Survivable Network System Analysis: A Case Study

Robert J. Ellison, Richard C. Linger, Thomas Longstaff, and Nancy R. Mead

Carnegie Mellon University

ABSTRACT

Survivability is receiving increasing attention as a key property of critical systems. Survivability is the capability of a system to fulfill its mission, in a timely manner, in the presence of attacks, failures, or accidents. We present a method for analyzing survivability of distributed network systems and an example of its application. Survivability requires system capabilities for intrusion resistance, recognition, and recovery. The Survivable Network Analysis (SNA) method permits assessment of survivability at the architecture level. Steps in the method include system mission and architecture definition, essential capability definition, compromisable capability definition, and survivability analysis of architectural softspots that are both essential and compromisable. Intrusion scenarios play a key role in the analysis. SNA results are summarized in a Survivability Map that links recommended survivability strategies to the system architecture. The case study summarizes application of the SNA method to a subsystem of a large-scale, distributed healthcare system.

Keywords

Survivability, network systems, essential services, Survivability Map, architecture analysis, intrusion scenarios

1. network system survivability

Survivability Concepts

Modern society increasingly depends on large-scale networked systems to conduct business, government, and defense. Survivability of these systems is receiving increasing attention, particularly critical infrastructure protection.[1] As part of its Survivable Systems Initiative, the CERT® Coordination Center (formerly Computer Emergency Response Team) of the Software Engineering Institute at Carnegie Mellon University is helping to foster a survivability research community[5] and developing technologies to analyze and design survivable network systems.[2–4] Survivability analysis helps identify the essential functions which must survive attacks and failure, the effects of attacks and failures, the associated risks, and the architecture changes which could improve system survivability.

We define *survivability* as a system's capability to fulfill its mission (in a timely manner) in the presence of attacks, failures, or accidents. Unlike traditional security measures that require central control and administration, survivability addresses highly distributed, unbounded network environments with no central control or unified security policy. The focus of our survivability research is on delivery of *essential services* and preservation of *essential assets* during attack and compromise, and timely recovery of full services and assets following attack. We define essential services and assets as those system capabilities that are critical to fulfilling mission objectives. Survivability in the presence of attacks depends on three key system capabilities: *resistance, recognition*, and *recovery*. Resistance is a system's capability to repel attacks. Recognition is the capability to detect attacks as they occur and to evaluate the extent of damage and compromise. Recovery, a hallmark of survivability, is the capability to maintain essential services and assets during attack, limit the extent of damage, and restore full services following attack. As an emerging discipline, survivability builds on existing disciplines, including security,[6] fault tolerance,[7] and reliability,[8] and introduces new concepts and principles.

In this article, we focus exclusively on attacks, although our trace-based, compositional Survivable Network Analysis method applies to analysis of failures and accidents as well. In contrast to runtime patching in reaction to survivability problems, the SNA method focuses on systematically designing survivability into systems during development and evolution. A small team of evaluators, typically three or four people, can apply the method to an existing or proposed system. Evaluators must be knowledgeable in a number of disciplines, including architecture analysis, intrusion techniques, and survivability strategies.

2. The Survivable Network Analysis method

Figure 1 depicts the SNA method's four steps. To use the method, a team reviews mission objectives and requirements for a current or candidate system and elicits the structure and properties of its architecture in Step 1. In Step 2, the team identifies essential services and assets, based on mission objectives and the consequences of failure. *Usage scenarios* characterize essential service and asset uses. These scenarios are mapped onto the architecture as execution traces to identify the composition of corresponding *essential components*, which must be available to deliver essential services and maintain essential assets. In Step 3, the team selects *intrusion scenarios* based on the system environment and an assessment of risks and intruder capabilities. CERT's extensive knowledge base of intrusion strategies also influences selections.These scenarios are likewise mapped onto the architecture as execution traces to identify corresponding compositions of *compromisable components*, or components that intrusion could penetrate and damage.

