We Need to Think About the Foundations of Computer Security

Marvin Schaefer
CTA Incorporated
6116 Executive Blvd., Suite 800
Rockville, Maryland 20852

1 By Way of Apology

It has been said that much of computer security (information security) has been reduced to practice. While this is true, it is important to understand that there are no general closed form solutions to computer security problems. In particular, it has been my experience that what may be a solution for one particular set of circumstances does not necessarily address the requirements of a related, but subtly different, set of circumstances. Often, the difficulty manifests itself at a point where two trusted systems need to communicate with each other, e.g., where each trusted computing base needs to make assumptions about decisions or mediations performed by the other. In other cases, the identified difficulties have come about because of a fundamental misunderstanding about the properties of the composition of the access control policies resulting from the interactions of two trusted systems.

The open literature abounds with examples of generic security flaws of systems that were not designed to implement a stated computer security policy (or access control policy). These flaws generally involve misuse of privilege, inadequate or incomplete parameter checking, circular functional dependencies, lack of data hiding or modularity, etc. While the problems of concern in this paper involve variations on such flaws, something more significant appears to be involved in the downfalls of the systems that were designed to be secure. Also, just as there is no generally applicable technique that applies to securing arbitrary system environments, it is not clear that there is a general characterisation of what tends to go wrong with these latter systems. This note provides examples of a few such problems by drawing anecdotally from my experiences over the years with providing assurance while attempting to solve trusted system design, development, and certification problems. A preliminary draft of the present note, revisiting the 1989 symbol security paper,[0] was presented at the Foundations of Computer Security Workshop in Franconia this summer.

In 1985, my NCSC colleague D. Elliott Bell and I wrote a paper, as did also John Rushby, on why we believed trusted networks were really just, at least in an abstract mathematical sense, a special class of trusted systems. We advocated against the writing of a separate Trusted Network Criteria[1], contrary to the expressed feeling of most serious network security practitioners at the time. The resulting Trusted Network Interpretation[2] elaborated on TCSEC principles, addressed new concepts—especially, the composition problem—while establishing broadened interpretations of existing TCSEC wording to support approaches to providing a class of integrity and assured service requirements to eliminate this recognised Orange Book "deficiency."

In 1989, my confreres W. Curtis Barker and Charles P. Pfleeger, with participation by Frank L. Mayer and Lynne Vidmar, identified a large class of fundamental flaws in the TNI s oversimplified composition and partitioning principles. The example distributed system studied at the time could be argued to fully satisfy an interpretation of the class A1 TNI requirements. Yet after the paper design was completed (and approved by the sponsor), further analysis found that the design succumbed to direct attacks that would fully compromise confidentiality, integrity and assured service![3] Although our example interpretation had been found to lack the desired security properties, it was a "reasonable-looking" interpretation and architecture that initially gave confidence. Only after analysis was it found to be "obviously" dependent on certain properties of what turned out to be a naive integration of cryptography with trusted systems principles. To date, no means of eliminating such vexing counterexamples has been identified to us, nor are we aware of a generalisation of our post-analysis techniques that would identify similar flaws in somewhat different network architectures.

In late 1986, David Bell and I identified a serious deficiency in the Woods Hole report s advocated "spraypaint" approach[4] to using an untrusted DBMS to retrieve multilevel data from a sealed database. Our discovery was that no matter what the nature of the trusted front-end guard or filter, a conspiracy of untrusted subjects could reliably transmit all of the high-level data in the database to a lower sensitivity level. We refined this result to recognise that such problems exist for a large class of front-end guard applications.

In 1987, David and I published on our discovery that we knew we did not know how to implement a trusted path between the user of a standard workstation (i.e., a PC) and a trusted host. We also found that the problem becomes much more difficult if one communicates with a distant host over a network, say while using Kermit or another network communications protocol. We even found no easy answer to the question of how to implement a "secure attention key" as required by the TCSEC. We, along with Martha Branstad, chaired a panel on this problem at an Oakland conference, but the panel only brought on a sense of futility and malaise to the participants.

In 1988, my colleagues Barbara A. B. Mayer and David Bell asked me to write a guideline on the formal specification and verification requirements for B3 and A1 systems. This proved to be a very difficult task for me. I was pretty sure I knew how to provide evaluators with the documentation required by the TCSEC. But I was very sure I didn t know how to claim that such documentation would lead to the design, implementation, or conclusion that such a system was secure in any real sense.

