Pretty Good Assurance

Jeffrey R. Williams
williams@arca.com

Marv Schaefer
marv@arca.com

Douglas J. Landoll
landoll@arca.com

Arca Systems, Inc.
8229 Boone Blvd., Suite 610
Vienna, VA 22182

Abstract

This paper describes the need for pretty good assurance: clearly stated claims about the security properties of systems, accompanied by evidence that explains in clear terms why we should believe that these claims are substantiated. Several different types of threats are identified and their relationships to assurance are explored. The developer's role in creating an assurance argument is distinguished from the user's role consuming assurance. Finally, some thoughts on the future are presented.

1. AN INFORMATION TECHNOLOGY CRISIS

Why Assurance?

There is a crisis emerging in information technology. Reliance on this technology is increasing much more quickly than our ability to deal with the also increasing threats to information security. Users need to know if their products are susceptible to threats, but are usually oblivious to the scope and nature of the problem. Often, technology is blindly trusted, although it is neither understood nor trustworthy. What is needed is assurance. Unfortunately, the need for assurance is largely ignored by the ordinary users of information technology because it is seen as relevant only to government, military, or classified information technology.

Assurance is defined as the confidence that the security needs of the user are met by the information system [WITAT95]. Perfect assurance is impossible to achieve. It has been discouragingly costly and time-consuming to produce assurance evidence satisfactory for evaluation, especially that required for achieving the higher ratings of the TCSEC and the ITSEC. In the best case, certifiers and accreditors may require that all developers and quality-assurance personnel hold security clearances and work in costly secure environments. Even then, they are often forced to settle for the best assurance that can be collected and assessed under a fixed schedule and budget. Often, this approach has been wasteful in its efforts, while being ad hoc and haphazard in its conclusions. The difficulty is knowing where to focus efforts (and resources) for maximum payoff.

Most vendors cannot afford the costs of producing formal evaluation evidence for the government, and most organizations cannot afford the costs of buying assurance evidence of that quality. The rest of us have had to settle claims made about products' security properties — even though these claims are sometimes produced by a vendor's advertising department.

What the rest of us need is pretty good assurance: clearly stated claims about the security properties of systems, accompanied by evidence that explains in clear terms why we should believe that these claims are substantiated. We want to know, for example, who produced the product, how qualified they are in the technology, and how well their previous products' security mechanisms have stood up under real-world use. We don't want to pay for assurances we don't need for our anticipated use of a product. Pretty good assurance cannot just consist of a preponderance of empty claims. If the vendor claims that the results of a test suite are part of the product's assurance, there needs to be enough evidence present that the interested consumer will be capable of independently assessing their adequacy and completeness. Similarly, if the vendor claims that the product can safely be placed on the Internet, then the vendor should furnish assurance evidence that supports this claim. (For example, if it is a Unix-based product, that it can be configured such that it satisfies analysis by a tool like COPS or SATAN.)

In order to address this need, we need a full understanding of the various dimensions of assurance.

0-8186-7318-4/95 $4.00 © 1995 IEEE

Ordinary People Need Assurance

Ordinary people are affected by the operation of information systems, and therefore need assurance that their interests are protected. If your business depends on the services or data provided by an information system; if your life is affected by decisions made or aided by information systems; or if any information system contains your personal information, then you are a stake-holder in assurance.

In most cases, reliance on the security features of information systems is unfounded. The recent flurry of security incidents exacerbates everyone's need for assurance in information systems. The rapid increase in public postings of automated tools for penetrating systems are striking examples of a growing international threat base. These tools make it easy for anybody, not just highly skilled attackers, to exploit the most complex vulnerabilities in an information system. Another class of examples includes the egregious flaws recently discovered in both hardware and software.

Generally, users have no way to judge whether or not such flaws are likely to exist in a product or system. In fact, users are not even able to assess the seriousness of known flaws. For example, the floating-point divide (FDIV) flaw on the Intel Pentium was very unlikely to affect most users, yet many demanded a new chip. At the same time, these users are likely to have similar significant undiscovered flaws in application and/or system software.

A New Look

Current approaches to assurance tend to focus on establishing correctness. While correctness is certainly important, it is also very difficult and time-consuming to establish. Even technology that is formally proven correct might still be vulnerable to trivial attacks, because a mechanism was left out or is too weak to stop the threat.

