

AD-A108 187

SYSTEM DEVELOPMENT CORP SANTA MONICA CA
KVM/370 TECHNICAL DEMONSTRATION.(U)

F/G 9/2

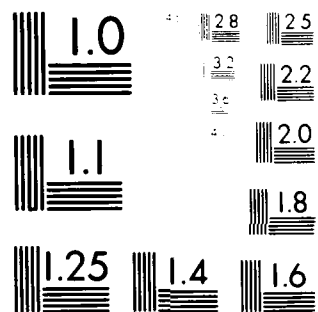
UNCLASSIFIED
SEP 81 M SCHAEFER
SDC-TM-7146/000/00

RADC-TR-81-254

F30602-81-C-0018
NL

1 08 1
AD A
FORM 1

END
DATE
FILMED
01-82
DTIC



MICROCOPY RESOLUTION TEST CHART
NBS 1963-A

(12) LEVEL II

RADC-TR-81-254
Final Technical Report
September 1981



KVM/370 TECHNICAL DEMONSTRATION

System Development Corporation

Marvin Schaefer

DTIC
ELECTE
DEC 8 1981
S B

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

81 12 08 248

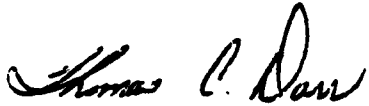
AD A108187

DTIC FILE COPY

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-81-254 has been reviewed and is approved for publication.

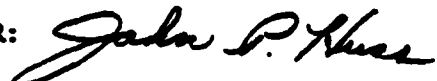
APPROVED:


THOMAS C. DARR, Major, USAF
Project Engineer

APPROVED:


JOHN J. MARCINIAK, Colonel, USAF
Chief, Information Sciences Division

FOR THE COMMANDER:


JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (ISCP) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-81-254	2. GOVT ACCESSION NO. HD-A 108 187	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) KVM/370 TECHNICAL DEMONSTRATION	5. TYPE OF REPORT & PERIOD COVERED Final Technical Report 27 Oct 80 - 17 Apr 81	
7. AUTHOR(s) Marvin Schaefer	6. PERFORMING ORG. REPORT NUMBER TM-7146/000/00	
9. PERFORMING ORGANIZATION NAME AND ADDRESS System Development Corporation 2500 Colorado Ave. Santa Monica CA 90406	8. CONTRACT OR GRANT NUMBER(s) F30602-81-C-0018	
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (ISCP) Griffiss AFB NY 13441	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 64740F 22390302	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same	12. REPORT DATE September 1981	
	13. NUMBER OF PAGES 52	
	15. SECURITY CLASS. (of this report) UNCLASSIFIED	
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) Same		
18. SUPPLEMENTARY NOTES RADC Project Engineer: Thomas C. Darr, Major, USAF (ISCP)		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Computer Security Security Kernel Operating Systems Virtual Machines		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report documents the current status, future plans and results of an operational test and demonstration of a prototype trusted operating system called KVM/370 (for "Kernelized Virtual Machine"/370). The report includes a description of an effort to install KVM/370 as a stand-alone operating system on an Amdahl V-7/A Computer in Patuxent River MD. The implementation, verification, and ultimate certification for multilevel use of this operating system is a major goal in RADC's program to demon-		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

strate the viability of trusted operating systems for DOD.

77

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

ABSTRACT

This represents CDRL Item A002 for Rome Air Development Center Contract F30602-81-C-0018, KVM/370 Technical Demonstration, and serves as a final technical report summarizing work performed on the preparation and demonstration of the Kernelized VM/370 security design.

This report presents the current status and plans of the Kernelized VM/370 Project (KVM/370), a Department of Defense initiative in which a security retrofit is being performed on IBM Corporation's Virtual Machine Facility/370 (VM/370) system that will provide a time-shared environment in which user processes bearing differing military classification levels may be operated simultaneously without compromise to military security policy. The strategy entails drawing together into a secure kernel those system functions that otherwise could be exploited to violate security. This report describes the principal results of the first four years of this program and discusses our current plans for KVM's future.

The report also describes the first installation of a prototype version of KVM, the KVM-Alpha Release, at a government test site. The report includes preliminary benchmark test results obtained at that site, the Naval Air Test Center at Patuxent River, Maryland.

Accession For	
NTIS CRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Avail and/or	
Dist	Special
A	

1.0 INTRODUCTION

The primary goal of the KVM/370 Program is the security retrofit of a popular commercial operating system (VM/370) into a secure system that will be certified for use as a trusted computing base in support of multi-level classified data processing environments. The KVM/370 Program is a research and development initiative, originally funded by the Defense Advanced Research Projects Agency (DARPA), and continuing under the auspices of the Department of Defense Computer Security Initiative. Contract administration transitioned to the Air Force Weapons Laboratory and the Rome Air Development Center as the project matured into its current implementation and testing activities. The project is now funded and administered by the Rome Air Development Center.

This report is organized in three major sections. We begin by giving a short motivational history of the project. We then present a brief discussion of the KVM architecture. We conclude with a report on the current status and plans for KVM. The latter report includes a description of the installation of the KVM-Alpha Release at the Naval Air Test Center, Patuxent River, Maryland, and preliminary performance characteristics of the prototype system that were observed at the test site.

2.0 HISTORICAL OVERVIEW

In the late 1960's, the ADP community became conscious of the need to protect sensitive data from access by users who were authorized to use the computing resource but did not have a Justifiable 'need-to-know' for all of the data accessible from that resource. It was also apparent

that many of the existing operating systems offered no direct protection of files from casual browsing by the user population. Worse, even those systems in which some form of protection (e.g., passwords or access control lists) was part of the design appeared to be impotent to prevent the success of malicious attempts to access such data. At this time, SDC was awarded a DARPA contract that resulted in the implementation of the first general purpose multi-user time sharing system explicitly designed to enforce a formally specified security policy model. [13]<1> This system, ADEPT-50, was reasonably good at defending against direct penetration attempts, but succumbed to more sophisticated indirect attacks it was not designed to protect against.

At the inauguration of the last decade, SDC performed research studies as a pioneer in the evolving computer security community. The majority of work being done at that time centered heavily on security analyses and penetration studies of computer operating systems. When it was first conjectured that a formal penetration methodology could be developed to identify generic security flaws in such systems, SDC commenced an investigation into the viability of the Virtual Machine Monitor (VMM) concept [2, 3] as a certifiable means of achieving secure isolation of user processes on a common mainframe.

In 1974-5, the Systems Security Department of SDC's Research and Development Division performed a study with and at the request of the IBM Thomas J. Watson Research Laboratory. The objectives of the study were

<1>

Numbers in brackets refer to bibliographic entries in the References Section of this report.

- (1) To derive a scientific approach to the identification of security flaws in operating systems; and
- (2) To evaluate the security properties of the VM/370 system.

The results of the completed study included analysis criteria that are best known as the SDC Flaw Hypothesis Methodology (FHM)[15], and a detailed enumeration of hypothetical and verified security flaws in the commercial Release 2.0 version of VM/370.[1]

The FHM was subsequently successfully applied to the security analysis of a number of other commercial operating systems. However, the security properties of VM/370 stood out as exceptional evidence of the validity of the VMM hypothesis. The Virtual Machine Monitor concept is so fundamentally simple that it is implemented in VM/370 by a body of code that comprises less than 100,000 assembly language instructions. This body of code is called the VM/370 Control Program (CP). Its function is the simulation of the privileged portion of an IBM S/370 computer.<2> VM/370-CP had far fewer security flaws than any other operating system we have examined. Many of these flaws were actually flaws present in the IBM S/370 architecture itself, perfectly simulated by the Control Program for exploitation by the penetrator.

