


Advancing Your Application Security Program by Putting the OWASP Top Ten into Practice



A DEVELOPER'S PERSPECTIVE

Table of Contents

Introduction	3
What is the OWASP Top 10 List?.....	4
How to compile detailed information about application security best practices.....	5
Information from OWASP	5
Other Resources – An Example (TeamMentor)	5
How the OWASP Top 10 and other resources can be integrated into the SDLC.....	7
1. Requirements and analysis	7
2. Architecture and design.....	9
3. Development	9
4. Testing	10
5. Deployment	11
6. Maintenance.....	11
How to justify investment in security and demonstrate progress to management	12
Lower Direct Costs.....	12
A Widely Accepted Standard	12
A Compliance Requirement.....	12
A Source of Metrics	13
Conclusion	13
Tools to accelerate the adoption of best practices related to the OWASP Top 10	14
To Learn More:	14
Appendix	15
Appendix I: OWASP Top 10 Application Security Risks – 2010	15
Appendix II: Key Links on the OWASP Web Site	16
Notes.....	17

Introduction

Software developers and architects face a few major challenges related to application security:

- Understanding the complexities of security threats and vulnerabilities and the defenses needed to combat them.
- Developing policies, practices and training so application development teams can include application security consistently in design, coding and testing.
- Improving the collaborative efforts between the security organization and development teams in understanding what it takes to meet the performance and security goals respectively.
- Justifying to management why time and funds should be invested in application security so that software is delivered more efficiently and securely.

Many organizations have found that the OWASP Top 10 list of critical security risks can help in these three areas.

First, a great deal of technical information about key application security risks and counter-measures is available from OWASP project and other organizations.

Second, the OWASP Top 10 list can be used at each stage of the software development life cycle (SDLC) to strengthen design, coding and testing practices.

Third, the OWASP Top 10 list can be used to justify security activities to management, and to show progress over time toward industry standard security and compliance.

In this technologist's briefing we will discuss:

- What is the OWASP Top 10 List?
- How to use the OWASP Top 10 and other security resources to compile detailed information about threats and application security best practices.
- How the OWASP Top 10 and other key resources can be integrated into the SDLC.
- How the OWASP Top 10 can help justify investment in security.
- Tools to accelerate the adoption of best practices related to the OWASP Top 10.

What is the OWASP Top 10 List?

The Open Web Application Security Project (OWASP) is a worldwide free and open community focused on improving the security of application software. It currently supports more than 40 projects related to application security risks, secure coding practices, and integrating security into the software development life cycle.ⁱ

Since 2003, the OWASP Top 10 project has published a list of the ten most critical web application security risks. This list represents a consensus among many of the world's leading information security experts about the greatest risks, based on both the frequency of the attacks and the magnitude of their impact on businesses.

The list includes very well-known risks like injection and cross-site scripting attacks (A1 and A2 on the list), and slightly less prominent issues such as insecure direct object references and insufficient layer protection (A4 and A9). Short descriptions of each of the OWASP Top 10 are included in the appendix.

The OWASP Top 10 materials provide very detailed information on vulnerabilities and concrete suggestions on how to protect web applications and data.ⁱⁱ This contrasts with many government regulations and industry standards, which describe high-level requirements but provide little prescriptive guidance on how to deploy specific defenses.

In addition, the OWASP Top 10 is also noteworthy in that it has been adopted or referenced by a large number of government agencies, industry standards bodies, prominent information technology vendors, and leading-edge IT organizations, as shown in the table on this page.