This was partly because of an exercise I directed at the NCSC in which a candidate A1 distributed message processing system rather dramatically failed to provide its specified and verified strong access controls. The most important of the demonstrated counterexamples did not require employing sophisticated covert signalling channels since it was possible for any interloper to circumvent the alleged TCB (ATCB), thereby rendering it and all of its costly accompanying assurance evidence profoundly irrelevant! Admittedly, the formal evidence was dauntingly impressive and, not to impugn the reputations or capabilities of the team that produced it, correct as far as it went! The more serious problems evidently included the fact that:

- the formal model, an adaptation of the Bell-LaPadula Model, was not adequately (or naturally) adapted to reflect the application;
- the formal specification was written and processed with "verification tools," by a group of formalists who, as viewed by the implementors, had their own agenda independent of the implementation;
- the formal documentation and evidence were not consulted by the implementors;
- the implementation was not consulted by the formalists; and
- the distributed processing nature of the application was not addressed by the formal analysis (this latter property was not relevant to finding or demonstrating the principal flaws or vulnerabilities, however).

My faith had further been shaken by a series of profound communications from John McLean that began while I was serving at the NCSC. John s questions were not at all comfortable, particularly since the answers one would have liked to give were far from being printable in a [family-oriented] formal mathematics journal. John, along with many others, had made it clear to me that I did not know what we really meant or should have meant either by requiring either a "formal inductively" secure model or by one that was proven to be consistent with its axioms. An evaluated A1 product had been demonstrated to have a security flaw by another group of investigators. To make matters worse, new panic had set in over the potential speed of timing channels and the shocking discovery of highly efficient and unauditable designer channels.

So I wrote a report that proved to be disappointing to some of the sponsors. A modified version of that paper was published at the 1989 IEEE Symposium on Security and Privacy as "Symbol Security Condition Considered Harmful" to emphasise my concern that too much attention was being paid to the manipulation of symbols and too little attention was being focused on the real requirements and actualised properties of the implementation and its platform.

In the period 1988 92, I was engaged in research on trust properties of hardware with Terry C. V. Benzel, Jaisook R. Landauer, Charles Pfleeger, Fred Pollack, the late Christian Jahl, and several other colleagues. In the conduct of these studies, it was discovered that many classes of hardware include full-blown operating systems within certain of their components.[5] Among these components is an important module that provides the capability for a maintenance engineer, e.g.,[6] to step through the hardware microinstruction steps and, as need be, to change global hardware state between these microsteps!

Since 1989, James P. Anderson has advised and published that the covert channel problem is now so severe, particularly due to the discovery of "Designer Channels," that truly secure installations will not permit the introduction of untrusted, unknown, applications code to be run by the users. In the summer of 1993, Robert T. Morris expressed a new value for the fastest acceptable covert channel for certain applications: 10^{-3} bits per second!

This year, my colleagues James Freeman, Richard Neely, Max Heckard and I discovered that a large class of hardware capabilities can be used as the basis for high-speed covert channels. This discovery lends support to an intuitive position taken in the 1980s by Norm Hardy and Susan A. Rajunas that to be safe, a capability-based high-assurance trusted operating system would need to implement inscrutable capabilities!

2 Security Defined?

Since the symbol security paper s publication, everything I have seen indicates that the computer security[7] problem has become much richer and far more difficult. Nothing has gotten simpler: our knowledge about the nature of the problem has increased, and the problem appears to be advancing rapidly ahead of our ability to provide high-assurance solutions—if, indeed, what we provide can justly be called a solution.

The TCSEC and the work leading to it gave recognition to the fact that security is not a binary property of a system. Not only does the TCSEC focus primarily on just the confidentiality aspects of the general security problem, but it also produces characterisations of seven combinations of features and assurances that are asserted (i.e., without citing evidence or proof) to provide increasing levels of control to prevent unauthorised disclosure of information. The TCSEC does not provide specifications for secure systems, but instead provides them for trusted systems, a distinc- tion made to avoid the need to impose specific requirements on the process- ing environment. The TCSEC does not impose requirements that a trusted system make stored or protected information available to authorised users; the TCSEC does not impose requirements that authorised users or subjects perform correct/acceptable updates or modifications to the objects they are allowed to change. Only in classes B2, B3 and A1 is it assumed that a user or subject might attempt to violate the access control policy by means of a concerted and planned attack. Nothing in the express TCSEC requirements, even for these high-assurance classes, addresses the threats to users posed by the popular classes of virus or worm. The TCSEC is essentially mute with respect to the topic of trusted subjects and, hence, does not provide any form of strong requirements on the nature or operation of those bodies of code that can violate the rules of information-flow confinement that are to be imposed on all untrusted subjects. So those subjects that are permitted to change the classifications of sensitive data can be specified and implemented pretty much ad libitum.

This leads me to wonder whether computer security is becoming a field like the abstract mathematics of the early XXth century. Bertrand Russell once wrote that

> Pure mathematics consists entirely of such asseverations as that, if such and such a proposition is true of anything, then such another proposition is true of that thing. It is essential not to discuss whether the first proposition is really true and not to mention what the anything is of which it is supposed to be true ... if our hypothesis is about anything and not about some one or more particular things, then our deductions constitute mathematics. Thus mathematics may be defined as the subject in which we never know what we are talking about, nor whether what we are saying is true.

