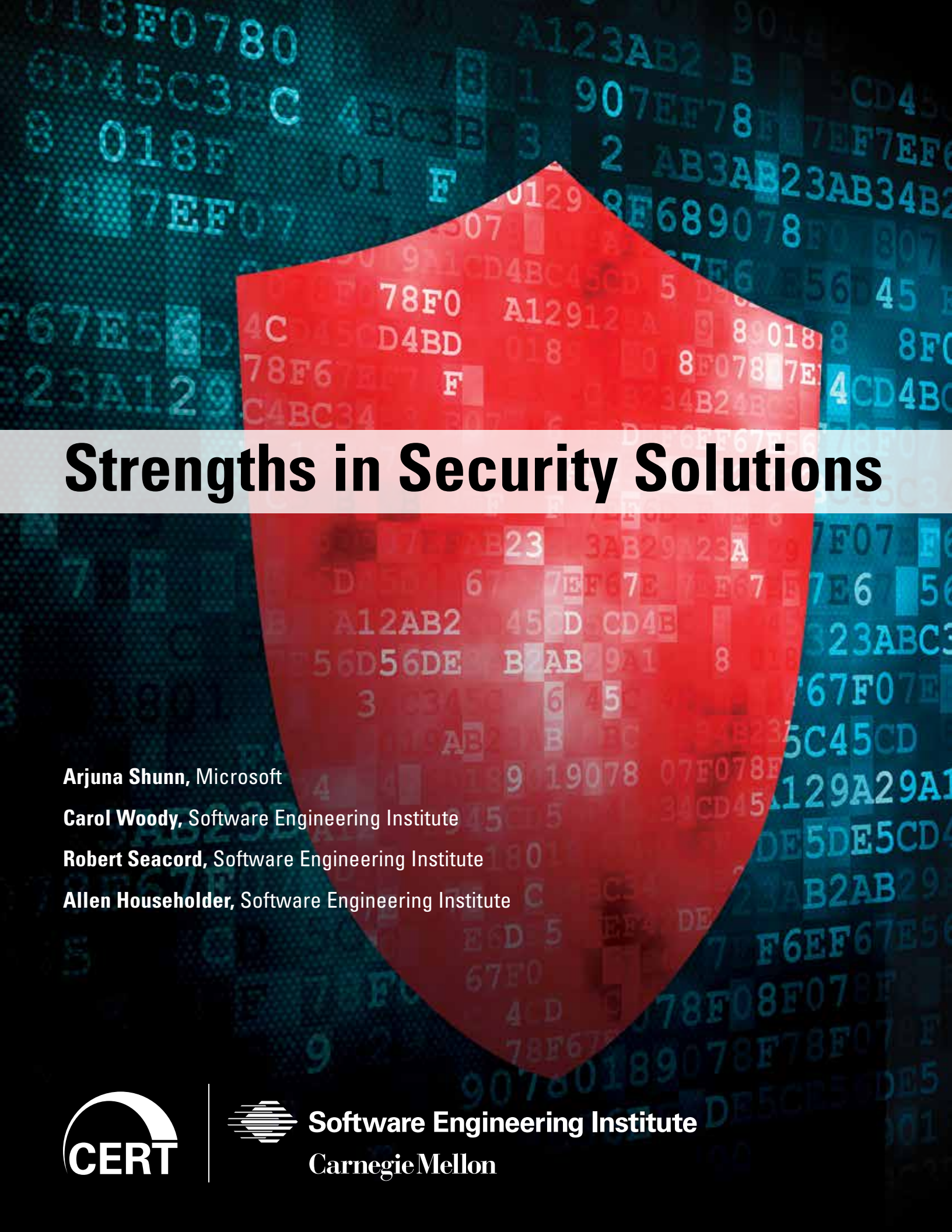


Strengths in Security Solutions

Arjuna Shunn, Microsoft

Carol Woody, Software Engineering Institute

Robert Seacord, Software Engineering Institute

Allen Householder, Software Engineering Institute

CERT

Software Engineering Institute

Carnegie Mellon

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM-0000394

Introduction

The Software Engineering Institute and the CERT® Division

The Carnegie Mellon Software Engineering Institute (SEI) works closely with defense and government organizations, industry, and academia to continually improve software-intensive systems. Our core purpose is to help organizations improve their software engineering capabilities and to develop or acquire the right software, defect free, within budget and on time, every time. The CERT1[â] Division of the Software Engineering Institute (SEI) works to anticipate and solve our nation's cybersecurity challenges. Our cybersecurity work spans the acquisition and development lifecycle and includes research, development, deployment of methods and tools, and operational support. Our main customers include the military services, defense and other federal agencies, state and local government agencies, commercial entities, and academia. As a federally funded research and development center, the SEI is primarily funded by the government and may not compete with private industry.

Our research has confirmed that **decisions made in the acquisition and development of new software and software-based systems have a major impact on operational security.** The challenge begins with properly stating software requirements and ensuring they clearly and practically define security. This is fundamental to the development and fielding of effectively secure systems. When these systems must interoperate with other systems built at a different time and with varying degrees of security, effective operational security becomes much more complex. Software and systems acquired, designed, and developed with operational security in mind are more resistant to both intentional attack and unintentional failures. The goal is to build and acquire better, minimally defective software and systems that can

- possess, through testing and analysis, some measurable level of assurance of minimal vulnerabilities
- operate correctly in the presence of most attacks by either resisting the exploitation of weaknesses in the software or tolerating the failures that result from such exploits
- recognize an attack and respond with expected behaviors that support resistance and recovery
- limit the damage from failures caused by attack or unanticipated faults and events and recover as quickly as possible

Managing complexity and ensuring survivability requires engineering methods based on solid foundations and the realities of current and emerging systems. A great deal of security response is reactive—addressing security issues in response to an attack. A more effective approach is to reduce the potential of such attacks by removing the vulnerabilities that allow a compromise in the first place. Our efforts to address issues before they become a security problem focus on the following key areas:

- **Secure Coding** addresses tools, techniques, and standards that software developers and software development organizations require to eliminate vulnerabilities resulting from coding errors before software is deployed.
- **Vulnerability Analysis** reduces the security risks posed by software vulnerabilities by addressing both the number of vulnerabilities in software that is being developed and the number of vulnerabilities in software that is already deployed. Our vulnerability analysis work is divided into two areas. Identifying and reducing the number of new vulnerabilities before the software is deployed is the focus of our vulnerability discovery effort, while our vulnerability remediation work deals with existing vulnerabilities in deployed software. We regularly comment on issues of importance to the vulnerability analysis and security community through the CERT/CC Blog.
- **Cyber Security Engineering** addresses research needed to prepare acquirers, managers, developers, and operators of large-scale, complex, networked systems to address security, survivability, and software assurance throughout the design and acquisition lifecycles. This research encompasses four areas: Software Assurance, Security Requirements, Software Supply Chain Risk Management (SCRM), and Software Risk Management. Because much of the DoD software is vendor developed, the research addresses both internal development and acquired software sources.

The recommended practices and processes found in this document are not and can never be complete because the CERT Division is continually researching better methods, practices, and tools to improve software engineering and software assurance. This document highlights a sample of results with readily apparent connections to the Security Development Lifecycle (SDL) but there are many other results from the SEI and the CERT Division that should also be considered.