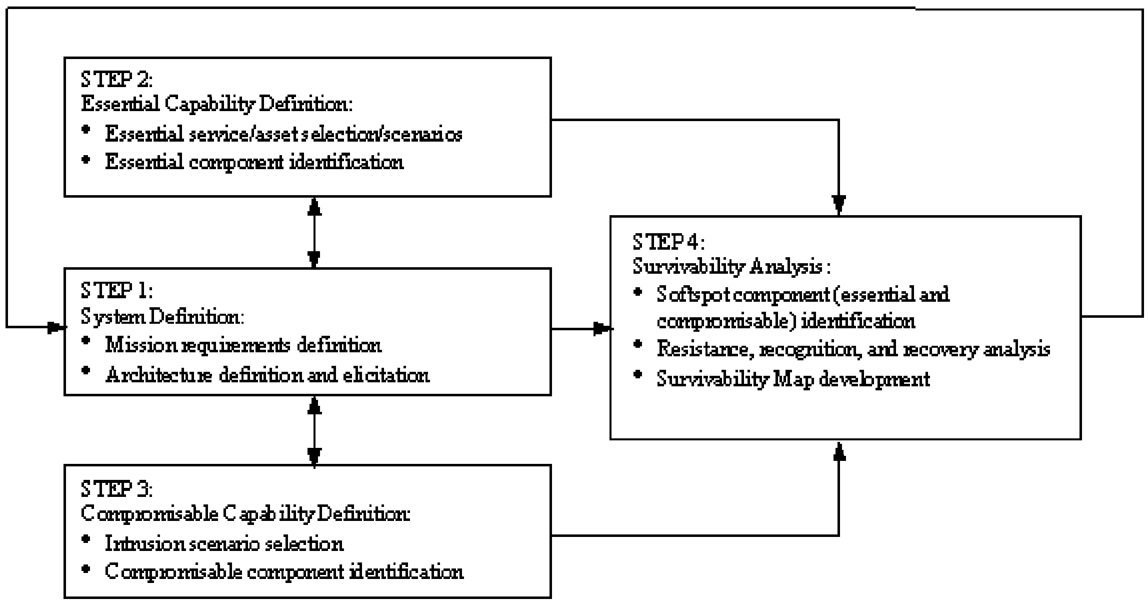
STEP 2:
Essential Capability Definition:
• Essential service/asset selection/scenarios
• Essential component identification
STEP 1:
System Definition:
• Mission requirements definition
• Architecture definition and elicitation
STEP 3:
Compromisable Capability Definition:
• Intrusion scenario selection
• Compromisable component identification
STEP 4:
Survivability Analysis:
• Softspot component (essential and compromisable) identification
• Resistance, recognition, and recovery analysis
• Survivability Map development

Figure 1. The Survivable Network Analysis Method

The SNA method takes COTS components' strengths and weaknesses into account, as well as any known security and reliability flaws. In Step 4, the team identifies the architecture's *softspot components* as components that are both essential and compromisable, based on the results of Steps 2 and 3. The team then analyzes softspot components and their supporting architectures for the key survivability properties of resistance, recognition, and recovery. The analysis of the "three Rs" is summarized in a *survivability map*. The map is a matrix that enumerates, for every intrusion scenario and its corresponding softspot effects, the current and recommended architecture strategies for resistance, recognition, and recovery. The survivability map provides feedback on the original architecture and system requirements and often results in an iterative process of cost-benefit analysis and survivability improvement. Although we developed the SNA method for use with large-scale distributed-network systems, it is equally applicable to other architectures, including host-based and real-time systems. SNA's scenario-based approach is a generalization of operation sequence[9] and usage scenario methods.[10,11]

3. Sentinel: the case study subsystem

Mental health treatment management is often a manual process with handwritten forms and informal communication. As a result, substantial time and effort are consumed to coordinate various treatment providers, including physicians, social service agencies, and healthcare facilities. CarnegieWorks Inc. is developing a large-scale, comprehensive management system to automate, systematize, and integrate multiple aspects of regional mental healthcare. The CWI system, named Vigilant, will ultimately contain some 22 subsystems that will operate on a distributed client/server network and maintain a large, complex database of patient and provider records.

