SELECTIVE SECURITY CAPABILITIES IN ASAP - A FILE MANAGEMENT SYSTEM

by

R.W. Conway, W.L. Maxwell, and
H.L. Morgan

Technical Report
No. 70-62
June 1970

Department of Computer Science
Cornell University
Upson Hall
Ithaca, New York 14850

DEPARTMENT OF OPERATIONS RESEARCH
COLLEGE OF ENGINEERING
CORNELL UNIVERSITY
ITHACA, NEW YORK

TECHNICAL REPORT NO. 112

May 1970

SELECTIVE SECURITY CAPABILITIES IN ASAP - A FILE MANAGEMENT SYSTEM

by

R. W. Conway, W.L. Maxwell and H.L. Morgan

Selective Security Capabilities in ASAP - a File Management System

R. W. Conway, W. L. Maxwell and H. L. Morgan

Although the general requirements of data security in file management and retrieval systems are fairly obvious, and have been often stated (1,2) current practice does not provide even an approximation of these requirements and, in fact, current languages and operating systems, with few exceptions (3,4) make it relatively difficult to achieve these objectives when one tries to do so. Apparently, current languages and hardware serve as an informal security and privacy protection system that is more or less satisfactory for most current applications. While security and privacy controls are often discussed, they do not appear to be high on most managers' lists of pressing problems.

The authors' concern with security is limited to the systems implications which arise when one or both of the following circumstances exist:

1. A common file is used for two or more different purposes, e.g. an integrated management information system.
2. A file is directly accessible by those who are responsible for generating input, and those who use the information it contains.

Questions of physical security of facilities, tampering with hardware etc. have been treated elsewhere (5,6) and will not be discussed here.

At present, the programming languages and operating systems that are used for most file maintenance and information retrieval applications are sufficiently complex and esoteric that professional programmers and/or production controllers have been interposed between the ultimate users of the file and the file itself. Moreover, the majority of such applications are still handled by relatively conventional batch processing systems, without remote access devices being placed directly in the hands of the information supplier or consumer. Both the physical separation, and the necessity of human review of access to the file serve to provide an informal but adequate security system to protect against abuse by the information consumer. The programmers and production controllers in such a system, however, often have considerable freedom to attack the security or privacy from within. (How many data processing departments are able to keep salary information from their programmers?)

It is clear that remote access systems represent the future for such applications -- the only question being how rapidly the future will arrive. It is also clear that truly user-oriented languages now exist (7) and once data-consumers have experienced the power and convenience of these languages they will not willingly return to a situation in which they are at the mercy of a staff programmer. Both of these developments suggest that formal and explicit treatment of the questions of data security is rapidly becoming a necessity. The following describes the approach to these questions that has been taken by the authors in ASAP, a file maintenance and information retrieval system designed and implemented by the authors.

There are three principal questions associated with the security of a particular file:

1. Which people with physical access to the hardware containing the file should have access to any particular file?
2. What subset of the file is each authorized user permitted to access?
3. What types of processing operations are permitted to each authorized user?

ASAP approaches each of these questions by using a complete directory of authorized users which is appended to the master file in question. While ASAP cannot prevent the data-set that represents the master file from being processed by independent programs written in other languages, it can and does supervise all requests for information entry, update, or retrieval which are written in the ASAP language, and in doing so enforces the security restrictions that are explicit in the user directory. This directory contains, for each authorized user:

1. "password" identification information.
2. A description of the file subset accessible to this user.
3. A description of the processing actions this user is permitted.

Record Selection

The first method of specifying the subset accessible to a particular user is to describe which records of the file are accessible to him. This is done by including in the directory entry for each user a Boolean expression that describes by content those records to which he is permitted access. When a user makes a request, a monitoring routine based upon this filed Boolean expression is interposed between the file management utility routine and the user program. The result is unobtrusive, but absolute -- the user program only "sees" records for which the Boolean expression is satisfied; other records essentially do not exist for this user. The syntax of ASAP permits the user to easily further restrict the selection of records from the file, but this individual Boolean constraint is automatically and invisibly 'ANDed' to the selection criterion for each and every request submitted by this user and there is no way he can override it without going completely outside of ASAP and obtaining independent access to the master file.

