

Position Paper for the Workshop on the Application of Engineering Principles to Information System Security

William Hugh Murray, CISSP

It is recorded that the ancient Roman generals required the engineers to stand under the bridge as the army marched across it.

Introduction

This is a position paper for a workshop. It stakes out a position. It is not and is not intended to be an argument. Rather it is that which will be argued for in the workshop. While I take full responsibility for it, it is to some extent a recapitulation of the discussions among the organizers of the workshop.

This workshop is about the application of engineering principles to the design, building, operation, and maintenance of secure systems and security systems. It is not limited to software or hardware. It is not about “software engineering.” There are many other workshops and forums that deal with these topics. It does not equate the problems of security with those of software quality. While it certainly includes the engineering of reliable multi-user multi-application shared-resource computing systems, it is not limited to these.

Rather it is about the application of tried and tested engineering principles to what we do. I suggest that we continue to ignore these principles at our peril.

The workshop is not so much about how to build secure software but how to build secure systems using available software. It is less about engineering software with fewer security flaws than about how to compensate for the inevitable limitations of that software. It is not so much about what might be learned from comparing Multics to Windows as what can be learned by comparing Intel to Microsoft. It is not so much about how to secure Unix and Windows as about how to use them and when to use alternative operating systems.

I see it very much as a cultural problem. Engineers expect different things of one another. For example, they expect rigor and discipline. They expect training, experience, and maturity. Have you ever heard of a twenty-year-old "engineering expert?" They share common history and common stories. Every engineer is expected to know about the Firth of Tay and the Tacoma Narrows but our “experts” have not read the seminal papers, do not share any history, much less a common one. Many have not read the Cuckoo's Egg, much less can they tell you the three attacks in the Morris Worm. Hell and damn, we do not even have a song.

Therefore, I set forth my position in a number of propositions. While the propositions are less than self-evident, they are at least arguable. While some, not to say many, practitioners will reject them categorically, there is some evidence to support them. There is certainly value to discussing them. Such discussion may even lead to action.

Propositions for the Workshop

Engineering as both an art and a science embraces and consistently applies a set of principles that are intended to ensure the consistency and quality of the results that it produces. These principles pervade both the tradition and culture of everything that engineering does.

IT people in general and security people in particular do not consistently apply traditional engineering principles to their work. Gene Spafford has suggested that they lack knowledge of them. They certainly are not included in the culture in which they work.

Engineers sign their work. They take responsibility for the results that they produce. They do not blame their superiors or their subordinates. They do not blame their tools or their materials. They expect rigor and discipline from their fellows; they reject “shoddy.” Security people excuse their failures. They do so, in part, on the basis that the problem that confronts them is a very difficult one. While the problem that confronts security people is difficult, it is not as difficult as the one that confronts the builders of airplanes. Yet the quality of the results that aeronautical engineers achieve consistently exceeds ours.

Engineers build for the ages. Their work does not fail under its own weight. It does not fall down under normal load. It does not fail in the face of easily anticipated abuse and misuse.

Engineers do not begin work until the requirements of the system are well understood and documented. While they may experiment, they do not do so on production systems. Security people do not do a good job of identifying and expressing the requirements of their systems. This is in part the result of confusing requirements with acceptance or evaluation criteria.

Engineers address requirements as an ordered list. They do not attempt to treat any requirement in isolation. They would not attempt to design for security in the absence of the requirements for function, performance, or generality of application.

Engineers study the strength and limitations of the materials that they use. They exploit the strengths of those materials and compensate for the limitations. They do not use materials for purposes for which they are not fit. By contrast, security people prefer popular materials (e.g., Unix and Windows) in applications for which they are patently unfit.

Engineers prefer simplicity. They avoid gratuitous complexity and even functionality. For example, engineers understand that demonstrability is achieved at the expense of

functionality and generally prefer the former. We tend to prefer functionality at the expense of almost everything.

Engineers do not mix the controls intended for the exclusive use of management with those intended for end users. An engineer would not think of replicating the controls for the autopilot of a 747 between every two passenger seats and then labeling it with a placard that says, "Please do not touch." Security people replicate the controls intended for the exclusive use of management right next to those intended for users and then whine about how rude the users are when they insist upon playing with them.

Engineers prefer the reuse of proven designs. They introduce novel designs conservatively. For example, while it has been shown that airplanes can have their wings both for and aft, most airplanes have wings attached at the center. By contrast, most security designer reserve the right to re-engineer logon even at the risk of repeating the errors of earlier designers.

Conclusion

Engineering is a mature discipline. While it has only had that name for about 200 years, it has been around for millennia. Many of its artifacts have stood for millennia. We are puerile, not yet even adolescent. That we are not as mature as engineering should come as a surprise to no one, including us.

Unfortunately we do not have millennia available in which to mature. We must accelerate the pace. One way to do that is to embrace the traditions and principles of engineering. While they may not cure all our ills, they are not likely to misdirect our efforts. These traditions and principles include accepting responsibility, not simply for our process but also for our results. They include complete and ordered requirements identification and documentation. They include a clear preference for simplicity, structure related to functionality, modularity, proven components, standard interfaces, redundancy, demonstrability, early testing, appropriate materials, early binding, *inter alia*.