MS_logo_KEssential Software

June 2, 2010

For the latest information, please see <http://www.microsoft.com/sdl>.



This document is provided “as-is.” Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.

Some examples depicted herein are provided for illustration only and are fictitious.  No real association or connection is intended or should be inferred.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2010 Microsoft Corporation. All rights reserved.

**Contents**

[Introduction 2](#_Toc259654343)

[The Need for Software Security Training 2](#_Toc259654344)

[Bug-Infested Software 2](#_Toc259654345)

[Breaches 3](#_Toc259654346)

[Specialized Knowledge 4](#_Toc259654347)

[Software Security vs. IT Security 4](#_Toc259654348)

[ROI for a Trained Workforce 5](#_Toc259654349)

[Commitment to Training 6](#_Toc259654350)

[Training New Hires 6](#_Toc259654351)

[Periodic Training 6](#_Toc259654352)

[Backfill for Attrition 7](#_Toc259654353)

[New Threats and Technologies 7](#_Toc259654354)

[Characteristics of Good Security Training 7](#_Toc259654355)

[Experienced Instructors 8](#_Toc259654356)

[Role-Based Training 8](#_Toc259654357)

[Computer-Based and Instructor-Led Training 9](#_Toc259654358)

[Outsourced vs. In-House Training 9](#_Toc259654359)

[Description of the Microsoft Security Development Lifecycle (SDL) Core Training Courses 10](#_Toc259654360)

[Introduction to the Microsoft Security Development Lifecycle (SDL) 12](#_Toc259654361)

[Introduction to Microsoft Threat Modeling 12](#_Toc259654362)

[Basics of Secure Design, Development, and Test 12](#_Toc259654363)

[Privacy in Software Development 13](#_Toc259654364)

[Description of Advanced SDL Training Content – 200 Level 13](#_Toc259654365)

[Secure Design: Attack Surface Reduction 13](#_Toc259654366)

[Secure Development: Secure Coding for Managed Code (C#/Java/.NET) 13](#_Toc259654367)

[Secure Development: Secure Coding for Native Code (C/C++) 14](#_Toc259654368)

[Security Testing 14](#_Toc259654369)

[Description of Advanced SDL Training Content – 300 and 400 Level 15](#_Toc259654370)

[Advanced Secure Design 15](#_Toc259654371)

[Advanced Secure Development 15](#_Toc259654372)

[Advanced Security Testing 15](#_Toc259654373)

[Description of Ancillary Courses 16](#_Toc259654374)

[Security Response and Incident Management 16](#_Toc259654375)

[Security for Managers 16](#_Toc259654376)

[Security Tools 16](#_Toc259654377)

[Conclusion 16](#_Toc259654378)

[Appendix A: Resources 18](#_Toc259654379)

# Introduction

Understanding software security problems is a foundational part of building better software. While computer science and software development are established disciplines in business and education, software security remains something of an afterthought. There seems to be an ongoing misconception that good software security is not compatible with tight schedules and cutting-edge functionality. A commitment to software security training is a key tenet of the Microsoft Security Development Lifecycle (SDL) and vital to ensuring that secure software can take its place as a top priority along with software features and delivery timelines.

# The Need for Software Security Training

Businesses must ensure that the software they create or acquire includes the security properties required to meet their current and evolving business and compliance needs. Accomplishing this requires that skilled individuals have a clear understanding of software security with respect to applicable business objectives and technologies in use, along with the skills required to specify, develop, test, and field the software appropriately.

Change is a large part of the software security problem, and companies are struggling to keep pace. New technologies, such as Web 2.0 and rich Internet applications, are stretching existing skill sets. The evolution from “drive-by” attacks to scripted attacks authored by highly skilled professionals is changing the basics of risk-management, regulatory, and compliance requirements. An additional issue is that some security technologies simply don’t work as advertised and can leave software unexpectedly unprotected. Software security is emerging as a business imperative, and success in this area requires that all stakeholders in the software life cycle be trained in building security into software as it is being created.

## Bug-Infested Software

Statistics from the [United States Computer Emergency Response Team](http://www.cert.org/stats/) (CERT) show a rapid progression in total software vulnerabilities catalogued, hovering at about 7,000–8,000 per year during 2006 through 2008, up from about 1,100 in 2000. The [National Vulnerability Database](http://nvd.nist.gov/home.cfm) shows similar growth.

These and related knowledge repositories have actually become so large that additional government and industry projects were started to categorize the software vulnerability information and to make it accessible by others. The [Common Vulnerabilities and Exposures database](http://cve.mitre.org/) provides unique identifiers for software vulnerabilities to help ensure that everyone is talking about the same problem. The [National Vulnerability Database](http://nvd.nist.gov/) contains vulnerability management data in a common format that enables various kinds of automation. Ironically, these efforts to document software vulnerability information became so widespread and divergent that the [Common Weakness Enumeration program](http://cwe.mitre.org/) was established to help tie them all together.

In fact, the software security problem has become so pervasive that it has spawned entire industries. Organizations now spend hundreds of millions of dollars per year on tools and technologies to help find software vulnerabilities. Web application scanners perform dynamic analysis, code review tools perform static analysis, and software fuzzers look for edge and boundary conditions. These technologies, and their related services, are all aimed at catching bugs before customers or criminals find them, not at preventing software vulnerabilities—both coding bugs and architecture design problems—in the first place.

