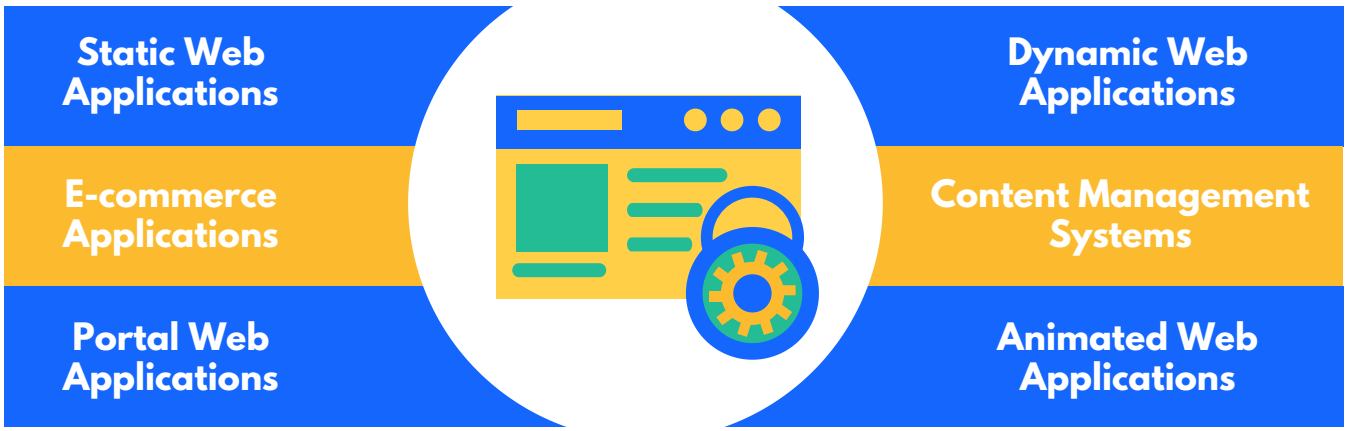
astra

The Ultimate Website/ Web Application Security Audit & Penetration Testing (VAPT) Checklist

Today, it's hard to find a website security guide that does not list web application security audit as a must. Cybersecurity enthusiasts have been highlighting the need for a website security audit for a while now. But it is only now that web owners started acknowledging it as a necessity for their business due to increasing risk of cyber attacks targeting their poorly secured websites or web applications.

While developing web applications, developers are provided objectives based solely on functionality and very little if any security measures are taken into account. After all, if it's not written into their job description, why should it concern them? This in turn produces applications containing multiple vulnerabilities ranging from weaknesses around input validation, error handling, session management, and failure to implement proper access controls.



Sometimes it takes an exploited vulnerability resulting in a data breach or application defacement for developers and managers to realize the impact of having these weaknesses in their application.

In this checklist, we will discuss steps to take to perform a detailed security audit and penetration testing for your web system and its security standards for finding and fixing such security vulnerabilities & loopholes in your web applications.

Web Application Security Audit Checklist

Data Validation	1
1	Ensure the data received is what is expected and the data returned to the users is of an expected output
3	Perform Input Validation for All Inputs (prevent “buffer overflow” errors)
4	Ensure asynchronous consistency to avoid timing and sequence errors, race conditions, deadlocks, order dependencies, synchronization errors, etc.
5	Use Commit-or-Rollback semantics for exception handling
6	Use multitasking and multithreading safely
7	Set initial values for data
8	Avoid or eliminate “backdoors”
9	Avoid or eliminate shell escapes
10	Validate configuration files before use
11	Validate command-line parameters before use
12	Validate environment variables before use
13	Ensure communication connection queues cannot be exhausted

Data Protection 2

1	Examine all functions requiring user input or providing user output to ensure proper data protection for any sensitive information being utilized
2	Resource connection strings must be encrypted
3	Sensitive data should never be in code
4	Cookies containing sensitive data should be encrypted
5	Sensitive data should not be passed from page to page on client side
6	Pages containing sensitive data must only retrieve the sensitive data on demand when it is needed instead of persisting or caching it in memory

Error Handling 3

1	Examine the error handling of all functions to ensure the errors are sanitized to the point of providing “need to know” information back to the user
2	Production applications should not provide the same error messages as used in development
3	For any functions that return output to the user, ensure all error messages provide the user with the minimal amount of information needed to correct the error
4	Never display any errors to the user that would reveal information about the system or application itself
5	Ensure that your application has a “safe mode” which it can return if something truly unexpected occurs. If all else fails, log the user out and close the browser window. Production code should not be capable of producing debug messages
6	Assign log files the highest security protection, providing reassurance that you always have an effective 'black box' recorder if things go wrong
7	Simply be aware of different attacks. Take every security violation seriously, always get to the bottom of the cause event log errors and don't just dismiss errors unless you can be completely sure that you know it to be a technical problem.

Communication Security 4

1	Ensure sensitive or personal data is never be passed in clear text through the URL String
2	Ensure message freshness; even a valid message may present a danger if utilized in a “replay attack”
3	Work with the developer to trace the data flow from the functions identified in “Where to Start” to ensure that proper encryption is taking place
4	Calls to the database should use parameterized stored procedures
5	Protect message confidentiality

Authentication 5

1	Authentication of an individual should be validated prior to disclosing or providing any information specific to that individual
2	Determine all areas where the user interface changes between public and privately visible information
3	Re-authenticate user for high value transactions and access to protected areas
4	Authenticate the transaction, not the user
5	With form-based authentication, use a trusted access control component rather than writing your own
6	Use strong authentication (tokens, certificates, etc)