TOWARD A COMPREHENSIVE INFOSEC CERTIFICATION METHODOLOGY *

Charles N. Payne, Judith N. Froscher and Carl E. Landwehr
Center for High Assurance Computing Systems
Naval Research Laboratory
Washington, D.C. 20375–5337

Abstract

Accreditors want to know what vulnerabilities will exist if they decide to turn on a *system*. TCSEC evaluations address *products*, not systems. Not only the hardware and software of a system are of concern; the accreditor needs to view these components in relation to the environment in which they operate and in relation to the system's mission and the threats to it. This paper proposes an informal but comprehensive certification approach that can provide the accreditor with the necessary information. First, we discuss the identification of *assumptions* and *assertions* that reflect system INFOSEC requirements. Second, we propose the definition of an *assurance strategy* to integrate security engineering and system engineering. The assurance strategy initally documents the set of assumptions and assertions derived from the requirements. It is elaborated and refined throughout the development, yielding the *assurance argument*, delivered with the system, which provides the primary technical basis for the certification decision. With the assurance strategy in place, certification of the trusted system can become an audit of the development process.

Keywords: Certification, Trusted Systems, INFOSEC, Software Engineering

INTRODUCTION

Computer security certification is the assessment that the computer hardware and software is trustworthy.[1] It supports the accreditation decision to allow the computer to process classified information in an operational environment.

Trusted product evaluation is the computer security certification of the product against the criteria of the *Trusted Computer System Evaluation Criteria* (TCSEC) [1]. Trusted system certification[2], on the other hand, comprises several technical and procedural certifications, including a technical computer security certification. The outcome of the trusted system certification influences the criteria for other certifications, such as administrative security and TEMPEST requirements. If the protection features of the system are deficient in any way, other protection measures must be used to protect the information maintained by the system.

While the security feature requirements of the TCSEC are targeted primarily at "information processing systems employing general–purpose operating systems that are distinct from the application programs being supported" (i.e., trusted products), the assurance requirements extend "to the full range of computing environments"[1, p. 2].

According to the TCSEC, both trusted products can be evaluated and trusted systems can be certified against its criteria. However, the evaluation approach that is used for trusted products does not satisfy the needs of an accreditor for a trusted system. In particular, the approach does not identify the risks of using the system in its operational environment. Accreditors want to know what vulnerabilities will exist if they turn on the system. A better approach is needed for certifying trusted systems. This paper proposes an informal but comprehensive approach that can be used by project managers, designers, and implementors of a system and can provide the accreditor with the risks of using the system.

We want to clarify our use of the terms *trusted product*

* In proceedings of the 16th National Computer Security Conference, Baltimore MD, Sept. 20 – 23, 1993, NCSC/NIST, pp. 165–172.

[1] The computer is *trusted* if we rely on it for security enforcement. It is *trustworthy* if that reliance is justified technically. A computer may be trusted even though it is not trustworthy, because the Designated Approving Authority (DAA) may permit its use despite known weaknesses.

[2] In the TCSEC[1], this is called a *certification evaluation*.

and *trusted system*. We have adopted the definitions of *product* and *system* from the European community's *Information Technology Security Evaluation Criteria* (ITSEC) [2]. According to the ITSEC, a *system* is a specific installation "with a particular purpose and a known operational environment". A *product*, on the other hand, is "a hardware and/or software package that can be bought off the shelf and incorporated into a variety of systems". A product or system is *trusted* if we rely on it for some critical purpose, such as (in the present context) security enforcement.

The characteristics and requirements of a trusted system's end–users and the threats to a trusted system's security can be determined with some certainty. The security requirements that it must enforce derive from national security policy. If a trusted system is based on an evaluated product, the person deploying the trusted system must ensure that the assumptions of the product are valid for the operating environment. It may be necessary to develop additional trusted code to enforce environment–specific security requirements.

The computer security certification of a trusted system can be based on the same criteria as the evaluation of a trusted product. But the system's certification may require more resources than the product's evaluation, because the system's security requirements are based on the known operational environment and so are more comprehensive than those of the product.

In the following sections, we discuss why the evaluation approach for trusted products does not scale up for trusted systems. Then we identify our objectives for trusted system certification and describe how the development process must be improved to support these objectives:

- adopt the accreditor's perspective and identify the system INFOSEC policy,
- base the development (and consequently the certification) on trade–offs between assumptions and assertions, and
- define an *assurance strategy* to motivate the development process.