## Breaches

One need look no further than the [Open Security Foundation DataLoss Database](http://datalossdb.org/) and [Privacy Rights Clearinghouse Chronology of Data Breaches](http://www.privacyrights.org/ar/ChronDataBreaches.htm) to see the magnitude of the privacy breaches occurring when organizations lose control of the personally identifiable information (PII) entrusted to them. The [Web Application Security Consortium (WASC) Web Hacking Incidents Database](http://whid.xiom.com/) catalogs an ever-increasing number of public incidents associated specifically with Web application security vulnerabilities.

Attackers are quickly evolving their methods. They are moving “up the stack” to remotely attack applications in addition to hosts and network devices. Attacks are also becoming stealthier and more targeted as attackers also set their sights on the computer BIOS, hypervisors, and other low-level software that sits below the applications. Of course, attackers are also exploiting inappropriate trust relationships between business applications and various technologies, such as wireless, mobile, virtualization, and cloud computing. These attacks are in addition to the day-to-day attacks from malicious software, “bot” networks, and highly paid criminals targeting specific organizations and applications.

* In January 2007, TJX Companies, Inc., revealed that 45 million customer credit and debit card numbers were stolen due to what investigators described as malicious software placed on the company network. Losses have been estimated at $256 million.[[1]](#footnote-1)
* In March 2008, the East Coast supermarket chain Hannaford Brothers disclosed that 4.2 million credit and debit card accounts were compromised due a breach caused by malicious software.[[2]](#footnote-2)
* In January 2009, Heartland Payment Systems discovered malicious software on its computer network that exposed the card numbers and the cardholder names of the 100 million payment card transactions it processed per month.[[3]](#footnote-3)
* Details are still emerging regarding what appears to be a highly coordinated 2008 attack on RBS WorldPay (a payment processor) that included software intrusions and subsequent ATM fraud of a reported $9 million. An initial intrusion into RBS WorldPay networks that ultimately accessed personal data for approximately 1.3 million people, including nearly 1.1 million Social Security numbers, provided data for thieves to create 100 payroll and debit cards.

These incidents are not without cost. With respect to breaches of PII, annual studies from the [Ponemon Institute](http://www.pgp.com/insight/newsroom/press_releases/2008_annual_study_cost_of_data_breach.html) indicate the cost per compromised customer record in 2008 has grown nearly 40 percent to $202 from $138 in 2005. The unanticipated costs associated with disclosing a breach with even a relatively small number of customer records can have a major impact on an organization’s cash flow. A *customer record* here is data that can be used to uniquely identify a single individual (for example, address, phone number, or employment) and may include additional items, such as Social Security numbers, credit card numbers, and health information.

Why is all this software insecurity happening? There are probably many reasons, but two stand out:

* Attacking software is recession-proof. While organizations may have to learn to do more with less, professional cyber-thieves do not.
* The people implementing software—from product managers and business analysts, to architects, developers, and testers, to operations—often do not have the skills required to build security into software. Training is part of the solution for this problem.

## Specialized Knowledge

Competence is commonly defined as the personal characteristics that allow superior performance. These characteristics include knowledge, skills, aptitude, attitude, and similar characteristics amonth others. For any given task, it is possible to create a competency model, which is effectively the road map for the desired performance. Of course, a road map alone does not guarantee acceptable results. Again, training and education help to ensure that developers are appropriately equipped.

Education is the process that fills minds with knowledge and expands one's ability to reason about the issues faced. Training, by contrast, teaches skills that one can use to accomplish a task. It is important to establish a solid educational foundation before training occurs if you expect the person to be able to react appropriately when circumstances are outside their training. As the degree of variability in the tasks to be performed increases, so will the training required to achieve the needed day-to-day proficiency.

Of course, individuals are very unique, and as such, education does not always require an extended stay at an institution of higher learning—everyone learns at a different pace and through different methods. Some people require a structured learning environment and hours of study, while others just “get it.” Training, on the other hand, is a necessary thing, and a periodic refresh with current information is important in maintaining skill level. The cycle of education (for example, in software security or static analysis theory) followed by training (for example, in secure code development or use of a specific code review tool) will likely be repeated several times as organizations mature the skills of their employees.

Why is this important? Every stakeholder in the software development life cycle has a part to play in ensuring software security and, therefore, every stakeholder requires training. It is important to understand several key points here. An organization includes people with a wide variety of education levels and does not train everyone to do everything; therefore, role-specific training is required. Even within specific roles (for example, architect, developer, or tester), not everyone has the aptitude for a given technical approach. It‘s important that those struggling with new concepts are given extra support.

Similarly, it is important to use the training sessions to find people throughout the organization who have the aptitude (and attitude) for software security and to bring them together informally to build a *satellite*—a group of people dedicated to the cause of software security regardless of the organization in which they work or the day-to-day roles they fill. These individuals are the source for many grass-roots efforts and often act as mentors for others.

## Software Security vs. IT Security

Software security is not the same as security software. The category of security software can include things like crypto libraries and authentication and authorization subsystems. It can even be stretched to include other things more traditionally in the IT security realm, including intrusion detection and prevention applications, log analyzers, network access control devices, and even firewalls. Security software—and IT security overall—is very important and must be aligned with the security needs of the organization. Naturally, IT (system, desktop, and network) security practitioners require significant training specific to their roles.