In 1976, SDC was awarded a study contract by the Defense Advanced Research Projects Agency. The primary purpose of the study was to ana-

<2>

Later commercial releases of VM/370 are significantly larger than this, but the majority of additional code is present for the support of new I/O devices, having essentially no effect on the overall logic or functionality of the Control Program.

lyze the feasibility and cost/effectiveness of designing and implementing a certifiable security retrofit to VM/370. The underlying concepts behind the project were fairly simple:

- (1) The security relevant operations on an S/370 are a proper subset of the privileged operations on an S/370;
- (2) All privileged operations in VM/370 are simulated by the Control Program;
- (3) Therefore CP, if correctly implemented, could enforce a security policy over all virtual machines as a reference monitor;[4]
- (4) However, a certifiable reference monitor must be sufficiently small and simple as to be 'verifiably' correct in its functionality, and even at 100,000 instructions, CP was too large to verify by formal state-of-the-art means;
- (5) It should be possible to partition CP into its security relevant and its non-security relevant components such that the former would form a smaller and simpler Security Kernel;
- (6) This Kernel would be rewritten in a 'verifiable' high order language and formal argument would be produced to substantiate the claim that the Kernel enforced a security policy as an implementation of the Reference Monitor Concept;[4]
- (7) Only the Kernel would execute in Privileged Mode, so only the Kernel would be capable of performing security relevant operations (because of hardware enforcement);
- (8) The remainder of CP (the Non Kernel Control Program, or NKCP) could essentially remain unchanged except for the requirement that it operate in problem state and properly interface with the Kernel, thus eliminating the cost and risk of implementing an entirely new system; and finally,
- (9) Since the Kernel and NKCP would together simulate CP, most applications that could operate under VM/370 would also operate under the new Kernelized system KVM/370.

During this phase of the study, the SDC researchers were in active contact with the DARPA Computer Security Working Group and with an oversight committee of other Government-selected computer security experts. As a result of interaction with these groups, the emerging KVM security architecture was subjected to thorough peer review on a quarterly basis for somewhat more than the first two years of its evolution. Joint initiatives by SDC and the Department of Defense resulted in periodic interactions between the SDC researchers and the IBM VM/370 Development Group. The product of our labors and the helpful criticisms given us during the reviews was the design and formal specification of KVM/370 that were published in the Spring of 1978, at which time DARPA awarded SDC an implementation contract for the development of a KVM prototype.<3>

A preliminary version of the KVM prototype was demonstrated to a select group of DoD personnel in October, 1979. An Alpha-test version of the prototype was installed at the Naval Air Test Center, Patuxent River, Maryland in April 1981 for preliminary testing and measurement. Planned measurement tasks include some analysis of user acceptance of the security interface to the system, in addition to the all-important measurement of system throughput. Data obtained from the Alpha test site will be used to identify system bottlenecks, and serve as a basis for tuning (optimizing) the performance of the ultimate system. It is

<3>

The interested reader is referred to [16, 17] for a more detailed treatment of the KVM architecture and its rationale than is presented in this report.

planned that a test version of the prototype system, the KVM-Beta Release, will be installed at selected government sites in the late Spring of 1982 for more intensive testing and user evaluation. At that time, the formally verified specification of KVM, along with all system documentation, will be delivered to the Government in order that KVM may be considered for certification to operate in a multi-level environment.

3.0 KVM ARCHITECTURAL CONSIDERATIONS

3.1 Retrofit Strategy

A methodology was developed for partitioning the VM/370 control program (VM/370-CP) into security relevant and non-security relevant modules. The decision process is based on the principles of least privilege and least common mechanism. It defines security relevant code in CP as that code which executes privileged instructions or the code which accesses global system data (e.g., the control blocks used to support processes on the real machine necessarily contain data from all security levels). Using this definition, security relevant CP modules are directly identifiable.

It was found that most CP system data need not be truly global, but global only over the VMs operating at a given security level. The VMs at a given security level<4> could be supported by a combination of a

<4>

The reader is referred to Section 3.2 for an explanation of the security level concept.

formally verified Kernel operating in real supervisor state and a Non-Kernel Control Program (NKCP) executing in real problem state and consisting of most of the non-security relevant VM/370-CP code. The NKCP would execute as a virtual machine, having access only to global system data for the virtual machines it is supporting at its security level.

The reader should note the significant difference between a security retrofit to VM/370 and a new design of a secure VM/370. In both cases it is necessary to design and specify the security enforcement mechanisms for the security Kernel, as well as to derive the set of formal security invariants the Kernel must preserve over the system. It is found, however, that much of the code in an operating system that virtualizes a computer, such as that found in VM/370-CP, either has no security relevance or can be trivially modified<5> so that it no longer has any security relevance. Hence, much of the existing code which provides functional capabilities to the virtual machines is essentially usable as it stands.

<5>

While these modifications are straight-forward, there does remain the more difficult task of interfacing the old code and data structures with the Kernel and its redesigned data structures. It was found that considerable planning was required in order that the interface could evolve smoothly during actual implementation. Control and flexibility were achieved through the creative and judicious use of macros and CMS EXECs. The project did not attempt a parallel effort of performing a direct reimplementaion of VM in lieu of the retrofit, so there is no empirical data with which to compare levels of effort for future related endeavours on other operating systems.

Secondly, such code, in fact all of VM/370, can be virtualized. A strategy derived from these observations involves designing and verifying a relatively small body of code which is just powerful enough to provide primitive virtualization and which controls all forms of I/O access in compliance with the security policy. It is thus conceptually possible to run numerous copies of VM/370 atop this simple kernel, each running in virtual supervisor state. Since the Kernel is the final arbiter and all access to real devices must eventually pass through it (these accesses all require invocation of privileged instructions, i.e., real supervisor state), no action of the virtualized VM/370-CP can compromise security. The users would run their programs atop the untrusted copies of virtualized VM/370. If a virtualized VM/370-CP were to attempt to perform actions contrary to the security policy, the Kernel would prohibit such actions from taking place. These potential denials of service could be avoided by deleting the related code from the virtualized VM/370-CPs, but it is important to observe that these matters have no effect upon the enforcement of security since (1) the Kernel would have been designed to enforce a specific security policy, (2) the Kernel would have been formally verified to support the enforcement of that policy, and (3) the correctness proof of the Kernel made no assumptions about any of the virtual machines running atop the Kernel, particularly none with respect to an NKCP itself.

In the interest of enhancing the performance of such a Kernelized system, it would be necessary to give certain system modules access to multi-level system data. These are the modules which control the

sharing of real system resources among virtual machines at different security levels. In order to maintain system security, it is necessary to ascertain that such resource management modules properly use the privileges granted them by the added common mechanism. Such modules become 'Trusted Processes'. Where possible and practical, the Trusted Processes are to be given the same formal verification the Kernel Processes should receive.

Where this is not practical or possible, <6> the Trusted Processes must be subjected to a thorough integrity audit, encapsulated into a limited address space, and restricted so that they operate in real problem state with virtual addresses. These latter processes are known as Global Processes.

3.2 Security Policy

The KVM/370 Kernel is designed to enforce a formal model of the military security policy. This requires the preservation of two security conditions, the 'Security Property', and the 'Confinement Property' (or 'X-Property'). [9, 10] These properties are described in terms of three types of entities: subjects, objects and security levels. Subjects are the active elements of the system for which data access must be con-

<6>

The Global Processes serve as schedulers and allocators of global resources and have the potential to be used as an illicit signalling path in violation of the MITRE Confinement Property by modulating the global state variable, time. There is no known method for formally demonstrating that an algorithmically correct, Trojan Horse-free, scheduler cannot be manipulated by users in such a way that the users can cause clock time to become a signal to other users.

trolled (e.g., users, processes). Objects are the data or data containers, access to which must be controlled by the Kernel. There is a security level associated with each subject or object. The security level of a subject represents the subject's 'security clearance', serving as a measure of the level of trust placed in the subject; the security level of an object represents the object's classification. A security level $\{C, K\}$ consists of two components: a hierarchical classification C from the ordered set {Unclassified, Confidential, Secret, Top Secret}, and a category K consisting of a subset (possibly empty) of the set of special access compartments.<7>

The Security Condition requires that no subject may access an object for the purpose of reading or updating unless the security level of the subject dominates that of the object.<8> The Confinement Property demands that a subject may have write access to an object (permission to both read and write) only if the security level of the subject and object are precisely the same security level.<9>

<7>

A security level SL1 is said to 'dominate' a security level SL2 if the hierarchical component of SL1 is at least as high as the hierarchical component of SL2 and all of the compartments of SL2 are contained in the compartments of SL1. The two security levels are called 'non-comparable' if the compartments of SL1 are not a subset of those of SL2 and those of SL2 are not a subset of those of SL1. Therefore, the dominates relation is a partial order on security levels.

