15

Computer System Protection:

An Inventory

W. Douglas Climenson

TECHNICAL REPORT No. 15, MARCH 14, 1972

Computer System Protection: An Inventory

W. Douglas Climenson

14 March 1972

Computer System Protection: An Inventory

This paper is a survey of methods for protecting the integrity of a computer system - its data, processes, and equipment - and of the conditions which might exist in a computer system to motivate the use of such methods.

The survey is more descriptive than evaluative. A straightforward inventory approach is used. Hopefully the items it contains can be used as a checklist for those who are concerned about such matters - designers, managers, administrators.

The General Problem

Since this paper is largely a survey, the notion of protection is very broadly defined here. Emphasis is placed on software techniques and problems, but hardware and procedural aspects are not ignored. Similarly the system context is not restricted to time-sharing; other multi-user environments are also appropriate for consideration.

The general goal for system protection in the context of a computer utility was well stated by Schroeder and Saltzer (T13).

"... a computer utility serves multiple users who have different authorities. Such a diverse group will use the same system only if it is possible for them to achieve independence from one another. On the other hand, a great potential benefit of a computer utility is its ability to allow users to easily communicate, cooperate, and build upon one another's work. The role of protection in a computer utility is to control user interaction -- guaranteeing total user separation when desired, allowing unrestricted user cooperation when desired, and providing as many intermediate degrees of control as will be useful."

Most multi-user systems recognize both goals: user independence and user cooperation. However, Licklicker, a strong advocate of such systems, remarked in (S3) that evidence of productive user cooperation is much below his expectations. Perhaps this is why protection methods tend to emphasize the need for independence rather than controlled cooperation.

Some have been so bold as to suggest an ideal system of protection. Friedman (T5) does so for the important subproblem of file access control. He says the ideal system would have four features:

- The user should be assured of no unauthorized disclosure, but never denied rightful access.
- The system should be unbreakable even if the protection mechanism is in the public domain.
- The user should be able to build files and be able to

specify who can access the file and in what way.
- The protection mechanism should not inhibit normal file management functions.

Schroeder and Saltzer note that such statements of functional capability must be balanced against three other criteria of utility: economy, simplicity, and programming generality.

A variety of dichotomies have been defined in the literature to help us order our thinking about the protection problem. Ware (S8) and Petersen (S6) distinguish between accidents and deliberate acts; in (S9) Ware uses differences in motivation, authority, and history to differentiate the terms security and privacy. Watson (T14) separates intrinsic and technological methods. He also distinguishes the need to protect resources and the need to protect access paths to resources. A similar dichotomy is one that distinguishes protection methods which focus on the requesting process and the object being protected (see Graham - T6 and COSINE - T3).

The dichotomy chosen here is simple: problem and solution. And the approach is bottom-up rather than top-down. That is, an inventory of specific system conditions and specific protection mechanisms is given in a somewhat arbitrary sequence. Common threads are drawn among them with extensive use of cross references.

After some general observations are made, a list of protection mechanisms is given followed by a list of system conditions to which they might be applied. (The reversal of the usual order of problem-solution seems natural in this case.) Labelling conventions are as follows: Bibliographic references are prefixed with T or S, for technical or survey paper respectively. System conditions are prefixed by C. Protection methods are prefixed with A for automatic, N for non-automatic (i.e. manual or procedural) or P for policy mechanisms.

Some Observations

The most striking impression one gets in reviewing the literature on system protection is that although the literature is sparse, whenever a paper does appear, the author comes on like gangbusters: The enormity of the problem is underscored and the soapbox is relinquished only reluctantly at the end of the paper. In some cases the concern borders on paranoia. Ware (S8) and Petersen (S6) cite some bizarre examples. Petersen: " processor authentication may satisfy the user that he is not merely conversing with an infiltrator's console. Applied to certain messages from the processor to the user ..... these could be authenticated as coming from the processor and not a piggy-back penetrator." Or: "...the infiltrator can use his terminal to enter the system between communications from the legitimate user. In this situation the use of once-only passwords must be extended to each message generated by the user." Worst case situations are mentioned in the same breath with protection problems that are faced daily by all large systems. An all-or-nothing approach is simply not practical. Risks must be taken to keep computer systems productive and to keep costs reasonable. The grossly insecure system has the unreasonable cost of lost privacy or lost business. Absolute security has the unreasonable cost of inefficiency and great sums of money. Reasonable security means protecting against the most likely problems and accepting risks associated with the other.

