Architectural Refinement for the Design of Survivable Systems

Robert J. Ellison
Andrew P. Moore

October 2001

Networked Survivable Systems

Technical Note
CMU/SEI-2001-TN-008

Contents

Abstract **v**

1 Introduction **1**

1.1 Problem 1

1.2 Organization 2

1.3 Background 2

2 Survivable Architectural Refinement **4**

3 An E-Commerce Application **7**

3.1 Problem Definition 7

3.2 Network-Based Attacks 9

3.3 Application-Based Attacks 11

3.4 Data-Centered Attacks 14

3.5 Discussion 16

4 Conclusion **18**

References **19**

List of Figures

Figure 1: Architecture Refinement Process 5
Figure 2: Workflow Architecture Notation 6
Figure 3: E-Commerce Workflow 8
Figure 4: Simple Architectural Model of E-Commerce Workflow 9
Figure 5: Architecture Extended to Counter Network-Based Attacks 10
Figure 6: Architecture Extended to Counter Application-Based Attacks 13
Figure 7: Architecture Extended to Counter Data-Centered Attacks 16

Abstract

This paper describes a process for systematically refining an enterprise system architecture to resist, recognize, and recover from deliberate, malicious attacks by applying reusable design primitives that help ensure the survival of the enterprise mission. Systems of interest may be unbounded; that is, have no central administration and no unified security policy. The survivable architecture refinement is an iterative risk-driven process which adopts the structure of Boehm's Spiral Model [Boehm 88]. The cycles of the spiral structure represent different types of attack that need to be considered – network-based attacks, application-based attacks, and data-content attacks. We illustrate our survivable architecture refinement process through its application to e-commerce. E-commerce examples are representative of the lack of full control and visibility that characterize unbounded systems.

1 Introduction

Major investment in information security technology by a business or military enterprise often translates into little, or questionable, value to that enterprise's operational mission. A primary reason for this is that many development efforts focus on deciding which popular security components to integrate rather than on a rational assessment of how to address the attacks that are likely to compromise the enterprise's mission. Such an assessment is a critical aspect of an emerging discipline called survivability, which builds on related fields such as security, fault tolerance, safety, reliability, and performance. Survivability is the capability of an enterprise to continue to fulfill its mission by preserving essential services, even when systems are penetrated and compromised [Anderson 93]. This paper describes a process for systematically refining an enterprise system architecture to resist, recognize, and recover from deliberate, malicious attacks by applying reusable design primitives that help ensure the survival of the enterprise mission.

1.1 Problem

Computer systems, including network technology, play a crucial role in the survivability of an enterprise's mission. System survivability often emerges from the composition of a core set of survivability design primitives. Examples include replication, redundancy, distribution, separation, access control, intrusion monitoring, diversity, and adaptive reconfiguration. Bass uses design primitives, which he calls *mechanisms*, to characterize quality attributes [Bass 00]. We adopt this terminology, but focus on survivability. In particular, the architecture refinement process that we define must iteratively introduce survivability mechanisms into the architecture in a way that balances overall functionality with the need to continue essential services during or after malicious attacks.

Assuring system survivability requires showing that the system architecture is adequately resilient to likely patterns of attack. Many engineering disciplines rely on engineering failure data to improve designs. Unfortunately, information system engineers are generally not using security failure data – particularly attack data – to improve the security and survivability of systems. The increased availability of attack data in books, news groups, and CERT® Coordination Center security advisories, for example, suggests that we can start using this data in a structured way to improve information system security and survivability. We need to abstract, structure, organize, and compose this often low-level attack data in a usable manner [Moore 01].

® CERT and CERT Coordination Center are registered in the U.S. Patent and Trademark Office.

This paper addresses the development of large-scale, highly distributed, and inter-networked systems. Systems cross organizational boundaries and typically have no central administration and no unified security policy. One cannot fully control, or even know, the number and nature of nodes connected to unbounded networks. Any particular observer has only limited knowledge of the complete system. The distinction between insiders and outsiders may be dynamic. A partner for one activity may be a competitor for another. Although no amount of hardening can ensure that intrusions on unbounded systems do not occur, “the discipline of survivability can help ensure that such systems can deliver essential services and maintain essential properties such as integrity, confidentiality, and performance, despite the presence of intrusions” [Ellison 99].

1.2 Organization

We define and illustrate a refinement process for survivable architectures that is both mechanism based and intrusion aware. Section 1.3 provides a context for understanding our survivable architecture refinement process in terms of the general context of system quality attributes. Section 2 describes a spiral-type (i.e., iterative and risk-driven) model for the process. We demonstrate in Section 3 how to use the process through its application to refine a survivable architecture for e-commerce. Finally, we conclude with directions for future work in Section 4. The fact that the process is founded in the Spiral Model should facilitate its integration into the development process of most organizations and thereby make visualizing the benefits of the process and experimenting with its incorporation easier [Moore 01].

1.3 Background

A widely held premise of the software architecture community is that architecture determines quality attributes such as performance, reliability, and modifiability. Bass characterizes the relationship between architecture and quality attributes using *general scenarios* and *general mechanisms* [Bass 00]. 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*. Bass proposes that a collection of such system-independent scenarios can serve to completely characterize a quality attribute requirement. Furthermore, a set of general mechanisms exists for each quality attribute that can serve as primitives for architecting systems to satisfy attribute requirements.

