9TH NATIONAL BUREAU OF STANDARDS/
NATIONAL COMPUTER SECURITY CENTER

NATIONAL COMPUTER SECURITY CONFERENCE

15-18 SEPTEMBER 1986

ADDENDUM

PROCEEDINGS

COMPUTER SECURITY-
for today...
and for tomorrow.

TABLE OF CONTENTS

RISK ANALYSIS METHODS ADAPTED TO COMPUTER SECURITY

Rex V. Brown
Decision Science Consortium, Inc.
7700 Leesburg Pike, Suite 421
Falls Church, Virginia 22043
(703) 790-0510

Abstract

Risk analysis is an established field, but it is not yet well adapted to the needs of computer security evaluation. This paper reviews deficiencies in the state-of-the-art and proposes some promising directions for remedying them.

Introduction

Risk analysis as a methodological area was originally developed to handle the purely monetary and well-documented risks of accident and life insurance, and could rely on established statistical techniques[1]. It was then extended to cover multiple, poorly documented (but still well-defined) risks to health and safety, especially in the context of regulating new technologies[2,3]. This required consideration of subjective uncertainty and of tradeoffs between conflicting objectives such as the dollar value of human life. It had to draw on the concepts of personal probability and decision theory[4,5]. More recently, it has been extended to cover more diffuse risks such as the environmental impact of major proposed projects (as required by the National Environmental Protection Act) and this has required the use of multiattribute utility theory, involving intangible effects[6].

The emergence of computer security as a major risk management area has surfaced a new level of complexity, due to its unusually diffuse effects and to the source of the risk being a human adversary. The diffuseness of computer security risk stems from the fact that the interests of several different constituencies are being served (e.g., the nation's security, the facility operator's cost, the government's bureaucratic convenience) and that any specific consequences are difficult to enumerate, much less measure. In fact, computer security represents a larger category of risk management problems typified by diffuse risks from adversarial sources, which also includes risks of terrorism, nuclear theft, espionage, and nuclear proliferation.

Risk analysis problems can be thought of as varying along two dimensions: diffuse versus focused effects; and adversarial versus non-adversarial sources. As Figure 1 shows, many kinds of computer security problems, particularly those which have to do with safeguarding of information, rate high on both dimensions, and therefore appear in the top right of the figure. They

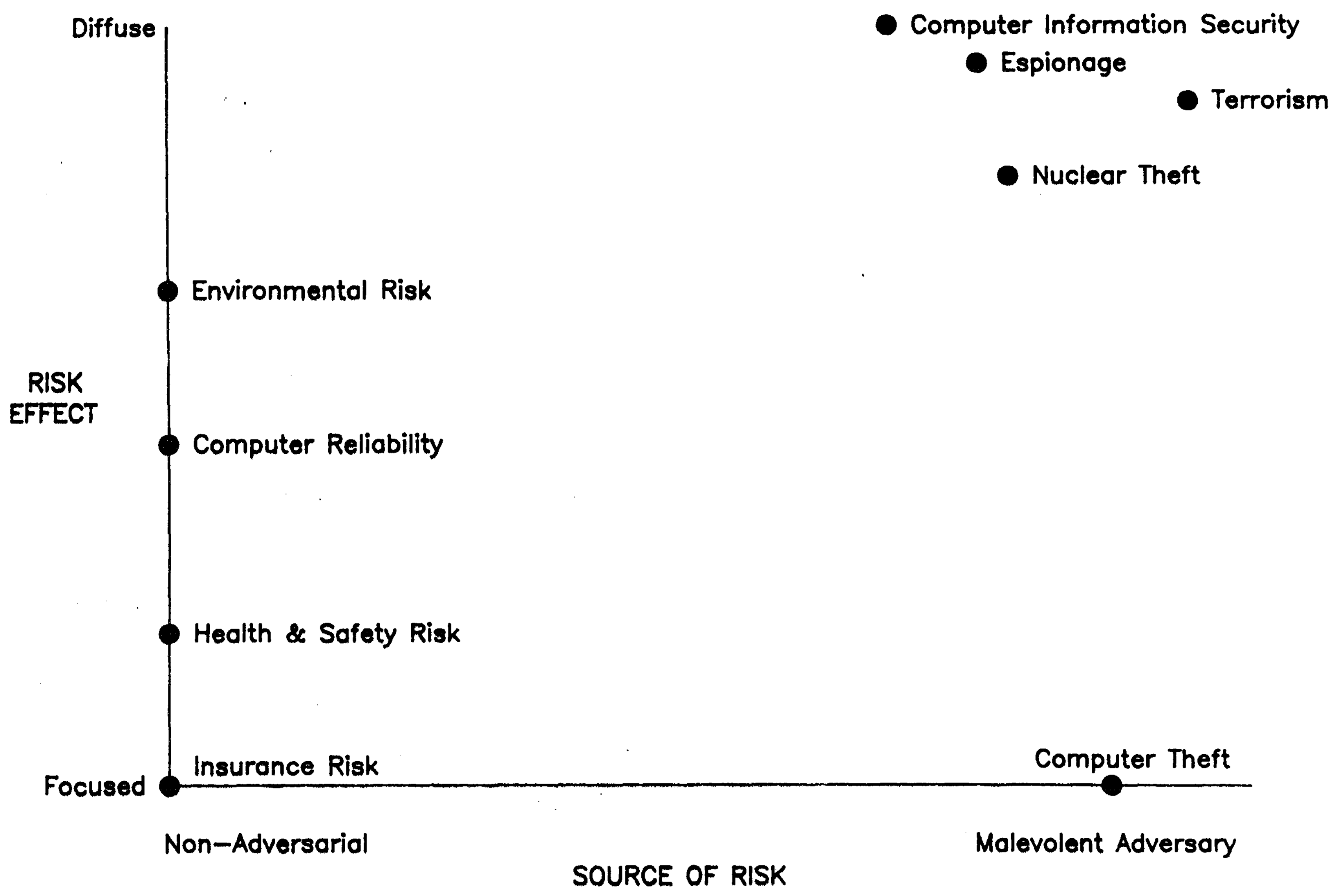


Figure 1: Categorization of Risk Management Problems by Effect and Source of Risk

have diffuse risks and adversarial sources, and for simplicity, we will call such problems DR/AS. Note, however, that computer theft has highly focused risks--mainly money; and computer reliability, on the other hand, though somewhat diffuse in effect, usually stems from a non-adversarial source. This is reflected in their locations in the figure.

In this paper, I am mainly concerned with discussing methodology to be developed for aiding DR/AS problems, and computer information security problems in particular.

In stating that DR/AS is a new methodological area, I do not wish to imply that no serious or high-quality risk analysis has been done on computer security or other DR/AS problems. The one-day computer security risk analysis conference preceding this one provided some interesting examples. However, I do suggest that the state-of-the-art is very primitive, either based largely on some ill-fitting adaptations of risk analysis techniques developed for other, less complex purposes; or that it has been developed, ad hoc, for specific problems, usually not in computer security, but in some other DR/AS area such as nuclear safeguards. In any case, I would agree with Lance Hoffman, in the talk preceding mine, that major work is needed to develop general-purpose risk analysis methodology, including the basic data to feed such methodology[7].

I will now suggest some innovative directions for DR/AS risk analysis methodology to be developed, and the role that the practitioners and managers in the field of computer security (as contrasted with risk analysis specialists) can play in its development.

General Principles

A promising conceptual framework which has guided several of the more successful DR/AS studies is *personalized decision analysis* (PDA) which is a well established technique for quantifying the judgments of uncertainty and value that go into any decision[4]. We call it "personalized" to distinguish it from the many other approaches to analyzing decisions.

For those of you who are familiar with decision trees, Figure 2 gives a schematic representation of a hypothetical case, where some national computer security policy alternatives are being evaluated in terms of their ultimate impact on the interests of relevant constituencies (such as society as a whole, facility operators, and the central government). In principle, such a tree could be fleshed out with relevant branches spelled out in detail, with uncertainties measured by probabilities, and with values represented by a utility function. If the tree properly captures all the policy maker's data, perceptions, and judgments, his preferred option would be the one with the highest "expected" utility (i.e., a probability weighted average). In practice, such an ideal and

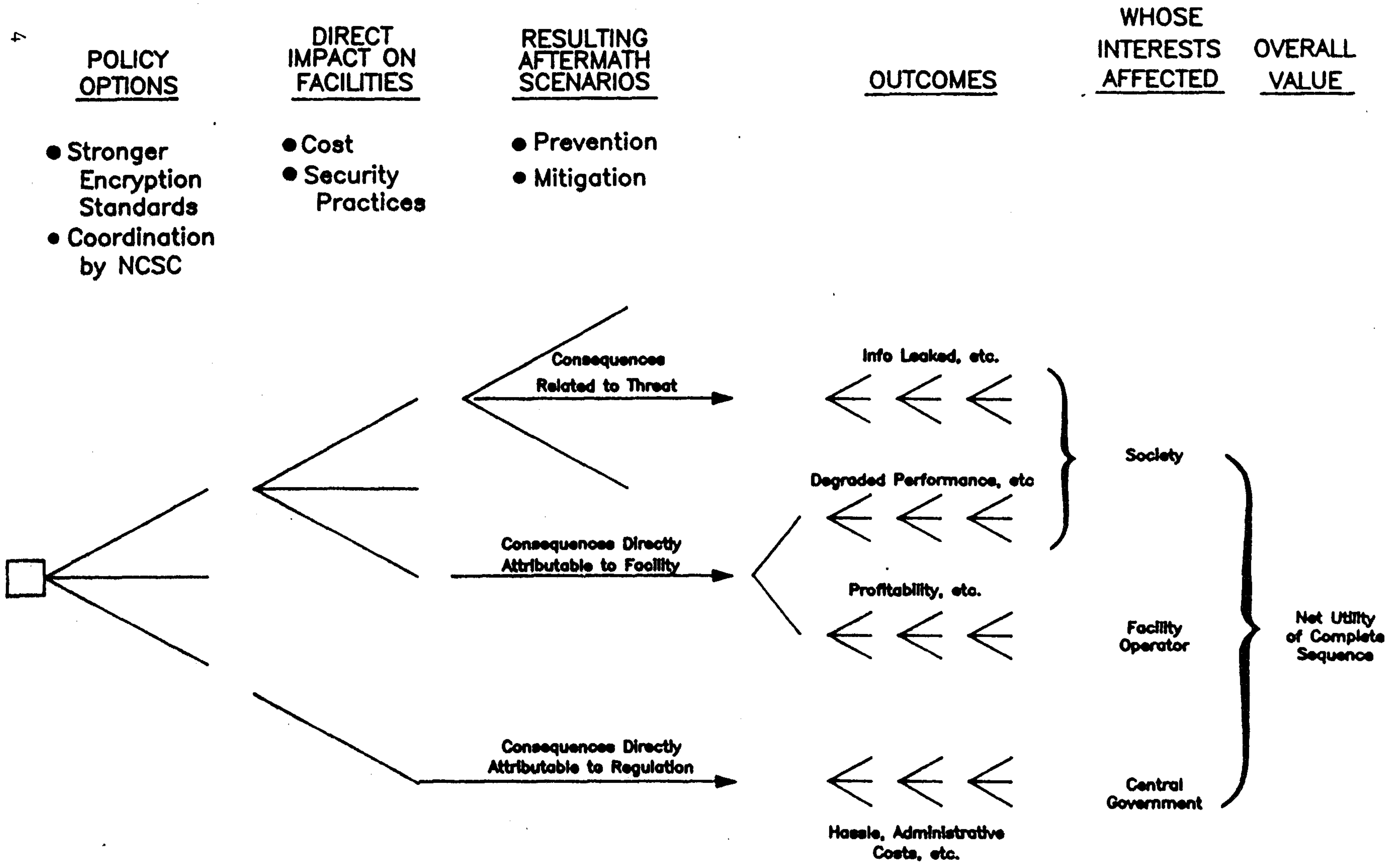

POLICY OPTIONS
● Stronger Encryption Standards
● Coordination by NCSC
DIRECT IMPACT ON FACILITIES
● Cost
● Security Practices
RESULTING AFTERMATH SCENARIOS
● Prevention
● Mitigation
OUTCOMES
WHOSE INTERESTS AFFECTED
OVERALL VALUE
Consequences Related to Threat
Consequences Directly Attributable to Facility
Consequences Directly Attributable to Regulation
Info Leaked, etc.
Degraded Performance, etc
Profitability, etc.
Hassle, Administrative Costs, etc.
Society
Facility Operator
Central Government
Net Utility of Complete Sequence