This paper steps away from a criteria based assurance paradigm and presents a way of thinking about assurance that allows fuzzy decisions, trade-offs, and priorities without requiring absolute proof. This approach assists developers in producing information technology which can be trusted. It also helps users ask for and understand the proof they need to trust their systems.

2. BACKGROUND

Users, Products, Systems, and Operations

The notion of a "user" is especially complex in the case of assurance. The user of technology is generally the person at the keyboard. However, the person who needs assurance might be that person, the buyer, the owner of information assets, etc. This paper uses the term "user" to mean all of these people — including anyone who is a stake-holder in the security of the product or system.

In this paper we do not make a distinction between products and automated systems, except to note that automated systems are typically are made up of several products. However, this is a minor distinction at best, as every product is made up of other products.

However, there is a major distinction between products and operational systems. An operational system consists of people, processes, environment, and technology. Trying to determine assurance in an operational system by looking only at the products involved is a bit like assessing a cake by looking only at the ingredients.

Product vendors should be concerned with product security, independent of a specific set of threats and assets. Product assurance is confidence that a product will behave as advertised, without defects.

Buyers, users, and integrators need to be concerned about the security of an operational system. They must consider their specific threats and assets and determine their assurance needs. Operational assurance is the confidence that all threats to the operational system have been adequately countered.

Threats

The primary concern when determining assurance is whether all the relevant threats have been countered. For purposes of this discussion, we interpret threat rather broadly to mean anything that the user does not want to happen. Table 1 shows several types of threats and the assurance that countering these threats yields.

Threat Type	Examples	Countered With	Assurance Gained
Threats to Operation	Disclosure Disruption Deception Usurpation	Operational People, Processes, and Environment Operational System Design and Configuration	Confidence that assets are protected by something "Are protections present?"
Threats to Technology (Defects)	Incorrectness Complexity Insufficient Features Unnecessary Features Misconfiguration Weakness	Evidence about Development People, Processes, Tools, and Environment Evidence about the Technology	Confidence that the technology not have exploitable defects "Are they good enough?"
Threats to Evidence	Incorrectness Insufficient Evidence Unnecessary Evidence Misrepresented Evidence Complexity	Circumstantial Evidence Developers' Experience Developers' Honesty	Confidence that the evidence actually shows what it was supposed to "Are they really good enough?"

Table 1 — Establishing assurance concerning several types of threats.

Categorizing the threats to information technology in this manner [WILL95b] allows us to discuss which ones are more or less important to assurance. In general, addressing the direct threats to assets is more important than ensuring that the evidence has been accurately prepared. In other words, you should lock all the doors before you check how well one of the locks works.

Assurance and Cereal Boxes

The security community has tried to simplify assurance to a single view [WILL94]. This approach is flawed because different users of assurance have different needs. For example, an ordinary user may simply believe someone else's claim that a system is highly trustworthy. A system integration manager needs some additional evidence and lower level claims. Finally, technical engineers need to understand the assurance at a very detailed level.

Consider the different ways that we discover the nutritional value of a box of cereal. At one level, the front of the box prominently displays that the cereal is "Nutritional." The concerned user may notice a smaller label that describes the cereal as "No Sugar, with Eight Essential Vitamins and Minerals." For those experts who are very health conscious, the side of the box lists all the ingredients and percentages of the U.S. RDA contained. They may even discover that, despite all the healthy ingredients, the cereal contains a high level of fat.

Just as each of the different views of the nutritional information is important to a different type of user, the capability to provide different views of the same assurance information is critical to meeting the needs of the many different assurance users.

3. DEVELOPING ASSURANCE

Assurance Claims and Evidence

Users are confronted with a multitude of assurance claims for the products they intend to use. These assurance claims include government certifications, capability evaluations, vendor reputation, and marketing hype. Unfortunately, users have little guidance on how to interpret the relative merits of the various claims. Users are frequently forced to accept these claims on blind faith, as there is no practical way for the user to validate them.

This paper uses several terms to describe different aspects of assurance.

- Assurance Claim: an assertion made by a developer that a system has some security relevant properties. At the highest level, these claims should cover all the threats that the information technology is likely to encounter [SSECMM].
- Assurance Evidence: data on which a judgment or conclusion about trustworthiness may be based. Assurance evidence indicates whether or not information technology has exploitable flaws.
- Circumstantial Evidence: evidence that confirms the reliability of the primary evidence. Circumstantial evidence is likely to involve the materials, process, people, or environment used to create information technology [SSECMM].