<8>

If the security levels of the subject and the object are non-comparable, then the subject cannot obtain read access to the object, even if the subject's hierarchical security level is higher than that of the object.

<9>

National Security Policy requires that a subject, in addition to possessing the proper security clearance, must have an appropriate 'need to know' for the data in any classified object to which he is to be

In the main, subjects in KVM/370 are interpreted as the individual NKCPs. The Kernel provides isolation among the NKCPs, but provides little or no additional isolation between VMs under the same NKCP beyond that already provided in VM/370. Since all VMs operating under the same NKCP act at the same security level, the Kernel protects each VM from other VMs at different security levels, but not necessarily from VMs at the same level. Global Processes, which must interface with several NKCPs at different levels, are also treated as subjects.

The objects in KVM/370 are collections of data areas on DASD devices (or the entire DASD volume), tape volumes, unit record devices, real core pages, processes, and VM working environments (control blocks, scratch storage, registers, etc.).

The evolution of United States National Security Policy has been monitored closely in order to make KVM/370 more responsive to the realistic needs of Government agencies. Recent modifications to this policy, documented in Executive Order 12065, DoD 5200.1R, and Air Force Regulation 300-8, make it clear that although computer systems must not divulge information to unauthorized individuals the overclassification of data is also prohibited. KVM/370 enforces the Confinement Property to prevent unauthorized declassification of data, and produces detailed historical collateral classification information for every new volume created by the system. This access record should help establish a cred-

accorded access. This latter requirement is a form of discretionary access policy, and is enforced in KVM through the use of passwords, as in VM/370, and through the use of access control lists. These controls are maintained and enforced by the Kernel.

ible data classification as a function of the classifications of all data from which the virtual machine that created it had access. This historical classification information may then be reviewed by a security officer possessing original classification authority in order to determine the appropriate classification for the data.

3.3 Overall System Architecture

The architecture of Kernelized VM/370 (KVM/370), consists of the following domains:

1. The Kernel and verified Trusted Processes, <10> executing in real supervisor state;
2. The audited Global Processes, having access to some global system data, executing in real problem state, but having access only to virtual addresses;
3. The NKCPs, one per security level, <11> having

<10>

At the time of KVM's design, we distinguished between Kernel and Trusted Process. Our distinction was related to verification dependencies. The KVM Trusted Processes are subject to modification more frequently than the rest of the Kernel, since they implement login protocols, accounting algorithms, the KVM security policy, etc.. They are designed so that the Kernel's formal verification is not influenced by modifications in Trusted Process modules. Consequently, if a Trusted Process is modified, our design discipline requires that only it be reverified.

Other investigators do not make this distinction, and would interpret our Trusted Processes as part of the Kernel, since both are accorded the same level of formal verification. Under the definitions used by the other investigators, KVM's Global Processes would be called 'Trusted Processes' or 'Non Kernel Security Related' (NKSR) Software.

For historical consistency, KVM will continue making the distinction.

<11>

Here, of course, we mean one NKCP data area per security

access to system data for the supported security level only, executing in real problem state, having access only to virtual addresses;

4. The user VMs, each controlled by the appropriate NKCP for its security level, executing in real problem state.

Formal specifications for the KVM/370 security Kernel and the four Trusted Processes<12> were written in the SDC formal specification language Ina Jo. [18]<13> The specifications, written at a comprehensive level of abstraction, are documented in the SDC TM-6062 series.

It was intended that all Kernel code and Trusted Process code would be written in a strongly typed Pascal-based programming language such as the EUCLID[5] language in order to facilitate formal verification.

However, as the time for system implementation drew near, it was found that there were no available production quality compilers for EUCLID or any other suitable, verifiable programming language that would permit efficient system programming on an IBM/370 machine base. The require-

level. The NKCP code is implemented reentrantly, and is generally resident in a unique copy.

<12>

These trusted processes are: The Authorization Process, controlling Login, user authentication, the interpretation of Security Policy for the entire system, and the management and allocation of unit record devices between security levels; The Updater Process, which maintains the security profile data base used by the Authorization Process, and communicates with the system security officer; The Operator Process, which provides services for the operator of the real computer; and the Accounting Process, which produces bills for computer services while not serving as a high-bandwidth illicit communication channel for would-be interlopers.

<13>

Ina Jo is a Trade Mark of the System Development Corporation.

ment that the system programming language permit addressing and manipulating IBM/370 dependent data structures is essential since the Kernel must analyze, prepare, and maintain numerous tables and control blocks whose structure is dictated by the hardware. In order to provide for the future verification and certification of KVM/370, it is desirable that a maximum of detail on the manipulation of these data structures be expressed in the higher-order language rather than in assembly language and that the compiler reliably produce efficient executable code, lest the performance costs of the system be impractically high. It is also necessary that the compiled code not require a run-time support package for its execution, since the run-time package could be a possible source of security compromise.

After serious consideration of numerous languages for which compilers either existed or were proposed, it was decided by DARPA that system efficiency was sufficiently important to permit the use of a programming language that was not Pascal-based, providing it satisfied the other exigencies of the project.

The language selected for the implementation of KVM/370 was the J3 dialect of the JOVIAL Programming Language, whose optimizing compiler is in use on the AWACS project[11]. JOVIAL was not designed to support formal verification. Consequently, while the formal specifications of KVM/370 are currently being formally verified against the security policy enforcement criteria, the first implementation will not be formally verified. Subsequently, when a production quality compiler exists for a verification-oriented systems programming language, KVM/370 could be recoded in that language and then verified.

In the meantime, rigorous programming conventions were established for producing JOVIAL code from the formal Ina Jo specifications. In particular, data templates were produced for mapping Ina Jo data structures into JOVIAL tables according as they were large or small, and densely or sparsely populated. Similarly, coding templates were produced for reference conditions, existentially and universally quantified expressions, etc. Where practicable and within the constraints of the compiler's symbol tables, naming conventions for Ina Jo state variables were transferred directly to the corresponding JOVIAL variable identifiers. In this way, attempts were made to maintain an informal correspondence between the formal specifications and all trusted code.

3.3.1 Design Tradeoffs

The user's system-use expectations influenced the KVM architecture, size of the Kernel and Trusted Processes, overall system performance, and level of effort required for implementation and formal verification. Resource scheduling and management can be performed on either a system-global or an NKCP-local basis. If done on a system-global basis, the size of the Kernel and Trusted Processes is increased, the interfaces between the NKCPs and the global processes become more intricate, verification becomes more difficult and costly, system modification becomes less facile, but system performance improves. If done on a local basis with most resource management decisions performed by the NKCPs and perfunctory reconciliations performed by the Kernel, the opposite results hold; system design, implementation, verification and inter-

faces are simplified, while system performance may be adversely affected.

In terms of greatest all-around adaptability to applications, ease of implementation and verification, and best multi-level security, we concluded that:

- * DASD page areas will be global;
- * Main page frame management will be based on global allocation with global page replacement;
- * Multi-Level shared reentrant systems will be provided with all shared pages locked into core;
- * The CPU will be scheduled by the NKCPs.

3.3.2 Kernel and Trusted Process Design

The Kernel and Trusted Processes are the portions of KVM/370 whose formal specifications are being formally verified. The system has been designed such that there are no 'upward' functional dependencies (i.e., no function at level of abstraction i depends on any function from level of abstraction j for its correct operation, if $i < j$) [6] and [7]. In this way, it can be demonstrated that no trusted code depends on the correctness and non-maliciousness of any untrusted, unverified code. Further, the formal verification of KVM/370 will require only that (1) the Kernel and Trusted Processes be shown to satisfy the requirements of enforcing the security policy, and (2) the absence of unauthorized signalling capabilities within the Global Processes be demonstrated.

