Common Techniques in Fault-Tolerance and Security

Karl N. Levitt & Steven Cheung
Department of Computer Science
University of California, Davis
Davis, CA 95616
levitt@cs.ucdavis.edu

Panelists: Yves Deswarte, Catherine Meadows, Peter Neumann, Brian Randell, Kent Wilken.

It is our opinion that the security community should draw on the techniques developed by the fault-tolerance community to a significant extent; presently, there seems to be less opportunity for transfer in the other direction. The fault-tolerance community has given significant attention to the development of systems where reliability is a key requirement which pervades all aspects of the design. Security, although clearly a requirement, is often achieved through retrofitting, e.g. intrusion detection, and firewalls.

To motivate a serious attempt to explore the utility of the fault-tolerance paradigm to security, we propose a system to significantly improve the security of large networks. The system will provide a capability to deal with a variety of attacks on network components, e.g. on user accounts, hosts, and routers. It will provide a capability to detect attacks through real-time detection (e.g. intrusion detection) and diagnosis. In addition, it will provide a capability to halt the spread of attacks (e.g. through dynamically settable firewalls). And, it will provide a capability to recover to a secure state. Thus, this system embodies most of the steps common to reasonably complex, albeit state-of-the-art fault tolerant systems.

The attached table is an initial attempt to provide security counterparts to the most common fault-tolerance terms. A few issues bear some discussions.

Is there a conceptual difference between the threat to security and that to reliability? Catherine Meadows speaks in more detail to this question. The expansionist's view is that security is just another subarea of Dependable Computing, since it deals with just another manifestation of faults. However, the exact nature of the threat to security deserves a closer look. The threat to security is usually a human or processes (or programs) that trace their ancestry to humans. Being human in origin, the security threat can adapt

Basics

Fault Tolerance	Security
Faults	Vulnerabilities (e.g. software bugs, improper system setups)
Latent faults	Time bombs, Trojan horses
Intermittent faults	Race conditions
Errors	Exploitation of vulnerabilities (intentionally or accidentally)
Failure	Violation of policies
Random/accidental in nature (except Byzantine faults)	Malicious/intentional in nature
Independent fault assumptions	Correlations among attacks (e.g. trusts among hosts allow attacks to spread)

Evaluation

Fault Tolerance	Security
Reliability	Mean time between break-ins, covert channel capacity
Availability	Denial of service

Techniques

Fault Tolerance	Security
Redundancy, majority voting	Sending packets over multiple communication paths, storing critical files in more than one sites, using multiple servers for authentication
Error detection or correcting codes	Cryptography
Heterogeneity (e.g. N-version programming)	Having heterogeneous hosts and routers which run different communication protocols; cost: standardization of protocol and OS
Error containment	Access control, firewalls, diversity

Diagnosis/ Detection

Fault Tolerance	Security
System diagnosis (e.g. active probing for faults)	ISS, COPS, Netsweep, NSM, IDS, anomaly and misuse detection, auditing, testing or monitoring by site administrators, virus scanners, integrity checking

Response

Fault Tolerance	**Security**
Reconfiguration	Disconnecting a host from the net, adding rules to the packet filter to block packets from suspicious hosts, routing packets around suspicious routers
Recovery from failure (e.g. recovery blocks)	Recovery from attacks (e.g. reload a clean copy of the routing table to a compromised router, changing passwords).

as it attempts to thwart the defenses launched against the threat. This situation does not arise in the fault-tolerance world, except that the entire motivation for Byzantine agreement is the hypothesis that faults can have malicious manifestations, and it might be prudent to study this model in connection with security. There seem to be security threats that do not correspond to faults studied by the fault-tolerance community. As Fred Cohen has noted, it is unlikely that a computer virus could accidentally appear in a program. On the other hand, a common malicious program that causes denial of service in some Unix systems could be accidentally introduced: a program that claims all of the system's process table slots by creating processes faster than they are killed.