The notion that mechanisms can serve as architectural design primitives for achieving a quality attribute has clear relevance to the design of survivable systems. The notation that we use for describing high-level architectures specifies survivability mechanisms as primitive. Examples include redundancy, detection and response, adaptive reconfiguration, access control, and deception. The architecture refinement process that we define introduces such mechanisms into the architecture iteratively, to survive different types of attacks that require different degrees of attacker sophistication. Just as with other quality attributes by Bass, these mechanisms serve to satisfy survivability scenarios, which characterize survivability [Bass 00].

2 Survivable Architectural Refinement

This section describes the survivable architecture refinement process. We adopt the structure of the Spiral Model since it reflects the iterative, risk-driven process needed to refine survivable architectures [Boehm 88, Marmor-Squires 89]. Rather than modeling the whole development process, we model only that part having to do with system 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. The incorporation of the architecture refinement activities with more general process models of system development is the subject of future work.

The model of the process for refining a system architecture to survive attack is shown in Figure 1. The spiral structure shows a problem-definition stage, starting in the center with the arrow in quadrant I, followed by three cycles of the spiral, each starting at an arrow in quadrant III. The cycles represent different types of attack that need to be considered – network-based attacks, application-based attacks, and data-centered attacks. These attack types correspond to types of survivability scenarios. The spiral proceeds through four quadrants – (I) Survivability Planning, (II) Usage Modeling, (III) Intrusion Modeling, and (IV) Survivability Risk Analysis. Each cycle starts with intrusion modeling in quadrant III, performs survivability risk analysis and planning in quadrants I and IV, and finishes with a revision of the usage model in quadrant II based on the analysis.

The problem-definition stage starts by elaborating the overarching mission of the enterprise context for the system under design. System survivability is defined in terms of maintaining services that are essential to mission fulfillment. We determine the primary adversaries that may compromise the ability to carry out these essential services and elaborate the system's role in ensuring mission survivability. In quadrant II, we express the essential services in terms of transactions or workflows. A system's survivability depends on the workflow interaction of its components as well as the topology of its architecture. We describe an initial high-level architecture and map onto it the workflows associated with services that are essential to achieving the mission.

As mentioned, each of the three spiral cycles starts in quadrant III and ends in quadrant II. Attack trees, developed in quadrant III, elaborate the ways that an attacker could compromise system survivability given the architecture started in quadrant II. We define a system intrusion as a sequence of successful attacks that results in a compromise to system survivability. Paths through the attack tree represent possible intrusions and can be mapped onto the architecture as intruder workflows. Such mappings may provide additional insight during the quadrant IV vulnerability and impact assessments. Risk analysis identifies the potential intrusions that are most likely to cause significant compromise. Quadrant I compares alternative, mechanism-based resolutions to these high-priority intrusions. We identify a particular solution to resist and/or recognize and recover from the intrusions. Quadrant II incorporates this mechanism-based solution into the evolving architecture and refines the essential-service traces accordingly.

