REQUIREMENTS FOR CLASS A1 SYSTEMS AND MAJOR DIFFERENCES BETWEEN DIVISION A AND DIVISION B SYSTEMS

Dr. Carl Landwehr

Naval Research Laboratory

I am not an author of the Orange Book (I work for the Naval Research Laboratory, not for the Computer Security Center) so my view is that of a consumer of the Criteria rather than a producer. I look at it as something that I might employ sometime if I were to attempt to build a secure system. So, I've tried to aim this presentation towards people like myself who may try to use the Criteria.

The crucial fact about Division A is that A stands for Assurance. That's what the A level is all about. In my view, it requires a higher level of assurance than any of the other divisions shown on Slide 1.

As Sheila mentioned, the categories of criteria that are listed in the document are four: Security Policy, Accountability, Assurance, and Documentation (at least at the A level, there is Documentation) (see Slide 2). In fact, the A level levies no new requirements on the first two of those categories. The functional requirements for a level-A system are identical to those requirements for a level-B3 system that Dan has just described. So, you might say, "I've got a B3 system. Can I just change it into an A system by implanting a little more assurance - more tests, and a little extra documentation?" The answer is, "No." In fact, the decision to meet the level-A criteria affects the entire life cycle of a system, because those requirements, even though they do not change the functions of the system, have to do with how that system is developed.

I've tried to indicate on Slide 3 a simple view of the life cycle of a system: starting with requirements specification, then the top level or system specification, detailed specification, implementation, testing, presumably, and then operation. The notations along the sides show where level A has an affect. In the first place, as you can see, it requires more strict configuration control. Configuration control gets pushed back all the way to the design phase. You must have configuration control if you want a level-A system; you have to show that the control was in place during the design and was applied to the design documentation. I've noted a security model on the right-hand side because that's required. As Dan noted, a formal description of a security model is, in fact, required at the B2 level by the Orange Book. The main new requirement for A level is a Formal Top-Level Specification (FTLS). At the B level, only a Descriptive Top-Level Specification (DTLS) is required.

An FTLS has to be formal, and there is a definition in the back of the Orange Book of what "formal" means when applied to "Top-Level Specification." I don't think anyone has a perfect definition of that word. In any case, you have to demonstrate a correspondence between the security model and the Formal Top-Level Specification. As I read it, that correspondence does not have to be demonstrated formally; that is, you don't have to have formal mapping from the security model to the top-level specification, but the correspondence has to be demonstrated somehow. I suspect that if you can do it formally, it will be more convincing. But a lot of this has to do with whom you're convincing, and what they have in mind. The dashed lines on the side of Slide 3 represent the requirement to demonstrate the correspondence between the formal top-level specification and the detailed specifications. There is not a requirement for a formal detailed specification, and so that correspondence is probably going to be informal, unless you've done even more than is required. Similarly, there needs to be a demonstrated correspondence between the detailed specification and the implementation.

Dan talked about covert channels and described them. There is a requirement at the A level for a formal analysis of storage channels. That requirement is (the way I read the Criteria, anyway) the one place where the formal top-level specification really gets used. You can't do a formal analysis of storage channels without a formal top-level specification. So that requirement is a short sentence in the Criteria that, in many respects, actually levies the requirement for an FTLS.

Finally, in the operations and maintenance phase, there is a requirement for a trusted distribution facility. There is quite a lot there that I've gone through very quickly. You may still ask, "If B3 specifies all the necessary functions, and we have all the functions we need in B3, why bother with level A? Wonderful, you've got such-and-such documentation, but who needs it?" The reason, (Slide 4) it's needed is that we want assurance that the functions operate as intended. And the reason we want that assurance is that we want to place greater reliance on the automated controls of the system. Then we can reduce the procedural and personnel controls and operate these systems more flexibly and effectively. Without that additional assurance, we can't relax procedural or personnel controls.

Now, (Slide 5) I'll back off just a little bit, philosophically, and say, "Let's look at these criteria. Why would you want to take this particular way of gaining assurance?" The Orange Book says that we get higher assurance by having formal specifications and things like that. So I thought a little about what people do to get assurance in other systems.

The first thing people do is test. And there are requirements for testing in the Criteria as well. I don't mean to overlook those. I would categorize "test" as positive, that is, trying to demonstrate that the specified functions of the systems are there. It's the kind of thing people do all the time. There is also negative testing, where you try to see whether the system will break, or if you can break it. That seems to be one way of getting assurance: testing.