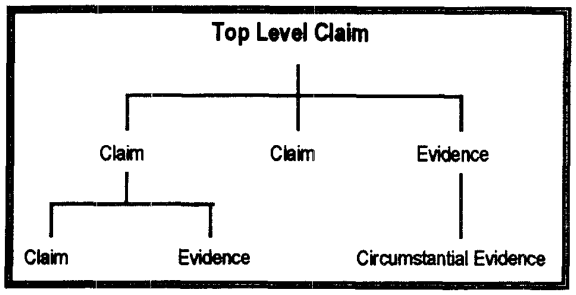

Figure 1 — Assurance claims are supported by evidence and additional claims

Figure 1 shows how a top level claim, such as "my product is secure" is supported by increasingly specific claims and evidence. This structure allows a user to determine the assurance they have in an information system. Where they are left with an unsubstantiated claim, the user can either accept it or demand some evidence.

Properties

A product may have many properties. Here we limit our discussion to those properties that are relevant to assurance, i.e., those that relate to the ability to enforce a security policy. The relevant properties are correctness, completeness, strength, simplicity, and ease-of-use. We call these property claims to signify that these are statements that require evidence to be substantiated.

These five properties are based on the three principles of the reference monitor concept (i.e., correctness, completeness, and simplicity), strength of mechanism (strength) and the importance of operational security (ease-of-use). Assurance that the product can enforce its allocated policies is based on these properties of the product.

- Correctness of the product ensures that the product performs as specified. In other words correctness indicates the absence of bugs or flaws, which exist whenever a lower level representation of a product does not correspond to a higher level.

- Completeness describes the product's coverage of the specified functions. Completeness refers both to the coverage of all specified functions and the protection of all indicated resources and assets. For example an operating system may provide mandatory access control over a subset of its resources (i.e., named objects). Such a product employs incomplete MAC protection over objects i.e., there are objects that do not adhere to the MAC rules. [Typically these objects are called covert channel resources.]

- Strength attests to a product's ability to withstand attack. This property focuses solely on the product design and mechanisms capabilities and not their implementation. The following example illustrates the importance of strength. To provide confidentiality services, a product may use a cryptographic algorithm based on alphabetic substitution (i.e., an "a" is encoded as a "z", a "b" is encoded as a "y", etc.). This algorithm may be perfectly implemented and pass functional testing with flying colors. However, the mechanism is inherently weak in that nearly any adversary could decode the communications leading to disclosure.

- Ease-of-use is concerned with the ability of the users and operators to install, operate, and maintain a secure configuration. Without this property a product could be seen as strong, correct, and complete and still be so difficult to configure that vulnerabilities are introduced by unwitting users or administrators. To illustrate this concept further consider the implications to a operational system if access control lists are updated incorrectly.

- Simplicity describes a product's analyzability. This is an important assurance concept since an understanding of the product is a prerequisite to gaining assurance. Among the factors considered are minimal design (least common mechanism, small domains), layering (data hiding, data abstraction), and least privilege (components, modules, and processes).

Note that these claims do not have anything to do with implementing a security policy. The reasoning is that products themselves are policy neutral. They just provide features that can be used to implement various operational policies.

Developing an Assurance Argument

It is the developer's job to assemble a set of claims that the system meets the security needs of the users and to provide the evidence appropriate to back those claims.

Often, developers produce assurance evidence that is lost because it is not recorded and refined. In particular, this occurs during the product's early design stages: meetings and blackboard design sessions frequently address the completeness and isolation properties of their envisioned security policy enforcement mechanisms. This is when the concepts of "how it works" and "why it works" are derived, analyzed, and assessed. Such information constitutes the confidence the developers have in the system's security enforcement. It should be recorded to document important design decisions and to serve as a basis for subsequent analysis and testing. Unfortunately, it is often not recorded because of the informal environment in which it is produced.

Any evidence that supports a claim is relevant. This approach gives the developer a large amount of freedom to establish claims with the cheapest, fastest, and best evidence available. For example, a vendor wanting to claim that some technology is error-free may provide a formal model as evidence. Other evidence choices include using a peer-review process, hiring only well-qualified designers and developers, or performing extensive testing.

