

# 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
<input checked="" type="checkbox"/> 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. For authentication errors, do not indicate that the username exists.	CWE-209
<input checked="" type="checkbox"/> 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
<input checked="" type="checkbox"/> 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
<input checked="" type="checkbox"/> Log all authentication and validation activities	Log any authentication and session management activities along with all input validation failures. Any security-related events should be logged. These may be used to detect past or in-progress attacks.	CWE-778
<input checked="" type="checkbox"/> Log all privilege changes	Any activities or occasions where the user's privilege level changes should be logged.	CWE-778
<input checked="" type="checkbox"/> Log administrative activities	Any administrative activities on the application or any of its components should be logged.	CWE-778
<input checked="" type="checkbox"/> 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
<input checked="" type="checkbox"/> Do not log inappropriate data	While logging errors and auditing access are 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
<input checked="" type="checkbox"/> Store logs securely	Logs should be stored and maintained appropriately to avoid information loss or tampering by intruders. Log retention should also follow the retention policy set forth by the organization to meet regulatory requirements and provide enough information for forensic and incident response activities.	CWE-533



## Cloud Security and DevSecOps Best Practices

AND

## Securing Web Application Technologies (SWAT) CHECKLIST

Version 1.11

Ingraining security into the mind of every developer.

[sans.org/cloud-security](http://sans.org/cloud-security)

DATA PROTECTION		
BEST PRACTICE	DESCRIPTION	CWE ID
<input checked="" type="checkbox"/> Use HTTPS everywhere	Ideally, HTTPS should be used for your entire application. If you have to limit where it's used, then HTTPS must be applied to any authentication pages as well as to 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 HTTPS. Always link to the HTTPS version of URL if available. Relying on redirection from HTTP to HTTPS increases the opportunity for an attacker to insert a man-in-the-middle attack without raising the user's suspicion. EXAMPLE: <a href="#">sslstrip</a>	CWE-311 CWE-319 CWE-523
<input checked="" type="checkbox"/> Disable HTTP access for all protected resources	For all pages requiring protection by HTTPS, the same URL should not be accessible via the insecure HTTP channel.	CWE-319
<input checked="" type="checkbox"/> Use strong TLS configurations	TLS must be configured to the secure configurations that only support the recent versions of TLS, prefer the use of the strongest cipher suites and avoid the use of any weak ciphers. For example, SSL and TLS protocols prior to TLS 1.2 have known weaknesses and are not considered secure. Additionally, disable the cipher suites using RC4, DES or MD5 and prefer the ciphers that support Perfect Forward Secrecy. EXAMPLE: Qualys SSL Labs	
<input checked="" type="checkbox"/> Use the Strict-Transport-Security header	The Strict-Transport-Security header ensures that the browser does not talk to the server over HTTP. This helps reduce the risk of HTTP downgrade attacks as implemented by the <a href="#">ssltinse tool</a> .	
<input checked="" type="checkbox"/> Store user passwords using a strong, iterative, salted hash	User passwords must be stored using secure hashing techniques with strong algorithms like PBKDF2, bcrypt, or SHA-512. Simply hashing the password a single time does not sufficiently protect the password. Use adaptive hashing (a work factor), combined with a randomly generated salt for each user to make the hash strong. EXAMPLE: <a href="#">https://haveibeenpwned.com</a>	CWE-257
<input checked="" type="checkbox"/> 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.	
<input checked="" type="checkbox"/> 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. EXAMPLE: AWS Key Management Service (KMS), Azure Key Vault, AWS CloudHSM	CWE-320
<input checked="" type="checkbox"/> Use valid HTTPS certificates from a reputable certificate authority	HTTPS 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: Let's Encrypt <a href="#">https://letsencrypt.org</a>	
<input checked="" type="checkbox"/> 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 attribute set to off in the HTML form to instruct the browser not to cache the credentials.	CWE-524
<input checked="" type="checkbox"/> Encrypt sensitive data at rest	Encrypt sensitive or critical data before storage.	CWE-311 CWE-312
<input checked="" type="checkbox"/> Limit the use and storage of sensitive data	Conduct an evaluation to ensure that sensitive data elements are not being unnecessarily transported or stored. Where possible, use tokenization to reduce data exposure risks.	

CONFIGURATION AND OPERATIONS		
BEST PRACTICE	DESCRIPTION	CWE ID
<input checked="" type="checkbox"/> Automate application deployment	Automating the deployment of your application, using Continuous Integration and Continuous Deployment, helps to ensure that changes are made in a consistent, repeatable manner in all environments.	
<input checked="" type="checkbox"/> 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: DevOps Audit Defense Toolkit <a href="#">https://itrevolution.com/devops-audit-defense-toolkit</a>	CWE-439
<input checked="" type="checkbox"/> 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.	
<input checked="" type="checkbox"/> 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. This helps you integrate appropriate countermeasures into the design and architecture of the application.	CWE-701 CWE-656
<input checked="" type="checkbox"/> 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. Leverage automated tools to maximize breadth of coverage and consistency.	CWE-702
<input checked="" type="checkbox"/> Perform security testing	Conduct security testing both during and after development to ensure that the application meets security standards. Testing should also be conducted after major releases to ensure that vulnerabilities did not get introduced during the update process. Leverage automation by including security tests into the CI/CD pipeline.	
<input checked="" type="checkbox"/> 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
<input checked="" type="checkbox"/> 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.	
<input checked="" type="checkbox"/> 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 have periodic software security awareness training.	

AUTHENTICATION		
BEST PRACTICE	DESCRIPTION	CWE ID
<input checked="" type="checkbox"/> 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. Proper secrets management tools can provide proper encryption and credentials rotation to provide extra resiliency to attacks. EXAMPLE: Hardcoded passwords in networking devices <a href="#">https://www.us-cert.gov/control_systems/pdf/ICS-12-243-01.pdf</a>	CWE-798
<input checked="" type="checkbox"/> Develop a strong password reset system	Password reset systems are often the weakest link in an application. These systems are often based on users answering personal questions to establish their identity and in turn reset the 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: Sarah Palin password hack <a href="#">http://en.wikipedia.org/wiki/Sarah_Palin_email_hack</a>	CWE-640
<input checked="" type="checkbox"/> Implement a strong password policy	A password policy should be created and implemented so that passwords meet specific strength criteria. EXAMPLE: <a href="#">https://pages.nist.gov/800-63-3/sp800-63-3.html</a>	CWE-521
<input checked="" type="checkbox"/> Implement account lockout against brute-force attacks	Account lockout needs to be implemented to prevent brute-force attacks against both the authentication and password reset functionality. After several tries on a specific user account, the account should be locked for a period of time or until it is 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
<input checked="" type="checkbox"/> 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 that reveal that the user ID is valid but that the corresponding password is incorrect confirm to an attacker that the account does exist on the system.	
<input checked="" type="checkbox"/> 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
<input checked="" type="checkbox"/> 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 need 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
<input checked="" type="checkbox"/> Ensure that session identifiers are sufficiently random	Session tokens must be generated by secure random functions and must be of sufficient length to withstand analysis and prediction.	CWE-6
<input checked="" type="checkbox"/> 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
<input checked="" type="checkbox"/> 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
<input checked="" type="checkbox"/> 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
<input checked="" type="checkbox"/> 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.	
<input checked="" type="checkbox"/> 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 cannot be accidentally revived.	CWE-613
<input checked="" type="checkbox"/> Place a logout button on every page	The logout button or logout link should be easily accessible to users on every page after they have authenticated.	
<input checked="" type="checkbox"/> Use secure cookie attributes	The session cookie should have the HttpOnly, Secure, and SameSite flags set. This ensures that the session ID will not be accessible to client-side scripts, will only be transmitted over HTTPS, and will only be sent with requests from the same site (mitigates CSRF).	CWE-79 CWE-614
<input checked="" type="checkbox"/> 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.	
<input checked="" type="checkbox"/> Use non-persistent cookies	If a cookie has the "Max-Age" or "Expires" attributes, the browser treats it as a persistent cookie and stores it to disk until the expiration time. Do not do this for session cookies.	

	<b>SEC488: Cloud Security Essentials</b> License to learn cloud security.
	<b>SEC510: Public Cloud Security: AWS, Azure, and GCP</b> Multiple clouds require multiple solutions.
	<b>SEC522: Defending Web Applications Security Essentials</b> Not a matter of "if" but "when." Be prepared for a web attack. We'll teach you how.
	<b>SEC534: Secure DevOps: A Practical Introduction</b> Principles. Practical Tools. Oh my. Start your journey on the DevSecOps road here.
	<b>Curriculum</b>

INPUT AND OUTPUT HANDLING		
BEST PRACTICE	DESCRIPTION	CWE ID
<input checked="" type="checkbox"/> Conduct contextual output encoding	All output functions must contextually encode data before sending the data 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 a JavaScript context within the HTML page.	CWE-79
<input checked="" type="checkbox"/> 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 meet 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
<input checked="" type="checkbox"/> 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.	CWE-89 CWE-564
<input checked="" type="checkbox"/> Prevent insecure deserialization	Do not accept serialized objects from untrusted sources, define known good data types when deserializing data, and implement integrity checks on serialized objects.	CWE-502
<input checked="" type="checkbox"/> 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
<input checked="" type="checkbox"/> Prevent Server Side Request Forgery (SSRF)	Features that require requests to be sent to web services need to carefully restrict URLs by validating input and properly encoding output.	CWE-172
<input checked="" type="checkbox"/> 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 the 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.	
<input checked="" type="checkbox"/> 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, and ensure that it is not possible to override the destination path for the file.	CWE-434 CWE-616 CWE-22
<input checked="" type="checkbox"/> Use the nosniff header for uploaded content	When hosting user uploaded content that 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
<input checked="" type="checkbox"/> Prevent tabnabbing	Use the "rel" anchor tag attribute with values of "noopener" or "noreferrer" to prevent an opened tab from tampering with the calling tabs location in the browser. In JavaScript this can be prevented by setting window.opener to null.	CWE-1022
<input checked="" type="checkbox"/> 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
<input checked="" type="checkbox"/> X-Frame-		