In the case of the Start I/O request to the Kernel, the entire channel control program is copied into a portion of the Kernel's domain where it

is protected from modification by asynchronous attack. Address translation is performed on the channel program. Then it is legality-checked by the Kernel to guarantee that the program performs only valid accesses, that all referenced pages are locked into core, and that the channel program is not self-modifying and contains no puns or other security threats[6]. The I/O scheduler is then invoked and control is eventually passed to the dispatcher module, simulating an I/O Fast Release. When the I/O interrupt finally takes place, the relevant pages are unlocked, and the condition code is passed as an interrupt to the appropriate NKCP.

Each security level has a unique address space for use by its NKCP. With the exception of the functions enumerated below, no NKCP can communicate outside its address space or its VMs' address spaces. Spool files, virtual channel-to-channel adapters (CTCA), and inter-VM messages are handled by the NKCP and consequently cannot violate the Kernel's enforcement of the security policy.

The notable exceptions are:

- * Append-up for writing machine error records and accounting data which are processed at the highest security level in the system;
- * Read-down to obtain access to a DASD whose classification is dominated by the clearance of the user's VM.

For purposes of design simplicity, each NKCP will appear to be uninterruptible just as VM/370-CP currently is, i.e., the NKCP's critical regions will be preserved. The NKCP may, in practice, be interrupted

by the Kernel, but only if no NKCP shared variable (e.g., its set of active page frames or real addresses) is modified while it is servicing a user request. This constraint on NKCP shared variables is the result of considerable effort in the area of Kernel-NKCP interaction. An NKCP terminates a critical region (a locally uninterruptable code segment) when it either schedules a VM (issues an LPSW) or when it issues an I/O request.

3.3.3 NKCP Design Considerations

The overall architecture for KVM provided us with little insight from which to go about performing the security retrofit. We knew that we had to formally specify, design and implement the security relevant portions of CP as the Kernel and Trusted Processes, and to modify the remainder of CP into NKCP and the Global Processes such that the latter did not contain security relevant code. The remaining difficult problems were discovering the right definition of 'security relevant function', identifying the associated code, and determining an efficient means of interfacing the security relevant and non-security relevant components of the resulting system.

Our initial belief was that all privileged instructions were security relevant since, in order to execute them, a process had to be in (the highly security relevant) supervisor state. While this potential violation of the Least Privilege Principle is security relevant, it neither identifies all security relevant operations, nor is it sufficiently fine-grained to be of practical use -- since CP simulates the privileged

portion of an S/370, it has a high and uniformly distributed number of occurrences of privileged instructions.

Ultimately we arrived at the following working definition: Code is security relevant if it can influence unmediated I/O directly through the use of privileged instructions that manipulate devices or user domains, or indirectly through the use of data structures that either contain security enforcement data or can be viewed and modified by processes operating at different security levels. Data is security relevant if it contains global information which traverses several security levels.

Under this definition, a considerable amount of CP code is security relevant because it makes wide use of global system tables. Many of these tables can be distributed so that each copy contains only data relevant to a unique security level. The code manipulating these distributed tables can then be made reentrant so that most of the code loses its security relevance (however, privileged instructions would still be security relevant).

Our approach was thus driven by identifying non-security relevant code in CP and virtualizing it out of the privileged execution domain. The non-virtualized portion, plus additional security enforcement code, became the Kernel. One can become tempted to virtualize almost everything out of the Kernel since the more code that is virtualized, the less there is to verify. However, since the purpose of global system tables is the efficient management of the real machine, the wholesale

partitioning of this information would cause system efficiency to degrade as code is virtualized out of the Kernel and Global Process domains.

Consequently, considerable attention had to be paid to the characteristics of KVM's expected use environment in order for us to identify those functions that would have to share global data in order for us to attain reasonable system performance efficiency. Our analysis was based on the assumption that typically KVM would concurrently support a moderate number of security levels (approximately five), and that each level would support essentially the same number of virtual machines. We analyzed a number of alternatives for each form of inter-component communication on the basis of security characteristics, performance impact, and difficulty of implementation. The resulting design and implementation are such that an installation's deviation from these assumptions could significantly alter the performance characteristics of KVM.

3.4 System Verification

Formal verification of Kernel and Trusted Processes at the specification level will demonstrate that these functions be shown to correctly implement the sharing policy between subjects and objects in terms of the basic security principle and the Confinement Property.

This form of proof involves demonstrating that all system state transitions preserve a set of security invariants. The proof of correctness will be achieved with the assistance of an interactive theorem prover (ITP) which is used to produce a fully annotated formal proof that the

state transitions of the Kernel and Trusted Processes preserve the invariance of the KVM correctness criteria.<14> Formal verification of the KVM/370 specifications is in progress, and should conclude in early 1982. Verification activity had been postponed until October 1979 because of funding anomalies. Consequently, the KVM formal specifications have been rewritten into abstract top-level specifications and more concrete refined specifications in order to facilitate the tasks of proving them and of demonstrating their correspondence to the JOVIAL implementation. The two levels of formal specifications are rigorously linked with Ina Jo mappings. The top-level specifications were formally verified as of October 1980. The second-level specifications will be submitted to the formal verification process in 1981-1982.

3.5 Possibility of Hardware Errors

The project staff was concerned over the possibility of violations of the security policy occurring because of failure of the hardware security controls. Some possibilities considered were:

- * failure of the privileged operation protection mechanism;
- * an error in address translation or the Translation Lookaside Buffer (TLB);
- * failure of the storage protection mechanism;

<14>

The ITP is one of the tools used in SDC's Formal Development Methodology. The theorems proved by the ITP are generated automatically from the Ina Jo specifications by a mechanical Specification Processor. The theorems are derived algorithmically from the Correctness Criteria, Invariants, Constraints and Mappings used in Ina Jo specifications. Full rigor is maintained throughout this process.

- * misinterpretation of a CCW by a channel;
- * an I/O device responding to the wrong device address;
- * mishandling of a command by an I/O device.

Each of these possibilities was considered in depth, and where appropriate, KVM addresses the problems through a combination of architectural and procedural (administrative) measures. The interested reader is referred to [17] for a discussion of these problems and their resolution.

3.6 Elimination of Known Security Flaws

There are about 30 security flaws known in Release 3.0 of the commercial version of VM/370. A hardened version could be produced by fixing these errors, but there might be other errors which had not been detected and the fixes might themselves introduce new security flaws. KVM/370 goes further and eliminates both known and unknown security flaws. It is instructive, however, to see how the presence of a security Kernel and the constraints it places on the system design eliminate some of these errors.

Almost every security flaw in the VM/370 system involves the input/output functions[1]. Since there is no address space validation of input/output by the hardware, other than that performed by the storage protection keys, VM/370 must validity-check all channel programs and relocate all virtual addresses, including both main storage addresses, and DASD cylinder addresses in seek arguments and home

addresses. The same I/O logic is repeated for several different requirements: virtual spooling support, virtual console support, virtual channel-to-channel adapter support, and a special VM/370 I/O interface. Each variation of this support means that errors may be present.

These errors occur in the translation of channel programs as a result of the complexity of the channel command language. For example, the same word in a channel program might be used as a command or as an operand address depending upon the execution sequence of the program[1]. Since the System/370 architecture allows "puns" in the channel, in that a word's interpretation depends on whether it is received as the leading or trailing portion of a long command, it is possible to bypass checking in these modules and access DASD records [1].

Under KVM/370, channel command words are not permitted to take on different meanings depending on the sequence of execution. Primarily, this means that an NKCP cannot submit certain channel commands with transfers or with certain modifier bits set. This does not preclude users (VMs) from constructing such channel programs, it merely requires the NKCP to put them into a standard form before submitting them to the Kernel. Further, these commands will be copied into the Kernel's data space and translated and modified there, preventing their modification by an NKCP between the time of translation and time of execution. Also, self-modifying channel programs such as those used by OS/360 ISAM, will not be permitted, since doing so would permit bypassing the Kernel's reference monitor function and open the system to the possibility of penetration.

Certain VM/370 penetrations[1] dependent upon simultaneous input/output and CPU execution are being countered by removing from the address space of the requestor all pages which are buffering input. This applies to both NKCP and user VMs. In the event either NKCP or a VM needs access to such a page, its execution will be delayed until the I/O completes and the page is made available. For example, under VM/370, careful timing of asynchronous execution could be used to exploit a bizarre oversight in condition-code checking to gain a total system penetration (real supervisor state)[1].

