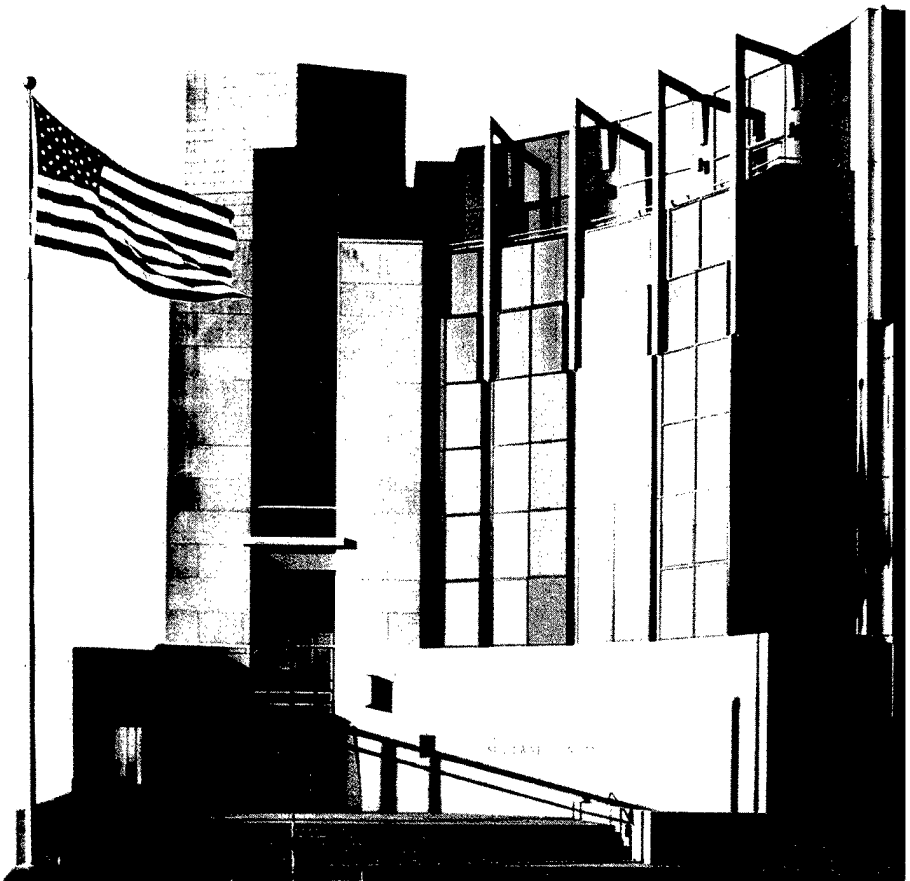
Carnegie Mellon
Software Engineering Institute

Trustworthy Refinement Through Intrusion-Aware Design

Robert J. Ellison
Andrew P. Moore

October 2002

TECHNICAL REPORT
CMU/SEI-2002-TR-036
ESC-TR-2002-036



1122 101

Carnegie Mellon
Software Engineering Institute

Pittsburgh, PA 15213-3890

Trustworthy Refinement Through Intrusion-Aware Design

CMU/SEI-2002-TR-036
ESC-TR-2002-036

Robert J. Ellison
Andrew P. Moore

20021122 101

October 2002

Networked Systems Survivability Program

Unlimited distribution subject to the copyright.

This report was prepared for the

SEI Joint Program Office
HQ ESC/DIB
5 Eglin Street
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER

Norton L. Compton, Lt Col, USAF
SEI Joint Program Office

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

Copyright 2002 by Carnegie Mellon University.

NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.

Internal use. Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

External use. Requests for permission to reproduce this document or prepare derivative works of this document for external and commercial use should be addressed to the SEI Licensing Agent.

This work was created in the performance of Federal Government Contract Number F19628-00-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.
The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (http://www.sei.cmu.edu/publications/pubweb.html).

Table of Contents

Executive Summary ..... **v**

Abstract ..... **ix**

1 Introduction **1**
- 1.1 Background 2
- 1.2 Related Work 4
- 1.3 Structure of this Report 5

2 IAD Model Overview **7**
- 2.1 Model Structure 7
- 2.2 Sources and Effects of Change 10

3 Architectural Strategy Sector **13**
- 3.1 Approach 13
- 3.2 Supporting Techniques 16

4 Architectural Instantiation Sector **21**
- 4.1 Approach 21
- 4.2 Supporting Techniques 23

5 Environmental Analysis Sector **27**
- 5.1 Approach 27
- 5.2 Supporting Techniques 29

6 IAD Model Context **37**
- 6.1 Model Instantiation 37
- 6.2 Model Incorporation 39

7 Conclusion **41**

Bibliography **43**

List of Figures

Figure 1: TRIAD Overview 8

Figure 2: Data Relationships 10

Figure 3: Architectural Strategy Formulation 13

Figure 4: Top-Down Requirements Tracing from Mission Objectives 15

Figure 5: Example Mapping Tables 15

Figure 6: Architectural Instantiation 21

Figure 7: Bottom-Up Requirements Tracing from Survivability Primitives 22

Figure 8: Threat Environment Analysis 27

Figure 9: A Feedback Loop for Controlling Vulnerability 31

Figure 10: The Effects of Vulnerability Publication on Internet Vulnerability 32

Figure 11: eBusiness Threat Analysis 33

Figure 12: Attack Tree Representation 34

Figure 13: High-Level eBusiness Attack Tree 35

Figure 14: Example Model Instantiation 38

Figure 15: TRIAD in SDM Process (1) as Mini-Spiral or (2) Through Integration 40

Executive Summary

High confidence in a system's survivability requires an accurate understanding of the system's threat environment and the impact of that environment on system operations. Unfortunately, existing development methods for secure and survivable information systems often have a patchwork approach in which the focus is on deciding which popular security components to integrate rather than making a rational assessment of how to address the attacks that are likely to compromise the overall mission. This preliminary report proposes an intrusion-aware design (IAD) model called trustworthy refinement through intrusion-aware design (TRIAD). TRIAD enables information system engineers to use known and hypothesized attack patterns to iteratively improve and continually maintain system survivability, even as the system and threat environment evolve over time. The model helps engineers understand complex interactions among the information system, its mission, and its threat environment at all levels of system architectural refinement. Information systems include any combination of information technology and people's activities for using that technology to support operations, management, and decision-making. We focus primarily on large-scale, highly distributed, and inter-networked information systems, such as Internet-based applications.

TRIAD focuses on patterns of attack and strategies for surviving attack at an architectural level to avoid getting overwhelmed by the details of individual component vulnerabilities or piecemeal security solutions. We focus on malicious attacks, rather than non-malicious failures or accidents, because of the increasing sophistication, frequency, and severity of such attacks and the inadequacy of existing approaches for dealing with them. Where available, TRIAD promotes using available security and survivability building blocks to help prevent, monitor, detect, and respond dynamically to likely intrusions. We consider both technological and procedural building blocks, since individual technological solutions to specific survivability problems may be unavailable, too immature, or too costly for the organization building the system. TRIAD facilitates planning for the inevitable change to the threat and operational environment and helps trace the effects of change back to the survivability requirements and architecture.

Survivable systems development is a domain in which the optimal refinement strategy is very unclear during the early stages of system design, particularly where unbounded, network-based systems are involved. Much experimentation and analysis is needed before a solution can be found with an acceptably small degree of residual risk of mission failure. We thus adopt the structure and philosophy of Boehm's spiral model as the basis for TRIAD. Each iteration of TRIAD gradually refines the system architecture based on the whiteboard prototyping, risk analysis, and risk mitigation of any previous iteration. This iteration permits ad-

justments and corrections to be made to the requirements, architecture, or resulting risks based on new experience and evidence.

The spiral structure of TRIAD proceeds through three sectors:

(I) Architectural Strategy – This sector derives justifiable system survivability requirements and high-level conceptual architecture from the need to ensure mission success despite penetrations and compromises.

(II) Architectural Instantiation – This sector refines the technical architecture within the constraints set by the conceptual architecture by identifying and integrating the critical technical building blocks.

(III) Environmental Analysis – This sector represents the threat environment and analyzes its impact on system operation, including the system's ability to carry out its mission successfully.

Whereas Sector I activities proceed primarily top-down from the overall mission, Sector II activities proceed primarily bottom-up by identifying survivability primitives to instantiate the conceptual architecture. The combination of the conceptual architecture, produced in Sector I, and the technical architecture, produced in Sector II, constitute the system's survivability architecture. Sector III activities ensure that the threat environment is considered consistently through all iterations of architectural refinement. The refinement and analysis on which TRIAD is based uses generic, reusable information that should make the overall process affordable and efficient. This report illustrates a detailed instantiation of TRIAD that shows how one might initiate the process, followed by iteration through the sector activities, and completing when an acceptable degree of residual risk is determined. A similar illustrative model could be developed for enhancing an already existing architecture into one with improved survivability properties.

TRIAD does not deal specifically with many issues required of a comprehensive system development life cycle. Developers will need to resolve these issues to incorporate TRIAD into their system development and maintenance process. We believe that mission-related survivability requirements must be used to determine the overall shape of the architecture and must, therefore, be the focus of the initial iterations of the design process. Functions or properties required or desired that do not contribute to the mission must fit within the parameters defined by the survivability architecture and must not significantly lower the confidence that the system owners have in that architecture. This report outlines two approaches for incorporating TRIAD into a comprehensive system development life cycle, as a separate up-front mini-spiral or by more fully integrating design activities into the life-cycle process. A detailed approach of how to do this depends largely on the details of the system problem domain and the development environment, and is beyond the scope of this report.

Although the model described in this report presents only a broad-brush sketch of IAD activities, it provides a starting point for the further refinement, experimentation, and validation of

an approach to exploit knowledge of intruder behavior to improve system architecture design and operations. In the near term, we plan to continue to explore the viability of and refine TRIAD through its application to the focused analysis of very specific problem situations. Each example will involve the identification of a specific problem situation, a TRIAD analysis and mitigation of that situation, and a characterization of the improvement gained through the analysis and mitigation. By focusing on specific problems in a diverse set of narrow domains, we expect to get quick feedback on the efficacy, flexibility, and scalability of the model and insights into how to improve it.

Later work will involve a full-scale application of TRIAD to demonstrate its scalability to more complex problems. TRIAD targets systems where there should be tighter integration between the security and system architectures. TRIAD demonstrations could target

- a new system in the early phases of development
- an existing system where there are significant survivability reengineering issues
- an ongoing development where TRIAD could document and analyze the tradeoffs between the system and security architectures

Demonstrations will require assembling TRIAD activities and structures into a working system development life-cycle model appropriate to the application domain and development environment. In addition to refining TRIAD based on the full-scale application, we plan to develop a tutorial for its use, with relevant examples, and initiate transition of the technology to an interested organization. Documentation of these TRIAD case studies and a detailed set of guidelines for TRIAD's application in varied settings should help make a compelling case for the model's use and transition. Ultimately, with evidence of its efficacy, we expect that TRIAD will be integrated with more comprehensive life-cycle models for the development and maintenance of high-confidence systems.

Abstract

High confidence in a system's survivability requires an accurate understanding of the system's threat environment and the impact of that environment on system operations. Unfortunately, existing development methods for secure and survivable information systems often have a patchwork approach in which the focus is on deciding which popular security components to integrate rather than making a rational assessment of how to address the attacks that are likely to compromise the overall mission. This report proposes an intrusion-aware design model called trustworthy refinement through intrusion-aware design (TRIAD). TRIAD enables information system engineers to use known and hypothesized attack patterns to iteratively improve and continually maintain system survivability, even as the system and threat environment evolve over time.

1 Introduction

High confidence in a system's survivability requires an accurate understanding of the system's threat environment and the impact of that environment on system operations. Unfortunately, existing development methods for secure and survivable information systems often pursue a patchwork approach, deciding which popular security components to integrate, rather than a rational assessment of how to address the attacks that are likely to compromise the overall mission. Reductionist techniques that delve into low-level design and implementation while losing sight of the overall environment are doomed to failure. This preliminary report proposes an intrusion-aware design (IAD) model called trustworthy refinement through intrusion-aware design (TRIAD). TRIAD enables information system engineers to use known and hypothesized attack patterns to iteratively improve and continually maintain system survivability, even as the system and threat environment evolve over time.

TRIAD helps engineers understand complex interactions among the information system, its mission, and its threat environment at all levels of system architectural refinement. Information systems include any combination of information technology and people's activities using that technology to support operations, management, and decision-making. We focus primarily on large-scale, highly distributed, and inter-networked information systems, such as Internet-based applications.[1] Modern computer/network-based information systems typically cross organizational boundaries and have no central administration and no unified security policy. One may not control, or even know the number and nature of, nodes connected to unbounded Internet-based information systems. The distinction between insider and outsider may be dynamic in that a partner for one activity may be a competitor or adversary for another.