Figure 2: Decision Tree Framework to Evaluate National Policies on Computer Security

comprehensive analysis is not realizable, and probably should not be attempted, but it provides a useful conceptual framework within which to construct a more manageable and useful analysis.

A more limited, and therefore more usable, analysis within this framework is indicated in Figure 3 where the current computer security risk for a particular facility is assessed. (A distinction is made in risk analysis circles between "risk assessment" as referred to here, which assesses factually what the risks are, and "risk management," where specific measures to combat the assessed risks are evaluated.)

A second general principle (in addition to PDA) I would advocate for computer security risk analysis is **plural analysis**[8]. By this, I mean developing multiple approaches to given problems when no single approach can assure adequate confidence in the findings. This is certainly the case with the current state-of-the-art of risk analysis for computer security. Two half-baked approaches are likely to be more useful than one three-quarter-baked approach! I should warn you that you are likely to find resistance to this idea from the research community, for reasons that have to do with the parochial perspective of a technician, rather than the best interests of the client. Your typical technician sets high store on avoiding technical criticism, and two half-baked approaches present more tempting targets for a critic than one three-quarter-baked approach.

We will now discuss some specific avenues for developing appropriate methodology for DR/AS.

Analyzing Diffuse Risks

There are two critical problems with the diffuse risk aspect of DR/AS, formulating risk consequences, and evaluating them.

The main problem with formulating consequences for diffuse risks is deciding at what level of aggregation to describe them. One end of the scale would be to specify in great detail all possible scenarios. This has the advantage of concreteness and ease of comprehension, but may pose an unmanageably burdensome task and it runs the risk of being seriously incomplete. For this reason, traditional Monte Carlo simulation is normally infeasible or inadequate. We have developed an alternative, called "step-through simulation," which avoids having to prejudge and anticipate the whole panoply of possible scenarios. However, it has not yet been developed very far except for certain military combat cases[9]. There have also been specific applied attempts at a

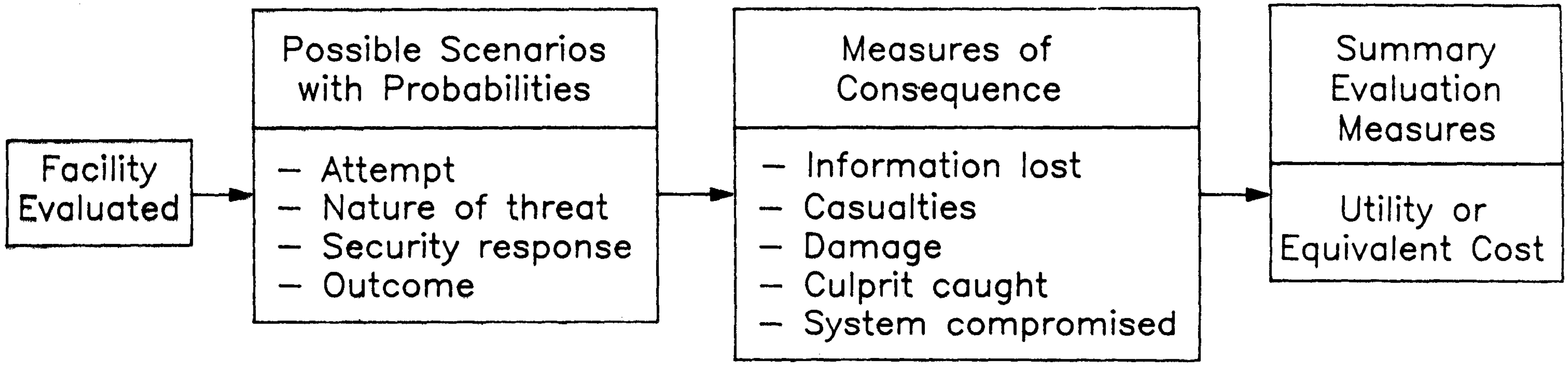


Figure 3: A Limited Facility Risk Assessment Model

slightly more aggregated level to model societal consequences of malevolent acts against energy facilities[10].

The other end of the scale is to work with broad, but comprehensive, attributes of interest. This makes for a simple structure, but very difficult assessment. Figure 4 shows an attempt to list general concerns to be addressed in evaluating national computer security policy options. It also shows a set of importance weights designated to be general enough that they will fit a wide variety of specific policy being evaluated. The weights indicated here, for example, that risks to national security from defense data sources are considered five times as important as national security risk from civilian sources, and ten times as important as economic loss to the government. This apparently simple structure masks some very subtle and important issues, such as how exactly you define the importance weights and how you handle interactions between different attributes, which we will not go into here.

A simple and largely qualitative version of multiattribute utility analysis is shown in Figure 5 (it relates to some organizational options for managing and coordinating national computer security policy, which were considered in a study conducted for the congressional Office of Technology Assessment[11]. The plusses and minuses in the body of the table indicate the assessed impact on each of the attributes listed across the top of each of the options listed down the left. The asterisks in the "weight" row represent the relative importance of each attribute.

Since my purpose is to discuss methodology rather than substance, I will leave you to guess at what the specific options under consideration (cryptically abbreviated here) may have been! In any case, the inputs here are to be treated as purely hypothetical. They could have been generated in any of a number of ways: as the personal judgment of a single expert (or a collection of complementary experts); or as the consensus of a group workshop; or as the product of any of a number of ad hoc studies, addressing particular parts of the input.

The Figure 5 chart, with its inputs, can then be used simply as a compact summary of "pros and cons" on the basis of which a policy maker makes up his own mind informally. Alternatively, the plusses and minuses can be totted up (with due account for differential weighting) as done here in the right-hand column (which suggests the three middle options are best). It might be argued that this is a hopelessly "unscientific" risk analysis, which no reputable risk analyst would sully his reputation with. I would argue, on the other hand, that this is often as far in the direction of scientific rigor as you will want to go in presenting the case to a busy, non-technical decision maker; and it is the naive analyst who will burden him with erudite findings! Indeed, I have been involved in studies where a sophisticated quantitative analysis was done first and then trans-

RISKS AND VULNERABILITIES	WEIGHTS*
National Security (vs. Foreign Threat)	
Defense Source of Data (100)	10
Civilian Source (100)	2
Economic Loss Risk (e.g., Via Comp. Crime)	
Government ($1B)	1
Business ($1B)	.2
Public ($1B)	.5
Other Risks	
Public Privacy (100)	2
Gov't Services (e.g., Reliability) (100)	2
$ COSTS OF OPTIONS	
Direct $ Cost (Out-of-Pocket)	
Government ($1B)	1
Business ($1B)	.2
Public ($1B)	.5
Indirect Cost (e.g., Impaired Service) ($1B)	1
OTHER IMPACTS	
Bureaucratic Upset (100)	
Democratic Values (Civil Liberties, Open Government, etc.) (100)	1

* EQUIVALENCE IN BILLIONS OF FEDERAL DOLLARS A YEAR FOR THE VALUE SWING IN PARENTHESES (EITHER $1B A YEAR OR 100, DEPENDING ON WHETHER SCALE IS MONETARY OR QUALITATIVE).

THE 100 PT. SCALES ARE DEFINED AS FOLLOWS:
0 = STATUS QUO FROZEN INDEFINITELY (NOT THE SAME AS THE "DO NOTHING" OPTION, WHICH CAN GET WORSE).
−100 = MAXIMUM PLAUSIBLE DETERIORATION (DEFINE ARBITRARILY, BUT BE CONSISTENT).
+100 = AS MUCH BETTER AS −100 IS WORSE.

Figure 4: Attributes for Valuation of Consequences with Importance Weights

ATTRIBUTES	Risks Addressed								Option Costs				Other Attributes			Net Evaluation
	Nat. Sec.		Econ. Loss			Priv–	Serv–		Direct			Ind.	Bur.	Democ.		
	Def.	Civ.	Gov.	Bus.	Pub.	acy	ices		Gov.	Bus.	Pub.	'tax'	Upset	Value		
WEIGHTS	***	**	**	*		*	*		**			**	*	**		
OPTIONS																
0. Do Nothing																
1. Cancel 145		–											–			–3
2. Impl. 145		+	–	+		–	+		–			–	–	–		–7
1. Modify 145																
1. Civil. Input		+		–			+		–				+	+		+3
2. Civil. Agency																
1. NBS+			+	+					–				+			+3
2. New			++	++		+	+		––				–			+3
3. Govt.–wide Ag.																
1. CSC+	–		+	+					–				––	+		–2
2. Broader	–		+	+		++			––				––			–3

Figure 5: Qualitative Evaluation of Organizational Options for Information Security

lated into a more digestible qualitative form for the decision maker. (For example, a decision analysis on U.S. export policy was turned into a qualitative argument for Henry Kissinger, who was known to distrust numbers.) This is another case where the researcher should be discouraged from doing what he likes best (doing fancy mathematics), if something simpler to present is more useful to the client.

Figure 6 shows a similar, but one degree more quantified, analysis of another set of policy options, but using the same attributes of value. The case addressed here has to do with civilian telecommunications security options. It suggests that, if the numbers and analysis are accepted, the option of performing R&D on encryption is the option to be preferred.