There are several comments in the literature about the close relationship between protection objectives and system performance objectives. That is, protection is implicit in those design concepts which are concerned with keeping the system from failing, keeping users out of each others hair, and so on. Hoffman (S2) calls these "methods necessary for a properly operating time-sharing system". Fortunately there is much in common between these objectives. But this high degree of commonality has overshadowed the need for mechanisms which the user needs assuming he already is working with a smoothly operating system. He may want assurance, for example, that when he enters a command to delete a file, the file itself gets deleted - not just the access path to the file (i.e. the directory entry). The latter is sufficient for proper system operation, but perhaps not for the user's sense of privacy.

This is related to another concern, perhaps the most important one: how can a lay user, security auditor, or executive be assured that he is correct in putting his trust in the system? The conventional arguments do not apply in today's systems; he cannot see and touch the protection mechanism. It may become necessary someday, in some system, after some catastrophic incident, for some executive to insist that he indeed must be able to see, touch, and understand the protection mechanism.

Finally, there are two rather curious aspects of the protection problem:

(a) Although Peters (S5) and others believe that a well designed protection system should not have to be kept secret, what about a poorly designed one? Weaknesses need to be discussed, but there is the argument that weaknesses, when known, will be exploited. How far should the desire for detailed technical communication go in this case?

(b) The product of a computer operation is service, but protection has a negative connotation - protection <u>against</u> something. This is an alien concept in many ways to the professional.

Protection Mechanisms and Methods

Thirty-eight items are listed here, each a proposed or practiced method for dealing with some aspect of the system protection problem. They are by no means distinct; all are interrelated in some way. Some specific relationship are noted in the text. Others are implicit in that they have common references to the literature.

Only a gross categorization of the methods is given here. They are distinguished as Automatic, Nonautomatic, and Policy mechanisms. The greatest amount of attention is given to automatic methods. These are roughly ordered: hardware followed by software, with software methods grouped by data gathering, process control and access control.

The comments under each item are brief. Evaluative remarks are few and are included only to assist in cross referencing to other methods.

Automatic Mechanisms

A1. Supervisor/user modes. This hardware facility, now almost standard in medium and large systems, categorizes some instructions as "privileged" or " reserved". The instructions involved are those which are the most sensitive from the standpoint of system integrity. The instructions can only be executed when the system is in the supervisor or executive state. Obviously the instruction which sets the state should be a privileged instruction. Examples of other priveleged instructions in the IBM 360 are start I/O, halt I/O, test I/O, test channel, and diagnose (T8). This gross level of control of course does not protect the system from fatal errors caused by faulty programs which are executed in the supervisor mode. This and other aspects of system control protection are discussed in COSINE (T3), Graham (T6), Molhe (T11), Roder (T12), Watson (T14), Weissman (T15), Peters (S5), Ware (S9).

A2. Memory protection keys. An example here is the IBM 360 (T8): Each block of 2048 bytes of memory has a 5 bit key associated with it. The four most significant bits of the key are matched against a 4 bit key set in the Program Status Word or the Channel Address Word. A comparison is made before any instruction is executed which could change a memory block. Instructions which attempt to access data from a protected block are handled the same way if the low order bit of the storage key is set to indicate that fetch protection is in force. The memory block is protected either against reading or writing or against writing; it cannot be protected against reading only. Note that this scheme, as well as similar schemes used on other computers are provided at the physical, not the logical level of process control. See Molho (T11), Watson (T14), Hoffman (S2), Petersen (S6).

