On virtual machine integrity

Dear Editor,

J. J. Donovan and S. E. Madnick[1] in their presentation of an approach to computer system integrity are under the misconception that the improvement in integrity of a virtual machine can be merely accomplished by a hierarchical decomposition of the system and by providing redundant checks. In their paper the authors have claimed to have shown that a hierarchically structured operating system, in particular the one produced by the combination of a virtual machine monitor and several independent operating systems, provides substantially better software security and reliability than a conventional two-level multiprogramming operating system approach.

Security and Reliability are collectively known as integrity which is stated in their paper as "may be existing when an operating system functions correctly under all circumstances." This definition is very vague and seems to imply that integrity and correctness of an operating system are synonymous. Furthermore, on page 193 of their paper the authors state that "a reliability failure is any action that causes the system to cease correct operation" and "a security failure is a form of reliability failure." This leads us to believe that security, reliability, integrity, and correctness are synonyms.

Forum

We disagree with the way these terms have been defined and used interchangeably throughout their paper. We will give the currently accepted definitions for these terms and point out only some places of incorrect usage of these in the text.

The second point of our refutation is the main conclusion of their paper. We will show that the assumptions on which the hierarchically structured operating system provides better software security do not hold in practice.

The title of the paper seems to indicate that the notion of system integrity would be investigated. On starting to read the abstract, one would expect it to be on security. On further reading, one concludes that virtual machine systems offer extra protection using redundant independent mechanisms.

It is very useful to draw a distinction between security and protection in computer systems. This is based on the concept of separation between mechanism and policy[2-4] in the design of operating systems.

A *security policy* is a finite set of rules which delimit the access to information during the progress of a computation. A specific

case of making security policies regarding the dissemination of information of, or about, a person or a group, when controlled by that person or group, is known as *Privacy*. A *Protection Mechanism* is the facility in terms of which security policies can be implemented and enforced.[5] For example, the policy to support mutually suspicious subsystems is enforced by the processor protection mechanisms designed by Schroeder.[6]

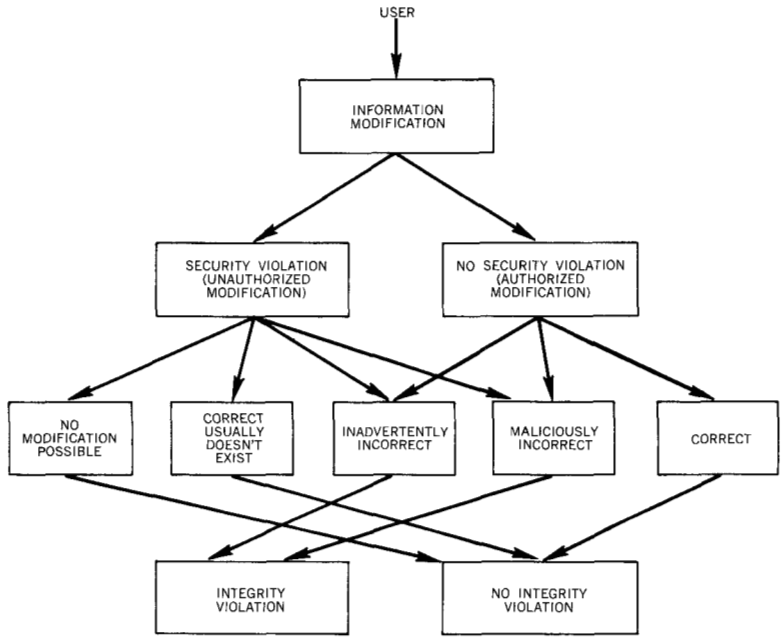
By *Reliability*[7] is meant a low probability of failure of a system whatever the source-hardware or software. A program is said to be *correct*[8] if its execution terminates and yields the desired final results. It is said to be *Partially Correct*[8] if given that its execution terminates and it yields the desired final results. It is clear from above that correctness is a necessary (but not sufficient) condition for reliability. For example, a protection mechanism enforcing a particular security policy may not be reliable even if proven to be correct.