Society is becoming increasingly vulnerable to high-impact threats to complex, unbounded systems. TRIAD focuses on patterns of attack and strategies for surviving attack at an architectural level to avoid getting overwhelmed by the details of individual component vulnerabilities or piecemeal security solutions. We focus on malicious attacks, rather than non-malicious failures or accidents, because of the increasing sophistication, frequency, and severity of such attacks and the inadequacy of existing approaches for dealing with them. We focus on attacks that are likely with respect to the system of interest, rather than on all attacks that are theoretically possible, to ensure cost efficiency and relevancy of TRIAD application and the solutions that it promotes. Where available, the model promotes using available secu-

[1] Henceforth, unless otherwise indicated, our use of the term "system" specifically refers to such a large-scale, highly distributed, inter-networked information system, which includes both information technology and its operational context in combination.

rity and survivability building blocks to help prevent, monitor, detect, and respond dynamically to likely intrusions. We consider both technological and procedural building blocks, since individual technological solutions to specific survivability problems may be unavailable, too immature, or too costly for the organization building the system.

TRIAD facilitates planning for the inevitable change to the threat and operational environment and helps trace the effects of change back to the survivability requirements and architecture. In particular, we require traceability of the architectural solutions back to the intrusions that they are supposed to address. Traceability documentation is essential for system modifications caused by changes in the organization's risk profile, the appearance of new attack patterns, the availability of new technology supporting both functional and security requirements, and changes in the underlying work processes that impact the vulnerability and risk analysis.

This report uses a broad brush to sketch the primary elements, key relationships, and supporting techniques of TRIAD. The model does not represent the whole development process, but only that part having to do with architectural refinement and only from the perspective of survivability. In particular, we do not represent those parts of the process needed to refine more general system function or to consider other quality attributes in addition to survivability. Nevertheless, this model serves as a starting point for the further refinement, experimentation, and validation of an approach to exploit knowledge of intruder behavior to improve system architecture design and operations.

1.1 Background

Developers in many engineering disciplines rely on engineering failure data to improve their designs. Imagine the result if bridge builders had ignored the lessons learned from the torsional oscillations that caused the Tacoma Narrows Bridge to collapse. Or if ship builders had ignored the lessons learned about inadequate lifeboat space and manning that allowed the great loss of life when the Titanic sank. Engineering success requires that we also learn from the less famous disasters. The aerospace community, for example, has institutionalized a means for learning from air traffic accidents that has resulted in very low risk of death during air travel, despite its inherent hazards. Successful architects design structures to survive known faults in building materials, construction methods, and the environment.

Businesses and governments have historically been reluctant to disclose information about security failures, i.e., intrusions, on their systems for fear of losing public confidence or for fear that other attackers would exploit the same or similar vulnerabilities. However, increased public interest and media coverage of the Internet's security problems have resulted in in-

creased publication of attack data in books, Internet newsgroups, and CERT® security advisories, for example. Unfortunately, information system developers are using information on security failures, i.e., intrusions, in only a reactive way, to patch systems that they have already fielded, and even then in a very incomplete and inefficient manner [Arbaugh 00]. Information systems being built and managed today are prone to the same or similar vulnerabilities that have plagued them for years.

Exacerbating this problem is the increased dependence on extremely complex, internetworked systems. The complexity and openness of these systems to the general public increases the exposure and vulnerability to malicious activity. The result is that increasingly sophisticated attacks are exploiting exposed vulnerabilities at an alarming rate. As seen by recent Internet worms and viruses released (e.g., Melissa, Love Letter, Code Red, Nimda), attackers share tools and knowledge to amplify their capability [CERT 02]. Each attack method builds off the knowledge, experience, and code of the previous attack method, which ironically makes the attack (virus, worm, etc.) more survivable as a result. Increasingly sophisticated tools currently available permit relatively inexperienced individuals to execute very sophisticated attacks.

In addition, we have seen such attacks escalate with the intensity of political conflicts, such as the war in Kosovo, the tensions between the United States and China, and the conflict between India and Pakistan [Vatis 01]. While these attacks are often in the form of embarrassing Web site defacements, attackers are starting to surreptitiously target the perceptions of users, such as the attempts to modify the content of major news publications or company press releases. We are seeing increasingly stealthy attacks that fly "under the radar" of existing intrusion detection systems (e.g., a single probe executed once per day). Attacks that fly "over the radar" of intrusion detection systems, such as social engineering and physical attacks, need to be taken as seriously as technological attacks [Anderson 01]. In general, attacks can target a system's internal users and commercial off-the-shelf (COTS) components, as well as external trusted systems and user communities. Attacks by individuals more sophisticated than the average recreational hacker (e.g., industrial spies and international cyberterrorists) are becoming more likely and more difficult to counter.

A solely reactive approach to building and maintaining system security and survivability is doomed to failure, because of practical limits on shrinking the window of exposure of vulnerable systems [Arbaugh 00, Schneier 00a]. In fact, no amount of hardening can ensure that intrusions on unbounded systems do not occur. Survivability is the capability of a system to fulfill its mission by preserving essential services, even when systems are penetrated and compromised. Survivability requires strategies to monitor, detect, and adapt to intrusions to ensure mission success, as well as, to the extent possible, preventing intrusions in the first place. Survivability properties typically emerge from the architectural interaction of system

® CERT and CERT Coordination Center are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

components, and therefore must be considered very early in the development process. Considering survivability too late usually results in a system architecture that "hard-codes" mission vulnerabilities, making it too difficult, too costly, or downright impossible to build survivable implementations of that architecture.

TRIAD is built on the premise that a much more proactive use of available attack information is needed to build cost-effective systems that survive attack with high confidence. Building affordable survivability architectures demands understanding the threat environment of the system under construction so that effort is spent on the likely intrusions rather than all possible ones. Attacks may target people, processes, and physical structures, as well as the system technology. Likewise, survivability strategies must allow procedural, physical, and technological remedies to mission vulnerabilities to ensure the viability and affordability of the remedy.

Gaining confidence in a system's survivability requires showing that the system is adequately resilient to likely patterns of attack. The dynamic nature of the intrusion environment demands that TRIAD, and the analysis techniques on which it is based, help discover and hypothesize about new sources and patterns of attack, in addition to known attacks by known adversaries. Unfortunately, much of the available attack information is very detailed in terms of software versions, enterprise-specific configurations, and attacker-specific scripts. Such details have a relatively short life as the attackers create and revise their tools and methods. However, the general patterns of attack are much less variable over time. Attack patterns describe general attack strategies, such as the various forms of denial-of-service attacks, and can be structured so that they can be applied in a variety of contexts [Moore 01]. The CERT Coordination Center's experience analyzing the survivability of real systems across industry and government and collecting actual Internet-based attack data is leading to a more in-depth understanding of attack patterns, trends, and countermeasures [Ellison 99, CERT 02].

1.2 Related Work

A few efforts across industry and government are pursuing research to improve development methods for secure and survivable inter-networked systems of systems with a focus on the threat environment. Neumann provides important insights into and an overview of supporting mechanisms for the development of survivable system architectures [Neumann 00]. The Information Assurance Technical Framework (IATF) provides extensive guidelines for choosing security mechanisms to incorporate into potentially large-scale, mission-critical systems, based on a high-level characterization of the threat and value of information protected [IATF 00]. Another paper outlines a secure system engineering methodology based on a more extensive analysis of the threat environment, and is, therefore, somewhat more aligned with our approach [Salter 98]. Work in the area of intrusion tolerance, which is primarily funded in the United States by the Defense Advanced Research Projects Agency (DARPA) and in Europe through project MAFTIA, generally is intrusion focused and tackles large-scale distributed

systems survivability, but the research has usually ignored non-technical attacks and countermeasures [MAFTIA 02].

While the above work contributes to developing a model for IAD, none of the efforts take advantage of the full potential of exploiting available attack information for improving system survivability. No one that we know of is looking in-depth at the problem of using attack patterns and trends during system architecture refinement to maintain system security and survivability, in a way that copes well with the transient nature of the threat environment. The objective of our work is to address this problem, dealing directly with survivability maintenance as the system mission, architecture, and threat environment change. A great range of work in security risk analysis contributes to our effort, including the areas of adversary modeling, attack specification, vulnerability/threat analysis, security-related taxonomies and databases, impact analysis, and red teaming. Although the volume of work in this area precludes recounting all of the efforts that might contribute to IAD, we talk more about the general lessons learned from this research in later sections of this report.

1.3 Structure of this Report

This report is organized according to the structure of TRIAD. Section 2 provides an overview of the model structure containing three primary sectors of activities: Architectural Strategy, Architectural Instantiation, and Environmental Analysis. A more detailed description of each of these sectors, including a discussion of techniques that support the sector activities, is given in Sections 3 through 5. Section 6 discusses the context for applying TRIAD both in the small, through its instantiation to more specific IAD processes, and in the large, through its incorporation in the larger system development life cycle. Finally, Section 7 summarizes the report and the directions for future work.

2 IAD Model Overview

It is widely accepted that much of system architecting is creative in nature:

> *Architectural design processes are inherently eclectic and wide-ranging, going abruptly from the intensely creative and individualistic to the more prescribed and routine. While the processes may be eclectic, they can be organized. Of the various organizing concepts, one of the most useful is stepwise progression or "refinement" [Maier 00].*

TRIAD was formulated around the central notion of refinement in architecting, which motivated the 'R' in TRIAD. Maier goes on to note that the process of system architecting is best "characterized as episodic, with episodes of abstraction reduction alternating with episodes of reflection and purpose expansion" [Maier 00]. To reflect this episodic nature of system architecting, TRIAD adopts the structure and underlying philosophy of the spiral model of system development [Boehm 88, Marmor-Squires 89].

The spiral model is intended for system and software development and enhancement in complex domains with which the developers have little experience, domains where the best (or even a good) direction for system refinement is highly uncertain. Such domains require iterated refinement where each iteration gradually refines the system requirements, design, and implementation based on the experience of any previous iteration. This iteration permits adjustments and corrections to be made in the directions chosen for system refinement based on new evidence such as risk analysis, prototyping, and simulation. The original spiral model proceeds through four quadrants, each quadrant making progress toward improved understanding and refined documentation of the system requirements, design, and/or implementation [Boehm 88]. Like the spiral model, TRIAD is equally applicable to the development of new systems and the enhancement of existing systems.

2.1 Model Structure

Survivable systems development is certainly a domain in which the optimal refinement strategy is very unclear during the early stages of system design, particularly where unbounded, network-based systems are involved. Much experimentation and analysis is needed before a solution can be found with an acceptably small degree of residual risk of mission failure. The spiral structure of TRIAD, which is shown in Figure 1, proceeds through three sectors: (I) Architectural Strategy, (II) Architectural Instantiation, and (III) Environmental Analysis. Although the figure shows only the general structure of the model, a fully instantiated model

involves multiple iterations through these sectors. Consistent with the original spiral model, each iteration gradually refines the system architecture based on the whiteboard prototyping, risk analysis, and risk mitigation of any previous iteration. Progress starts in the middle of the figure in Sector 1 and proceeds along the spiral, the angular dimension of which indicates cumulative progress. An instantiation of the model involves multiple iterations through the sectors, which permits adjustments and corrections to be made to the requirements, architecture, or resulting risks based on new experience and evidence. Subsequent discussion describes the primary activities in each sector. Section 6 illustrates a fully instantiated model.

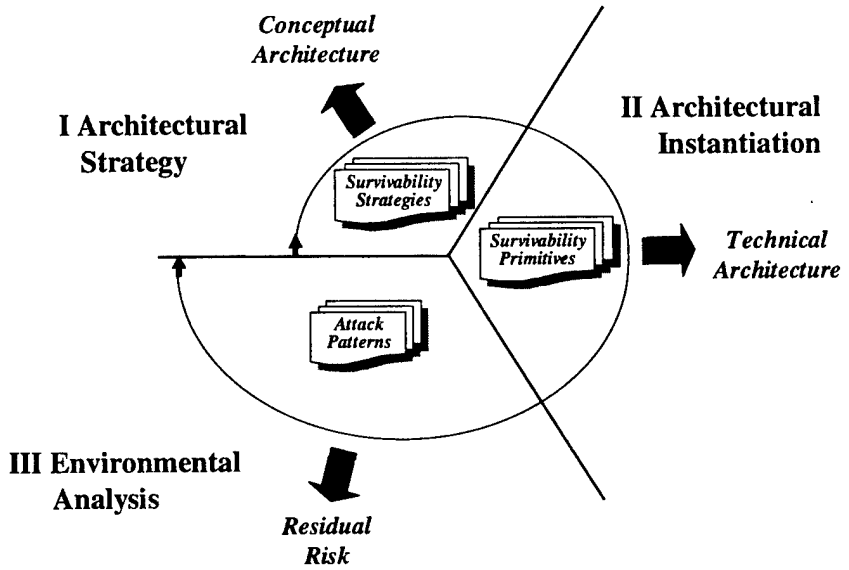


Figure 1: TRIAD Overview

The Architectural Strategy sector starts by elaborating the overarching mission of the system under design. Activities in the sector derive justifiable system survivability requirements and high-level conceptual architecture from the need to ensure mission success despite penetrations and compromises. As shown, requirements are derived from a collection of survivability strategies. A *survivability strategy* is a generic representation of an architectural approach to prevent, monitor, detect, or adapt to some pattern of attack in a specific context. Example strategies include redundancy, diversity, deception, separation, intrusion detection, recovery, and personnel management.

