COMPUTER ASSISTED SECURITY SYSTEM DESIGN

by

Don Clements and Lance J. Hoffman

Memorandum No. ERL-M468

November 1974

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

COMPUTER ASSISTED SECURITY SYSTEM DESIGN†

Don Clements and Lance J. Hoffman
Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California, Berkeley, California 94720

ABSTRACT

A computer software package has been implemented which partially automates the selection of security techniques applicable to a particular data processing system design. The program is a table-driven, interactive, information retrieval system which takes "objects" and "threats" as input and produces suggested "countermeasures" as output. The system is a small prototype; possible extensions are discussed at the end of the paper.

---

† Research sponsored by National Science Foundation Grant GJ-36475.

1. OVERVIEW

An automated management tool has been implemented to aid in the selection of security features which would enhance the security of an existing data processing facility or define security requirements in a system to be designed. An interactive decision making program was developed which employs information retrieval techniques to produce listings of "desireable" security measures to combat known or suspected "threats". This paper describes the algorithm used and illustrates (with sample output) the nature of its operation. General qualitative evaluations and proposed extensions to the system are also discussed. The actual programming was done in Stanford Basic using the IBM System 370 facility at Stanford University.

2. BACKGROUND

The selection of security measures constitutes one component of the security system synthesis process. The entire synthesis procedure is defined in "Privacy and Security in Databank Systems" (Turn c.1973?). In this essay, Turn develops a definition of "Data Security Engineering" as a "...methodology, and a set of techniques, being developed to provide a framework for synthesizing security systems, comparing alternative designs, and providing tradeoff analyses." He views the security environment as a combination of a protected domain, a user domain, a threat domain, the security system, and a set of external constraints (see Figure 1).

The protected domain contains all of the entities (objects) which require protection. This domain may be subdivided into several levels