The assurance strategy documents the assertions that must be true as the result of design decisions, and identifies the methods used to demonstrate the validity of each assertion for the system. Finally, we propose that with the assurance strategy in place, certification of the trusted system can become an audit of the development process.

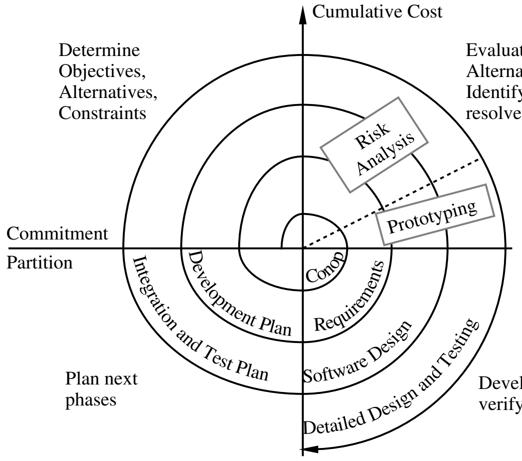

Figure 1: Boehm's spiral model

WHY THE PRODUCT EVALUATION APPROACH DOES NOT WORK FOR SYSTEMS

Boehm [3] defined a software process model that views the software process as an iteration through four phases: planning, risk identification, risk resolution and development. In this model (see Fig. 1), a project begins its life in the center and follows a spiral trajectory outwards: distance from the origin represents cumulative project cost, and angular displacement corresponds to the current project phase. We will use this model to illustrate the two ways described previously ([4]) for how the product evaluation approach can be applied to systems. Later, we will use it to illustrate our proposed system certification approach.

The NCSC's product evaluation approach provides security engineering expertise to the developer during the product's construction (Vendor Assistance and Design Analysis Phases) and then assigns an evaluation team to assess the completed product and documentation (Formal Evaluation). If we apply this approach to a system, as illustrated in Figure 2, security engineering support would be provided throughout the spiral, but certification would begin only after system delivery. The certification team must be isolated from the security engineering support effort in order to preserve the independence of the certification.

Although this process is likely to deliver a system, its certification may be quite protracted, because disagree-

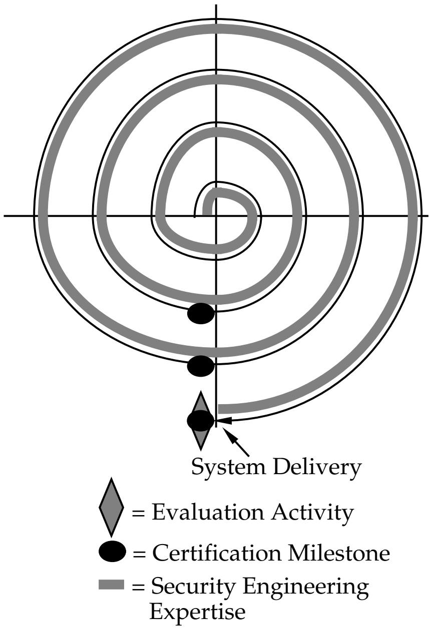

Figure 2: Certification after delivery

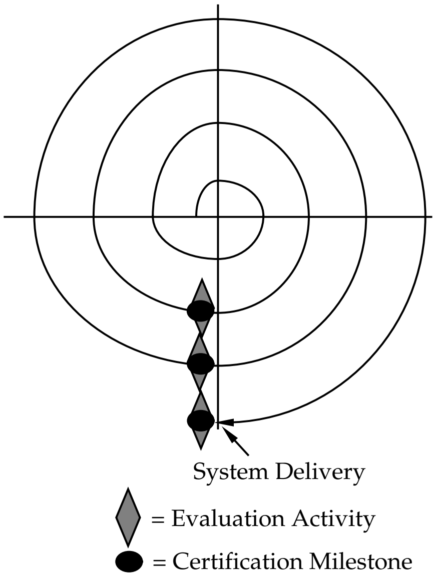

Figure 3: Milestone–based certification

ments between the security engineers and the certification team may require redevelopment of portions of the system. A product developer can accept this lengthy process, because the product can eventually be sold to many customers, but a system is often built for a single customer. The customer cannot tolerate lengthy delays between the delivery and operational use of the system, nor can the developer redevelop the system without further funding. The accreditor is left in the uncomfortable position of permitting the uncertified system to operate, accepting unevaluated risks, or denying the user a needed capability. For these reasons, delaying the start of system certification activities until after the system is completed seems impractical.