Activities in the Architectural Instantiation sector refine the technical architecture within the constraints set by the conceptual architecture by identifying and integrating the critical technical building blocks. Whereas Sector I activities proceeded primarily top-down from the overall mission, this sector's activities proceed primarily bottom-up (or middle-up) by identifying survivability primitives to instantiate the conceptual architecture. A *survivability primitive* is a generic representation of an architectural building block that is useful in a specific

context to implement some survivability strategy. Example primitives for recovery include restoration from backup and surveillance and apprehension of suspected intruders.

Activities in the Environmental Analysis sector represent the threat environment and analyze its impact on system operation, including its ability to carry out its mission successfully. The threat environment is derived from a collection of attack patterns. An *attack pattern* is a generic representation of a deliberate, malicious attack that commonly occurs in a specific architectural context. An attack pattern can target people (e.g., social engineering attacks that use a computer virus), the operation of the technology (e.g., distributed denial-of-service attacks), or the context in which people do work (e.g., dumpster-diving attacks).

The distinction between the sectors may not always seem clear, and there is bound to be some overlap, just as there was in the original spiral model. However, we have fairly concrete distinctions between each of the three sectors. The difference between Sector I and Sector II is similar to the difference between requirements and specification. Requirements may describe a general solution strategy, but leave open many design/implementation-level details; a specification makes many of the concrete decisions on how to proceed, often in terms of specific components and connectors. The conceptual architecture, produced in Sector I, describes the system function and structure at a level appropriate for the customer. The technical architecture, produced in Sector II, describes the function and structure of the system at a level of technical detail sufficient to actually build the system. Also Sector I typically refines the architecture top-down from mission objectives, whereas Sector II typically instantiates an architectural concept bottom-up (or middle-up) from available architectural primitives. Both top-down refinement and bottom-up refinement are an essential part of the system development process, and TRIAD supports them explicitly in each iteration. The combination of the conceptual architecture and the technical architecture make up the system's survivability architecture. Finally, Sector III focuses on the analysis of threat and impact given the architectural constraints specified in Sector II, whereas Sector I focuses on the description of requirements to mitigate the resulting risk. Sector III activities ensure that the threat environment is considered consistently through all iterations of architectural refinement.

The essential relationship of the data on which each sector is based is shown in Figure 2. Survivability strategies to ensure mission success suggest the use of specific survivability primitives. These primitives, in turn, have certain vulnerabilities within the context of a system architecture that promotes certain attack patterns. Attack patterns, in turn, suggest the adoption of other survivability strategies. Of course, this results in a never-ending cycle of analysis. The challenge of the intrusion-aware designer is to converge gracefully to a set of survivability strategies that address likely attack patterns in an affordable and effective manner, each member of which is implemented as a set of survivability primitives.

2.2 Sources and Effects of Change

The three sectors of the model represent the three major points at which change can occur: requirements, architecture, and threat environment. A change in requirements may occur through an expansion of the system mission. For example, a military command and control system must now also operate jointly with coalition forces. A change may also require contracting the mission or modifying its fundamental nature. For example, an eBusiness focusing on sales of high-end merchandise may transition to a strategy of high volume discounted sales of lower-end merchandise because of various market pressures.

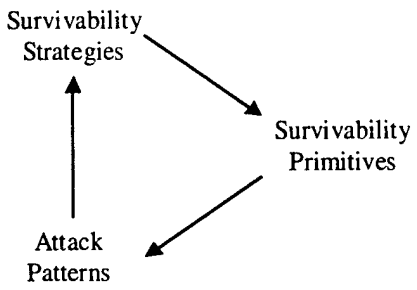


Figure 2: Data Relationships

Changes in the system architecture may be procedural or technological in nature. A business may decide to relax hiring practices in response to a highly competitive job market. A technological change may result when an eBusiness expands to physical sales of merchandise at multiple distributed sites. This change would require online inventory management and a level of trust in the workflows between the distributed sites.

Finally, a change in the threat environment may involve new types of attackers that need to be considered, or old types of attackers using new methods. New types of attackers may threaten an eBusiness when, for example, recent news reports publicize the eBusiness's dealings with unpopular organizations, making its system operations more susceptible to "hactivist" attack. Attackers who were previously considered a threat might take on new relevance when a new class of attack tools is increasingly used to penetrate corporate perimeters and take control of intranet operations.

TRIAD emphasizes the interrelationships between each of the three sectors. Changes in mission objectives may lead directly to changes in system structure to support the modified objectives. Changes in system structure can, in turn, affect the threat environment, for example, through increased exposure. Finally, a change in the threat environment may lead to modified requirements to preserve survivability, and ultimately structural changes to support these requirements. The documentation promoted by our approach promotes traceability between sectors to support continued maintenance of system survivability even after the system is fielded.

The next three sections of this report discuss the activities undertaken in each of the three sectors in more detail. Each sector involves the production of a set of artifacts developed using as input

- the generic survivability and attack information available—i.e., the survivability strategies, survivability primitives, and attack patterns as shown in Figure 1
- the output of any previous sectors' artifacts as progress is made along the spiral of Figure 1

The set of artifacts produced during the execution of any particular sector along the spiral may only be partially complete. Nevertheless, the set represents the output at that stage of progress and may be used in the activities of the next sector along the spiral. Progress continues until the set of artifacts produced for each sector is final and the level of residual risk determined by the Environmental Analysis sector's activities is acceptable to the stakeholders involved. The number of iterations of the spiral required for completion is application dependent, but we typically expect convergence on an acceptable solution in two to four iterations.

3 Architectural Strategy Sector

Any successful organization has an implicit or explicit mission that characterizes its primary purpose as a set of high-level objectives. The primary goal of the Architectural Strategy sector (Sector I) is to derive system requirements and a high-level conceptual architecture that promote the survivability of an organization's mission in the face of active attack. An organization's information technology, policies, procedures, personnel, and overall work context all exist to support the mission. Survivability must be maintained even as an organization's objectives, structures, behaviors, or threat environment evolve over time, possibly in unexpected ways. This requires proactive change management strategies that help determine and, when possible, contain the effects of change.

3.1 Approach

Figure 3 outlines the general structures for documenting the survivability requirements derived from the overall organizational mission. Inputs include any generic survivability strategies that help ameliorate the significant risks associated with the threat environment of the technical architecture characterized by previous sector activities. The ultimate output of the Architectural Strategy sector is the conceptual architecture describing the system features needed to ensure that the mission is achieved given the threat environment characterized.

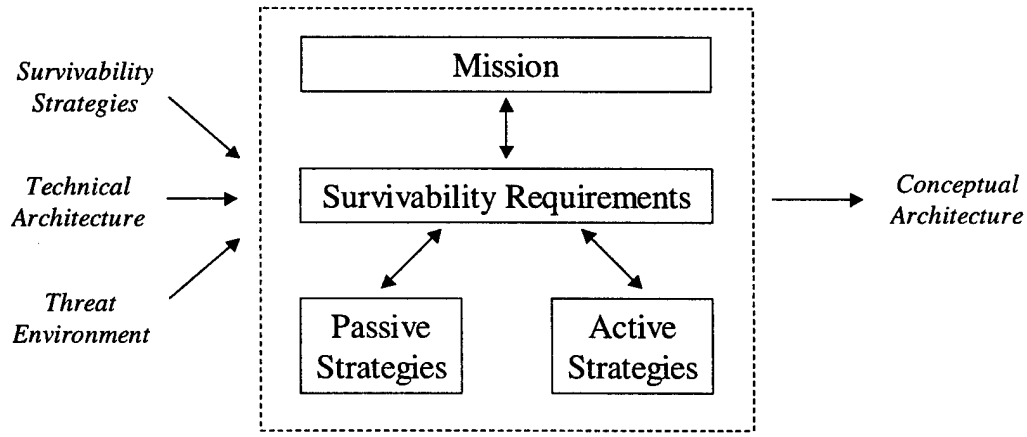


Figure 3: Architectural Strategy Formulation

Survivability strategies are broad approaches to ensuring the survivability of information systems and mitigating the threats to mission success. Experience with real attacks on systems

through the years emphasizes the need to consider the big picture, including both the technology and its operational environment, in order to develop strong and cost-effective solutions [Anderson 01, Schneier 00b]. Survivability has three primary components: the detection of, the prevention of, and the recovery/adaptation to an attack. Preventive techniques such as authentication, authorization (access control), and encryption increase the resistance to attacks and are typically passive. The system response to an attack in terms of recovery of services or continued (possibly degraded) operations may be to change the system configuration or security policy, for example, to strengthen authentication or tighten access control. The responses in this context are active in the sense that the system must detect the attack and then actively reconfigure system operations to ensure mission success in the presence of the attack.

The distinction between passive and active strategies often partitions the architectural design effort. It is not unusual for a large system to have a security architecture that concentrates on the passive defenses. The security architecture encompasses support for user authentication and authorization, encryption, and management of networks with firewalls or specialized security configurations for applications such as databases or Web servers. Passive security is often connected with a physical architectural view. Often vulnerabilities are associated with implementation or administrative errors, so that many aspects of passive security arise late in the design cycle. The design of the security architecture can demand expertise in terms of identifying covert channels, management of distributed security and authorization, and key management for encryption. For large or complex systems, a security architect can be designated to design this segment of the overall architecture.

Active responses to an attack are typically the responsibility of the system architect rather than the security architect. The overall objective for survivability is to continue to support the mission. Survivability analysis identifies essential services, maps those services onto the system architecture, and identifies the impact of attack scenarios on those services and hence on the mission. The system architecture has to support continued service in the presence of attacks through redundant or alternative services or to provide data recovery services so as to restore full service following an attack.

The division of responsibility between the system and security architectures can be a source of confusion. Attacks could target the directory services that maintain system-wide authentication and authorization services. So the security architecture also has to provide active responses in terms of recovery or continued operation. The architectural style chosen for the system, such as using a central data store, affect the active responses and the options for the security architecture. The selection and weights given to passive and active strategies, as well as the selection of appropriate architectural styles, is a critical aspect of activities in the Architectural Strategy sector.

Activities in the Architectural Strategy sector also maintain the traceability of the mission-related requirements. As shown in Figure 4, mission objectives are traced through the mission threats down to the survivability requirements, which include both the quality and the structural requirements of Figure 3. The threats addressed are those identified in the Environmental Analysis sector. The Architectural Instantiation sector, described in the next section, is responsible for tracing the survivability requirements through the survivability architecture down to the individual component responsibilities. Since IAD is an iterative refinement process, the mission threats and survivability architecture may be only partially defined on any given iteration of the spiral. Therefore, the requirements definition and traces are incrementally refined as well. The requirements tracing can be conveniently documented in tabular format, as exemplified in Figure 5.

Figure 4: Top-Down Requirements Tracing from Mission Objectives

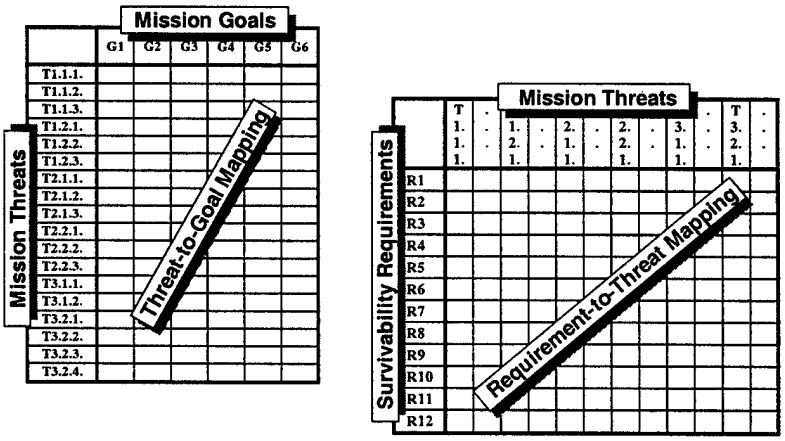


Figure 5: Example Mapping Tables

3.2 Supporting Techniques

3.2.1 Requirements Traceability

Requirements traceability has long been used to help ensure that a system's design and implementation conform to its requirements. However, perhaps even more important for our purposes, requirements traceability is essential to managing changes in a way that maintains an organization's survivability over time. In this broader context, requirements traceability can be defined as "a characteristic of a system in which the requirements are clearly linked to their sources (backward traceability) and to artifacts created during the system development life cycle based on these requirements (forward traceability)" [Ramesh 97]. In this definition, linkages are considered bidirectional. The Architectural Strategy sector is responsible for backward traceability, whereas the Architectural Instantiation sector is responsible for forward traceability, as we will explain in Section 4.