Although the treatment of storage and timing channels[8] is beyond the scope of a system such as VM/370, they must be controlled in a military system such as KVM/370. In this respect, we have thoroughly audited all Global Processes that allocate and schedule resources among VMs at different levels for Trojan Horses. Hence, it will not be possible to transmit information over a covert communication channel at a high enough bandwidth to make such attempts worthwhile.

4.0 CURRENT STATUS AND PLANS

4.1 Recent KVM Status and Thrusts

The Fiscal Year 1981 tasking for KVM was directed in the two following general areas:

- I. Preparation of the KVM/370-Alpha Release
- II. Installation, Demonstration and Measurement of the KVM/370-Alpha Release

In October, 1979, KVM development reached an important and significant milestone. A demonstration was given of a development version of KVM system consisting of three distinct address spaces, (the Kernel and two NKCPs) which was able to support the address spaces and functional requirements of two users at two security levels. The basic architectural design of the system had been proven feasible at that point. One fifth of the control program (20,000 assembly language instructions distributed throughout the 120,000 lines of original CP) had been significantly altered, and this successfully makes service calls to a security Kernel consisting of 3000 JOVIAL statements. This represented a major architectural accomplishment and presented a solid foundation for the addition of more functionality to the system.

Two major milestones were achieved at that time:

- (1) The Kernel supported one NKCP and two interactive virtual machines (VMs); and
- (2) The Kernel concurrently supported two NKCPs (i.e., two distinct pseudo-classified security levels) each with one active VM.

Two major test versions of KVM/370 were subsequently constructed and tested successfully, each KVM version capable of supporting two users at the same level and two users at different levels. The levels were designated "High" and "Low". Each user was able to exercise most of the control program (CP) commands, viz LOGON, LOGOFF, QUERY, MESSAGE, SLEEP, BEGIN, DUMP, SET, LINK, IPL, etc. In addition, he could perform simple CMS edit commands, using the created edit file for assemblies, etc. As expected, CMS commands that involve input/output sequences (e.g., IPL)

were executing an order of magnitude more slowly than they do in the standard, tuned, VM/370 release. This performance degradation was a temporary result of redundant NKCP page calls to the Kernel, VM/370 overhead,<15> and a few Kernel calls (JOVIAL) that take several milliseconds to complete. A portion of the KVM effort since then has been devoted to improving system performance by a combination of performance measurement and tuning.

As of November 1980, when the project had transitioned to RADC, the work on the prototype KVM/370 system had reached the point where it is possible to demonstrate the capability of concurrently supporting simple virtual machines at each of two pseudo-classified security levels, complete with paging support.

The major thrust for the project since October of 1979 has been the maturation of KVM/370 into a viable multi-level secure system. The project resumed support for an effort to obtain KVM's certification and ac-

<15>

The current test version is being supported by VM/370. As such, it is being run interpretively by VM/370-CP. Hence, the system is currently undergoing two levels of paging, and all privileged operations are being simulated by both VM/370-CP and by the KVM Kernel. By November 1980, however, the VM/370 support for the test environment was temporarily eliminated when KVM was successfully installed on SDC's IBM 4331 computer. A preliminary version of KVM-Alpha was demonstrated on the bare 4331 in October 1980. As trusted [non-pasable] code was added to the system, however, the developing KVM-Alpha prototype grew larger than the available memory of the 4331. A replacement upgrade to an IBM 4331-II was ordered.

The system development process progressed far more rapidly on the 4331 than had been the case with the remotely-accessed computers used in the past. It is expected that information gained from installations at the planned Alpha- and Beta-test sites will further improve the testing, analysis and development of the full system.

creditation for multi-level site-specific applications. In order that KVM/370 be certified for use in multi-level security applications, the Government requires evidence that KVM correctly enforces DoD security policy over its users and their processes. Although certification activities will involve significant testing, penetration analyses and documentation reviews, current DoD certification trends require formal evidence of the security-worthiness of candidate systems. This is because no feasible amount of testing can produce conclusive evidence of the absence of security flaws from a system, even though such testing may uncover no such flaws.

The DoD has begun to require that all candidate systems be presented for their review complete with formal system specifications written in a non-procedural logically complete predicate-calculus based specification language, along with formal proofs that the specifications are complete and consistent with respect to enforcement of DoD security policy. Formal models of DoD security policy have been expressed as system-specific interpretations of the LaPadula and Bell model of security produced by the MITRE Corporation. [9, 10]

The KVM/370 project originally produced an unverified single-level abstract formal specification of the KVM Kernel and Trusted Processes in the 1978 dialect of SDC's formal specification language Ina Jo, along with a formal statement of the Security Condition that is to be preserved as an invariant by the Kernel and Trusted Processes.

Under the direction of DARPA, the formal KVM specifications were written as formal implementation-level specifications. Consequently, the formal specifications for KVM were not in a form that would permit their formal verification using the formal verification tools in SDC's trusted software Formal Development Methodology (FDM).<16>

Since the start of Fiscal Year 1981, SDC has modified the KVM formal specifications to conform with the evolving requirements of the FDM. In October, 1980, proofs were produced of the new TLS. Formal verification of the SLS will begin in May 1982 under a KVM-Beta Release development contract.

4.2 Summary of Progress at RADC

Prior to the start of the KVM Technical Demonstration effort, the project plan was carefully reviewed and a new development schedule was developed. The coding schedule showed an orderly progression of 'coding rounds' that would lead to the KVM-Alpha and KVM-Beta Releases, with well-defined intermediate incremental system releases.

<16>

FDM requires that formal specifications originate from a highly abstract Top Level Specification (TLS) which is formally proven to satisfy a Criterion for Correctness. Following verification of the TLS, the TLS is refined into a more concrete Second Level Specification (SLS). Formal correspondences (mappings) are produced between the abstract TLS objects and the more concrete objects of the SLS. Formal proof evidence is then produced to demonstrate that the SLS is a faithful and correct implementation of the TLS vis-a-vis the Security Condition. That is, the EFFECTS section of each SLS state transition is shown to imply the EFFECTS section of each TLS state transition. Upon completion of this proof, transitivity of implication yields the proof that the SLS preserves the Criterion for Correctness.

Detailed functional descriptions of the Kernel and Trusted Process internals were also initiated during this time period.

Under RADC contract management, coding proceeded according to the revised schedule, and by mid-March the prototype KVM-Alpha Release was in final testing.

4.2.1 KVM Alpha Test Site at NATC

In the Spring of 1981, a test version of KVM was delivered to the government. This version of KVM is called "KVM Alpha".

KVM Alpha is a running prototype of the eventual "full KVM" system. It offers most of the functionality of Release 3 PLC 15 of VM/370, the notable exceptions being no support for virtual-virtual operating systems and no secure accounting process. The Authorization process and the Updater process offer rudimentary support for the proper separation of security levels and virtual machines,<17> but is not readily amenable to

<17>

The function of the Updater process is to maintain the integrity of the security profile directories of users, virtual machines, and peripheral devices as they are updated by the system security officer. Some of these updates are reflected immediately in the user's directory, whilst others are delayed until the user does not have an active process on the system. The data bases built and maintained by the Updater are used by the Authorization process (and its Initiator component) as the fundamental means of interpreting security policy for all active subjects and objects in the system. In effect, the Authorization process must act as a small multi-level data base management system that implements appropriate views of user directories for the KVM Kernel, Trusted Processes and NKCPs. These views are derived from a set of Third Normal Form relational data bases, through a filter which interprets DoD security policy under a model similar to the MITRE security policy model. The Updater process provides the data base generation, maintenance, and update/recovery functions, and has responsibility for overall preservation of data base integrity for KVM. Since the data

rapid modification of user directories, nor does it support protected special-access passwords or the construction of audit trails. Dial-up access is not supported in KVM Alpha.

Essentially, KVM Alpha will be a basic test vehicle for performance measurement and user evaluation/familiarization. New features will be incorporated into KVM Alpha throughout FY81 and into FY82, until the Summer 1982 release of the complete system.