A3. Memory relocation and bounds registers. With this hardware facility in force, a process is constrained to operate within a lower bound (usually defined by the setting of a base register) and an upper bound (defined by the setting of a bounds register). Thus an arbitrary but contiguous area of memory is fenced off. Systems which have paging or segmentation hardware can provide such protection in a natural way: each element of the system's page or segment table can have part of the entry to indicate access control, and bounds are naturally associated with the physical blocks (pages) or logical blocks (segments). See COSINE (T3), Dennis (T4), Graham (T6), Molho (S11), Roder (T12), Watson (T14), Hoffman (S2), Peters (S5), Ware (S9).

A4. Redundant hardware for critical logic. Critical is used here from the protection standpoint. An example here would be redundant paths to test the bit which indicates if the system is in the supervisor or user mode (mechanism A1). This and several other proposals for redundacy directed specifically toward the IBM 360/50 are in Molho (T11). See also IEEE (T9).

A5. Use of ultra-reliable hardware. The space program and military requirements have motivated the development of components with extremely high reliability. Such components can reduce the need for mechanisms such as automatic fault detection (A6) or redundant hardware (A5). See IEEE (T9), Molho (T11), Petersen (S6).

A6. Automatic fault detection and diagnosis. A substantial amount of theory and follow-on development is underway which may have an impact of protection mechanisms, not only in reducing errors, but in those areas where they theory includes "faults" of the intentional variety. For example, tampering which may be undetected by conventional diagnostic means might be uncovered with more elaborate and complete fault detection schemes. See IEEE (T9).

A7. Terminal authentication. A unique identifier can be associated with a terminal. When interrogated by the attached processor (who are you?), the terminal could respond automatically. For example, Such a facility could be used to handle a requirement suggested by Friedman (T5): that a system response not be forwarded to a terminal until it is verified to be the right one to receive it. It is my understanding that all newly announced IBM terminals, for example, will have this facility. See Petersen (S6).

A8. Encryption of data. The term "privacy transformation" has been used to describe the process of encrypting data transmitted between terminal and computer. Hardware or software is used at each end of the line to perform a transformation so that data becomes unintelligible in the communication channel. A primary reason for the mechanism is the need for protection of authentication replies over accessible circuits. See Hoffman (S2), Peterson (S6), Ware (S9).

A9. User authentication by coded card, button combinations, etc. Included here are hardware methods for verifying the identity of a user at a terminal: cards with magnetic encoding or embossing, dials or buttons which must be operated in a specific sequence. These and other user authentication methods (A10, A13) require a software process as well as hardware. See Petersen (S6).

A10. User authentication by innate characteristic. The use of voice prints, finger prints, or physical measurements which provide unique identification have been suggested as a means of authentication.

A11. Physical or logical isolation of protection mechanisms. The intent here is emphasize the distinction between protection processes and others in the system, particularly that protection functions should not be performed by the operating system nor perhaps even by the central processor. Such isolation would facilitate special controls over protection mechanisms (inspection, physical locking, thorough debugging, etc.). Hardware techniques are discussed in Molho (T11) and software techniques in Friedman (T5). A specific example was the separate processor for memory management in the Berkeley Computer Company machine, Lampson (T10).

A12. Legality checks on I/O and other requests to monitor. Extensive use of this technique is made in the ADEPT system, Weissman (T15). Here the I/O system performs legality checks on a channel program before it is executed, including device address check, source of I/O request (must be from supervisor), and designated buffer areas. Hoffman (S2) notes that such procedures were recommended for hardware implementation in a report to the Rome Air Development Center by Bingham.

A13. User authentication by password. The most popular method of authentication. Peters (S5) suggests the use of one-time passwords. That is, a new password is used each time authentication is required. Hoffman (S2) suggests the use of passwords to gain access to various levels in the system (files, sensitive processes, etc.). See also COSINE (T3), Dennis (T4), Friedman (T5), Hansen (T7), Lampson (T10); and Parnas (S4).

A14. User authentication by user performed computation. Hoffman (S2) mentions a proposal by Earnest where the user is asked to perform a simple transformation (known only to him) on a random number provided by the computer (e.g, adding the left half and the right half of the random number plus the hour of the day). This is proposed to avoid sending intelligible authentication information over a communication line.