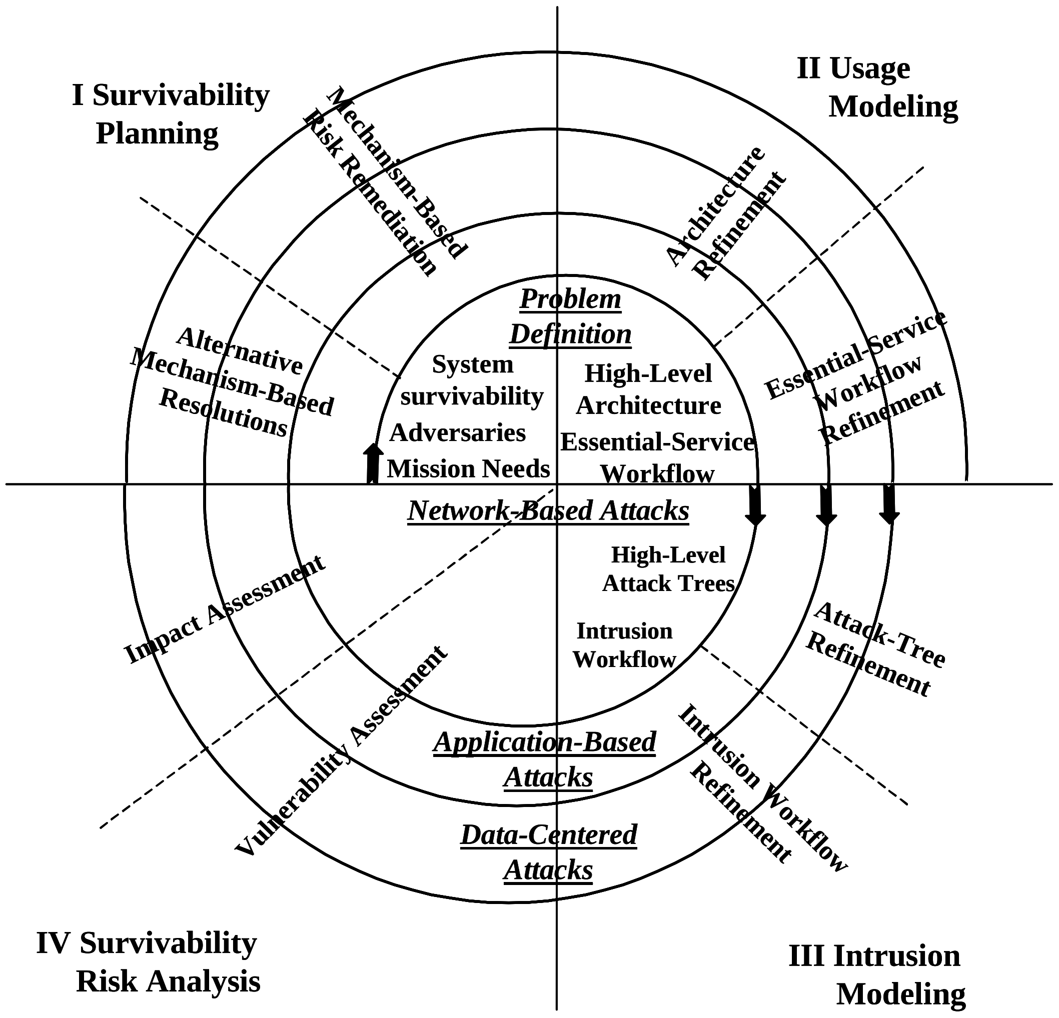

Figure 1: Architecture Refinement Process

This analysis proceeds through each of the three cycles of Figure 1. Each cycle is intrusion aware and results in mechanism-based improvements to the evolving architecture. The first cycle of the spiral, network-based attacks, considers attacks on the communication infrastructure and supporting services. Such attacks typically involve little knowledge of the system architecture or underlying workflows. Mechanisms that counter such attacks are widely applicable and represent best practices in terms of firewalls, virtual private networks, host hardening, and network-intrusion detection. The second cycle of the spiral, application-based attacks, considers attacks on the architecture component applications such as a Web server, e-mail services, or supporting application infrastructure. The components are likely commercial off-the-shelf software (COTS) and the vulnerabilities are vendor specific or caused by errors in operations or administration. The third cycle of the spiral, data-centered attacks, considers attacks that are transaction specific or design specific. The vulnerabilities exploited may be in the underlying workflows, the design of the application, the protocols that support interoperability, or the administrative systems.

As in the original Spiral Model, the completion of one cycle of the spiral is not necessarily followed by the next cycle of the spiral [Boehm 88]. Changes to the architecture that result from one cycle may require the iteration of the tasks of previous cycles. For example, suppose that during cycle 3, we discover a potential data-centered attack over the Internet that uses legal transactions to wage a denial-of-service (DoS) attack. We resist the attack by adding redundant Web servers to handle the increased load, while other mechanisms monitor and respond to the attack. These added components might introduce application-based vulnerabilities to the architecture, requiring a return to cycle 2 of the spiral. This type of iteration is not shown explicitly to simplify the figure's presentation. Nevertheless, the reader should assume that these iterations can take place, as needed, to address new vulnerabilities introduced by the evolving architecture.

We use the notation shown in Figure 2 to refine the architecture for our e-commerce example. This notation extends Use Case Maps (UCMs), which characterize "in a high-level way, how the organizational structure of a complex system and the emergent behavior of the system are intertwined" [Buhr 98]. UCMs provide a notation for mapping essential-service and intruder workflows onto the components of an enterprise's architecture using responsibility paths. Responsibility paths are drawn as continuous lines that wind their way through the architecture indicating the behavioral responsibilities of the components along the way. By hiding low-level architectural details, such as connectors between components, and focusing on responsibility paths, the extended UCM notation allows assessing and improving the survivability of a system architecture.

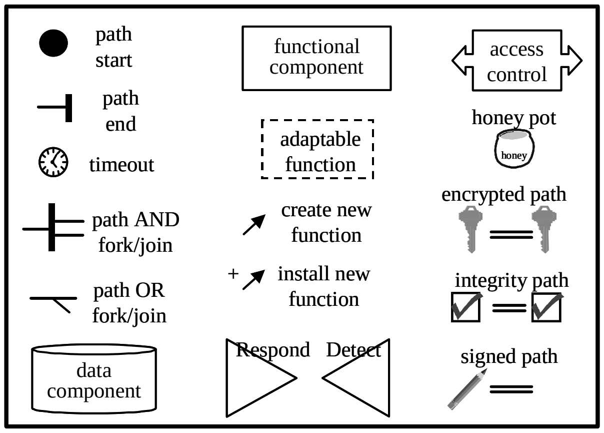

Figure 2: Workflow Architecture Notation

3 An E-Commerce Application

This section summarizes an application of our process to an example in the e-commerce domain. E-commerce systems are typically unbounded in that no stakeholder has full control or visibility into all system nodes. E-commerce workflows include multiple organizations, each of which likely has different survivability requirements and security policies. We assume that there is a demander of some supplier's product which is available for purchase over the Internet. We take the point of view of the supplier who does not control the configuration or security policy of the demander. The supplier can, however, make certain assumptions about, or place certain requirements on, the demander for the purposes of doing business.

We organize the discussion of the e-commerce example according to the spiral structure of Figure 1. There is a subsection for the problem-definition stage and one for each of the spiral cycles. Each subsection is partitioned according to the relevant spiral quadrants: survivability planning, intrusion modeling, survivability risk analysis, and usage modeling. For simplicity and due to space limitations, we limit our discussion to the technical (not procedural) issues associated with one iteration of each spiral cycle. Full-scale application of the process requires considering security procedures that complement the use of the technology and revisiting previous spiral cycles due to vulnerabilities introduced in later cycles.

3.1 Problem Definition

This section describes the e-commerce problem and the high-level supplier/demander workflow.

