Customizable and Adaptive Survivability

An SOS 2004 Position Paper

Matti Hiltunen and Richard Schlichting
AT&T Labs–Research
180 Park Avenue
Florham Park, NJ 07932
hiltunen/rick@research.att.com

A survivable system is one that is able to continue providing service in a timely manner even if significant portions are incapacitated by attacks or accidents [1]. Survivability builds on research in security, reliability, fault tolerance, safety, and availability, as well as the combination and interaction of these different properties [16]. Our thesis is that facilities for *customization* and *dynamic adaptation* are necessary to build survivable systems. Customization is needed, since different techniques used to ensure survivability attributes differ both in their coverage and performance overhead; for example, more resources are needed to tolerate f Byzantine failures than f crash failures. Similarly, the coverage requirements vary based on the characteristics of the services provided by the system; for example, a web site can be down for few hours for maintenance, but the telephone system can only be down for a few minutes a year. Customization allows the coverage of types and number of failures and attacks to be traded off against the resources required and the runtime performance overhead. Adaptation allows a system to change execution behavior dynamically as these tradeoffs change, as well as to react to actual or anticipated changes in the execution environment.

Our prior work on configurable and adaptive distributed protocols and services has addressed a number of the above issues [10, 11]. Specifically, we have developed services with customizable fault tolerance, security, and timeliness properties, e.g., [3, 14, 7, 8]. On the issue of coverage vs. cost, we have investigated services in which fault tolerance can be customized for failure models ranging from crash to Byzantine [9] and in which runtime adaptation can be used to alter behavior based on a change in the failure model exhibited by the underlying system [2]. Additionally, some of our work has been done in the context of survivability and presented in that community [12, 15, 13]. In the remainder of this position paper, we highlight two specific areas in which our current work addresses survivability issues.

Redundancy and Diversity

Survivability has many parallels with fault tolerance, and as such, many of the techniques used for fault tolerance can be adapted for survivability. Redundancy in fault tolerance usually takes the form of either *space redundancy* such as replication of data or computation, or *time redundancy* such as repeated execution or repeated message transmission. Redundancy that has proven useful for survivability includes techniques such as *layered protection*, e.g., encryption of critical files to provide protection in case file system security mechanisms are compromised [5], and *data fragmentation and replication* [4, 6]. In our recent work, we have explored redundancy in the form of *redundant methods* [14]. With redundant methods enforcing a given security attribute such as privacy or integrity, the attribute should remain valid if at least one of the methods remains uncompromised. For example, to tolerate an attack against a public key based PKI, a service might use two completely different authentication mechanisms (e.g., PKI and Kerberos).

While redundancy can be a useful survivability technique, as with fault tolerance, its effectiveness depends on the details of how it is used. One important goal is maximizing the independence of the redundant

elements, where two elements A and B are independent if compromising A provides no information that makes it easier to compromise B, and vice versa. For example, if two independent methods m_1 and m_2 are used to authenticate a user, breaking m_1 does not make it easier to break m_2. A simple example of non-independence is when two encryption methods use the same key, since if one method is broken or the key stolen, privacy is compromised. This type of independence is very much analogous to the fault-tolerance concept of independent failure modes for redundant hardware or software components. Components are independent in this sense when the failure of one component does not affect the correct execution of any other component.

While in fault tolerance, it is often feasible to argue that failures— in particular, hardware failures—can be relatively independent, this assumption does not hold for survivability. Specifically, if an intruder successfully uses some method on one machine, they will probably try the same method on the next attacked machine. The key for survivability is *artificial diversity*, that is, making the system components (e.g., machines, servers, routers) different enough that the same attack is not likely to succeed in all (or many) of them. Diversity can be introduced at different levels ranging from the hardware and operating systems to the applications and security policies. Each of these levels have their issues and limitations. For example, N-version programming for application level diversity may be too expensive.

We are currently exploring *local interface diversity* as a possible means for creating low cost artificial diversity. Specifically, we are working on *system call scrambling* as a method to make each installation of the scrambled operating system different from another. We customize each operating system instance by reassigning all the system call numbers using a deterministic function and a unique key. All application programs that are to run on this operating system instance are then processed using a binary rewriting system that replaces the original system call numbers with the "scrambled" ones. The same technique can also be applied at the library call level and, in principle but probably not in practice, on the user command level. As a result, it is very difficult for an attacker to generate an attack code that will compromise one of these machines and, maybe more importantly, the same attack will not work on another such machine.

Probabilistic Cost-Benefit Decision Making

Detecting intrusions and attacks can be very difficult. The current intrusion detection systems (IDSs) typically generate large numbers of false positives (and false negatives). This implies that a survivable system is often in a state of uncertainty concerning if it is under attack or not. Furthermore, even if it is likely that the system is under attack, it is often hard or impossible to tell which activity is part of the attack and which is part of normal system load. Finally, it is often hard to quantify the effect of a survivability technique on the attack since adversaries constantly develop new attack techniques that can invalidate previous assumptions about the level of effectiveness of a specific technique.