Microsoft Software Development Lifecycle

In the book *Writing Secure Code, Second Edition* (Microsoft Press, 2003), Michael Howard and David LeBlanc describe a set of process improvements and practices for improving the security of software. They also outlined one of the first practical approaches for threat modeling: the process of systematically thinking through threats to an application. In 2005, Steve Lipner and Michael Howard then publicly described a security development process fit for widespread use that was based on Microsoft's internal process, the Security Development Lifecycle (SDL).

Use of the SDL has been mandatory development policy at Microsoft since 2004. The SDL was designed to reduce the number and severity of vulnerabilities for enterprise-scale software development. It is specifically tailored to Microsoft development practices and business drivers.

A longitudinal study performed by Dan Kaminsky (Kaminsky, 2011) used fuzzing to identify exploitable or probably exploitable vulnerabilities in Microsoft Office and OpenOffice. The results suggest that the SDL has helped Microsoft improve software security. Figure 1 shows that the number of exploitable or probably exploitable vulnerabilities in Microsoft Office decreased from 126 in Microsoft Office 2003 down to only 7 in Microsoft Office 2010. CERT researcher Will Dorman performed a follow-on study, which is described on the CERT/CC Blog at http://www.cert.org/blogs/certcc/2011/04/office_shootout_microsoft_offi.html.

Although evidence exists that the SDL has helped improve software security, it was designed to meet the needs of Microsoft rather than the broader software development community. To address this community need, Microsoft published the *Simplified Implementation of the Microsoft SDL* (Microsoft, 2010) which is based on the SDL process used at Microsoft but reduces the SDL to a more manageable size. The SDL is licensed under a non-proprietary Creative Commons License. It is platform and technology agnostic and suitable for development organizations of any size using any software development processes, regardless of whether they are based on a waterfall, spiral, agile, or other model.

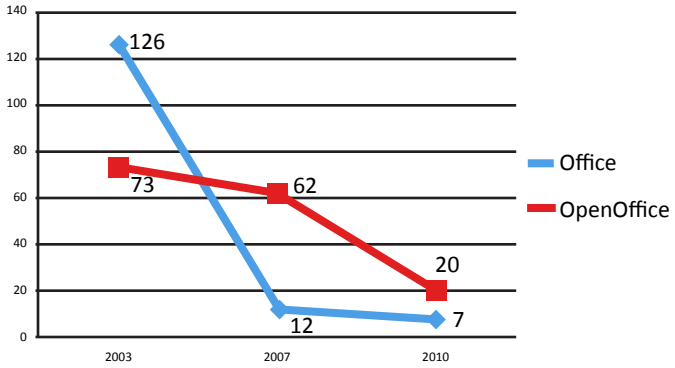


Figure 1: Vulnerabilities in Office versus OpenOffice, 2003-2010.

Microsoft's Simplified SDL outlines a set of security activities to be performed at each stage of the software development lifecycle. These activities begin at inception by gathering security requirements and then move through design, threat modeling, source code scanning, training, security testing, and secure deployment. Figure 2 shows a graphical representation of Microsoft's Simplified SDL.

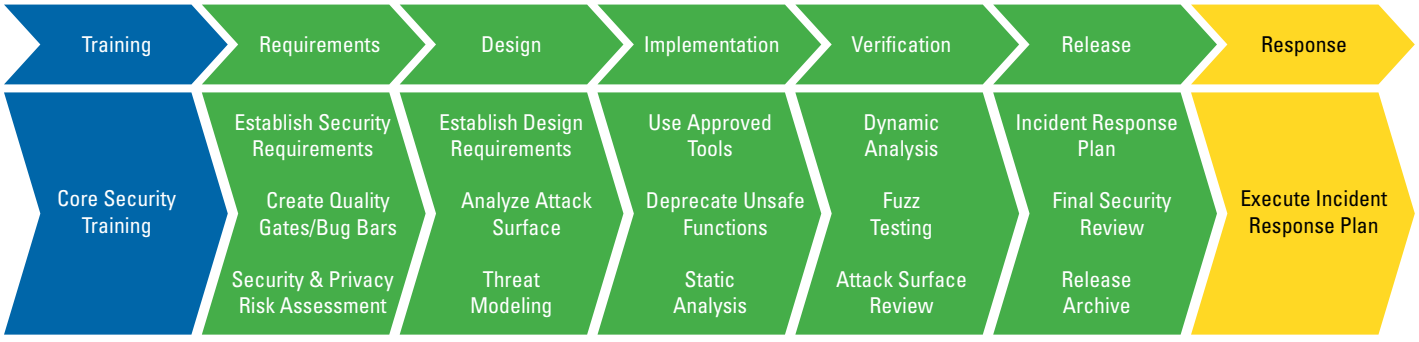


Figure 2: An outline of Microsoft's Simplified Security Development Lifecycle. The diagram shows the key activities of the SDL at each stage of the software development lifecycle.

A detailed discussion of Microsoft's Simplified SDL can be found at www.microsoft.com/sdl.

The Simplified SDL is supported by a limited number of training modules, processes, and tools. However, Microsoft and the CERT Division are working to enhance SDL usability by linking activities from the SDL to CERT-developed solutions. This publication provides a cross reference of current CERT capabilities to key areas of the SDL. All activities identified in the SDL can be supported with CERT-developed solutions. The following sections describe each CERT-developed solution with cross-references to activities in the SDL where each should be used. Pointers to additional information to facilitate adoption are also provided.

ISO/IEC 27034 (Application Security) is designed to help organizations build security throughout the lifecycle of applications. ISO/IEC 27034 is a risk-based, management-supported, and continuously improving software security management system applied across the application lifecycle. The SDL is an ISO/IEC 27034-conformant process, which the CERT tools, processes, and practices can support.

CERT Solutions

Secure Coding Standards

Description

An essential element of secure coding is well-documented and enforceable coding standards. Coding standards encourage programmers to follow a uniform set of rules and guidelines determined by the requirements of the project and organization rather than by the programmer's familiarity or preference.

The CERT Division coordinates the development of secure coding standards by security researchers, language experts, and software developers using a wiki-based community process. More than 1,200 contributors and reviewers have participated in the development of secure coding standards on the CERT Secure Coding Standards wiki. Companies such as Cisco and Oracle have adopted the CERT Division's secure coding standards.

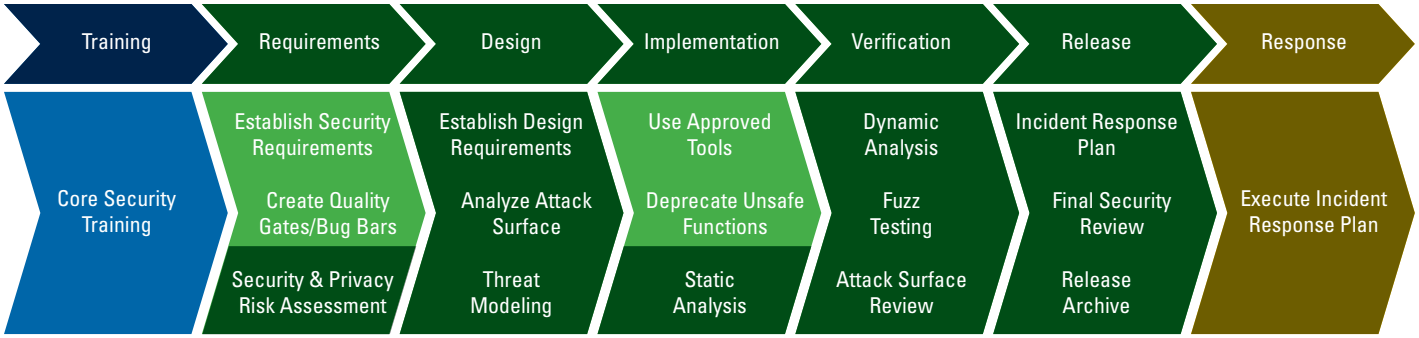
Secure coding standards define a set of rules and recommendations against which source code can be evaluated for conformance. They provide a metric for evaluating and contrasting software security, safety, reliability, and related properties. To date, the CERT Division has released secure coding standards for C and Java and is readying a standard for C++ and Perl.