A15. <u>Automatic Call-back procedures</u>. In this method, requests for user access are handled as follows; the user identifies himself; if the user or a process or file he is requesting is tagged as sensitive, the user process is interrupted, the system logs off the user and calls him back in order to verify that the user and his terminal are as indicated in the original request; if so, the process is reactivated. See Babcock(T1) for an implementation example. See also Petersen (S6).

A16. <u>Audit Trails; Logs</u>. After passwords, this is the protection mechanism most often mentioned. It is the automatic recording of activities in the system of potential significance in monitoring system integrity. Quite often it is an extension of the accounting activity. A wide spectrum of meanings for "significant" are employed, ranging from simple recordings of log-on's and log-off's to recording of all I/O operations, calls to a specific procedure, abortive file accesses, usually long terminal sessions, etc. Reports produced from these data similarly vary in content and scope. Such data gathering is sometimes called "threat monitoring". See Friedman (T5), Weissman (T15), Hoffman (S2), Peters (S5), Petersen (S6).

A17. <u>Operator notification of significant events</u>. This is an active version of the logging activity. (A16). Specific events of an unusual nature could be brought to the attention of the operator, particularly in cases where real-time action of his part might be sensible. Examples might be all attempts (even valid ones) to copy very sensitive files, user CPU activity above a certain threshold, etc. See Petersen (S6).

A18. <u>Use of a Communication intermediary between processes</u>. The most obvious way to implement control through an intermediary is to insist that sensitive operations be performed in the supervisor state (see A1). Thus the supervisor, i.e. the operating system, must control such operations. Peters (S5) notes the general need for this autonomy, but the point is made quite directly in specific proposals in Hansen (T7) and in COSINE (T3). The interface between processes (which could otherwise be mutually interfering) is made explicit in Hansen's implementation by a few simple functions which must be used by communicating processes. The COSINE report takes this generalization one step farther in postulating a method of access control through an "access matrix". Specific implementations of inter-process control are described as particular ways of handling the information in such a matrix. The access matrix is a convenient way of expressing the elements of this protection problem. It has three components: objects (named things which are to be protected), domains (processes, procedures, processors, etc. which want access to these objects), and access functions (attributes associated with an (object, domain) pair such as read, write, owner). Note that a named domain, such as a particular process, can be an object. And (perhaps most important) the access matrix itself is subject to the same control. (Note: These concepts of central control can be thought of as generalizations of several mechanisms inventoried here, particularly A12 and A19-A24.)

A19. Interpretative handling of all user processes. If all user commands are in languages which are processed interpretatively, the operating system has an opportunity to exercise control of what the user is doing at every step. Not only can the legality of a command be checked, but also the possible bad effects of the command can be analyzed before it is executed since the context of the user command is available. Both Babcock (T1) and Graham (T6) note the protection inherent in interpretative systems.

A20. Hierarchical process control. Autonomous control at the top level of the system can be unwieldy. Given certain protocols, the operating system can delegate responsibility for inter-process communication to a process below the level of the operating system, and (recursively) down through several levels. For several specific instances of hierachical process control (where a process which creates another one is responsible for its actions) see especially Hansen (T7); also see Dennis (T2 and T4), Lampson (T10), Roder (T13). The trick is provide this control so that it is complete (i.e. so it covers all contingencies), yet does not inhibit legitimate inter-process communication.

A21. Pages or segments tagged with access information. Whereas method A2 dealt with the hardware facility for such tags, the concern here is with the use of such a facility. The access indicator, part of a segment descriptor (and associated ring structure) in MULTICS is an example (Graham, T6). The access indicator records the legitimate uses of the segment: read only, executable, etc. A ring number is recorded as well, providing rudimentary hierachical control over inter-process communication. See Watson (T14) and Hoffman (S2) for comments on this method. A hardware scheme for exercising access control via rings is given in Schroeder and Saltzer (T13).

