UNIVERSITÀ DEGLI STUDI DI SALERNO

Ingegneria del Software

SECURITY AND RECOVERY TESTING

ANNO ACCADEMICO 2015/2016

Versione 1.1

Top Manager:

Nome
Prof. De Lucia Andrea

Project Manager:

Nome	Matricola
De Chiara Davide	0222500088
Longo Alessandro	0222500121

Partecipanti:

Nome	Matricola
Castellano Dario	0512102552
D’Avanzo Antonio Luca	0512102502
De Blasio Christian	0512102268
De Rosa Federico	0512101904
Di Domenico Carlo	0512102316
Esposito Fabio	0512102426
Korniychuk Alina	0512102256
Martiniello Pasquale	0512102616
Pecorelli Fabiano	0512102390

Shevchenko Sergiy	0512102368
Tufano Giuseppina	0512102588
Zanin Elvira	0512102396

Revision History:

Data	Versione	Descrizione	Autore
05/01/2015	1.0	Stesura del documento	Shevchenko Sergiy
05/01/2015	1.1	Revisione del documento	Shevchenko Sergiy Korniychuk Alina

Indice

1. Introduzione

Il Security o Penetration test è il processo operativo di valutazione della sicurezza di un sistema o di una rete che simula l'attacco di un utente malintenzionato. L'analisi comprende più fasi ed ha come obiettivo evidenziare le debolezze della piattaforma fornendo il maggior numero di informazioni sulle vulnerabilità che ne hanno permesso l'accesso non autorizzato. L'analisi è condotta dal punto di vista di un potenziale attaccante e consiste nello sfruttamento delle vulnerabilità rilevate al fine di ottenere più informazioni possibili per accedere indebitamente al sistema.

2. Fasi

Nel caso di SimplEx, il Security test è stato suddiviso in 3 fasi:

- SQL Injection test
- JavaScript-HTML XSS test
- Privilege Escalation test

3. SQL Injection

Un SQL injection (SQLi) è un attacco mirato a colpire le applicazioni web che si appoggiano su un DBMS di tipo SQL. Questo attacco sfrutta l'inefficienza dei controlli sui dati ricevuti in input ed inserisce codice maligno all'interno di una query SQL. Le conseguenze prodotte sono imprevedibili per il programmatore, l'SQL injection permette al malintenzionato di autenticarsi con ampi privilegi in aree protette del sito anche senza essere in possesso delle credenziali di accesso e di visualizzare e/o alterare dati presenti del database.
Simplex interagisce con l'utente, che può inserire dei dati, e quindi potenzialmente effettuare una SQLi. La SQLi in se, deve contenere dei caratteri specifici della sintassi SQL, come ad esempio ‘ (l'apostrofo), “ (gli apici), ; (punto e virgola) ecc... La verifica dell'esistenza di questi caratteri nella input, garantisce l'impossibilità di effettuare una iniezione.

Tutti i campi input di simplex, prima di essere inseriti nella query verso il db, vengono validati con dei pattern regex (ad esempio nome utente viene validato con /^[a-zA-Z0-9_ èàòù]+$/, il quale rende impossibile l'inserimento dei caratteri necessari per una SQLi), oppure se sono

numerici con funzione is_numeric(). La validazione avviene nei due momenti diversi: lato client e lato server. Lato client non è sicuro, siccome un malintenzionato potrebbe eseguire una richiesta direttamente al server sorpassando validazione con jquery. La seconda verifica, lato server, impossibile sorpassarla, quindi rende il sistema sicuro.
I campi, dove sono necessari i caratteri specifici, ad esempio risposta alla domanda aperta, vengono utilizzati conversioni dell'input, con funzioni mysql_real_escape() oppure conversione di base con base64_encode().
I tool utilizzati per il testing sono: selenium, per testare corrispondenza dei pattern ai valori reali inseriti, e PostMan per eseguire le richieste POST/GET direttamente agli script del sistema.

4. JavaScript-HTML XSS test

JavaScript Injection consiste nel inserimento dei codici javascript nel form del sistema e una successiva esecuzione al momento della visualizzazione.
Facciamo un esempio: uno studente inserisce il codice javascript nella risposta ad una domanda aperta (è possibile inserire qualsiasi tipo di carattere) il quale elimina i cookie. Il docente, una volta aperta la pagina per verificare la risposta dello studente, involontariamente esegue il codice javaScript e viene reindirizzato alla pagina di login. Altro tipo di attacco potrebbe essere nell'utilizzo di XSS, ad esempio un codice inserito nella form della risposta provocherà lo stesso effetto del js sopra descritto.
Il sistema SimplEx prevede questi tipi di attacco, quindi qualsiasi input nel sistema dove sono necessari tutti tipi di carattere (domande e risposte) vengono ripuliti dal codice HTML con funzione strip_tags().

5. Privilege Escalation test

Il Privilege Escalation consiste nel tentativo di ottenere i privilegi più alti nel sistema. Ad esempio, uno studente potrebbe tentare di eseguire una richiesta alle pagine del docente.
Il sistema SimplEx ha adattato il sistema del routing avanzato per prevenire questo tipo di vulnerabilità. Qualsiasi richiesta al sistema, viene reindirizzata al router (index.php) dove ci sono terne di url e corrispettivi script php.

Il routing contiene 4 macro zone:

1. /auth/* - hanno accesso tutti;
2. /admin/* - hanno accesso solo gli utenti con tipologia: admin;
3. /docente/* - hanno accesso solo gli utenti con tipologia: docente;
4. /studente/* - hanno accesso solo gli utenti con tipologia: studente.

Importante notare, che non sono i singoli script php che sono responsabili alla verifica dei privilegi, bensì il router. Quindi se uno script ha la path /docente/modificaDomanda solo ed esclusivamente il docente può avere l'accesso a questo script.
Altro livello di sicurezza adattato è la limitazione di accesso diretto agli script, ad esempio /core/Auth/Login.php non è accessibile dal browser, invece route che porta a questo script (/auth/login) si.

6. Recovery testing

Un altro aspetto importante del sistema SimplEx è la corretta gestione dei fallimenti sia del sistema stesso, sia della rete. Il recovery test prevede due fasi: recovery del sistema durante il test, e la consistenza del sistema in generale.
La prima fase è quella più importante, perché la probabbilità di fallimento del sistema o del collegamento durante il test è alta, visto che di solito un test può durare circa 2 ore. SimplEx all'inizio del test scarica al lato client una webapp javascript, che:

- verifica continuamente se il server è online
- ogni risposta data viene immediatamente salvata, lato client e server

Nel caso di interruzione del collegamento a causa della caduta del server oppure problemi di rete, la webapp sospende il test, fino al ripristino del funzionamento del sistema. Nonostante la sospensione del test, dopo il ripristino, vengono verificate l'inizio e fine del test e se il test è scaduto, lo studente può o consegnare o ritirarsi altrimenti può tranquillamente continuare il test da dove aveva lasciato.

La consistenza del sistema in generale è garantita dal fatto che qualsiasi dato persistente viene salvato nel DB, ed ogni operazione è atomica. Quindi nel caso di fallimento, all'utente sarà visualizzato il messaggio di fallimento della richiesta e potrà riprovare.

7. Conclusione

Durante lo sviluppo di SimplEX sono state adottate diverse tecniche per garantire la sicurezza e stabilità del sistema stesso. Tutte le tecniche citate in questo documento sono state testate e all'atto del rilascio del sistema (07/01/2016) tutto risulta funzionante e coerente con i requisiti non funzionali definiti all'interno del requirements analysis document.