- CERT C Secure Coding Standard
  - Version 1.0 (C99) published in 2009
  - Version 2.0 (C11) under development
  - ISO/IEC TS 17961 C Secure Coding Rules DRAFT Technical Specification
  - Conformance Test Suite
- CERT C++ Secure Coding Standard
  - Version 1.0 (C++11) under development
- CERT Oracle Secure Coding Standard for Java
  - Version 1.0 (Java 7) published in 2011
  - Java Coding Guidelines
- CERT Perl Secure Coding Standard
  - versions 5.12 and later of the Perl programming language

Value to Security

These standards can be used as a metric to evaluate source code (using manual or automated processes). Faithful application of secure coding standards can eliminate the introduction of known source-code-related vulnerabilities.

Connection to Microsoft SDL



Core Security Training

- Secure Coding in C and C++ http://www.sei.cmu.edu/training/P63.cfm

Establish Security Requirements

Create Quality Gates/Bug Bars

Use Approved Tools: tool selection provides the capability for enforcement of secure coding standards

Deprecate Unsafe Functions: secure coding standards define a list of appropriate functions for safer results

Resources for Further Solution Details

1. Software Engineering Institute. *Secure Coding Standards.* https://www.securecoding.cert.org (2012).
2. Long, Fred. *The CERT Oracle Secure Coding Standard for Java.* https://www.securecoding.cert.org/confluence/display/java/The+CERT+Oracle+Secure+Coding+Standard+for+Java (2012).
3. Seacord, Robert C. *The CERT C Secure Coding Standard.* Addison-Wesley Professional, 2008 (ISBN: 0-321-56321-2).
4. Fred Long, Dhruv Mohindra, Robert C. Seacord , Dean F. Sutherland , David Svoboda. *Java Coding Guidelines: 75 Recommendations for Reliable and Secure Programs.* September 23, 2013 ISBN-10: 032193315X

Source Code Analysis Laboratory (SCALe)

Description

The CERT Secure Coding Standards (described in section 9.6) define a set of secure coding rules. These rules are used to eliminate coding errors that have resulted in vulnerabilities for commonly used software development languages as well as other undefined behaviors that may also prove exploitable. The Source Code Analysis Laboratory (SCALe) can be used to test software applications for conformance to *The CERT C Secure Coding Standard* [4]. Although this version of the standard was developed for C99, most of these rules can be applied to programs written in other versions of the C programming language or in C++. Programs written in these programming languages may conform to this standard, but they may be deficient in other ways that are not evaluated by SCALe.

SCALe analyzes a developer's source code and provides a detailed report of findings to guide the code's repair. After the developer has addressed these findings and the SCALe team determines that the product version conforms to the standard, CERT issues the developer a certificate and lists the system in a registry of conforming systems. As a result, SCALe can be used as a measure of the security of software systems.

SCALe evaluates client source code using multiple analyzers, including static analysis tools, dynamic analysis tools, and fuzzing. CERT reports any rule from secure coding standards to the client. The client may then repair and resubmit the software for reevaluation. Once the reevaluation process is completed, CERT provides the client a report detailing the software's conformance or nonconformance to each secure coding rule.

Multiple analysis tools are used in SCALe because analyzers tend to have non-overlapping capabilities. For example, some tools might excel at finding memory-related defects (memory leaks, use-after-free, null-pointer dereference), while others may be better at finding other types of defects (uncaught exceptions, concurrency). Even when looking for the same type of defect (detecting overruns of fixed-sized, stack-allocated arrays, for example), different analysis tools will find different instances of the defect.

SCALe uses both commercial static analysis tools such as Coverity Prevent, LDRA, and PCLint and open source tools such as Compass/ROSE. CERT has developed checkers to diagnose violations of *The CERT Secure Coding Standards* in C and C++ for the Compass/ROSE tool,[2] developed at Lawrence Livermore National Laboratory. These checkers are available on SourceForge.[3]

SCALe does not test for unknown code-related vulnerabilities, high-level design and architectural flaws, the code's operational environment, or the code's portability. Conformance testing is performed for a particular set of software, translated by a particular implementation, and executed in a particular execution environment [ISO/IEC 2005].

Successful conformance testing of a software system indicates that the SCALe analysis did not detect violations of rules defined by a CERT secure coding standard. Successful conformance testing does not provide any guarantees that these rules are not violated or that the software is entirely and permanently secure. Conforming software systems can be insecure, for example, if they implement an insecure design or architecture.

Software that conforms to a secure coding standard is likely to be more secure than nonconforming or untested software systems. However, no study has yet been performed to prove or disprove this claim.

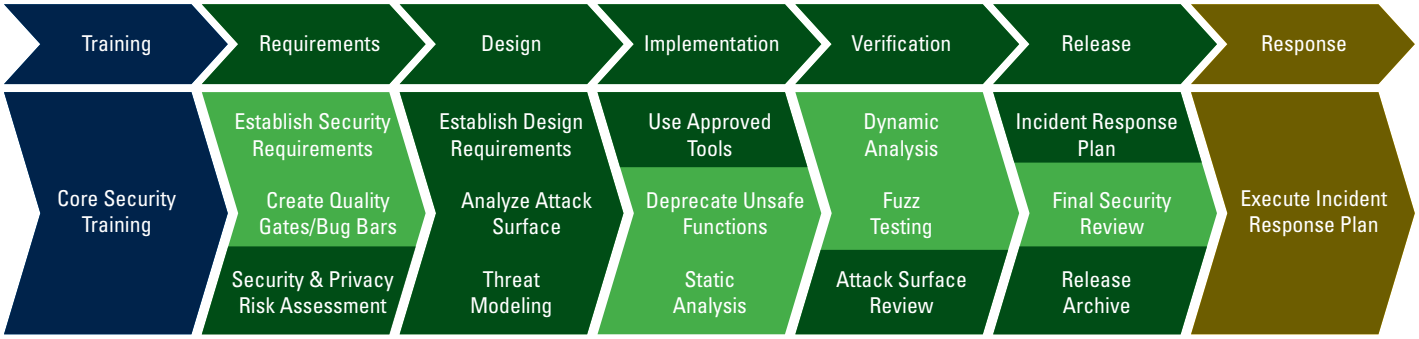
Value to Security

Faithful application of secure coding standards can eliminate the introduction of known source-code-related vulnerabilities. These secure coding standards establish the requirements to which code is developed. However, errors can be made in implementing code to these standards resulting in violations of secure coding rules and potential vulnerabilities. Consequently, it is necessary to verify conformance of the implementation to the standards. SCALe provides a capability to validate conformance to a coding standard to ensure the value of coding standards is realized.