A22. Naming control for data and processes. This method, described by Lampson (T10), can be characterized by the notion that in order to get access to an "object" in the system (a procedure, a file, a protection key), the process needing the object must know its name, and the point at which such names are assigned and interpreted becomes the point of access control. This concept is employed, but not described in these terms, by Friedman (T5), Hansen (T7), and earlier by Dennis and Van Horn (T4). The Lampson paper is difficult to follow. Parnas (S4) remarks "this paper can best be described as a poorly written paper with highly valuable contents."

A23. File access control by user "profile". This mechanism though related conceptually to A21, is considered to be distinct by those systems which treat files differently from virtual memory segments (the usual case). In this case a variety of access tags are employed. A comprehensive set, for example, is described by Weissman for ADEPT (T15). He categorizes access properties as authority (such as top secret, secret), category (components such as crypto, eyes only), and franchise (read only, write only). A rather complete treatment of this method is also given by Friedman (T5). See also Watson (T14), Peters (S5), and Petersen (S6).

A24. <u>Security tag applied to each file object</u>. The method just mentioned (A23) considers an entire file as the object to be given access control: the whole file or none of it is made available. Security tags could also be attached to smaller elements: records, subrecords, fields. Or instead of tagging specific instances of such elements, access to <u>types</u> of records, for example, could be conrolled (say payroll summary records or individual payroll records in the same file). The latter kind of control is suitable for user "profile" implementation. Hoffman (S2) mentions the possible need for access control over objects smaller than files and Friedman (T5) describes a possible implementation.

A25. <u>Software probes to monitor performance of protection mechanisms</u>. This facility is meant to exercise software, hardware, and procedural mechanisms for system protection in a systematic way, i.e. so that if the system is behaving properly, the results of the tests should be predictable. Molho (T11) recommends strongly that such monitoring be performed. He suggests random checking; Peters (S5) recommends it be done continuously.

A26. <u>Comparison of master and running copies of operating systems</u>. Both Molho (T11) and Petersen (S6) suggest that this be done to insure that the system running at the moment is that which the designers intended. No specific methods for doing this are suggested.

A27. <u>Erasure of sensitive portions of core and secondary storage after use</u>. Weissman (T15), Hoffman (S2), and Petersen (S6) mention the possible need for selective (or total) erasing of storage after use, or between uses, by sensitive processes. Weissman states that in ADEPT, for example, pages in core were zeroed when reallocated, and always swapped out even if no substantive changes were made.

<u>Nonautomatic Methods</u>.

N1. <u>Certification of adequacy of system protection</u>. Peters (S5) mentions that an operating system must be "approved by appropriate authority". Molho (T11) notes that the outlook for hardware certification is "bleak".

N2. <u>Control of physical access to system elements</u>. In one way or another, concern is expressed for every component in the system, from terminals to tapes to logic boards, by several authors: Molho (T11), Hoffman (S2), Peters (S5), Petersen (S6), and Ware (S9).

N3. <u>Control of access to software protection mechanisms</u>. Aside from the obvious need to protect encryption keys, password tables and the like, some authors emphasize the importance of protecting other elements of the software protection mechanism. Lampson's (T10) interests are in a general protection scheme which can, as part of its design, protect the protection mechanism itself. (See comments under A22.) Friedman recommends such protection through isolation of the mechanism, thus contolling access. (see comments under A11.)

N4. Switching of storage devices off/on line. Physical disconnection of devices, or removable file media, offers the surest method for protection of sensitive data while it is not in use. Petersen (S6) mentions this, but Peters (S5) notes that "a properly designed monitor for a multiprogramming system will provide sufficient logical separation of peripheral devices to make electrical separation unnecessary."

N5. Evaluation of system audit reports. The data gathering aspect of system monitoring is mentioned under A16; effective use of such data is another matter. Little mention is made in the literature of how to exploit the capability the system has for monitoring itself.

N6. Tape Degaussing. Magnetic tapes can be rendered unreadable when exposed to a strong magnetic field.

Policy Mechanisms

P1. Certification of reliability of employees in sensitive positions. Employees having access to protection mechanisms, critical hardware elements, sensitive files, and the operating system are mentioned by Peters (S5) and Petersen (S6) as personnel whose job reliability must be assured.