KVM Alpha was installed at the Naval Air Test Center, Patuxent River, Maryland, between 29 March and 10 April 1981.

4.2.2 Experiences During the NATC Installation of KVM-Alpha

The month of April marked the successful installation of the KVM-Alpha prototype at the Naval Air Test Center, Patuxent River, Maryland. The installation was performed over the period 29 March through 10 April by a team consisting of Dick Linde, Pat Ward, Guy Schroeder and Marvin Schaefer at NATC; with contemporaneous support provided by Robin Peeler and Marty Massoslia at SDC in Santa Monica.

SDC would like to acknowledge the high level of support given to the project by Mr. Gordon Fullerton and Mr. Danny Weddel of NATC during the two weeks of the installation. NATC provided the KVM project with third shift dedicated computer time on weekdays; additional shift hours were

base management functionality of the Authorization process is conceptually much simpler than that of the Updater; the initial release of KVM Alpha does not provide Updater facilities directly. Rather, the KVM Directory is manually prepared and compiled directly into the release for each test site.

provided on 29 March and on 4 April. Because of installation difficulties encountered at NATC, RADC postponed the installation of KVM-Alpha at the designated Army Communications Agency test site to some future date.

The KVM-Alpha installation at NATC was plagued with unanticipated technical difficulties. A majority of the problems were related to incompatibilities between SDC's IBM 4331 system environment (hardware, software and configuration) and NATC's AMDAHL V7A system environment. Other major problems arose in the areas of the IBM 8809 Tape Drive, and the KVM Directory Macros. These are discussed below.

The installation team carried two complete 4-tape backups of the KVM-Alpha system to NATC, including source code, the loadable (under VM/370-CP) 4331 configuration, documentation, editors, etc. Mr. Fullerton provided the staff with disk space, user IDs, and revised configuration data. Repeated attempts failed to restore our files from the backup tapes using either the IBM program VMFPLC2 or the University of Waterloo PETAPE utility. Analysis of tape dumps (made by Mr. Fullerton with MVS utilities) confirmed that the tapes were compatible with the NATC tape drives<18> and were produced under SDC's version of PETAPE. It was then determined that NATC was using an older, incompatible, release of PETAPE.

<18>

For a while it was suspected that they might have been generated in data-streaming mode by the IBM 8809, in which case new backup tapes would have had to be created and shipped from SDC.

A new tape was constructed at SDC, using standard IBM utilities, containing PETAPE and some key modules. The tape was air freighted to NATC, but was not received for three days. Meanwhile, Guy Schroeder decrypted the PETAPE formatting code (undocumented) and reconstructed PETAPE from our backup tapes, using Mr. Fullerton's knowledge of the MVS tape dump utilities. Once this was done, it was possible to retrieve the files from the backup tapes. Unfortunately, two nights were lost because of this problem.

It was subsequently discovered that some of the files on the backup tapes had not been copied successfully from the 3370 disk. The problem, identical on both sets of backup tapes, involved several consecutive file pairs: in each pair, the two files were written contiguously such that the last part of the first file and the first part of the second file of each pair were missing. Not all of the damaged files were essential to the KVM-Alpha installation: some of the damage was to source code; some was to object code; some was to utilities. New tapes were shipped to NATC containing the most essential modules, including the JOVIAL compiler. As file-damage was discovered in other essential files, the missing data was recovered as hard-copy listings obtained through a 300-baud remote dial-up to the SDC 4331 in Santa Monica. (The SDC 4331 is remotely accessible via TYMNET and Dial-Up lines.) The missing data was then manually typed into files on the NATC machine and incorporated into the system. (In most but not all cases, this required typing in less than 100 lines of JOVIAL or assembly language.)

Once the KVM-Alpha system was restored, an attempt was made to load it under VM/370 for preliminary testing under the NATC configuration. Spurious error messages were received on the AMDAHL console, and the system crashed repeatedly and nondeterministically while KVM was being IPLed or shortly after the IPL sequence ended. Most of the errors indicated problems in an I/O Control Unit. Attempts to isolate the error failed. The NATC VM system failed only when attempts were made to run KVM under it.

Guy Schroeder learned from other AMDAHL users that there is a known, but relatively undocumented, bug in the AMDAHL architecture that prohibits running a 2K virtual paging system (e.g., KVM, DOS/VS, etc.) under a 4K virtual system (e.g., VM/370, MVS, etc.). The suggested resolution to the problem involved "cutting" cache memory in half, a modification (done from the operator's console) that severely impacts AMDAHL performance. The modification was made and, on the fourth night, it was first possible to bring the test system up under VM/370 on the modified machine.

The NATC configuration had changed somewhat by the time the installation team arrived at NATC. In order to represent the new configuration, it was necessary to construct a new KVM Directory with the Macros that had been written as temporary tools for this purpose.

Several problems were encountered in implementing the revised directory, primarily due to the complexity of the macros. The majority of these problems were overcome. Each attempt, however, required a recompilation

of several KVM modules, since the System Security Officer functionality of the Updater Process is not scheduled for implementation for several months.

It was found that under KVM, users must have a writable A-Disk at each security level at which they intend to use CMS. This required the creation of a large number of A-Disks for the user IDs for which we had configured the system. (A user cleared to TS, having two categories in his clearance, could require as many as 16 distinct A-Disks in order to operate at all of his potential security levels.)

4.2.3 NATC System Performance Test

The KVM-Alpha system prototype was successfully IPLed and executed on the bare NATC AMDAHL V7A on Friday morning, 10 April 1981. It was possible to run one limited performance benchmark test prior to releasing the machine for NATC's normal daily operation. This test involved the running of a JOVIAL compilation of several KVM Kernel modules. The compilation was run first under VM/370-CMS in stand-alone mode; it was then run in stand-alone mode under KVM-CMS on the bare AMDAHL V7A. The test results were (in wall-clock time): under VM/370 -- two minutes, 11 seconds; under KVM/370 -- four minutes, 21 seconds. The compilation was I/O bound; the testcase included a combination of 100 syntactic and semantic JOVIAL errors.

Earlier, during the second week of the KVM-Alpha installation, Susan Rajunas (MITRE) tested the system Execs for NATC's PL/I compiler. Unfortunately, during the period of her visit KVM-Alpha could not yet be

executed on the bare V7A because of hardware-specific problems in the KVM IPL sequence. These problems were not resolved until she and Major Darr had returned to their organizations. The PL/I test cases could not be run because documentation for the use of the PL/I compiler was unavailable at the time of the tests.

It had originally been intended to run at least the following performance tests: a common job with one user, two users on the same NKCP, and two users on two separate NKCPs; then a mix of jobs on one NKCP, and then on two NKCPs; and finally, the above on the machine with different amounts of memory available (in order to force paging activity). However, KVM-Alpha did not reach a demonstrable state until a few hours before the end of the final run on the NATC computer. In final preparation for the performance tests, the installation team encountered timing problems with the TELEX IBM 3278-compatible terminals. The Directory also had to be modified when it was discovered that some line addresses improperly designated 3278 equivalent terminals as 3277s.<19>

Preparations had been completed for the demonstration of KVM-Alpha and the training classes that were to have been given to the NATC operators and users. Mr. Fullerton was shown how to IPL the Alpha system, and was

<19>

Of the two 3277s in the machine room, it was not possible to use one for testing an assembly concurrent with the compilation because the Directory defined the minimum pseudo-security level of both terminals as higher than the security level of the A-Disks (and clearance) of the user with access to the assembler. There was not sufficient time to recompile the Directory to run this test. We believe that it would have been possible to run a more statistically meaningful set of performance benchmarks had the team been able to obtain access to the AMDAHL for one more shift.

present for some of the testing under VM/370. Major Darr logged into KVM-Alpha under VM/370, and entered a few CMS commands. Major Darr and Ms. Rajunas were given copies of the Operator's and Users' Guide for KVM-Alpha, and of the visuals that had been prepared for the training class and demonstration. Other than this limited exercise, no demonstrations or trainings were performed.