2 http://www.rosecompiler.org/compass.pdf

3 http://rosecheckers.sourceforge.net/

Connection to Microsoft SDL



Establish Security Requirements

Create Quality Gates/Bug Bars

Deprecate Unsafe functions

Static Analysis

Dynamic Analysis

Fuzz Testing

Final Security Review

Resources for Further Solution Details

1. Seacord, Robert; Dormann, Will; McCurley, James; Miller, Philip; Stoddard, Robert; Svoboda, David; & Welch, Jefferson. *Source Code Analysis Laboratory (SCALe)* (CMU/SEI-2012-TN-013). Software Engineering Institute, Carnegie Mellon University, 2012. http://www.sei.cmu.edu/library/abstracts/reports/12tn013.cfm

Vulnerability Discovery and Fuzz Testing

Description

During the process of producing software products, engineers unintentionally create vulnerabilities that are later discovered and mitigated. The CERT Vulnerability Discovery team operates on the principle that by paying greater attention to the early phases of the development lifecycle, we can change the nature of the engineering process to detect and eliminate—and later avoid—vulnerabilities before products ship. We plan to achieve this goal by placing knowledge, techniques, and tools in the hands of engineers to help them understand how vulnerabilities are created and discovered so that they can learn to avoid them.

Fuzz testing is a dynamic software testing technique that can be used to find bugs that result in the crashing of an application. Every bug that results in a crash has the potential of being a vulnerability. Depending on the specific circumstances of a crash, these bugs may also result in vulnerabilities that allow an attacker to execute arbitrary code. Fuzz testing can be used as one measure of the number of vulnerabilities that an application may contain.

Furthermore, by testing what the software actually does, as opposed to what it is expected to do, fuzz testing can often demonstrate integration-level issues that may not be discovered during unit, functional, or integration testing.

The CERT Vulnerability Discovery Team has developed several projects in this area:

- The CERT Basic Fuzzing Framework (BFF) is a mutational file fuzz testing tool that consists of a Linux virtual machine, the zzuf fuzzer, and associated scripts. A version of the BFF that runs natively on Mac OS X is also available.
- The CERT Failure Observation Engine (FOE) is a mutational file-based fuzz testing tool for finding defects in applications that run on the Windows platform.
- The CERT Triage Tools consist of a triage script and a GNU Debugger (GDB) extension named 'exploitable' that classify Linux application defects by severity. We have developed the CERT Triage Tools to assist software vendors and analysts in identifying the impact of defects discovered through techniques such as fuzz testing.

The CERT Basic Fuzzing Framework (BFF) is a software testing tool that finds defects in applications that run on the Linux and Mac OS X platforms. The CERT Failure Observation Engine (FOE) does the same for applications that run on the Windows platform. BFF and FOE perform mutational fuzzing on software that consumes file input. (Mutational fuzzing is the act of taking well-formed input data and corrupting it in various ways, looking for cases that cause crashes.) BFF and FOE automatically collect test cases that cause software to crash in unique ways, as well as debugging information associated with the crashes. The goal of BFF and FOE is to minimize the effort required for software vendors and security researchers to efficiently discover and analyze security vulnerabilities found via fuzzing.

Traditionally fuzzing has been very effective at finding security vulnerabilities, but because of its inherently probabilistic nature results can be highly dependent on the initial configuration of the fuzzing system. BFF and FOE apply machine learning and evolutionary computing techniques to minimize the amount of manual configuration required to initiate and complete an effective fuzzing campaign. BFF and FOE adjust their configuration parameters based on what they find (or do not find) over the course of a fuzzing campaign. By doing so they can dramatically increase both the efficacy and efficiency of the campaign. As a result, expert knowledge is not required to configure an effective fuzz campaign, and novices and experts alike can start finding and analyzing vulnerabilities very quickly.

Some of the specific features BFF and FOE offer are:

- Minimal initial configuration is required to start a fuzzing campaign
- Minimal supervision of the fuzzing campaign is required, as BFF and FOE can automatically recover from many common problems that can interrupt fuzzing campaigns
- Uniqueness determination through intelligent backtrace analysis
- Automated test case minimization reduces the effort required to analyze results by distilling the test case to the minimal changes to the input data required to induce a specific crash
- Online machine learning applied to fuzzing parameter and input file selection to improve the efficacy of the campaign
- Distributed fuzzing support
- Crash severity / exploitability triage

A common complaint is that fuzz testing produces can produce too many results, many of which are not security vulnerabilities. In 2009, Microsoft released a set of security extensions for the Windows debugger, including a command named !exploitable, that provides automated crash analysis and security risk assessment for software that runs on the Windows platform. Subsequently, Apple released a tool called CrashWrangler (Apple Developer Connection account required) to perform similar analysis on crash logs for software that runs on the Mac OS X platform. In the course of our vulnerability discovery work in developing the CERT Basic Fuzzing Framework, we noted the lack of such a tool for software that runs on the Linux platform.

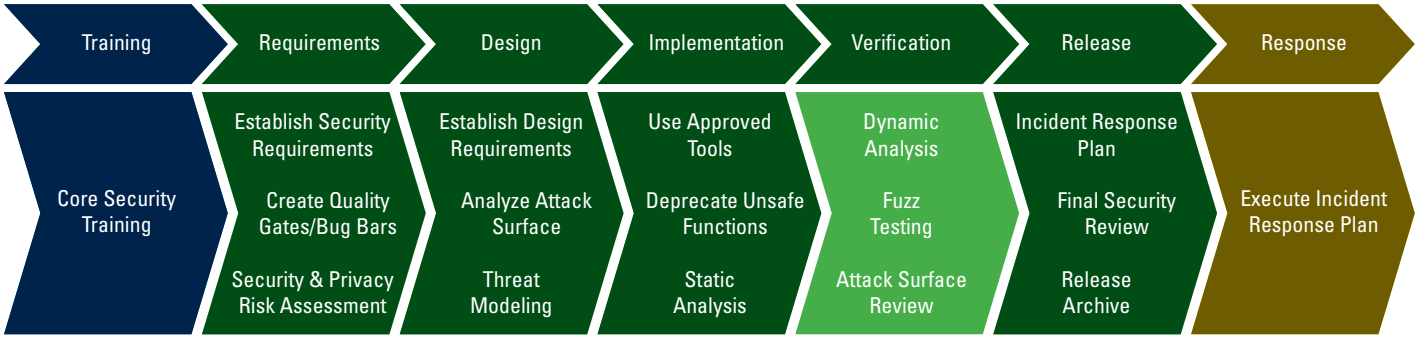
The CERT Triage Tools can be used to assist software vendors and analysts in identifying the impact of defects discovered through techniques such as fuzz testing and prioritizing their remediation in the software development process. The CERT Triage Tools include a GNU Debugger (GDB) extension called "exploitable" that classifies Linux application bugs by severity and a wrapper script for batch execution.

Value to Security

The CERT Basic Fuzzing Framework and Failure Observation Engine provide a freely available, open source solution for organizations wishing to integrate dynamic analysis and fuzz testing into their development or acquisition processes.