P2. Use of the "need-to-know" principle. A collarary: keep the number of key people to a minimum. See Petersen (S6).

P3. Application of a code of ethics. Sen. Ervin is quoted by Hoffman (S2): "While there is still time to cope with the problem, they [the computer industry and its people] must give thought to the contents of professional ethics codes for the computer industry and for those who arrange and operate the computer's processes." See also Harrison (S1), Ware (S9).

P4. Isolation of sensitive applications on separate machines (or on shared machines at separate times). In the confusion of analyzing alternatives, this one obvicusly should not be ignored. Protection problems can sometimes be solved best by avoiding them.

P5. Minimization of number of file copies. Mentioned by Petersen (S6).

System Conditions

An attempt is made here to list rather exhaustively the conditions or situations in a computer system that have been pointed out as protection problems somewhere in the literature. The more neutral term "condition" is used here because what is a potential problem in one environment may not exist or may be acceptable or tolerable in another. For the same reason there is no attempt to evaluate the relative seriousness of the listed conditions.

The listed references under each item contain a description of the condition and possibly comment on it. The references do not necessarily suggest use of the protection methods and mechanisms which follow. References to literature on these methods are mentioned in the preceding section. The list of protection methods is my attempt to tie these two sections together. In some cases the methods are complimentary; in other cases, the use of one of them might eliminate the need for others. Only the most directly applicable methods are mentioned. (Suggestions such as the use of the need-to-know principle are so fundamental to the concept of protection that they would have to be included under almost every item.)

C1. Program errors or "infiltrator's" actions could destroy or modify another user's data or programs.

References: Dennis (T2 and T4), COSINE (T3), Graham (T6), Hansen (T7), Schroeder and Saltzer (T13), Watson (T14), Hoffman (S2), Peters (S5), Petersen (S6), Ware (S9)

Methods: A1. Supervisor/user modes
A2. Memory protection keys
A3. Memory bounds registers
A12. I/O legality checks
A18. Inter-process communication control
A19. Interpretative processing

A20. Hierarchical process control
A21. Access tags on segments
A22. Object naming control
A26. Checks on O.S. integrity
N3. Control over access to software protection mechanism.

C2. Debugging assistance routine is required to monitor and control execution of a faulty program without allowing errors to alter performance of debugging routine.

References: COSINE (T3), Lampson (T10), Watson (T14), Weissman (T15), Peters (S5)

Methods: A2. Memory protection keys
A3. Memory bounds registers
A12. I/O legality checks
A18. Inter-process communication control
A19. Interpretative processing
A20. Hierarchical process control
A21. Access tags on segments
A22. Object naming control

C3. A procedure (B) called by procedure (A) might never return control to procedure (A).

References: Dennis (T2 and T4), COSINE (T3), Lampson (T10), Schroeder and Saltzer (T13), Watson (T14).

Methods: A18. Inter-process communication control
A19. Interpretative processing
A20. Hierarchical process control
A21. Access tags on segments

C4. One of a user's programs could destroy one of his own programs or files.

References: Graham (T6), Lampson (T10), Watson (T14), Ware (S8).

Methods: A2. Memory protection keys
A3. Access tags on segments
A23. User profile file access control
A24. Access tags on file objects

C5. A tape or disc pack could be mounted on the wrong device.

Reference: Hoffman (S2).

Methods: A12. I/O legality checks
A22. Object naming control
A23. User profile file access control
A23. Access tags on file objects

C6. Messages from the central processor to a terminal could be misrouted.

References: Friedman (T5), Hansen (T7), Weissman (T15), Petersen (S6).

Methods:

A7.	Terminal authentication
A8.	Data encryption
A9-A10.	User authentication
A15.	Call back procedures
A18.	Info-process communication control
N4.	Storage device switching
P5.	Applications on separate machines

C7. Computer logic might be wired erroneously when manufactured.

Reference: Molho (T11).

Methods:

A4.	Redundant hardware
A6.	Automatic fault detection
A11.	Isolation of protection mechanism
N1.	System certification

C8. A fault, violation, or other unusual situation might be detected, but its cause may not be evident, making corrective action difficult.