A Protection Mechanism is said to be *Complete* when every attempt to access information during the progress of a computation is validated. A Protection Mechanism is said to be *Tamperproof* when it is protected from unauthorized alteration. It is obvious that if the protection mechanism can be tampered with, its ability to protect information can be destroyed. *Certification* is the process of checking for the Completeness and Correctness of a Protection Mechanism (not a security Policy). For a computer system to be secure, its Protection Mechanism must meet the above three requirements.[9]

There is no currently accepted definition of integrity (to our knowledge) but it closely relates to the alteration principle.[10] The *alteration principle* is defined to be the unauthorized alteration of information. This will be an incomplete definition for integrity (if used) because it does not cover all the possible causes for committing an integrity violation as shown in Figure 1. The figure clearly distinguishes between security and integrity violations and shows that an integrity violation may occur with or without a security violation. Unauthorized modification of information is one case of a potential security violation.[11] The possibility of correct information modification after a security violation has taken place usually does not occur in practice. If one could design a system such that there is no modification of information possible after a security violation has been detected, this would solve two possible cases of committing an integrity violation. The other two possible cases of integrity violation occur when one is authorized to modify but does it incorrectly either maliciously or inadvertently.

An *Integrity Policy* can be defined as a finite set of rules which specify the modification of information during the progress of a

Figure 1 System integrity concept



computation. Any sound integrity policy has to make sure that the paths taken always lead to the box shown as NO INTEGRITY VIOLATION in Figure 1. Note the use of the word policy as before for security. In this case also (as for security) one would have to face the hard task of providing the mechanisms (may be part of protection mechanisms) to support and enforce the desired integrity policies.

The above terms (in particular, security and protection) have been used interchangeably throughout their paper. The following points are only a few instances.

Abstract. The added protection is not derived from a redundant security (as stated in their paper) but merely by two independent checks before a decision is made.

Page 188. Operating system integrity requires the enforcement of a correct modification of information and not the system functioning correctly under all circumstances.

Page 188. The notion of an operating system crash is a violation of the Guaranteed Service Principle.[10] Another good example of the violation of the guaranteed service principle are operating system deadlocks.

Page 193: It is not clear what is meant by ideal circumstances under which most operating systems provide isolation security.

Page 194: A single logical error in the operating system invalidates the entire protection mechanism and not the security mechanism as stated. Even this is not true due to the concept of the security kernel in which the protection mechanisms are totally isolated from the rest of the operating system.[3,12,13] Hence, a logical error in the operating system cannot invalidate any of the protection mechanisms.

After establishing the notions of security and protection it is instructive to note that the security policy supported by the virtual machine monitor system is one of complete isolation.

A *complete isolation* security policy is one in which users are separated into groups between which no flow of information or control is possible. In fact, the problem solved is even more restrictive because the group consists of only one user. Thus, there is no sharing between users, i.e., the user of such a system may just as well be using his own private (multiplexed) computer, as far as protection is concerned. Most of the first generation of commercial time-sharing systems provide a protection scheme with this policy. It is interesting to note that VM/370[14] falls into one of these.

The implementation complexity of such a policy happens to be the easiest[15] to achieve. Mechanisms which enforce much more complex policies such as mutually suspicious subsystems[6] and memoryless subsystems[16] are being implemented and investigated. It might be a significant advantage if some changes are made to the classical concept of a virtual machine to provide for some flexible sharing. Work in this direction has been reported recently,[17] where some experimental additions have been made to support a centralized program library management service for a group of interdependent users.

Dynamic Verification[18] was an approach taken to increase the reliability of a protection mechanism. The idea is to perform a consistency check on the decision being made using independent hardware and software in a fashion analogous to hardware fault tolerance techniques. This seems to fit in very nicely in the context of virtual machine systems as the two decisions could be made independently, one in the virtual machine monitor and the other in any given operating system. Hence, reliability in protection (and not redundant security as stated throughout their paper) is achieved merely by redundancy.

Based on the above, it has been concluded in the paper that the penetrator has to subvert the operating system first and then, having taken control of the operating system, attempt to subvert the virtual machine monitor. Thus the work effort involved is the sum of two work efforts as shown in the following equation (Equation 8 in their paper).

$$W_s(P|\text{OS}(n)|\text{VMM}(k)) = W \quad (P|\text{OS}(n)) + W_s(\text{OS}|\text{VMM}(k))$$

This idea appears to be very attractive in theory, but it does not work in practice as shown by Belady.[19] The reason for this is the ease with which penetration could be carried out in conjunction with the data channel programs requested for execution by the operating system. Hence, penetration is not a two-step process (as claimed in the paper) due to the trap/interruption facility by which the VMM - OS communication occurs.