The CERT Division has already used BFF and FOE infrastructure to find a number of critical vulnerabilities in products such as Adobe Reader and Flash Player; Foxit Reader; Apple QuickTime, Preview, and Mac OS X; Xpdf; Poppler; FFmpeg; JasPer; Wireshark; VMware VMnc video codec; the Indeo video codec; Shockwave player; Microsoft Office and Windows; Google Chrome; Oracle Outside In; Autonomy Keyview IDOL; and many others.

Connection to Microsoft SDL



Dynamic Analysis

Fuzz Testing

Attack Surface Review

Resources for Further Solution Details

1. Software Engineering Institute. *CERT Basic Fuzzing Framework (BFF).* http://www.cert.org/vuls/discovery/bff.html (2012).
2. Software Engineering Institute. *CERT Failure Observation Engine (FOE).* http://www.cert.org/vuls/discovery/foe.html (2012).
3. Software Engineering Institute. *CERT Triage Tools.* http://www.cert.org/vuls/discovery/triage.html (2012).
4. Microsoft. "!exploitable Crash Analyzer – MSEC Debugger Extensions." *CodePlex.* http://msecdbg.codeplex.com/ (2013).
5. Apple. *CrashWrangler.* https://connect.apple.com/cgi-bin/WebObjects/MemberSite.woa/wa/getSoftware?bundleID=20390 (2013).

SQUARE Method for Security Requirements and Tool

Description

Requirements problems are the primary reason that projects

- are significantly over budget and past schedule
- have significantly reduced scope
- deliver poor-quality applications that are little used once delivered, or are canceled altogether

One source of these problems is poorly expressed or analyzed quality requirements, such as security and privacy. Requirements engineering defects cost 10 to 200 times more to correct during implementation than if they are detected during requirements development. Moreover, it is difficult and expensive to significantly improve the security of an application after it is in its operational environment.

Major software vendors such as Microsoft are now addressing security requirements early, as are other leading companies. Vulnerabilities in mission-critical systems can lead to compromised systems, failures, and possible loss of life. While security needs to be addressed at every lifecycle phase, there are clear benefits to addressing it at the earliest possible stage, before architectural decisions have been made that may preclude optimal security solutions. Acquirers of developed software need to feel confident that security has been addressed early. Likewise, acquirers of COTS software need to factor security requirements into their purchasing decisions.

The Security Quality Requirements Engineering (SQUARE) methodology consists of nine steps that generate a final deliverable of categorized and prioritized security requirements. Although SQUARE could likely be generalized to any large-scale design project, it was designed for use in software systems. The SQUARE process is most effective and accurate when it is conducted by a team of requirements engineers with security expertise and the stakeholders of an IT project. It begins with the requirements engineering team and project stakeholders agreeing on technical definitions that serve as a baseline for all future communication. Next, assets are identified and business and security goals are outlined. Third, artifacts and documentation are created, which are necessary for a full understanding of the relevant system. A structured risk assessment determines the likelihood and impact of possible threats to the system.

Following this work, the requirements engineering team determines the best method for eliciting initial security requirements from stakeholders. This determination depends on several factors, including the stakeholders involved, the expertise of the requirements engineering team, and the size and complexity of the project. Once a method has been established, the participants rely on artifacts and risk assessment results to elicit an initial set of security requirements. Two subsequent stages are spent categorizing and prioritizing these requirements for management's use in making tradeoff decisions. Finally, an inspection stage is included to ensure the consistency and accuracy of the security requirements that have been generated.

Subsequent to the development of SQUARE, pilot projects were conducted to validate the method, and associated tools, academic lectures, and industry workshops were developed and delivered. CERT collaborated with researchers in the U.S. and elsewhere in the research of the SQUARE method and its transition into practice. The method was extended to address privacy (P-SQUARE) and acquisition (A-SQUARE).

A robust tool to help projects easily use the SQUARE process for security, privacy, or both is available.

Value to Security

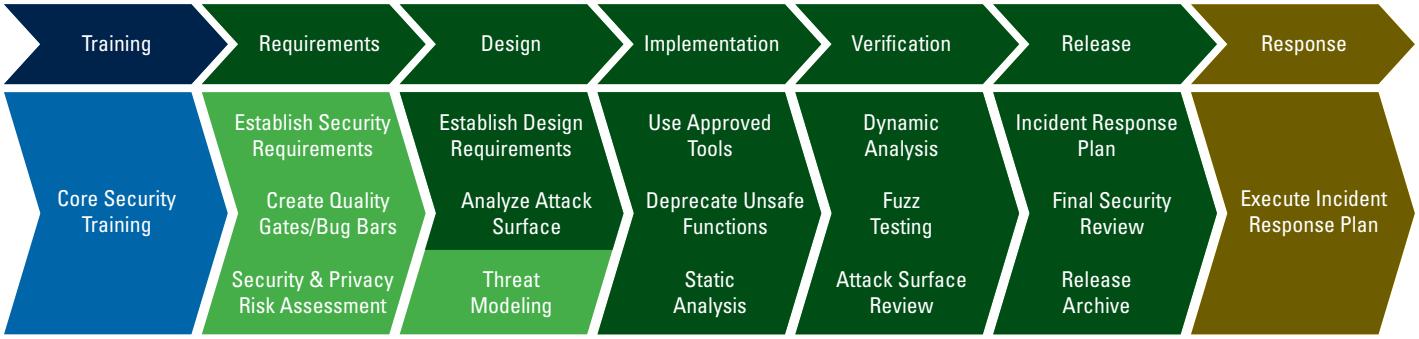
The 9-step SQUARE process results in a set of categorized and prioritized security requirements that have passed inspection. It can be used to establish requirements for both development and acquisition projects. Security and/or privacy risk assessment occur in SQUARE Step 4.

Using SQUARE can enable organizations to develop more secure, survivable software and systems, more predictable schedules and costs, and achieve lower costs. Organizations that are acquiring software have the same security concerns as those that are developing software, but they usually have less control over the actual development process. Accordingly, SQUARE has been enhanced to support use during acquisition.

In our work with SQUARE we have found that many forward-thinking organizations already have documented processes and are not ready to embark on a whole new process. For those organizations, there is more benefit to enhancing their existing processes

to ensure that security requirements are adequately addressed. Likewise, they may have existing requirements engineering tools in use and need to understand how to address security requirements in the context of those tools. This approach can be applied to organizations that are concerned with security requirements, privacy requirements, or acquisition of secure developed products or commercial, off-the-shelf (COTS) products.

Connection to Microsoft SDL



Core Security Training

- Security Requirements Engineering Using the SQUARE Method http://www.sei.cmu.edu/training/P104.cfm

Establish Security Requirements

Create Quality Gates/Bug Bars

Security & Privacy Risk Assessment

Threat Modeling

Resources for Further Solution Details

1. Mead, N. R., Hough, E., & Stehney, T. *Security Quality Requirements Engineering (SQUARE) Methodology* (CMU/SEI-2005-TR-009). Software Engineering Institute, Carnegie Mellon University, 2005. http://www.sei.cmu.edu/publications/documents/05.reports/05tr009.html
2. Bijwe, A., Mead, N.R. Faculty Advisor. *Adapting the Square Process for Privacy Requirements Engineering* (CMU/SEI-2010-TN-022). Software Engineering Institute, Carnegie Mellon University, 2010.