Proc. 14th National Computer Security Conference, 1991

Another way is redundance. It seems to me that there are different kinds of redundance. One kind, that we've seen in the space shuttle, for example, is to say, "Here's one specification. Let's have two independent implementations of it. You do it, and you do it. And then we're going to run them and compare the answers." If the answers come out the same, even though different people implemented the specification, I may have a little higher confidence that the answer is what I wanted. At least both implementors made the same mistake. That's one way of getting some increased confidence, and I think that it is based on a kind of redundance.

Another way, the one advocated in the Orange Book is to construct different descriptions of the system and show that those descriptions are equivalent. These are, in a sense, redundant descriptions of the system. You start with a security model, which is a very high-level, abstract description of the system behavior. You must have a formal security model, even for B2, as I've already said. The A level requires an FTLS as well, and you must show that it corresponds to the security model. The correspondence may be informal, but it must be demonstrated. I think that's a kind of redundance. That's where the assurance comes from, in my view.

The guideline for testing is also strengthened for A-level systems. It says more people have to fail to penetrate an A level system than a B level system, and they have to be smarter people. You'll have to look up the details.

The next two slides (Slides 6 and 7) review the criteria that are in the Orange Book. There are additional criteria at the A level in two categories - Assurance and Documentation. Two kinds of assurance are described: operational assurance and life cycle assurance. Under operational assurance, the requirement for formal methods for covert channel analysis is levied. That is the only requirement added in operational assurance, I believe. In life cycle assurance, first comes security testing - this requires a demonstration that the implementation is consistent with the formal top-level specification. I raised the issue at one point that tests (unless they are exhaustive - a practical impossibility for large systems) can't really demonstrate that two things are consistent; they can only demonstrate that they're inconsistent. Apparently, this wasn't considered a serious discrepancy. Second under life cycle assurance comes the requirement for design specification/verification. Here, the requirement for a formal top-level specification is imposed. The formal top-level specification and the descriptive top-level specification have to include everything visible at the TCB interface. The idea here is that the trusted computing base provides certain functions. All of those functions visible to users at the TCB interface have to be called out in the formal top-level specification, and that includes functions that are not even software implemented - they could be implemented by firmware or hardware. The SCOMP FTLS, for example, includes some hardware functions.

Slide 6 also covers the third and fourth A level requirements under life cycle assurance. These requirements are more mundane; they don't push the state-of-the-art, except that they're rarely applied as extensively as the Criteria implies. The first requirement is for configurations management. There must be a configuration management system to control changes to the formal security policy model (should you wish to make any such changes), the descriptive top-level specification, the formal top-level specification, and so on, throughout the entire life cycle. The "and so on" includes design documentation. Let's back up for a second. We have a view of a wonderful system in which we have a formal top-level specification, and we have an implementation, and we have demonstrated some correspondence between them, and then we have code. We want to rely on the controls in that code. So we must be sure that the code that runs operationally is the code that we built - that it hasn't gotten subverted somewhere along this path. That is the motive for having tools to compare a new version of the trusted computing base with a previous version and show the changes. If, in turn, we are to rely on what these tools tell us, it becomes important that the tools work properly. So those tools have to be under configuration control too, because subverting them can be equivalent to subverting the mechanism for releasing new versions - it could allow a Trojan horse to be inserted unnoticed. Similarly, the material for generating the trusted computing base must be protected. The requirement for a trusted distribution facility follows from this line of reasoning. The specific requirements here are to assure the integrity of that mapping between the specification master copy and the code master copy.

Slide 7 shows the requirements added by Level A for documentation. First, test documentation is required. This must include the results of mapping the trusted computing base source to the formal top-level specification. The Criteria does not specify the form of this documentation; that will probably be determined on an adhoc basis. As part of the design documentation, the correspondence between the formal top-level specification and the implementation must be described. This description can be informal. The way that the elements of the trusted computing base correspond to the FTLS must also be documented; this too can be informal. I find this a little confusing. I'm not sure what the elements of the formal top-level specification are versus the elements of the trusted computing base. Maybe Sheila can enlighten me on that.

SHEILA: Yes.

LANDWEHR: The final requirement is for a description of the hardware, firmware, and software mechanisms strictly internal to the TCB. Elsewhere, a description of the functions (hardware, software, and firmware) available at the TCB interface is required. Here, that requirement is extended to require a description of any mechanisms internal to the TCB that are not reflected in the FTLS. I suspect the motive is to uncover sneak paths within the TCB that are not covered in the TCB interface specification. But, again, I'm uncertain exactly what the considerations were for including this requirement.

