Making Real-time Systems Survive Malicious Attacks

Partha Pal, Joe Loyall, Franklin Webber, Rick Schantz
BBN Technologies
10 Moulton Street
Cambridge, MA 02138
{ppal,jloyall,fwebber,schantz}@bbn.com

A computer application that assumes a lot about its operating environment has a disadvantage against cyber attackers that is not shared by those that assume less. Manipulation of the environment to violate the assumptions made by the application presents a vulnerability that an intelligent, malicious adversary can exploit. If correct functioning of the application depends on a set of assumptions (e.g., a message from A must arrive at B within *t* ms), attacks on the environment can cause the application to fail in ways that may be hard to detect and diagnose. The stronger the assumptions, the greater is the risk that an attacker will be able to exploit it to succeed without the need to understand or break the application's interface or functionality. Real-time[1] systems typically have strong dependence on their environment (e.g., threading and scheduling), and real-time requirements are usually met only under specific assumptions. Furthermore, real-time requirements provide another opportunity for attackers—in order to succeed, they only need to degrade the (real-time) QoS aspects of the system rather than completely disrupting or corrupting the functionality. For instance, delaying a radar track can be as disruptive as the more difficult attack of actually corrupting it.

As more systems with real-time requirements become interconnected and/or connected to the Internet, we observe the following:

- New attack paths and attack targets become available to the attackers because of the added remote accessibility, and
- The interconnection introduces new dependency, sometimes on remote entities that essentially act as implicit assumptions that are more easily breakable, putting the correct operation of the system at risk.

Because real-time systems often are part of critical infrastructure (such as the national grid) or related to national security (such as missile defense), it is important that we consider this threat seriously. The implication is clear—systems with real-time requirements are more vulnerable, the adversary can more easily exploit the vulnerability, and the resulting damage can have significantly more impact.

Recent advances in survivable information systems of the non-real-time kind encourage us to ask whether we can build real-time systems that can survive malicious attacks. We know from past experience that complete attack prevention is pretty much impossible—some attacks will occur, and may not even be detected on time. A more realistic goal is to try to limit their success and deal with their effects. The real question then is: "Can we

[1] A *real-time system* is any system that has real-time requirements and ensures they are satisfied, given some assumptions about its environment. A *real-time requirement* sets a time limit for completing an operation.

prevent attacks against such systems from achieving total success, and tolerate the attacks that succeed sufficiently to continue critical operations?

To explore this idea a bit more, let us consider several scenarios. If a system is designed with slack in its deadlines or includes over-provisioned resources, it may be possible to *tolerate* the impact of the failure of task T assigned to processor p1 by restarting T on a different processor p2. It will still be possible to meet T's deadline because the processor p2 is very fast, lightly loaded and has a high speed bus. However, merely maintaining T's deadline may not completely mask the impact of T's failure. In a system in which recovery from attack-induced failures cannot be completely masked, the system might be able to temporarily transition into a down state or stop satisfying real-time requirements, if it can be guaranteed to remain in that state for a short, bounded period of time. In some systems, it might be possible to provide statistical guarantees, e.g., the probability that the system will not recover within time *t* is very low.

Providing high levels of survivability and real-time performance can conflict in some systems-especially in terms of resource requirements. Solutions will need to trade off the relative importance of each and provide the ability to adjust to the requirements and conditions over time. In some cases, survival is the primary concern, even with degraded real-time performance. In other cases, guaranteeing real-time performance might be worth sacrificing some protection from cyber attack, e.g., when the threat posed by operational adversaries (such as incoming missiles) is greater than that posed by cyber adversaries.[2] Given this, we ask: what design principles and technology enablers are necessary to build such systems? And once built, how does one test and validate such a system?

There is research to build upon in the area of fault tolerant real-time systems. However, it is worthwhile to remember that the probability of multiple simultaneous accidental failures in a redundant system may be low, but may become close to certain shortly after the discovery of a new common vulnerability.

In many ways, the research community is well positioned to explore the intersection of survivability and real-time systems. Advanced middleware frameworks such as QuO [1] have been used to build a number of real-time QoS-aware adaptive distributed applications. Advances in the area of information assurance, intrusion tolerance and survivable systems recently produced a new high-water mark in survivable system design and implementation in the form of a survivable combat information management system that withstood multiple hours of attacks by sophisticated Red Teams who were given full knowledge, and often inside access to the defended system [2]. State-of-the-art research in real-time performance in networked embedded system has been demonstrated in a number of high-profile flight demonstrations using applications in the US Air Force and Navy domains [3, 4, 5]. It is certainly the time when one can envision developing survivable real-time systems.

---

[2] For example, a cooperative dynamic defense could employ defenses against cyber attacks up to the instant that anti-missile defense systems need to be engaged, at which time the cyber defense could shut off (or be dialed down) and its resources be rapidly reallocated to the missile defense system.

However, based on our current level of understanding, we feel that significant progress in that direction requires a larger scale initiative. Before such substantial and coordinated research is launched, we will do well to take a few smaller steps to study issues and sub-problems and deepen our understanding of the problem space. Examples of such steps, whose exploration can help define the future research direction, are as follows:

- Elimination of strong dependency on the environment is unrealistic for real-time systems, but can the application be shielded from malicious manipulation of its environment?
- Similarly, real-time requirements cannot be discarded or compromised, but can the adversary be denied the privilege of succeeding just by breaking the real-time QoS?

Both questions point to further exploration of redundancy, diversity, adaptive response and modularized design in the context of real-time requirements.

Finally, we also note that the confluence of real-time and survivability is not only important for making real-time systems survivable. Real-time attributes, in particular timeliness of defensive response, and the decision-making process underlying it, is critical for adoption of survivability technologies in real-life systems. Often times, the systems that need a high level of survivability, such as military information systems, run or support time-sensitive “missions”. In such a context, defensive responses must also be time-sensitive. Many of the current survivability mechanisms do not address the timeliness issue. Incorporation of real-time techniques and methodologies into defense mechanisms can help address that shortcoming.

References

[1] QuO Website at http://quo.bbn.com

[2] Lee Badger, DARPA Weekly Activity Report, Dec 23, 2005- Jan 6, 2006

[3] J. Loyall, et al., “A Distributed Real-Time Embedded Application for Surveillance, Detection, and Tracking of Time-Critical Targets”, Real-Time and Embedded Technology and Applications Symposium (RTAS), pp. 88-97, March 2005

[4] Christopher D. Gill, et al., “Integrated Adaptive QoS Management in Middleware: A Case Study”, Real-Time Systems, Springer Science+Business Media B.V., Vol. 29 No. 2-3, pp. 101-130, March 2005

[5] Roy Cambell, et al., “Toward an Approach for Specification of QoS and Resource Information for Dynamic Resource Management”, 2nd RTAS Workshop on Model-Driven Embedded Systems (MoDES), May 2004