Organizations Adopting or Referencing the OWASP Top 10 List		
<i>Government Agencies and Industry Standards Bodies (Partial list)</i>	<i>Technology Vendors and Corporate IT Groups (Partial list)</i>	
Center for Internet Security (CIS)	A.G. Edwards	Microsoft
Cloud Security Alliance (CSA)	British Telecom	National Australia Bank
Defense Information Systems Agency (DISA)	Bureau of Alcohol, Tobacco, and Firearms (ATF)	Oracle
European Network and Information Security Agency (ENISA)	Citibank	Price Waterhouse Coopers
Federal Financial Institutions Examination Council (FFIEC)	Citrix	REI
National Institute of Standards and Technology (NIST)	HP	Samsung SDS (Korea)
Payment Card Industry Security Standards Council (PCI SSC)	IBM Global Services	Sprint
SANS Institute	Michigan State University	Symantec
		The Hartford

How to compile detailed information about application security best practices

There are three reasons for using the OWASP Top 10 as a focus for information security education:

- Addressing the 10 will dramatically improve your organization's risk profile, since they represent the most prevalent and serious risks for web-facing applications.
- The practices and techniques that address the OWASP Top 10 will help you address many critical software vulnerabilities across different environments.
- A lot of very useful information is available both from the OWASP project and from third parties.

Here we will look at the kind of information that is available both from OWASP and other sources.

INFORMATION FROM OWASP

The OWASP web site provides both overview information on the Top 10 as a group, and excellent "drill-down" materials on each risk and recommended counter-measures.

For example, an overview slide presentation includes three slides on each risk: a description, an illustration (usually with a diagram), and a set of high-level recommendations. This slide presentation is available at:

http://owasptop10.googlecode.com/files/OWASP_Top_10_-_2010%20Presentation.pptx

The OWASP team then provides detail on each individual risk, including:

- A reference page.
- A "cheat sheet."
- A set of related articles and references.

Links to these materials are included in the appendix.

OTHER RESOURCES – AN EXAMPLE (TEAMMENTOR)

Because the OWASP Top 10 list is so widely accepted, several third parties have created additional resources to build on the work done by the OWASP team. We are going to look briefly at one example, the TeamMentor knowledgebase from Security Innovation.

TeamMentor contains over 3,500 guidance "assets" such as vulnerability descriptions, checklists, design patterns and code samples.

The TeamMentor knowledgebase can be searched using a variety of criteria, including each of the OWASP Top 10 risks. For example, if you are interested in Injection flaws, you can drill down into 50 relevant topic areas related to injection attacks, best practices for validating input, controlling access to the database server, etc., with topics classified by technology (Java, ASP.NET) and phase of the software development lifecycle (design, implementation, test). (Figure 1)

Now showing 1 - 30 of 111 | << < 1 2 3 4 > >> |

Title	Technology	Phase	Type	Category
+ AJAX Injection Attack	Any	Implementation	Attack	Input and Data Validation
+ All Data Passed Between Native and Java Code is Validated	Java	Implementation	Checklist Item	Unmanaged Code
+ All Database Input is Validated	ASP.NET 3.5	Implementation	Checklist Item	Input and Data Validation
+ All database input is validated	ASP.NET 4.0	Implementation	Checklist Item	Input and Data Validation
+ All Database Input is Validated	Java	Implementation	Checklist Item	Data Access
+ All input is validated	ASP.NET 4.0	Implementation	Checklist Item	Input and Data Validation
+ All Input Parameters are Validated for Length, Range, Format, and Type	ASP.NET 3.5	Implementation	Checklist Item	Parameter Manipulation
+ All the Input is Validated For Length, Range, Format, and Type	ASP.NET 2.0	Implementation	Checklist Item	Input and Data Validation
+ All the Input is Validated For Length, Range, Format, and Type	ASP.NET 3.5	Implementation	Checklist Item	Input and Data Validation
+ All the Input Parameters are Validated For Length, Range, Format, and Type	ASP.NET 2.0	Implementation	Checklist Item	Input and Data Validation
+ All untrusted input is validated inside data access methods.	ADO.NET 2.0	Implementation	Checklist Item	Input and Data Validation
+ Allow Only Trusted Hosts to Connect to the Database Server	ASP.NET 4.0	Design	Guideline	Data Access
+ Application Does Not Rely Only on Client-Side Validation	ASP.NET 2.0	Implementation	Checklist Item	Input and Data Validation
+ Application Does Not Rely Only on Client-Side Validation	ASP.NET 3.5	Implementation	Checklist Item	Input and Data Validation
+ Application Does Not Rely Only on Client-Side Validation	Java	Design	Checklist Item	Input and Data Validation

Figure 1: The TeamMentor reference guide is a third party tool that provides an in-depth selection of security best practices for each of the OWASP Top 10 risks.