Software security is different. It is the property of software that allows it to continue to operate as expected even when under attack. Software security is not a specific library or function call, nor is it an add-on that magically transforms existing code. It is the holistic result of a thoughtful approach applied by all stakeholders throughout the software development life cycle. As discussed previously, software security practitioners also require significant training specific to their roles.

With respect to training, a comparison of IT security training prevalence versus software security training prevalence provides some insight. A late 2006 [CompTIA survey](http://www.awareity.com/public/Awareity-Validation-CompTIA.pdf) reported that, although human error accounted for 60 percent of information security breaches (up from 47 percent in 2005), only 29 percent of the 574 organizations surveyed required security training for IT staff and only 36 percent even offered it. This, while 84 percent of the organizations that did provide training credited it specifically with reducing the number of major security breaches. If one believes that the more mature and established organizational use of IT and computer security training is much more widespread than the use of software security training, there is indeed a very small percentage of software security stakeholders who are receiving adequate training.

Even as one makes the distinction between software security and other activities, it is important to note that software security and security software intersect in several places. Perhaps the most common is in the reuse of secure-by-design architectures and libraries when building more secure software. Similarly, software security and IT security intersect with the need for good infrastructure and platform security when deploying more secure software. Applications must be deployed on appropriately configured hosts and networks behind proper firewalls and related security devices. This means that IT security personnel must be included in foundational software security awareness training.

## ROI for a Trained Workforce

Calculating return on investment (ROI) can be difficult in the best of times. Predicting return can be downright impossible when it comes to increasing skills as a means of preventing events that might not have happened anyway. There is not a one-to-one correlation between software vulnerabilities and material loss, just as there is not a one-to-one correlation between any other kind of vulnerability and successful attacks. That is, software bugs are not always vulnerabilities, and vulnerabilities are not always exploited.

With risk management and ROI in mind, it seems clear that one wants to prevent as many software vulnerabilities as are economically feasible. However, the explosive growth in vulnerability data discussed earlier indicates that one does not want to attempt this vulnerability reduction by simply moving costs to deploying more black-box testing in the software assembly line with hopes of catching all the security and privacy issues after they are created. More simply, one wants to get the greatest return by creating fewer software vulnerabilities in the first place. That requires specialized, role-based training for every participant in the development lifecycle.

The cost savings of finding and fixing vulnerabilities very early in the development cycle has been well established for several years. In its 2002 report, [The Economic Impacts of Inadequate Infrastructure for Software Testing](http://www.nist.gov/director/prog-ofc/report02-3.pdf), NIST reports (in more than 300 pages) that it is 30 times more costly to repair a bug after product release than it is to have the same bug repaired during requirements analysis or architectural design.

Along these lines, [The Depository Trust and Clearing Corporation (DTCC)](http://searchsecurity.techtarget.com/magOnline/0,,sid14_gci1346790,00.html) has direct experience with having to do “more with less.” DTCC reports that inserting security controls early into the software development life cycle, and weeding out vulnerabilities well before they appear in functional code that ends up in production, has resulted in close to $2 million in productivity gains on a base of $150 million spent for development. These efforts are part of a three-year-old initiative that leans heavily on training.

# Commitment to Training

Everyone involved in software development must contribute to creating secure software. This is not possible without sufficient training, and sufficient training does not happen without an ongoing corporate commitment. A true commitment to making time available, providing useful courses, and facilitating real progress tells everyone that training is important and that their efforts are both valued and contributing to corporate success. Remember, unfunded mandates simply do not get done.

A [recent study of nine large firms](http://bsi-mm.com) provides insight into the importance of training in organizations committed to software security. These firms cover the financial services, independent software vendors, and high-technology verticals, and their development groups ranged in size from 450 to 30,000 engineers. Teams dedicated to software security ranged in age from 2.5 to 10 years. All of these firms have evolved mature, role-based training programs, many of which include material in both instructor-led and e-learning forms that are available at onboarding and for later refresher training. All credited their ongoing commitment to training as contributing to their overall software security success.

## Training New Hires

Old habits are hard to break, so it is far better to teach correct procedures from the beginning. Training developers as soon as they are hired provides several advantages. It offers a way to immediately indoctrinate new employees in organizational software security policies and prescriptive guidance. It also gives new employees a chance to show their strengths and to provide some of their experience and ideas back to the organization. Training during the onboarding process can also help ensure that the employer will benefit from the entire term of the developer’s employment, as opposed to some uncertain future point when mistakes have already been made.

Although developers are used as the example here, it is important to remember that the principle applies for everyone who contributes to software security. The list of direct contributors depends on local culture and processes but likely includes product managers, business analysts, architects, developers, testers, and operations. Remember, your software security stakeholders may also include customer service representatives, legal staff, and executive management—they should receive software security awareness training, too!

## Periodic Training

Periodic reinforcement of skills is an important part of increasing competence. Initially, it is often necessary to allocate time to bring everyone up to the same level with respect to software security. This requires some foundational training that addresses vocabulary, concepts, and organizational prescriptive guidance.

However, the foundational training only sets the baseline and does so only at a point in time. This training must be refreshed to keep pace with changes in business objectives, threats, technologies, attacks, requirements (for example, compliance), and related issues. This requires a concerted effort aimed at software security. It also requires a commitment to long-term employee growth in these particular skill areas.

## Backfill for Attrition

When backfilling teams, organizations need to start asking questions about software security in the hiring process. Prefer candidates that have direct experience, all other things being equal. Absent direct experience, look for candidates that have an interest in software security and are excited about the prospect of creating better software.