A vital part of the Vigilant system is its development and management of treatment plans. A provider develops a treatment plan for a patient. The provider identifies each patient's problems together with a set of goals and actions, including medication and therapy, to achieve those goals. An interdisciplinary and interorganizational action team composed of providers carries out each treatment plan.

Treatment plan development and management and action team definition and coordination are key functions of Sentinel (a subsystem of Vigilant), which resides on a node in the network architecture and communicates with other nodes through network protocols. Sentinel interacts with other subsystems, with individual providers, and affiliations of providers that deliver healthcare services. Sentinel maintains the action teams and treatment plans as part of the Vigilant patient database, and it applies regulatory and business rules for treatment plan development and validation. Because of the critical nature of mental health treatment, the need to conform to regulatory requirements, and the severe consequences of system failure, CWI personnel identified survivability of key Sentinel capabilities as an essential requirement.

4. Applying the SNA method to sentinel

We applied the SNA method to the Sentinel subsystem architecture through a structured series of meetings between our team and Sentinel project personnel (the customer and development teams). We break down specific results from this case study below in terms of the four SNA steps and their corresponding artifacts.

Step 1: System definition

The following normal usage scenarios, or NUS, elicited from Sentinel requirements documentation characterize the subsystem's principal mission objectives. Each scenario includes a statement of the primary Sentinel responsibility with respect to the scenario:

- NUS1—Enter a new treatment plan. A provider assigned to a patient admitted into an affiliation performs an initial assessment and defines a treatment plan that specifies problems, goals, and actions. Sentinel must apply business rules to the treatment plan definition and validation.

- NUS2—Update a treatment plan. A provider reviews a treatment plan, possibly adding or changing problems, goals, or actions, and possibly updating the status of these items. Sentinel must apply business rules to the treatment plan update and validation.

- NUS3—View a treatment plan. A provider treating a patient views a treatment plan to learn the status of problems, goals, and actions. Sentinel must ensure that the displayed plan is current and valid.

- NUS4—Create or modify an action team. A provider defines or changes treatment team membership in a patient's affiliation. Sentinel must ensure that the treatment team definition is current and correct.

- NUS5—Report the current treatment plans in an affiliation. An administrator views the current state of his or her affiliation's treatment of a patient or set of patients. Sentinel must ensure that the treatment plan summaries are current and correct.

- NUS6—Change patient medication. A provider changes the medication protocol in a treatment plan for a patient, possibly in response to unforeseen complications or side effects. Sentinel must ensure that the treatment plan is current and valid.

Figure 2 shows the original Sentinel architecture obtained from design documentation. The evaluation team used execution traces of the normal usage scenarios identified in Step 1 to illuminate and understand architectural properties. The traces revealed component sequencing within the architecture, as well as the referencing and updating of database artifacts.