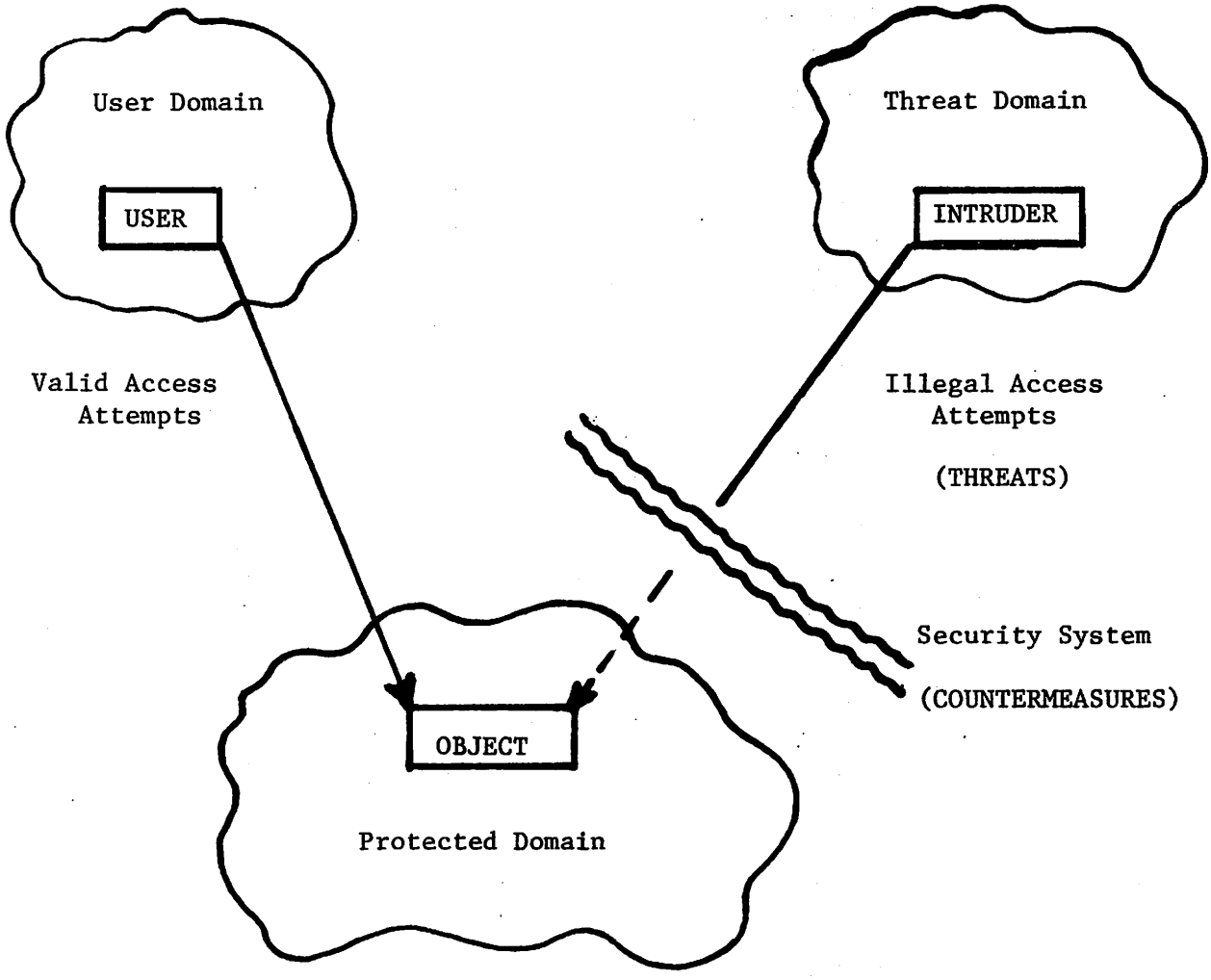

Figure 1 - The Security Environment

of protection according to the value placed upon the individual objects. The user domain is made up of all personnel authorized to enter the protected domain. All subversion techniques and the set of all personnel not authorized access to the protected domain form the threat domain. The security system employs hardware, software, physical procedures, and administrative controls to erect barriers between the threat domain and the protected domain. These barriers must not greatly impede access from the user domain. External constraints include those policies, economic considerations, technological limitations, etc. which limit the ideal security system design. Constraints of this nature are, of course, common to all engineering design problems.

An analysis of the protected domain and the subsequent identification of the objects of interest constitutes the first phase of security system design. All valid access paths must be defined; all vulnerabilities must be identified and carefully described. Techniques of Network Theory have been proposed (Turn c. 1973?) as a means of identifying these paths. Levels of required protection are determined by a value analysis of each of the protected elements. Next a risk analysis is required to determine the subset of the threat domain promising the greatest potential gain from the intruder's point of view. Risk probabilities are assigned so that the threat domain is also subdivided into levels of importance.

The synthesis of security barriers constitutes the third step of the design procedure. A set of security countermeasures is selected from the inventory of all known security techniques. Criteria include: effectiveness under the circumstances defined by the present environment, cost, degree of discrimination, and reliability. Care must be exercised

that all interfaces with the computing system are well understood and that no "holes" in the security barrier network develop as a result of an incomplete or erroneous evaluation.

The proposed system must be analysed for completeness and effectiveness, either by modeling/simulation, detailed analysis, or full scale penetration exercises. Several iterations of the above design process may be required to insure the specified security goals are attained. It is highly desireable that auditing and certification finalize the synthesis process.

3. SCOPE OF THE PROJECT

This work is concerned with the selection of appropriate security countermeasures: the third step in the design process discussed above. The program to be described accepts a pairing of elements from the object and the threat domains as inputs, producing a single "suggested" security countermeasures as output. The program iterates over all object-threat combinations of interest to the user and produces a set of countermeasures which then define the security system. If implemented, these countermeasure will form the security barriers for protection of the set of objects which make up the protected domain. The risk analysis which yields the most likely object-threat combinations is the responsibility of the Security Officer, system designer, or manager using the program. The risk analysis process is somewhat facilitated by the program flow; the user is forced to list his objects in order of decreasing value and the associated threats in order of decreasing probability.