For example, consider a master student file at a university in which there is one record for each student. One user of this file might be the administrative office of the College of Engineering. (The system is, of course, quite content with such 'corporate' users, actually representing a number of different individuals.) The selection condition for this user would probably be something like:

```
COLLEGE = 'ENG'
```

where COLLEGE is the name of a field in the record and 'ENG' is one of the possible values of the field. The user representing the office of the Dean of Women might be assigned the selection restriction:

```
SEX = 'F'
```

and a user concerned with those football players on athletic scholarships who are in academic difficulties might be restricted with:

```
ATHLETICS = 'F' AND FINANCIAL_AID > 1000.00 AND GRADE_AVG < 2.0
```

A complete program in ASAP could consist of the following statement:

```
FOR ALL STUDENTS WITH CLASS = 'FR', PRINT LIST: NAME, CAMPUS_ADDRESS,
                    CLASS_RANK, ORDERED BY CLASS_RANK;
```

If this request were submitted by the College of Engineering it would produce a listing of all freshman engineering students. Exactly the same request submitted by the Dean of Women would yield a list of all freshman women.

While the primary purpose of providing this mechanism is to restrict the distribution of information to those users with a pre-established need-to-know, it has a secondary effect which may be more important and interesting than the security aspect for many applications. This is the fact that each user can deal with his accessible subset of the file exactly as if that constituted the complete file. He is in every sense unaware of the existence of other records and can employ the full facility of the language and the system for his subfile, without requiring any special treatment because it is a subfile. This means that most of the difficulties associated with multiple, decentralized files can be avoided without sacrificing their simplicity of use. One can avoid the currently typical situation in which an individual student is the possessor of a score or more records in different and separated files in various stages of update and decay.*

Field Security

A second method of defining the subset of the file accessible to a particular user is to assign different 'security classes' to different fields within each record. When the record is defined each data element (field) is assigned to one of four security classes. One class is called unrestricted and this is the default when no explicit security designation is given; the others are designated 1, 2 and 3. The classes are independent, with no

*In conventional processing of sequential files the 'invisible' records in such a system would impose a prohibitive processing penalty on the user of a small subset. This is avoided in ASAP by simultaneously processing many independent requests in one pass through the master file.

hierarchy implied by the level numbers.

When each user is enrolled in the system the security classes to which he has access are explicitly listed. While every user has access to unrestricted information, access to each of the other three classes must be explicitly granted or denied. Field and record security are multiplicative - a user may see only authorized fields in authorized records.

To continue the previous example, the fields of the student record might be classified in the following way:

Unrestricted information:
Name, campus and home address, telephone, college, class.

Security Class 1 - Personal/biographical information:
Medical history, draft status, disciplinary actions, race, religious preference.

Security Class 2 - Academic information:
Class standing, individual course records, standardized test scores

Security Class 3 - Financial information:
Treasurer's account, financial aid, family financial report.

Now the administrative office of the College of Engineering, whose access is already limited to records for which COLLEGE = 'ENG', could also be limited to the fields in those records that are unrestricted and class 2 information. Fields containing personal and financial information, as well as the complete records for students not in engineering are inaccessible to this user.

As in the case of record, a significant advantage of the system is the opportunity to combine several different types of information in a single master file, and keep that information out of reach of unauthorized users, out of the way of users who don't want it, and still have complete information readily available when it is required.

Restriction of Processing Operations

A third, almost independent type of security concerns the type of actions that a particular user can perform. Actions are classified in the following way in the ASAP system:

1. Read only access
2. Read/write access to existing records
3. Record creation and deletion
4. Ability to extend language by adding definitions*
5. Attach read-only external procedures
6. Attach read/write external procedures
7. Alter the record definition
8. Modify the user directory -- add or delete users, or change the security restrictions for existing users

The directory entry for each user includes a list of the types of actions that are permitted to this particular user. Read only access to the accessible fields of accessible records is assumed; all other types of actions must be explicitly authorized. No hierarchy is implied in the above list so that, for example, the ability to extend the language does not imply write access to the file+. Obviously, the ability to modify the user list must be granted with great frugality since this implies the ability to obviate the entire security system.