To recap, suppose I want to build an A level system. How will its life cycle differ from that of a system intended for Level C or B? (Slide 8) In my view, the formal top-level specification should be developed prior to, or at least in parallel with, the descriptive top-level specification. The descriptive specification corresponds to a conventional software design document. The formal specification should control the informal detailed specification; at the least it ought to track the changes in the detailed specification.

Otherwise, it will be difficult to show that a mapping exists between the two. For system developers this is a very important point. To get the benefits of this approach, the implementors have to understand the language in which the FTLS is written, and they have to be competent to update that specification. Otherwise, one group will write the FTLS and then another will implement it. If the implementors can't read the FTLS, they may just use the informal specifications. Differences will arise among the different specifications, some will get out of date, and demonstrating the necessary correspondences will be impossible.

Increased configuration control will also change the life cycle, as will closer controls on distribution and maintenance.

For those of you who haven't seen a formal top-level specification it's typically a collection of functions defined in a particular non-procedural notation (Slide 9). Non-procedurality is not a requirement, but that's the most common form. In any case, it's a collection of functions analogous to those you might see described in English, only presented in a more structured, less ambiguous (more formal) way. It will be a more mathematical-looking document than usual specifications. Slide 10 lists some available languages. You can read about them in an article written by Maureen Cheheyl, Morrie Gasser, George Huff, and Jon Millen, entitled "Verifying Computer Security," in *ACM Computing Surveys*, September 1981. That's a good place to begin learning about formal specifications for computer security.

Another thing I would want if I were going to build an A level system would be some examples (Slide 10). It's always easier to do something new if you have an example to follow. Unfortunately, no A level systems have been certified yet, but there are some evaluations in progress, and there is documentation available for some of those. Unfortunately, I can't tell you where to get these documents. However, I did my best in an article that appeared in *IEEE Computer* in July 1983, and I have provided some references there. You might also ask the people at the Computer Security Center, since I think that they ought to develop a library of such documents or at least provide references to them. I will point out one other recent article, by Jon Silverman, on the verification of the SCOMP kernel. It is in the Proceedings of the Ninth Symposium on Operating Systems Principles - ACM SIGOPS.

Earlier drafts of the Criteria included an A2 level, which has been deleted. I think the reason for the deletion is that people at the Center think that meeting those requirements is not within the state-of-the-art yet. What we might see in the future (Slide 11) are requirements for using verified tools to produce secure systems. We might have verification requirements on compilers, for example. We might also see some proofs, not just of formal top-level specifications, but of lower-level specifications, and proofs of correspondence between levels. Perhaps test data will be generated automatically from specifications. Dan alluded to the idea that in a more structured system one might be able to do a more intelligent job of testing. We may also see proofs of different sorts of security properties. The primary emphasis of security properties now, as was pointed out this morning, is on disclosure. There may be other properties people could formulate, and they might like to prove that they are preserved by some system.

I will close with one problem that I can't resist pointing out (Slide 12). There have been some small systems built, perhaps a thousand lines of GYPSY in length - small but, nevertheless, functional systems that have been implemented and have had their code verified, as well as their formal top-level specifications. So they've actually met the fundamental requirements for assurance that are levied by level A. In fact, they've not only met them, they've exceeded them. However, these are small, special-purpose systems. They don't provide the functions that are required of even a B1 system and they don't need to provide them. So, under the current Criteria, if I had to evaluate them, I'd probably have to rate them somewhere in level C. This, to me, is a problem. The structure of the Criteria now gradually increases what's required on all components as you advance from one level to the next. There is an increase in the formality with which the security policy is stated, in what labelling is required, in how much testing is required, and so on. There is a gradual increase in requirements in each category from C1 to C2, C2 to B1, B1 to B2, and so on until we make the jump from B3 to A. Level A primarily increases requirements in a single category: assurance. So, I see an unaesthetic difference between the way A is defined relative to the other levels and the gradual entry of the others. I'm not sure exactly how to improve this.

SHEILA: What would you see as a better rating scale?

LANDWEHR: Do you want me to propose one?

SHEILA: Yes. Since you brought it up.

LANDWEHR: I don't have a ready answer. One possibility is to have ratings apply independently to each of several axes. I think separating concerns and being able to say that a system has one level of assurance and another level of function, for example, might be useful. I think that's quite a possible scheme, though it's not the initial one. There may yet be a different color document after the orange one. (I'm speculating.)