Although both Figures 5 & 6, in fact, reflect a particular individual's preliminary reflections on the issues concerned, they could easily serve as "macro models" which are a distillation of more intensive studies, possibly involving many expert judgments, field surveys or mathematical models[12]. By using a macro model as the primary link with the decision maker, we reap several important advantages. It simplifies communication with the decision maker (who may not have the time or inclination or competence to evaluate a fine-grained micro model). It permits plural analysis by allowing the user to consider and merge alternative inputs to the chart, including his own intuition. It can also be used to guide further analytic effort, by indicating, through sensitivity analysis, where firming up inputs is most likely to affect conclusions. (This can help counteract a tendency among researchers to want to do new research on those topics that they already know most about--rather than where it is most needed.)

There is an important point to be made in using such analyses to guide a decision maker. There will typically be considerations (like "bureaucratic upset" in this set of attributes), which the decision maker may not want to make public. In such cases, any formal analysis will either be embarrassing (if made public) or may lead to unwelcome conclusions (if the factor is omitted). Whether this is a good or a bad feature of formal analysis for risk management problems depends on whether you want to discourage the decision maker from taking into account considerations which he does not want to announce publicly.

Adversarial Source of Risk

The above macro model examples were at such a high level of aggregation that no specific risk event was assessed probabilistically. In a micro analysis they would be, for which a well-developed array of Probabilistic Risk Assessment (PRA) methods are available[13]. However, the fact that the source of risk may be a human adversary calls for some distinctive methodology. It

ATTRIBUTES[b]	Risks Addressed								Option Costs				Other Attributes			Net[e] Evaluation
	Nat. Sec.		Econ. Loss						Direct							
	Def.	Civ.	Gov.	Bus.	Pub.	Priv.	Gov. Serv.		Gov.	Bus.	Pub.	Ind. 'tax'	Bur. Upset	Democ. Value		
WEIGHTS[c]	10	2	1	.2	.5	2	2		1	.2	.5	1	.5	1		
OPTIONS[d]																
0. Do Nothing	0	−30	−20	−10	0	−50	−30		0	0	0	0	0	0		−2.41
1. Encryption	0	+20	+5	0	0	+20	0		−20	−10	0	−30	0	0		+.33
1. DES	0	+25	+10	0	0	+20	0		−50	−20	0	−35	−10[z]	0		−.30
2. Stronger																
2. R&D on Encryp	0	+20	+10	+20	0	+20	0		−30	−10	0	−15	0	0		+.47
3. Drop Satellite & Microwave	0	+20	+10	−10	0	+20	0		−40	−20	0	−10	−10	0		−.27
4. Dedicated Lines	0	+20	+10	−10	0	+20	0		−60	−20	0	−10	−10	0		+.09

Notes

a. See guidelines on completing form.
b. Dimensions of concern to government.
c. Relative importance of stakes under each attribute, or of $1B swing for costs & losses.
d. Scores are % of potential deterioration from present.
e. Weighted sum of scores.

Comments

z. NSA upset.

Figure 6: Quantitative Evaluation of Civilian Telecommunications Security Options[a]

is not only that the source is human, and therefore involves considering psychological issues. Predicting human error in the operation of a nuclear plant, for example, also has this property. It is also that it involves deliberate (and hostile) intentions that themselves are changed by the security measures taken (at least to the extent that the adversary knows about them). If you close up one avenue of attack for him, it leads him (probabilistically) to try a different mode of attack.

An early DR/AS problem we worked on was in the area of nuclear proliferation. The task was to assess the probability that the International Atomic Energy Agency would detect a country diverting fissionable materials from peaceful uses, by analyzing each "diversion path" the proliferating country might adopt. We had not only to take into account the probability of detection if the country followed each diversion path, but also have the probability that he would choose that diversion path, which is itself a function of the detection probability. Much the same would apply to the behavior of a foreign agent seeking to breech a computer security system.

Distinctive tools for assessing adversarial behavior do exist in the literature, but, to our knowledge, have not been very fully developed or applied. For example, game theory is logically well-established, but the necessary assumptions needed in most versions of it are rarely found in the real world, and I am not aware of any relevant successful applications[14].

A more promising alternative is "imputed PDA," in which we model the adversary's behavior *as if* he were using PDA to determine *his* decision. This approach has been used to predict *non*-adversarial behavior, for example, to predict when NATO would mobilize in the event of an impending (but unknown) Warsaw Pact attack[15]. It is critically important, however, to allow for "slippage" between the prescriptive PDA model and descriptive reality (which in the NATO case led us to almost double the mobilization delay implied by PDA). Variants of this approach have been developed by psychologists relating probability of action to relative expected utility[16].

Conclusion

In this paper, there has not been space to do more than touch lightly on the current state-of-the-art of risk analysis and some new developments, as they bear on the distinctive problem of computer security. My overall conclusion is that existing risk analysis techniques need substantial development and augmentation before the critical needs of computer security can be well served.

Acknowledgment

The preparation of this paper was supported, in part, by the Decision and Management Science Program of the National Science Foundation, under Grant No. SES 8360335. The author wishes to express appreciation to Lance Hoffman, who provided stimulus to work on this topic and suggested the idea for Figure 1; and to Granger Morgan for some valuable basic insights into the problem.

References

1. Kendall, M.G., & Stuart, A. *The advanced theory of statistics. Volume 2: Inference & relationship.* London: Charles Griffin & Company, 1961.

2. Lave, L.B. (Ed.). *Quantitative risk assessment in regulation.* Washington, DC: Brookings Institution, 1982.

3. Covello, V.T., and Menkes, J. Issues in risk analysis. Hohenemser, C., and Kasperson, J.X. (Eds.). *Risk in the technological society.* AAAS Selected Symposium 65. Boulder, CO: Westview Press, 1982, 287-301.

4. Brown, R.V., Kahr, A.S., and Peterson, C.R. *Decision analysis for the manager.* New York: Holt, Rinehart, and Winston, 1974.

5. Raiffa, H. *Decision analysis.* Reading, MA: Addison-Wesley, 1968.

6. Keeney, R.L., and Raiffa, H. *Decisions with multiple objectives: Preferences and value tradeoffs.* New York: Wiley, 1976.

7. Hoffman, L.J., *A research agenda for computer security risk analysis* (draft). Report on a DOD-sponsored workshop, Washington, DC: The George Washington University, Department of Electrical Engineering and Computer Science, January 1986.

8. Brown, R.V., and Lindley, D.V. Plural analysis: Multiple approaches to quantitative research. *Theory and Decision*, *20*, 1986, 133-154.

9. Ulvila, J.W., and Brown, R.V. Step-through simulation. *Omega: The International Journal of Management Science*, 1978, *6*(1), 25-31.

10. Hill, G.A. *Societal consequences of malevolent situations: Implications for safeguards policy* (Technical Report 81-8). Falls Church, VA: Decision Science Consortium, Inc., May 1982.

11. Brown, R.V. *Personalized decision analysis as an expert elicitation tool: An instructive experience in information security policy* (Report to OTA - Task 2) (Technical Report No. 85-9). Falls Church, VA: Decision Science Consortium, Inc., February 1985.

12. Brown, R.V., and Feuerwerger, P.H. *A macromodel of nuclear safeguard effectiveness* (Interim Report PR 78-6-80). McLean, VA: Decisions and Designs, Inc., March 1978.

13. *Risk Analysis*, Special issue on nuclear probabilistic risk analysis. Vesely, W.E. (Guest Ed.), December 1984, *4*(4).

14. Shubik, M. *Game theory in the social sciences.* Cambridge, MA: M.I.T. Press, 1982.

15. Brown, R.V., Kelly, C.W., III, Stewart, R.R., and Ulvila, J.W. A decision-theoretic approach to predicting the timeliness of NATO response to an impending attach (U). *Journal of Defense Research*, May 1977, Special Issue 77-1 (Crisis Management), 126-135.

16. Luce, R.D. *Individual choice behavior: A theoretical analysis.* New York: Wiley, 1959.

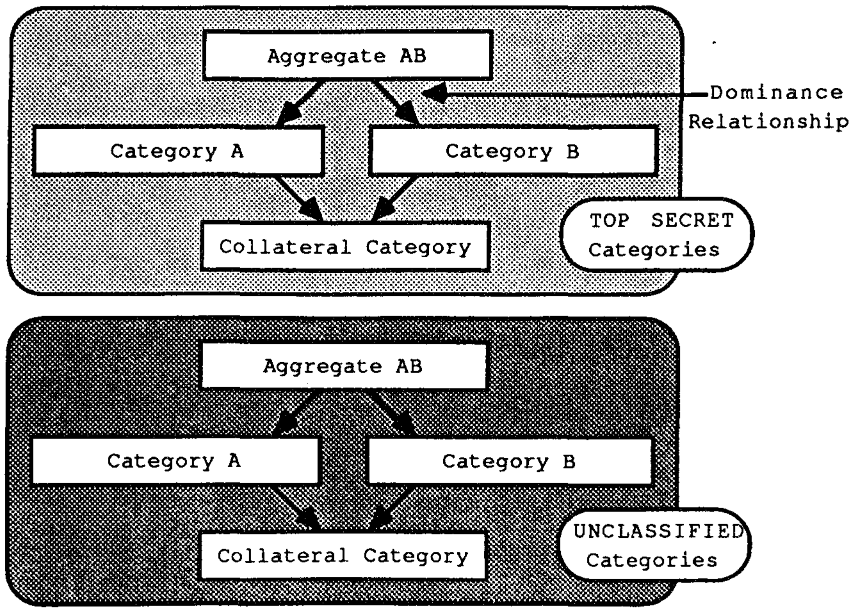
A PROPOSED POLICY for DYNAMIC SECURITY LATTICE MANAGEMENT

C. T. Ferguson
C. B. Murphy

Honeywell, Inc.
Secure Computing Technology Center
2855 Anthony Ln. So., Suite 130
St. Anthony, MN 55418

I. INTRODUCTION

In designing the System Security Officer (SSO) interface for the Secure Ada Target (SAT) [1], it has become apparent that the Trusted Computing System Evaluation Criteria (TCSEC) description of security levels and the activities of an SSO is insufficient and does not accurately reflect the intention of DOD security policy in all application areas. This discussion will describe the issues related to implementing a security lattice and SSO functionality for a Trusted Computing Base (TCB) and will describe a modified lattice model which permits arbitrary lattices to evolve which represent a wider range of security environments. The discussion will include a descriptive policy for controlling the creation, deletion, and aggregation of security levels and other level maintenance operations.