An alternative approach is illustrated in Figure 3. Here, security certification is structured as an independent verification and validation process that reviews the development at each contractual milestone. The security engineering support is eliminated, but the certification team's review of the deliverable provides feedback to the developer and the customer. If certification evidence provided at a particular milestone is inadequate, the developer must remedy the deficiencies before proceeding. Certification still does not occur until after the system is delivered, but the delay between system delivery and system certification should be much reduced from the previous approach. If the system is delivered, it should be certified promptly.

While this approach eliminates some of the risks of the previous one, it introduces new ones. Since the TCSEC describe assurance evidence only for a completed system, the certification team is challenged to identify what certification evidence is needed at each milestone in order to assure that the resulting system can be certified – and this activity may recur each time a new system, TCSEC class, or milestone structure is addressed. The need to revise deliverables to meet certification requirements at each milestone may substantially increase system development time. Finally, the necessary interaction between the certification team and system developer can threaten the independence of the certifiers.

CERTIFICATION OBJECTIVES

We have identified problems with two certification approaches. These difficulties are related to how trusted systems are procured and developed. While results from the certification process should influence programmatic decisions, programmatic constraints alone should not define the assessment process. The ideal certification process should be applicable to any project, regardless of lifecycle model and programmatic milestone definition.

Certifiers should view the development as it progresses. Certification decisions should be made throughout the system lifecycle — not just when particular pieces of assurance evidence have been completed. Also, certification decisions should be based on more than just the form of the assurance evidence. The certification team and the developer need a systematic way to reason about countermeasures, their effectiveness, and the design, development, and correctness of the protection mechanisms. In fact, if the developer made trade-off/design decisions based on the same concerns and in the same context as the certifier, the certification process could become an audit of the development process. The development process should produce assurance that countermeasures are effective and correct. The assurance strategy should drive the development of a trusted system and should begin at system concept with threat identification. We have now identified objectives for the development process as well as the certification process.

STEPS TO A SOLUTION

Consider the Accreditor's View

We believe that to develop a better approach to system certification, we must consider the accreditor's perspective of the system. The accreditor's primary concern is protecting classified information with the most cost–effective controls available. Given a particular mission, the threats to the successful execution of the mission, and a system intended to help accomplish it, the accreditor must first understand the risks of operating the system and whether there are countermeasures outside the system for reducing those risks to an acceptable level. When all means for reducing risk have been considered, the residual risk must be weighed against the contribution of the system to its intended mission, and the decision to operate it or not must be made.

The accreditor's view thus includes not only system hardware and software components but also the people who use the system and the administrative measures that regulate their use. Indeed, physical and personnel security measures often are instituted to compensate for uncertainties or weaknesses in automated systems; by bringing different security disciplines into a common framework, we hope to make it easier to formulate rational trade-offs among them and to clarify, if not quantify, the risks a particular set of choices presents when the system is operated.

Identify the System INFOSEC Policy

Past work in computer security has frequently identified "security policy" (as in "Formal Security Policy Model") with mandatory and discretionary access controls. But in the context of systems rather than products, this focus is clearly too narrow.

Sterne [7] recognized this problem and identified three levels of security policy: security policy objectives, organizational security policy, and automated security policy. One of Sterne's goals was to separate policies applied to people from those applied to machines. Although we agree that people must know what's expected of them and what they can expect of their machines, we want to provide a framework that accommodates trade-offs between human and machine policies. None of the policies Sterne identified seems to correspond directly to the view of the accreditor, who often is concerned with the mission of the system as well as the policies Sterne identified.

For this purpose, we identify the *information security (INFOSEC) policy*. As illustrated in Figure 4, the INFOSEC policy is derived from the organizational security policy and from other constraints. The automated security policy, i.e., the trusted system's security requirements, is then derived from the INFOSEC policy and the operational requirements.

The trade–off analysis begins with the system requirements analysis phase of the lifecycle and continues informally throughout system development. The resulting decisions drive the development, and consequently the certification, of the trusted system. So that the developer and the certifier can understand these decisions, we need a framework in which to capture and document them. With such a framework, if a trade–off decision leads to an unworkable design, the developer can better assess other alternatives, and the developer, customer and certifier can determine the impact of any changes on the development and certification of the system.