Traceability of requirements from an organization's mission helps determine a system's survivability dependencies. Studying the backward traceability of the requirements can help assess the impact of changes to an organization's mission or threat environment. Studying the forward traceability of the requirements can help assess the impact of changes to the system architecture. Conventional wisdom in the requirements traceability community dictates that traceability be maintained only for mission-critical requirements [Ramesh 98]. This wisdom is exactly aligned with the mission focus of survivability, since the organizational mission provides the starting point for TRIAD requirements tracing, as we will describe in the next section.

In summary, requirements traceability helps

- demonstrate that mission-critical requirements are satisfied
- identify the source and justification for each requirement
- understand the impact of errors and failures on the system's ability to achieve its mission
- understand the impact of change due to the evolution of the organization's objectives, structures, behaviors, or threat environment

Survivability requirements traceability requires a broader scope than typically adopted for general requirements traceability because of the breadth of the threats (e.g., from social engineering to technological compromise) and the countermeasures (e.g., from personnel, to procedural, to technological). Requirements must be consistently managed throughout the system lifetime to support continual risk management so that new threats and system operations do not lead to mission failure.

Usage Scenarios

Requirements may also usefully be described as a set of scenario descriptions in a way that helps map them onto the system architecture [Jacobson 99, Bass 01]. Potts describes scenarios as a sample execution of a system, the antithesis of specifications:

> *Whereas a specification describes behavior generally, a scenario exemplifies behavior by presenting specific, concrete episodes. Whereas it is possible to deduce scenarios from a specification, it is possible only to induce a specification from a collection of scenarios [Potts 95].*

System requirements for both functional and non-functional requirements can be defined as scenarios. A thread through the scenario literature involving *obstacle analysis* deals explicitly with a negative view [Potts 95, van Lamsweerde 00]. Obstacle analysis identifies and analyzes obstacles to realizing usage scenarios. Intrusions, or attacks, can be viewed as an obstacle to the security of a system. Negative scenarios are sometimes used during obstacle analysis, but primarily only to show the reality of the obstacle [Potts 95, Weidenhaupt 98]. We discuss techniques that support the documentation of intrusion scenarios in Section 5.

3.2.2 Survivability Strategies

As described earlier, generic survivability strategies provide a useful advanced starting point for deriving system requirements from mission goals given a particular threat environment. There are many sources across the security and survivability literature of potential strategies for ensuring mission success. The RAND Corporation, for example, has published a method for improving the survivability of systems based on categories of predefined survivability vulnerabilities and techniques [Anderson 99]. Although RAND's method has not been applied extensively, the study surveyed a wide range of existing systems and research efforts on security and survivability to derive vulnerability and technique categories. Examples include, but are not limited to, redundancy, diversity, deception, identification and authentication, intrusion detection, recovery/adaptation, physical separation, logical separation, cryptographic separation, temporal separation, and personnel management. These strategies may be implemented manually (through human procedures), automatically (through technology), or through a combination of the two. The survivability techniques identified provide a good start at identifying strategies for building survivable systems that should be useful for IAD. Other work on survivability architectures should also provide useful input to the IAD process [Knight 00, Neumann 00].

3.2.3 Architectural Styles

The importance of architecture for survivability suggests the utility of defining architectural styles that promote survivability [Shaw 96]. An architectural style is primarily characterized by a set of components, a set of connectors, and a topological layout that determines runtime relationships [Bass 98]. Progress in the area of defining architectural styles may be useful in

characterizing survivability strategies. A system's survivability, however, depends more on the workflow-based interaction of its components than the topology of its architecture. Maintaining the workflows that support the essential services is critical. In addition, the focus of survivability on the enterprise mission makes survivability requirements, by definition, highly application-dependent. Defining survivable architectural styles is difficult since those styles would have to reflect the broad variety of possible enterprise missions.

There are a variety of styles associated with a runtime view for an architecture. Client/server, shared data, and pipe and filter are examples of general styles. A shared data style could use a central database or a distributed data store. A shared data style is not likely to use diversity as a general survivability strategy. It often is too expensive and too complex to support database systems from multiple vendors. But data redundancy either logically or geographically is a widely used technique to support the survivability of a data service. Systems can support multiple styles. For example, a system that supports a common operational picture (COP) probably supports both a shared data and a pipe and filter style. The sensors for such a system are radars. A pipe and filter style is often used to model the flow of sensor data to a central processing point where the various data feeds can be correlated and tracks created. The various tracks are eventually consolidated into a shared data store that can be accessed from multiple locations. Survivability of the pipe and filter style depends in part on the survivability of the communication links and the sensors, but also on analysis algorithms that can provide useful output when some sensors or communications links are unavailable. The survivability of shared data depends on the redundancy of the data store and communication to the users of the shared information. The nature of the mission is a critical piece of the analysis. For shared financial data, it could be a critical survivability requirement to restore the contents and maintain data integrity after an attack. For a COP, the immediacy of information is critical, so the survivability requirement is expressed in terms of the time required to restore the display to represent the latest information available from the sensors.

System architectural styles also affect the passive strategies and hence the security architecture. A shared data style could require access control in terms of the data fields, while a pipe and filter style likely operates at a level of granularity of the dataflow that uses that pipe. A central data store for use by a geographically dispersed organization could also require central management of authorization and authentication information via a directory service.

Some active strategies should have a significant impact on the security architecture. For example, dynamic changes in security policies, such as changes in authorization in a distributed system, could require central management of user security information so that a change could be propagated quickly to all affected systems. Some of the active options could lead to significant administrative costs for the security architecture.

The security architect has many of the same kind of decisions as the system architect. A complex system will have multiple workflows that pass through a boundary controller such as a

firewall. It may be better to separate services and, for example, associate a firewall with each essential workflow implementing a distributed security architecture.

4 Architectural Instantiation Sector

The primary goal of the Architectural Instantiation sector (Sector II) is to develop a technical architecture that supports the survivability requirements and instantiates the conceptual architecture identified in Sector I. To do this, the passive and active survivability strategies used in Sector I are implemented in terms of available architectural primitives. These primitives may be available either commercially or through government-sponsored research and development programs. Sector II activities identify the responsibilities of individual components of the technical architecture that help achieve the survivability requirements. Refining requirements may involve tradeoffs in terms of costs or complexity of administration that may suggest refinements to the conceptual architecture. The conceptual and technical architectures taken together constitute the survivability architecture.

4.1 Approach

Figure 6 outlines the general structures for refining the technical architecture. Inputs include any survivability primitives that help instantiate the conceptual architecture, given the risks identified by previous sector activities. The ultimate output of the Architectural Strategy sector is the technical architecture describing the system technology and interconnections needed to implement the survivability requirements. Whereas Sector I activities proceed primarily top-down from the overall mission, this sector's activities proceed primarily bottom-up (or middle-up) by identifying survivability primitives to instantiate the conceptual architecture.

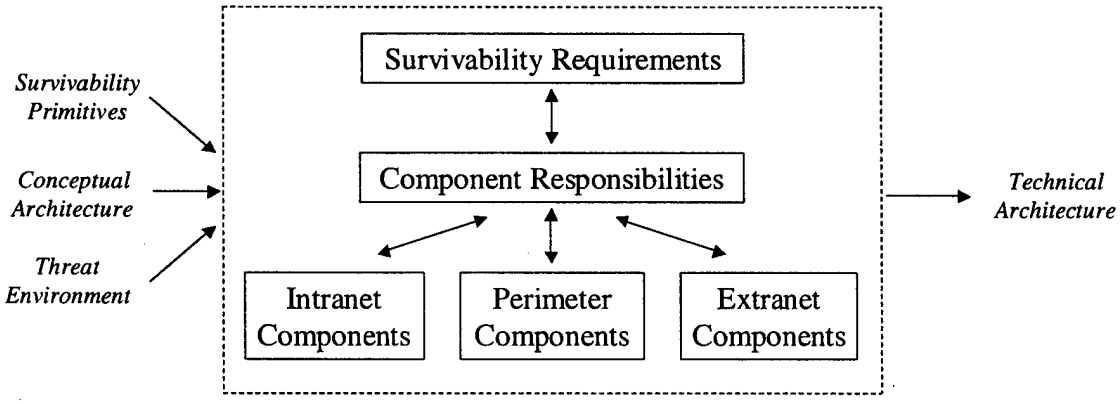


Figure 6: Architectural Instantiation

Critical areas of consideration include

- the intranet, which includes the organization's databases, applications, servers, workstations, internal networks, and procedures for their use
- the perimeter, which includes firewalls, gateways, and physical mechanisms that protect the organization's intranet assets from external access
- the extranet, which includes any networks outside the perimeter that must be relied on to achieve the organization's mission

Sector activities must maintain the traceability of the survivability requirements through the survivability architecture to the responsibilities of the survivability primitives, as shown in Figure 7. When combined with the Sector I requirements traces, the transitivity of the mapping relation ensures the traceability of the mission objectives to the component responsibilities. This traceability helps determine the impact of any changes to the system mission, threat environment, or architecture on the overall system survivability. Mapping tables can help to support this traceability, as described previously.

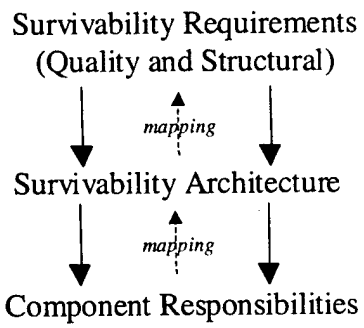


Figure 7: Bottom-Up Requirements Tracing from Survivability Primitives

Following are some additional activities that may need to be carried out in this sector. Experience applying TRIAD to real applications will help both the researcher and eventual user understand these activities in more depth.

- Map passive and active strategies into the architecture.
  - Authentication and authorization can be centrally managed by a portal or can be the responsibility of each application.
  - Threats that exploit content, such as viruses, are countered by application-level monitors.
  - Application-level transactions support recovery from failures in communications.
  - There is a central boundary controller or an individual boundary controller for critical hosts or workflows.
- Match primitives with the system architectural style: data centered, pipe and filter.
  - Use a data field access control model for data-centered style.
  - Use central data recovery mechanisms for data-centered design.

- Identify architecture mechanisms that support proposed architectural strategies.
  - Use asynchronous messaging for communications among distributed processors.
- Incorporate threat analysis into the selection of architectural primitives. Demonstrate how primitives mitigate known risks; i.e., demonstrate that instantiation satisfies the theory of operation specified in the strategies.
- Refine requirements to reflect available commercial solutions.
  - Refine authorization and authentication requirements—portals, directory services, access control associated with databases.
  - Refine requirements and specify and develop preliminary architecture for boundary controllers—firewalls.
  - Identify architectural patterns: filters, proxy, roles, directory, wrapper, sandbox.
- Identify tradeoffs—e.g., administration costs, COTS software integration, ease of use for user and/or administrator, modifiability. COTS security tools currently have limited interoperability.
- Refine requirements for, specify, and develop preliminary architecture for system-wide administration. Centralized management for authentication and authorization could require directory services.

4.2 Supporting Techniques

At the architectural instantiation level, developers need to identify a particular means to realize the survivability strategies. We realize survivability strategies via survivability primitives, similar to the way that attribute primitives support achieving quality attributes in an architecture as described by Len Bass et al. [Bass 01]. There have been many listings of primitives for ensuring system security through the years. Two recent works by Anderson and Ramachandran are particularly noteworthy, the latter of which describes useful primitives in the context of architectural design [Anderson 01, Ramachandran 02]. The IATF, described earlier, also presents many security and survivability primitives graduated according to strength of protection against malicious attack. Depending on the threat level expected and properties required of the application domain, the framework recommends particular primitives along with the Common Criteria assurances for the system as a whole.

4.2.1 Quality Attribute Primitives

A widely held premise of the software architecture community is that architecture determines quality attributes such as performance, reliability, and modifiability. The SEI is currently studying the relationship between software architecture and quality attributes [Bass 00]. That work addresses several questions:

- Why are architecture and quality attribute behavior so intrinsically related?
- What key architectural decisions affect specific quality attributes?
- How do architectural decisions serve as focal points for tradeoffs between several quality attributes?