Survivability Planning

The supplier's overall mission is to make an adequate profit from selling its product. The e-commerce system's role in accomplishing this mission is to facilitate profitable sales of the product. Attacks on the e-commerce system could come from a malicious competitor who desires to disrupt the process or to extract information about the supplier or demander, which could benefit the competitor. Attacks could also come from those who have a political or ethical objection to the company's product or practices. The Internet connection is a main source of the attack. While there certainly are other avenues for attackers to achieve their objectives, we focus on Internet-based attacks to simplify our analysis.

Usage Modeling

Figure 3 depicts a typical e-commerce workflow between the demander and supplier [Hauswirth 01]. The workflow proceeds top to bottom except as specified. After looking at advertisements on the supplier's Web site, the demander makes an offer to purchase the product or accepts the supplier's price. Further negotiation may involve any number of offer and/or counteroffer interactions. Negotiation completes when one party accepts the other's offer, at which point the supplier sends the product and receipt to the demander. For simplicity we do not consider the case where the parties cannot find agreeable terms.

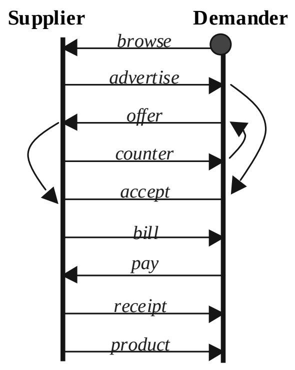

Figure 3: E-Commerce Workflow

For this e-commerce example, the listed transactions represent the public workflow between the supplier and demander. Each organization has its own internal workflow, which it may want to hide from outside surveillance. The overall system architecture (and internal workflow) will support the implementation of the public workflow.

Figure 4 shows a simple architectural model for the workflow described in Figure 3. We split the workflow into supplier and demander portions and expand on the supplier-side architecture. We assume an arbitrary number of demanders, each of which is outside the administrative control of the supplier. Notice that we maintain the top-to-bottom flow of the scenario, starting with the initial Web browsing by the demander. We also use a bidirectional path between the supplier's Web server and back-end data store to represent the data accesses that are required to execute the supplier's responsibilities.

3.2 Network-Based Attacks

This section refines the e-commerce architecture of Figure 4 to protect against network-based attacks.

Intrusion Modeling

Network-based attacks target the network itself or those nodes that are exposed to direct Internet access. DoS attacks often target the network. Network-based attacks also include general scans of the open communications ports. Attackers might compromise a Domain Name Service server and route the transaction through a site under their control to observe or change the content. Other attacks could exploit vulnerabilities with the IP protocol, such as a SYN flood attack, or target the operating system on exposed nodes. Network scans and probes are part of an information-gathering stage for the attacker. The analysis of responses to network probes can identify the version of routers, operating systems, and exposed applications, and such configuration information can support an application-specific attack.

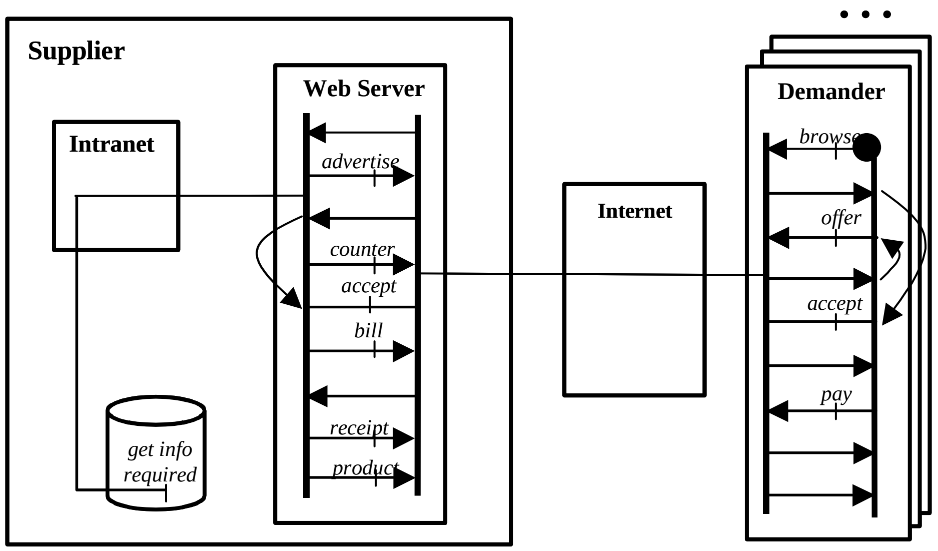

Figure 4: Simple Architectural Model of E-Commerce Workflow

Survivability Risk Analysis

The architecture in Figure 4 exhibits neither the protection of the communication links between the supplier and demander nor the authentication of demander interactions. This architecture is very vulnerable to network-based attacks by attackers with relatively little sophistication. Toolkits exist that can automate a significant part of those attacks which could have a significant, negative impact on the supplier's ability to sell the product.

Survivability Planning

The general strategy is to limit system access and visibility to the attacker. The architecture in Figure 4 hides the private workflows for both the supplier and demander from external view. The Web server and the associated business logic implement the public interface to the transactions and support a limited set of commands. One design possibility is to disallow commands that directly access any of the internal systems. All user input would be reviewed before being submitted for processing by the internal systems.

We also need to limit network-based attackers on those components in Figure 4 that are exposed. Such mechanisms manage the permitted connections, protect the network traffic through encryption, and monitor network traffic. A boundary controller can be used to filter incoming communications so that only selected ports can receive external traffic. Traffic over other, disallowed ports should go no further. Network-intrusion detection can identify scans and network-protocol attacks.