Think in Terms of Assumptions and Assertions

The framework that we propose for documenting the trade–off decisions is based on identifying the assumptions and assertions that must be true for the system as a whole to be secure. The notion of identifying assumptions and assertions is not new; it derives from earlier work documented by Landwehr [5] and Froscher [6]. Its use to identify relationships among various security disciplines, however, has not been advocated previously, nor has it been suggested as a tool for documenting trade-

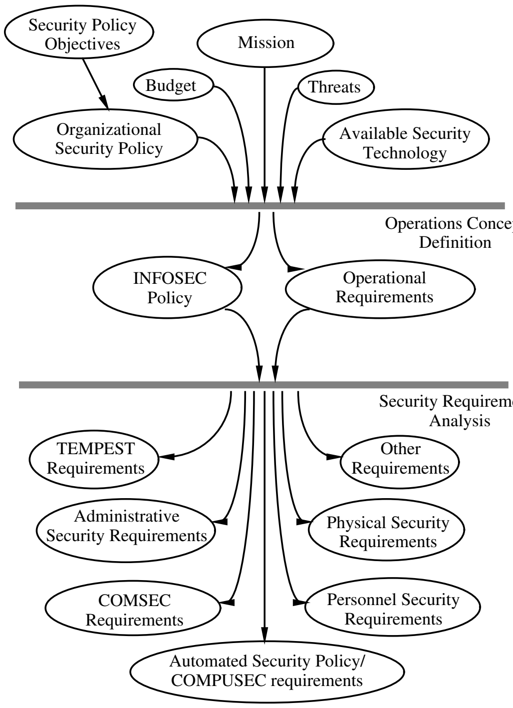

Figure 4: Introducing the INFOSEC Policy

off decisions taken during system development.

For a given system, *assertions* are predicates that are enforced by the system, and *assumptions* are predicates that are enforced in the system's environment. The system is unable to enforce the assumptions, but relies upon them. Together assumptions and assertions represent what must be true of the system in its environment to satisfy the security policy. If an assumption or an assertion is false, a security violation may occur.

An example assertion is

> *A user can only view on a workstation screen information for which (s)he is cleared.*

This is a relatively high level assertion about system behavior, and it might lead to several other, lower level assertions concerning the access controls implemented by the operating system, user authentication, and so on. Taken together, the collection of lower level assertions should support the argument that the higher level assertion will be enforced.

But suppose that the system developer can only use an operating system that lacks effective access controls, or that users cannot be required to authenticate themselves? In this case, other ways of supporting this assertion must be found. For example, one might require that:

- *All individuals able to view workstation screens are cleared to the SECRET level*, and
- *No sources that produce information at a level higher than SECRET are connected to the system.*

From the standpoint of computer security, these last two predicates are assumptions, for they cannot be enforced by computer software and hardware. From the standpoint of physical and personnel security, however, these are assertions: measures within those security disciplines should be sufficient to enforce them.[3]

The purpose of stating the assertions and assumptions explicitly is to facilitate a systematic analysis that demonstrates that all stated security predicates are true for the system and its environment. The systematic analysis compares the system's computer security assumptions to the assertions of other security disciplines. This exercise continues for all defined security disciplines. If assumptions for any discipline are identified that do not correspond to assertions for some other entity, then these assumptions represent *vulnerabilities* in using the system. If the vulnerabilities result in a risk that is too great, the trade–off analysis is revisited. This analysis is illustrated for administrative security, a physical security officer and computer security in the simple example of Figure 5.

The trade–off continues throughout the design of a system that must enforce the COMPUSEC requirements. Assertions and assumptions are determined for individual hardware and software components, then for modules, etc. At each new level of specification, assertions are derived from the previous level. Assumptions are propagated from higher levels or they may become assertions for this level. In addition, new assumptions may be introduced.

The assumptions and assertions approach can represent clearly the effects of decisions made during the development cycle and can make explicit the risk of using the system. It also makes explicit the risk of interconnectivity and allows protection measures to be developed that promote secure interoperability. Because it captures the role each part of a system plays in the development of a countermeasure, this approach allows the maintainer and the accreditor to reason about the effects of any proposed change to the accredited system. The maintainer and the certifier can identify what parts of the system assurance argument must be reexamined to reaccredit the system as a result of any change. We also believe that

[3] Naturally, this example is incomplete! For example, we have said nothing about the system's initial state. Establishing the completeness of a set of assumptions and assertions, relative to some set of security objectives, is not something we expect to be able to do algorithmically.