A typical best practice topic, say *All the Input is Validated For Length, Range, Format, and Type*, might include sections like “What to Check For,” “Why,” “How to Check,” “How to Fix” (including a code sample), “Problem Example,” and “Solution Example.”

This is an example of how an industry-wide focus on the OWASP Top 10 has produced an exceptionally deep range of educational resources and guidelines covering application security.

TeamMentor OWASP Edition is available at no charge at <http://owasp.securityinnovation.com/>. It includes a subset of the complete TeamMentor library. (TeamMentor Enterprise Edition contains the 3,500+ guidance assets)

How the OWASP Top 10 and other resources can be integrated into the SDLC

Application security is not only about how to write secure code. One study of security defects in web-based applications found that seventy percent of the defects were due to design flaws, most of which could have been fixed in the application design phase.ⁱⁱⁱ Analysts also point to the importance of systematic testing for security flaws.^{iv}

However, it is not easy to implement security best practices across an entire software development lifecycle (SDLC). The lifecycle of the typical mission-critical web applications is complex, involving dozens of people and hundreds of tasks. What threats are the most serious? How can design choices reduce vulnerabilities? What secure coding practices are most important, and who needs to learn them? What testing is required to uncover hidden vulnerabilities? What deployment errors can create security holes?

The OWASP Top 10 list and related resources can be used to address these issues and questions. As shown in **Figure 2**, these tools can be used at each phase of the SDLC to guide analysis and action.

Phase	OWASP Top 10 Use
Requirements and Analysis	Threat Modeling: use Top 10 as guide to potential attacks. Determine countermeasures.
Architecture and Design	Security Design Guidelines: Adopt design guidelines that will harden applications against Top 10.
Development	Adopt coding standards to counter Top 10. Search for Top 10 code reviews .
Testing	Develop test plans for Top 10. Test for Top 10 with static analysis tools . Scan for Top 10 with web scanning tools .
Deployment	Check for configuration and physical deployment errors related to Top 10.
Maintenance	Conduct ongoing scanning for Top 10.

Figure 2: The OWASP Top 10 can play a role in each phase of a secure development life cycle.

In fact, the advantage of using the OWASP materials in this way extends beyond protecting against 10 specific risks. Organizations that put in place the people, tools and processes to protect against the OWASP Top 10 risks will develop first-class application security programs capable of handling a wide range of web-based threats.

We will now look at how the OWASP Top 10 list and TeamMentor (an example of a security tool that builds on the OWASP Top 10) can be used with each phase of the SDLC.

1. Requirements and analysis

In the **Requirements and Analysis** phase, analysts consider the requirements and goals of the application, as well as possible problems and constraints. Part of this process involves threat modeling, which identifies threats and vulnerabilities relevant to the application.

The OWASP Top 10 can be used as guide to potential attacks. A thorough examination of which of those 10 risks could affect the software will suggest ways the application design can be shaped to achieve security objectives, and where resources could be applied to develop countermeasures.

Figure 3 shows the types of questions that should be addressed at this stage.

Objective Category	Questions to ask
Tangible assets to protect	<ul style="list-style-type: none"> • Are there user accounts, passwords, confidential information, intellectual property, etc to protect? • Can this system be used as a conduit to access other corporate assets?
Intangible assets to protect	<ul style="list-style-type: none"> • Is there potential for an attack that may be embarrassing, although not otherwise damaging?
Compliance requirements	<ul style="list-style-type: none"> • Are there corporate security policies or standards that must be adhered to? • Are there security or privacy legislations you must comply with?
Quality of service requirements	<ul style="list-style-type: none"> • Are there specific availability or performance requirements you must meet?

Figure 4: Questions to ask in the *Requirements and Analysis* phase of the SDLC.

