

La prima fase di un Penetration test si sviluppa in un'analisi passiva volta a raccogliere il maggior numero di informazioni possibili sull'applicazione del cliente e sul contesto. Il secondo e più importante momento di un Penetration test consente al tester di raccogliere informazioni sulla sicurezza dell'applicazione applicando la metodologia descritta da OWASP in combinazione con la propria esperienza. Gli steps che compongono il test attivo sono 9 e si suddividono in più di 60 controlli diversi.

1. Configuration Management Testing Spesso le analisi della infrastruttura e la tipologia della architettura può portare ad importanti informazioni sulla applicazione web. Informazioni come codici sorgenti, metodi HTTP consentiti, funzionalità di amministrazione, metodi di autenticazione e configurazioni infrastrutturali.
2. Business Logic Testing L'autenticazione è l'azione che stabilisce o conferma qualcuno o qualcosa come verificato e abilitato. L'autenticità è un oggetto che dipende da uno o più fattori di autenticazione. Nella computer security, l'autenticazione è il processo che si attesta a verificare l'identità digitale dei soggetti di una comunicazione.
3. Authentication Testing Il cuore di ogni applicazione web è la modalità di mantenimento dello stato per gestire le relazioni tra utente e interfaccia. La gestione delle sessioni fonde il sistema di controllo della autenticazione con l'uso della applicazione. http è un protocollo stateless, significa quindi che i web-servers rispondono alle richieste dei client in modo non correlato. Come per tutte le applicazioni logiche si ha la necessità di legare le operazioni multiple degli utenti in una unica sessione. Per fare ciò si necessita di una soluzione middleware di terze parti e di soluzioni web server. Le soluzioni più comuni sono basate sui cookies o sulle sessionIDs.
4. Authorization testing L'autorizzazione è il concetto di assegnare le risorse solo se l'assegnazione è concessa. Testare l'autorizzazione significa quindi capire come lavora il processo di autorizzazione e usare queste informazioni per circuire il meccanismo. Si andrà quindi a verificare la sicurezza dopo aver ottenuto delle credenziali valide associate a ben definiti livelli di accesso. L'intento è quello di scalare l'assegnazione dei privilegi attraverso la vulnerabilità di path trasversal.
5. Session Management Testing Questo tipo di vulnerabilità non è identificata con uno scanner di vulnerabilità perché è specifica per l'applicazione. Dipende quindi dallo skill e dalle capacità del tester andare a identificare le problematiche legate alla specificità del prodotto web.
6. Data Validation Testing I problemi più comuni riscontrati nelle applicazioni web sono legati alla validazione dei dati di interazione con gli utenti. I dati delle interazioni con l'esterno (Entry Points) della applicazione devono essere verificati per garantire l'attendibilità delle operazioni svolte dal software web.
7. Denial of Service Testing Il tipo più comune attacco di Denial of Service (DoS) è quello usato in una rete per rendere un server irraggiungibile da parte di altri utenti validi. Un attacco DoS di rete è basato sul rendere inutilizzabile una macchina inviandole eccessive quantità di dati, rendendo il bersaglio incapace di tenere il passo con la volume di richieste che riceve. Quando l'utente malintenzionato utilizza un gran numero di macchine per floodare un computer di destinazione, la tecnica è generalmente nota come un distributed denial of service (DDoS). Questi tipi di attacchi sono generalmente al di là di ciò che uno sviluppatore di applicazioni può prevenire all'interno del codice. La "battaglia in rete" deve essere quindi mitigata attraverso l'architettura della rete. Vi sono, tuttavia dei tipi di vulnerabilità all'interno di applicazioni che possono consentire a un utente malintenzionato di rendere alcune funzionalità inutilizzabili o, talvolta, l'intero sito web non disponibile.
8. Web Services Testing Le architetture SOA / Webservices sono sistemi dinamici dal punto di vista dell'allocazione delle risorse. Dovendo soddisfare i bisogni degli utenti questi servizi possono essere raggiunti direttamente dai client piuttosto che dai server. Le vulnerabilità a cui sono soggetti sono simili alle vulnerabilità delle comuni applicazioni, come problematiche di SQL injection, come la potenziale perdita di informazioni ecc.
9. Ajax Testing Utilizzando le tecniche AJAX si possono avere benefici enormi dal punto di vista dell'usabilità per le applicazioni web. Le applicazioni AJAX però hanno una superficie di attacco maggiore delle applicazioni web normali e sono spesso sviluppate con una particolare attenzione su ciò che si può fare piuttosto che ciò dovrebbe essere fatto. Inoltre, le applicazioni AJAX sono più complicate perché il trattamento è svolto sia lato client che lato server. L'uso di framework per limitare la complessità può contribuire a velocizzare lo sviluppo, ma può anche condurre a situazioni in cui gli sviluppatori non comprendono appieno come il codice che stanno scrivendo verrà eseguito. Questo, in caso di particolari applicazioni e funzionalità, potrebbe rendere difficile valutare correttamente il rischio. Le applicazioni AJAX sono vulnerabili alla gamma completa delle vulnerabilità tradizionali. Inoltre le applicazioni AJAX possono essere vulnerabili a nuove classi di attacco come Cross Site Request Forgery (XSRF). I test di applicazioni AJAX possono essere difficili, anche perché gli sviluppatori si prendono spesso troppa libertà rispetto ai modi in cui avviene la comunicazione tra il client e il server, rendendo di fatto impossibile effettuare test automatizzati e obbligando quindi il controllo ad un'attività manuale, al contrario delle applicazioni web tradizionali in cui richieste GET o POST seguono formati standard ed è quindi molto più semplice inviare dati modificati per testare le risposte del server.

NOTA: In effetti alcuni controlli non vengono considerati quando l'applicazione testata presenta particolarità implementative che non ne prevedono la necessità.