Another important point to be emphasized here is the notion of certification of the virtual machine monitor. Since all programs in the operating system run in the problem mode, the privileged instructions are trapped and interpretively executed by the virtual machine monitor. But then what is the guarantee that control is passed to correct code unless the virtual machine monitor has been proved correct? After all, a single error is sufficient for compromising the virtual machine monitor. This is exactly the idea behind going through all the pains to prove the correctness of the hypervisor,[12] the security kernel,[13,20] and even the whole operating system.[10]

Another interesting way of penetration as mentioned by Belady[19] is due to the omission of a double check in the virtual machine monitor.

A different way of phrasing our criticism would be that Table 1 (on page 198 of their paper) is not complete. It points out the redundant protection features in the main memory, the secondary memory, and the processor allocation mechanism. There is no mention of protection features for I/O at all which is the door for penetration.

We also do not agree with Equation 1 (on page 195 of their paper) which says that the probability of system failure tends to increase with the load on the operating system. This is based on the assumption that all users are equally good (or bad) which is certainly not true in general, and in particular because of penetrators.

It has been stated on page 190 by Donovan and Madnick that the integrity of an operating system is improved by a careful decomposition, separating the most critical functions from the successively less critical functions. This static decomposition does not achieve any more integrity unless the dynamics of execution are incorporated in the protection mechanisms. These are the mechanisms that enforce the isolation of different layers. The call bracket in MULTICS[21] is one way of enforcing the isolation of nested layers.

CITED REFERENCES

1. J. J. Donovan and S. E. Madnick, "Hierarchical approach to computer system integrity," *IBM Systems Journal* **14,** No. 2, 188–202 (1975).
2. P. Brinch-Hansen, "The nucleus of a multiprogramming system," *Communications of the ACM* **13,** No. 4, 238–250 (April 1970).
3. W. A. Wulf, et al., "Hydra—The kernel of a multiprocessor operating system," *Communications of the ACM* **17,** No. 6, 337–345 (June 1974).
4. A. K. Jones and W. A. Wulf, "Towards the design of secure systems," *Proceedings of the Workshop on Protection in Operating Systems*, IRIA, Rocquencourt, France, 121–134 (August 1974).
5. A. K. Jones and R. J. Lipton, "The enforcement of security policies for computation," *5th Symposium on Operating System Principles, Operating Systems Review* **9,** No. 5 (1975).
6. M. D. Schroeder, *Cooperation of Mutually Suspicious Subsystems in a Computer Utility*, Ph. D. Thesis, MAC TR-104, Massachusetts Institute of Technology (September 1972).
7. W. A. Wulf, "Reliable hardware/software architecture," *IEEE Transactions on Software Engineering* **SE-1,** No. 2, 233–240 (June 1975).
8. Z. Manna, "The correctness of programs," *Journal of Computer and System Sciences* **2,** 119–127 (May 1969).
9. J. P. Anderson, "Computer security technology planning study," ESD-TR-73-51 (October 1972).
10. P. G. Neumann et al., *A Provably Secure Operating System*, SRI Project 2581, Stanford Research Institute, Menlo Park, California 94025.
11. J. P. Anderson, "Information security in a multi-user computer environment," *Advances in Computers*, M. Rubinoff, editor, 1–35, Academic Press, New York, New York (1972).
12. G. Popek and C. Kline, "Verifiable secure operating system software," *AFIPS Conference Proceedings, National Computer Conference* **43,** 145–152 (1974).
13. W. L. Schiller, *Design of a Security Kernel for a PDP-11/45*, MITRE Technical Report, Bedford, Massachusetts (June 1973).
14. R. A. Meyer and L. H. Seawright, "A virtual machine time-sharing system," *IBM Systems Journal* **9,** No. 3, 199–218 (1970).
15. G. S. Graham and P. J. Denning, "Protection—Principles and practice," *AFIPS Conference Proceedings, Spring Joint Computer Conference* **40,** 417–429 (1972).
16. J. S. Fenton, "Memoryless subsystems," *The Computer Journal* **17,** No. 2, 143–147 (May 1974).
17. J. D. Bagley et al., "Sharing data and services in a virtual machine system," *5th Symposium on Operating System Principles, Operating Systems Review* **9,** No. 5, 82–88 (1975).
18. R. S. Fabry, "Dynamic verification of operating system decisions," *Communications of the ACM* **16,** No. 11, 659–668 (November 1973).
19. L. A. Belady and C. Weissman, "Experiments with secure resource sharing for virtual machines," *Proceedings of the Workshop on Protection in Operating Systems*, IRIA, Rocquencourt, France, 27–33 (August 1974).
20. J. K. Millen, "Security kernel validation in practice," *5th Symposium on Operating System Principles, Operating Systems Review* **9,** No. 5, 10–17 in Supplement to Proceedings (1975).
21. J. H. Saltzer, "Protection and the control of information in MULTICS," *Communications of the ACM* **14,** No. 7, 388–402 (July 1974).

C. S. Chandersekaran and K. S. Shankar*
Burroughs Corporation
Paoli, Pennsylvania 19301

*Although the authors are associated with the Burroughs Corporation, the views expressed in this letter are solely their own.

Authors' response

Virtual machine advantages in security, integrity, and decision support systems

In our paper,[1] we showed that a hierarchically structured operating system, such as produced by a virtual machine system, should provide substantially better software security than a conventional two-level multiprogramming operating system approach. As noted in that paper, the hierarchical structure and virtual machine concepts are quite controversial and, in fact, the paper has received a considerable amount of attention, such as in the letter by Chandersekaran and Shankar.

This letter provides a further confirmation, clarification, and elaboration upon concepts introduced in our earlier paper. Furthermore, based upon our recent research, it is shown that such virtual machine systems have a significant advantage in the development of advanced decision support systems.

background and terminology

In recent years there has been a significant amount of research and literature in the general areas of security and integrity. As noted in our paper, the reader is urged to use the references in that paper as a starting point for further information on these subjects. Of special note, the report by Scherf[2] provides a comprehensive and annotated bibliography of over 1,000 articles, papers, books, and other bibliographies on these subjects. Other important sources include the six volumes of findings of the IBM Data Security Study[3] (the Scherf report is included in Volume 4).

Although there has been a considerable amount of attention and writing devoted to these areas, a precise and standardized vocabulary has not yet emerged. As stated in the recent paper by Saltzer and Schroeder:[4] "The words 'privacy,' 'security,' and 'protection' are frequently used in connection with information-storing systems. Not all authors use these terms in the same way." As an example of the lack of a comprehensive terminology source, Chandersekaran and Shankar found it necessary to draw upon six different references to define less than a dozen terms. Hopefully, as this area matures and stabilizes, it will be possible to reconcile these different viewpoints and arrive at a mutually agreed upon and standardized set of terminology. In the meantime, the reader may wish to study the glossary provided in Saltzer and Schroeder,[4] which by the way, indicates that protection and security are essentially interchangeable terms in agreement with our usage and in contrast to the opinions of Chandersekaran and Shankar.

hierarchical approach to computer system integrity

In our paper, it was shown that a hierarchically structured operating system can provide substantially better software security and integrity than a conventional two-level multiprogramming operating system. A virtual machine facility, such as VM/370,[5] makes it possible to convert a two-level conventional operating system into a three-level hierarchically structured operating system. Furthermore, by using independent redundant security mechanisms, a high degree of security is attainable.

The proofs previously presented support the intuitive argument that a hierarchical-structured redundant-security approach based upon independent mechanisms is better than a two-level mechanism or even a hierarchical one based on the same mechanism. More simply stated, if one stores his jewels in a safe, he may think his jewels are more secure if he stores that safe inside another safe. But the foolish man might (so he won't forget) use the same combination for both safes. If a burglar figures out how to open the first safe (either accidently or intentionally), he will find it easy to open the inside safe. However, if two different locking mechanisms and combinations are used, then the jewels are more secure as the burglar must break the mechanism of both safes. As explained in our earlier paper, the virtual machine approach can provide that additional security.

clarification of certain points

The concept of "load" used in our paper is sometimes misunderstood, such as in the letter of Chandersekaran and Shankar. It refers to "the number of different requests issued, the variety of functions exercised, the frequency of requests, etc." (Reference 1, page 195), not merely the number of users. Hence, our conclusion is supported in that a complex operating system supporting a wide range of users and special-purpose functions is more likely to contain design and/or implementation flaws and is thus susceptible to integrity failures than a simpler operating system. Others have also come to this conclusion. For example, it is noted in the concluding remarks of the recent study of VM/370 integrity by Attanasio et al.[6] that: "The virtual machine architecture embodied in VM/370 greatly simplifies an operating system in most areas and hence increases the probability of correct implementation and resistance to penetration."