Related resources like TeamMentor can take this process even further. For example, TeamMentor includes guidelines for “Identify Security Objectives” and a detailed “how to” article “Create a Threat Model.” The latter provides concrete suggestions for tasks like identifying security objectives, decomposing a web application and identifying vulnerabilities (Figure 4).



Create a Threat Model

Summary

This How To describes an approach for creating a threat model for a Web application. The threat modeling activity helps you to model your security design so that you can expose potential security design flaws and vulnerabilities before you invest significant time or resources in a flawed design and/or problems become difficult to reverse. This How To provides prioritized vulnerability categories and a threat list to make the threat modeling activity easier.

Contents

- Objectives
- Overview
- Before You Begin
- Input
- Output
- Summary of Steps
- Step 1: Identify Security Objectives
- Step 2: Create an Application Overview
- Step 3: Decompose Your Application
- Step 4: Identify Threats
- Step 5: Identify Vulnerabilities
- What to Do Next
- Agile Considerations
- MSF Agile Considerations
- Additional Resources

Figure 3: A TeamMentor “how to” topic on creating a threat model.

2. Architecture and design

In the **Architecture and Design** phase, specific design guidelines can be adopted that are proven solutions to the Top 10 risks.

Figure 5 illustrates the kind of design guidelines that are actionable and relevant to the OWASP Top 10 risks, and that can have wide-ranging impact on the application.

Category	Guidelines
Input / Data Validation	Do not trust input; consider centralized input validation. Do not rely on client-side validation. Be careful with canonicalization issues.
Authentication	Use strong passwords. Support password expiration periods and account disablement. Do not store credentials (use one-way hashes with salt).
Authorization	Use least privileged accounts. Consider authorization granularity. Enforce separation of privileges. Restrict user access to system-level resources.
Configuration Management	Use least privileged process and service accounts. Don't store credentials in clear text. Don't use Local Security Authority (LSA).
Sensitive Data	Avoid storing secrets. Secure the communication channel. Provide strong access controls for sensitive data stores.
Cryptography	Do not develop your own. Use proven and tested platform features. Keep unencrypted data close to the algorithm. Cycle your keys periodically. Avoid key management (use DPAPI).
Exception Management	Use structured exception handling. Do not reveal sensitive application implementation details. Consider a centralized exception management framework.
Auditing and Logging	Identify malicious behavior. Know what good traffic looks like. Audit and log activity through all application tiers. Secure access to log files

Figure 5: Examples of actionable, relevant and impactful design guidelines.

TeamMentor includes topics that can help in this phase such as guidelines for conducting a security architecture and design review and a checklist on applying security design guidelines.

3. Development

In the **Development** phase, specific coding standards that have been proven to defend against the Top 10 risks can be adopted.

To use our injection flaw example again, developers could be required to use an interface that supports bind variables (e.g. prepared statements or stored procedures), to always perform “white list” input validation on all user-supplied input, and to always minimize database privileges.

Again, related resources can be used to “drill down” into these best practices. For example, if a developer is creating a web input form using Java, TeamMentor provides a checklist for validating database input, with information on topics like identifying database entry points, identifying validation routines, and using parameterized queries (**Figure 6**). Additional articles are included for forms written using ASP.NET 3.5 and ASP.NET 4.0, with different code examples. Other articles elaborate on details, including topics like constraining and sanitizing input, client side and centralized validation, untrusted file name and path input, parameter and schema validation, and many others.

Code reviews are another activity that typically occurs during the Development phase. Code reviews can be performed on every check-in, on every build, or on some other regular interval. Developers must review code not only to make sure that it has the features and functions described in the specification, but also to uncover vulnerabilities in the code related to the OWASP Top 10 and other security issues. This usually requires security training for the entire development staff, as well as a focused approach using checklists.

OWASP has a code review project that published a useful guide on this topic:
https://www.owasp.org/index.php/OWASP_Code_Review_Guide_Table_of_Contents.