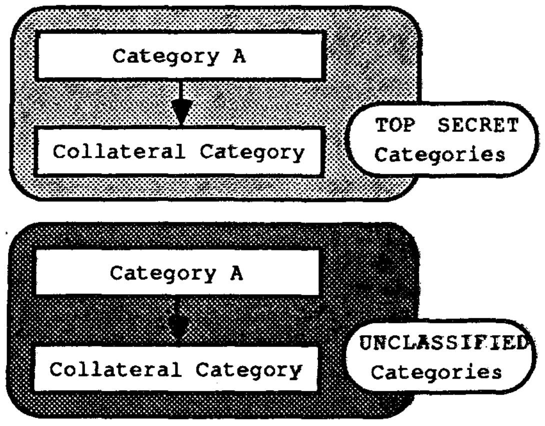
Complete Lattice After Category Addition
Figure 1

II. ISSUES

Lattice Configuration Issues

The TCSEC refers to the hierarchical levels and non-hierarchical categories of a security lattice as though they were independent and orthogonal attributes [2]. Hence, the TCSEC lattice is often thought of as a *complete lattice* having exactly one SYSTEM-HIGH security level and exactly one SYSTEM-LOW security level. The hierarchical levels form a *linear ordered lattice* arranged in order of dominance. At each hierarchical level exists an identical set of non-hierarchical categories. Dominance for these categories is computed by means of set theory. Thus, the set of categories is treated as a *lattice of subsets.* [3]

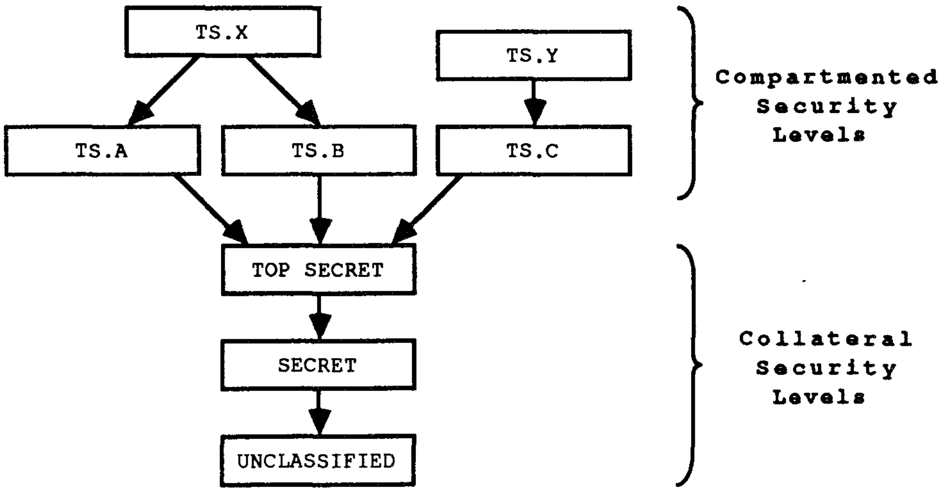
This description of a lattice does not correspond well to many security lattices. In reality, many security lattices are not complete in that there is not exactly one SYSTEM-HIGH. Furthermore, the TCSEC's concept of a non-hierarchical category does not map straightforwardly into environments where the need to know compartments available at each hierarchical level are not the same for all levels. To exemplify the problem, one might consider a complete security lattice having two hierarchical levels (TOP SECRET and UNCLASSIFIED) and two non-hierarchical categories (A and B). A complete lattice would require there be three UNCLASSIFIED non-hierarchical categories (A, B, and the aggregate AB) as illustrated in Figure 1. These UNCLASSIFIED categories may not be appropriate.



Complete Lattice with One Category
Figure 2

Another problem with the complete lattice model is that it may require the creation of undesirable compartments as a side effect of creating a new desired compartment. Consider a lattice with hierarchical levels consisting of UNCLASSIFIED and TOP SECRET. At each level there exists one category, A, as in Figure 2. Addition of another category, B, to the system would result in the creation of four new compartments (TS.B, TS.AB, U.B, and U.AB) as illustrated in Figure 1. Two of the newly created compartments will be aggregations of A and B which may not be desirable. Furthermore, it may have been desirable to create category B at TOP SECRET but not at UNCLASSIFIED.

Many security lattices are more accurately described as *partially ordered sets (poset's)* than lattices. The organization of the security poset is essentially arbitrary and there is no SYSTEM-HIGH. In contrast, there are multiple *maximal security levels* which can be thought of as local SYSTEM-HIGH's. Creation of need-to-know compartments and aggregation of compartments into dominating compartments in practice is under complete control of the *System Security Officer (SSO)* or other *poset administrators*. An example of such a poset is illustrated in Figure 3.



Typical Security Poset
Figure 3

Furthermore, some sites have been known to create additional hierarchical levels for a specific category. The configuration of such a poset would have one level hierarchically dominating a second level where both levels are of the same "conventional" hierarchical level. This situation is illustrated in Figure 3 by the levels TS.C and TS.Y.

<u>System Security Officer (SSO) Functionality Issues</u>

Security policies for the pen and paper world compartmentalize information for good reasons. Those reasons are equally valid when the paper world policy is mapped into TCB policy. For this reason, it is desirable for ***<u>all</u>*** users of a TCB, including the SSO, to be constrained by the principle of least privilege. Hence, the System Security Officer (SSO) should not be cleared any higher than would ordinarily be necessary for his pen and paper world activities. Many SSO functions require access to objects and all objects on a TCB should have a security level associated with them. The SSO, as with any other user of the TCB, has a distinct clearance level. The actions of the SSO should be limited to objects for which he has proper clearance and should be auditable.

Performing security level maintenance functions in a system requiring a single SYSTEM-HIGH would require some personnel to have a SYSTEM-HIGH clearance level. Such a requirement is considered dangerous in many environments. For a system employing a security poset with multiple maximal security levels, it would be equally dangerous to require a single person to be cleared to all maximal levels in order to perform level maintenance. Such a design defeats the purpose of *need to know* compartmentalization and introduces significant risks associated with SSO errors and system faults.

Security Lattice Management Issues

The TCSEC specifies or implies numerous requirements on SSO functionality and requires the SSO functions "shall be identified" ([2] section 4.1.3.4). Unfortunately, the TCSEC is silent on functions relating to lattice maintenance such as control of compartment aggregation, creation of new compartments, and deletion of old compartments. Not surprisingly, most multi-level TCB's to date have assumed the security lattice is static between system regenerations.

III. KEY ELEMENTS OF PROPOSED POLICY

Summarizing the issues presented previously:

- A *complete* lattice is ill-suited to actual DOD security environments.
- Requiring a single SYSTEM-HIGH is generally unacceptable for compartmented security environments.
- Requiring any personnel to be cleared to all compartments or to a SYSTEM-HIGH is considered dangerous and defeats the purpose of compartmentalization.
- The TCSEC does not specify how security levels are to be created, deleted, aggregated, or maintained.

The proposed remedy for the above issues is based upon the following concepts:

- **Explicit Creation of Security Levels and Dominance Relationships** - Realistic security posets will not necessarily need or want identical category sets at each hierarchical level. A method for explicit level creation and explicit denotation of dominance relationships will be defined and the capability of building completely arbitrary security posets with numerous maximal security levels will be provided.

- **Delegation of SSO Authority** - The SSO functionality cannot conform to the requirement of least privilege if a single SSO is required to perform all poset maintenance functions. Therefore, the responsibilities of the SSO will be delegated to other authorized personnel. A set of distinguished users, referred to as *Security Level Administrators (SLA's)*, will be designated by the SSO to execute security maintenance functions on specific security levels. The SSO will only be responsible for maintenance functions on security levels at or below his or her clearance level.

- **Two Key Control** - Critical functions, such as aggregation or deletion of a security level, will require approval of two authorized users. The authorized users will generally be the SLA's, the SSO, or a combination of both.

IV. DESCRIPTION OF PROPOSED POLICY

In the proposed policy, a hierarchical level and a non-hierarchical category set is replaced by a single entity referred to as a *security level.* Dominance relationships between security levels are designated explicitly. The collection of security levels and dominance relationships forms a *partially ordered set (poset).* The creation of new levels *(vertices)* or dominance relationships *(edges)* will be by distinct auditable action.

The maintenance of a security level is the responsibility of a distinguished user who will be referred to as the Security Level Administrator (SLA). There exists at least one SLA for every security level in the poset. The SSO is a distinguished SLA who generally performs security level maintenance for all security levels which are dominated by his or her clearance level.

Each security level is represented by a distinguished object whose contents describe attributes of the security level. This object is referred to as a *Security Level Descriptor Object (SLDO)*. The SLA is the only user authorized to modify the SLDO. The SLDO contains the following information:

- The names of security levels which this level immediately dominates.
- The number of security levels which immediately dominate this level.
- The names of the Security Level Administrators (SLA's) for this level.
- The names of all users cleared to this level.
- The names of all devices cleared to this level and the corresponding labels (if any) which represent this level on those devices.
- Names of devices and users that have been cleared to any immediately dominating levels.

The SLDO of a security level is created at the time of level creation and is classified at the level it represents.

Every security level contains at least one user that is designated as the Security Level Administrator (SLA). A SLA is designated at the time of level creation. This user is responsible for maintaining the security level. Alternate SLA's may be designated.

One function of the SLA will be to maintain the list of cleared users contained in the Security Level Descriptor Object (SLDO) for the security level. This list will contain one entry for every user cleared to the level. The entry will contain the user's name and may contain other information about the user such as group name, date of last password change, object ownership list, etc. The SLA will be responsible for keeping the data in the list of cleared users up to date. The SLA will be able to modify existing user entries, delete users, and add new users.

The System Security Officer (SSO) is responsible for maintaining an UNCLASSIFIED user database. All users are initially made known to the system by the SSO and given a clearance level of UNCLASSIFIED. The various SLA's of the system can grant clearance to their security level to any user in the User Database who is cleared to all security levels immediately dominated by the SLA's security level. This requirement will insure that for all levels a user dominates, the SLA of each level has properly approved the user's access to the level.

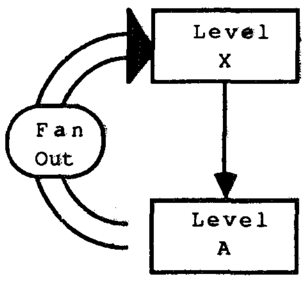
The SLA will also be responsible for maintaining the list of cleared devices contained in the security level's SLDO. Each device cleared to the level will have an identifier in this list. If the device is label preserving, the entry will contain labelling information which when sent to the device will represent the level. The label may be human readable. A device may have one or more human readable labels such as the character strings 'TOP SECRET' and 'TS'. Another human readable label might be an escape sequence representing a color or special font. An attribute of the labelling information in the cleared device list might be a description of a labeller to be used in sending information to the device.

Transferring ownership of objects is another function which must be provided for the SLA. This function is accomplished via tools which modify the object ownership lists for entries in the Cleared User List. The SLA will want to transfer object ownership is when it becomes known to the SLA that a user is no longer cleared to the level (the user might have died, left the company, or sold out to the Red hordes). In this case, all objects owned by the problematic user will be removed from his ownership list and given to another user cleared to the level.

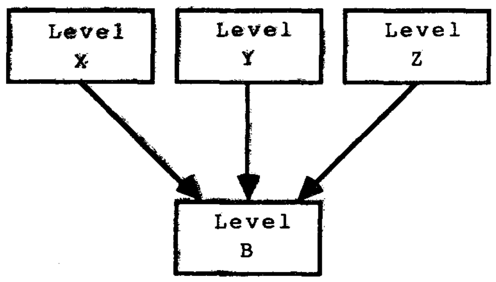
The SLA is authorized to import, export, and downgrade objects. The SLA has the capability of designating a new SLA or co-SLA for his level. The SLA may delegate a subset of his capabilities to other users cleared to his level. Delegatable capabilities might be the authorization to export and import objects to or from non-label preserving devices or authorization to downgrade objects.