The final countermeasure set produced is by no means to be assumed

optimal. The target is a set which is "adequate" in that all objects are protected, and "minimal" in that any countermeasure which protects more than one object (or defends against more than one threat upon a single object) is given preference. Since it is recognized that a weak countermeasure covering more than one threat may be a poor choice in a very high security design, the "weighting" feature (see Section 5) may be used to offset this "minimal set" characteristic where desired. It is the user's responsibility to ascertain the cost-effectiveness of the security barrier set and apply any external constraints which may preclude the adoption of one or more of the given countermeasures.

This work should not be viewed as a rigorous mathematical model of an ideal security system. The intent is rather the production of a useful design aid in the selection of the pertinent security measures. The program is very general; it will aid in the design of a wide variety of security systems at diverse installations. Many of the selection criteria in the program logic are somewhat coarse since the program is still undergoing experimentation and "tuning".

4. SAMPLE DESIGN CONVERSATION

The terminal conversation for a complete design run is shown in Figure 2. User inputs may be distinguished by two characteristics: inputs always follow a prompt ("?") and are always in lower case type. Note the use of plain text for input keywords and the high degree of redundancy in the repeated generation of partial lists. The summary phase shows the effect of multiple use of a countermeasure and the special result ("none available") when all design alternatives are rejected by

```
HERE IS A LIST OF SECURITY OBJECTS:
TYPE "YES" FOR THOSE TO BE CONSIDERED.

CORE MEMORY ?  yes
FILES ?  yes
OPERATING SYSTEM ?  no
REMOTE TERMINALS ?  yes

HERE ARE THE REMAINING OBJECTS:
   1. - CORE MEMORY
   2. - FILES
   3. - REMOTE TERMINALS

CHOOSE (BY NUMBER) THE MOST VALUABLE: 3

HERE IS A LIST OF THREATS AGAINST REMOTE TERMINALS:
TYPE "YES" FOR THOSE TO BE CONSIDERED.

BETWEEN LINES ENTRY ?  yes
BOGUS TERMINAL ?  no
INTERCEPTING SIGNOFF ?  yes
MASQUERADING ?  yes
WIRE TAPPING ?  no

HERE ARE THE REMAINING THREATS:
   1. - BETWEEN LINES ENTRY
   2. - INTERCEPTING SIGNOFF
   3. - MASQUERADING

CHOOSE (BY NUMBER) THE MOST PROBABLE: 3

HERE ARE THE REMAINING THREATS:
   1. - BETWEEN LINES ENTRY
   2. - INTERCEPTING SIGNOFF

CHOOSE (BY NUMBER) THE MOST PROBABLE: 2

HERE ARE THE REMAINING OBJECTS:
   1. - CORE MEMORY
   2. - FILES

CHOOSE (BY NUMBER) THE MOST VALUABLE: 1

HERE IS A LIST OF THREATS AGAINST CORE MEMORY:
TYPE "YES" FOR THOSE TO BE CONSIDERED.

ILLEGAL ACCESS ?  no
READING RESIDUE ?  yes

HERE IS A LIST OF THREATS AGAINST FILES:
TYPE "YES" FOR THOSE TO BE CONSIDERED.

ALTERATION ?  yes
BROWSING ?  yes
DESTRUCTION ?  yes

HERE ARE THE REMAINING THREATS:
   1. - ALTERATION
   2. - BROWSING
   3. - DESTRUCTION

CHOOSE (BY NUMBER) THE MOST PROBABLE: 2
```

Figure 2 - Sample System Design

```
HERE ARE THE REMAINING THREATS:
   1. - ALTERATION
   2. - DESTRUCTION

CHOOSE (BY NUMBER) THE MOST PROBABLE: 2
```