Static analysis tools can make a major contribution to code reviews, because they can find common coding errors much faster than manual code reviews by humans. They can be programmed to look for clues that applications may be vulnerable to Top 10 risks. Sometimes significant effort is required to review and reject false positives, but on balance the static analysis tools can find simpler problems faster, freeing up human reviewers to focus on more sophisticated vulnerabilities.

4. Testing

When the quality assurance group builds the test plan, it should ensure that specific tests are run to simulate attacks related to the Top 10 risks. The threat modeling activities performed during the Requirements and Analysis phase of the SDLC can be very useful.

In addition, security testing requires a mindset that is new to many QA people, as well as developers. Instead of measuring software against functional specifications, developers and testers need to constantly probe for what the application is *not* supposed to do. They must learn how hackers and cybercriminals attack, and look for subtle signs of successful attacks, such as unexpected changes to file systems and unusual network traffic.

If people are trained on how to detect attacks related to the OWASP Top 10 risks, they will also gain the skills and mindset needed to prevent other types of attacks.

Web scanners and penetration testing can play a very useful role during the test phase of the SDLC. Web scanners can identify common vulnerabilities quickly, and can discover large amounts of information about devices, such as misconfigurations, exposed usernames and passwords, vulnerable scripts, hidden fields on forms, and potentially vulnerable helper files, Java applets and ActiveX controls.

How to Check

Use the following steps to check if all input passed to database is validated:

1. **Identify database entry points.** Identify all locations within your application that interface with the database servers. Potential sources of input include:
 - URL based parameters
 - Form based parameters
 - Hidden fields
 - Cookies
 - Local filesystem
 - Javascript variables
2. **Trace data from source to sink.** Trace each source of input from the immediate source, through your application and to the backend database.
3. **Identify validation routines.** Each input source should have a validation routine associated with it. Ideally the validation will occur as soon as the input reaches your application. Shared validation routines are better than creating many throughout the code base, so check for consolidation of routines to aid testing and reduce the chance of one-off bugs. If any input source does not have a validation routine associated with it, flag it for fixing.
4. **Ensure quality of validators.** Validation routines should check for length, range, format and type. The validator should first check for good data via whitelisting and then for known malicious data via blacklisting. Do not rely on client-side validation alone as it can be bypassed easily. For more information see Checklist Item: Input is Validated for Length, Range, Format and Type.
5. **Ensure that parameterized queries are used.** Using the Type Safe SQL Parameters are Used checklist, verify that all database transactions are handled using

Figure 6: A TeamMentor check list with best practices for validating data input on Java forms.

Organizations should never rely entirely on penetration testing, because by the time an application is stable enough for penetration testing the cost of fixing bugs is going to be very high. Nevertheless, penetration testing based on the OWASP Top 10 risks can give you confidence that earlier activities, such as threat modeling, design reviews, and code reviews, have successfully hardened the software against attack.

But testing tools must be complimented with training and guidance tools, so that the team will have the knowledge to interpret the results of the tools and ultimately move to the remediation phase, where for example, if vulnerabilities have been identified through a false-positive from a static analysis tool. From there, prioritization is critical so that the critical flaws and vulnerabilities that pose a major business risk are addressed first.

5. Deployment

Even the most securely designed and coded application can be compromised by an error during deployment. Organizations need to validate that their patching, updating, logging and auditing procedures are adequate to prevent and detect likely attacks, that account privileges and file permissions are set with due consideration for security, and that systems are as locked down as possible in terms of ports, protocols and services.

Again, using the OWASP Top 10 list as a benchmark is an excellent way to create specific policies and procedures.

Resources like TeamMentor can also help, for example by providing a deployment review checklist with suggestions about server security categories that should be reviewed during deployment (accounts, auditing, ports, protocols, registry, etc.).

6. Maintenance

Finally, in the **Maintenance** phase of the life cycle, a focus on the OWASP Top 10 will ensure that organizations conduct ongoing reviews and code scanning, to find out if changes to the application over time might have created any new vulnerabilities.