Developers should look for ways to reduce the amount of evidence needed to establish their claims. Some examples are:

- Avoid redundant evidence. Although it does not hurt anything, once a claim has been established to a sufficient degree of assurance, additional evidence does little to strengthen the top level security claims of the system.
- Try to even out the coverage of evidence across the top-level claims. It is not practical to concentrate on one particular threat and ignore all the others.
- Use the assurance claims and evidence of other products to support their own claims. For example, one part of an integrator's claim that all activities on the system are audited would be the operating system vendor claims to provide an audit mechanism and vendor evidence supporting that claim.
- Produce evidence that will support multiple claims. Establishing the correctness of the hardware, for example, will support a claim of correctness for almost all parts of the system.

A lengthy example is pictured in Figure 2. The relevance of the evidence to the top-level claim is directly related to its proximity to the top-level security claim. Consider the following example of establishing a security claim that a system provides confidentiality of the systems assets. Notice that the claims supporting the top-level claim directly address threats to assets. Claims at the next levels are substantiated by indicating which mechanisms have been implemented. Claims that these mechanisms are good are substantiated with evidence and circumstantial evidence.

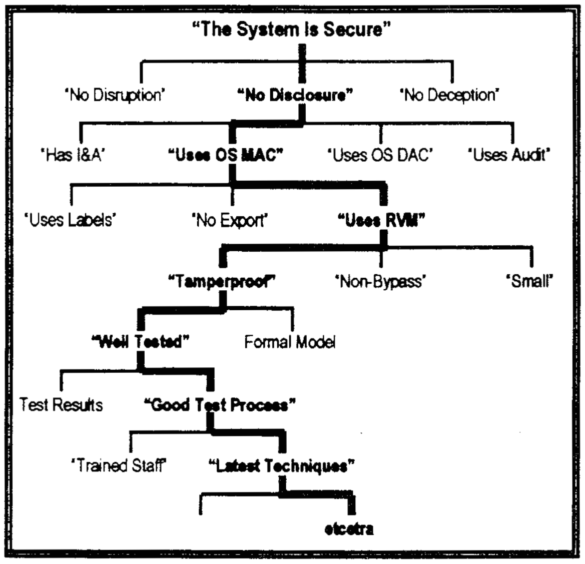

Figure 2 — Support for a developer's security claim

If the vendor has chosen wisely, their claims will closely match the security needs of the user. If the vendor has provided sufficient support for high-level claims, then the user can quickly and easily use the product with a sufficient degree of assurance.

Tradeoffs

Given a completed framework of evidence produced by candidate assurance methods, some pieces of evidence that support the same assurance claim can be considered for substitution. The easiest trade-off decisions are those between pieces of "sibling" evidence, which directly support the same assurance claim. However, any evidence which supports any assurance claim no matter how indirectly may be considered for substitution.

The approach described above will also be instructive in highlighting those assurance claims which do not require additional substantiation. This view can be very helpful when considering additional evidence to increase the overall assurance of a product or an operational system. Providing additional evidence to support an established claim is redundant, unnecessary, and a waste of time and money. Efforts would be better focused on the weaker areas of the assurance argument.

4. CONSUMING ASSURANCE

Deciding assurance questions is not a hard science. Some have argued [WITAT94] that there is a certain "physics envy" in information security. This tendency is reflected in desires for absolute certainty and highly precise numerical calculations of risks.

As Figure 2 illustrates, there is no way to determine perfect assurance. We must choose some level of checking, after which we will have to settle for claims. The situation is reminiscent of a chess-playing computer which must evaluate an enormous decision tree in order to find the best move.

Assessing a hierarchy of claims and evidence can be an intractable task. There must be ways of simplifying the task without losing too much assurance. Using a hierarchy of assurance claims allows pruning of parts of the hierarchy that are not relevant to the information system in question. There are several ways of reducing the search space:

- Establish a standard of evidence. For each part of the hierarchy, claims only need to be examined until the standard of evidence has been achieved.
- Disregard parts of the hierarchy that do not make sense for the information system security policy. For example, a site which assumes that all users are authorized to see all information on a system does not have to worry as much about the "disclosure" part of the hierarchy.
- Use the Flaw Hypothesis Methodology to select and prioritize the parts of the hierarchy to check.