V. SPECIAL LATTICE OPERATIONS

There are four lattice operations which modify the lattice configuration significantly and require special treatment. These operations are *Fan-Out, Fan-In, Forced SLA Replacement,* and *Security Level Deletion*. Two key approval is required to perform all these operations except Fan-Out.



Simple Fan-Out
Figure 4a



Complex Fan-Out
Figure 4b

Fan-Out

The Fan-Out operation creates a new security level immediately dominating the level from which it is created. A simple Fan-Out operation is illustrated in Figure 4a where level A has Fanned-Out to create level X. A SLA can perform a Fan-Out by creating a new Security Level Descriptor Object (SLDO) and performing a TCB operation which creates a new level from the object (*activates* the SLDO). By performing repeated simple Fan-Out's, a SLA can create a complex Fan-Out as illustrated in Figure 4b.

When the SLA of level A in Figure 4a Fans-Out to create level X, the following steps will be executed:

1. SLA (A) creates a SLDO. The new security level exists at this time but is inoperative.

2. SLA (A) assigns a unique name to the new SLDO. This name may be provided by the SLA or randomly generated by some dictionary on the system. In this example, the new level has been assigned the name X.

3. SLA (A) assigns at least one interactive console type device to security level X. This device is also given exactly one human readable label. This device should be one, such as a CRT terminal, which will allow the SLA for the new level to perform security level maintenance functions.

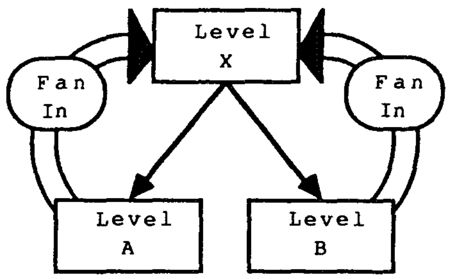
4. SLA (A) designates a user to be the Security Level Administrator for level X -- SLA (X). This user is the sole user cleared to level X at this time.

5. SLA(A) executes the TCB *Let-Dominate* operation and the level becomes operative.

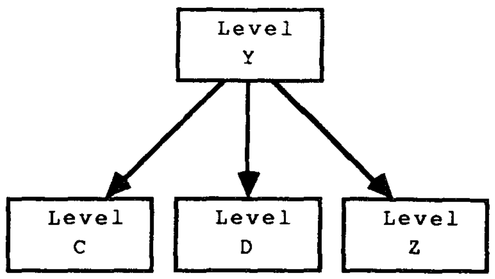
The new security level X will not become operative until all five steps have been completed. SLA (X) is now responsible for adding more users and devices to his level. SLA (A) can repeat the above operations to create a complex Fan-Out similar to that in Figure 4b.

Fan-In

The Fan-In operation creates a level which is an aggregation of two existing levels. A simple Fan-In is illustrated in Figure 5a where level A and level B have Fanned-In to create level X. The Fan-In operation requires the approval of both aggregating SLA's -- either SLA can veto the operation.



Simple Fan-In
Figure 5a



Complex Fan-In
Figure 5b

When the SLA's of levels A and B of Figure 5a decide to aggregate, following steps are executed to perform the operation:

1. SLA (A) and SLA (B) agree to Fan-In.

2. It must be decided between the SLA's of the aggregating levels which SLA will initiate the aggregation. For this example, SLA (A) will initiate.

   In the following steps, that level which initiates the aggregation will be referred to as the first aggregating level. That level which does not initiate the aggregation will be referred to as the second aggregating level.

3. The first aggregating level performs a Fan-Out to create the new level. In our example, SLA (A) Fans-Out creating level X.

4. The first aggregating level will communicate the name of the Fanned-Out level to the second aggregating level. In our example, this operation requires the SLA (A) to communicate the name of Level X to SLA(B).

5. The SLA of the second aggregating level will communicate a name representing his level to the SLA of the new level. In our example, SLA (B) communicates the name for security level B to SLA(X).

6. The SLA of the second aggregating level initiates the creation of a dominance relationship with the Fanned-Out level by executing a *Let-Dominate* TCB function. The name of the new level is a parameter to this function. In our example, SLA (B) will execute the *Let-Dominate* function using the name of level X.

7. The SLA of the Fanned-Out level completes the creation of the dominance relationship by executing the *Dominate* TCB function. The name of the second aggregating level is a parameter to this function. In our example, SLA (X) executes the *Dominate* function using the name for security level B.

The new security level becomes operative after step 4 - the Fan-Out step. However, the aggregation is not complete until the *Dominate* and *Let-Dominate* functions are performed. By this procedure, the SLA of either the second aggregating level or the new level can veto the aggregation by refusing to perform his *Let-Dominate* or *Dominate operation.*

Forced SLA Replacement

There will be occasions in which an SLA of a level must be replaced without his or her consent. This operation is referred to as Forced SLA Replacement and requires the approval of two different authorized users. Authorized users may include a combination of the following:

- A Co-SLA, if one exists.
- The SLA of any level immediately dominated by the level of the problematic SLA.
- One of the System Security Officers (SSO's). Two SSO's should not be allowed to replace an SLA except for levels which immediately dominate the SSO's level.

To illustrate the procedure for replacing a SLA, suppose that in Figure 4a it has been determined that the SLA of level X must be replaced. To do so the following steps will be performed:

1. Two of the above authorized users are 'made aware' that SLA (X) must be replaced.
2. Each of these users independently performs the *Replace-SLA* TCB operation.
3. If there were no Co-SLA's at level X, then the SLA of the immediately dominated level will designate a new SLA for level X as a parameter to the *Replace-SLA* operation.
4. The newly designated SLA will correct any deficiencies noted at the level.

In this operation, the SLA's provide a two key control function. Either SLA can veto the operation by refusing to execute the *Replace-SLA* TCB operation.

Security Level Deletion

The capability to delete levels from the security poset is also required. A security level can not be deleted if it is dominated by any other level, hence, the SLDO for each level contains a count of the number of security levels which immediately dominate the level. This operation will require the approval of the SLA of the level to be deleted and the SLA's of all levels immediately dominated by the level to be deleted.

A security level can not be deleted if it is dominated by any other level. Once this requirement is met, the following steps may be performed to delete the level.

1. All objects existing at the level to be deleted must be removed from the level.

2. All users cleared to the level to be deleted must lose their clearance. When the SLA removes a user's clearance, the user maintains his clearances to levels immediately dominated by the SLA's level.

3. All devices cleared at the level to be deleted must have their clearances downgraded to levels immediately dominated by the level to be deleted.

4. The SLA of the level to be deleted executes the *Delete-Me* TCB operation. For example, in Figure 4a, security level X is to be deleted. After downgrading the clearance of all users at X and downgrading all devices cleared at X, SLA (X) will execute the *Delete-Me* TCB operation.

5. The SLA's of the levels immediately dominated by the level to be deleted each execute the *Delete-Up* TCB operation. In our example, SLA (A) will execute this command.

The security level is not deleted from the system until both the *Delete-Me* and *Delete-Up* functions are performed by all SLA's involved. In this way, the SLA of either the deleting level or the level being deleted can veto the deletion by refusing to perform his *Delete-Up* or *Delete-Me* operation.

VI. SUMMARY

The proposal described above provides a capability to design arbitrary security posets which can reflect the exact desire of a specific site. The delegation of a subset of the SSO's authority to Security Level Administrators allows a practical SSO interface to be designed which adheres to the principle of least privilege. The special operations for creation and deletion of new levels provide the capability for the SSO designated SLA's to customize the poset configuration to his or her needs without the need for SSO oversight. The two key control features of the design prevent potentially hostile poset modifications from occurring unchecked.

VII. REFERENCES

1. Boebert, W. E., Kain, R. Y., Young, W. D., and Hansohn, S. A., "Secure Ada Target: Issues, System Design, and Verification," Proc. 1985 Symp. on Computer Security and Privacy, 176-183, April 1985.

2. Department of Defense, "Trusted Computer Systems Evaluation Criteria," CSC-STD-001-83 August 15, 1983.

3. Denning, D. E. "A Lattice Model of Secure Information Flow," in Communications of the ACM, vol. 19, no. 5 (May 1976), pp. 236-243.

GOULD COMPUTER SYSTEMS DIVISION SECURE UNIX® PROGRAM STATUS

Gary Grossman
Gould, Inc.

ABSTRACT

Gould Computer Systems Division is committed to an intensive development program whose goal is to produce a system with UNIX® functionality that is evaluated at Class A1 by the National Computer Security Center (NCSC). To satisfy customers' needs for security in the short term, Gould is producing a graded series of secure UNIX products. The first product, called UTX/32S™ 1.0, is in formal evaluation by the NCSC as a candidate for Class C2. This paper explains Gould CSD's goals for its secure UNIX products, discusses standard UNIX in the light of the C2 criteria, describes the characteristics and features provided by UTX/32S 1.0, reviews Gould's experience in producing its first product for evaluation, and gives a preview of future products.

INTRODUCTION

Gould Computer Systems Division (CSD) is committed to an intensive development program to meet DoD and industry needs as expressed in the National Policy on Telecommunications and Automated Information Systems Security.[1] The goal of this program is to produce a system with UNIX® functionality that is evaluated at Class A1 by the National Computer Security Center (NCSC). To satisfy customers' needs for security in the short term, Gould is producing a graded series of secure UNIX products. The first product, called UTX/32S™ 1.0[2,3,4,5], is in formal evaluation by the NCSC as a candidate for Class C2. This paper discusses Gould CSD's goals for its secure UNIX products, informally evaluates standard UNIX from the point of view of security, describes the characteristics and features provided by UTX/32S 1.0, reviews Gould's experience in producing its first product for evaluation, and gives a preview of future Gould security products.

GOALS

Gould CSD has adopted four primary goals for its secure UNIX™ products:

1. The security of each product must be formally evaluated by the NCSC.

2. The security features of each product must be convenient and easy to use.

3. Each product must be as compatible as possible with UNIX standards and with other Gould CSD products.

4. Each product should maintain Gould's high standard of performance.

Formal evaluation

Gould's goal is to produce a graded series of secure UNIX products leading to a product in the early 1990's that is evaluated as Class A1 under the DoD Trusted Computer Systems Evaluation Criteria[6] (TCSEC) by the National Computer Security Center. Each product will be developed under a developmental evaluation and will be submitted for formal evaluation once development is complete.

Gould's first secure UNIX product, UTX/32S 1.0, was developed under a developmental evaluation and was submitted for formal evaluation in June 1986. Gould hopes to complete formal evaluation by the end of calendar 1986.

Gould is presently developing a system targetted at Class B1, also under developmental evaluation, and expects to submit it for formal evaluation sometime in calendar 1987.

