat resources consistently and be easily extendable for new applications. Its interfaces to the system should be modifiable so that, with simulation, its decision making processes could be more easily tested, understood and verified. With a well-planned set of interfaces via the system control program, it would be easy to use for application programs. Since it would be removed from the application programs themselves, application programmers need not know the exact decision making process that would be used. Conversely the decision process could be easily modified without having to modify each of the application programs. And finally, since it would be removed from the physical resource control, it could easily control conceptual resources such as program paths.

Examples of program paths are transactions, command sequences, and operating system processes such as "open". A program path can also be defined to include the flow of control within a module. This enables an installation to define different security levels for different paths within an application program, without having to rewrite different application programs due to the differing requirements for security. It also avoids the need for coding authorization checks within the application program itself.

All authorization decisions would be made within a central control facility which would be a service function of the operating system. The central facility for resource control would make decisions when requested by other parts of the operating system or application programs. Implementation of the decision would be left to the requestor. Possible responses by the central facility could include a yes or no answer, a limit on a quantitative resource, an error message to be displayed to the user, or the name of an error routine to which control should be transferred.

The title of this paper, Centralized Resource Control Information Facility, is the title of a SHARE Security and Data Management Project requirement which states:

> "There should be a centralized bank of resource control information and an installation replaceable operating system provided service for accessing and maintaining it. The resource control information must relate resources (such as datasets, program paths, etc.), conditions under which they can be made available (such as level of validation), and user identifiers. New

> types of resources, the resources themselves, the conditions, and the user identifiers must all be installation definable.
>
> All authorization and delegation must flow through the single operating system access and maintenance service, and this service must be invocable during normal production operation. Invocation for the purpose of validating access to a resource should return a yes or no answer and optionally a variable length byte string to be used in corrective action (e.g. an error message, a module name, or a limit on a quantitative resource). Security violation recovery routines should be modular and designed for easy user replacement.
>
> Both algorithmic grouping and grouping by itemization for both resources and user identifiers should be possible."

The incentive given for this requirement was that "demonstrable consistent application of resource control has become a requirement." In addition it stated that administration of resource control will be facilitated by its centralization within the data processing system.

Having a central facility, such as that described above, means that more effort can be spent in making it a sophisticated product. And sophisticated it must be, if it is to be useful. Algorithmic grouping of resources is an example of this sophistication. In addition to itemizing resources, the requirement states that resources must be able to fit into groups as described by some algorithm. The TSO notation of asterisk for dataset naming conventions to indicate all qualifiers as in Userid.PROGRAM.* is an example of this. This means that the security information database need not be updated each time a dataset is created or deleted and will automatically assure whatever level of protection that is specified by the algorithmic group that describes it.

The implications of the system described are interesting. Security is a joint customer-vendor endeavor. It is the responsibility of the vendor to supply the services within the operating system to access the centralized resource control information facility; to provide an easily extendable set of routines that comprise the facility for the basic decision making of the operating system; and finally to provide the capability for performing maintenance on the information used by the security system. With vendor supplied delivery systems, such as IMS or CICS, security support should be provided as additions to the central security facility rather than as independent routines. And finally, support for a system-wide user identifier and an easy-to-use journaling facility are required

It is the installation's responsibility, however, to verify that their own application programs formulate requests correctly to the facility and take the correct action as specified in the facility's response. This means that application programs should formulate their requests to the central security facility in a manner consistent with the operating system's use of this facility. This is an obvious example of the consistency of application criterion for security rules.

Although it is not entirely satisfying to have a security system that is neither totally the vendor's responsibility nor the customer's, this set of recommendations seems to offer the best base for continued extendability. We must acknowledge that this is an era when systems are constantly being used for new applications at the same time that there is constant pressure for consolidation of facilities and databases. These pressures are in turn forcing less specialized and more general purpose operations.

The concept of the centralized resource control information facility, along with a system-wide user identifier and proper journaling techniques, is capable of simultaneously satisfying the requirements of a batch system, an interactive timesharing system, and a data base/data communications system. It is time to develop a coordinated system-wide approach to the solution of our resource control problem.

Continuing the Work

The SHARE Security and Compliance Project continues to advance the practice of better data security. Project Volunteers as of December 2011 include:

Brian Cummings – Tata Consultancy Services – Project Manager

Greg Boyd – IBM Corporation

Glinda Cummings – IBM Corporation

Phil Emrich – Vanguard Integrity Professionals

Carla Flores – CA Technologies

Saheem Granados – IBM Corporation

Paul Robichaux – NewEra Software, Inc.

Roxane Rosberg – Vanguard Integrity Professionals

Barry Schrager – Vanguard Integrity Professionals

Jerry Seefeldt – NewEra Software, Inc.

Maria Tzortzatos – IBM Corporation

The Project encourages all security professionals to join the Project. More information is available on the SHARE Security and Compliance Project website – www.SHARE-SEC.com

If you would like to download a softcopy of either of the System z Audit books referred to, please use the links below to access them.

Link to ***CICS Essentials – Auditing CICS – A Beginner's Guide***

http://www.newera-help.com/CICS-Essentials.html

Link to ***z/Auditing Essentials – Volume 1 – zEnterprise Hardware, An Introduction for Auditors***

http://www.newera-help.com/zAudit.html