It is counterintuitive to the expectations of its users; because the TCSEC does not impose requirements in the areas of assured service, availability, integrity, performance efficiency, robustness, correctness, etc., and because of the duration, expense and extent of the evaluation process, many evaluated products have been avoided by the user community and at least one has been called a "secure brick." We have recently heard that one branch of the DoD has decided not to use a certified A1 network security product because it is too expensive to maintain during its remaining life cycle. Others have begun abandoning high-assurance evaluated systems and products because they are somewhat obsolete (though they do appeal to the antiquari- an community!) and lack the now obligatory snazzy and snappy window, multimedia object and GUI features, etc.

3 Expanding Universe

No one promised us a rose garden, but we got a bramble of new thorny problems anyway. Here are a few that I care about:

- Multilevel DBMS: System-high database management (e.g., that based on the Hinke Schaefer or SeaView models) is not useful in most real life applications because people need to be able to affect (i.e., modify, change, delete, augment) relatively unsen- sitive portions of a database as a consequence of insights that derive from higher-sensitivity inputs. This problem may not be tractable—no evidence suggests that it is—but users will work around the system if it is too restrictive with respect to their operational requirements. In 1982, Bob Morris told me that 17.3 is a Top Secret number (though he didn t tell me why). I have come to realise that sometimes 17.3 is not classified at all, but in those cases where 17.3 is classified, it does no good to pretend it isn t, or to believe that if one uses a trusted subject to modify an unclassified tuple with 17.0 that one has either concealed the truth or that one has placed the system in a state that won t eventually lead to a more serious compromise than unauthorised disclosure.

- Asynchronous Transactions (long duration): Technology in specification and analysis has advanced to accommodate macro-level events (e.g., database transactions, Clark Wilson transactions, etc.) in multilevel contexts. Integrity and serialisability results are beginning to be assimilated into the accepted technology in DBMS contexts, but has also to be integrated into the C3I contexts of general purpose distributed processing contexts.

- Maintained Hardware: Computer processor technology is progressing at an incredible rate. Microprocessor chips appear to have a half-life of about six months now before they are replaced by something more powerful. Things attached to a processor bus appear to be appearing with even more unpredictable regularity. All of these components have the capability of interfering (often and rapidly) with the actions of other components, in many cases at the level of interrupting microinstruction execution sequences. Formal expressiveness of specification languages does not presently address the temporal issues of such interactions, and thus makes it very difficult to perform analysis on those critical interactions that need to appear logically as atomic events. Unlike the long duration asynchronous multilevel transactions described above, well-behavedness requirements need to be formulated and expressed for the atomic steps in the class of security-relevant transforms that typically show up in a well-formed FTLS.

- Total Processing Environment Considerations: A real need exists to address the issues of the processing environment. Recently, it has become evident that people talking on cellular phones have interfered with the computerised landing systems of commercial aircraft, and one can envision similar mishaps with airborne multilevel systems. How or where in the specification and analysis of trusted systems do considerations of the environment get formal, rather than ad-hoc, consideration?

- Real-Time Multilevel Requirements: Flight controllers on certain military aircraft are increasingly integrated into LANs that include weapons controllers, inputs from intelligence sensors, communications with unclassified telemetry or control stations, etc. Pilot and platform safety are considered to be important, and hence priority to meeting certain avionics standards is a dominant issue. Since denial of service may only be a one-time event, there is a need to produce accurate maintenance logs that include records of all faults encountered during a mission (since these must be read by personnel not necessarily cleared to system high, the potential for covert storage channels is a real consideration). It is unclear, under these circumstances, that a TCSEC style audit log should ever record an access-denied event, since such denial could lead to an unacceptable loss of life and probably represents an error condition—though it is unclear to me that one could prove this other than from a candidian (panglossian?) axiomitisation of the problem space. Clearly, the real-life consequences of access control (or the lack thereof) in such life-critical systems needs to be studied.

- Object-Oriented Architectures: Well-defined interfaces have been epitomised in such architectures by the use of highly-specialised message-based parameter passing conventions. Regardless of whether the Smalltalk, RPC, or some other protocol is employed for communication between objects and their methods, a potential exists for a tremendous amount of information to be passed between objects. This makes it extremely difficult to securely implement the BLP read-down unless one uses trusted subjects—whatever that means—since all messages from higher levels may still, even while satisfying specified syntactic and semantic properties, contain an illicit encoding of sensitive information that ought not be visible to lower levels. Because of the object-oriented architectures friendliness to the introduction of multimedia objects into existing architectures, some objects are so large that (a) they need to be passed by pointer rather than by value (leading to shared multilevel address spaces) and, (b) because of the nature of certain specialised objects, many objects include provision for their own storage manager (or external pager), which can lead to classical time-of-check-to-time-of-use validation problems. As a consequence, the very notion of multilevel interprocess communication in multilevel object-oriented environments needs to be re-addressed.