References: Graham (T6), Molho (T11).

Methods:

A6.	Automatic fault detection
A11.	Isolation of protection mechanism
A12.	I/O legality checks
A16.	Audit trails
A17.	Operator notification
A25.	Protection performance monitoring
A26.	Checks on O.S. integrity
N5.	Audit report evaluation

C9. Hardware mechanisms for system protection might fail.

References: Molho (T11), Ware (S8).

Methods:

A4.	Redundant hardware
A5.	Ultra-reliable hardware
A6.	Automatic fault detection
A25.	Protection performance monitoring
N1.	System certification

C10. The system might fail or its performance might be seriously degraded due to program error or intentional malicious action.

References: COSINE (T3), Graham (T6), Molho (T11), Peters (S5), Petersen (S6), Ware (S8).

Methods:

A16.	Audit trails
A17.	Operator notification
A26.	Checks on O.S. integrity

C11. Software of hardware failure might disengage or damage protection mechanisms.

References: Molho (T11), Ware (S8).

Methods: A4. Redundant hardware
A5. Ultra-reliable hardware
A6. Automatic fault detection
A25. Protection performance monitoring
N1. System certification

C12. One user might accidently or intentionally gain access to anothers private data.

References: COSINE (T3), Friedman (T5), Lampson (T10), Molho (T11), Schroeder and Saltzer (T13), Watson (T14), Weissman (T15), Hoffman (S2), Peters (S5), Petersen (S6), Ware (S8 and S9).

Methods: A2. Memory protection keys
A3. Memory bounds registers
A4. Data encryption
A9,10,13,14. User authentication
A12. I/O legality checks
A15. Call-back procedures
A20. Hierarchical process control
A21. Access tags on segments
A22. Object naming control
A23. User profile file access control
A24. Access tags on file objects
A26. Checks on O.S. integrity
A27. Storage erasure
P4. Applications on separate machines.

C13. A user with access rights to another's data might grant access to this data to a third party without the original owner's permission.

References: Dennis (T2 and T4), COSINE (T3), Graham (T6), Lampson (T10), Schroeder (T13), Watson (T14), Hoffman (S2).

Methods: A16. Audit Trails
A18. Inter-process communication control
A20. Hierarchical process control
A21. Access tags on segments
A22. Object naming control
A24. Access tags on file objects
N5. Audit report evaluation
P1. Employee reliability
P3. Code of ethics
P3. Need-to-know principle

C14. It might be advisable to give a user access to part of a file rather than the entire file.

References: Friedman (T5), Watson (T14), Hoffman (S2).

Methods: A27. Object naming control
A23. User profile file access control
A24. Access tags on file objects

C15. Residual data left in memory or on secondary storage by one user could be accessed by another.

References: Hoffman (S2), Petersen (S6)

Methods: A18. Inter-process communication control
A22. Object naming control
A27. Storage erasure

C16. An unauthorized terminal might get attached to the system.

References: Hoffman (C2), Petersen (S6)

Methods: A7. Terminal authenication
A8. Data encryption
A9,10,13,14. User authentication
A15. Call-back procedures
A16. Audit trails
N2. Physical access control

C17. One user might "masquerade" as another.

References: Friedman (T5), Peters (S5), Petersen (S6)

Methods: A9,10,13,14. User authentication
A15. Call-back procedures

C18. A user could "invade" the operating system and make changes to gain control of the system or to gain access to information.

References: Molho (T11), Petersen (S6), Ware (S8)

Methods: A1. Supervisor/user modes
A2. Memory protection keys
A3. Memory bounds registers
A11. Isolation of protection mechanisms
A16. Audit trails
A18. Inter-process communication control
A19. Interpretative processing
A25. Checks on O.S. integrity
N1. System certification
N3. Control over access to software protection
P1. Employee reliability

C19. Eavesdropper might try to get access to information by tapping communication lines or monitoring radiation from the computer system.

References: Hoffman (S2), Petersen (S6), Ware (S8 and S9)

Methods: A8. Data encryption
N2. Physical access control

C20. An "infiltrator" could piggyback on communication lines, intercepting messages and responding as though he were the addressee.