The primary objective of the work is to systematically codify the relationship between architecture and quality attributes so as to achieve the following benefits:

- A greatly enhanced design process—both generation and analysis. During design generation, a designer could reuse existing analyses and determine tradeoffs explicitly rather than in an ad hoc basis. Experienced designers do this intuitively but even they would benefit from codified experience. For example, during analysis, designers could recognize a codified structure and know its impact on quality attributes.
- A method for manually or dynamically reconfiguring architectures to provide specified levels of a quality attribute. Understanding the impact of quality attributes on architectural mechanisms will enable designers to replace one set of mechanisms for another when necessary.
- The potential for third-party certification of components and component frameworks. Once the relationship between architecture and quality attributes is codified, it is possible to construct a testing protocol that will enable third party certification.

Bass characterizes the relationship between architecture and quality attributes using *general scenarios* and *general mechanisms*. General scenarios characterize quality attribute requirements in terms of a stimulus and a response measure. For example,

- A modifiability general scenario is spurred by *changes arriving* and results in *their propagation through the system specification and implementation*. Modifiability general scenarios reflect the various classes of change possible.
- A performance general scenario is spurred by *events arriving* and results in *a response to the event with some latency*. Performance general scenarios reflect the various classes of performance response required.

Bass proposes that a collection of such system-independent scenarios can serve to completely characterize a quality attribute requirement. Furthermore, general mechanisms exist for each quality attribute that can serve as primitives for architecting systems to satisfy attribute requirements. For example,

- Encapsulation is a general mechanism intended to primarily improve modifiability by limiting the ripple effect of changes.
- Replication is a mechanism intended to improve performance by reducing response time through locality or improving reliability by providing redundant copies of function or data.

The vision promulgated is that "for a given mechanism we can divide the general scenarios into those that the mechanism is intended to achieve and those that the mechanism impacts as a side effect. The analysis for the intended general scenarios explains how the mechanism achieves its result with respect to that general scenario. The analysis for the other general scenarios describes how to refine the general scenario in light of the knowledge provided by the mechanism. This refinement reveals side effects the mechanism has on other general scenarios" [Bass 00]. A codification of this information would be a useful resource for an architect designing a system to meet its required quality attributes. In particular, specializations of general scenarios, called *specific scenarios*, describe system-dependent quality attribute re-

quirements. Likewise, specializations of general mechanisms, called *specific mechanisms*, are the actual components used to design a system.

The validation of this approach is ongoing [Liu 01]. Nevertheless, the notion that mechanisms can serve as architectural design primitives for achieving a quality attribute has clear relevance to the design of survivable systems. A survivability general scenario is spurred by *attacks perpetrated* and results in *resistance, recognition, and recovery so as to continue essential services*. Survivability general scenarios reflect the various classes of requirements to resist, recognize, and recover from attacks. Instantiated attack patterns represent specific scenarios. We can form intrusions that may compromise system survivability by stringing together coherent, specific scenarios. The architecture refinement process that we define introduces survivability primitives into the architecture iteratively, to address attacks that target different elements and that require increasing degrees of attacker sophistication. Just as with other quality attributes, these mechanisms serve to satisfy survivability scenarios, which characterize survivability.

4.2.2 Architectural Views

Traditionally, architectural views document the expected usage of the system. While Sector I describes the general theory of operations for the system with respect to the identified threat scenarios, Sector II documents the technological implementation that supports those operations. Two example views are the component and connector view and the architectural resource view [Clements 02]:

- A component and connector view concentrates on the runtime behavior of the system. Components in this view include servers, clients, communication links, processes, and data stores. A component and connector view also has associated styles such as client server, central data store, or pipe and filter, which impact the choice and implementation of survivability strategies. A runtime view documents the normal behavior of the system for the essential services. Such a view can also be used to document intruder behavior and the roles of components such as applications, firewalls, and portals in the response to an attack. The runtime view also documents data flow, which can assist in data recovery after an attack.
- An architectural resource view maps the components and connectors onto hardware. The resource view can be used to map the workflow associated with an essential mission service onto the physical components. This mapping now supports vulnerability and risk analysis. For example, a vulnerability associated with a service could affect the other services collocated on a server. The security policies and operating environment associated with a physical resource are also critical to the analysis.

5 Environmental Analysis Sector

The primary objective of the Environmental Analysis sector (Sector III) is to assess at any stage of architectural refinement the impact of a potentially evolving threat environment on the system and its overall mission as described. A broad, but not uncommon, view of threat includes the potential harmful results due to malicious attack, user errors/lapses, technological faults, and natural disasters. Our current efforts limit the scope of the analysis in the Environmental Analysis sector to malicious attack, since threats due to unintentional acts, faults, or accidents are random events that can be analyzed with existing dependability and fault tolerance techniques. Malicious attacks, however, often involve the worst possible set of contrived inputs or actions delivered at the most inopportune time, resulting in mission failure.

5.1 Approach

Figure 8 outlines the general structures for representing and analyzing the threat environment in TRIAD. Inputs include any generic reusable attack patterns and any architectural descriptions from progress in previous sectors. The ultimate output of the Environmental Analysis sector, an assessment of residual risk, is derived by looking at the threat environment from three perspectives: threat dynamics, intrusions, and individual attacks.

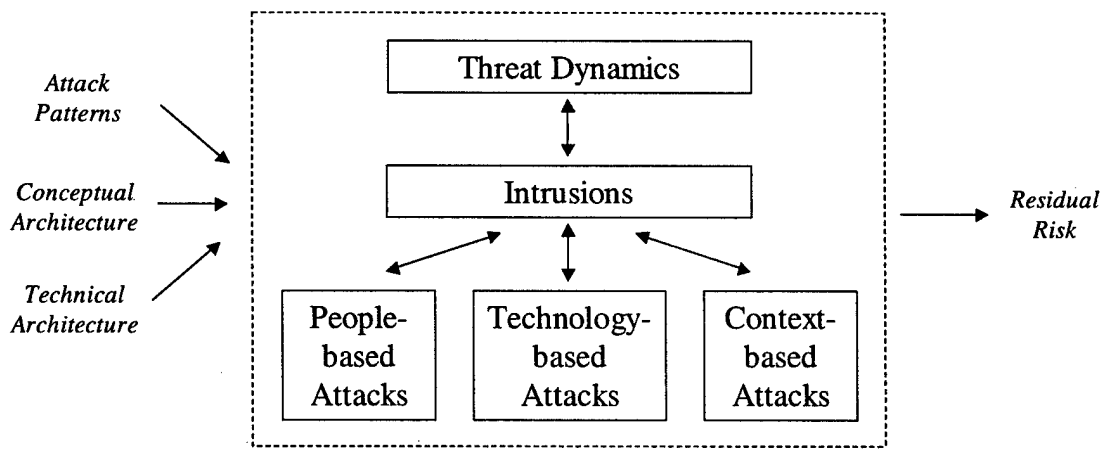


Figure 8: Threat Environment Analysis

Threat dynamics defines a holistic view of the threat environment. The objective is to provide an overview of the general influences that the threat environment can have on the ability of

the system to fulfill its mission. This big picture view permits analyzing dynamically the effects of

- changes in attacker activity
- system operational responses to attacker activity
- changes in system operations or architecture
- the availability of new data that characterizes perceived threats in a new light

The effects of primary concern are those that compromise the system's ability to achieve its mission.

Threat dynamics is based on a field of study called system dynamics, which has been used extensively to model the structure and dynamics of complex human-based systems [Sterman 00]. While system dynamics is widely applicable, it is most useful in systems that use derived information to exert feedback control over its resources. Such feedback control is a critical technique for building survivable information systems. Active defense strategies monitor attack activity and respond through a variety of recovery and adaptation techniques to ensure mission success. Thus, survivable systems control their information resources based partly on feedback from the attack-monitoring activity. System dynamics helps represent and analyze such feedback control but has generally not assumed the presence of hostile agents. Threat dynamics extends, or interprets, system dynamics to include explicitly hostile actions and the system operational response to such actions. Threat dynamics thus models the structure and dynamics of complex, human-based systems, of which the relationship between the Internet-based attacker community and Internet-based information systems is a specific example.

Elaborating relevant intrusions and attacks permits a more detailed analysis. We define an intrusion as a sequence of more primitive attacks that leads to a specific compromise of the system's mission. An attack may or may not be completely successful, but it always changes the state of the system in some way. An intrusion, on the other hand, always leads to a specific mission compromise through the execution of the sequence of at least partially successful attacks. Related intrusions can be conveniently organized into attack trees where the root of the tree describes the mission compromise to which the intrusions contribute [Schneier 00b].

Individual attacks can be broadly classified as to whether they are people-based, technology-based, or context-based. These attack classes, respectively, target

- people's wants, needs, capabilities, or perceptions. Examples include social engineering, semantic attacks, extortion, and physical harm. Such attacks can exploit greed, fear, or gullibility; corrupt morals; or incapacitate essential personnel.
- computing and networking technology. Examples include
  - network-based attacks: attacks on communication infrastructure and supporting services

- application-based attacks: attacks on the architecture component applications such as a Web server, email services, or supporting application infrastructure
- data-centered attacks: attacks on the data stream or content presented by transactions. Such attack patterns can exploit or corrupt data and services or disrupt or deny essential services.

- the context in which people perform their jobs. Examples include attacks on work support, customer demand, the value of corporate stocks, or legal constraints under which people and corporations work. Such attacks can exploit or deny critical resources or damage corporate market, capability, or assets.

The approach outlined above manages the complexity of the survivability risk analysis problem by focusing only on threats that can compromise the mission and only on vulnerabilities at a gross architectural level. The holistic nature of the threat dynamics starting point helps to ensure that all potential threat and solution areas are considered down to an architectural level of analysis. Threat dynamics provides a means for analyzing the effects of observed trends in attacker behavior. Linkages between the threats and the risk mitigators are preserved through the requirements traceability performed in the Architectural Strategy sector. Although, as we noted earlier, good incident and vulnerability data is increasingly becoming available, there are still large gaps in our understanding of intruder behavior. Our methods benefit from the availability of such data where it exists, but do not depend on it in order to provide useful insights into the impact of the threat environment on system operations. Threat dynamics analysis, and its system dynamics basis, can be performed in a qualitative, a quantitative, or combined manner [Wolstenholme 90, Coyle 00].

5.2 Supporting Techniques

5.2.1 Security Risk Analysis

Security risk analysis is an established field of study that involves the analysis of the threats to, and vulnerabilities of, a system and their potential impact on the system's mission. The three primary elements of risk can be defined as follows [DoD 99]:

1. threat: any circumstance or event with the potential to cause harm to a system
2. vulnerability: a system characteristic that could be exploited by a threat to harm a system
3. impact: the extent of harm to a system that results from a threat's exploitation of a system vulnerability

Risk is formally defined as "a combination of the likelihood that a threat will occur, the likelihood that a threat occurrence will result in an adverse impact, and the severity of the resulting impact" [DITSCAP 99].

For our purposes, then, a malicious threat can be viewed as any activity that exploits a vulnerability in a system and results in a negative impact on mission success.[2] TRIAD involves survivability risk mitigation at an architectural level, and the Environmental Analysis sector involves survivability risk analysis in support of the mitigation. We do not "reinvent" security risk analysis, but leverage existing analysis techniques as appropriate. In the longer term, we hope to improve the accuracy and speed of risk analysis techniques by documenting commonly recurring attack patterns in a generic and reusable form.

Experience over the years in security risk analysis suggests a number of pitfalls to avoid [Soo Hoo 00].

- *Complexity*. Techniques often require explicitly considering all threats and vulnerabilities from the most common to the most obscure, without some screening with regard to likelihood or impact. The resulting complexity tends to overwhelm the analysis.
- *Incompleteness*. Techniques often ignore key aspects of the risk management problem or make incorrect assumptions about the problem domain. This may, for example, result in technological threats or solutions being emphasized over procedural ones.
- *Data unavailability*. Techniques often require obtaining precise, quantitative data on likelihood of threats and severity of impact. In the real world, such data continues to be inconsistently collected and reported, and highly uncertain even when it is. Using highly uncertain "estimates" in places where precise data is required often leads to obviously faulty results or, even worse, to very misleading, but plausible, nonsense.
- *Threat/countermeasure decoupling*. Techniques of managing security risk through the use of security technology best practices tend to decouple the countermeasures with the risk they are supposed to reduce. This lack of traceability makes it difficult to accurately assess the actual residual risk resulting from the use of those practices.
- *Static analysis*. Techniques generally deal only with the current threat environment with little regard to managing the system under changing threats. Increasingly rapid changes in the threat environment, which is characteristic of modern Internet-based systems, demand techniques that can be applied as part of an evolutionary design and maintenance life cycle.

There are, of course, no easy solutions to these problems. Early research in security risk analysis generally promoted comprehensive solutions that became overly complex. More recent approaches simplified the methods at the expense of completeness [Soo Hoo 00]. While we make no claims to having solved these problems, we believe our approach to threat analysis takes a balanced approach that makes inroads to managing the risk analysis problem from the survivability perspective.