Implementation

ASAP is a compile-and-go system -- each run begins from source, and security tests are applied at the source language level. Since no object

*,+ In ASAP, users may define report formats, input formats, codes, and other frequently used constructs. These are then catalogued for future reference. Often, therefore, users will extend the language by adding special reports or macros to the system, an action quite independent of the ability to write on the master file.

decks are ever produced by the system, there can be no opportunity for security to be obviated by an alteration to the object program. Although the compile-and-go strategy is unusual in the data processing environment, it does not exact the penalty in compilation time that might be expected. The compiler is very fast, and recompiles source in time comparable, to if not better than, time required to load the object modules of a corresponding program. (On an 360/40 ASAP compiles at card reader speed, on a 360/65 at 50-100 statements per second).

The first card of each user program is an identification card bearing the user's password. This is used to search the user directory for the appropriate entry. When the user's entry is found, the record selection condition is passed, in source language form, to the compiler, along with two bit masks, one for field security, and one which specifies permissible actions.

The record selection condition is compiled into object code that is appended to the file management utility routines. This code is exercised for each record of the master file, thus representing some execution overhead. Since the compiled code is made part of each selection criterion, however, this often amounts to no more than two or three machine instructions per record. Only records that satisfy these conditions are passed on to the program representing the user-specified record selection criterion.

Both field selection and action restriction are entirely exercised at compilation time and represent no execution overhead whatever. Each entry in the symbol table, e.g., field name, report name, keywords, etc., contains a security mask. On each source reference to a field, the security class of that field is ANDed with a copy of the current user's security mask, and a simple test then decides whether or not the field is inaccessible to the user. If an attempt is made to reference a "classified" field, a condition code is set so that the run is terminated after compilation. When processing definitions of entities which may include information from several fields, such as reports or summaries, the security of the report is simply the OR of the security classes of all fields mentioned. Thus, while each field of a record may only have a single security class, a report may require the user to have access to several classes before he can use it.

The permissible action checking is done in a similar manner. Each action is tested against the action mask for the user for whom the code is being generated, and an improper action will cause the run to terminate after compilation.

Since references to the symbol table are made for every symbol scanned, the overhead involved in checking the security classification is rather small. And, since it is only exercised during compilation, at rather low cost a fairly large set of security restrictions can be enforced.

The different types of security are not quite independent in the ASAP system, although there is no technical reason why they could not be. When a user is permitted to attach an external procedure (which may be written in any programming language) the system does not attempt to enforce field security. Record security is still effective since the call to the external procedure is not made until after the record has been selected for processing. However, once the record is selected, the entire record is available to the external procedure. Although it would have been possible to pass an abbreviated copy of the record in which inaccessible fields had been blanked out, the substantial overhead this would require was considered unnecessary, since the use of external procedures requires a special linkedit which must be approved by a system controller.

Summary and Conclusions

The security provisions of ASAP are rudimentary, but are relatively simple to understand and use. Since the language is compiled, rather than interpreted, and most of the security features are fully exercised at compilation time, rather than at execution time, the security aspects of the system are not expensive to use. It will be interesting to observe in time whether these simple and inexpensive features of the system will be widely exploited by users. If not, then perhaps security considerations have been overemphasized by those in research and development and do not constitute an important general problem. If they are used, it is likely that users will become more sophisticated in their requirements and these simple classifications may not long suffice. It is, of course, no problem to increase the complexity of record selection conditions, or the number of classifications in either field or action restrictions, but it may turn out that other types of security exist, with concepts unlike those here.

Clearly, the ASAP security system is mainly designed to prevent the casual user from gaining access to information he should not see, and the determined professional would have little trouble going around these security measures, using other languages.

It is interesting to consider the problems of implementing even this rudimentary security system for a conventional file -- that is one created, updated, and interrogated by programs written in a contemporary general purpose language (PL/I, COBOL, etc.) and executed under a standard operating system. One might then conclude that the task is most formidable, and that if security is a real problem, the software now in general use is not up to solving it.

RFFERENCES

1. Hoffman, Lance. Computers and Privacy: A survey, Computing Surveys 1, 2(June 1969), 85-103.

2. Bates, William S. Security of computer-based information systems. Datamation 16, 5(May 1970), 60-65.

3. Weissman, Clark. Security Controls in the ADPET-50 time sharing system. Proc. AFIPS 1969 FJCC, 119-133.

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

5. Brandt, Allen, "Danger Ahead! Safeguard your computer." HBR, Nov-Dec. 1968, 97-101.

6. Molho, L. Hardware Aspects of secure computing. Proc. AFIPS 1970 SJCC, 135-142.

7. ASAP System Reference Manual. Compuvisor Inc., Ithaca, N.Y. 1970.