Figure 2 - (cont.)

```
HERE ARE THE SUGGESTED COUNTER-MEASURES:
TYPE "NO" FOR THOSE WHICH ARE UNAVAILABLE.

PASSWORD ?  yes
SYSTEM SIGNOFF CONFIRMATION ?  no
ENCRYPT NULLS ?  yes
OVERWRITE CORE ?  yes
BACKUP COPY ?  yes


HERE IS A SUMMARY IN PRIORITY ORDER:

OBJECT                  THREAT                  COUNTER-MEASURE

REMOTE TERMINALS        MASQUERADING            PASSWORD
                        INTERCEPTING SIGNOFF    NONE AVAILABLE
                        BETWEEN LINES ENTRY     ENCRYPT NULLS

CORE MEMORY             READING RESIDUE         OVERWRITE CORE

FILES                   BROWSING                PASSWORD
                        DESTRUCTION             BACKUP COPY
                        ALTERATION              PASSWORD

ALTERNATIVE DESIGN? yes

HERE ARE THE SUGGESTED COUNTER-MEASURES:
TYPE "NO" FOR THOSE WHICH ARE UNAVAILABLE.

PASSWORD ?  no
AUTHORIZATION ALGORITHM ?  yes
SYSTEM SIGNOFF CONFIRMATION ?  yes
ENCRYPT NULLS ?  no
OVERWRITE CORE ?  yes
ACCESS LIST ?  yes
BACKUP COPY ?  yes
CHECKSUM ?  yes


HERE IS A SUMMARY IN PRIORITY ORDER:

OBJECT                  THREAT                  COUNTER-MEASURE

REMOTE TERMINALS        MASQUERADING            AUTHORIZATION ALGORITHM
                        INTERCEPTING SIGNOFF    SYSTEM SIGNOFF CONFIRMATION
                        BETWEEN LINES ENTRY     NONE AVAILABLE

CORE MEMORY             READING RESIDUE         OVERWRITE CORE

FILES                   BROWSING                ACCESS LIST
                        DESTRUCTION             BACKUP COPY
                        ALTERATION              CHECKSUM
```

the user. This sample run took 5.5 minutes wall-clock time and 1.39 CPU seconds on the Stanford 370.

5. DESCRIPTION OF THE PROGRAM

Table 1 is an outline of the program logic. Indented entries represent user inputs via the terminal; the other steps are responses by the program logic to these inputs. Wherever possible, inputs are formatted as plain text "YES" or "NO" responses with the decision keying done on the first letter of the typed input (e.g. "y" if a "YES" input is expected). In this way the effects of typing errors are minimized. An exception to this policy is made in the interest of brevity during the ranking of Objects (step 5) and Threats (step 10). Here the listing is numbered and the user types the number of the next "most valuable Object" or "most probable Threat". As each Object or Threat is processed, it is eliminated from the current list (see Figure 2). The modified list is renumbered and relisted at the terminal for the next selection. Thus only a minimal amount of information must be remembered by the user during execution.

The names of all Objects, Threats, and Countermeasures which are recognized by the program together with the pointers which relate the various combinations of these three elements are retained permanently on secondary storage in a file system. The exact nature of this file structure is examined in detail in the Appendix. These files are random access in that individual records are directly addressable. During the execution of step 11 (Table 1), the current Object and Threat names are entered into a table in primary (core) storage with the pointers (record

OUTLINE OF PROGRAM LOGIC

PROGRAM ACTION/OUTPUT

USER INPUT

0. Initialization
1. List an object name
2. Accept or reject
3. Repeat 1 & 2 until Object file empty
4. List accepted subset of objects
5. Select "Most Valuable"
6. Search Threat file
7. List a threat name
8. Accept or reject
9. Repeat 7 & 8 until all associated threats listed
10. Select "Most Probable/Costly"
11. Save links to Countermeasure file for this threat
12. Eliminate threat & iterate 10-12 until no more threats
13. Eliminate object & iterate 5-13 until no more objects
14. Read in initial weights of saved countermeausres (CM's)
15. Increase weight of each CM saved more than once
16. List highest weight CM for this threat
17. Accept or reject
18. If rejected, zero weight and return to 16
19. If accepted, save CM name for summary
20. Eliminate threat and iterate 16-20 until no more threats
21. Print Object, Threat, Countermeasure triplets in a priority order summary
22. If Alternative Design requested, iterate 14-21
23. Otherwise, terminate

Table 1. Outline of the Algorithm

numbers) of the possible Countermeasures. In addition to providing fast retrieval during the summary printing of step 21, this tabular scheme automatically provides a "priority ordering" according to the user's terminal inputs.

The selection of one countermeasure from a set of alternatives involves the use of a weight value. This initial weight of each countermeasure recognized by the system is stored in a file which is read during step 14. Currently, all initial weights are given the value "1", the choice being somewhat arbitrary. The use of these initial weight values to "preset" the system is a topic for further experimentation.

There may be occasions when one countermeasure will be useful against more than one threat. The program logic causes the initial weight value from the Countermeasure Weights file to be added into a table which is initialized to zeros. Each time a countermeasure is referenced by a link value in the summary table this weight is added in to the accumulated value of the appropriate weight table entry. Thus, after complete scan of the summary table, those countermeasures which serve more than one threat will have attained higher weights. The program then merely selects the countermeasure with the highest weight whenever a choice is required.

It is possible (indeed likely) that one or more of the various countermeasures proposed may be unavailable, too costly to implement, or in some other way unacceptable to the user. This information is declared during step 17 as each proposed countermeasure is listed. When a countermeasure is rejected its weight is reset to zero so that the next search will ignore (recall all weights are at least "1" after

reading the weight file). If a countermeasure for more than one threat is ever rejected by the user, the program does not consider using it against any other threats in the set of interest. Finally, if all alternatives in the countermeasure list for a threat are rejected the message "none available" is saved for the summary and processing of that threat is terminated.

After the set of "best" countermeasures is thus determined, a summary is printed showing each object-threat pair in priority order. This ordering was developed during the input phase when the objects and threats of interest were declared by the user. A single countermeasure is listed following each object-threat pair. This summary provides the set of security features to be implemented and the order in which they should be developed to insure that objects of greatest value are protected first.

At the end of the summary phase an alternative design may be requested (see Figure 2). Countermeasure weights are reinitialized and a new set of "acceptable" countermeasures determined. Any number of alternative designs including the summary information may be produced without reentereing all of the object and threat information. When all alternatives of interest are produced, the program terminates.

A sample of the output from the summary portion of the program is shown in Figure 3. The two sets of results show the effect of a rejection of a countermeasure ("password") upon the summary when more than one threat is involved. A complete system design can be found in Section 4.

In support of the file system and main design portion of the program,

```
HERE ARE THE SUGGESTED COUNTER-MEASURES:
TYPE "NO" FOR THOSE WHICH ARE UNAVAILABLE.

PASSWORD ?  yes
ENCRYPT DATA ?  yes
BACKUP COPY ?  yes


HERE IS A SUMMARY IN PRIORITY ORDER:

OBJECT                  THREAT                  COUNTER-MEASURE

REMOTE TERMINALS        MASQUERADING            PASSWORD
                        WIRE TAPPING            ENCRYPT DATA

FILES                   BROWSING                PASSWORD
                        DESTRUCTION             BACKUP COPY
```

Figure 3a - Summary Output (note multiple use of Password)

```
HERE ARE THE SUGGESTED COUNTER-MEASURES:
TYPE "NO" FOR THOSE WHICH ARE UNAVAILABLE.

PASSWORD ?  no
AUTHORIZATION ALGORITHM ?  yes
ENCRYPT DATA ?  yes
ACCESS LIST ?  yes
BACKUP COPY ?  yes


HERE IS A SUMMARY IN PRIORITY ORDER:

OBJECT                  THREAT                  COUNTER-MEASURE

REMOTE TERMINALS        MASQUERADING            AUTHORIZATION ALGORITHM
                        WIRE TAPPING            ENCRYPT DATA

FILES                   BROWSING                ACCESS LIST
                        DESTRUCTION             BACKUP COPY
```

Figure 3b - Summary Output (Password countermeasure rejected)

file management routines are included for listing of all of the file contents and for the modification of files by addition or deletion of Objects, Threats, and Countermeasures. The modification routines are designed to ease the process of linking related Object - Threat - Countermeasure triplets by shifting file entries and pointer values as required to maintain alphabetical order in the name files and correct record numbers for linkage in the link files. Appendix A contains sample file listings and a description of the logic of the file management routines.

6. EVALUATION

This systhesis algorithm possesses two features which enhance its use in real world operations: it is extensible and interactive. The file structure is of sufficient generality to make practical the inclusion of a large number of objects, threats, and countermeasures. If information in these files is made sufficiently complete, it should be feasible to design or modify a realistic large scale computing facility. Most of the current work has been devoted to the development of this prototype and experimentation with the weighting feature; complete development of the file information sets is the next logical step.

The interactive nature of the program provides several advantages. The user is relieved of the exhaustive task of supplying all possible installation characteristics in advance. The probability of erroneous or unneeded inputs is also minimized. A batch type of program would also lock the user into one design per run, an undesirable feature in a design process which is usually iterative. As mentioned earlier, the

interactive property of the program flow also aids the user in the risk analysis and priority identification portions of the design process.

On the negative side, the logic employed in selecting a single "best" countermeasure from each set is somewhat simplistic. Recall that each countermeasure weight is initialized to 1. In the absence of an application of one countermeasure to more than one threat, this weight will remain unchanged throughout the search phase. The selection of the "best" countermeasure for a given object-threat pair is made by choosing that element of the countermeasure set with the largest weight value. In the event of a tie, the first entry encountered in the set is used. Thus the algorithm is very sensitive to the ordering of the countermeasures in each set. Some degree of improvement would result if individual initial weights were different. These initial weights could be governed by the amount of information known about the utility of a particular security measure at a given installation. Since this information is so dependent upon the individual installation characteristics, the best approach would involve the user. This proposal is further examined in Section 7.

Since multiple use of a countermeasure depends upon the establishment of appropriate linkages at file generation time, it is important that the name of a countermeasure be distinct whenever this multiple use is not appropriate. Referring to the sample summaries in Figure 3, note that "password" appears as a protective measure for both "remote terminals" and "files". This does <u>not</u> mean that one password is always sufficient to protect both objects. Flexibility may dictate a user password to the system and a separate password for the file structure. Discrimination

could be added by using the names "user password" and "file password" in the countermeasure file. The algorithm would no longer bias its selection of the "password" over alternative countermeasures since the linkages would now point to separate entries in the countermeasure file. This tradeoff of "specificity" for multiple application of a security feature must be carefully considered when the countermeasure file for this program is designed or modified.

7. PROPOSED EXTENSIONS

In the previous section, the weighting problem was discussed. One useful modification of the present program would be a "weight preset" option available to the user at the beginning of the program execution. User information about availability and/or suitability of various security features at his installation could be incorporated into the weighting file. This feature could also be offered during the summary phase to generate multiple designs showing user weightings. In this way cost tradeoffs would be facilitated.

There is some system learning during the summary phase with respect to the utility of various countermeasures. This "memory" is lost after the run since only the weighting tables are modified; the intial weights in the countermeasure weight files remain unchanged for the next run. A possible improvement here would involve additional logic during the search phase to increment entries in a "history file" which would preserve information about likely object-threat and threat-countermeasure pairings over a large number of runs. This information would be useful in any future modification of the linkage files to enhance efficiency by presenting

"probable" objects, threats, and countermeasures to the user first.

Finally, the exact definition of each specific object, threat, and countermeasure should be avialable to the designer. A glossary should be written in parallel with the expansion of the system files. This would minimize ambiguity and eliminate any need for the user to recognize "security jargon".

8. CONCLUSION

A table-driven program which automates the selection phase of security system design has been presented. Its position in the overall field of Data Security Engineering was discussed. The algorithm which implements this selection process was described in general terms. The program is straightforward and relatively easy to implement, though somewhat lacking in discrimination. Several extensions to the present system have been outlined. A sample design run was presented in Section 4. The contents of the files used in this prototype program are listed in the Appendix. Given sufficiently complete files, it is believed the program will function well in the synthesis of a large scale security system.

ANNOTATED BIBLIOGRAPHY

(Browne 1972) Browne, P.S., "Taxonomy of Security and Integrity", unpublished M.S. draft (in Hoffman, 1973).

Very complete treatment of the areas of consideration in the design of a total security system; administrative, physical, hardware, software areas outlined. Audit, testing, and certification procedures included.

(Canning 1970) Canning, R.G., "Data Security in the CDB", EDP Analyzer, vol. 8, no. 5, May 1970.

Application of security techniques to protection of the Corporate Data Base; types of threats (active, passive, accidental) illustrated in business data environment; good general overview of access management; suggested methods for terminal security improvement given.

(Hoffman 1973) Hoffman, L.J., Security and Privacy in Computer Systems, Melville Publishing Co., Los Angeles, CA., 1973.

Anthology of latest papers in the security and privacy field; sections on civil liberties, hardware and software secruity techniques, formal models, data banks; IBM's RSS used as example of existing systems.

(Hoffman 1974) Hoffman, L.J., "Computer Security Engineering", lecture notes for CS 244, Dept. of EECS, CS Div., UC Berkeley, Winter 1974.

Large variety of security techniques catalogued

with detailed examples of proposed and existing implementations; hardware, software, file management, O/S protection, physical, and administrative techniques; short section on privacy.

(IBM 1970) IBM, "The Considerations of Data Security in a Computer Environment, Form G520-2169-0, 1970.

Detailed examination of security techniques for management, system design, and operations personnel; use of authorization techniques, file protection, audit logs; details on development of security operating procedures.

(IBM 1972) IBM, "The Considerations of Physical Security in a Computer Environment", Form G520-2700-0, 1972.

Detailed treatment of protection against physical threats to the data system; fire and disaster procedures; backup facility design; terminal and network security; computer and I/O Room administration; system audit procedures; IBM's Advanced Administrative System used as a case study.

(Popek 1974) Popek, G.J., "Protection Structures", Computers, June 1974, pp. 22-33.

Discussion of the best known software mechanisms for protection and their flaws; capabilities, access lists, and domains; description of the kernel approach

to protection design; possibilities for program verification in proving systems; several formal models mentioned.

(Turn c.1973?) Turn, R., "Privacy and Security in Databank Systems", unpublished paper, Rand Corporation, Santa Monica, CA., undated.

General delineation of the security environment; conceptualization of the principles of Data Security Engineering; Discussion of Risk Analysis and Cost-Effectiveness measurements; advocacy of graph-theoretic approach to Security System Synthesis.

APPENDIX

THE FILE STRUCTURE

Contents of the Object, Threat, and Countermeasure name and link files are shown on the next two pages. These listings were produced by the file management routines. The file structure is detailed in Section A1 which follows the listings. Finally, a short description of the modification routines is given.

FILE NUMBER? 1 (OBJECT NAMES)

RECORD #	CONTENTS
1	CORE MEMORY
2	FILES
3	OPERATING SYSTEM
4	REMOTE TERMINALS

FILE NUMBER? 2 (THREAT NAMES)

RECORD #	CONTENTS
1	ALTERATION
2	BETWEEN LINES ENTRY
3	BOGUS TERMINAL
4	BROWSING
5	DESTRUCTION
6	DUMPING
7	ILLEGAL ACCESS
8	INTERCEPTING SIGNOFF
9	MASQUERADING
10	READING RESIDUE
11	TRAP DOORS
12	WIRE TAPPING

FILE NUMBER? 3 (COUNTERMEASURE NAMES)

RECORD #	CONTENTS
1	ACCESS LIST
2	APPEND ONLY BIT
3	AUTHORIZATION ALGORITHM
4	BACKUP COPY
5	BADGE ACTIVATED TERMINAL
6	BOUNDS REGISTERS
7	CHECKSUM
8	DEFINE ALL OP CODES
9	ENCRYPT DATA
10	ENCRYPT NULLS
11	FETCH PROTECT BIT
12	FILE ID
13	OVERWRITE CORE
14	PASSWORD
15	PRIVILEGED MODE
16	PROTECTED PAGING
17	READ ONLY ACCESS
18	RINGS
19	SYSTEM SIGNOFF CONFIRMATION
20	TERMINAL ID (WIRED IN)
21	TERMINAL LOCK
22	USER ID
23	WRITE RING (TAPE FILES)

Figure A1a - Contents of Name Files

FILE NUMBER? 4 (OBJECT-THREAT LINKS)