Usage Modeling

Figure 5 extends the architecture to protect the network path between the supplier and demander and to protect the communications ports on the supplier's system. Notice that interoperation requires that the supplier's and demander's sites have symmetric path-protection mechanisms in place. The boundary controller supports network-based intrusion detection. We assume that an analysis shows that three redundant Web servers provide an adequate capacity to handle the set of expected demanders, with enough spare capacity to ensure minimal loss of service in the face of a DoS attack.

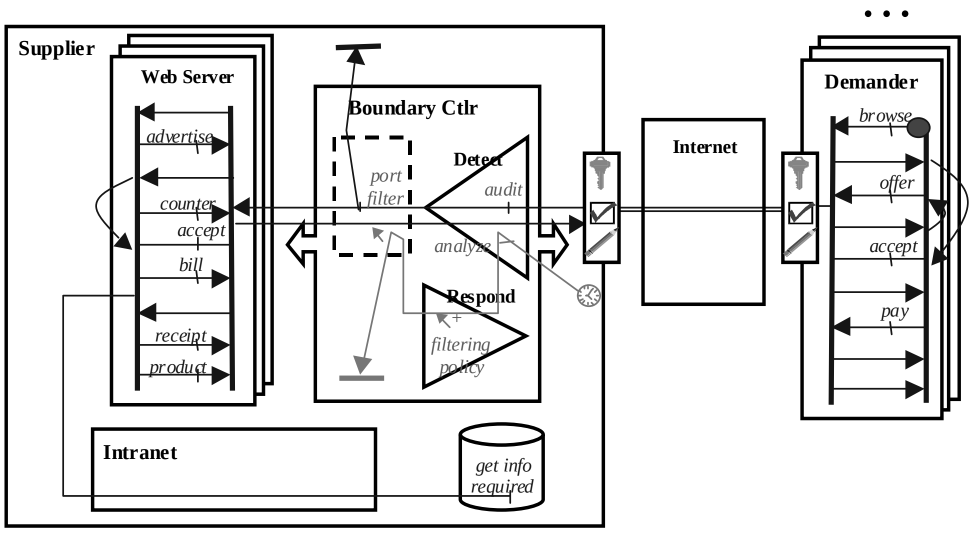

Figure 5: Architecture Extended to Counter Network-Based Attacks

3.3 Application-Based Attacks

This section refines the e-commerce architecture of Figure 5 to protect against application-based attacks.

Intrusion Modeling

Application-specific attacks are those that exploit general vulnerabilities in applications such as a Web or FTP server. For COTS applications, an attack can exploit a vulnerability in how the application is used or configured with the supporting technology such as Active Server Pages, Java Server pages, or Visual Basic scripts. Application-based attacks often exploit a failure in processing user input such as not checking the size of user input (e.g., a buffer overflow vulnerability). Other than applying patches to existing system components, such vulnerabilities are typically beyond the immediate control of the system maintainer.

Web servers are frequent attack targets. The targeted application may not be an essential mission component, but the application vulnerability could give the attacker increased user access that is then used to directly attack mission-critical components.

Survivability Risk Analysis

Application-based attacks are likely to represent the highest risk for the system. Such vulnerabilities are widely known and may be supported by toolkits. The e-commerce system may be an intermediate step rather than the attacker’s real target. An attacker who has been able to exploit a weakness in an application and gain increased access could be in a position to exploit other vulnerabilities in the supplier’s internal systems. While the primary concern with application-specific attacks is the threat to internal systems, a successful attack on an e-commerce subsystem such as the Web server could lead to attacks on other subsystems such as order confirmation.

Survivability Planning

The primary strategy is to maintain good configuration management of all systems so that the operating systems are up-to-date and the latest patches have been applied to applications and system components. Another general strategy is to separate the e-commerce system from the internal supplier’s systems and to separate the systems and functions that implement different workflows where appropriate. Such workflow separation includes implementing the workflows on distinct services and networks so that selected workflows could have their own boundary controllers.

For example, some workflows, such as order confirmation, could be implemented by e-mail or other message-based transport architecture. A high-availability requirement could also use a message-based ordering system as an alternative to the Web-based architecture.

Such a separation has multiple benefits. The capability to recover from an attack or continue service during an attack is an essential survivability property. The separation of services limits the impact of an attack and could allow alternative implementations of an essential service to continue. The recovery of individual services is likely easier than the recovery of an integrated set of services. The separation of ordering and order confirmation also protects auditing functions.

A variety of mechanisms can be applied. It is essential to apply all vendor patches to fix known application vulnerabilities. Hardened versions of the operating system may be available, and, at a minimum, all unneeded operating system functions should be disabled. Host-based intrusion detection can be installed as well as file-system checks to monitor any changes in critical system files.

User access to applications can also be controlled by means of authentication implemented by a portal or Web-server login. In the above architecture, the data interface is a Web server, with access control likely expressed in terms of URLs. The initial authorization could give access to general information, and a specific query for sensitive information could require stronger authentication such as the use of a certificate rather than a simple password.

The strong protection of the path between the supplier and demander and user authentication should require the attacker to pose as a legitimate demander of the supplier's services to get the application access required to exploit a vulnerability. The large number of demander sites that are not controlled or monitored by the supplier give the attacker the opportunity to exploit a vulnerability in one of the demander sites and obtain user access there. If attackers can monitor network traffic on a demander site, they may be in a position to try a replay attack on the supplier site. So additional requirements for the authentication mechanism could be resistance to replay attacks.