Security Risk Analysis Toolkit

Description

The term *risk* is used universally, but different audiences often attach different meanings to it. In fact, the details about risk and how it supports decision making depend upon the context in which it is applied. The SEI CERT Division is focused on security risk, which is defined as the probability of suffering loss due to the occurrence of a security event or threat. For several years, the SEI CERT Division has been developing methods for assessing and managing security risk across the lifecycle and supply chain. A key product of this effort is the Security Risk Analysis Toolkit, which includes multiple methods for analyzing risk. A unique feature of the toolkit is that includes methods that examine risk from two distinct perspectives: organizational and technical.

The organizational perspective of security risk is focused on establishing and maintaining confidence that an organization's security objectives will be achieved. It answers the following question: Do organizational programs and processes enable the achievement of security objectives? The Mission Risk Diagnostic (MRD) is designed to provide managers with a level of confidence that their organization's security objectives will be achieved. The MRD is a versatile method for assessing a project, program, or process and projecting its potential for success. It analyzes a set of systemic risk factors in relation to the mission and objectives being pursued and provides a foundation for decision making. The MRD can be applied across the lifecycle (i.e., acquisition, development, operations) and supply chain.

In contrast, the technical perspective is focused on engineering software-reliant systems to better protect information assets. It answers the following question: Are the security risks to information assets within an acceptable tolerance? The Security Risk Assessment (SRA) is designed to address the technical perspective of security risk. When applying the SRA, analysts conduct a detailed examination of the core components of security risk (i.e., threat, vulnerability, and consequence) for selected information assets. After they identify security risks for those assets, analysts prioritize the risks and develop plans for controlling the highest-priority risks. The SRA is a flexible engineering method that can be applied across the software acquisition and development lifecycle and supply chain.

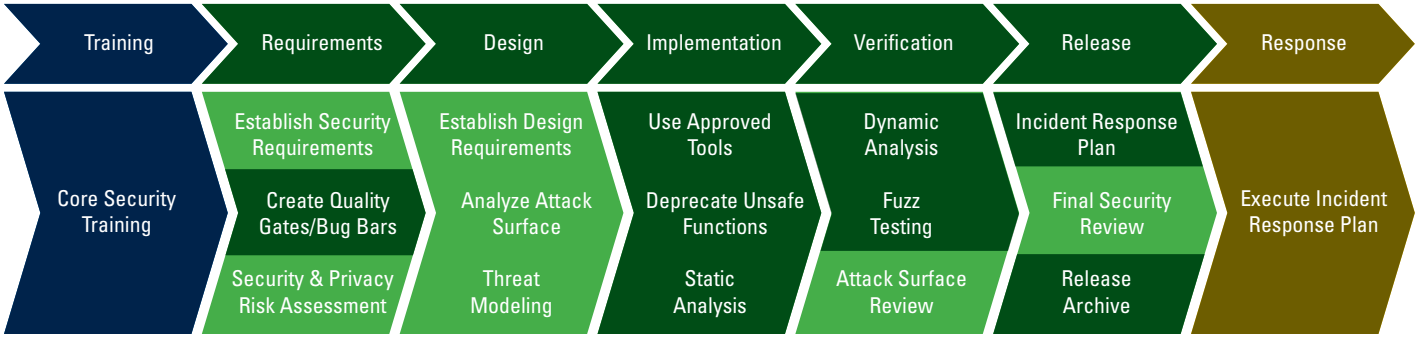
Finally, the Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE[SM]) can also be used to examine risk from the technical perspective. OCTAVE is an organization-wide approach for assessing operational security risks to critical information assets. In contrast to the SRA, OCTAVE is focused on the operational phase of the software lifecycle. It is not designed to assess risk during software acquisition and development.

Value to Security

The Security Risk Analysis Toolkit provides security value in two significant ways. First, managers can apply methods from the toolkit to address the organizational perspective of security risk. As a result, managers can establish a reasonable degree of confidence that their organizations programs and processes are in position to achieve their security objectives.

Second, engineers can apply methods that are designed to address the technical perspective of security risk. Here, they conduct a detailed examination of how security threats can exploit vulnerabilities to affect the confidentiality, integrity, and availability of information assets. Engineers can then implement appropriate controls to ensure that security risks to information assets are kept within an acceptable tolerance over time.

Connection to Microsoft SDL



Establish Security Requirements

Security & Privacy Risk Assessment

Establish Design Requirements

Analyze Attack Surface

Threat Modeling

Attack Surface Review

Final Security Review

Resources for Further Solution Details

1. Alberts, Christopher & Dorofee, Audrey. *Mission Risk Diagnostic (MRD) Method Description* (CMU/SEI-2012-TN-005). Software Engineering Institute, Carnegie Mellon University, 2012. http://www.sei.cmu.edu/library/abstracts/reports/12tn005.cfm
2. Alberts, Christopher; Allen, Julia; & Stoddard, Robert. *Risk-Based Measurement and Analysis: Application to Software Security* (CMU/SEI-2012-TN-004). Software Engineering Institute, Carnegie Mellon University, 2012. http://www.sei.cmu.edu/reports/12tn004.pdf
3. Alberts, Christopher & Dorofee, Audrey. *Risk Management Framework* (CMU/SEI-2010-TR-017). Software Engineering Institute, Carnegie Mellon University, 2010. http://www.sei.cmu.edu/library/abstracts/reports/10tr017.cfm
4. Alberts, Christopher & Dorofee, Audrey. *A Framework for Categorizing Key Drivers of Risk* (CMU/SEI-2009-TR-007). Software Engineering Institute, Carnegie Mellon University, 2009. http://www.sei.cmu.edu/library/abstracts/reports/09tr007.cfm
5. Alberts, Christopher & Dorofee, Audrey. *OCTAVE℠ Criteria, Version 2.0* (CMU/SEI-2001-TR-016). Software Engineering Institute, Carnegie Mellon University, 2012. http://www.sei.cmu.edu/library/abstracts/reports/01tr016.cfm
6. Alberts, Christopher; Behrens, Sandra; Pethia, Richard; & Wilson, William. *Operationally Critical Threat, Asset, and Vulnerability Evaluation℠ (OCTAVE℠) Framework, Version 1.0* (CMU/SEI-99-TR-017). Software Engineering Institute, Carnegie Mellon University, 1999. http://www.sei.cmu.edu/library/abstracts/reports/99tr017.cfm
7. Alberts, Christopher & Dorofee, Audrey. *Managing Information Security Risks: The OCTAVE℠ Approach*. Addison-Wesley, 2002. http://www.sei.cmu.edu/library/abstracts/books/0321118863.cfm

Supply Chain Assurance Guidelines and Self-Assessment

Description

This effort is developing top-level guidance for acquisitions to manage software system supply chain risks.

There has been increasing U.S. government activity in supply chain risk management for information, technology, and communication systems. The risks of interests include two kinds of malicious events: 1) the compromise of supply chain logistics to tamper with an item or replace it with a counterfeit one or 2) the deployment of software systems with embedded and unintentional weaknesses that could enable a threat actor to compromise that system. It can be argued that the second risk falls under system assurance rather than supply chain risk management, but it is still an acquisition risk that should be considered.

An acquisition challenge is the complexity of a system supply chain with contractors and subcontractors, and suppliers to those entities. Software system supply chains are complex in terms of the number of suppliers and components. An in-depth risk analysis for communication systems could involve more than 30 factors which is not practical.

