

Can Critical Information Infrastructure Protection be Achieved With Untested Software?

J. Voas (jmvoas@rstcorp.com)

Reliable Software Technologies, Sterling VA USA

Citizens of modern societies need access to information infrastructures that are reliable, secure, non-interruptible, and fault-tolerant. In October of 1997, President Clinton's Commission on Critical Infrastructure Protection (PCCIP) announced that the United States's infrastructure, which is responsible for relaying information and communications, is vulnerable to information warfare attacks [2]. The commission found that while the resources needed to conduct a physical attack against the infrastructure have not changed dramatically, the resources necessary to launch a comparable scale attack via information warfare are commonplace. They simply consist of a personal computer and Internet connection. Furthermore, the ubiquity of Internet access and the plethora of "hacker" tools and recipe attacks on "underground" Internet sites have significantly reduced the barriers to launching effective attacks against critical systems.

With roughly 95% of Defense Department communications relying on commercial infrastructure, the US government finds itself as a major stakeholder in the security of commercial systems [1]. Financial organizations are also heavily dependent. Wholesale payment systems such as the Federal Reserve's FedWire and automated clearing houses move trillions of dollars over electronic networks daily [3]. Further, as societies transition to paper-less commerce, individual privacy is threatened with each transaction.

Software is at the heart of modern information and communication infrastructures. Trust in the integrity of the infrastructure requires a high degree of trust in the underlying software. Software trust, however, has increasingly become a disappearing commodity. We are bombarded daily with news stories of incidents that can be tied directly to defective software.

Software trust is a "quality" issue. Software users must trust that the software meets their requirements, is available, reliable, secure, and robust. When a particular software system has these properties, most would agree that the software is of "high quality." But honestly, how many systems in use today have all of these characteristics?

In this column, I will argue that a research initiative focused on testing "systems of systems" must occur if we expect to achieve a comfortable level of infrastructure protection. In fact, in 1992, Clarke and Osterweil called for a similar initiative (except that their reasoning at that time was different than infrastructure protection and did not focus on "systems of systems"). They

wrote an article titled “A Proposed Testing and Analysis Research Initiative” in *IEEE Software* urging the government to make software testing a priority for research funding. As we know, that never occurred.

To begin, protection of the National Information Infrastructure is an admittedly daunting challenge. Fortunately, however, protection is not an “all or nothing proposition.” There are degrees of protection, and even small advances (pragmatic and theoretical) are beneficial to this challenge.

The core problems facing those in the government who are charged with this responsibility are three-fold: (1) determining the domain (scope) of the National Information Infrastructure, and (2) developing technical solutions to threats that are created by malicious individuals and organizations, and (3) developing technical solutions to threats that occur as a result of non-malicious anomalies (e.g., software errors, hardware failures, natural disaster, etc.). A focused research initiative in software testing can address items (2) and (3). Item (1) is a problem that needs to be handled separately.

Possibly the greatest problem facing the software industry stems from our inability to quantitatively assess software quality behaviors. Most assessment techniques are qualitative and focus on process assessment. A software testing research initiative can make a significant impact here if it focuses on testing methods that provide quantitative measures of survivability, security, etc., and will do so for megalithic systems. We must be realistic, however. Such an initiative must begin using small systems. This problem will not be conquered quickly.

There are several different terms being thrown around that all relate to the challenges of information infrastructure protection. Terms such as “cyber terrorism”, “information warfare”, “information assurance”, “information superiority” are excellent attention grabbers and head-liners. I, however, prefer the less flashy term of information assurance.

Information assurance encapsulates a broad set of information integrity issues: fault tolerance, reliability, safety, security, privacy, confidentiality, and availability. If an information system can thwart attacks, whether malicious in origin or simply unfortunate, and still provide “accurate enough” information on demand, then information assurance has been achieved. Put simply, information assurance is accurate enough information that is available on demand for a given application or situation.

While all of the “ilities” are important, I believe that the three key software “ilities” that must exist in order to provide any confidence in the information infrastructure are “security,” “fault tolerance, and “survivability.” But in order to provide this confidence, we must have believable data that demonstrates that a system or subsystem is: (1) *secure* (i.e., the system is extremely difficult to break into), (2) *fault-tolerant* (i.e., the system will continue to function even if key components of the system fail), and (3) can *survive* malicious attacks (this includes common attack scenarios that are well known and more importantly attack scenarios that are new and have never been seen before).

The “ilities” are an interesting mix of complimentary and conflicting behaviors. It is nearly impossible to build a simple system with a high degree of one “ility” (let alone three). Now consider doing so for system like the Internet that evolves continually.

The reader might be asking why the field of software testing theory has not kept up with the

pace and scale at which systems are now built. The answer is quite simple: software testing suffers from a variety of theoretical problems. Since they cannot be solved, tackling testing's practical problems is difficult to justify. For example, exhaustive "black box" testing of toy programs is often infeasible. If you cannot do a decent job of testing small programs, then it is obvious that you cannot do a decent job of testing huge programs. Further, the culture has nurtured a belief that large-scale systems cannot be tested to any degree of thoroughness. Therefore the automated tools out there are designed for small code bases.

This is particularly bad news for the information infrastructure. Further, the domain of the information infrastructure is far larger than what is typically classified as a large-scale system. The information infrastructure is a megalithic system.

And there are other problems related to leveraging existing software testing theories. "Off nominal" testing approaches (such as fault injection) could potentially be useful for survivability testing on the complete infrastructure but tools for doing so do not exist. And white box software testing theories that require source code are of no use for this problem. (White-box testing techniques are applicable to new software projects; it is foolish to think we could go back and begin testing all of the source code of the infrastructure.)

The key software testing research that is needed is: *integration testing of heterogeneous subsystems*. The infrastructure is composed from parts (hardware and software) supplied by thousands of different vendors that have known and unknown incompatibilities. I do not mean to suggest that we do not also need advances in how to test subsystems in isolation. That work also remains. But ultimately, we must be able to test "systems of systems of systems", and that is an integration testing problem. To do so will require: (1) testing theories that scale and address security, survivability, and fault tolerance, and (2) efficient tools. And given that the industry is moving toward a system integration paradigm for how new developments are architected, the results of this research can benefit systems other than the infrastructure.

In summary, as societies become electronically networked, the privacy and integrity of information becomes paramount since the threat of information attacks looms ever larger. Software testing is critical to every aspect of information infrastructure protection. The software testing problems that face us today are the same problems that have remained unsolved for 40 years. If these problems cannot be lessened, there is no doubt in my mind that information infrastructure protection can *never be guaranteed*. Admittedly, solving these testing problems still does not guarantee protection, it does provide ways to know the strength of the existing protection. Knowledge concerning infrastructure strength must be known to our Nation's leaders so that appropriate defensive plans can be formalized. But until the US government takes a *pro active* role in developing technologies that quantitatively assess information vulnerability, that knowledge will be null.

References

- [1] B. Graham. Lack of disclosure impedes development of safeguards. *The Washington Post*, page A6, February 28 1998.

- [2] PCCIP. Critical foundations: Protecting america's infrastructures. Available online. See http://www.pccip.gov/report_index.html., October 1997.
- [3] M. Sirbu. Credits and debits on the internet. *IEEE Spectrum*, 34(2):23–29, February 1997.