Usage Modeling

A standard approach to limit the attacker who has gained access on the server is to install a second firewall between the Web server (including its associated applications) and the supplier's Intranet as shown in Figure 6. This establishes a demilitarized zone (DMZ). The DMZ boundary controller performs port-based access control as before with the goal now to limit access from the Web server to the internal services.

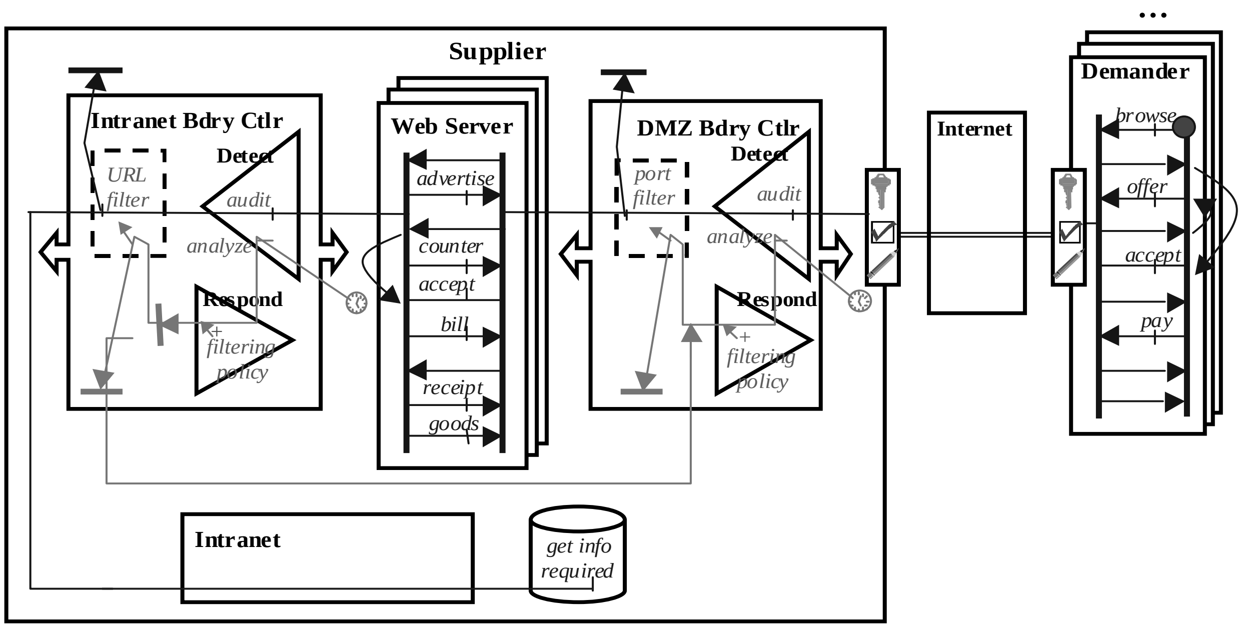

Figure 6: Architecture Extended to Counter Application-Based Attacks

While the DMZ offers protection for the supplier's internal systems, it does not necessarily prevent the attacker from accessing proprietary information maintained by the e-commerce system. If attackers can locate the confidential information, they need to establish a channel back to a machine under their control to export the captured data. This new channel establishes a new port and remote connection that goes unhindered and undetected without filtering and auditing the outbound channel. The modified architecture, therefore, allows for monitoring the outbound channel and adjusting the security policy as needed.

The attacker may also be able to change the content of the Web site or change or exploit the application logic associated with the Web server by reading or changing that logic (which might be implemented as scripts). In addition to a DMZ, the Web server could be replaced by a server proxy and Web server moved inside the DMZ, as shown in Figure 7 on page 16, to limit the impact of direct attacks on Web content or scripts associated with the Web server. All Web requests go first to the server proxy. Those requests that are consistent with the access policy are passed through the boundary controller to an available Web server. The generated Web page is passed back to the server proxy and then back to the user. The server proxy is much simpler and, thus, less vulnerable to attack. Filtering and auditing is performed on both the incoming and outgoing traffic. Suspicious activity at the intranet boundary controller is used to adjust the filtering policy at both the inner and outer boundary controllers. Thus, an attacker who is attempting to get confidential information from a compromised Web server may be caught in the act and prevented from further malicious activity.

3.4 Data-Centered Attacks

This section refines the architecture of Figure 6 to protect against data-centered attacks.

Intrusion Modeling

Data-centered attacks exploit the data flow associated with the transactions. The objective of such attacks could be to extract confidential information, to overload a system with extraneous requests or orders, or to compromise data integrity.

While data-centered attacks include attacks on the encryption of the data stream, the frequency of such attacks is likely to increase as systems support new, more powerful application interaction. The Extended Markup Language (XML) is increasingly used in e-commerce applications. Attackers can exploit vulnerabilities in an XML-formatted data stream that is processed by the supplier's server. Data-centered attacks can also involve active content, which is a data stream that represents a set of executable commands. Examples of active content include ActiveX controls, Java applets, email attachments with macros, and Web pages with Javascript. For e-commerce, an active content data stream could be represented by a remote procedure call as implemented by Simple Object Access Protocol (SOAP).

Survivability Risk Analysis

The extent of vulnerability to data-centered attacks depends on the complexity of the demander's transactions that are permitted and the demander's flexibility in executing those transactions. Generally, the greater the complexity and flexibility, the greater the chances will be of significant harm coming from data-centered attacks. The analysis for data-centered attacks should concentrate on the semantics of the transactions and unanticipated interactions among transactions.