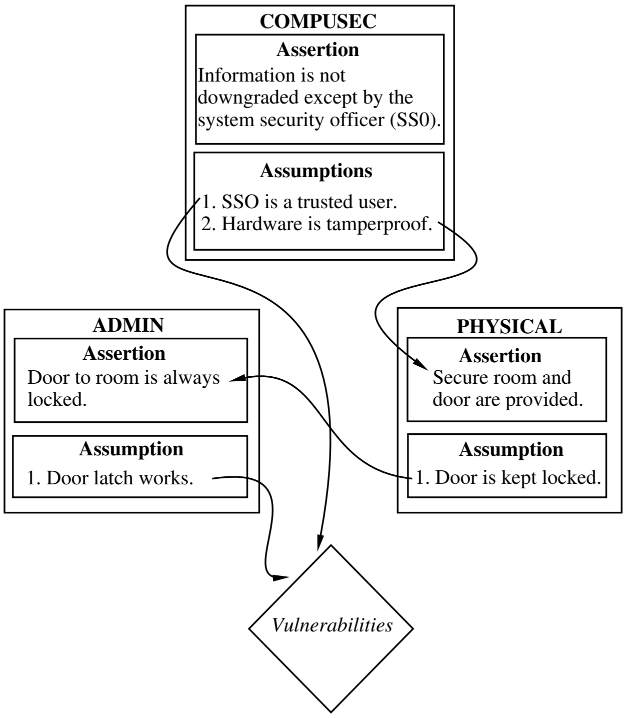

Figure 5: Using assumptions and assertions to find vulnerabilities

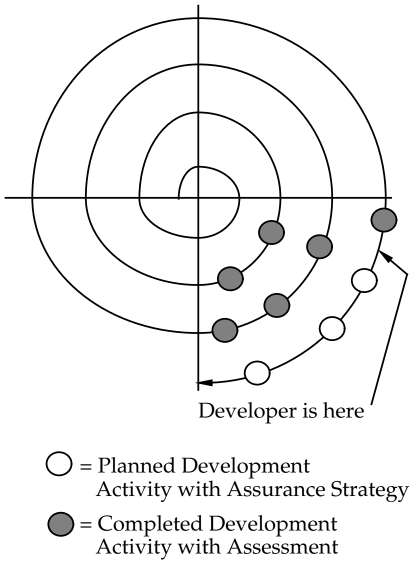

Figure 6: The development process — in progress

the assumptions and assertions approach can support trusted system technology's migration to the open systems goals for insertion of new technology into existing systems.

Define an Assurance Strategy

Too often in trusted system developments the security engineering team is isolated from the system engineering team. Many of the TCSEC's assurance requirements are satisfied by documentation that can be produced independently of the system engineering process. There is little incentive to integrate the security engineering process and the system engineering process. If a security flaw is detected late in the design, it may be very expensive to fix. Developers and certifiers need a strategy for assessing how the system's assurance requirements will be satisfied, so that they can identify vulnerabilities *before* significant development resources have been expended. Ideally, the development process is structured so that this assessment can be completed with little additional effort.

We propose that an *assurance strategy* be defined and maintained by the developer to record the assurance trade–off decisions — in the form of assumptions and assertions — and to identify the techniques and methods that will be used to satisfy the assurance requirements. A strategy for demonstrating assurance is defined and assessed for each development activity and reassessed when that activity is completed. Figure 6 illustrates the relationship of the assurance strategy to the development activities.

The assurance strategy initially documents the set of assumptions and assertions derived from the requirements. It is elaborated and refined throughout the development, yielding the *assurance argument*, delivered with the system, which provides the primary technical basis for the certification decision. The assurance strategy streamlines the certification effort and makes the satisfaction of the assurance requirements a major force in the development process. In ITSEC terms, it addresses two facets of assurance: effectiveness and correctness. Evaluation of effectiveness assesses suitability of functionality, binding of functionality, vulnerabilities, ease of use and strength of mechanisms. Evaluation of correctness includes the construction and operation of the trusted system.

The assurance strategy allows certifiers to assess the role a design decision plays in the overall assurance argument and to determine whether the proposed assurance techniques are effective for demonstrating the validity of the decision. This determination, which can occur early in the system's lifecycle, also facilitates recertification and accreditation when the system is modified or when the operational configuration changes. Development of the assurance strategy allows the developer, the accreditor, and the user to decide how much assurance is needed for different countermeasures. Every countermeasure may not need to satisfy the same assurance requirements.

