

The Need for a Failure Model for Security

Catherine Meadows

Code 5543

Center for High Assurance computer Systems

Naval Research Laboratory

Washington, DC 20375

Researchers in fault tolerance have long made use of the notion of a failure model, which describes the different ways a component in a system can fail. For example, a node can fail quietly (that is, it send out no information), it can fail with respect to timing (that is, send out information too late), it can fail arbitrarily, or it can fail maliciously. The intent of the failure model for fault tolerance is to make it possible to develop different types of algorithms that address different kinds of failures.

Security has traditionally followed a different type of approach. When security is modeled, it is assumed that the system is trying to operate in the face of a hostile adversary with unlimited capabilities in certain areas. For example, the modeling of secure operating systems assumes the existence of Trojan Horse code that is able to signal information along covert channels, while the modeling of secure protocols assumes the existence of an intruder who is able to read and modify all traffic, gain control of nodes, and compromise old secret information. In the actual development of secure systems, such assumptions may be relaxed, of course, but, for theoretical models the most stringent assumptions usually apply.

It is my opinion that it is time to start moving away from worst-case assumptions and begin developing a failure model for computer security. The reasons are two-fold: first, that the theories devoted to the more traditional computer security problems are reaching the stage of maturity where such models are necessary, and that is impossible even to consider the less traditional problems without at least the beginning of such a model.

We begin with the issue of maturity. The concentration on worst-case assumptions is characteristic of a developing theory; one's first goal is to test the limits of theory by applying it to the most severe cases possible. It is also usually paradoxically the case that it is easier to develop theories under worst-case assumptions; worst-case assumptions are usually the simplest to develop or formulate. However, as a field matures, the limitations of this approach become more clear. Theories developed for worst-case assumptions may turn out to be impractical to apply. Moreover, it may also turn out that, as we delve

more deeply into the problem, that it is always possible to extend our notion of a “worst-case” assumption so that our model fails to handle it. Such has been the case in natural language processing, for example; it is always possible to come up with some kind of bizarre formulation that a given theory cannot handle but is that is nevertheless natural language.

My position is that we are beginning to reach this point in the theory of secure systems. This has been particularly noticeable in the recent developments of theories of information flow. For any theory that has been developed, it has been possible to come up with a type of covert channel that can defeat it. This has culminated in theories that are able to handle such subtle notions as probabilistic channels that convey information by varying the probability that events will occur [Gra91].

Such theories are difficult to apply, both because of the difficulty of calculating the probabilities of events in a realistic system, and because of the amount of variables that must be considered. Perhaps at this point it is time to re-evaluate these information flow theories by developing realistic “failure models” for covert channels and evaluating theories in terms of these models.

The second reason for the need for failure models is the changing emphasis of security research. In the past research in secure computer systems has concentrated on secrecy. In theory, as long as one has sound access controls and has eliminated all covert channels, there is no way that secrecy can be compromised in a secure system, even though all code not entrusted with enforcing the security policy is actively attempting to subvert it. This is not the case with other security attributes such as integrity or denial of service. An access control mechanism can decide which processes modify what data items, and in what order, but whether the data items are modified correctly is the responsibility of the processes themselves. Likewise, an access control mechanism can assist a process in providing a service by preventing other processes from denying it accesses to resources such as memory, but it cannot guarantee that that process will actually provide the service. Thus, if we assume that processes are always actively attempting to subvert the security policy will not result in meaningful model of a secure system. On the other hand, we do not have the resources to guarantee that all processes are completely trustworthy. We will need some notion of different degrees of trust, which can be supplied by a failure model. One needs to be able to characterize the different ways that a process can attempt to subvert the security policy, and to have some means of ranking these different kinds of attacks in order of their likelihood.

The failure model we are attempting to build should have two attributes. First of all, it should characterize the different attacks that can be made against the components of a secure system. For example, in a network, we might include both passive and active wiretapping in our failure model. Secondly, it should include some way of ranking the the different kinds of attacks in order of their likelihood.

Much of the work on characterizing failures is spread throughout the literature, and it remains only to pull it together. Indeed, work along these lines has already been done by Landwehr et al. in [LBMC93].

The problem of ranking, however, is more of a challenge. In some cases such a ranking is straightforward. For example, the ability to perform active wiretapping subsumes the ability to perform passive wiretapping. Other cases are not so clear. For example, it is generally believed that timing channels are harder to exploit than storage channels. But there exist high capacity timing channels that are easier to exploit than many low capacity storage channels [Hu91].

Moreover, since we are dealing with an intelligent adversary, the likelihood of a failure can change according to circumstances. For example, as attackers become more experienced and sophisticated, certain attacks may become more likely. On the other hand, as security measures become more sophisticated, certain kinds of failures may become less likely, as attackers turn to other, easier, ways to break into systems. We conclude that any ranking of failures must be nonlinear and allow for overlaps, and such a ranking will probably have to be dynamic. Moreover, the ranking may change, not only from system to system, but during the lifetime of the system itself.

Such a failure model will not be easy to design, and we may have to be satisfied by approximate success. However, it is my belief that even a limited success in this direction will vastly improve our ability to evaluate the success of our attempts to provide security.

References

- [Gra91] J. W. Gray III. Toward a Mathematical Foundation for Computer Security. In *Proceedings of the 1991 IEEE Symposium on Security and Privacy*, pages 21–34. IEEE Computer Society Press, May 1991.
- [Hu91] W. M. Hu. Reducing Timing Channels with Fuzzy Time. In *Proceedings of the 1991 IEEE Symposium on Security and Privacy*, pages 8–24. IEEE Computer Society Press, May 1991.
- [LBMC93] C. E. Landwehr, A. R. Bull, J. P. McDermott, and W. Choi. A Taxonomy of Computer Program Security Flaws with Examples. NRL Report 9591, Naval Research Laboratory, November 1993.