Since KVM-Alpha exceeds the memory size of the 4331-I, and because of the project's continuing need for the 4331-I as a support machine, it was not possible to perform bare machine performance tests on KVM-Alpha at SDC. Additional performance testing is planned for a return visit to the NATC test site, and for the SDC 4331-II in future months.

4.3 KVM Plans

Since its initiation in 1976, the primary goal of the KVM project has been to produce a certified multi-level secure system that is essentially compatible with the commercially available IBM VM/370 system. This compatibility would permit KVM's user community to install their existing 370 applications with minimal modification.

KVM development has reached the point where it is appropriate to address the requirements of its potential user community. Over the next one and a half years, the system will be submitted to a selected user group that will participate in its early testing and evaluation. In the Summer of 1982, KVM should reach the point where it can be submitted to the government, along with substantiating formal documentation, for multi-level certification consideration.

In Fiscal Year 1982, the KVM prototype will be matured from a test environment into an operational environment, that can be supported by a competitively selected system maintenance organization.

4.3.1 KVM Alpha Test Sites

Currently, it is planned that a second visit be made to the NATC Alpha test site to install an updated and corrected version of KVM-Alpha. At that time, extensive performance benchmarking and measurement activities will be conducted. The second visit will conclude with a demonstration of the KVM-Alpha release to NATC site personnel.

4.3.2 July 1982 KVM Beta Release

In July 1982, KVM will have been tested and tuned during a year of user evaluation. Concurrent with its testing, additional features will be added to the implementation so that the resulting version, 'KVM Beta', is functionally consistent with all of the VM/370 Release 3 PLC 15 features it was intended to support.

During the testing period, the evaluators of KVM Alpha may identify system 'rough edges': quirks that make KVM Alpha difficult to use. The KVM implementation team will analyze the nature of these quirks and, where consistent with budget, schedule and security constraints, make appropriate modifications to the KVM user/security-officer interface. KVM Beta, the system evolved from this crucible of testing and evaluation, should be a more practical system than its predecessor.

Any known security deficiencies, either those indicated from the formal specification verification effort, or those indicated by the user community, will have been analyzed and resolved. These security deficiencies may be of several types. These are: situations in which the specification cannot be verified without modification; <20> situations in which formal analysis of the specifications uncovers an actual security design flaw; <21> and those in which a coding error is discovered through testing. The latter can occur because of the fact that the code is not being formally verified for its consistency with respect to the formal specifications.

KVM Beta, then, will be a tuned system which, to the best of our knowledge, is a complete and certifiably secure implementation of KVM, tested by outside users as well as its implementors, and in reasonable correspondence with its verified formal specifications. However, the tasks described in Paragraph 4.3.4, below, will be required in order to more formally demonstrate the correspondence between specification and implementation.

<20>

This condition can arise as a consequence of current limitations in the verification technology, and has no relationship to the security properties of the underlying system. Of course, modifications to the specification would have a corresponding impact on the implementation.

<21>

Here, the specifications would have to be modified and reverified, and the implementation correspondingly modified.

4.3.3 Formal Documentation for Certification

In order to support the detailed laboratory analysis required for KVM certification, we will annotate the formal KVM specifications so that the evaluators can understand the security model and the abstractions used in the specifications. This annotation will be provided in the form of an English language companion document to the specifications and their associated mappings.

In addition to the formal specifications, the government will be provided with the annotated proofs. These proofs will show that the transforms for the Kernel and Trusted Processes preserve the security correctness criterion as an invariant. The second level specifications will be shown to be consistent with the Top Level Specifications.

If requested by the government, additional formal analysis could have been performed to prove the internal self-consistency of the transforms. This analysis involves proving the truth of a set of automatically generated existential self-consistency conjectures.

Finally, documentation will be produced detailing the correspondence between the verified formal specifications and the implementation.

Accompanying this documentation will be an analysis of any known security anomalies or deficiencies in the KVM design or implementation. This latter report will include a listing of usage restrictions on KVM implementations, and all known covert signalling channels along with their estimated worst-case bandwidths. Because of the nature of its contents,

the government may require that this report be published as a classified document.

4.3.4 Preparation for Certification

The KVM Beta system released in July 1982 will be a tuned, user-oriented system that is essentially compatible with VM/370 Release 3 PLC 15.<22> KVM Beta will support special-access passwords, and prepare protected audit trails for each mini-disk. The audit trail for each mini-disk to which a subject has written data will contain data from the classification header for those mini-disks to which the subject had read access during his session, in concert with the KVM security policy. This data can be used to aid the security officer in justifying the classification of a mini-disk. KVM Beta will support dial-up access terminals, although certain installations may elect to not enable this feature.

The Top-Level Specifications and the refined Second Level Specifications will have been formally verified against their correctness criteria. KVM Beta will essentially be ready for submission for consideration for certification in multi-level environments as well as for protracted penetration analysis by outside agencies.

<22>

The commercially available version of VM/370 permits the user to perform certain operations which result in his assuming supervisor state on the computer. Since the KVM Kernel prevents these penetrations from occurring by not supporting these operations, there are incompatibilities between the two systems.

The remaining step required to satisfy certification submission requirements is a demonstration of the correspondence between the KVM implementation and its specification. During the formal verification activities, it is possible that the formal specifications will have been modified in order to achieve verification. The implementors will have been notified by the verification staff of such modifications, and attempts will have been made to make corresponding modifications to the HOL implementation. However, there may still exist discrepancies between the verified formal specifications and the HOL code.

A final manual pass will be made over the KVM Kernel and Trusted Process code in order to ensure that the required correspondences are perspicuously consistent with the specifications. The transformations between Ina Jo specification constructs and JOVIAL J3 constructs will be documented. Any non-obvious transformations, made in the name of efficiency, will be explicitly documented.

4.3.5 Maintenance Support

It is not expected that the test site will immediately place heavy demands or expectations on the initial release of KVM Alpha. While recognizing that KVM Alpha is a prototype system, SDC will provide a moderate level of support and maintenance to the selected test site during the period of KVM's evolution from KVM Alpha to KVM Beta.

Once KVM has been tested and tuned, it will have reached the release stage. At this point, the government plans to release a Request For Proposal to the software community for the purpose of selecting, as was

done in the case of the Kernelized Secure Operating System, an organization that will professionally support and maintain the releases of KVM in its installations.

4.3.6 Documentation

KVM system documentation will be updated so that it correctly represents the KVM-Beta Release of the system. In order to support the KVM maintenance staff, we will produce a set of 'difference' or 'exception' documents to complement IBM's existing Release 3 PLC 15 VM/370 Control Program system maintenance documentation.

A User's Guide to the security features of KVM, as well as documents for the System Security Officer and for computer operators will also be produced.

SDC will maintain an internal project record documenting the installation and checkout experience of the KVM Alpha test site. The objective of this effort will be to identify configuration-specific/installation-specific anomalies in order to establish direct system installation procedures that will efficaciously circumvent both common and site-unique problems for installing future versions of KVM.

4.3.7 KVM Upgrade to Support Selected VM/370 Release 6 Peripherals

KVM Beta will be upgraded to support some devices not supported by Release 3 PLC 15 of VM/370. Included in this set are the IBM 3278 terminal, and a number of new storage devices such as the IBM 3370 series disks.

Completion of the modifications to KVM device support will permit KVM Beta to execute on the IBM 43xx series mainframe.

5.0 ACKNOWLEDGMENTS

KVM would not be what it is had it not been for the knowledge, skill and dedication of the KVM Development Staff. Because of the difficulties involved in obtaining access to a KVM development computer, these computer scientists have worked more by moonlight than by sunlight on Government computers so distant that turn-around for delivery of computer listings averaged one week following their generation. The author, as the outgoing KVM Program Manager, is honoured to have worked with Barry Gold, Dick Linde and John Scheid on the KVM design; with Pat Ward, Robin Peeler, Guy Schroeder and Marty Massoslia on the implementation and testing, and with Sue Landauer, Judy Hemenway and Judy Stein on its verification. A special nod of thanks goes to Clark Weissman, Tom Hinke and Harvey Gold for holding the project leadership reins during my extensive travels.

The author wishes to acknowledge the contributions, support and suggestions made by Mr. Steven T. Walker, Chairman of the Department of Defense Computer Security Technical Consortium, whose enthusiasm and encouragement made the project and its realization possible.