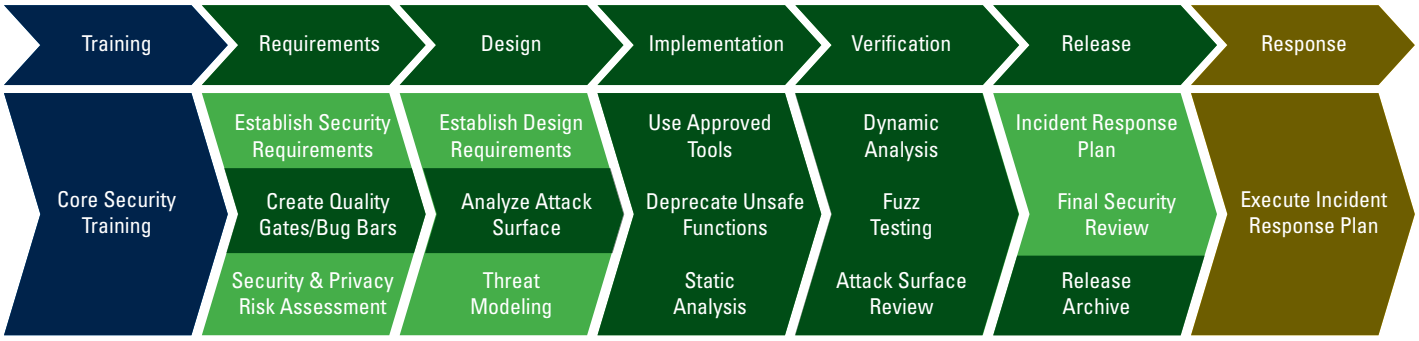
An acquisition has to recognize that some supply chain risks arise from requirements such as the use of mobile communications. Resource constraints limit the number of threats that can be addressed, and the increasing complexity of software systems likely introduces new vulnerabilities. An acquisition has to consider what the acceptable risks are.

Value to Security

This effort is connected to attack surface analysis and threat modeling in the Microsoft SDL.

Government acquisitions often concentrate on security functionality such as authentication and authorization or on the security infrastructure such as securing networks. Reducing the risk of unintentional vulnerabilities depends on incorporating security throughout the system development lifecycle. But that transition also has to consider source selection criteria, and for contracted development, that there is sufficient evidence associated with contractor deliverables to validate that security has been appropriately considered.

Connection to Microsoft SDL



Establish Security Requirements

Security and Privacy Risk Assessment

Establish Design Requirements

Threat Modeling

Incident Response Plan

Final Security Review

Resources for Further Solution Details

1. Survivability Assurance for System of Systems, Robert J. Ellison, John Goodenough, Charles Weinstock, & Carol Woody, CMU/SEI-2008-TR-008 May 2008 http://www.sei.cmu.edu/library/abstracts/reports/08tr008.cfm
2. Software Supply Chain Risk Management: From Products to Systems of Systems, Robert J. Ellison, Christopher Alberts, Rita Creel, Audrey Dorofee, & Carol Woody, CMU/SEI-2010-TN-026 December 2010 http://www.sei.cmu.edu/library/abstracts/reports/10tn026.cfm

Survivability Analysis Framework (SAF)

Description

The Survivability Analysis Framework (SAF), a structured view of people, activities, and technology, helps organizations characterize the complexity of multisystem and multi-organizational business processes. The SAF is designed to

- identify potential problems with existing or near-term interoperations among components within today's network environments
- highlight the impact on successful execution as constrained interoperation moves to more dynamic connectivity
- increase our assurance that operational mission and critical work processes can be successfully executed in the presence of stress and possible failure

Sample work processes (mission threads) are analyzed as follows:

- Define the criteria for a successful execution of a work process.
- Describe the individual sequenced activities (as shown in the diagram below) that compose a work process; for each activity include preconditions, actions, post-conditions, dependencies, the resources required, and the roles of any participants.
- Describe, again for each activity, the operational conditions (stresses) that could compromise the overall work process and the system's response to those conditions. Such conditions include the data exchanges among systems, resource constraints and failures, and user and administrator actions.

Value to Security

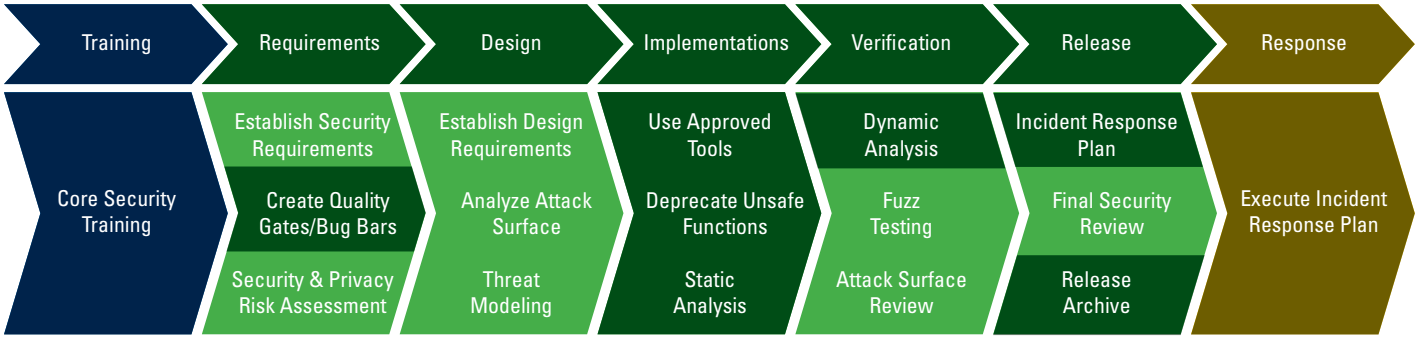
A great many of the software problems we have today have little to do with malice. Things break. Complexity and change pervade today's organizations. Simpler systems could be understood and their behavior characterized, but software is taking on more and more functions, resulting in greater complexity along with unintended consequences. Complex systems break in complex ways.

The growing need for interconnected and often global operations means that an operational mission or critical work process is usually supported by multiple, often independently developed systems. Mission problems can arise from the ways in which the shift was made from "stovepipes" (stand-alone systems) to interdependent systems that must work together. The pieces were not designed to address the ways in which the whole must function. The likelihood of failure increases when assumptions and decisions within one organizational area are inconsistent with those of another. Addressing this problem requires an evaluation of technology (software and systems) within the context of the mission:

- establishing the role of technology in mission success
- analyzing the technology impact on potential mission failure
- identifying mission-critical technology dependencies
- considering threats to mission success: operating conditions or actions triggered by threat agents

Building examples of the operational flow of people, actions, and technology interactions to successfully address an organizational mission provides a means to (1) look across multiple systems and organizations to identify integration challenges, (2) consider architecture tradeoffs that carry impacts beyond a single component or a single system, and (3) consider the linkage of technology to critical organizational capabilities.