Our e-commerce example is not sufficiently complex to allow a very interesting vulnerability analysis for data-centered attacks. However, the fact that the DMZ is based only on port and URL filtering, suggests that it may be fairly easy for attackers to pretend to be other known demanders of the supplier's product. In addition, a review of the supplies ordered could provide information to the demander (or a malicious interceptor) about the supplier's product design or manufacturing process. Insiders at one demander site could use access to obtain information that would be valuable if they were hired by a competitor. System recovery for data-centered attacks that compromise data integrity is costly for demander and supplier sites. The recovery time and requirements is a form of denial of service.

Survivability Planning

Possible architectural refinements to counter data-centered attacks are associated with the application logic. For email, content scan is likely to occur at multiple points in the architecture. Mail could be scanned at the mail gateway as a component of the boundary controller. Scans could also occur on the mail server and on the user-mail client. A similar review of content could apply to other data streams. Other refinements could include finer data-access control, the periodic analysis of the database transactions in search of malicious activity, and the redesign of workflow and recovery techniques to automate undoing a transaction across multiple systems.

Usage Modeling

Figure 7 shows refinements to counter data-centered attacks. In addition to the use of the DMZ structure described in the last section, the new architecture inserts a mechanism for data-access control. The architecture uses redundant back-end databases to permit backup and integrity checks of demander and supplier information. Auditing database transactions permits their periodic analysis in search of malicious activity. Responses to malicious activity include modifying the access-control policy to constrain or lock out malicious users and corrections to data that has been corrupted. Misinformation, in the form of a honey pot, is embedded in the database to throw off attackers and to facilitate the detection of malicious activity. Notice that such defenses are invaluable against insider attacks as well.

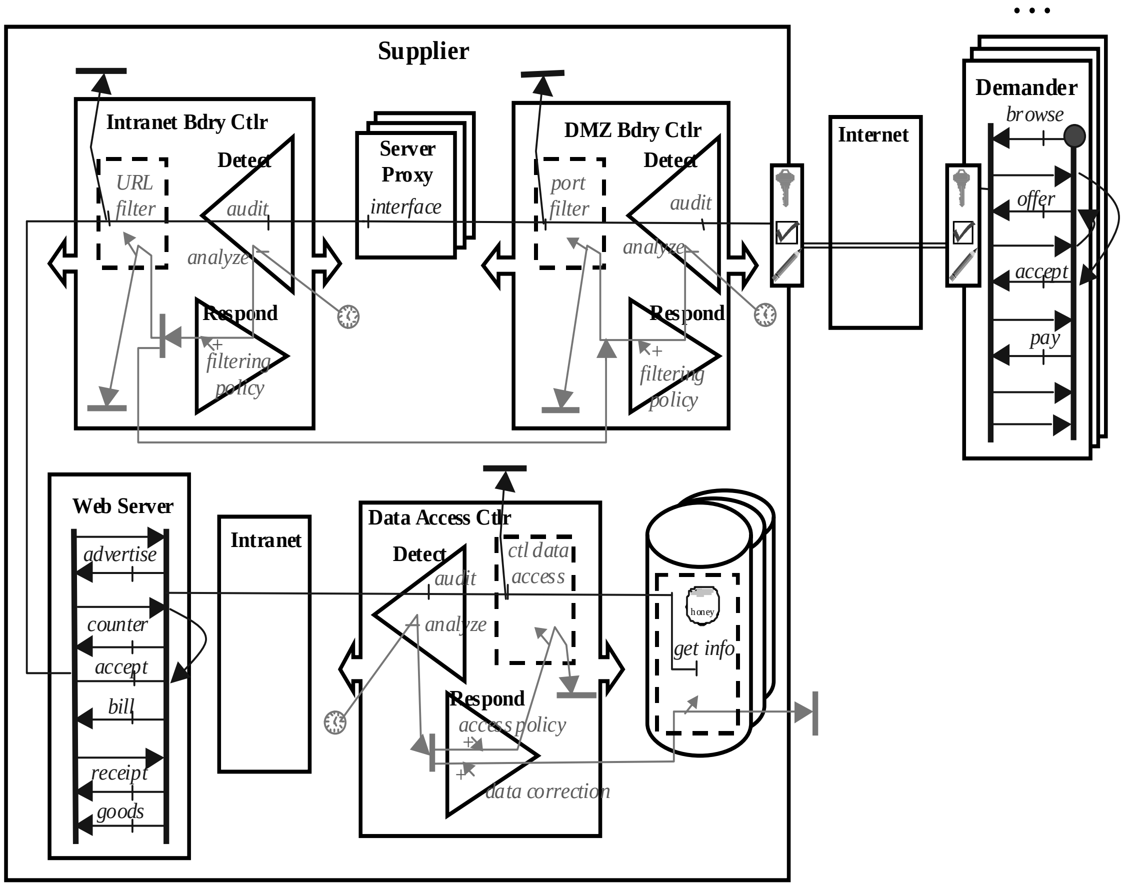

Figure 7: Architecture Extended to Counter Data-Centered Attacks

3.5 Discussion

The above refinement made many simplifications to control the complexity of the example that may have to be considered in real-world situations. Realistic e-commerce systems will have to manage more complex interaction among components. For example, while the architecture examples concentrated on mechanisms that are associated with a system implemented with an interactive Web interface, e-commerce workflows could have a server-to-server interface where demander requests are generated automatically by their manufacturing or inventory system.