References: Molho (T11), Hoffman (S2), Petersen (S6)

Methods:		
	A7.	Terminal authentication
	A8.	Data encryption
	A9,10,12,13	User authentication
	A15.	Call-back procedures
	A25.	Protection performance monitoring
	N2.	Physical access control

C21. A user could exercise control over the system by getting access to critical instructions.

References: COSINE (T3), Molho (T11), Watson (T14), Petersen (S6), Ware (S8)

Methods:		
	A1.	Supervisor/user modes
	A3.	Memory bounds register
	A25.	Protection performance monitoring
	P4.	Applications on separate machines

C22. Hardware devices might be intentionally rewired to bypass protection mechanisms.

References: Molho (T11), Petersen (S6), Ware (S8)

Methods:		
	A6.	Automatic fault protection
	A25.	Protection performance monitoring
	N1.	System certification
	N2.	Physical access control
	P1.	Employee reliability

C23. Sensitive tapes or discs could be removed from the computer center.

References: Hoffman (S2), Petersen (S6)

Methods:		
	N2.	Physical access control
	P5.	Few copies of files

Bibliography

Technical Publications

T1. Babcock, J.D. A brief description of privacy measures in the system. Proc. SJCC 1967, 301-302

T2. Dennis, Jack B. Segmentation and the design of multiprogrammed Computer Systems. JACM Oct. 1965, 589-602

T3. COSINE Task Force VIII Report, Module 6-protection. June 1971

T4. Dennis, Jack B. and Van Horn, Earl C. Programming Semantics for Multiprogrammed Computations. CACM March 1966, 143-155

T5. Friedman, T. D. The authorization problem in shared files. IBM Sys. J. Vol.9, No.4, 1970, 258-280

T6. Graham, R.M. Protection in an information processing utility CACM May 1968, 365-370

T7. Hansen, Per B. The nucleus of a multiprogramming system. CACM April 1970, 238-241, 250

T8. IBM system 1360 Principles of operation, IBM Form A22-6821-5, Jan. 1967, 17, 68-70, 74

T9. IEEE Special issue on fault-tolerant computing, IEEE TRANS E C Nov. 1971

T10. Lampson, B.W. Dynamic Protection Structures, Proc. FJCC 1969, 27-38

T11. Molho, L. Hardware aspects of secure computing. Proc. SJCC 1970, 135-171

T12. Roder, J. and Rosene, A.F. Memory protection in multiprocessing systems. IEEE Trans. EC June 1967, 320-326

T13. Schroeder, Michael D. and Saltzer, Jerome H. A hardware architecture for implementing protection rings. Proc. Third Symposium on Operating System Principles Oct. 1971,

T14. Watson, Richard W. Timesharing system design concepts, sections 4.1, 5.4, and 6.16, McGraw-Hill, 1970

T15. Weissman, C. Security controls in the ADEPT-50 time sharing system. Proc. FJCC 1968, 119-133

Surveys

S1. Harrison, Annette. The problem of privacy in the computer age: an annotated bibliography. RAND Corp. (RM-5495-PR/RC) Dec. 1967

S2. Hoffman. L. J. Computers and privacy: a survey. Computing Surveys Vol. 1, no.2, 1969, 85-103

S3. Licklider, J.C.R.. Man-Computer Communication, in Annual Review of Information Science and Technology Vol. 3, 1968

S4. Parnas, D.L. Review of "Dynamic protection structures" Computing Reviews (no. 20511), Jan. 1971, 29-30

S5. Peters, Bernard. Security considerations in a multi-programmed computer system. Proc. SJCC 1967, 283-286

S6. Petersen, H.E. and Turn, R. System implications of information privacy. Proc. SJCC 1967, 291-300

S7. Titus, J.P. Washington Commentary-Security and Privacy. CACM June 1967, 379-380

S8. Ware, Willis H. Security and privacy in computer systems, Proc. SJCC 1967, 279-282

S9. Ware, Willis H. Security and privacy: similarities and differences. Proc. SJCC 1967, 287-290