Connection to Microsoft SDL



Establish Security Requirements

Security & Privacy Risk Assessment

Establish Design Requirements

Analyze Attack Surface

Threat Modeling

Fuzz Testing

Attack Surface Review

Final Security Review

Resources for Further Solution Details

1. Ellison, Robert & Woody, Carol. *Survivability Analysis Framework.* http://www.sei.cmu.edu/library/abstracts/reports/10tn013.cfm
2. Ellison, Robert J.; Goodenough, John; Weinstock, Charles; & Woody, Carol. *Survivability Assurance for System of Systems* (CMU/SEI-2008-TR-008). Software Engineering Institute, Carnegie Mellon University, 2008. http://www.sei.cmu.edu/library/abstracts/reports/08tr008.cfm

Computer Security Incident Response Team (CSIRT) Management

Description

CSIRT management establishes capabilities to develop, operate, and improve incident management. The CERT Division's CSIRT development and training (CDT) team makes products, training, reports, and workshops available to the global internet community.

Products

- a suite of public incident management and CSIRT training courses (http://www.cert.org/training/)
- publications and resources for creating and operating incident management and CSIRT capabilities (http://www.cert.org/csirts/resources.html)
- Incident Management Capability Metrics Version 0.1 (http://www.sei.cmu.edu/library/abstracts/reports/07tr008.cfm)
- Incident Management Mission Diagnostic Method, Version 1.0 (http://www.cert.org/archive/pdf/08tr007.pdf)

Services

- assisting organizations in planning, implementing, and improving their incident management capabilities
- developing and delivering public courses (http://www.cert.org/training), on-site courses, and facilitated workshops
- evaluating incident management capabilities (http://www.cert.org/csirts/metrics.html)
- certifying incident handlers (http://www.cert.org/certification/)
- licensing of training materials and a train-the-trainer program (http://partners.clearmodel.com/guide/becoming-a-partner/)

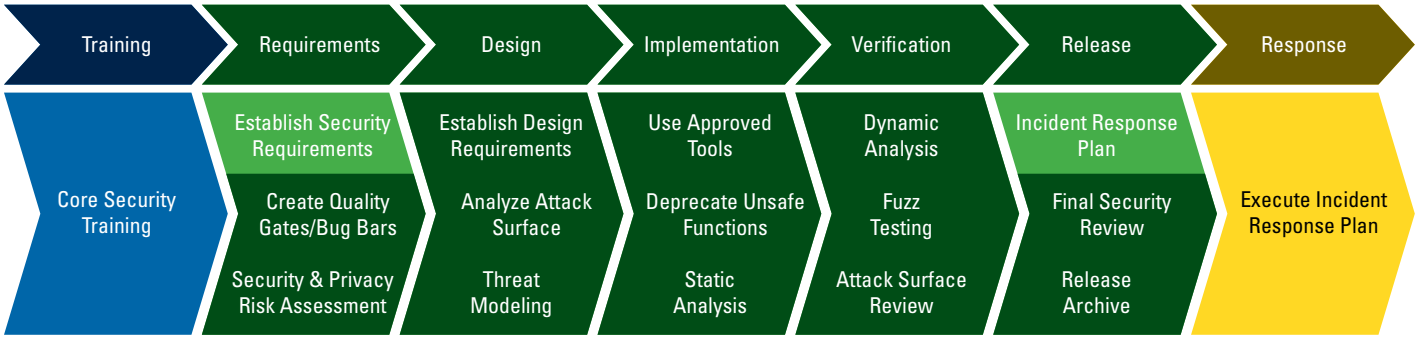
Research

- incident management and CSIRT best practices
- CSIRT evaluation metrics, measurements, and tools
- incident management body of knowledge
- response strategies and best practices for various incident types

Value to Security

Not all security problems can be prevented. The capability to recognize, resist, and recover from security issues discovered during system operation must be established and supported as part of the secure lifecycle responsibilities. Operational security problems are referred to as incidents. Incident response can be effectively handled by a well-trained computer security incident response team.

Connection to Microsoft SDL



Core Security Training

- Overview of creating and managing CSIRTS (http://www.sei.cmu.edu/training/P68.cfm)
- Creating a Computer Security Incident Response Team (http://www.sei.cmu.edu/training/P25.cfm)
- For CSIRT managers: Managing Computer Security Incident Response Teams (http://www.sei.cmu.edu/training/P28.cfm)
- For new incident handlers: Fundamentals of Incident Handling (http://www.sei.cmu.edu/training/P26.cfm)
- For experienced incident handlers: Advanced Incident Handling for Technical Staff (http://www.sei.cmu.edu/training/P23B.cfm)

Establish Security Requirements

Incident Response Plan

Execute Incident Response Plan

Resources for Further Solution Details

1. Software Engineering Institute. *Training and Education Courses.* http://www.cert.org/training/ (2013).
2. Software Engineering Institute. *Resources for Computer Security Incident Response Teams (CSIRTs).* http://www.cert.org/csirts/resources.html (2013).
3. Dorofee, Audrey; Killcrece, Georgia; Ruefle, Robin; & Zajicek, Mark. *Incident Management Capability Metrics Version 0.1* (CMU/SEI-2007-TR-008). Software Engineering Institute, Carnegie Mellon University, 2007. http://www.sei.cmu.edu/library/abstracts/reports/07tr008.cfm
4. Software Engineering Institute. *Evaluating Incident Management Capabilities.* http://www.cert.org/csirts/metrics.html (2013).
5. Software Engineering Institute. *CERT®-Certified Computer Security Incident Handler.* http://www.cert.org/certification/ (2013).
6. CMMI Institute. *Becoming a Partner.* http://partners.clearmodel.com/guide/becoming-a-partner/ (2013).
7. Software Engineering Institute. *Creating a Computer Security Incident Response Team.* http://www.sei.cmu.edu/training/p25.cfm (2013).
8. Software Engineering Institute. *Managing Computer Security Incident Response Teams.* http://www.sei.cmu.edu/training/p28.cfm (2013).
9. Software Engineering Institute. *Fundamentals of Incident Handling.* http://www.sei.cmu.edu/training/p26.cfm (2013).
10. Software Engineering Institute. *Advanced Incident Handling.* http://www.sei.cmu.edu/training/p23b.cfm (2013).
11. Software Engineering Institute. *Overview of Creating and Managing CSIRTs.* http://www.sei.cmu.edu/training/P68.cfm (2013).

Summary Mapping and Recommended Use

CERT Solution	Microsoft SDL																
	Training	Requirements			Design			Implementation			Verification			Release			Response
	Core Security Training	Establish Security Requirements	Create Quality Gates / Bug Bars	Security & Privacy Risk Assessment	Establish Design Requirements	Analyze Attack Surface	Threat Modeling	Use Approved Tools	Deprecate Unsafe Functions	Static Analysis	Dynamic Analysis	Fuzz Testing	Attack Surface Review	Incident Response Plan	Final Security Review	Release Archive	Execute Incident Response Plan
CSIRT Management	●	●												●			●
Secure Coding in Java, C and C++	●	●	●					●	●								
Source Code Analysis Laboratory (SCALe)		●	●						●	●	●	●			●		
Vulnerability Discovery and Fuzz Testing											●	●	●				
SQUARE Method for Security Requirements and tool	●	●	●	●			●										
Security Risk Analysis Toolkit		●		●	●	●	●						●		●		
Supply Chain Assurance Guidelines and Self-Assessment		●		●	●		●							●	●		
Survivability Analysis Framework (SAF)		●		●	●	●	●					●	●		●		

Software Engineering Institute

Carnegie Mellon®