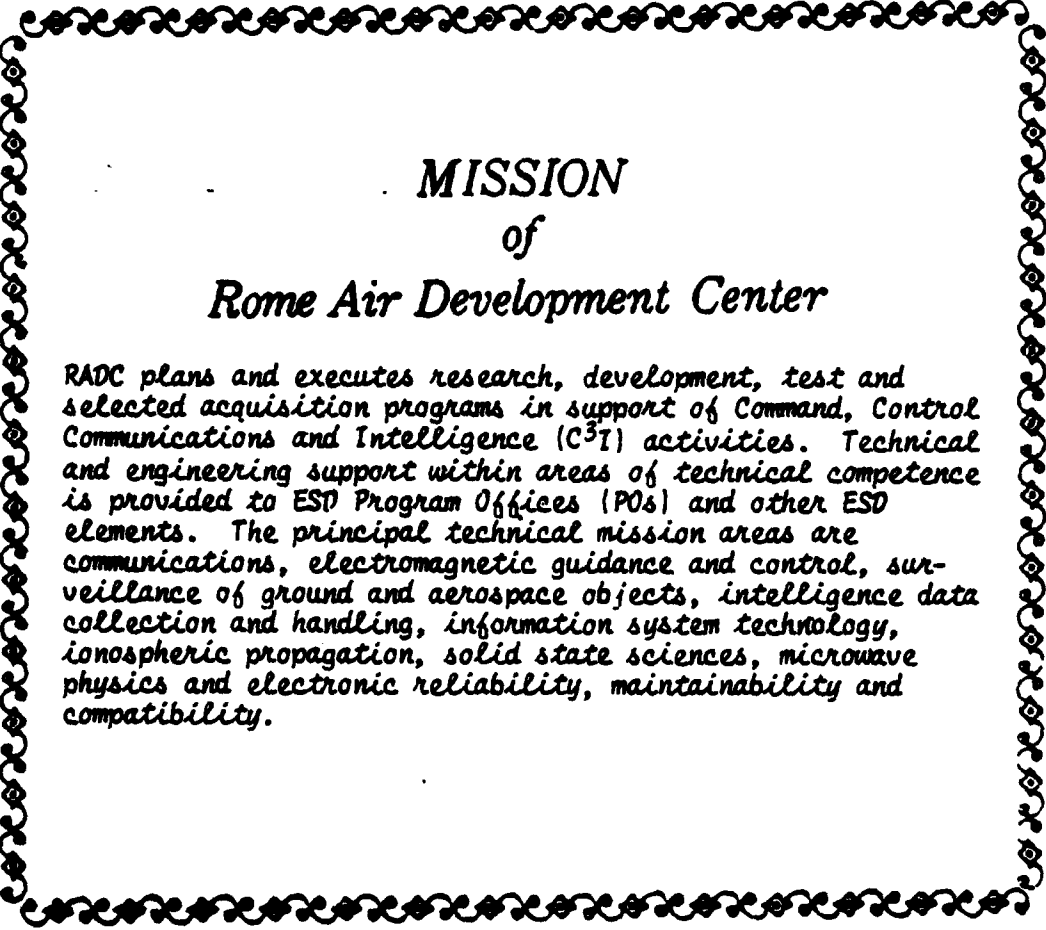
The project has been favoured with the technical and managerial attentions of a growing community of dedicated professionals, of whom we should like to thank the following individuals: S. Ames, J. P. Anderson, C. Andresen, P. Beck, R. Bilek, E. Book, M. Branstadt, E. Burke, W. E. Carlson, P. Cohen, M. Corasick, T. C. Darr, S. Durowski, D. Edwards, W. Eisner, J. Fergusson, J. Fraley, J. Frustace, G. Fullerton, N. Hardy, D. Hollingsworth, G. Goldberg, J. Grewe, J. M.

Ives, A. K. Jones, B. Lampson, J. Lathrup, H. C. Lauer, T. M. P. Lee, R. E. Lyons, E. J. McCauley, D. McIntyre, C. A. Melkerson, M. Moriconi, P. J. Neumann, M. J. Orceure, G. J. Porek, S. A. Rajunas, C. A. Savant, W. M. Shasberger, S. J. Sherman, L. Stans, P. Tasker, D. H. Thompson, W. Wilson, J. P. L. Woodward, J. Wright, J. H. Yott, and D. Zucker. We would also like to extend our appreciation to our technical editor, J. A. Stein; and to J. Brookshire, L. L. Parris and N. Spies, who provided typing support.

6.0 REFERENCES

- [1] C. R. Attanasio, P. W. Markstein and R. J. Phillips (SDC), 'Penetrating an Operating System: a Study of VM/370 Integrity', pp. 102-116, IBM Systems Journal, vol. 15, no.1, International Business Machines Corp., 1976.
- [2] L. A. Belady and C. Weissman, 'Experiments with Secure Resource Sharing for Virtual Machines', SDC Publication SP-3769, System Development Corporation, 15 May 1974.
- [3] C. Weissman, 'Secure Computer Operation with Virtual Machine Partitioning', National Computer Conference Proceedings, 1975, pp. 929-934.
- [4] J. P. Anderson, 'Computer Security Technology Planning Study', James P. Anderson and Co., Fort Washington, Pa., USAF Electronic Systems Division, Hanscom AFB, Ma., ESD-TR-73-51, vols. I and II, October, 1972. (AD 758206 and AD 772806)
- [5] B. W. Lampson, J. J. Hornins, R. L. London, J. G. Mitchell, and G. J. Porek, 'Report On The Programming Language Euclid', Xerox Research Center, University of Toronto, USC-ISI and UCLA respectively, December 1976.
- [6] P. Janson, 'Using Type Extension to Organize Virtual Memory Mechanisms', MIT/LCS/TR-167, Massachusetts Institute of Technology, September, 1976.
- [7] D. P. Reed, 'Processor Multiplexing in a Layered Operating System', MIT/LCS/TR-164, Massachusetts Institute of Technology, June, 1976.
- [8] B. W. Lampson, 'A Note on the Confinement Problem', Communications of the ACM, October, 1973, pp. 613-615.
- [9] D. E. Bell and L. J. LaPadula, 'Secure Computer Systems: A Refinement of the Mathematical Model', MTR-2547 vol III, the MITRE Corporation, Bedford, Massachusetts, 28 December 1973.

- [101] D. E. Bell and L.J. LaPadula, 'Computer Security Model: Unified Exposition and Multics Interpretation', ESD-TR-75-306, The MITRE Corporation, Bedford, Massachusetts, June 1975.
- [111] AWACS JOVIAL Staff, 'JOVIAL User Manual for AWACS J3', SDC TM-WD-751/250/00J, July 12, 1976.
- [121] J. K. Millen, 'Security Verification in Practice', Communications of the ACM, May 1976, Vol. 19, Number 5, pp. 243-250.
- [131] C. Weissman, 'Security Controls in the ADEPT-50 Time-Sharing System', Fall Joint Computer Conference Proceedings, 1969, pp. 119-133.
- [141] D. E. Dennings, 'Secure Information Flow in Computer Systems', Ph.D. Thesis, Purdue Univ., Computer Sci. Dept., West Lafayette, Indiana, May 1975.
- [151] R. R. Linde, 'Operating System Penetration', Proceedings of the AFIPS 1975 National Computer Conference, Vol. 44, pp. 361-368.
- [161] M. Schaefer, B. D. Gold, R. R. Linde, et al, 'Program Confinement in KVM/370', Proceedings of the National ACM Conference, October, 1977, pp. 404-410.
- [171] B. D. Gold, R. R. Linde, et al., 'A Security Retrofit of VM/370', Proceedings of the 1979 National Computer Conference, Volume 48, AFIPS Press, New York, 1979, pp. 335-344.
- [181] R. M. Locasso, J. F. Scheid, et al., 'The Ina Jo Specification Language Reference Manual', TM-(L)-6021/001/00, System Development Corporation, Santa Monica, Calif., 27 June 1980.



MISSION of *Rome Air Development Center*

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C³I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.