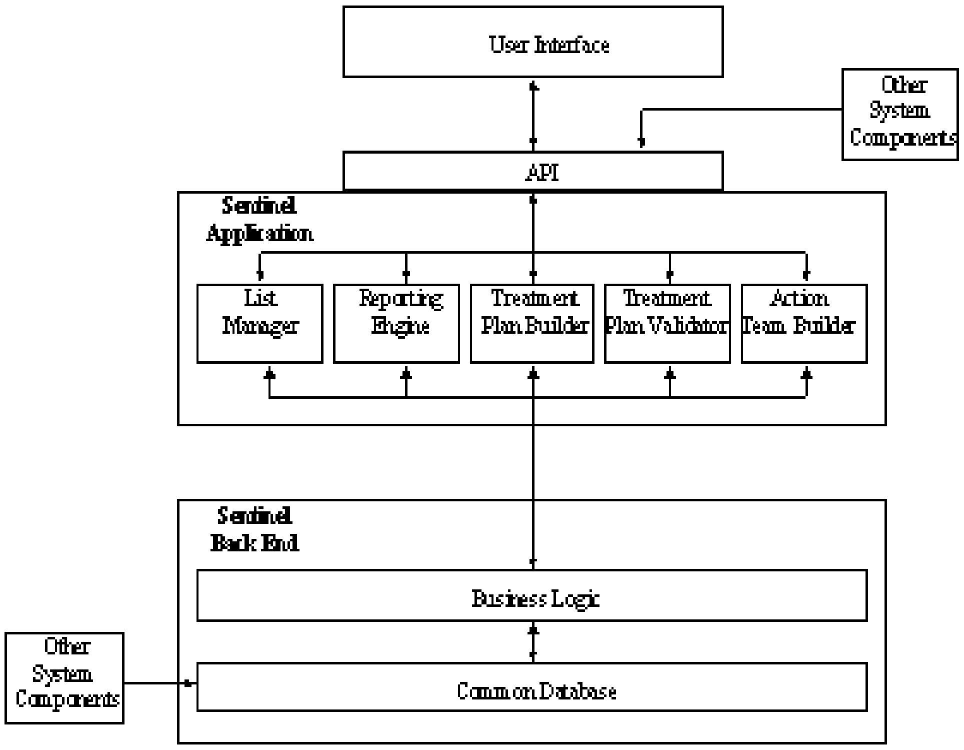

Figure 2. Original Sentinel Architecture

As shown in Figure 2, the user interface resides outside of Sentinel to allow a single interface to serve multiple subsystems and components. The application program interface, or API, provides synchronous remote procedure call and asynchronous messaging facilities for the user interface and other system components to use. The list manager maintains lists, including patients, affiliations, providers, and action teams, and the relationships among them. The reporting engine provides read-only viewing and reporting of Sentinel artifacts, including current treatment plans and their histories. The treatment plan builder creates treatment plans for patients, including problems, goals, and actions; the treatment plan validator checks the completeness and consistency of treatment plan development and modification. The action team builder provides capability to define and modify action team membership. Business logic contains enterprise-defined business rules, including validation checks for treatment plan development and logging triggers that manage change control of sensitive data. In the common database, Sentinel shares access to data with other subsystems and components.

Step 2: Essential capability definition

We based the essential service and asset analysis on the normal usage scenarios identified in Step 1. The analysis pointed to a single essential service, namely NUS3: the capability to view treatment plans. This service, more than any other, was deemed most essential to delivery of mental health treatment because providers depend on real-time, on-demand access to treatment plans in clinical situations, particularly for emergency situations involving medication or therapy problems.

The other services could be postponed for hours or even days in the event of system intrusion and compromise.

The analysis also identified a single essential asset: the treatment plans. Preservation of treatment plan integrity and confidentiality was deemed essential to meeting Sentinel mission objectives. The other Sentinel artifacts, such as action teams, affiliations, and providers, could all be reconstructed or updated hours or days after intrusion with no irreversible consequences.

The execution trace of the NUS3 scenario revealed that the reporting engine and the database components, as well as their supporting components and artifacts, are essential to maintaining the capability to perform the scenario. As essential assets, the integrity and confidentiality of treatment plans depends on database components for security and validation.