How to justify investment in security and demonstrate progress to management

Technologists know that application security is absolutely vital, but it is often difficult to show management that any particular expenditure on security is justified, and to demonstrate progress toward improving security. The OWASP Top 10 list can be very useful in these areas.

LOWER DIRECT COSTS

Improving application security can lower:

- Bug fixing and retesting costs, because security-related bugs are avoided through better training and practices.
- Vulnerability remediation costs, because security-related bugs and missing security features are identified early in the design and development process, where remediation is less costly than in the final stages.
- Management costs, because following best practices make projects more efficient and more predictable.
- Security incident and data breach costs, because vulnerabilities are eliminated and risks are reduced.
- Compliance costs, because it is easier to show auditors that requirements and best practices are being followed, and because more activities and events will be logged.

A WIDELY ACCEPTED STANDARD

When management asks “How much security do we really need?” (translation: “What is the least amount we can get away with?”) A good answer is: “At least as much as the average company in our industry.” After all, if a major security breach does occur, organizations that can be shown to be behind their typical competitors in security are extremely vulnerable to criticism and bad publicity.

Because so many organizations have embraced the OWASP Top 10 list as a benchmark, often it can be used as a proxy for reasonable, prudent industry standards. In many industries failing to address the OWASP Top 10 might be viewed as neglecting to protect against foreseeable risks.

A COMPLIANCE REQUIREMENT

The OWASP Top 10 is being referenced by an increasing number of industry standards and government regulations.

Most notably, PCI DSS requirements 6.1 to 6.9 map directly to 8 of the OWASP Top 10, as shown in **Figure 7**.

In addition, the standards bodies referencing the OWASP Top 10 as a useful set of guidelines include the Center for Internet Security (CIS), the Cloud Security Alliance (CSA), the Defense Information Systems Agency (DISA), the European Network and Information Security Agency (ENISA), the Federal Financial Institutions Examination

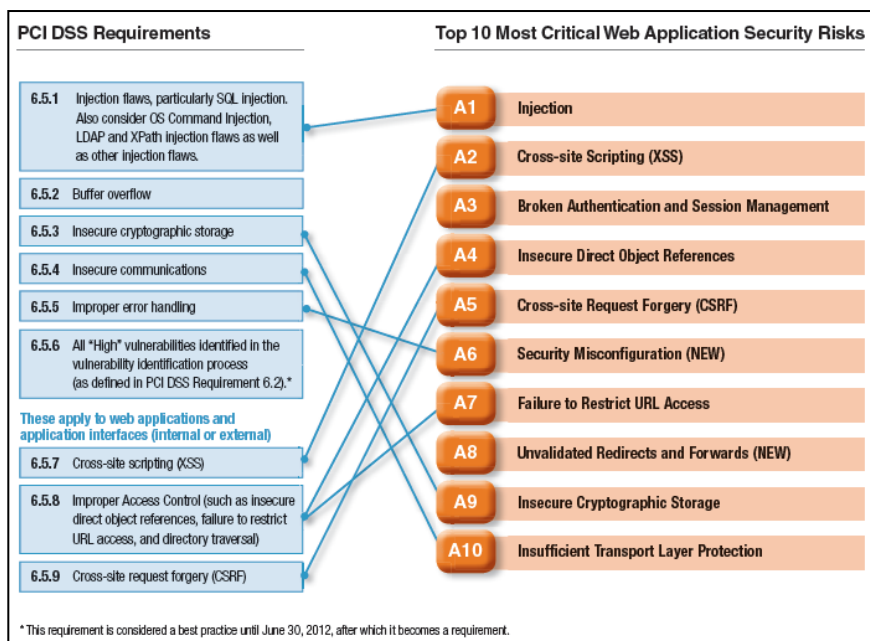


Figure 7: PCI DSS requirements map to 8 of the Top 10 risks.

Council (FFIEC), the National Institute of Standards and Technology (NIST), and many others.

Essentially, auditors are likely to view the failure to address the OWASP Top 10 as a sign that an organization is falling short of compliance with many standards, while integrating the Top 10 into the software development life cycle shows them that many best practices have been implemented.