Aside from including software security discussions in the hiring process, ensure that backfill employees are given some initial time to go through new hire training before settling in to their daily routines.

## New Threats and Technologies

In this context, a *threat* is an undesired event, often best described as an effect that might damage or compromise an asset or objective. Note that threats are not always malicious in nature. One way to keep pace with evolving threats is to prioritize business security requirements to define a security strategy from a defensive perspective. This helps focus resources on immediate needs.

When you begin to think about threats, it may be easier to take a business requirement approach compared to taking only an adversarial approach to the problem. The adversarial approach often causes organizations to work with spotty or non-existent attacker data on motives, resources, and similar concepts, rather than focusing on information clearly pertinent to the software at hand.

Whether it is about threats or technology stacks, teaching people the wrong information can be a destructive waste of time. The initial burst of enthusiasm for training dries up quickly when the material is not relevant to current needs and regenerating that interest may be difficult. Ensure that while your organization is, for example, moving from mainframe programming to managed code to rich Internet applications, the appropriate threat and technology details are made available to everyone.

The point is that threats and business requirements are constantly evolving, and both training and the broader software security processes must keep pace. Regardless of whether you are maintaining older software in relatively stable back-end environments or creating new software with the latest technology stacks, making sure that you have the best threat information and technology-specific knowledge helps ensure good software security.

# Characteristics of Good Security Training

Everyone has been in training that left them bored and disinterested. This did not necessarily happen because the student was already an expert but probably because there was no clear connection between the student’s responsibilities and a few key items:

* Base knowledge
* Interest level
* Job goals
* Individual learning style
* Material presented

You must know your audience. This means that when you are creating or outsourcing technical training, first consider these three fundamentals:

* “This training is appropriate for people who...”
* “In this training, you will learn how to...”
* “After this training, you will be able to...”

If you cannot answer these questions—if you cannot outline a basic competency model—you are not prepared to spend time and money. This would be roughly the equivalent of building code before understanding the requirements—but still expecting success. It will not happen.

For your organization, good security training is based on your technologies and on your prescriptive guidance. If you are building new training, customizing existing training, or integrating commercial training into your development processes, ensure that it is specific to the skills and knowledge your people need.

Another way to get requirements for training material is to look at the problems found in the organization—where coding bugs are created and where they are being missed. Those make excellent sources for training. Ensure that the appropriate subject-matter experts (SMEs) are included in the process. These people verify correctness and appropriateness.

Training must be supported by executive management and reinforced as part of the culture, whether that is through the human resources department, test results, metrics, or something else. Everyone has to see evidence that training is an integral part of business as usual, that management supports it, that there is time for it built into schedules, and that attendance and improvement is required. If the students have a firm grasp on the right thing to do, your training program stands a much better chance of success.

The little things account for a lot in training material—one idea per slide, good notes, a logical flow, quick reviews interspersed with new ideas, and slides that are not simply about the topic but that actually show how to execute the actions required for the topic. (Have you ever attended a course that described a topic for hours and you left still not knowing what to actually do?) Examples that really are examples (since people reuse them) and are not broken in various ways are also a must-have for good training.

Include examples from the business, both to build interest in the content and to prevent recurrences. Remember that good training changes behavior, which is likely exactly the outcome you are seeking.

## Experienced Instructors

It is always a pleasure to attend a course presented by an enthusiastic professional instructor; there is no substitute for field experience. However, when it comes to software security training, it is important that the instructor be able to deviate from the planned lesson and answer questions based on their real-world experience. It is never the case that all the information is in the slides. Students will ask questions that are outside the material and will feel something is missing if the instructor has no such experience.

Whether you build or buy training material, realize that you are placing some portion of your future success in the hands of the instructors. Be sure that they are adequately prepared for the task.

## Role-Based Training

One size does not fit all in training. You also have to get the right material to the right people. In a smaller organization with a handful of small development teams, this may not be much of a problem. In larger organizations with many applications, thousands of developers, and dozens of test teams, it is highly unlikely that every developer, for example, requires exactly the same training. Even so, they should all start with the same foundational material, with the expectation that they will diverge and specialize later on.

Primary roles for which specialized training may be required include those who:

* Manage, fund, or create policy for any portion of the software development life cycle.
* Create business requirements for software.
* Translate business requirements into technical requirements.
* Create architectures and designs.
* Develop software.
* Automate processes.
* Create and execute testing strategies and tests.
* Perform security testing or auditing.
* Conduct data center operations.
* Respond to software security incidents.
* Have privileged access to software or data (for example, customer service representatives).

Even within a role, it is important to get the right people to the right courses. When advertising internally, for example, avoid generic descriptions, such as: “This course discusses threats and what to do about them.” This description does not grab the interest you desire and may attract or repel the wrong people. It’s better to clearly state something like: “This course shows architects and lead developers how to determine the attack surface of their code, assign threats to specific attack paths, and align security decisions with anticipated risk.” This description makes it much easier to get the right people to the right classes and helps to ensure the training program’s ultimate success.

## Computer-Based and Instructor-Led Training

E-learning (also known as *computer-based training*) is an excellent medium for building awareness. It can be inexpensively deployed and taken on demand. Once created, it is easy to maintain and republish, meaning you can keep it fresh, and students have ready access to the latest materials. E-learning can be delivered in chunks that do not consume too much time, and the technology can be used to push customized curriculums to specific individuals.

Instructor-led training is better for building day-to-day practitioner skills. In-person training provides the opportunity for hands-on exercises that look at real-world problems similar to those faced by the organization. It also provides the opportunity to develop social relationships between employees who may rarely see one another but discover that they are very dependent on each others’ work. In-person training, although it does take individuals away from their day-to-day duties for a brief time, allows students to ask an expert the questions that are most relevant to their immediate needs.

## Outsourced vs. In-House Training

Hiring external trainers who come with their own training material can quickly address a number of issues. It can free up internal SMEs to meet deadlines. It can also inject new ideas into the organization since these outside instructors likely have different experiences to relate. However, it is important to require that external instructors actually have experience with the organization’s needs.

On the other hand, internal instructors may be able to get on the task immediately. It may even be possible to start with no materials at all—simply put an internal SME at the whiteboard and let him or her talk. The information will probably be timely but may also be focused on one or two in-house issues. It will be sufficient for those who already “get it” and are interested but may not work for others who need guidance through their portion of the software security process. The problem gets larger when there are no local SMEs or when an expert in one area, like development, is trying to clarify issues for others in very different areas, like requirements or operations.

For many organizations, there is a natural evolution in training rollout. It typically evolves from in-house sessions, to small outsourced sessions, to broader in-house training, to role-based outsourcing, to more mature in-house curriculums. Of course, there are many factors involved in this evolution, including company size and immediacy of need.

When deciding whether to develop in-house materials, consider the following statistics on training development:[[4]](#footnote-4)

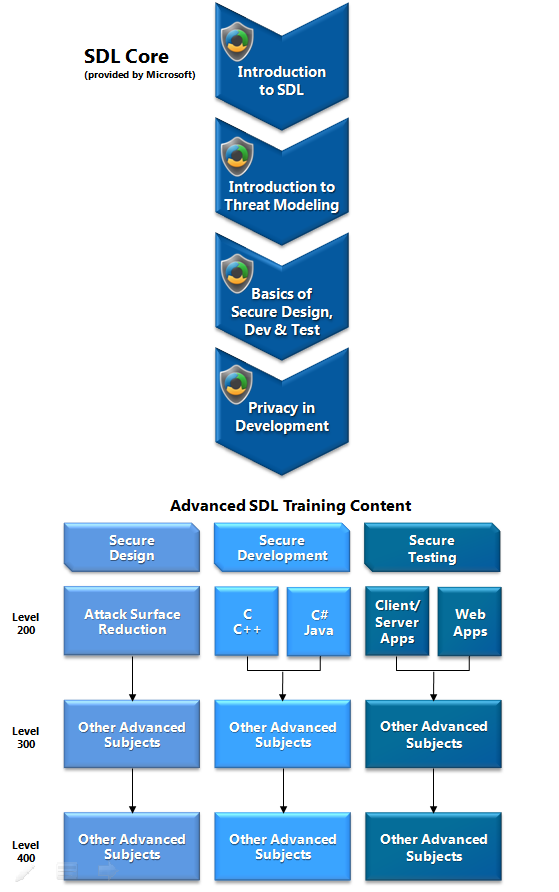
* One hundred minutes of total effort allocated for someone who understands the material to create each new technical content slide, from concept to "camera-ready" copy, where such effort includes planning, creation, review, editing, trial runs, and time with SMEs (this translates into about three days for about 20 content slides, which is about one hour of classroom material).
* Ten hours or more of combined effort to create 15 minutes of technical hands-on exercises.
* Fifty minutes of total effort allocated to repurpose (customize) an existing content slide. As an example, you might customize an existing slide discussing input validation to contain code and explanations from a particular development project.
* Three to four minutes, on average, to present each technical content slide, including discussion, anecdotes, and questions. This assumes you have well-constructed, single-topic slides.
* Lose two hours in a given eight-hour day: one hour for lunch and at least one hour for breaks and distractions (for example, starting up and shutting down hands-on exercises). Lose 15–30 minutes more if students must bring laptops with specific configurations.
* About 90–100 slides of technical content and group discussion (for example, a slide that asks, “How does this work in your organization?” or “Is it better to do A or do B?”) can be presented in an eight-hour day. Keep discussions brief—get two or three good answers and then move on.
* About seven-and-one-half working days (60 hours) of calendar time are required to complete 40 hours of content work. That is, there is often a 50-percent penalty in waiting for SMEs, scheduling meetings, and getting review comments, among other delays. If you are working on a three-day course, allow for a six-to-eight calendar week project.
* Fifteen to 20 technical, interactive students can typically be handled by one instructor. Given Web and video technologies, this number can be quite large if you are simply lecturing. If you have more than 15–20 interactive students, plan to have a second instructor help with logistics and hands-on exercises. If you are sending instructors on travel for more than a few days at a time (for example, for two weeks halfway around the world), send at least two.
* One hour is usually required to create, review, and complete each technical question on a course practical exam. Plan to have at least one question per important concept.

# Description of the Microsoft Security Development Lifecycle (SDL) Core Training Courses

The Microsoft [SDL core training courses](http://go.microsoft.com/?linkid=9708478) focus on areas that are the fundamental building blocks of the SDL process. All software project members must have at least some awareness of these areas and, depending on their role, need to have the ability to perform the activities themselves. The focus here is on activities that enable software development organizations to find and fix software security problems as early as possible in the development process, thereby reducing the overall cost related to security and, at the same time, delivering software that is more robust. Training courses covering the SDL core teach the students techniques for identifying the risks that the system may face from adversaries and then mitigating these risks through secure design, development, and testing. Figure 1 provides an example of the core and advanced SDL courses that should be taken to complete an effective course of study. It may be necessary to consider other courses to address specific needs, however training that covers these topics will provide a solid baseline of security knowledge.

The SDL Pro Network vendors regularly teach these concepts as part of a prescribed course of study, or as part of a customized curriculum to fit the needs of the student. They specialize in application security and have substantial experience and expertise with the methodology and technologies of the Microsoft SDL.

*Figure 1. Example of SDL core and advanced courses**available through the SDL Pro Network*

**

## [Introduction to the Microsoft Security Development Lifecycle (SDL)](http://download.microsoft.com/download/9/3/5/935520EC-D9E2-413E-BEA7-0B865A79B18C/Introduction%20to%20the%20Microsoft%20Security%20Development%20Lifecycle%20(SDL).ppsx)

This course introduces the Microsoft SDL, the software security assurance process developed by Microsoft. It seeks to illustrate the types of engineering and cultural changes necessary to create lasting change as well as the impact of effective security practices when used consistently. These practices and techniques are not unique to Microsoft – they are common sense security practices that have been around for years, presented in a logical framework.

## [Introduction to Microsoft Threat Modeling](http://download.microsoft.com/download/9/3/5/935520EC-D9E2-413E-BEA7-0B865A79B18C/Introduction_to_Threat_Modeling.ppsx)

Threat modeling is a key component of secure system design and is used to anticipate the attacks to which the system may be subjected. It is also used to help ensure that proper mitigations are in place to thwart or reduce the effectiveness of these attacks. People who are new to the concept of threat modeling often find it very difficult to get started, even after reading extensive documentation on the subject. If possible, it may be highly beneficial for individuals at an organization who may be involved in threat modeling to attend a class where the instructor explains what it is, how to perform it, and reinforces the concepts with hands-on exercises. This helps the students get over the first hurdle, putting them more at ease about starting to build their own. If this is not an option, working the tools using the supplied documentation may well provide the necessary boost to current capabilities.

In the threat modeling course, students are introduced to the SDL threat modeling process that includes thinking about security during requirements specification, creating appropriate system diagrams, identifying threats, identifying mitigations, and validation to ensure that the identified threats and mitigations are appropriate.

During the course, the instructor teaches the students to think about ways in which the system can be misused or abused (either by itself or after being induced by the attacker) while specifying software security requirements. The instructor further shows students the different types of security-relevant diagrams that are helpful towards evaluating the security posture of the system design. Students learn about the different diagram layers that can be used and techniques to validate completeness of the diagrams. Students then learn how to identify threats using the Microsoft Spoofing, Tampering, Repudiation, Integrity, Denial of Service, and Escalation of Privilege (STRIDE) model. Students are taught how to document how each of the above threats affects different system elements (for example, external entities, processes, data stores, and data flows) that were identified during previous threat modeling steps. Approaches to identifying mitigations for each of the enumerated threats are then discussed. Specific examples are given to make the students aware of what a good mitigation looks like. Finally, the instructor covers validation techniques that help to ensure that threats and corresponding mitigations are appropriate. All of these threat modeling process steps are reinforced through real-world examples and hands-on labs during the class.

## [Basics of Secure Design, Development, and Test](http://download.microsoft.com/download/9/3/5/935520EC-D9E2-413E-BEA7-0B865A79B18C/Basics%20of%20Secure%20Design%20Development%20Test.ppsx)

This class focuses on teaching students techniques for secure design and implementation, along with testing techniques that help to ensure the required security properties of the system are met. First, it is important to make sure that the mitigations compiled to address the threats identified in the threat model are properly designed, implemented, and tested. However, proactive software security goes much further than that. As security is an emergent property of the system, it is important to ascertain that the system’s design and implementation are created with security best practices in mind to make sure that doors are not opened for the attacker. Design and implementation of security features require special security attention, but security weaknesses that enable an exploit may reside in other areas of the system that are unrelated to the security features.

## [Privacy in Software Development](http://download.microsoft.com/download/9/3/5/935520EC-D9E2-413E-BEA7-0B865A79B18C/Privacy%20in%20Software%20Development.ppsx)

Privacy is an important subject that is related to and yet distinct from security. Privacy relates to how user information is collected and purposed, how user consent is granted, how user information is stored and handled, and any applicable statutes. Security controls may be used to provide protection mechanisms for the data, but privacy concerns establish the policy that drives the need for these security controls. In the privacy class, students learn about the multifaceted aspects of privacy best practices and learn to understand the data classification frameworks. Various case studies are presented to the students to teach them the proper privacy concerns for handling anonymous data, pseudo-anonymous data, PII, and sensitive PII.

In order to promote privacy, everyone in the organization must receive training on privacy best practices. Privacy violations often have nothing to do with technology, and so the controls are not merely technical in nature. It may be, at times, difficult to discern the correct course of action that promotes privacy. Therefore, instructor-led sessions that encourage discussions amongst students by examining concrete case studies provide the best mechanism for making sure that everyone in the organization is up to speed in terms of protecting customers and their data.