[2] Henceforth, we refer to "malicious threat" simply as "threat," since this is our primary focus. We specifically refer to "non-malicious threats" where that distinction is needed.

5.2.2 System Dynamics

System dynamics is one model that can help describe and enable understanding of the impact of an evolving threat environment on the operation of a system. The originator, Jay Forrester, developed system dynamics to show how a model of the structure of a human activity system and the policies used to control it could be used to deepen our understanding of the operation and behavior of that system [Forrester 61]. System dynamics has been used extensively since then as a general modeling tool to enable better understanding of the structure and dynamics of complex human-based systems, particularly in the area of business strategy and public policy [Sterman 00]. Preliminary literature searches have yielded very little work that applies system dynamics to study the effectiveness of information technology. One of the few works in this area describes an approach using system dynamics to study the impact of introducing a management information system on an organization's business objectives [Wolstenholme 93]. We are aware of no work using system dynamics to study the threat environment or its impact on system operations.

Nevertheless, system dynamics does appear to help with the documentation and analysis of the threat environment, the area that we call threat dynamics. The simplest form of qualitative problem description and analysis in system dynamics is the influence diagram, which is based on feedback loops. Figure 9 illustrates a feedback loop that describes an aspect of the behavior to control the vulnerability of Internet-based systems [Arbaugh 00]. Starting at the "effectiveness of vulnerability exploit" element at the lower left-hand side of the figure, we see that effectiveness positively influences the rate of publication about the vulnerability, in the sense that an increase in effectiveness leads to an increase in publication (perhaps due to increased media attention) with all other things being equal. Likewise, increased publication leads to increased incentive to fix, and the ultimate availability of relevant patches. This leads to patching of systems, which in turn reduces the effectiveness of the vulnerability exploit (all other things being equal). The delay in patching, signified by the "**D**" along the arrow on the right side of the figure, is a trend that has been described as a major reason for heavy Internet-based attack activity, and the general vulnerability of the Internet. Nevertheless, the overall feedback loop described is a balancing one (indicated by the negative loop symbol in the center), in that patching generally helps to control overall Internet vulnerability.

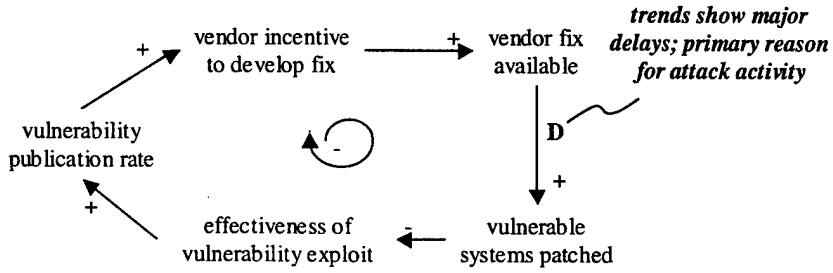


Figure 9: A Feedback Loop for Controlling Vulnerability

System dynamics influence diagrams can be composed as illustrated in Figure 10. The right side of the figure shows the influence diagram described above. The left side shows a feedback loop that describes an effect of the vulnerability publication rate on the publication of exploit tools and, ultimately, on the attacker exploit of the vulnerability. This is an example of a positively reinforcing feedback loop as indicated by the positive loop symbol in the center of the feedback loop. This figure illustrates a great debate ongoing in the Internet community as to whether publishing vulnerability data helps or hinders the overall security of the Internet. Recent analysis indicates that delays in patching are the primary cause of Internet vulnerability, while the publication of vulnerability data is a secondary driving force [Arbaugh 00]. The diagram does not, of course, help resolve the debate since it is strictly qualitative in nature.

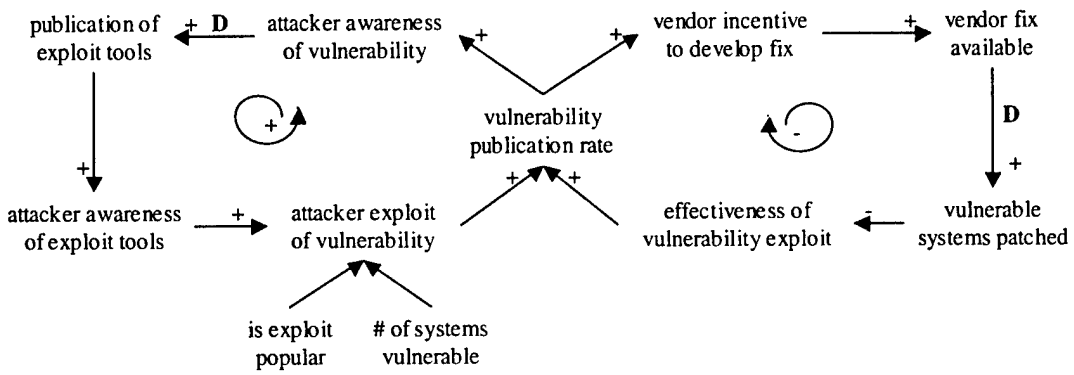


Figure 10: The Effects of Vulnerability Publication on Internet Vulnerability

Figure 11 presents an example of the use of system dynamics to characterize the impact of recent trends in the threat environment on an eBusiness's recent decline in profits, despite an attempt to use more stringent technological security controls to curb high customer repudiation rates. The analysis indicates that while increased security might control fraudulent transactions, it also drives customers away. This is a plausible scenario if, for example, the eBusiness required digital signatures, for non-repudiation, on all transactions. Most customers will go somewhere else before subscribing to the third-party certification of digital signatures that would be required to do business with the company. In this case, therefore, the likely result would be that the increases in profit due to lower rates of repudiation (arrow from repudiation rate to profit, bottom-left) would be overtaken by decreases in profit due to loss of customers (arrow from customer loss rate to profit, lower left side).

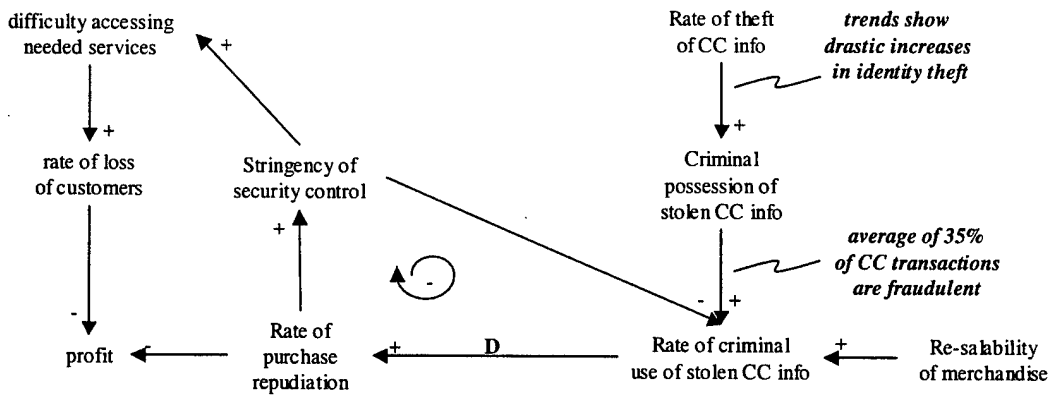


Figure 11: eBusiness Threat Analysis

5.2.3 Intrusion Scenarios

Intrusion scenarios involve interactions from the adversary's view, a negative view with respect to system functionality, rather than a normal legitimate user's view, a positive view. We define an intrusion scenario as a description of people and systems interacting in a way that characterizes malicious behavior causing harm to an organization. A related technique called abuse cases leverages the use case concept of the Unified Modeling Language™ (UML) for information security [McDermott 99]. The most common view is that a use case is a general specification of a set of related concrete usage scenarios. Abuse cases are to use cases as intrusion scenarios are to usage scenarios; i.e., they take an adversary's view rather than a user's view. Therefore, we can view abuse cases as a standard way to describe a set of related intrusion scenarios. UML explicitly identifies actors in a use case diagram and shows how these actors interact with the system. Attackers correspond to the actors of an abuse case diagram. Abuse cases describe these malicious actors in detail according to their resources, skills, and objectives.

The large number of intrusions possible for any nontrivial system necessitates a scheme to organize related intrusions. Attack trees provide such an organizational scheme [Salter 98, Schneier 99, Schneier 00b]. They refine information about intrusions by identifying the compromise of enterprise security or survivability as the root of the tree. The ways that an attacker can cause this compromise are refined incrementally as lower level nodes of the tree. For example, suppose that a malicious competitor to an eBusiness may compromise that business's ability to make a profit selling their product by

- hampering development of the product
- disrupting sales of the product
- undermining customer demand for the product

Each of these attack classes can be refined as a separate branch of the attack tree.

™ Unified Modeling Language is a trademark of Rational Software Corporation.

A system typically has a set, or forest, of attack trees that are relevant to its operation. The root of each tree in a forest represents an event that could significantly harm the system's mission. Each attack tree enumerates and elaborates the ways that an attacker could cause the event to occur. Each path through an attack tree represents a unique intrusion on the enterprise. We decompose a node of an attack tree as one of the following:

- a set of attack subgoals that is represented as an AND decomposition. All of these goals must be achieved for the attack to succeed.
- a set of attack subgoals that is represented as an OR decomposition. If any of these goals is achieved, the attack succeeds.

We represent decompositions graphically as shown in Figure 12. The AND-decomposition represents a goal G_0 that can be achieved if the attacker achieves all of the goals G_1 through G_n. The OR-decomposition represents a goal G_0 that can be achieved if the attacker achieves any one of goals G_1 through G_n. In practice, we often represent attack trees textually, since the graphical representation can be awkward for nontrivial attack trees. Figure 13 shows the high-level attack tree for the eBusiness described above.

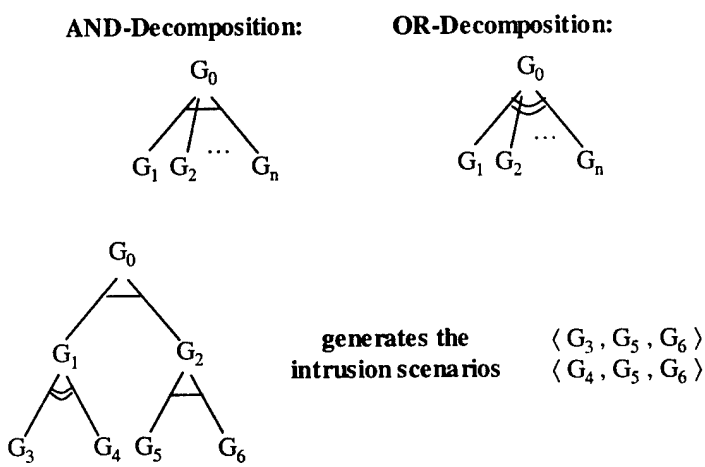


Figure 12: Attack Tree Representation

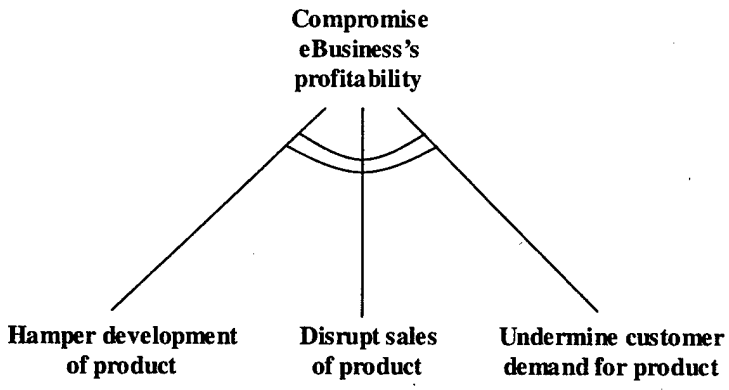


Figure 13: High-Level eBusiness Attack Tree

Attack trees consist of any combination of AND- and OR-decompositions. We generate individual intrusion scenarios from an attack tree by traversing the tree in a depth-first manner, an example of which is shown in Figure 12. In general, leaf goals are added onto the end of intruder workflows as they are generated. OR-decompositions cause new workflows to be generated. AND-decompositions cause existing workflows to be extended. Intermediate nodes of the attack tree do not appear in the intrusion scenarios, since they are elaborated by lower level goals.

Attack trees allow the refinement of attacks to a level of detail chosen by the developer. They exhibit the property of referential transparency as characterized by Prowell:

> *Referential transparency implies that the relevant lower level details of an entity are abstracted rather than omitted in a particular system of higher level description, so that the higher level description contains everything needed to understand the entity when placed in a larger context [Prowell 99].*

This property permits the developer to explore certain attack paths in more depth than others, while still allowing the developer to generate intruder workflows that make sense. In addition, refining the branches of the attack tree generates new leaves resulting in intruder workflows at the new lower level of abstraction. The notion of referential transparency is critical to managing the complexity inherent in attack tree representations by constraining the refinement to an architectural level of abstraction.