Gould plans to continue work on systems aimed at the higher Division B classes, with an eye toward reaching A1 in the early 1990's.

User-friendliness

As with any system, the easier it is to use and administer a secure system, the more successful it is likely to be. One of Gould's goals is to make the security of the system transparent to the ordinary user who is operating at a single level. Where users must interact explicitly with the secure features of the system, the security-relevant commands will be made to look like analogous UNIX commands wherever possible.

The TCSEC define the minimum requirements for the evaluation of the security of a system. They are not a prescription for how the system is to look or what features will make the system most useful from a security point of view in any given context.

Gould is investigating integrating features into its secure systems that increase the security of terminals, as well as features that increase the power of administrators and decrease the probability of administrative error.

Compatibility

UNIX systems are coming to represent a substantial segment of the U.S. Government computer market, both within the DoD and within civilian agencies. This is primarily due to the relative ease of software portability between different versions of UNIX, even across hardware vendors. This portability, in turn, is due to the standardization of the interface to UNIX systems. To be competitive in this market place, a hardware vendor like Gould must keep track of the developing UNIX standards and ensure that its products are compliant. Gould's UTX/32 is compatible with both AT&T's System V Interface Definition and with the University of California at Berkeley's 4.3 BSD release.

Just as Gould has been tracking the UNIX standards with UTX/32, UTX/32S will remain compatible with UTX/32 to the fullest extent possible consistent with security. There will be no change in the user interface that is not dictated by compliance with the TCSEC.

When changes in the user interface are necessary, upward compatibility will be fostered by introducing changes as early as possible in the product line to permit users to make each change only once. For example, the TCB protection features that have been introduced in Gould's C2 candidate system are expected to carry over all the way to Gould's A1 system.

UTX/32S will support all of the same languages as UTX/32, as well as most applications, both UNIX commands and those supplied by third parties. Gould is encourage third parties to develop secure applications that take advantage of the UTX/32S architecture. For instance, Gould is currently investigating third-party development of secure DBMS and secure OA packages.

Gould's secure UNIX systems will be compatible with its current PN6000 and PN9000 systems, as well as systems now in development. As evidence of Gould's commitment to secure systems, memory management modifications have been made to the PN6000 and PN9000 lines. These modifications are gradually replacing the previous versions in the field.

Gould's secure UNIX systems will support the same I/O devices as its standard UNIX products.

Performance

Gould expects its secure systems to provide highly competitive performance in the secure systems marketplace.

For the most part, the performance of UTX/32S 1.0 is the same as that of standard UTX/32. One exception to this is the I/O bandwidth burden imposed by audit when many types events are enabled. There is also some additional loss due to the way that printer spooling is implemented.

The performance burden imposed by security is expected to rise moderately for Class B1 because of the introduction of mandatory access control, especially for those applications which depend heavily on repeatedly opening and closing files.

A much greater increase in this burden is expected at Class B3 because of the overhead inherent in the kernelization of security-relevant functions. Because of the high raw power of Gould's systems, Gould expects even the kernelized systems to deliver good absolute performance even if relative performance were to be cut in half.

UNIX and Class C2

Standard UNIX systems, like UTX/32, meet many of the criteria for Class C2.[7] But a brief discussion of how well UNIX meets these criteria will show that there are a number of deficiencies[8] that must be rectified.

Security policy

Discretionary Access Control. For the most part, standard UNIX meets the C2 discretionary access control (DAC) criterion. The UNIX DAC mechanisms consist of a set of permissions that controls read, write, and execute access to each file by each of

1. a distinguished user called the file's "owner" (usually the creator of the file),

2. a distinguished group (usually the group under which the creator of the file was operating when the files was created), and

3. any other user of the system.

For directories, the execute permission is interpreted as search permission. Only the owner of the file can change its DAC permissions.

But there is one distinguished user, called the *superuser*, who is not subject to the DAC controls. The superuser can access any file and can also change the DAC permissions of any file.

Object reuse. For most storage objects, UNIX conforms well to the C2 object reuse criterion. For removable media such as magnetic tapes, however, there is no mechanism for preventing any user from reading a tape that has been mounted for access by another user.

Accountability

Identification and authentication. UNIX provides for a distinct user id for each user, which is authenticated during login via a password that is stored in encrypted form. This user id is associated with every process that is executed on behalf of the user. These provisions seem to amply meet the C2 identification and authentication criterion.

But the existence of the superuser, and how it is used in practice in most UNIX systems, poses a problem. The superuser id is used to perform most system administration functions. The superuser password is therefore often known to more than one person; it is known and used by all those who fulfill the role of system administrator. This is in contradiction to the requirement that each user be uniquely identified and that the user's identity be associated with all auditable actions taken by that user.

Audit. UNIX systems provide some audit capability in the form of accounting information down to the process level. They do not provide audit for many of the types of events required by the C2 audit criterion. In particular, they do not provide a record of events that involve access or deletion of storage objects, nor do they record failed attempts at login or file access.

Assurance

Operational assurance: system architecture. Most UNIX systems loosely meet the requirements of the C2 system architecture criterion: the elements of the system are protected through a combination of hardware memory management and the DAC mechanism, and all visible resources are under control of the system. But there are significant weaknesses inherent in the size of the TCB, the existence of the "setuid" mechanism, and the use of DAC for protection.

The criterion implicitly requires that the Trusted Computing Base (TCB) — the security-relevant parts of the system — be identifiable. In a standard UNIX system, security-relevant functions are performed by a relatively large number of separate programs. Considerable analysis is required to delimit the TCB.

Standard UNIX systems provide a feature called *setuid* that is used to implement a number of UNIX features. The setuid mechanism provides a controlled means for an untrusted user to perform trusted functions. It consists of a flag that can be set on an executable file. When the file is executed, the resulting process executes with the owner and group of the executable file instead of the user id and group of the process that spawned it. This feature is useful for implementing trusted functions and subsystems such as database management systems. But this means expanding the number of programs that must be trusted. Keeping track of these trusted programs on a site-by-site basis presents a difficult administrative problem.

Each element of the TCB must be correctly protected by DAC to prevent tampering. An incorrect DAC setting on any trusted program could provide an opportunity for penetration.

Operational assurance: system integrity. This criterion is probably met, at least at the C2 level, by the standard hardware diagnostics that are available for most hardware for which UNIX systems are targetted.

Life cycle assurance: security testing. No test suite that specifically tests the security features of standard UNIX is available.

Documentation

Standard UNIX systems provide almost none of the documentation required by the criteria. This holds not only for the Security Features User's Guide, the Trusted Facility Manual, and the test documentation, but also for the design documentation on which much of the security evaluation must be based.

UTX/32S 1.0

UTX/32S™ 1.0, targetted for Class C2, is a commercially available product. It is a security-enhanced version of Gould CSD's standard UTX/32™, which provides the functions of both Berkeley and AT&T System V UNIX. UTX/32S provides enhanced system protection, audit facilities, and access control.

System Architecture

The principal differences between UTX/32S 1.0 and standard UNIX lie in the architectural features that protect the UTX/32S TCB[9].

Restricted Environments. The UNIX file system employs a hierarchical directory structure (see Figure 1). A file's *pathname* ("/usr/joe/f1" in the figure) consists of the concatenated names of the components on the path to the file from the root of the file hierarchy (denoted "/"), separated by "/"s. User processes can address any part of the file system, although they can be prevented from unauthorized access by the UNIX file protection mechanisms.

Since the standard UNIX file protection consists only of DAC, it is possible for any important element of the system to be made vulnerable to tampering if its DAC is improperly set. For instance, the file from which the UNIX kernel is initialized is usually kept in the root directory as "/unix"; if its protection were improperly set, a user program could alter the kernel program at will.

A stronger protection mechanism is desirable for any system that is to provide security. UTX/32S provides such a mechanism, called *restricted environments*.

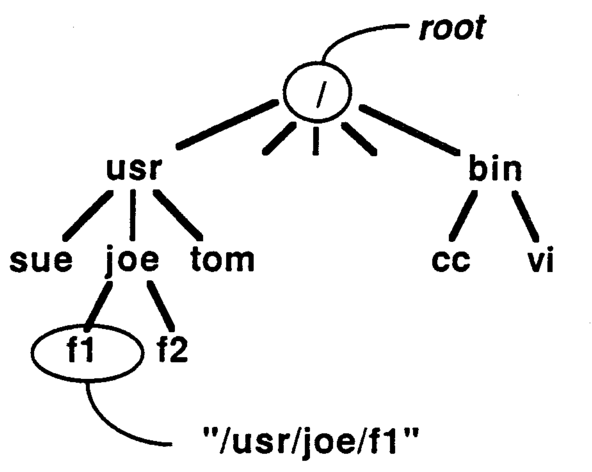


Figure 1. The UNIX File System

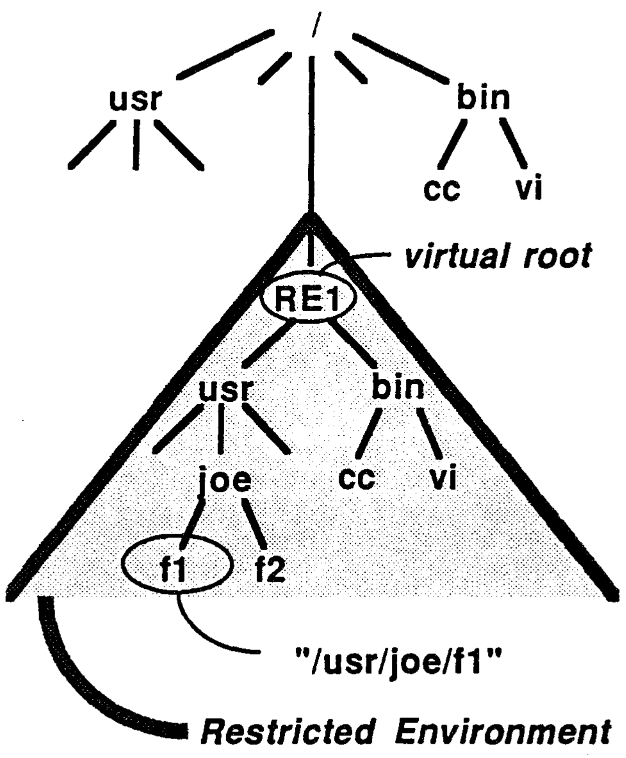


Figure 2. A Restricted Environment

A restricted environment (RE) is a subtree of the file system to which a process and all of its descendants is confined (see Figure 2). For the confined process, the base node of the subtree becomes the virtual root of the file system. Files outside the subtree cannot be addressed by the process. Because of the importance of the file system in UNIX, the RE mechanism is an important component of the protection of the elements of the UTX/32S TCB.

When a user logs into UTX/32S, the user's command interpreter process ("shell" in UNIX terminology) is confined to a restricted environment. Since every action the system takes on behalf of the user, except for those performed by the TCB, is performed by the user's shell process or one of its descendants, all user actions, except for those performed by the TCB, are performed in the context of the RE. Thus no user action can directly access any element of the TCB.

A process confined to an RE cannot exercise superuser privilege, even if it were able to somehow become labelled with the superuser id.

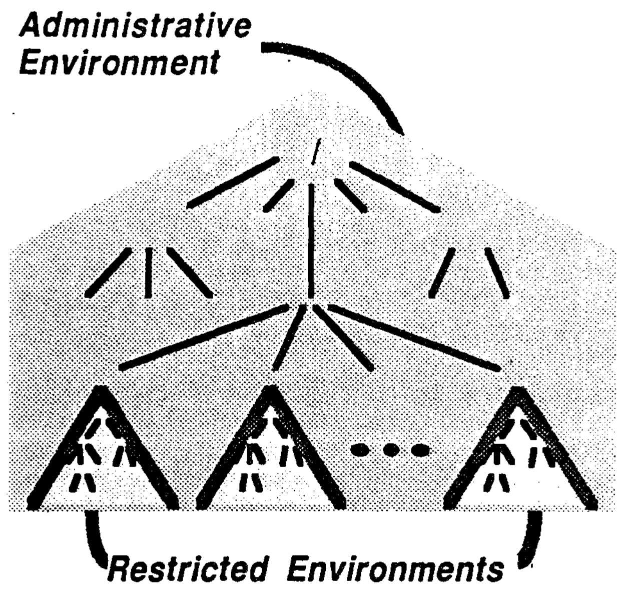


Figure 3. The Administrative Environment

Administrative Environment. The file system outside the REs is protected from access by any user process. It is there that all trusted elements of the system reside, and where administrative actions are performed. This part of the file system is called the Administrative Environment. (See Figure 3.)

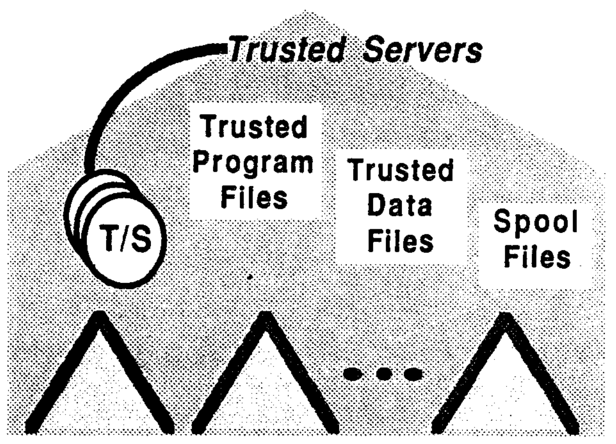


Figure 4. The UTX/32S TCB

The UTX/32S TCB. The UTX/32S TCB consists of the kernel and a set of trusted server processes together with a corresponding set of program files, a set of data files (such as the password file), and the system spool files. (See Figure 4.)

The process images of the kernel and the trusted servers are protected by the memory protection hardware. The other elements of the TCB are protected by the RE mechanism.

Trusted Servers and Secure Sockets. The non-kernel active elements of the TCB are a set of trusted processes called *trusted servers*. The trusted servers perform all privileged operations in the system that are not performed by the kernel. Each trusted server is spawned at system initialization and is responsible for a specific set of operations. For instance, there are trusted servers that perform logins, printer spooling, mail, and device allocation.

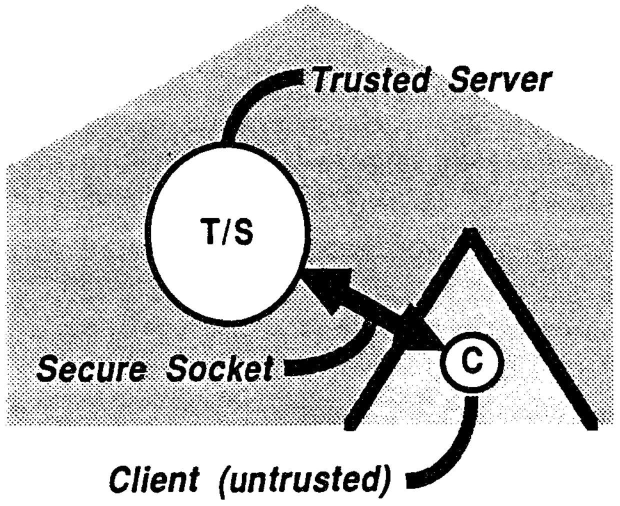


Figure 5. Trusted Server

A user process ("client") that requires action from a trusted server requests the action via a secure communication path called a *secure socket*. (see Figure 5). When the client initiates the secure socket, the kernel guarantees that the client is connected to the specified trusted server. When the server receives the connection, the kernel provides to it information about the client, including user, group, process id, and RE id. (In Division B and A systems, the process' Mandatory Access Control label will also be supplied.) The kernel guarantees the correctness of this information, which is used by the server to make authorization decisions and to properly label objects and output on behalf of the user.

System Administration. There are two steps by which an authorized user gains the access and the privileges necessary to perform system administration functions. First the user must log in to the administrative environment rather than into an RE. Then the user must specifically requrest superuser privilege.

During the login sequence, a user signals the authorization server of intent to log in to the administrative environment by preceding his/her user name with the special character "*". If the user completes the login sequence properly, the authorization server checks the authorization data base to determine whether the user is an authorized system administrator, and completes the login successfully only if so. Once the user is logged in to the administrative environment, he/she is subject to the standard DAC rules for files within the administrative environment unless he/she specifically requests superuser privileges.

A user requests superuser privileges via an audited command that creates a new command interpreter process with superuser privilege. This process and its descendants are all still tagged with the user's id, and all auditable events requested by the user are so tagged.

No user can log in to UTX/32S as the superuser.

Protected files and directories. UTX/32S provides a further mechanism for protecting against Trojan Horses within RE's. Any directory or file that is owned by the "superuser" cannot be modified, deleted, renamed, or its DAC changed, by non-privileged users, regardless of the DAC setting on the directory or file. Thus compilers, subprogram libraries, and system commands can be installed in an RE that is shared by many users, and be secure from tampering. This mechanism is not used to protect elements of the TCB.

Device control. In standard UNIX, access to devices such as terminals and tape drives is provided via objects in the file system called *special files*. When a process opens a special file, all I/O requests are handled by a corresponding device driver instead of the disk file system. For system maintenance, even the disks that contain the file system are represented via special files (see Figure 6). As with other elements of the file system, access to devices is controlled only by DAC. An incorrect DAC setting on a special file could lead to unathorized access to the entire file system.

In UTX/32S, each device is allocated to one of three categories: privileged, owned, or discretionary.

Privileged devices are accessible only by elements of the TCB. The system disks are privileged devices. They can be accessed only via the file mechanisms in the kernel. The line printer is also a privileged device. It can be accessed only by the printer spooling mechanisms of the TCB.

Owned devices are allocated by a trusted server to at most one user at a time. Only processes initiated by that user can access the device. Terminals and tape drives are owned devices.

Discretionary devices are protected only by the DAC and RE mechanisms. There are no standard UTX/32S devices in this category; it is provided for specialized devices.

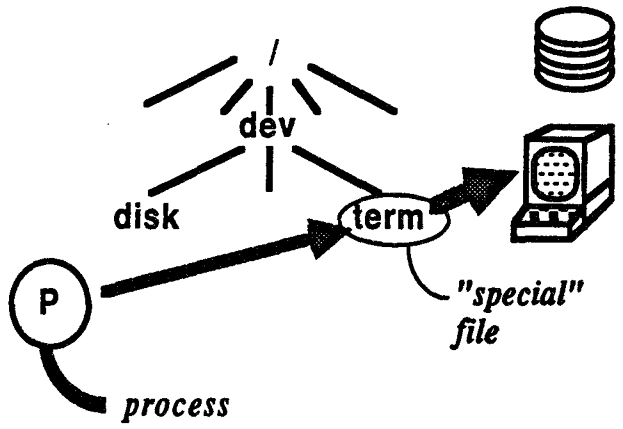


Figure 6. UNIX Device I/O

Multiple file system links. In the UNIX file system, a *link* is an entry in a directory that refers to a file or a subdirectory directly contained within the directory. Typically, there is only one link to each file or directory, and the file system is thought of as a tree or a strict hierarchy. But it is possible for there to be more than one link to a given file; these links can be created by any user who has the appropriate access to the file and the directory involved. It is even possible in some versions of Berkeley UNIX for there to be more than one link to a given directory, although the creation of these links is restricted to the superuser and the documentation cautions against it. These properties make the UNIX file system a directed acyclic graph rather than a tree. Figure 7 illustrates a situation in which directory D4 is a subdirectory of both D1 and D3, and D6 is a subdirectory of D1 and D4. The figure also shows file f7 in both D2 and D5, and similar situations for files f9 and f11. (The way that file links work is somewhat simplified in this example.)

One result of the possibility of multiple links is that some files may have many pathnames. f11, for instance, has pathnames "/D1/D4/D8/f11", "/D3/D4/D8/f11", and "/D3/D5/f11". One implication of this for secure systems is that the TCSEC requires recording "the name of the object" in audit records at Class C2 and above.[10] Presumably, ambiguous names are not acceptable. The ambiguity of multiple pathnames could be circumvented by also including some unique identifier for each object in the audit record. In UNIX, each file has a unique data structure, called an *inode*, that describes the file's physical storage. Each inode is identified by a distinct integer. UTX/32S includes the inode identifier in the audit record to disambiguate multiple pathnames.