RECORD #	CONTENTS				
1	7	10	0	0	0
2	1	4	5	0	0
3	1	6	11	0	0
4	2	3	8	9	12

FILE NUMBER? 5 (THREAT-COUNTERMEASURE LINKS)

RECORD #	CONTENTS				
1	7	14	17	23	0
2	10	0	0	0	0
3	3	14	20	22	0
4	1	2	12	14	0
5	4	0	0	0	0
6	6	16	0	0	0
7	6	11	0	0	0
8	19	0	0	0	0
9	3	5	14	21	22
10	13	0	0	0	0
11	8	0	0	0	0
12	9	0	0	0	0

FILE NUMBER? 6 (COUNTERMEASURE WEIGHTS)

RECORD #	CONTENTS
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1
12	1
13	1
14	1
15	1
16	1
17	1
18	1
19	1
20	1
21	1
22	1
23	1

FILE NUMBER? 7 (FILE LENGTHS)

RECORD #	CONTENTS
1	4
2	12
3	23

Figure A1b - Contents of Link, Weight & Length Files

A1. FILE STRUCTURE

This program utilizes a direct-access file structure for the permanent retention of lists of Object, Threat, and Countermeasure names. Most BASIC interpreters support secondary storage files with direct access to individual records. Minor modification of the search routines would allow true sequential access files to be employed. Some degradation of execution time would result.

Seven files are used in the system. File #1 contains a list of names of Security Objects stored in fixed size records, one name per record. Currently, names are limited to 32 characters. This restriction is easily modified to any reasonable length. Most BASIC systems restrict string lengths to 256 characters maximum. The Object names are stored in alphabetical order for ease in reading file listings and to simplify the search routines.

Files #2 and #3 are similarly organized lists of Threat (#2) and Countermeasure (#3) names. The contents of these files are read into fast core tables by the search routines during program execution and printed on the user terminal.

File #4 contains the linkages between each Object in File #1 and its associated Threats on File #2. Each record on File #4 corresponds (in order) to an Object name and contains a list of integers which index the Threat name file. Currently, each record contains 5 numbers. This record length is a compromise allowing a reasonable number of threats per object while holding search time and file space to minimum values. Record size is easily increased if required.

File #5 is identical in structure to File #4. It contains the linkages between Threats (File #2) and Countermeasures (File #3). File #6 serves as permanent storage for "weights" of Countermeasures. Each record contains one integer field. This value represents a relative utility ranking of this Countermeasure against others associated with the same threat. Currently, these weights are all set to "1". Some proposals for the use of this weight field are discussed in Section 7 above.

File #7 contains the lengths of each of the other 6 files. This information is needed to interface properly with certain features of the BASIC file handling routines which are peculiar to the system upon which this program was developed.

A2. FILE MODIFICATION

In addition ot the basic Security System Synthesizer which makes up the main program the system contains routines to allow addition of new Objects, Threats, and Countermeasures to the files. These routines accept a name as input from the terminal and perform the required shifting of entries in the appropriate file (File 1, 2 or 3) to maintain alphabetical ordering. The associated link file (in the case of Objects and Threats) or weight file is also shifted. The new link is calculated from further terminal input in the following manner. If an Object is added, the system requests associated Threats. The Threat name file is searched for each name entered at the terminal. The position of the named Threat in the file is then recorded in the new location in File 4 (the Object-Threat link file). Up to five entries per new Object are

possible currently. Similar action occurs when a new Threat is added with respect to Countermeasures. When a new Countermeasure is added, the associated weight is merely entered directly from the terminal and stored on File #6.

When a new Threat is entered somewhere in the middle of File #2, all entries which are lexicographically greater must be moved up one record to free a record. Accordingly, the Object-Threat links to these shifted entries are no longer valid. A correction is achieved by scanning the link file (File #4) and adding 1 to all links having a value greater than or equal to the position of the new entry. A similar procedure is applied to correction File #5 when a new Countermeasure is entered in File #3.

It is also possible to delete entries by specifying only the name and the type (Object, Threat, or Countermeasure). The proper name file is searched for a duplication of the entered name. When located, the file entries are shifted down by one record so that the deleted name is overwritten. The link files are patched up in a manner analogous to the procedure for adding entries described above.