Can security requirements be expressed probabilistically? Most security requirements are expressed as negative properties, for example, files are not to be released to unauthorized users. Generally, the requirement is not expressed as the probability of unauthorized release of a file should be less than some very small number, mainly because there is no clear approach to determine if such a requirement is realized by the underlying system. A counterexample to the "assertion" approach to security is the information-theoretic approach used to compute the bandwidth of covert channels.

Error Containment: Redundancy is effective only if the likely faults affect a single protected component, e.g., a processing element. Moreover, fault-masking prevents the fault from inducing errors that propagate beyond the component that suffered the fault. There seems to be a related concept in the security domain. If a computer on a network is compromised by an attacker, it should be difficult for him to use this compromised machine as a base to attack other machines. (Of course, the Unix concept of trust among systems in a singly administered domain is a counterexample to this principle, favoring ease of use over security.) Access control mechanisms and firewalls associated with network components can block or at least limit the spread of attacks.

Recovery Blocks, Diagnostic Routines: The technique of recovery blocks rests on the effectiveness of acceptance test in detecting errors, due to hardware or software faults. For the most critical applications, recovery blocks would not be used as it is impossible to make precise statements about the effectiveness of acceptance tests. However, the increasingly promising technique of intrusion detection involves comparing a user's (or a computer's or a network's or entire system's) current activity with expected behavior. If the comparison is with respect to the user's previous activity, then it is *anomaly detection* that is being implemented. If the comparison is with attack profiles, then it is *misuse detection*. In either case, it is the strength of the comparison algorithms that determines the effectiveness of the technique. Moreover, the operating system itself supplies audit logs that constitute the raw data for the comparison. When the system-generated audit trails are at too low a level of abstraction to enable effective detection, it is incumbent on the application programs to generate audit logs appropriate to the system's security requirements. Particular problems are what support the operating system should provide for application-specific auditing and can the audit trails be made tamperproof? Also yet to be seriously considered are the issues of security-related audit trails for server-based operating systems (such as Mach) and for network components (such as routers, name servers, file servers, etc.) Feather et al.'s paper uses security-like techniques, anomaly and misuse detection, to locate faulty components on Ethernets.

The fault-handling cycle: In an advanced fault-tolerant system, the handling of a fault can involve the following steps: error detection, damage assessment, reconfiguration, and recovery. Current attempts at security are much less ambitious, typically relying on prevention or on attack detection that triggers a message to be sent to a security officer; thereafter it is the responsibility of the human security officer to deal with the situation, e.g., to remove an offending user or site, to request additional audit logs for a particular user, and to save audit logs as evidence. We envision that human intervention at this level will not be possible for much longer, particularly when attacks are on a network's infrastructure or have the potential for rapid propagation, such as the Internet worm. The intrusion detection system of the future will be much like the alluded to advanced fault-tolerant systems. Just as a fault-tolerant system might contain fault handling sites (that effect fault masking, fault analysis, reconfiguration, and recovery — all carried out through communication among the sites), the intrusion detection system should have intrusion management components distributed across a large network with a protocol for exchanging information among the sites that impact the network's security.

Redundancy: Yves Deswarte has been considering the problem of dispersing the blocks of a file among sites. The goal is to protect against unauthorized release of files due to a single site being subverted. The similar problem of avoiding unauthorized access to a message in transmission can be solved by the decomposition of the message into blocks that are guaranteed to be dispersed among different routes between message source and destination. Of course, encryption for confidentiality, including threshold schemes, can be used in conjunction with dispersal.

Here is a final question that bears on the analysis of a system with respect to a reliability or security goal.

Can the reliability or security performance of a system be degraded by fault-tolerance or security measures? In the case of fault-tolerance the answer is yes. A fault-tolerant system contains more components than its non-redundant counterpart, and if the redundancy is not carefully managed, the result could be a less reliable system. A similar situation can exist when measures are taken to improve security. Tables holding information on the security state of a system can be used by an attacker to identify vulnerable sites. Also, a secure system might include mechanisms to logically disconnect sites that have been determined to be subverted; an attacker could use these mechanisms to disconnect sites that might expose him or even to disable quickly an entire subnetwork.