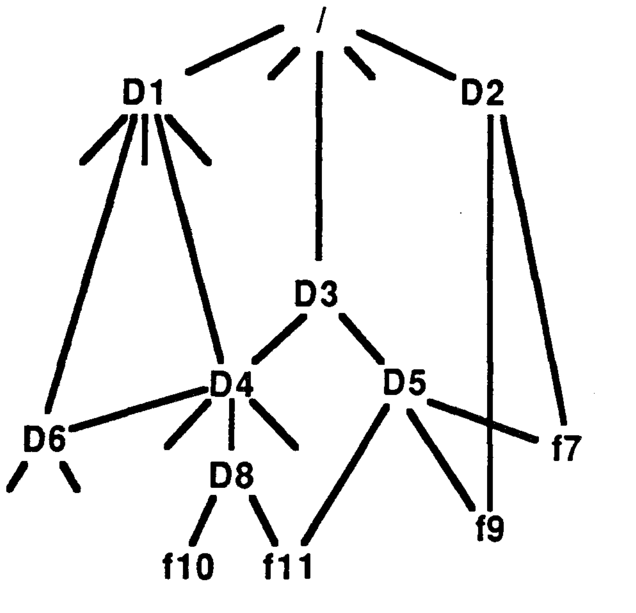


Figure 7. Multiple File System Links in UNIX

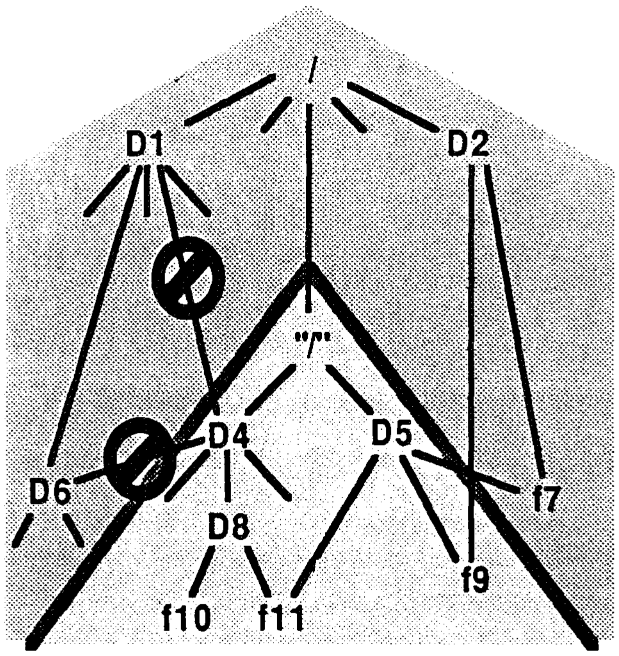


Figure 8. Multiple File System Links in UTX/32S

More important for UTX/32S, the RE mechanism depends for its confinement properties on a strict hierarchy of directories (not necessarily of files). This is because the UNIX pathname rules include an operator ("..") that permits processes to address files above a given directory. Thus, in Figure 8, if the links between D1 and D4 or D4 and D6 were permitted, a process in the RE that had the appropriate DAC permissions could address D1 and the real root (via a pathname such as "/D4/D6/../.."), and thus ultimately the entire file system. This would obviate the RE mechanism. For this reason, multiple links to directories are not permitted in UTX/32S. (There is another UTX/32S mechanism, derived from Berkeley UNIX, called *symbolic links*, which is simply an indirect pathname mechanism; it has none of the above properties, and can be used between directories at will.)

Using multiple links to files does not pose the same problems as it does for directories, and so it is permitted both within an RE and between the administrative environment and an RE. But, as always, system administrators must exercise care not to base any security-relevant decisions on data that is contained in an object to which any untrusted user has had write access.

"setuid" eliminated. In addition to the administrative burden represented by the setuid mechanism (see **Operational assurance: system architecture** above), it permits an untrusted user to execute trusted software in an environment that has been defined by the user. This presents opportunities for the user to "spoof" the trusted software.

A further objection to setuid (and setgid — an analogous mechanism for groups), is based on Gould's goals of upward compatibility and of introducing changes, when necessary, as early as possible. The Class A1 criteria require, under Design Specification and Verification[11], a formal model of security policy and a formal top level specification; the model must be proven consistent with its axioms and the specification must be shown to be consistent with the model. Class B3[12] — and even Class B2[13] — have weaker versions of this requirement. In a system with many processes per user that are dynamically created and destroyed, it would be useful to be able to make a simple universal statement about the inheritance of privileges from a process to its descendants, in order to be able to meet these requirements.

Without the setuid feature, it is possible to state that once a process' privileges have been established, that process' privileges and those of its descendants monotonically decrease. The setuid feature makes it impossible to make this assertion.

For these reasons, setuid is not a part of UTX/32S 1.0. The system features that depend on it have been reimplemented using the trusted server mechanism.

/dev/kmem, /dev/mem. UNIX provides two special files for debugging and to implement certain utilities. "/dev/kmem" provides the ability to read and write in the kernel's virtual address space. "/dev/mem" provides the ability to read and write in the physical address space of the machine. These capabilities are dangerous, to say the least, in a secure system. They have been changed to provide read access only, only to superusers in the administrative environment. They will be eliminated in subsequent versions of the system.

Discretionary Access Control

Groups. In System V UNIX, a user operates in one group at a time, DAC decisions are made accordingly, and all files are created with the user's current group. Berkeley UNIX permits a user to operate in multiple groups simultaneously, with frequent confusion as to which group newly created files will be labelled. UTX/32S uses the single group mechanism.

Public DAC access. In UNIX, each user has a default DAC setting that is applied to newly created files, called the *umask*. In standard UNIX, the initial setting of each user's umask permits the creation of files with automatic public read and execute access. In UTX/32S, it is not possible to set the umask to permit any automatic public access. Public access for each file must be set by explicit user action.

Audit

Audit trail file. UTX/32S provides an audit trail of security-relevant events. The audit trail can be written only by the UTX/32S kernel. Records are written to the audit trail file as a result of security-relevant system calls (such as file creation, file open, etc.) by both trusted and untrusted processes. Trusted processes can also directly request the kernel to write audit records based on events that are visible only to the non-kernel parts of the TCB (such as logins). (See Figure 9.)

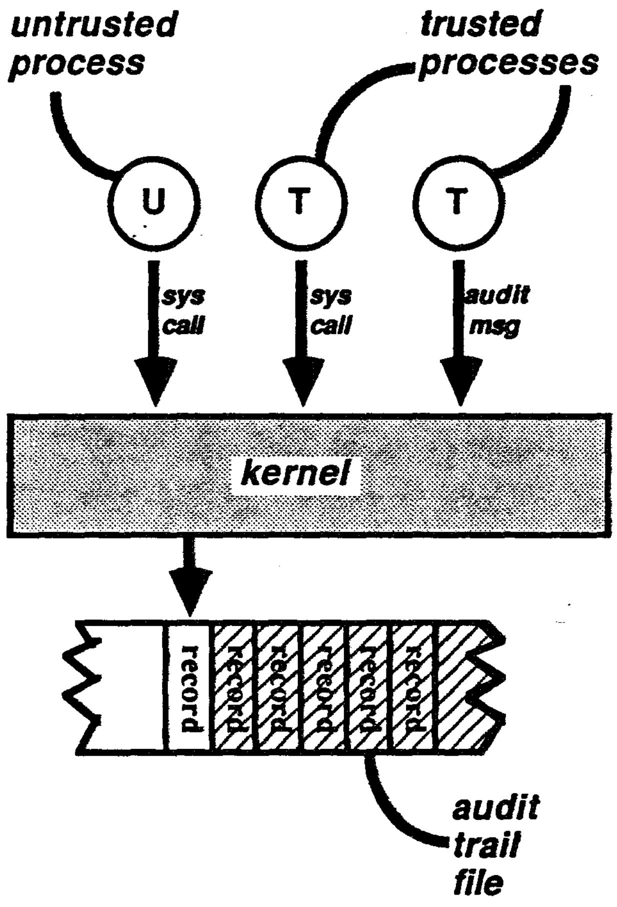


Figure 9. Audit Trail

Each audit record contains a time stamp, user and group id's, process id, and controlling terminal, as well as information determined by the type of event, such as file pathname for file open.

Events audited.

Table 1 gives a synopsis of the types of events that are recorded in the UTX/32S audit trail file. As the table indicates, some events are optional; these can be turned on and off at will by the system administrator.

Event type	Optional?
Reboot system	No
System version and configuration	Yes
Changes to user/group definitions	No
Stop/change audit collection	No
Access to audit trail file	Yes
Start/stop/change accounting	No
Successful logins	No
Unsuccessful logins	Yes
Suppress printer labelling	Yes
Change mail source designation	Yes
Process creation/termination	No
Make process superuser	No
Change process user id	No
Change process working directory	No
File/device access failure	Yes
Create/open file	Yes
Create/delete file link	Yes
Create/remove directory	Yes
Create special file	No
Change file owner/access permissions	Yes
Change device owner	No
Rename file/directory/special file	Yes
Mount/unmount file system	No

Table 1. Auditable Events in UTX/32S

Audit administrative operations.

The system administrator can

1. turn audit collection on and off

2. direct the audit trail to any file or medium

3. select the types of events to be audited

EVALUATION EXPERIENCE

UTX/32S™ 1.0 was developed under a developmental evaluation by the the NCSC. The NCSC appointed a developmental evaluation team, which met with project personnel at the early stages of the project and several times thereafter. Some weeks prior to each meeting, project personnel supplied the team with documentation that reflected the state of the project's design and implementation. Each meeting consisted of presentations by project personnel followed by a discussion period.

Any development group must approach ongoing oversight by an outside agency with some anxiety. The opportunities for substantial communication overhead and project design backtracking pose real dangers for project schedules and budgets.

These anxieties were not borne out. Instead, the effect of the evaluation was almost entirely beneficial. As one might expect, the evaluation team was helpful in interpreting the criteria and in evaluating potential approaches to meeting them. Further, the prospect of preparing for periodic review by a competent team of outsiders who were to evaluate it according to its primary goal — enhanced security that meets the criteria — served to focus the project efforts.

Even a major change of team personnel (two of the four team members were replaced approaximately halfway through the effort) had little negative effect. Continuity was provided by the remaining team members with little observable burden on development.

UTX/32S 1.0 was officially accepted for formal evaluation in June 1986.

THE FUTURE

UTX/32S™ 1.0 will be succeeded by a system targetted at Class B1. It will have all of the features of its predecessor, plus mandatory access control, discretionary access control based on access control lists, and further enhancements to system protection. Development has begun on this product, which is expected to be available in calendar 1987.

Gould also plans to continue work on higher B level systems with an eye toward reaching A1 in the early 1990's.

ACKNOWLEDGEMENTS

The author can take no credit for the work described in this paper, although he took part at several stages in the project review process. The work was performed by the Gould CSD UTX/32S 1.0 development staff under Andreas Schuelke, Project Manager, and Steve Sutton, Manager, Secure Systems Development.

---

1 The White House, National Policy on Telecommunications and Automated Information Systems Security, National Security Decision Directive 145 (Unclassified Version), 17 September 1984.

2 Using Gould UTX/32S, Release 1.0, Gould, Inc., Computer Systems Division, Ft. Lauderdale, March 1986.

3 Product Definition for Gould UTX/32S, Release 1.0, Gould, Inc., Computer Systems Division, Ft. Lauderdale, March 1986.

4 System Administrator's Guide for Gould UTX/32S, Release 1.0, Gould, Inc., Computer Systems Division, Ft. Lauderdale, March 1986.

5 System Administrator's Guide for Gould UTX/32S, Release 1.0, Gould, Inc., Computer Systems Division, Ft. Lauderdale, March 1986.

6 National Computer Security Center, Department of Defense Trusted Computer System Evaluation Criteria, CSC-STD-001-83, 15 August 1983.

7 Ibid, pp.15-17.

8 Grossman, G. "How Secure is 'Secure'?", UNIX Review, August 1986, pp.50-63.

9 Miller, G., Sutton, S., Matthews, M., Yip, J., and Thomas, T., "Integrity Mechanisms in a Secure UNIX: Gould UTX/32S", accepted for presentation at the Aerospace Computer Security Conference, McLean, VA, 3-4 December 1986.

10 National Computer Security Center, p.16.

11 Ibid, p. 48.

12 Ibid, p. 39.

13 Ibid, p. 31.

® UNIX is a registered trademark of AT&T.

™ UTX/32 and UTX/32S are trademarks of Gould, Inc., Computer Systems Division.