A SOURCE OF METRICS

Security improvements are normally extremely difficult to quantify. The OWASP Top 10 offers a means to mark progress. Metrics can be developed in the form of X% or personnel trained, or Y% of new applications developed with appropriate policies, procedures and tools, or programs put in place to address so many of the Top 10 this year and so many next year.

While these are admittedly crude forms of measurement, they can at least give management an idea of concrete progress being made. However, it also depends on the goals that an organization sets out to achieve. For example, in implementing an application security training program, it might be easier up front to focus on improving technical competence relative to security as opposed to changing the culture of the organization's views toward security, which is a longer-term initiative.

Conclusion

At the beginning of this paper we discussed the major challenges related to application security facing developers. We have now seen how the OWASP Top 10 list and related resources can help address these by:

- Providing detailed information on the complexities of security threats and vulnerabilities, and on technologies and best practices to combat them.
- Helping organizations develop policies, practices and training that can be applied across the entire software development lifecycle (SDLC).
- Contributing ROI and compliance justifications to management for investment in application security.
- Ensuring that security and development teams are aligned in their approach to building software that addresses business needs in a secure way without limiting performance.

In short, although application security is a complex subject, an approach based on the OWASP Top 10 list can give structure and coherence to application security programs and help organizations demonstrate progress toward reducing business risks.

Tools to accelerate the adoption of best practices related to the OWASP Top 10

Security Innovation provides products, training and consulting services to help organizations build and deploy secure software, and also to implement a best practices model based on the OWASP top 10.

These offerings include:

- **TeamMentor** is a real-time reference guide, the industry's largest repository of secure software development knowledge, acts as the perfect "In-Practice" reference guide for novice and advanced developers and designers, architects, project managers and security teams.
- **TeamProfessor** computer-based training, including courses like "OWASP Top Ten: Threats and Mitigations," "How to Test for the OWASP Top Ten," and many courses on secure coding practices for ASP.Net, Java, C++, Windows and other development environments.
- **Consulting services** to assess application risk across the entire application portfolio, how to implement a secure software development life cycle, including SDLC assessment and optimization, code reviews, threat modeling and penetration testing.

To Learn More:

For more information, please visit Security Innovation's web site at

<http://www.securityinnovation.com/solutions/by-regulation-standard/owasp.html>

To evaluate the company's eLearning products, please contact us at

+ 1.877.694.1008 x1 or getsecure@securityinnovation.com.

Appendix

APPENDIX I: OWASP TOP 10 APPLICATION SECURITY RISKS – 2010

A1 - Injection	<ul style="list-style-type: none">• Injection flaws, such as SQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing unauthorized data.
A2 - Cross-Site Scripting (XSS)	<ul style="list-style-type: none">• XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation and escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
A3 - Broken Authentication and Session Management	<ul style="list-style-type: none">• Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, session tokens, or exploit other implementation flaws to assume other users' identities.
A4 - Insecure Direct Object References	<ul style="list-style-type: none">• A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.
A5 - Cross-Site Request Forgery (CSRF)	<ul style="list-style-type: none">• A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.
A6 - Security Misconfiguration	<ul style="list-style-type: none">• Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. All these settings should be defined, implemented, and maintained as many are not shipped with secure defaults. This includes keeping all software up to date, including all code libraries used by the application.
A7 - Insecure Cryptographic Storage	<ul style="list-style-type: none">• Many web applications do not properly protect sensitive data, such as credit cards, SSNs, and authentication credentials, with appropriate encryption or hashing. Attackers may steal or modify such weakly-protected data to conduct identity theft, credit card fraud, or other crimes.
A8 - Failure to Restrict URL Access	<ul style="list-style-type: none">• Many web applications check URL access rights before rendering protected links and buttons. However, applications need to perform similar access control checks each time these pages are accessed, or attackers will be able to forge URLs to access these hidden pages anyway.
A9 - Insufficient Transport Layer Protection	<ul style="list-style-type: none">• Applications frequently fail to authenticate, encrypt, and protect the confidentiality and integrity of sensitive network traffic. When they do, they sometimes support weak algorithms, use expired or invalid certificates, or do not use them correctly.
A10 - Unvalidated Redirects and Forwards	<ul style="list-style-type: none">• Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

APPENDIX II: KEY LINKS ON THE OWASP WEB SITE

OWASP Top 10 overview document <http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202010.pdf>

Overview slide presentation

http://owasptop10.googlecode.com/files/OWASP_Top_10_-_2010%20Presentation.pptx

A1 Injection

Reference page: https://www.owasp.org/index.php/Top_10_2010-A1

Cheat Sheet: https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

A2 Cross-site Scripting (XSS)

Reference page: [https://www.owasp.org/index.php/Top_10_2010-A2-Cross-Site_Scripting_\(XSS\)](https://www.owasp.org/index.php/Top_10_2010-A2-Cross-Site_Scripting_(XSS))

Cheat Sheet: [https://www.owasp.org/index.php/XSS_\(Cross_Site_Scripting\)_Prevention_Cheat_Sheet](https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet)

A3 Broken Authentication and Session Management

Reference page: https://www.owasp.org/index.php/Top_10_2010-A3-Broken_Authentication_and_Session_Management

Cheat Sheet: https://www.owasp.org/index.php/Authentication_Cheat_Sheet

A4 Insecure Direct Object References

Reference page: https://www.owasp.org/index.php/Top_10_2010-A4-Insecure_Direct_Object_References

A5 Cross-site Request Forgery (CSRF)

Reference page: [https://www.owasp.org/index.php/Top_10_2010-A5-Cross-Site_Request_Forgery_\(CSRF\)](https://www.owasp.org/index.php/Top_10_2010-A5-Cross-Site_Request_Forgery_(CSRF))

Cheat Sheet: [https://www.owasp.org/index.php/Cross-Site_Request_Forgery_\(CSRF\)_Prevention_Cheat_Sheet](https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet)

A6 Security Misconfiguration

Reference page: https://www.owasp.org/index.php/Top_10_2010-A6-Security_Misconfiguration

A7 Insecure Cryptographic Storage

Reference page: https://www.owasp.org/index.php/Top_10_2010-A7-Insecure_Cryptographic_Storage

A8 Failure to Restrict URL Access

Reference page: https://www.owasp.org/index.php/Top_10_2010-A8-Failure_to_Restrict_URL_Access

A9 Insufficient Layer Protection

Reference page: https://www.owasp.org/index.php/Top_10_2010-A9-Insufficient_Transport_Layer_Protection

Cheat Sheet: https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet

A10 Unvalidated Redirects and Forwards

Reference page: https://www.owasp.org/index.php/Top_10_2010-A10-Unvalidated_Redirects_and_Forwards

Notes

ⁱ OWASP Project web site: <https://www.owasp.org/>. The About OWASP page: https://www.owasp.org/index.php/About_OWASP. Lists of OWASP stable, beta and alpha projects: https://www.owasp.org/index.php/Category:OWASP_Project.

ⁱⁱ OWASP Top 10 project home: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project.

Press release announcing the OWASP Top 10 for 2010: <https://www.owasp.org/index.php/OWASPTop10-2010-PressRelease>.

The OWASP Top 10 for 2010 report document:

<http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202010.pdf>

Slide presentation:

http://owasptop10.googlecode.com/files/OWASP_Top_10_-_2010%20Presentation.pptx

Videos about the report: <http://www.vimeo.com/9006276>

http://blip.tv/owasp-appsec-conference-in-europe/day2_track1_1430-1505-3936900

ⁱⁱⁱ @stake: 'Significant' Security Flaws in e-Biz Software, InternetNews.com, February 2002, <http://www.internetnews.com/asp-news/article.php/978941/stake-Significant-Security-Flaws-in-e-Biz-Software.htm>.

^{iv} See, for example, *The Case for Building in Web Application Security from the Start*, Charles Kolodgy, IDC, February 2011.