As the assets and threats involved increase, the standard of evidence should also increase. This directly relates to "assurance levels" used today. Establishing a standard of evidence for assurance would help to determine how far "down" an information system needs to be assessed. Stating these standards of evidence is an exceedingly difficult exercise. No matter what standard is chosen, there is the possibility of missing a critical flaw at the next level. The user must accept that a successful security assessment will reduce but not eliminate the likelihood of these critical flaws occurring.

One cannot divorce evidence from the standard of evidence. There are several common examples used in the legal system:

- Substantial Evidence
- Preponderance of the Evidence (more than the evidence against)
- Clear and Convincing Evidence (reasonable person would believe)
- Evidence Beyond a Reasonable Doubt (no reasonable person can doubt)

Further research is needed to determine workable standards of evidence for information technology.

The evaluation process used for trusted products includes generation of new evidence. Some examples are penetration testing and design analysis results. Another potential way to simplify the assessment process would be to rely solely on the developer for evidence. The disadvantage of this approach is that the developers may not remain objective. In an adversarial model of the assessment process, someone must produce counter-evidence.

5. CONCLUSIONS

Because absolute security is impossible, we are forced to make compromises. The idea of "pretty good assurance" is that these compromises can be made in a logical manner. Pretty good assurance looks at the depth and breadth of security required and allows developers and consumers to make tradeoffs. For example:

- Developers can build an assurance case using claims and evidence. The paper lists several ways to ease the development of such a case.
- Consumers can assess the security of information technology against their security needs. The paper lists several ways that this assessment can be simplified without sacrificing all the assurance gained.

In addition, we should continue to look at the new and different methods for substantiating assurance claims. Perhaps there are lessons to be learned from other industries to develop, assess, and convey assurance. It is tempting to look at the reliance placed on Consumer Reports, Underwriters Labs, etc. However, no single approach is likely to produce all the evidence needed to trust a system. Penetrators are not honor-bound to adhere to these conventions, and few consumer products are subject to the forms of updates and mix-and-match/plug-and-play integration that is typical of today's information technology products. Another alternative source is discreet organization(s) that can independently assess products and document their findings in the form of pretty good assurance evidence tied to specific envisioned patterns of use (application) and environment.

A drawback is that developers aren't going to want to offer up their designs and trade secrets as assurance evidence. This will present few problems to prospective buyers of products that provide traditional policies and access control mechanisms, as pretty good assurance can generally be provided by describing the philosophy behind the protection mechanism and a generic description of how it is designed and implemented. This kind informative design documentation was provided to purchasers of the Stac Electronics "Stacker" product. The problems are more difficult in cases where detailed information about the product and its internals are needed for assessment. Certainly, developers and government or institutional users can establish non-disclosure agreements to facilitate the open exchange of information. Perhaps it will be possible for individual citizen users to use the findings of an independent claim-verification agency as pretty good assurance.

The future holds increasing capability and risk for information technology. As ordinary users start to store assets with real value on their information technology, the demand for assurance is sure to continue to rise. By placing pretty good assurance arguments in their hands, they should have the ability to come to informed conclusions about the security worthiness of the products to which they entrust their assets.

BIBLIOGRAPHY

[ITSEC] Information Technology Security Evaluation Criteria, Bonn, May 1990.

[LAND93] Landwehr, Carl E., "How Far Can You Trust A Computer?" 12th International conference on Computer Safety, Reliability, and Security. October 1993.

[SSECMM] Security Engineering Capability Maturity Model Development Workbook, September 1994.

[WILL94] Williams, Jeffrey R., Sachs, Joel E., Landoll, Douglas J., Carpenter, Diann A., "Assurance is an N-Space, Where N is Hopefully Small", International Invitational Workshop on Developmental Assurance, June 26-27, 1994.

[WILL95a] Williams, Jeffrey R., Wichers, David R., "A Framework for Reasoning About Assurance." Workshop on Information Technology (IT) Assurance and Trustworthiness, March 1995.

[WITAT94] National Institute of Standards and Technology, "A Head Start on Assurance," Proceedings of an Invitational Workshop on Information Technology (IT) Assurance and Trustworthiness, March, 1994.

[WITAT95] National Institute of Standards and Technology, Invitational Workshop on Information Technology (IT) Assurance and Trustworthiness, March, 1995.

[TCSEC] DOD 5200.28-STD, "DoD Trusted Computer System Evaluation Criteria." December 26, 1985.

[Toulmin] Toulmin, Stephen. "The Uses of Argument"