Step 3: Compromisable capability definition

Based on the system environment and assessment of intruder objectives and capabilities, we selected the following set of five intrusion usage scenarios, or IUS, as representative of the types of possible attacks on Sentinel. We selected the intrusions based on customer priorities, as well as on our judgment and experience, for their ability to illuminate the risks and vulnerabilities that the essential services could experience. We judged the scenarios sufficient to cover the exposures; additional scenarios considered did not add significantly to the findings. The description for each scenario consists of an IUS number, the type of attack (shown in parentheses), and a brief explanation:

- ♦ IUS1 (data integrity and spoofing attack)—An intruder swaps the patient identifications between two validated treatment plans.
- ♦ IUS2 (data integrity and insider attack)—An insider uses other legitimate database clients to modify or view treatment plans controlled by Sentinel.
- ♦ IUS3 (spoofing attack)—An unauthorized user employs Sentinel to modify or view treatment plans by spoofing a legitimate user.
- ♦ IUS4 (data integrity and recovery attack)—An intruder corrupts major portions of the database, leading to a loss of trust in validated treatment plans.
- ♦ IUS5 (insider and availability attack)—An intruder destroys or limits access to Sentinel's software so that it cannot retrieve treatment plans.

The execution traces of the five IUSs revealed various component vulnerabilities. IUS1 compromised the treatment plan component; no validity checks on the treatment plans were made after their initial entry. IUS2 also compromised the treatment plan component; the treatment plan changes might have been made by an improper agent. In IUS3, the treatment plan component is likewise compromised; the system architecture included terminals in open areas that could be accessed by unauthorized users. IUS4 compromised the treatment component as well; the system architecture emphasized user capabilities, and system backup and recovery had not received equivalent attention. IUS5 affected all software components, including the reporting engine; although there were implicit user requirements for availability, they had not been considered in the original system architecture.

Step 4: Survivability analysis

As we noted earlier, softspot components are both essential and compromisable. Our analysis showed that the reporting engine component and the database treatment plan component can both be compromised in a variety of ways.

Analysis of the three Rs resulted in the survivability map depicted by Table 1. Development of the table began with the matching of each intrusion scenario trace, created in Step 3 above, to softspot components. We first checked each trace to determine if any current resistance components (described in the resistance column of the survivability map for each scenario) in the architecture could increase the difficulty an intruder would confront in reaching the softspots referenced in the trace. Because no detailed implementation information was available to identify specific vulnerabilities in these resistance components, we assumed that implementation vulnerablities would be identified and corrected later.

Table 1. The Sentinel subsystem survivability map.

Intrusion scenario	Resistance strategy	Recognition strategy	Recovery strategy
IUS1: An intruder swaps the patient Ids of two validated TPs.	**Current:** Two passwords required for TP access.	**Current:** Logging of changes made to DB. Provider might recognize an incorrect TP.	**Current:** Built-in recovery in commercial DB. Backup and recovery scheme defined.
	Recommended: Implement strong authentication supported in a security API layer.[1]	**Recommended:** Add cryptographic checksum when TP is validated.[3] Verify cryptographic checksum when TP is retrieved.[4]	**Recommended:** Implement a recovery mode in the user interface to support searching for and recovering incorrect TPs.[1]
IUS2: An intruder uses other legitimate DB clients to modify or view TPs controlled by Sentinel.	**Current:** Security model for DB field access.	**Current:** None.	**Current:** Scrap data and start over, or use a backup and verify entries.
	Recommended: Verify adequacy of existing security model with respect to the integration of future system components.	**Recommended:** Perform a validation on access of a TP for verification.[2] Add cryptographic checksum when TP is validated.[3] Verify this checksum when TP is retrieved.[4]	**Recommended:** Scan DB for invalid crypto-checksums and/or invalid TPs and recover to last known correct TP.[4]
IUS3: An unauthorized user employs Sentinel to modify or view TPs by spoofing a legitimate user.	**Current:** None. No timeout is specified—anyone can use a logged-in terminal. Intruder only has access to logged-in user's TPs.	**Current:** None, except for the unusual number of denied accesses to TPs as an intruder attempts to locate particular TPs.	**Current:** Can get list of modified TPs through the spoofed user's transaction history. Manually recover each modified record.
	Recommended: Add a short log-out timeout for any terminals in uncontrolled areas.[1]	**Recommended:** Add logging, access control, and illegal access thresholds to the security API.[1]	**Recommended:** Develop a recovery procedure and support it in the application user interface.[1]
IUS4: An intruder corrupts major portions of the DB leading to a loss of trust in validated TPs.	Current: Security model in the DB protects data against corruption.	**Current:** None, except when provider happens to recognize a corrupted TP.	**Current:** Locate an uncorrupted backup or reconstruct TPs from scratch.
	Recommended: Implement live replicated DB systems that cross-check for validity.[5]	**Recommended:** Add and check cryptographic checksums on records in the DB.[3,4]	**Recommended:** Reduce the backup cycle to quickly rebuild once a corrupted DB is detected.[5]