6 IAD Model Context

This section describes how TRIAD can be applied in practice. Section 6.1 illustrates how to instantiate the generic TRIAD model for use in a specific development environment by assembling the sector activities and structures in an organized manner. Section 6.2 describes how TRIAD can be used in the context of more comprehensive system development life cycles.

6.1 Model Instantiation

TRIAD, as described in Section 2, is very generic in nature, partitioning the design space into three primary sectors: Architectural Strategy, Architectural Instantiation, and Environmental Analysis. Sections 3 through 5 described in more detail the primary activities that need to occur and structures that need to be documented in each of the model sectors. These activities and structures could be assembled into a specific, working model in many ways. The details of the best assemblage will depend largely on the domain of application and the skills of the development team. Nevertheless, Figure 14 illustrates one approach that is still very general, but shows how one might start the IAD spiral process, followed by iteration through the sector activities, and completing when an acceptable degree of residual risk is determined. A similar illustrative model could be developed for enhancing an already existing architecture into one with improved survivability properties.

Figure 14 shows three iterations of the spiral, culminating in acceptable residual risk. The first iteration shown in the center of the figure starts the process off at a very high level of abstraction by characterizing overall mission objectives, the general concept of operations, and any structural constraints or external interfaces with which any architecture will have to conform. The idea at this point is not to necessarily have a firm idea of how the survivability of the mission would be ensured, but just to get a general idea of what the information system needs to accomplish and some idea of how this can be done within existing constraints. Sector III activities of this first iteration involve establishing who the adversaries to the organization are likely to be and, from a high-level point of view, how they may impact system operations. Threat dynamics, as discussed previously, should be useful in this preliminary analysis.

The second and third iterations of the spiral in Figure 14 cycle through the major sector activities. They cover strictly development concerns and do not deal with how they might be integrated with evolution and maintenance processes—the subject of the next section. As in-

dicated, the generic survivability strategies will play a primary role in mitigation analysis and the derivation of survivability requirements from this analysis. Survivability primitives are central to establishing the technical architecture in Sector II. Requirements traceability plays a major role in Sector I, through the top-down tracing of mission objectives to system operations, and in Sector II, through the bottom-up tracing of component responsibilities to the survivability requirements. Generic attack patterns provide a foundation for the threat dynamics analysis and attack tree refinement of Sector III. All of this information informs the residual risk assessment, which must also establish the likelihood and severity of potential attacks.

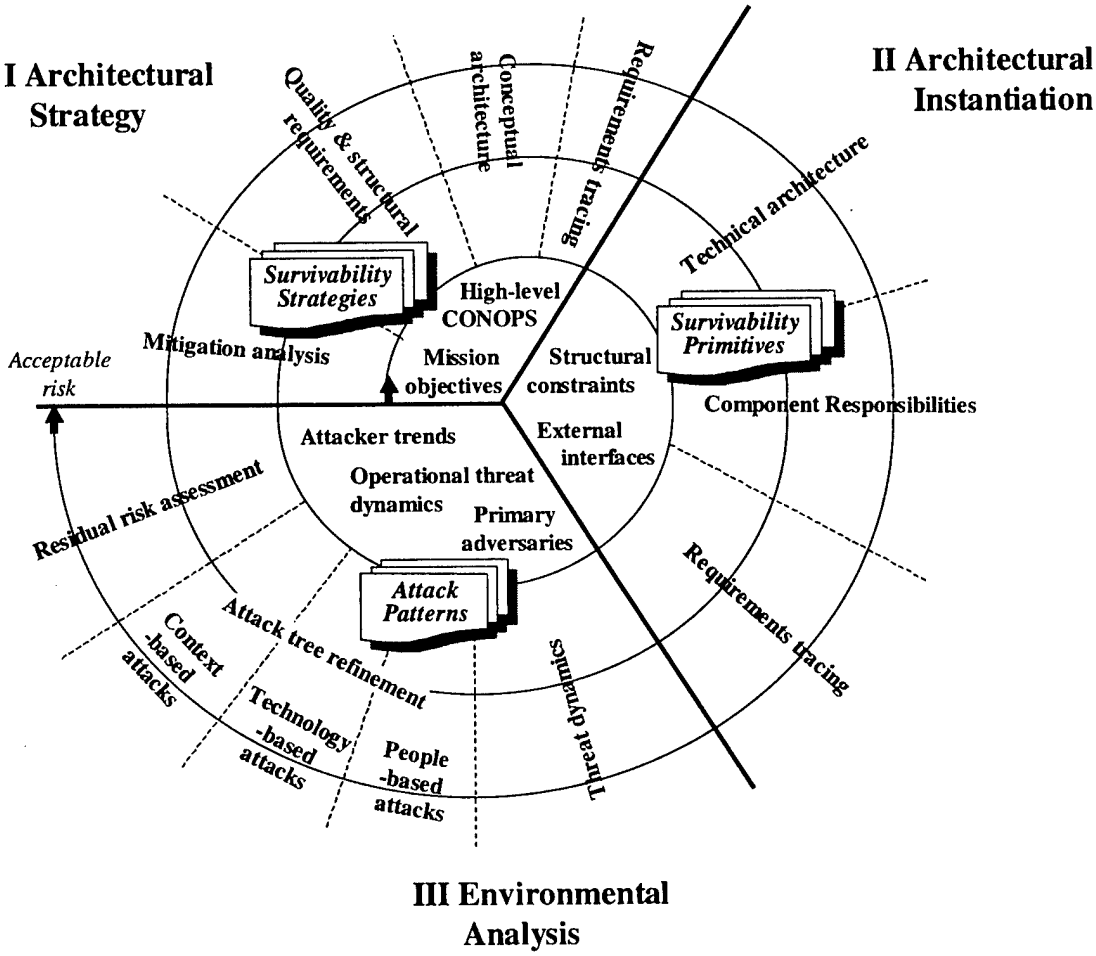


Figure 14: Example Model Instantiation

As mentioned previously, we justify the basis of TRIAD in the spiral model since survivable systems development is a domain in which the best directions for refinement are very unclear during the early stages of system conception and refinement. Experimentation and analysis is needed before a solution can be found with an acceptably small degree of residual risk of mission failure. The specification and analysis performed within each sector is gradually refined based on the experience of the previous iteration. Greater or fewer iterations may be

necessary, depending on the details of the specific application. We now turn to how we expect TRIAD design activities to be integrated with the general system development life cycle.

6.2 Model Incorporation

TRIAD deals with only a small, but important, part of the survivable system development life cycle. In particular the model does not deal specifically with

- the implementation, evolution, or maintenance of the derived survivability architecture
- functions or properties required or desired of the system that do not contribute to the mission
- survivability-relevant failures due to internal faults or accidents
- program risks, such as funding or development team shortfalls, that are not due to malicious activity

Incorporating TRIAD into an overall system development and maintenance (SDM) process will help to resolve many of these issues. A detailed approach of how to do this depends largely on the details of the system problem domain and the development environment, and is beyond the scope of this report. We do, however, discuss some of the issues involved to provide a basis for formulating a comprehensive SDM process that incorporates IAD concepts. Fortunately, iterative spiral models are as useful for characterizing system maintenance (or enhancement) as they are for system development [Boehm 88].

As mentioned previously, the development of high-confidence information systems in complex settings where the impact of intrusion failure is severe demands an iterated, risk-driven process like the spiral model to gradually resolve uncertainties in the most efficacious manner. Using TRIAD in the context of a comprehensive SDM spiral can proceed in two primary ways (see Figure 15):

1. viewing TRIAD as an up-front mini-spiral. In this case, execution of the IAD process leads to an advanced starting point for the larger SDM spiral.
2. unrolling TRIAD activities and documented structures into the first few cycles of the SDM spiral. In this case, a more comprehensive integration of the two processes occurs.

The first of these methods is possible due to TRIAD's focus on mission. We believe that mission-related survivability requirements must be used to determine the overall shape of the architecture and must, therefore, be the focus of the initial iterations of the design process. Functions or properties required or desired that do not contribute to the mission must fit within the parameters defined by the survivability architecture and must not significantly lower the confidence that the system owners have in that architecture.

The first method described above does not specifically permit considering risks associated with non-malicious activities or events during the refinement of the survivability architecture. TRIAD can be extended in a fairly straightforward manner to deal with survivability-related

non-malicious failures and accidents. Threat dynamics modeling and analysis of the impact of external failures and natural accidents can proceed in much the same manner as for malicious attacks. Accurately predicting the impact of internal faults on the mission may require specifying greater detail of internal operations in the threat dynamics model. In addition, attack tree modeling can be easily extended with fault tree analysis to analyze faults and accidents at a lower level of abstraction, because of the parallels between the two techniques.

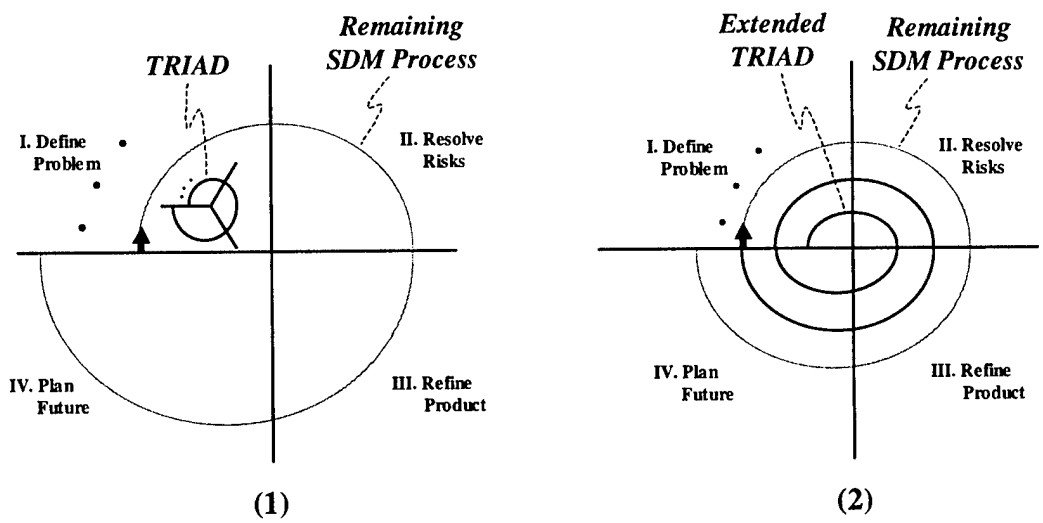


Figure 15: TRIAD in SDM Process (1) as Mini-Spiral or (2) Through Integration

Non-malicious program risks, such as resource shortfalls, are more difficult to handle with the first approach, since TRIAD deals only with operational risks during architecture formulation. Choosing the second approach is appropriate if program risks are high or if effectively dealing with them explicitly within the process cannot be postponed until after architecture formulation. In this case, integrating IAD activities into the first few cycles of the comprehensive system development spiral permits resolving program risk early on, before too many resources are expended in a programmatic dead end.

7 Conclusion

This report outlines an IAD model, called TRIAD, for systematically refining information system architectures in complex, potentially unbounded, domains to prevent, monitor, detect, and adapt to known and hypothesized patterns of attack. TRIAD facilitates planning for the inevitable change to the threat and operational environment and helps trace the effect of change back to the survivability requirements and architecture. The spiral structure of the model iterates through three sectors of activity for developing architectural strategies, for instantiating the architecture using architectural primitives, and for analyzing the impact of the threat environment on system operations. TRIAD can be incorporated into the full system development life cycle either as a separate up-front mini-spiral or by more fully integrating design activities into the life-cycle process.

Although TRIAD presents only a broad-brush sketch of IAD activities, it provides a starting point for the further refinement, experimentation, and validation of an approach to exploit our understanding of intruder behavior to improve system architecture design and operations. In the near term, we plan to continue to explore the viability of and refine TRIAD through its application to the focused analysis of very specific problem situations. Each example will involve the identification of a specific problem situation, a TRIAD analysis and mitigation of that situation, and a characterization of the improvement gained through the analysis and mitigation. The improvement characterization will be a comparison of the problem situation before and after TRIAD analysis and mitigation.

By focusing on a specific problem in a narrow domain, we expect to get quick feedback on the efficacy of the model and insights into how to improve it. Feedback will help us understand the relationship and dependencies among sector activities and data. Different problem/mitigation approaches will be investigated in the examples to increase the experience gained and insights gleaned, e.g., passive versus active defenses, military versus commercial domains, COTS versus custom solutions, and technological versus procedural countermeasures. In each case, the problem situation will be restricted to a particular malicious threat and its impact in the domain of interest. We expect this focusing to streamline the TRIAD mitigation and analysis to one iteration of the full model, with little or no formal requirements tracing, thus ensuring the relative expediency of results.