There seems to be general agreement on the key point that there should be ". . . mechanisms that enforce the isolation of different layers" as stated by Chandersekaran and Shankar. As previously stated (Reference 1, page 198), "in order to provide the needed isolation, future VMM's may be designed with increased redundant security . . ." A source of possible confusion may arise from the fact that some readers assume that our discussion of hierarchical operating systems and the VM/370 example are synonymous, whereas, the VM/370 example is exactly that: an example. Most of the VM/370 penetration problems, such

as I/O, noted by Chandersekaran and Shankar are attributable to the lack of independent redundant security mechanisms either in VM/370 or in the operating systems running on the virtual machines. For example, under standard VM/370, the CMS operating system provides minimal constraints on user-originated I/O programs. This is usually viewed as one of CMS's advantages, from a flexibility point of view, but this does present unnecessary opportunities for penetration. In the VM/370 integrity study by Attanasio et al.,[6] it was reported that "Almost every demonstrated flaw in the system was found to involve the input/output (I/O) in some manner." In other words, penetration was easiest in the area where the approach of independent redundant security mechanisms was not fully employed.

Flaws, such as noted above, need not exist in a hierarchically structured operating system. Without elaborating unduly, Goldberg[7] has shown that it is possible to build economical hardware support for the hierarchical structure so as to eliminate the need for the VMM to be trapped in order to process operating system level interruptions. In fact, IBM has adopted some of these approaches as part of the "VM assist"[8] hardware features.

Additional uses and benefits of the virtual machine approach

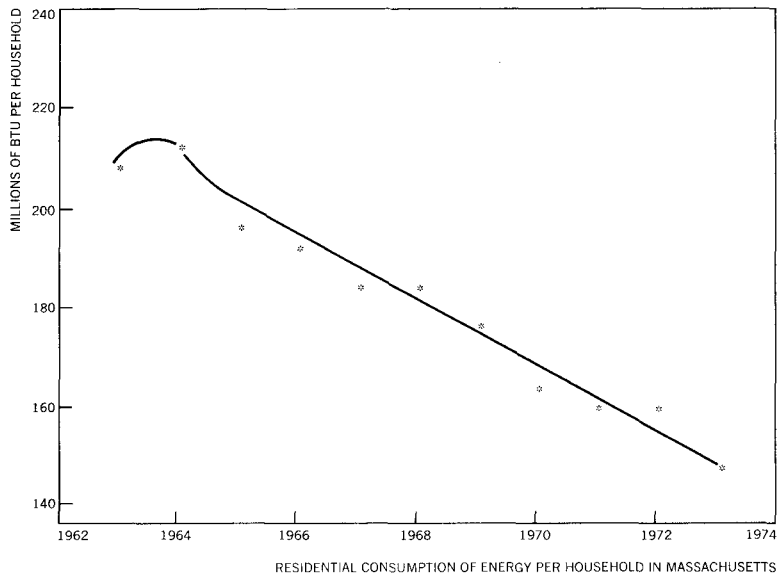
Thus, although VM/370 contains some flaws,[6] these flaws are not inherent in the virtual machine approach and can be eliminated. It is our understanding that various other computer manufacturers are also exploring this approach.

Our recent research in the development of advanced decision support systems, especially in the area of energy policymaking,[9, 10] has provided an example of additional uses and benefits of the virtual machine approach. Advanced decision support systems[11] are characterized by:

- specifics of problem area are unknown
- problem keeps changing
- results are needed quickly
- results must be produced at low costs
- data needed for those results may have complex security requirements since they come from various sources.

This class of problems is exemplified by the public and private decision-making systems we have developed in the energy area. We have found that the problems that decision-makers in the energy area must address have those properties.

Figure 1 Residential consumption of energy in Massachusetts



A specific example of such a system can be found in our recently developed New England Energy Management Information System (NEEMIS).[12] This facility is presently being used by the state energy offices in New England for assisting the region in energy policymaking.

Many of the NEEMIS studies are concerned about the economic impact of certain policies. For example, during a presentation of NEEMIS[12] at the November 7, 1975 New England Governors' Conference, Governor Noel of Rhode Island requested an analysis of the impact on his state of a proposed decontrol program in light of likely OPEC oil prices. These results could be used in a discussion at a meeting with President Ford later that afternoon. This situation illustrates several of the requirements (e.g., results needed quickly and problem not known long in advance) for an advanced decision support system.

In other studies, it is often necessary to analyze and understand long-term trends. For example, using data supplied by the Arthur D. Little Co.,[13] we were able to trace the trends in total energy consumption in an average Massachusetts home from 1962 to 1974. We were interested in studying the amount of increased consumption, the pattern of increase over the years, and the extent to which conservation measures may have reduced consumption in recent years. Figure 1 is the graph produced by NEEMIS showing energy consumption versus time. To our sur-

prise, it indicated a roughly continuous *decrease* in consumption for the average Massachusetts home throughout the entire period under study in spite of increased use of air conditioners and other electrical and energy-consuming appliances.

The object of this study suddenly changed to try to understand the underlying phenomenon and validate various hypotheses. In this process, it was necessary to analyze several other data series and use additional models. Several important factors were identified including: (1) census data that indicated that the average size of a home unit had been getting smaller, (2) weather data that indicated that the region was having warmer winters, and (3) construction data that indicated that the efficiency of heat-generating equipment had been improving.

We had begun the analysis thinking that only consumption data was needed; as it developed, a sophisticated analysis using several other data series was actually needed. This changing nature of the problem or perception of the problem is a typical characteristic in decision support systems. We have found similar problems in our work in the development of a system of leading energy indicators for FEA[9, 14] and in medical decision support systems.[15]

GMIS approach

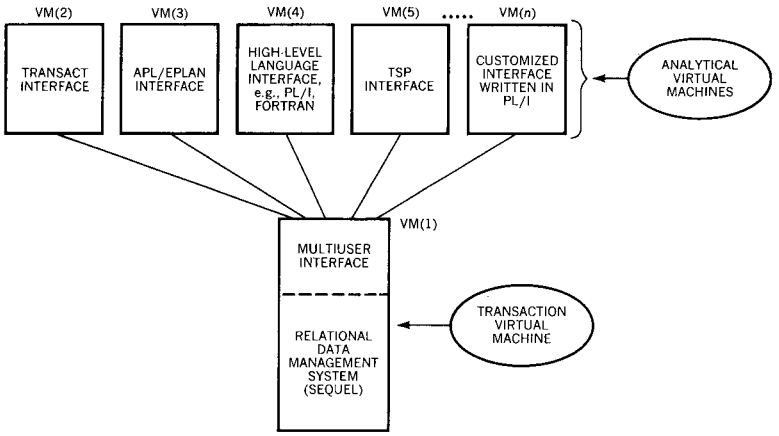
To respond to the needs for advanced decision support systems, we have focused on technologies that facilitate transferability of existing models and packages onto one integrated system even though these programs may normally run under "seemingly incompatible" operating systems. This allows an analyst to respond to a policymaker's request more generally and at less cost by building on existing work. Different existing modeling facilities, econometric packages, simulation, statistical, data-base management facilities can be integrated into such a facility, which has been named the Generalized Management Information System (GMIS) facility.

Further, because of the data management limitations of many of these existing tools (e.g., econometric modeling facilities), we have also focused on ways to enhance at low cost their data management capabilities. Our experience with virtual machines, discussed in the next section, indicates it is a technology that has great benefit in all the above areas.

GMIS configuration

Under an M.I.T./IBM Joint Study Agreement we have developed the GMIS software facility[16] to support a configuration of virtual machines. The present implementation operates on an IBM system/370 Model 158 at the IBM Cambridge Scientific Center.[17] The present configuration is depicted in Figure 2 where each box denotes a separate virtual machine. Those virtual machines across the top of the figure each contain their own operating sys-

Figure 2 Overview of the software architecture of GMIS



tem and execute programs that provide specific capabilities, whether they be analytical facilities, existing models, or data-base systems. All these programs can access data managed by the general data management facility running on the VM(1) virtual machine depicted in the center of the page.