IUS5: An intruder destroys or limits access to Sentinel's software so that it cannot retrieve TPs.	**Current:** No procedures defined.	**Current:** System does not work.	**Current:** Reload the system from backups.
	Recommended: Focus on quick recovery.	**Recommended:** None. Easy to detect.	**Recommended:** Maintain software archives and define procedures for fast recovery. Create a small subsystem that can retrieve TPs while Sentinel software is down. 6

ID stands for identification, TP for treatment plan, and DB for database. References are to architecture components.

For the recognition column, we followed a similar process. To assess the effectiveness of current recognition components, we made a number of assumptions (noted in the survivability map). For example, in IUS3 in Table 1, we assume that a provider will become suspicious when there are a large number of denied accesses to treatment plans. If this assumption is not valid, then there are no current recognition strategies associated with this scenario.

For the recovery column, we made assumptions regarding standard practice in distributed database management facilities—standard backup and recovery of the database and version control of the Sentinel software. Table entries for current recovery strategies included these assumptions. If they are not satisfied in the final system, the recovery strategies will be less effective than those described in the survivability map.

Once we identified the current resistance, recognition, and recovery strategies, we analyzed gaps and weaknesses to locate common points in the architecture where a particular survivability improvement could address multiple scenarios or strategies. These high-leverage recommendations are listed in a consistent form and identified as a common recommendation. We also addressed other gaps for which there is no existing strategy in the resistance, recognition, or recovery columns.

For the resistance column, we made recommendations even where an existing resistance mechanism existed, as this mechanism can be expected to degrade over time. Ultimately, the system architect must determine the costs and benefits of implementing these recommendations. The survivability map can help an architect determine the impact of accepting risks associated with weaknesses in the resistance, recognition, or recovery columns, as these have a correlation to the intrusion scenarios that can affect the system's essential services or assets. Table 1 identifies a number of gaps and assumptions in the current strategies. Of particular interest to an architect are those recommendations that deal with multiple intrusion scenarios. For example, adding a cryptographic checksum to the validation of a treatment plan can minimize the risks in several scenarios. A cryptographic checksum could be a hash function or message digest applied to the fields of the treatment plan to provide an integrity check on the contents.

Figure 3 illustrates the modified architecture resulting from the survivability map analysis. Many of the recommendations call for alterations to the same architectural component. To further illustrate the overlaps, the recommendations are annotated with a reference number (1-6) associated with the modified architecture. This makes it easy to determine which recommendations would alleviate risks in multiple intrusion scenarios. This view of the recommendations can help the architect allocate limited resources to high-impact modifications of the architecture. In modifying the architecture to address the recommendations in the survivability map, several natural locations emerged where changes could be implemented with minimal impact to the overall system. This beneficial localization was primarily due to the functional decomposition used in the original architecture. It is also likely that the scenario evaluations led to recommendations that were natural to the architecture, because their impact on individual modules was evident.