# Description of Advanced SDL Training Content – 200 Level

The courses in the 200 level build on the foundation in the SDL core training offering and go to the next level of detail and specificity in teaching students best practices for secure design, secure development, and security testing.

## Secure Design: Attack Surface Reduction

The larger the attack surface of the system, the harder the system is to secure. All of the entry points into the system through which data from the outside world can enter constitute the attack surface. Secure systems are designed in a way that minimizes their attack surface and makes explicit what data is expected from each entry point. Without this design goal in mind, poorly conceived interfaces with the outside world can cause a great deal of pain down the road and dramatically decrease the chances for secure implementation. Therefore, it is important for system architects and designers to learn about proper secure design techniques, such as attack surface reduction.

This course also introduces students to various types of vulnerabilities and attacks, and it provides advice on how to avoid them through good design. It also discusses the need to integrate threat model theory and data into secure-by-design approaches to software security. Key aspects of critical security features, such as cryptography, are also discussed.

## Secure Development: Secure Coding for Managed Code (C#/Java/.NET)

While managed code takes away a lot of the burden from the programmer that may otherwise lead to security problems (for example, memory management responsibilities), there are still many ways in which security issues can be introduced. For instance, a developer may not fully understand all of the security features provided by the language, causing the developer to misuse them in some way. Similarly, the security impact of using a particular API, or the way in which it should be used securely, may not be fully understood. The developer may also choose the wrong high-level structure to perform the needed operations, leading to problems. For instance, safe synchronization in multithreaded environments or proper usage of cryptography APIs may not be fully understood.

It is also important to note that people often use managed code to solve different problems than the ones for which they use native code. Given that the problem space is different, so are the potential vulnerabilities, and it is important for developers switching to managed code to get specialized training.

Secure development courses, with emphasis on secure coding for managed code, focus on the types of issues that developers writing code in C#, Java, or .NET may experience. Instructors show the students the typical programming pitfalls that may occur, along with guidance on how to avoid them. Students are invited to identify the security problems in sample code and are then asked to rewrite the sample code in a secure fashion. This kind of collaborative instruction environment helps the students internalize the concepts in the course.

## Secure Development: Secure Coding for Native Code (C/C++)

There is a wide range of programming errors that can lead to dangerous security consequences in unmanaged languages, such as C and C++. Since a lot of low-level housekeeping tasks fall on the shoulders of the developers, numerous opportunities exist to make simple mistakes that have security consequences. With code written in these languages, developers are given access to many low-level constructs, most notably the ability to directly access and manipulate memory through pointers. With this added power comes a great deal of responsibility. Further, many of the protections available in managed code languages (for example, type safety) are not available in C and C++. The complexity is often such that even experienced developers are likely to make mistakes that may be introduce exploitable security conditions into the code.

In this class, students learn many secure programming techniques for C and C++, including indentifying various types of vulnerabilities and determining how to avoid them. The course includes details on issues highlighted in SDL core courses, such as overflow problems, pointers, and memory usage. This course also addresses related secure programming concepts, such as taking advantage of available threat model information and understanding how to reason about fixing all instances of a problem—not just the one discovered most recently.

## Security Testing

Security testing is a key component of any secure development process. The security testing should accomplish several goals:

* Ensure that non-functional and functional security requirements from the requirements specification document are met.
* Execute misuse and abuse cases defined for the system, and make sure that the system resists them.
* Traverse the attacks identified in the system’s threat model, and ensure that the countermeasures have been properly implemented to resist these attacks.
* Focus on areas of security risk (as identified in the threat model), and ensure that everything works as expected.
* Understand various types of vulnerabilities and how to find them.
* Identify common automated methods for security testing.
* Understand how to discuss issues discovered from a “risk” perspective so that others can take the appropriate action.

Security testing is difficult because it is not a core competency for most QA personnel. In most cases, QA is constrained to functional testing and may not have the time or the skill to consider the negative use cases or ways in which the system can be abused. Training is needed to ensure that security testing capabilities are developed within the organization, which often requires the ability to both understand the business requirements and think like an attacker.

# Description of Advanced SDL Training Content – 300 and 400 Level

Once the students have completed the SDL core and the 200-level training courses, they are ready for the 300 and 400-level courses to further hone their security skills while covering some of the more advanced security topics. Although all of the designers, developers, and testers should go through training up to the 200 level, a smaller number of employees may need to go to the 300 and 400-level training. Usually these employees will become security SMEs on their projects. These may also be people who are working on software projects where security concerns are of paramount importance.

## Advanced Secure Design

This course introduces students to various secure design patterns that can be called into service to tackle common problems. Rather than reinventing the wheel each time, the students learn how to apply proven design patterns to solve certain problems in a way that promotes security. The students analyze various designs, identify the strengths and weaknesses in each design from a security perspective, and then develop an understanding of the design solution that promotes the most security and the context under which it is applicable.

This course also focuses on designing secure solutions using several popular technologies and understanding the intricacies associated with each technology that may have security implications. This positions the students to serve as SMEs on their respective projects when evaluating design selection and application of particular technology.

## Advanced Secure Development

In this course, students build on the concepts covered in the 200-level course to enhance their understanding of secure implementation when specific frameworks are used. The students learn about the subtleties of each framework and understand the intricacies involved with using each framework securely. This knowledge positions the students to be SMEs on their teams when particular security technologies are used. Essentially, the students move beyond the security concerns associated with programming using the core language elements, and they extend their knowledge into application of specific technologies.