- Other Life Cycle Maintenance Issues: Actually this is the incremental change and system composition problem wrapped together. Presently, the most common first improvement on TCSEC-evaluated products is the introduction of network protocols. While this seemingly-minor modification to existing evaluated products immediately introduces potentially complex TNI issues, it appears that this is only the tip of the hackneyed iceberg. Every time a new object is added to an object-oriented architecture, it appears that the TDI would be correct in suggesting that a complete re-evaluation of the TCB would be required as a matter of course. However, since each new object is essentially a policy regime in its own right, it would appear that the composition problem needs to be addressed within as well as between major systems and their components as part of the anticipated life-cycle of any product.
- Classical Vulnerabilities: Only for completeness I include reference to Jim Anderson s challenge as to whether the application of formal methods will ever be sufficient to discover any of the traditional classification vulnerabilities in a constructed system implementation.

4 Whither?

I don t think this all has to be bleak. I would like to believe that the integration of formal methods into the life cycle development process will eventually be utile and effective in providing more useful secure products and systems.

There is a slim ray of hope. Jaisook R. Landauer has commented that B3 systems are more efficient than A1 systems have to be in order to satisfy their verification requirements (actually, to satisfy the requirements imposed by the state of most available formal specification analysis tools) and she has credibly claimed that it is possible to verify code with pointers so that a verified implementation could be produced that would be no less efficient than a B3 system! Others are claiming that they have verified properties of implementations that have not been within the state of the verification are in the past. If her intuition is correct, then perhaps formal methods could be brought to bear on such problems as:

- examining the code in very large operating system TCBs
- validating layers and suites of protocols
- validating the correct functionality of computer hardware-based internal operating systems
- validating the absence of large classes of embedded malicious code families in applications

It is clear to me, though, that this journey of thousands of proof steps needs to be nourished with hundreds of research grants. The focus of such research must no longer restricted to just limited information-flow analysis specification verification, but must be expanded to include the issues of real-time, distributed, asynchronous system implementation-level analysis.

This paper does not leave me with a good feeling about the future. So at the suggestion of a friend, here are a few problems which, if solved, should help with finding solutions to providing assured information security in the future:

- Define the implementation semantics of what the B3 trusted path should have been and demonstrate that trusted applications are capable of communicating with both the TCB and with the user such that no party is spoofed;
- Find highly assured means of integrating cryptologic principles into trusted systems as means of assuring the unspoofable vetting and use of specific applications programs and of critical data;
- Find a formal means of differentiating between the behaviour of legitimate programs that must produce or modify non-user files and that of classes of illegitimate, malicious programs that imitate them;

- Achieve a formal basis for analysing trust properties (especially that of data flow) in object-oriented and message-passing systems architectures; and, of import perhaps only to me,

- Produce a convincing argument that the application of formal methods will prohibit a reasonable subclass of the compendium of classical system penetration attacks

Notes

[0]. M. Schaefer, "Symbol Security Condition Considered Harmful," Proceedings, IEEE Symposium on Security and Privacy, Oakland, 1989.

[1]. To become known as the Puce Book.

[2]. Bound, to the contrary, in Red covers.

[3]. Unfortunately, we are not at liberty to publish the counterexample in the open literature.

[4]. The application of cryptographic sealing is described in Chapter 1 of "Multilevel Data Management Security," Air Force Studies Board, National Academy of Sciences, 1982.

[5]. These operating systems are to be distinguished from the operating system with which the user traditionally interacts. The reader may suspect that I am exaggerating, and that these "operating systems" are simply event-driven schedulers for, e.g., a portion of the I/O subsystem. Would that that were the case! The reality of the situation is that in at least one case, the "hidden" operating system came complete with editors, compilers, etc., with a few computer games thrown in for good measure.

[6]. The question comes immediately to mind of whether a subject acting on behalf of the maintenance engineer could acquire this capability!

[7]. Many have recast the problem as the information security problem, and openly address the fact that networks, availability and integrity are ever present in addition to the older concerns of enforcing a form of confidentiality on a monolithic mainframe architecture. My present feeling is that all of the problems of networks are present in the inner cosmos of most computer systems. For only that reason do I continue to pretend to having a justification for using the word computer rather than information with security. I am less optimistic to finding a solution to the information security problem, on one or more connected computers, than finding one to the data security problem; my differentiation is along the lines that information denotes the semantic content of data , the latter being only an uninterpreted ordered string of bits.