A sample use of the GMIS architecture might proceed as follows. A user activates a model, say in the APL/EPLAN machine (EPLAN[18] is an econometric modeling package). That model requests data from the general data base machine (called the Transaction Virtual Machine, or TVM), which responds by passing back the requested data. Note that all the analytical facilities and data base facilities may be incompatible with each other, in that they may run under different operating systems. The communications facility between virtual machines in GMIS is described in References 16 and 19.

GMIS software has been designed using a hierarchical approach.[19-21] Several levels of software exist, where each level only calls the level below it. Each higher level contains increasingly more general functions and requires less user sophistication for use.

Users of each virtual machine have the increased protection mechanism discussed in our paper.[1] We have also found increased effectiveness in using systems that were previously batch-oriented but can be interactive under VM.

conclusion

We remain enthusiastic about the potential of virtual machine concepts and strongly recommend this approach. VM technology coupled with other technologies, namely, interactive data base systems and hierarchical system structuring have distinct comparative advantages for a broad class of problems, especially in decision support systems.

We suspect that we have only scratched the surface of realizing the potential of VM concepts. One such area is to extend the configuration of Figure 2 to add access to other data management systems. However, more research is needed in the unresolved issues of locking, synchronization, and communication between the virtual machines and related performance issues.

We suspect our arguments will not completely resolve the controversy regarding virtual machine systems. But for users, decision makers, and managers, we want to add hope that this technology can greatly aid in providing tools to them.

ACKNOWLEDGMENT

We wish to acknowledge the following organizations for their support of work reported herein.

The initial research on security and integrity issues was supported in part by the IBM Data Security Study.

The recent applications of the virtual machine approach to decision support systems have been supported in part by the M.I.T./IBM Joint Study on information systems. Special recognition is given to the staff of the IBM Cambridge Scientific Center for their help in implementing these ideas and to Mr. Richard MacKinnon, head of that Center, as well as Dr. Stuart Greenberg, coordinators of the Joint Study for their managerial support.

We acknowledge the members of the IBM San Jose Research Center for the use of their relational data base system, SEQUEL,[22] and for their assistance in adapting that system for use within GMIS and energy applications.

The development of the New England Energy Management Information System was supported in part by the New England Regional Commission under Contract No. 1053068.

Other applications of this work were supported by the Federal Energy Office Contract No. 14-01-001-2040 and The National Foundation/March of Dimes contract to the Tufts New England Medical Center.

We acknowledge the contributions of our colleagues within the Sloan School's Center for Information Systems Research and within the M.I.T. Energy Laboratory for their helpful suggestions, especially those of Michael Scott Morton, Henry Jacoby, and David Wood.