There could also be attacks on the administrative infrastructure. A complex e-commerce system could use a directory service that is accessed by the business logic to manage user authentication and authorization information and security policies. The directory server is a central repository for such security information and hence a likely target of attacks. In addition, the demander and supplier have to exchange information about authorized users and approved access. The administrative workflow could delegate the responsibility for managing user information to demander sites using an import/export mechanism for sharing directory information among sites. The attacks could also exploit the administrative procedures, and those components should have their own set of interfaces and boundary controllers. Finally if system administrators for a supplier site have access to demander information, social-engineering-style attacks could target such administrators to gain access to sensitive information. For example, one organization identified 30 users who required access to sensitive data, but a pool of over 100 system administrators had privileged access to the same data as part of routine administration. Part of the architecture design had to limit administrative access to systems.

4 Conclusion

This paper describes a process for systematically refining an enterprise system architecture to resist, recognize, and recover from deliberate, malicious attacks by applying reusable design mechanisms that help ensure the survival of the enterprise mission. We structure the process like Boehm's Spiral Model, since that model reflects the iterative, risk-driven process needed to refine survivable architectures. The spiral structure includes a problem-definition stage in the center followed by three cycles of the spiral, each covering a different type of attack that needs to be considered – network-based attacks, application-based attacks, and data-centered attacks. The spiral proceeds through four quadrants – Survivability Planning, Usage Modeling, Intrusion Modeling, and Survivability Risk Analysis. The mechanism-based, intrusion-aware defined process addresses the development of large-scale, highly distributed, internetworked, and potentially unbounded systems.

This work needs to progress along a number of lines to fully support survivable architectural refinement. Although many enterprise intrusions involve social engineering and physical attacks, our current process description focuses almost exclusively on technological attacks and countermeasures. The process needs to be expanded to counter the full range of attacks that may compromise an enterprise's survivability. Our notation needs to include a more comprehensive set of survivability mechanisms, both operational and technical, such as Anderson describes [Anderson 99]. We need to identify structured methods for vulnerability/impact assessment and for making survivability recommendations based on these assessments. Finally, we need to validate the process through its applications in other domains, while demonstrating how it can be integrated effectively into existing corporate development processes.

References

[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.
[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. Available WWW: <URL: http://www.sei.cmu.edu/publications/documents/00.reports/00tn017.html> (2000).
[Boehm 88]	Boehm, B. “A Spiral Model of Software Development and Enhancement.” *IEEE Communications 21*, 5 (May 1988): 61-72.
[Buhr 98]	Buhr, R. J. A. “Use Case Maps as Architectural Entities for Complex Systems.” *IEEE Transactions on Software Engineering 24*, 12 (December 1998): 1131-1155.
[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. Available WWW: <URL: http://www.sei.cmu.edu/publications/documents/97.reports/97tr013/97tr013abstract.html> (1999).
[Hauswirth 01]	Hauswirth, M.; Jazayeri, M.; & Schneider, M. “A Phase Model for E-Commerce Business Models and Its Application to Security Assessment.” *Proceedings of the 34th Hawaii International Conference on Systems Sciences*. Maui, HI, January 3-6, 2001. Los Alamitos, CA: IEEE Computer Society 2001.

[Marmor-Squires 88] Marmor-Squires, A. B. & Rougeau, P. A. “Issues in Process Models and Integrated Environments for Trusted Systems Development,” 109-113. *Proceedings of the 11th National Computer Security Conference*, October 17-20, 1988. Washington, DC: U.S. Government Printing Office, 1988.

[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.

[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. Available WWW: <URL: 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. Available WWW: <URL: http://www.sei.cmu.edu/publications/documents/01.reports/01tn001.html> (2001).

[Röhm 99] Röhm, A.W.; Herrmann, G.; & Pernul, G. “A Language for Modeling Secure Business Transactions,” 22-31. *Proceedings of the IEEE Annual Computer Security Applications Conference (ACSAC’99).* Phoenix, Arizona, December 6-10**,** 1999. Los Alamitos, CA: IEEE Computer Society, 1999.

[Salter 98] Salter, C.; Saydjari, O.; Schneier, B.; & Walner, J. “Toward a Secure System Engineering Methodology,” 2-10. *Proceedings of the New Security Paradigms Workshop*. Charlottesville, VA, September 22-25, 1998. New York, NY: ACM, 1999.

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

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE October 2001	3. REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE Architectural Refinement for the Design of Survivable Systems		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-2001-TN-008
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
11. SUPPLEMENTARY NOTES		
12A DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS		12B DISTRIBUTION CODE

13. ABSTRACT (MAXIMUM 200 WORDS)

This paper describes a process for systematically refining an enterprise system architecture to resist, recognize, and recover from deliberate, malicious attacks by applying reusable design primitives that help ensure the survival of the enterprise mission. Systems of interest may be unbounded; that is, have no central administration and no unified security policy. The survivable architecture refinement is an iterative risk-driven process which adopts the structure of Boehm's Spiral Model [Boehm 88]. The cycles of the spiral structure represent different types of attack that need to be considered – network-based attacks, application-based attacks, and data-content attacks. We illustrate our survivable architecture refinement process through its application to e-commerce. E-commerce examples are representative of the lack of full control and visibility that characterize unbounded systems.

14. SUBJECT TERMS Security architecture, network attack analysis, survivable systems			15. NUMBER OF PAGES 28
16. PRICE CODE			
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL