Securing Web Application Technologies (SWAT) CHECKLIST

The SWAT Checklist provides an easy to reference set of best practices that raise awareness and help development teams create more secure applications. It's a first step toward building a base of security knowledge around web application security. Use this checklist to identify the minimum standard that is required to neutralize vulnerabilities in your critical applications.

ERROR HANDLING AND LOGGING

BEST PRACTICE	DESCRIPTION	CWE ID
☐ Display generic error messages	Error messages should not reveal details about the internal state of the application. For example, file system path and stack information should not be exposed to the user through error messages.	CWE-209
☐ No unhandled exceptions	Given the languages and frameworks in use for web application development, never allow an unhandled exception to occur. Error handlers should be configured to handle unexpected errors and gracefully return controlled output to the user.	CWE-391
☐ Suppress framework generated errors	Your development framework or platform may generate default error messages. These should be suppressed or replaced with customized error messages as framework generated messages may reveal sensitive information to the user.	CWE-209
☐ Log all authentication activities	Any authentication activities, whether successful or not, should be logged.	CWE-778
☐ Log all privilege changes	Any activities or occasions where the user's privilege level changes should be logged.	CWE-778
☐ Log administrative activities	Any administrative activities on the application or any of its components should be logged.	CWE-778
☐ Log access to sensitive data	Any access to sensitive data should be logged. This is particularly important for corporations that have to meet regulatory requirements like HIPAA, PCI, or SOX.	CWE-778
☐ Do not log inappropriate data	While logging errors and auditing access is important, sensitive data should never be logged in an unencrypted form. For example, under HIPAA and PCI, it would be a violation to log sensitive data into the log itself unless the log is encrypted on the disk. Additionally, it can create a serious exposure point should the web application itself become compromised.	CWE-532
☐ Store logs securely	Logs should be stored and maintained appropriately to avoid information loss or tampering by intruder. Log retention should also follow the rention policy set forth by the organization to meet regulatory requirements and provide enough information for foresic and incident response activities.	CWE-533

THE MOST TRUSTED NAME IN INFORMATION AND SOFTWARE SECURITY

SANS

Security Roadmap POSTER

WINTER 2013 – 23RD EDITION

AND

DATA PROTECTION

BEST PRACTICE	DESCRIPTION	CWE ID
☐ Use SSL everywhere	Ideally, SSL should be used for your entire application. If you have to limit where it's used then SSL must be applied to any authentication pages as well as all pages after the user is authenticated. If sensitive information (e.g. personal information) can be submitted before authentication those features must also be sent over SSL. EXAMPLE: Firesheep	CWE-311 CWE-319 CWE-523
☐ Disable HTTP access for all SSL enabled resources	For all pages requiring protection by SSL, the same URL should not be accessible via the non-SSL channel.	CWE-319
☐ Use the Strict-Transport-Security header	The Strict-Transport-Security header ensures that the browser does not talk to the server over non-SSL. This helps reduce the risk of SSL stripping attacks as implemented by the sslsniff tool.	
☐ Store user passwords using a strong, iterative, salted hash	User passwords must be stored using secure hashing techniques with a strong algorithm like SHA-256. Simply hashing the password a single time does not sufficiently protect the password. Use iterative hashing with a random salt to make the hash strong. EXAMPLE: LinkedIn password leak	CWE-257
☐ Securely exchange encryption keys	If encryption keys are exchanged or pre-set in your application then any key establishment or exchange must be performed over a secure channel.	
☐ Set up secure key management processes	When keys are stored in your system they must be properly secured and only accessible to the appropriate staff on a need to know basis.	CWE-320
☐ Disable weak SSL ciphers on servers	Weak SSL ciphers must be disabled on all servers. For example, SSL v2 has known weaknesses and is not considered to be secure. Additionally, some ciphers are cryptographically weak and should be disabled.	
☐ Use valid SSL certificates from a reputable CA	SSL certificates should be signed by a reputable certificate authority. The name on the certificate should match the FQDN of the website. The certificate itself should be valid and not expired. EXAMPLE: CA Compromise (http://en.wikipedia.org/wiki/DigiNotar)	
☐ Disable data caching using cache control headers and autocomplete	Browser data caching should be disabled using the cache control HTTP headers or meta tags within the HTML page. Additionally, sensitive input fields, such as the login form, should have the autocomplete=off setting in the HTML form to instruct the browser not to cache the credentials.	CWE-524
☐ Limit the use and storage of sensitive data	Conduct an evaluation to ensure that sensitive data is not being unnecessarily transported or stored. Where possible, use tokenization to reduce data exposure risks.	

CONFIGURATION AND OPERATIONS

BEST PRACTICE	DESCRIPTION	CWE ID
☐ Establish a rigorous change management process	A rigorous change management process must be maintained during operations. For example, new releases should only be deployed after proper testing and associated documentation has been completed. EXAMPLE: RBS production outage (http://www.computing.co.uk/ctg/analysis/2186972/rbs-wrong-rbs-manager)	CWE-439
☐ Define security requirements	Engage the business owner to define security requirements for the application. This includes items that range from the whitelist validation rules all the way to nonfunctional requirements like the performance of the login function. Defining these requirements up front ensures that security is baked into the system.	
☐ Conduct a design review	Integrating security into the design phase saves money and time. Conduct a risk review with security professionals and threat model the application to identify key risks. The helps you integrate appropriate countermeasures into the design and architecture of the application.	CWE-701 CWE-656
☐ Perform code reviews	Security focused code reviews can be one of the most effective ways to find security bugs. Regularly review your code looking for common issues like SQL Injection and Cross-Site Scripting.	CWE-702
☐ Perform security testing	Conduct security testing both during and after development to ensure the application meets security standards. Testing should also be conducted after major releases to ensure vulnerabilities did not get introduced during the update process.	
☐ Harden the infrastructure	All components of infrastructure that support the application should be configured according to security best practices and hardening guidelines. In a typical web application this can include routers, firewalls, network switches, operating systems, web servers, application servers, databases, and application frameworks.	CWE-15 CWE-656
☐ Define an incident handling plan	An incident handling plan should be drafted and tested on a regular basis. The contact list of people to involve in a security incident related to the application should be well defined and kept up to date.	
☐ Educate the team on security	Training helps define a common language that the team can use to improve the security of the application. Education should not be confined solely to software developers, testers, and architects. Anyone associated with the development process, such as business analysts and project managers, should all have periodic software security awareness training.	

AUTHENTICATION

BEST PRACTICE	DESCRIPTION	CWE ID
☐ Don't hardcode credentials	Never allow credentials to be stored directly within the application code. While it can be convenient to test application code with hardcoded credentials during development this significantly increases risk and should be avoided. EXAMPLE: Hard coded passwords in networking devices https://www.us-cert.gov/control_systems/pdf/ICSA-12-243-01.pdf	CWE-798
☐ Develop a strong password reset system	Password reset systems are often the weakest link in an application. These systems are often based on the user answering personal questions to establish their identity and in turn resetthe password. The system needs to be based on questions that are both hard to guess and brute force. Additionally, any password reset option must not reveal whether or not an account is valid, preventing username harvesting. EXAMPLE: Sara Palin password hack (http://en.wikipedia.org/wiki/Sarah_Palin_email_hack)	CWE-640
☐ Implement a strong password policy	A password policy should be created and implemented so that passwords meet specific strength criteria. EXAMPLE: http://www.pcworld.com/article/128823/study_weak_passwords_really_do_help_hackers.html	CWE-521
☐ Implement account lockout against brute force attacks	Account lockout needs to be implemented to guard against brute forcing attacks against both the authentication and password reset functionality. After serveral tries on a specific user account, the account should be locked for a period of time or until manually unlocked. Additionally, it is best to continue the same failure message indicating that the credentials are incorrect or the account is locked to prevent an attacker from harvesting usernames.	CWE-307
☐ Don't disclose too much information in error messages	Messages for authentication errors must be clear and, at the same time, be written so that sensitive information about the system is not disclosed. For example, error messages which reveal that the userid is valid but that the corresponsing password is incorrect confirms to an attacker that the account does exist on the system.	
☐ Store database credentials securely	Modern web applications usually consist of multiple layers. The business logic tier (processing of information) often connects to the data tier (database). Connecting to the database, of course, requires authentication. The authentication credentials in the business logic tier must be stored in a centralized location that is locked down. Scattering credentials throughout the source code is not acceptable. Some development frameworks provide a centralized secure location for storing credentials to the backend database. These encrypted stores should be leveraged when possible.	CWE-257
☐ Applications and Middleware should run with minimal privileges	If an application becomes compromised it is important that the application itself and any middleware services be configured to run with minimal privileges. For instance, while the application layer or business layer needs the ability to read and write data to the underlying database, administrative credentials that grant access to other databases or tables should not be provided.	CWE-250

SESSION MANAGEMENT

BEST PRACTICE	DESCRIPTION	CWE ID
☐ Ensure that session identifiers are sufficiently random	Session tokens must be generated by secure random functions and must be of a sufficient length so as to withstand analysis and prediction.	CWE-6
☐ Regenerate session tokens	Session tokens should be regenerated when the user authenticates to the application and when the user privilege level changes. Additionally, should the encryption status change, the session token should always be regenerated.	CWE-384
☐ Implement an idle session timeout	When a user is not active, the application should automatically log the user out. Be aware that Ajax applications may make recurring calls to the application effectively resetting the timeout counter automatically.	CWE-613
☐ Implement an absolute session timeout	Users should be logged out after an extensive amount of time (e.g. 4-8 hours) has passed since they logged in. This helps mitigate the risk of an attacker using a hijacked session.	CWE-613
☐ Destroy sessions at any sign of tampering	Unless the application requires multiple simultaneous sessions for a single user, implement features to detect session cloning attempts. Should any sign of session cloning be detected, the session should be destroyed, forcing the real user to reauthenticate.	
☐ Invalidate the session after logout	When the user logs out of the application the session and corresponding data on the server must be destroyed. This ensures that the session can not be accidentially revived.	CWE-613
☐ Place a logout button on every page	The logout button or logout link should be easily accessible to the user on every page after they have authenticated.	
☐ Use secure cookie attributes (i.e. HttpOnly and Secure flags)	The session cookie should be set with both the HttpOnly and the Secure flags. This ensures that the session id will not be accessible to client-side scripts and it will only be transmitted over SSL, respectively.	CWE-79 CWE-614
☐ Set the cookie domain and path correctly	The cookie domain and path scope should be set to the most restrictive settings for your application. Any wildcard domain scoped cookie must have a good justification for its existence.	
☐ Set the cookie expiration time	The session cookie should have a reasonable expiration time. Non-expiring session cookies should be avoided.	

Website: http://software-security.sans.org
Free resources, white papers, webcasts, and more

Blog: http://software-security.sans.org/blog

Twitter: @sansappsec
Latest news, promos, and other information

Secure Coding Assessment:
http://software-security.sans.org/courses/assessment

CORE SECURITY TECHNOLOGIES
www.coresecurity.com

WHITEPAPER TITLE
"Becoming the APT"
http://coresecurity.com/files/attachments/Becoming-the-APT-Rebranded-Final.pdf

INPUT AND OUTPUT HANDLING

BEST PRACTICE	DESCRIPTION	CWE ID
☐ Conduct contextual output encoding	All output functions must contextually encode data before sending it to the user. Depending on where the output will end up in the HTML page, the output must be encoded differently. For example, data placed in the URL context must be encoded differently than data placed in JavaScript context within the HTML page. EXAMPLE: Resource: https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet	CWE-79
☐ Prefer whitelists over blacklists	For each user input field, there should be validation on the input content. Whitelisting input is the preferred approach. Only accept data that meets a certain criteria. For input that needs more flexibility, blacklisting can also be applied where known bad input patterns or characters are blocked.	CWE-159 CWE-144
☐ Use parameterized SQL queries	SQL queries should be crafted with user content passed into a bind variable. Queries written this way are safe against SQL injection attacks. SQL queries should not be created dynamically using string concatenation. Similarly, the SQL query string used in a bound or parameterized query should never be dynamically built from user input. EXAMPLE: Sony SQL injection Hack (http://www.infosecurity-magazine.com/view/27930/lulzsec-sony-pictures-hackers-were-school-chums)	CWE-89 CWE-564
☐ Use tokens to prevent forged requests	In order to prevent Cross-Site Request Forgery attacks, you must embed a random value that is not known to third parties into the HTML form. This CSRF protection token must be unique to each request. This prevents a forged CSRF request from being submitted because the attacker does not know the value of the token.	CWE-352
☐ Set the encoding for your application	For every page in your application set the encoding using HTTP headers or meta tags within HTML. This ensures that the encoding of the page is always defined and that browser will not have to determine the encoding on its own. Setting a consistent encoding, like UTF-8, for your application reduces the overall risk of issues like Cross-Site Scripting.	CWE-172
☐ Validate uploaded files	When accepting file uploads from the user make sure to validate the size of the file, the file type, and the file contents as well as ensuring that it is not possible to override the destination path for the file.	CWE-434 CWE-616 CWE-22
☐ Use the nosniff header for uploaded content	When hosting user uploaded content which can be viewed by other users, use the X-Content-Type-Options: nosniff header so that browsers do not try to guess the data type. Sometimes the browser can be tricked into displaying the data type incorrectly (e.g. showing a GIF file as HTML). Always let the server or application determine the data type.	CWE-430
☐ Validate the source of input	The source of the input must be validated. For example, if input is expected from a POST request do not accept the input variable from a GET request.	CWE-20 CWE-346
☐ Use the X-Frame-Options header	Use the X-Frame-Options header to prevent content from being loaded by a foreign site in a frame. This mitigates Clickjacking attacks. For older browsers that do not support this header add framebusting Javascript code to mitigate Clickjacking (although this method is not foolproof and can be circumvented). EXAMPLE: Flash camera and mic hack (http://jeremiahgrossman.blogspot.com/2008/10/clickjacking-web-pages-can-see-and-hear.html)	CAPEC-103 CWE-693
☐ Use Content Security Policy (CSP) or X-XSS-Protection headers	Content Security Policy (CSP) and X-XSS-Protection headers help defend against many common reflected Cross-Site Scripting (XSS) attacks.	CWE-79 CWE-692

ACCESS CONTROL

BEST PRACTICE	DESCRIPTION	CWE ID
☐ Apply access controls checks consistently	Always apply the principle of complete mediation, forcing all requests through a common security "gate keeper." This ensures that access control checks are triggered whether or not the user is authenticated.	CWE-284
☐ Apply the principle of least privilege	Make use of a Mandatory Access Control system. All access decisions will be based on the principle of least privilege. If not explicitly allowed then access should be denied. Additionally, after an account is created, rights must be specifically added to that account to grant access to resources.	CWE-272 CWE-250
☐ Don't use direct object references for access control checks	Do not allow direct references to files or parameters that can be manipulated to grant excessive access. Access control decisions must be based on the authenticed user identity and trusted server side information.	CWE-284
☐ Don't use unvalidated forwards or redirects	An unvalidated forward can allow an attacker to access private content without authentication. Unvalidated redirects allow an attacker to lure victims into visiting malicious sites. Prevent these from occurring by conducting the appropriate access controls checks before sending the user to the given location.	CWE-601

SOFTWARE SECURITY CURRICULUM

Defense

Securing the App (STA) Application Security Awareness

11 Modules – 10 minutes each providing awareness-level training for people involved in application development.

- Application Security training for development teams
- Duration = 2 hours
- Delivered = Computer-Based Training (CBT)
- Quizzes included

www.securingtheapp.org

Secure Coding

JAVA
DEV541 Secure Coding in Java/JEE (4-Day Course) *GSSP-JAVA*

.NET
DEV544 Secure Coding in .NET (4-Day Course) *GSSP-.NET*

C & C++
DEV543 Secure Coding in C & C++

Language Agnostic
DEV536 Secure Coding: Developing Defensible Apps

DEV522 Defending Web Applications Security Essentials *GWEB*

Attack

SEC542 Web App Pen Testing and Ethical Hacking *GWAPT*

SEC642 Advanced Web App Pen Testing and Ethical Hacking *New!*

Additional Software Security Courses http://software-security.sans.org

Security Awareness Roadmap

Just like computers, people store, process, and transfer information. However, very little has been done to secure this "human" operating system, or HumanOS. As a result, people rather than technology are now the primary attack vector. Security awareness training is one of the most effective ways to address this problem. This roadmap is designed to help your organization build, maintain and measure a high-impact security awareness program that reduces risk by changing people's behavior and also meets your legal, compliance, and audit requirements. To use this roadmap, first identify the maturity level of your security awareness program and where you want to take it. Then follow the detailed steps to get there.

1 No Awareness Program

Program does not exist. Employees have no idea that they are a target, do not know or understand organizational security policies, and easily fall victim to cyber or human-based attacks.

2 Compliance Focused

Program designed primarily to meet specific compliance or audit requirements. Training is limited to annual or ad-hoc basis. Employees are unsure of organizational policies, their role in protecting their organization's information assets, and how to prevent, identify, or report a security incident.

How To Get There:

- Identify compliance or audit standards that your organization must adhere to.
- Identify security awareness requirements for those standards, which will likely require coordination with compliance or audit officer.
- Develop or purchase training to meet those requirements.
- Deploy security awareness training.
- Track who completes training, and when.

Deliverables:

- Annual training materials such as videos, newsletters and on-site presentations.
- Reports of who has and who has not completed required training.

Standards Requiring Awareness Training

- ISO/IEC 27002 §8.2.2
- PCI DSS §12.6
- SOX §404(a).(a).(1)
- GLBA §6801.(b).(1).(3)
- FISMA §3544.(b).(4).(A),(B)
- HIPAA §164.308.(a).(5).(i)
- NERC §CIP-004-3(B)(R1)
- EU Data Protection Directive

3 Promotes Awareness & Change

Program identifies the training topics that have the greatest impact in supporting the organization's mission and focuses on those key topics. Program goes beyond just annual training and includes continual reinforcement throughout the year. Content is communicated in an engaging and positive manner that encourages behavior change at work, home, and while traveling. As a result, employees, contractors and staff understand and follow organizational policies and actively recognize, prevent and report incidents.

How To Get There:

- Begin by identifying stakeholders in your organization. These are the individuals who are key to making your program a success. Once identified, build and execute a plan to gain their support. Methods to gain support include a human risk survey, awareness assessments, root cause analysis of recent incidents, industry reports or cost-benefit analysis.
- Create a baseline of your organization's security awareness level, such as with a human risk survey or phishing assessment. For additional examples refer to the Metrics section.
- Create a Project Charter that gives you authorization to begin the planning process. The Project Charter should set key expectations including identifying the project manager, cost estimates, program scope, goals, milestones, and assumptions.
- Have management review the Project Charter. Once it is approved, planning can officially begin.
- Establish a Steering Committee to assist in planning, executing, and maintaining the awareness program. Steering Committee should include 5-10 volunteer advisors from different departments or business units within your organization.
- Identify WHO you will be targeting in your program. Different roles may require different or additional training, including employees, help desk, IT staff, developers, and senior leadership.
- Identify WHAT you will communicate to the different groups targeted by your program. The goal is to create the shortest training possible that has the greatest impact. Begin with a risk analysis to identify the different human-based risks to your organization, document those risks in a matrix, and then prioritize the risks from high to low. Then select which risks you will address in your program based on priority level, time restrictions and other organizational requirements. Create a separate Learning Objectives document for each topic that identifies the different behaviors you need to change.
- Once you have determined WHO is the target of your awareness program and WHAT you will teach them, determine HOW you will communicate that content. To create an engaging program focus on how people will benefit from the training, how most of the lessons apply to their personal lives. There are two categories of training: Primary and Reinforcement. Primary training teaches new content and is usually taught annually or semi-annually and either onsite or online. Reinforcement training is employed throughout the rest of the year to reinforce key topics. Common examples of reinforcement training include newsletters, posters, podcasts, assessments and blogs. When teaching a specific topic, refer to that topic's Learning Objectives document to determine what content to communicate. This way regardless of the different ways you communicate a topic, the message will always be consistent.
- Create an execution plan in coordination with your Steering Committee. The plan should begin with WHY you are launching a security awareness program and its goals and overall scope. Then document WHO you will target in your awareness program, WHAT you will teach them and HOW. Include a timeline that identifies key milestones and the launch date of the program, critical resources involved and any other relevant information your organization may require for planning purposes.
- Have management review the plan. Once the plan is approved, you can execute your awareness program. Have the most senior stakeholder (such as your CEO) announce the program to the organization, such as by email, blog posting, or taped video.

Deliverables:

- Stakeholder matrix
- Gaining stakeholder support presentation
- Human risk survey
- Project Charter
- Steering Committee matrix
- Topics matrix
- Learning objectives document for each topic
- Execution plan

4 Long-Term Sustainment

Program has processes and resources in place for a long-term life cycle, including at a minimum an annual review and update of both training content and communication methods. As a result, the program is an established part of the organization's culture and is current and engaging.

How To Get There:

- Identify when you will review your awareness program each year.
- Identify new or changing technologies, threats, business requirements, or compliance standards that should be included in your annual update.
- Conduct an assessment of your organization's security awareness level and compare that to the baseline taken in stage 3.
- Survey staff for feedback, including what elements they liked best about the program, what needs to be changed, which topic they found most interesting, and which behaviors they changed.
- Review all the topics you are communicating and identify if new topics need to be added, and which existing topics should be removed or updated.
- Once topic changes have been identified, review and update the learning objectives for each topic.
- Review how the topics are communicated, which methods have had the greatest impact, and which need to be updated or dropped.
- Conduct an annual review and update of the budget to address changing business objectives.

Deliverables:

- Content tracking matrix used to document which topics and learning objectives were updated, by whom, and when.

5 Metrics Framework

Program has a robust metrics framework to track progress and measure impact. As a result, the program is continuously improving and able to demonstrate return on investment. In addition, some set of metrics will be used in previous stages.

How To Get There:

- Identify key metrics that relate to business outcomes.
- Document how and when you intend to measure the metrics.
- Identify who to communicate results to, when, and how.
- Execute metrics measurement.

Deliverables:

- Metrics matrix

Examples of Metrics:

- No. of people who fall victim to monthly phishing assessments.
- No. of monthly infected systems.
- No. of monthly incidents reported.
- No. of people who completed the awareness training.
- No. of weak or shared passwords.
- Employee scores from before/after testing.
- % of users sampled with positive attitude towards information security.
- % of users sampled who believe their actions can have an impact on security.

Additional Materials:

- **NIST SP800-50**
  Building an Information Technology Security Awareness and Training Program
- **ENISA Awareness Guide (2010)**
  How to Raise Information Security Awareness
- **20 Critical Controls**
  Twenty Critical Security Controls for Effective Cyber Defense

About the Poster

This roadmap was developed as a consensus project by security professionals actively involved in security awareness programs. If you have any suggestions or would like to get involved please contact community@securingthehuman.org

Contributors Include: Randy Marchany (Virginia Tech), Cortney Stephens (Union Gas), Julie Sobel (Alliance Data), Tonia Dudley (Honeywell), John Andrew (Honeywell), Pieter Danhieux (BAE Systems Detica), Vivian Gernand (Corning), Christopher Ipsen (State of Nevada), Jenn Lesser (Facebook), Mark Merkow (PayPal), Sam Segran (Texas Tech University), Tracy Grunig (Arizona State University), Geordie Stewart (Risk Intelligence), Greg Aurigemma (Flight Safety), Janet Roberts (Progressive Insurance), Chris Sorensen (GE Capital), Mary Naphen (Lincoln Financial Group), David Vaughn (HP Enterprise Services), Tim Harwood (BP), Tanja Craig (BP), Dave Piscitello (ICANN), Eric Phifer (Seacost National Bank), Antonio Merola.

Documents followed by this icon may be downloaded at:
www.securingthehuman.org/resources/planning