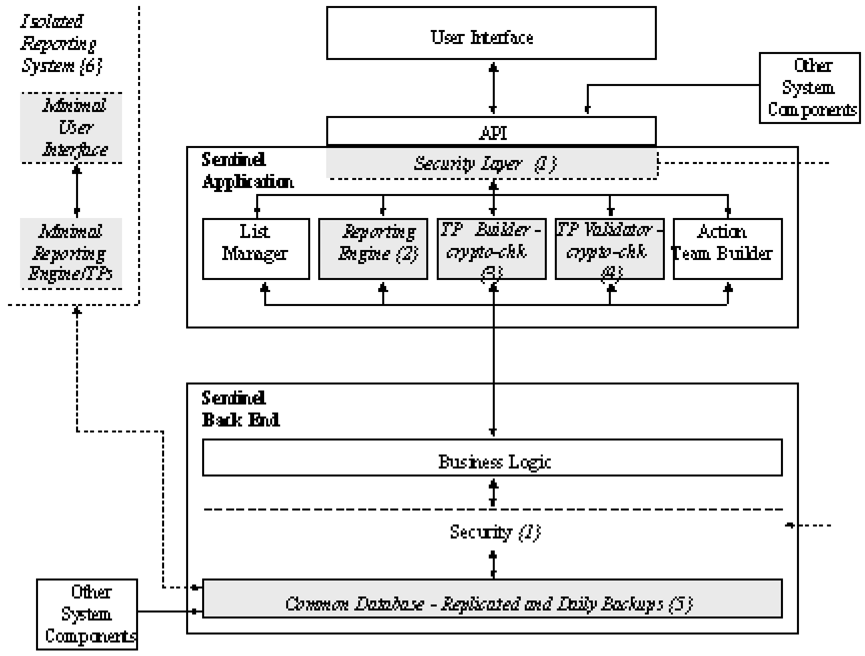

Figure 3. The Sentinel architecture with survivability modifications (additions and changes shown with dashed lines and shading).

In addition to using these findings for architectural analysis, Sentinel personnel could also use the findings to make modifications to system requirements. Step 1 revealed that few specific survivability requirements were specified other than requiring treatment plan validation, utilization of security features built into the standard login process and the database, and a development strategy that permits easy modification so that security features could be added. Changes are thus needed at the highest level to two areas of the requirements; under survivability conditions, there is a requirement for providers to view treatment plans within a reasonable time, and a requirement to protect the integrity of the treatment plans in the database.

The SNA method is under continuing development and additional case studies on large-scale network systems are underway. The method’s success depends on the effectiveness of its recommendations; that is, how well the system meets the survivability requirements selected by the customer in terms of essential services and assets, given the extent to which the SNA recommendations are implemented. Of equal importance is whether the customer can readily incorporate the SNA recommendations into the existing software development process, and thus be able to adopt the suggested changes. Because the survivability recommendations for Sentinel concentrated on refining an existing architecture, rather than on requiring a redesign, they satisfied this criterion.

Although most of the recommendations focused on revisions to the application architecture, several called for changes in design and implementation, or in operations and procedures. The study also raised questions about the system’s extensibility: Could the proposed architecture support, from a survivability perspective, the functionality desired in later versions? The SNA process raises questions about all of the design choices that are embodied in a system architecture. Future studies will explore how to leverage architectural choices to better support survivability, in the same way that this study leveraged the survivability capabilities of the relational database infrastructure.

The SNA method represents the first word, not the last, on the complex problem of assessing system survivability. Much work remains to be done, such as developing theory and practice for rigorous system behavior and architecture definitions, creation of canonical intrusion scenarios that embody current knowledge of attack strategies, integrating risk analysis and management techniques into the method, and quantifying survivability metrics and measures of success.

Acknowledgments

We thank Howard Lipson and David Fisher of the SEI CERT/CC for their contributions to the definition of the SNA method. We thank Cindi Carbine of CarnegieWorks Inc. and Carnegie Mellon University's Master of Software Engineering student developers Melissa Cheok, Ryan Soulier, Grama Srinivasan, Shinji Yamato, and Isabel Yebra for their support and cooperation in conducting this study. CERT and the CERT Coordination Center are registered in the US Patent and Trademark Office. The Software Engineering Institute is a federally funded research and development center sponsored by the US Department of Defense.