Given these uncertainties, a survivable system may have to rely on probabilistic cost-benefit analysis in making decisions on actions to be taken. Specifically, based on prior experience, it may be possible to estimate the probability that there really is an attack when the IDS reports an attack (= true positive) or that there is no attack when the IDS reports no attack (= true negative). Also, it may be possible to estimate the cost and benefit of action and inaction when there is an attack, etc. Given these estimates, the system may make decisions that attempt to minimize the overall cost or maximize the overall benefit even though the system state is only known probabilistically.

We believe such analysis and algorithms would be useful for fault tolerance as well since they allow explicit trading off between detection accuracy and detection delay. For example, consider a distributed system where a node A is waiting for a reply from another node B. If the cost of making a false failure detection and the cost of waiting are known and the probability of the failure of B can be estimated, node A can decide when to stop waiting based on the likely minimum cost.

We are currently working on formalizing such a model, developing decision heuristics, and validating

the model by mapping the problems of distributed denial of service attacks (DDOS), virus attacks, and traditional component failures in distributed systems into this model.

References

[1] M. Barbacci. Survivability in the age of vulnerable systems. *IEEE Computer*, 29(11):8, Nov 1996.

[2] I. Chang, M. Hiltunen, and R. Schlichting. Affordable fault tolerance through adaptation. In J. Rolin, editor, *Parallel and Distributed Processing, Lecture Notes in Computer Science 1388*, pages 585–603. Springer, Apr 1998.

[3] R. Das, M. Hiltunen, and R. Schlichting. Supporting configurability and real time in RTD channels. *Software: Practice and Experience*, 31(12):1183–1209, Oct 2001.

[4] Y. Deswarte, J.-C. Fabre, J.-M. Fray, D. Powell, and P.-G. Ranea. Saturne: A distributed computing system which tolerates faults and intrusions. In *Proceedings of the Workshop on Future Trends of Distributed Computing Systems in the 1990's*, pages 329–338, Hong Kong, Sep 1988.

[5] J. Fraga and D. Powell. A fault and intrusion-tolerant file system. In *Proceedings of the IFIP 3rd International Conference on Computer Security*, pages 203–218, Dublin, Ireland, 1985.

[6] J. Fray, Y. Deswarte, and D. Powell. Intrusion-tolerance using fine-grain fragmentation-scattering. In *Proceedings of the 1998 IEEE Symposium on Security and Privacy*, pages 194–201, Oakland, CA, Apr 1986.

[7] J. He, M. Hiltunen, M. Rajagopalan, and R. Schlichting. QoS customization in distributed object systems. *Software–Practice and Experience*, (33):295–320, 2003.

[8] J. He, M. Hiltunen, and R. Schlichting. Customizing dependability attributes for mobile service platforms. In *Proceedings of the 2004 International Conference on Dependable Systems and Networks*, Jun 2004.

[9] M. Hiltunen, V. Immanuel, and R. Schlichting. Supporting customized failure models for distributed software. *Distributed Systems Engineering*, 6:103–111, 1999.

[10] M. Hiltunen and R. Schlichting. An approach to constructing modular fault-tolerant protocols. In *Proceedings of the 12th Symposium on Reliable Distributed Systems*, pages 105–114, Princeton, NJ, Oct 1993.

[11] M. Hiltunen and R. Schlichting. A model for adaptive fault-tolerant systems. In K. Echtle, D. Hammer, and D. Powell, editors, *Proceedings of the 1st European Dependable Computing Conference (Lecture Notes in Computer Science 852)*, pages 3–20, Berlin, Germany, Oct 1994.

[12] M. Hiltunen, R. Schlichting, and C. Ugarte. Survivability issues in Cactus. In *Proceedings of the 1998 Information Survivability Workshop*, pages 85–88, Orlando, FL, Oct 1998.

[13] M. Hiltunen, R. Schlichting, and C. Ugarte. Using redundancy to increase survivability. In *Proceedings of the Third Information Survivability Workshop (ISW-2000)*, pages 79–82, Boston, MA, Oct 2000.

[14] M. Hiltunen, R. Schlichting, and C. Ugarte. Enhancing survivability of security services using redundancy. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN 2001)*, pages 173–182, Gothenburg, Sweden, Jul 2001.

[15] M. Hiltunen, R. Schlichting, C. Ugarte, and G. Wong. Survivability through customization and adaptability: The Cactus approach. In *DARPA Information Survivability Conference and Exposition (DISCEX 2000)*, pages 294–307, Hilton Head, SC, Jan 2000.

[16] J. Voas, G. McGraw, and A. Ghosh. Reducing uncertainty about survivability. In *Proceedings of the 1997 Information Survivability Workshop*, Feb 1997.