The assurance strategy describes how evaluated products will be used and what role they will play in enforcing the system security policy. If evaluated products are not used, the strategy should convince the certifier why no products will suffice. Since a goal of trusted system development is to minimize the assurance effort, the certification team should be involved at the outset in assessing correctness and effectiveness.

A COMPREHENSIVE CERTIFICATION APPROACH

By completing the steps outlined in the previous section, we reduce the certification task significantly. Most of the certification process would evolve into an audit of the development activities (as illustrated in Figure 7) that contribute to the construction of the assurance argument, e.g., the formal modeling effort, the specification of the interface requirements, the design refinement, the requirements decomposition, and so on. The certification team would assess the effectiveness of the assurance strategy for each development activity and would audit the activity's contribution to the overall trusted system assurance argument.

This certification approach satisfies several important objectives. It maintains the security certification as an Independent Validation and Verification activity that proceeds along with the development. It increases the frequency of checks on the development so that they are not limited to reviews of major documents produced at a relatively small number of major procurement milestones. Thus it is flexible enough to be applied to any development lifecycle. It has ongoing visibility into the development process, but it does not threaten the independence of the certification team. Finally, the assertions and assumptions framework provide a systematic way for the developer, certifier and accreditor to reason about countermeasures and their effectiveness.

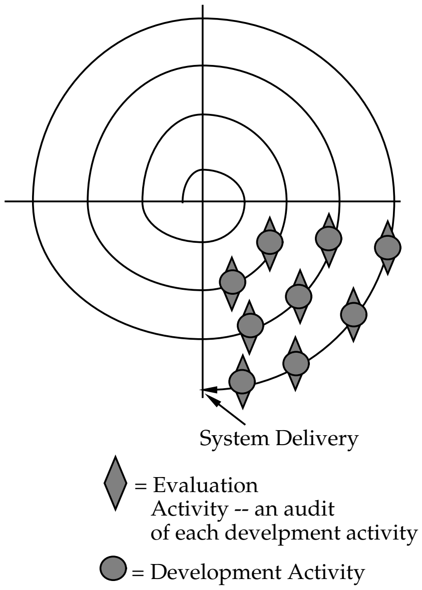

Figure 7: Certification as an audit of development

SUMMARY AND CONCLUSIONS

The trusted product evaluation approach cannot be easily applied to certifying trusted systems because it does not address the risks of operating the system in its environment. We have proposed a comprehensive certification approach that eliminates many of the shortcomings of the product evaluation approach. It combines the concepts of an INFOSEC policy, assumptions and assertions, and a development–motivating assurance strategy to move toward reducing certification to an audit of a trusted system development process.

We believe the system certification approach proposed here promises to improve upon previous methods in four ways:

1. it is based on a rigorous but flexible framework that makes the risk of using the system explicit,

2. it addresses the trusted system in its environment,

3. it includes a rigorous strategy for avoiding pitfalls in the certification and development process, and

4. it makes demonstrating assurance an explicit and useful part of the development process.

The next step is to evaluate this approach in more detail by applying it to a significant example.

Acknowledgements

The authors wish to thank H.O. Lubbes and the anonymous reviewers for their helpful comments.

References

[1] National Computer Security Center, Ft. Meade, MD, *DoD 5200.28-STD, Trusted Computer System Evaluation Criteria*, December 1985.

[2] Commission of the European Communities, Luxembourg, *Information Technology Security Evaluation Criteria (ITSEC)*, June 1991.

[3] B. W. Boehm, “A spiral model of software development and enhancement,” *Software Engineering Notes*, vol. 11, no. 4, Aug. 1986, pp. 22-42.

[4] J.N. Froscher, J.P. McDermott, C.N. Payne, and H.O. Lubbes, “Successful acquisition of certifiable application systems (or: How not to shake hands with the tar baby),” *Proc. Sixth Annual Computer Security Applications Conf.*, Dec., 1990, IEEE CS Press, pp.414-422.

[5] C. Landwehr, C. Heitmeyer, and J. McLean, “A security model for military message systems,” *ACM Transactions on Computer Systems*, vol. 2, pp. 198–222, August 1984.

[6] J. Froscher and J. Carroll, “Security requirements of navy embedded computers,” NRL Memorandum Report 5425, Naval Research Laboratory, September 1984.

[7] D. F. Sterne, “On the buzzword ’security policy’,” in *Proc. Symposium on Research in Security and Privacy*, IEEE, June 1991.