Students in this class are also trained on researching and understanding the different software attack patterns, and then they go through the process of creating relevant coding standards for the organization that makes the organization’s code resist the attacks. Since new attack techniques emerge continually, it is important for the top security experts in the organization to understand these attacks and to create standards, best practices, policies, and procedures for counteracting these attacks.

## Advanced Security Testing

In this course, students build on the concepts covered in the 200-level course for a deeper understanding of security testing techniques. The students are taught how to further step into the attacker’s role and conceive other attacks on the system. For instance, the students are encouraged to review the various attack patterns documented under Common Attack Pattern Enumeration and Classification (CAPEC), pick out the patterns relevant to the system under consideration, and determine whether the design and implementation of the system properly resist these types of attacks. The students also learn how to synthesize different attack techniques, as an attacker would, to try and achieve a single higher goal.

# Description of Ancillary Courses

Ancillary courses provide training on other security topics within the software development life cycle. In addition, these courses focus on non-development staff, such as software development managers and operations team members.

## Security Response and Incident Management

Despite best efforts on building secure software from the start, it is inevitable that some issues will be missed. These issues may contribute to a security-related incident in production. When that happens, the operations teams steps in to provide an initial security response that addresses the issue with a temporary solution. In the longer term, the development team likely needs to create a complete fix for the weakness that enabled the compromise. An important part of the overall process is to learn from each incident to understand why the problem crept into production in the first place. Was it a missing coding standard? Was it the result of insufficient training? Once root causes are identified, they need to be rectified. In this course, students learn some security response techniques along with the best practices for effective incident management.

## Security for Managers

Security of the software product does not start with the developer—it starts with the manager. Lack of managerial commitment to secure product development and delivery seriously weakens the security initiative. Managers must plan for integration of software security activities within the software development process adopted by their development organization. That means managers need to be able to include a security-aware software development life cycle in their project planning, budget for these activities (both in terms of dollars and in terms of time), and ensure that personnel with sufficient levels of software security expertise are present on their teams. Managers are also responsible for ensuring that an appropriate level of security training is achieved within their development teams, including the ability to estimate vulnerability repair costs and understanding the role of security response functions. Managers must be introduced to these concepts to be effective security champions.

## Security Tools

Students should endeavor to learn about the different tools they should use throughout the secure software development process to increase the chances of delivering a secure application. Tools can assist in threat modeling activities, source code analysis, dynamic black-box analysis of the system, and fuzzing tools. It is important to find training that explains how and when each of these tools should be used, who should operate them, what should be done with the findings, and how these tools can be customized to improve their utility.

# Conclusion

The vulnerability numbers and data breaches speak for themselves—software security is not going to happen without concerted effort. Security-aware software development practices come as a large transition for many organizations, and training is often a key boost needed to move processes and practices in the right direction.

Training also provides a great avenue for employee growth, whether it directly contributes to career progression or results in internal “belts” or certifications that show the benefits of investment in time and effort. Given all this, organizations must seriously consider role-based training as a wise investment in future software security success.

# Appendix A: Resources

The following URLs are embedded within the document for ease of use by online readers. A listing of those URLs is provided for the convenience of those reading a printed copy of the article.

* Microsoft Security Development Lifecycle   
  <http://www.microsoft.com/sdl>
* United States Computer Emergency Response Team (CERT)  
  <http://www.cert.org/stats/>
* National Vulnerability Database  
  <http://nvd.nist.gov/home.cfm>
* Common Vulnerabilities and Exposures Database  
  <http://cve.mitre.org/>
* Common Weakness Enumeration program  
  <http://cwe.mitre.org/>
* Open Security Foundation DataLoss Database  
  <http://datalossdb.org/>
* Privacy Rights Clearinghouse Chronology of Data Breaches  
  <http://www.privacyrights.org/ar/ChronDataBreaches.htm>
* Web Application Security Consortium (WASC) Web Hacking Incidents Database  
  <http://whid.xiom.com/>
* Ponemon Institute Press Release: Study Shows Data Breach Costs Continue to Rise  
  <http://www.pgp.com/insight/newsroom/press_releases/2008_annual_study_cost_of_data_breach.html>
* 2006 CompTIA survey  
  <http://www.awareity.com/public/Awareity-Validation-CompTIA.pdf>
* The Economic Impacts of Inadequate Infrastructure for Software Testing  
  <http://www.nist.gov/director/prog-ofc/report02-3.pdf>
* Depository Trust and Clearing Corporation (DTCC)  
  <http://searchsecurity.techtarget.com/magOnline/0,,sid14_gci1346790,00.html>
* Building Security in Maturity Model  
  <http://bsi-mm.com>

1. Source: <http://www.boston.com/business/globe/articles/2007/08/15/cost_of_data_breach_at_tjx_soars_to_256m/> [↑](#footnote-ref-1)
2. Source: <http://www.boston.com/news/local/articles/2008/03/28/advanced_tactic_targeted_grocer/> [↑](#footnote-ref-2)
3. Source: <https://www.sans.org/newsletters/newsbites/newsbites.php?vol=11&issue=6#sID309> [↑](#footnote-ref-3)
4. Cigital, a member of the Microsoft SDL Pro Network, provided these statistics. The organization derived these averages from the past 10 years of software security and quality training engagements. [↑](#footnote-ref-4)