References

1. *Critical Foundations: Protecting America's Infrastructures: The Report of the President's Commission on Critical Infrastructure Protection*, Government Printing Office, Washington, D.C., 1997.

2. R.J. Ellison et al., *Survivable Network Systems: An Emerging Discipline*, Tech. Report CMU/SEI-97-TR-013, Pittsburgh, Penn., Software Engineering Institute, Carnegie Mellon University, Nov. 1997 (revised May 1999).

3. R.C. Linger, N.R. Mead, and H.F. Lipson, "Requirements Definition for Survivable Network Systems," *Proc. Int'l Conf. on Requirements Engineering*, IEEE Computer Soc. Press, Los Alamitos, Calif., 1998, pp 14-23.

4. R.J. Ellison et al., *A Case Study in Survivable Network System Analysis*, Tech. Report CMU/SEI-98-TR-014, Pittsburgh, Penn., Software Engineering Institute, Carnegie Mellon University, Sept. 1998.

5. *Proc. Information Survivability Workshop*, IEEE Computer Soc. Press, Los Alamitos, Calif., 1998; www.cert.org/research/isw98.html .

6. R.C. Summers, *Secure Computing: Threats and Safeguards*, McGraw-Hill, New York, 1997.

7. V. Mendiratta, "Assessing the Reliability Impacts of Software Fault-Tolerance Mechanisms," *Proc. Int'l Symp. on Software Reliability Engineering*, IEEE Computer Soc. Press, Los Alamitos, Calif., 1992, pp 99-103.

8. J. Musa, A. Iannino, and K. Okumoto, *Software Reliability: Measurement, Prediction, and Application*, McGraw-Hill, New York, 1987.

9. R.A. Kemmerer and P.A. Porras, "Covert Flow Trees: A Visual Approach to Analyzing Covert Storage Channels," *IEEE Trans. on Software Eng.*, Vol.17, No.11, Nov. 1991, pp. 1166– 1185.

10. J.M. Carrol, "Five Reasons for Scenario-Based Design," *Proc. Hawaii Int'l Conf. on Systems Sciences*, IEEE Computer Soc. Press, Los Alamitos, Calif., 1999, p 123.**.**

11. S.J. Prowell et al., *Cleanroom Software Engineering: Technology and Process*, Addison/Wesley Longman, Reading, Mass., 1999.

Robert J. Ellison is a senior member of the technical staff in the Networked Systems Survivability Program at the SEI. His research interests include system survivability, software development environments, and CASE tools. He has a PhD in mathematics from Purdue University. He is a member of the ACM and the IEEE Computer Society.

Richard C. Linger is a visiting scientist in the Networked Systems Survivability Program at the SEI and an adjunct faculty member at the Heinz School of Public Policy and Management, Carnegie Mellon University. His research interests include survivable systems, semantics of network architectures, large-scale system development, cleanroom software engineering, software project management, and process improvement. He holds a BSEE from Duke University and is a member of the ACM and IEEE.

Thomas Longstaff is currently managing research and development in network security for the Networked Systems Survivability Program at the SEI. His research interests include information survivability and critical national infrastructure protection. He has a PhD in computer science from the University of California, Davis.

Nancy R. Mead is a senior member of the technical staff in the Networked Systems Survivability Program at the SEI, and a faculty member in the Master of Software Engineering, Carnegie Mellon University. She is currently involved in the study of survivable systems architectures, and the development of professional infrastructure for software engineers. She has also served as Director of Education for the SEI. Her research interests are in software requirements engineering, software architectures, software metrics, and real-time systems. She has a PhD in mathematics from Polytechnic Institute of New York. She is a senior member of the IEEE and IEEE Computer

Society, and a member of ACM.