CITED REFERENCES AND FOOTNOTE

1. J. J. Donovan and S. E. Madnick, "Hierarchical approach to computer system integrity," *IBM Systems Journal* **14,** No. 2, 188–202 (1975).
2. J. Scherf, *Data Security: A Comprehensive and Annotated Bibliography*, Master's Thesis, Massachusetts Institute of Technology, Alfred P. Sloan School of Management, Cambridge, Massachusetts (1973).
3. *Data Security and Data Processing*, Volumes 1–6, Nos. G320-1370–G320-1376, IBM Corporation, Data Processing Division, White Plains, New York (1974).
4. J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Proceedings of the IEEE* **63,** No. 9, 1278–1308 (September 1975).
5. *IBM Virtual Machine Facility/370: Introduction*, No. GC20-1800, IBM Corporation, Data Processing Division, White Plains, New York (July 1972).
6. C. R. Attanasio, P. W. Markstein, and R. J. Phillips, "Penetrating an operating system: a study of VM/370 integrity," *IBM Systems Journal* **15,** No. 1, 102–116 (1976).
7. R. P. Goldberg, *Architectural Principles for Virtual Computer Systems*, Ph.D. Dissertation, Harvard University, Cambridge, Massachusetts (November 1972).
8. F. R. Horton, "Virtual machine assist: Performance," *Guide 37*, Boston, Massachusetts (1973).
9. *Energy Indicators*, Final working paper submitted to the F.E.A. in connection with a study of Information Systems to Provide Leading Indicators of Energy Sufficiency, M.I.T. Energy Laboratory Working Paper No. MIT-EL-75-004WP (June 1975).
10. J. J. Donovan, L. M. Gutentag, S. E. Madnick, and G. N. Smith, "An application of a generalized management information system to energy policy and decision making—The user's view," *AFIPS Conference Proceedings, National Computer Conference* (1975).
11. G. A. Gorry and M. S. Scott Morton, "A framework for management information systems," *Sloan Management Review* **13,** No. 1 (Fall 1971).
12. J. J. Donovan and W. R. Keating, *NEEMIS: Text of Governor's Presentation*, M.I.T. Energy Laboratory Working Paper No. MIT-EL-75-018WP (December 1975).
13. *Historical Data on New England Energy Requirements* (prepared for the New England Regional Commission), Arthur D. Little, Inc., Cambridge, Massachusetts (September 1975).
14. J. J. Donovan and H. D. Jacoby, *Use of Virtual Machines in Information Systems*, M.I.T. Energy Laboratory Working Paper (March 1976).
15. J. J. Donovan, L. M. Gutentag, D. Bergsma, and S. Gellis, *An Application of Current Systems Implementation Technologies to a Genetic/Birth Defects Information System*, M.I.T. Sloan School of Management Working Paper (February 1976).
16. J. J. Donovan and H. D. Jacoby, *GMIS: An Experimental System for Data Management and Analysis*, M.I.T. Energy Laboratory Working Paper No. MIT-EL-75-011WP (September 1975).

17. Special recognition must go to Ray Fessell of the IBM Cambridge Scientific Center for his assistance with the implementation and to Stuart Greenberg and Richard MacKinnon of the IBM Cambridge Scientific Center for their support of the entire Joint Study.

18. F. Schober, *EPLAN—An APL-Based Language for Econometric Modeling and Forecasting*, IBM Philadelphia Scientific Center Report, IBM Corporation, Data Processing Division, White Plains, New York (1974).
19. L. M. Gutentag, *GMIS: Generalized Management Information System—An Implementation Description*, M.S. Thesis, Massachusetts Institute of Technology, Alfred P. Sloan School of Management, Cambridge, Massachusetts (1975).

20. S. E. Madnick and J. W. Alsop, "A modular approach to file system design," *AFIPS Conference Proceedings, Spring Joint Computer Conference* **34,** 1–13 (1969).
21. S. E. Madnick and J. J. Donovan, *Operating Systems*, McGraw-Hill Inc., New York, New York (1974).
22. D. D. Chamberlin and R. F. Boyce, "SEQUEL: A structured English query language," *Proceedings of 1974 ACM/SIGFIDET Workshop* (1974).

J. J. Donovan and S. E. Madnick
Center for Information Systems Research
Alfred P. Sloan School of Management
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Editor's Note. Because of the increasing importance of data security, IBM sponsored the Data Security Study, the purposes of which were twofold: to build a body of knowledge and to gain practical experience in the field of data security. The work of Donovan and Madnick was supported in part by this study. The study report, *Data Security and Data Processing*, was published in June 1974, in six volumes; copies were sent to major university libraries and may be obtained from the IBM Data Processing Division, 1133 Westchester Avenue, White Plains, New York 10604.

The paper by Donovan and Madnick has been the source of some controversy. The views represented by Chandersekaran and Shankar were first brought to our attention by R. S. Wasserman, who wrote: "It is my belief that most of the known security holes in existing VMM systems . . . represent instances in which an application program action which would be detected as a security breach by the hardware when running in a real environment is not detected and constitutes a security breach when running in a VMM environment. No OS security hole is required to exploit these VMM security holes."

In soliciting technical comments on the correspondence, we found our reviewers split on the merits of the arguments. All agreed, however, that the debate contributed to their understanding of the issues involved.

Because of its comprehensive presentation, we decided to publish the letter by Chandersekaran and Shankar, along with a response from Donovan and Madnick.

Although data security remains of great interest to us, we feel that this correspondence carries this particular argument as far as is necessary at this time. Consequently, we will not publish further correspondence unless novel and conclusive evidence is presented.

Reprint Order No. G321-5037.