Later work will involve a full-scale application of TRIAD to demonstrate its scalability to more complex problems. This demonstration requires assembling the TRIAD activities and structures into a working system development life-cycle model appropriate to the application

domain and development environment. This report documents one approach that shows how one might start the TRIAD spiral process, followed by iterations through the sector activities, and completing when an acceptable degree of residual risk is determined. In addition to refining TRIAD based on the full-scale application, we plan to develop a tutorial for its use, with relevant examples, and initiate transition of the technology to an interested organization. Documentation of these TRIAD case studies and a detailed set of guidelines for TRIAD's application in varied settings should help make a compelling case for the model's use and transition. Ultimately, with evidence of its efficacy, we expect that TRIAD will be integrated with more comprehensive life-cycle models for the development and maintenance of high-confidence systems.

Bibliography

[Anderson 93] Anderson, R. "Why Cryptosystems Fail." *Communications of the ACM* 37, 11 (November 1994):32-40.

[Anderson 99] Anderson, R. H.; Feldman, P. M.; Gerwehr, S.; Houghton, B. K.; Mesic, R.; Pinder, J.; Rothenberg, J.; & Chiesa, J. R. *Security of the U.S. Defense Information Infrastructure: A Proposed Approach* (RAND Report MR-993-OSD/NSA/DARPA). Santa Monica, CA: RAND Corporation, 1999.

[Anderson 01] Anderson, R. *Security Engineering: A Guide to Building Dependable Distributed Systems*. New York, NY: John Wiley & Sons, 2001.

[Arbaugh 00] Arbaugh, W. A.; Fithen, W. L.; & McHugh, J. "Windows of Vulnerability: A Case Study Analysis." *IEEE Computer 33*, 12 (December 2000): 52-59.

[Bass 98] Bass, L.; Clements, P.; & Kazman, R. *Software Architecture in Practice*. Boston, MA: Addison Wesley Longman, 1998.

[Bass 00] Bass, L.; Klein, M.; & Bachmann, F. *Quality Attribute Design Primitives* (CMU/SEI-2000-TN-017, ADA392284). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000. <http://www.sei.cmu.edu/publications/documents /00.reports/00tn017.html> (2000).

[Bass 01] Bass, L.; Klein, M.; & Bachmann, F. "Quality Attribute Design: Primitives and the Attribute Driven Design Method." *4th Conference on Product Family Engineering*. Bilbao, Spain, 4 October 2001. <http://www.sei.cmu.edu/plp/bilbao_paper.pdf> (2001).

[Boehm 88] Boehm, B. "A Spiral Model of Software Development and Enhancement." *IEEE Communications 21*, 5 (May 1988): 61-72.

[CERT 02] CERT Coordination Center. *Overview of Attack Trends*. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002. <http://www.cert.org/archive/pdf/attack_trends.pdf> (2002).

[Clements 02] Clements, P.; Bachmann, F.; Bass, L.; Garlan, D.; Ivers, J.; Little, R.; Nord, R.; & Stafford, J. *Documenting Software Architectures*. Boston, MA: Addison Wesley Longman, 2002.

[Coyle 00] Coyle G. "Qualitative and Quantitative Modeling in System Dynamics: Some Research Questions." *System Dynamics Review 16*, 3 (Fall 2000): 225-244.

[DITSCAP 99] U.S. Department of Defense. *DoD Information Technology Security Certification and Accreditation Process (DITSCAP)*. DoD Instruction 5200.40, 30 November 1999. <http://www.sabi.org/history.htm> (2002).

[DoD 99] U.S. Department of Defense. *Introduction to Threats to Department of Defense Information Systems*. Secret and Below Interoperability Initiative Report, 30 September 1999. <http://www.sabi.org/history.htm> (2002).

[DoD 00] U.S. Department of Defense. *Basic Risk Management for Department of Defense Information Systems: Informal Reference Guide Edition 1.1*. Secret and Below Interoperability Initiative Report, 21 January 2000. <http://www.sabi.org/history.htm> (2002).

[Ellison 99] Ellison, R. J.; Fisher, D. A.; Linger, R. C.; Lipson, H. J.; Longstaff, T. A.; & Mead, N. R. *Survivable Network Systems: An Emerging Discipline* (CMU/SEI-97-TR-013, ADA341963). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1997, revised 1999. <http://www.sei.cmu.edu/publications/documents/97.reports/97tr013/97tr013abstract.html> (1999).

[Fisher 99] Fisher, D. A. & Lipson, H. J. *Emergent Algorithms: A New Method for Enhancing Survivability in Unbounded Systems.* Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999. <http://www.cert.org/archive/html/emergent-algor.html>.

[Forrester 61] Forrester, J. W. *Industrial Dynamics*. Republished by Productivity Press, Portland, OR. Cambridge, MA: MIT Press, 1961.

[IATF 00] Information Assurance Technical Forum. "Information Assurance for the Tactical Environment." IATF Release 3.1, September 2002. <http://www.iatf.net/framework_docs/version-3_1/> (2002).

[Jacobson 99] Jacobson, Ivar; Booch, Grady; & Rumbaugh, James. *The Unified Software Development Process*. Boston, MA : Addison Wesley Longman, 1999.

[Knight 00] Knight, J. C.; Sullivan, K. J.; Elder, M. C.; & Wang, C. "Survivability Architectures: Issues and Approaches." *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition (DISCEX 2000)*. Hilton Head, South Carolina, Jan. 25-27, 2000. Los Alamitos, CA: IEEE Computer Society, 2000.

[Liu 01] Liu, A.; Bass, L.; & Klein, M. *Analyzing Enterprise JavaBeans Systems Using Quality Attribute Design Primitives* (CMU/SEI-2001-TN-025, ADA396123). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001. <http://www.sei.cmu.edu/publications/documents/01.reports/01tn025.html> (2001).

[MAFTIA 02] MAFTIA Partners. Malicious- and Accidental-Fault Tolerance for Internet Applications, IST Programme RTD Research Project IST-1999-11583. <http://www.newcastle.research.ec.org/maftia> (2002).

[Maier 00] Maier, M. W. & Rechtin, E. *The Art of Systems Architecting*. Boca Raton, FL: CRC Press, 2000.

[Marmor-Squires 89] Marmor-Squires, A. B.; McHugh, J.; Branstad, M.; Danner, B.; Magy, L.; Rougeau, P.; & Sterne, D. "A Risk Driven Process Model for the Development of Trusted Systems." 184-192. *Proceedings of the 1989 Computer Security Applications Conference.* Tucson, Arizona, December 4-8, 1989. Los Alamitos, CA: IEEE Computer Society, 1990.

[McDermott 99] McDermott, J. & Fox, C. "Using Abuse Case Models for Security." *Proceedings of the 15th Annual Computer Security Applications Conference*. Phoenix, Arizona, Dec. 6-10, 1999. Los Alamitos, CA: IEEE Computer Society, 1999. <http://www.computer.org/proceedings/acsac/0346/0346toc.htm> (2002).

[McHugh 00] McHugh, J.; Christie, A.; & Allen, J. "Defending Yourself: The Role of Intrusion Detection Systems." *IEEE Software 17*, 5 (September/October 2000): 42-51.

[Mead 00] Mead, N.; Ellison R.; Linger R.; Longstaff, T.; & McHugh, J. *Survivable Network Analysis Method* (CMU/SEI-2000-TR-013, ADA383771). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000. <http://www.sei.cmu.edu/publications/documents/00.reports /00tr013.html> (2000).

[Moore 01] Moore, A. P.; Ellison, R. J.; & Linger, R. C. *Attack Modeling for Information Security and Survivability* (CMU/SEI-2001-TN-001, ADA388771). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001. <http://www.sei.cmu.edu/publications/documents/01.reports /01tn001.html> (2001).

[Neumann 00] Neumann, P. G. *Practical Architectures for Survivable Systems and Networks*. Technical report. Computer Science Laboratory, SRI International, Menlo Park, CA, 30 June 2000. <http://www.csl.sri.com/neumann/survivability.pdf> (2000).

[Potts 95] Potts, C. "Using Schematic Scenarios to Understand User Needs." 247-256. *Proceedings of DIS'95—ACM Symposium on Designing Interactive Systems: Processes, Practices, Methods, & Techniques*. Ann Arbor, Michigan, Aug. 23-25, 1995. New York, NY: ACM Press, 1995.

[Prowell 99] Prowell, S. J.; Trammell, C. J.; Linger, R. C.; & Poore, J. H. *Cleanroom Software Engineering: Technology and Process*. Boston, MA: Addison Wesley Longman, 1999.

[Ramachandran 02] Ramachandran, J. *Designing Security Architecture Solutions*. New York, NY: John Wiley & Sons, 2002.

[Ramesh 97] Ramesh, B. *Annals of Software Engineering 3* (1997): 397-415.

[Ramesh 98] Ramesh, B. "Factors Influencing Requirements Traceability Practice." *Communications of the ACM 41*, 12 (December 1998):37-44.

[Salter 98] Salter, C.; Saydjari, O.; Schneier, B.; & Walner, J. "Toward a Secure System Engineering Methodology." *Proceedings Of New Security Paradigms Workshop*. Charlottesville, Virginia, Sept. 22-25, 1998. New York, NY: ACM Press, 1998.

[Schneier 99] Schneier, B. "Attack Trees: Modeling Security Threats," *Dr. Dobb's Journal*, December 1999.

[Schneier 00a] Schneier, B. "Closing the Window of Exposure: Reflections on the Future of Security." *Securityfocus.com*, 2000. <http://online.securityfocus.com/guest/3384>.

[Schneier 00b] Schneier, B. *Secrets and Lies: Digital Security in a Networked World*. New York, NY: John Wiley & Sons, 2000.

[Shaw 96] Shaw, M. & Garlan, D. *Software Architecture: Perspectives on an Emerging Discipline*. Upper Saddle River, NJ: Prentice Hall, 1996.

[Soo Hoo 00] Soo Hoo, K. J. *How Much Is Enough? A Risk-Management Approach to Computer Security*. Report of the Consortium for Research on Information Security and Policy, Center for International Security and Cooperation, Stanford University, June 2000. <http://ldml.stanford.edu/cisac/pdf/soohoo.pdf> (2002).

[Sterman 00] Sterman, J. D. *Business Dynamics: Systems Thinking and Modeling for a Complex World*. Burr Ridge, IL: McGraw-Hill Higher Education, 2000.

[Vatis 01] Vatis, M. A. *Cyber Attacks During the War on Terrorism: A Predictive Analysis*. Hanover, NH: Institute for Security Technology Studies at Dartmouth College, 2001. <http://www.ists.dartmouth.edu/ISTS/counterterrorism /cyber_attacks.htm>.

[van Lamsweerde 00] van Lamsweerde, A. & Letier, E. "Handling Obstacles in Goal-Oriented Requirements Engineering." *IEEE Transactions on Software Engineering 26*, 10 (October 2000): 978-1005.

[Weidenhaupt 98] Weidenhaupt, K.; Pohl, K.; Jarke, M.; & Haumer, P. "Scenarios in System Development: Current Practice." *IEEE Software 15*, 2 (March/April 1998): 34-45.

[Wolstenholme 90] Wolstenholme, E. F. *System Enquiry: A System Dynamics Approach*. New York, NY: John Wiley & Sons, 1990.

[Wolstenholme 93] Wolstenholme, E. F.; Henderson, S.; & Gavine, A. *The Evaluation of Management Information Systems: A Dynamic and Holistic Approach*. New York, NY: John Wiley & Sons, 1993.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE October 2002	3. REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE Trustworthy Refinement Through Intrusion-Aware Design		5. FUNDING NUMBERS F19628-00-C-0003
6. AUTHOR(S) Robert J. Ellison, Andrew P. Moore		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213		8. PERFORMING ORGANIZATION REPORT NUMBER CMU/SEI-2002-TR-036
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116		10. SPONSORING/MONITORING AGENCY REPORT NUMBER ESC-TR-2002-036
11. SUPPLEMENTARY NOTES		
12A DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS		12B DISTRIBUTION CODE

13. ABSTRACT (MAXIMUM 200 WORDS)

High confidence in a system's survivability requires an accurate understanding of the system's threat environment and the impact of that environment on system operations. Unfortunately, existing development methods for secure and survivable information systems often have a patchwork approach in which the focus is on deciding which popular security components to integrate rather than making a rational assessment of how to address the attacks that are likely to compromise the overall mission. This report proposes an intrusion-aware design model called trustworthy refinement through intrusion-aware design (TRIAD). TRIAD enables information system engineers to use known and hypothesized attack patterns to iteratively improve and continually maintain system survivability, even as the system and threat environment evolve over time.

14. SUBJECT TERMS survivability, intrusion-aware design, survivable systems development, system architecting	15. NUMBER OF PAGES 62
16. PRICE CODE	

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL