ELABORAREA ŞI CONFIGURAREA UNUI FRAMEWORK DE TESTARE A SECURITĂȚII INFORMAȚIONALE A UNEI APLICAȚII WEB

Felicia POJAR, studentă

CZU: 004.415.53 pojarfelicia@gmail.com

Security testing is an active, rigorous analysis of weaknesses, flaws and vulnerabilities. Through testing, you can identify the problems and repair them before data is lost. This is incredibly important when we are discussing about web applications that deal with a user's private information. In this project was investigated https://moodle.usm.md *against various types of cyber attacks, using Burp Suite. All the testing technics were automated within a special testing framework.*

Luând în considerație situația pandemică curentă și tendința de digitalizare a tuturor tipurilor de activități, în special în domeniul învățământului superior, instituțiile de stat au creat diferite platforme în care atât studeții, cât si profesorii au posibilitatea de a-și continua activitatea profesională în mod obișnuit. În cazul Universității de Stat din Moldova, decizia a fost luată în favoarea integrării platformei de management conținut Moodle.

Scopul lucrării respective este de a investiga și de a testa securitatea informațională a aplicației https://moodle.usm.md și a nivelului de asigurare a integrității datelor utilizatorilor logați în sistem, de asemeni posibilitatea de accesare a datelor străine și de logare pe profilurile străine. În aceeasi ordine de idei, se va investiga gradul de vulnerabilitate a aplicației web în fața diferitelor tehnici de atac cibernetic, printre care se numără:

- IDOR (*Insecure Direct Object References*) – este un tip de vulnerabilitate de securitate ce ține de controlul accesului, care apare atunci când o aplicație folosește datele de intrare ale utilizatorului pentru a accesa direct diferite obiecte.
- CSRF (*Cross-Site Request Forgery*) – este o vulnerabilitate de securitate web care permite unui atacator să inducă utilizatorii să efectueze acțiuni pe care nu intenționează să le efectueze. Permite

unui atacator să ocolească parțial aceeași politică de origine, care este concepută pentru a împiedica diferite site-uri web să interacționeze unul cu celălalt.

- CORS (*Cross-Origin Resource Sharing*) – este un mecanism de browser care permite accesul controlat la resursele situate în afara unui anumit domain. Acest fapt extinde și adaugă flexibilitate politicii de aceeași origine (SOP – *Same Origin Policy*). Cu toate acestea, oferă și potențial pentru atacuri bazate pe domain-uri multiple, dacă politica CORS a unui site web este slab configurată și implementată.
- *Reflected XSS (Cross Site Scripting*) – apare atunci când o aplicație primește date într-un request HTTP și include acele date în răspunsul imediat. Dacă un atacator poate controla un script executat în browser-ul victimei, atunci acesta poate compromite în totalitate acel utilizator. Atacatorul poate efectua oricare dintre acțiunile care sunt aplicabile impactului vulnerabilităților Reflected XSS.
- Stored XSS (*Cross Site Scripting*) – mai este cunoscută și ca second-order XSS sau persistent XSS, apare în momentul în care o aplicație primește date de intrare dintr-o sursă dubioasă și include acele date în răspunsurile HTTP ulterioare într-un mod nesigur.
- Formula injection – sau altfel denumită *CSV Injection* este o tehnică de atac cibernetic pentru exploatarea funcționalității „Export to spreadsheet" în aplicațiile web pentru a ataca utilizatorii, prin preluarea controlului asupra dispozitivului fizic în care fișierul a fost descărcat, și a fura conținutul foii de calcul.
- *Directory traversal* – cunoscută și sub numele de *file path traversal* este o vulnerabilitate de securitate web care permite unui atacator să citească fișiere arbitrare pe serverul care rulează o aplicație. Acestea pot include codul și datele aplicației, acreditările pentru sistemele back-end și fișierele sensibile ale sistemului de operare. În unele cazuri, un atacator ar putea să scrie în fișiere arbitrare de pe server, permițându-i să modifice datele aplicației sau comportamentul și, în cele din urmă, să preia controlul deplin al serverului.

În urma testării de penetrare a aplicației, fiecare caz de test executat a fost automatizat în cadrul unui framework de testare elaborat, fapt care asigura mentenanța proiectului și monitorizarea

continuă a stării de securitate a sistemului. În testarea de penetrare a aplicației a fost utilizat unul din cele mai populare instrumente de testare a securității web – *Burp Suite*, iar în crearea framework-ului autonom și independent pentru scrierea și executarea cazurilor de test automatizate au fost ulitizate: limbajul de programare JAVA, *Intellij IDEA* ca mediu de programare, framework-ul TestNG pentru stucturarea și configurarea claselor de test, *Selenium Web Driver* pentru interacțiunea cu elementele web de pe interfața aplicației și scrierea testelor automatizate, diferite biblioteci de interactiune cu interfața aplicației cât și cu fișiere externe (.xslx) pentru stocarea, și extragerea datelor de test.

În urma investigării aplicației, s-a ajuns la concluzia că orice rețea conectată la internet are un potențial ridicat de vulnerabilitate în fața anumitor atacuri sau acțiuni cu efecte distructive pentru resursele informaționale de care dispune un anumit sistem. Ceea ce privește aplicația testată în cauză, sistemul de management conținut Moodle în cadrul USM, s-a ajuns la concluzia că:

✓ Din punctul de vedere al tehnicii de atac cibernetic IDOR, și anume, controlul de acces orizontal, adică între utilizatorii cu un set de privilegii identice, în cazul dat studenții, sistemul este destul de bine

protejat. În urma a mai multor încercări de a incercepta anumite valori unice (userId-ul și valoarea MoodleSessionusmmd din cookie) care sunt obținute de către utilizator odată ce s-a logat în aplicație, și de a le modifica sau înlocui cu aceleași valori ale unui alt utilizator, nu s-a reușit obținerea accesului la datele personale ale utilizatorului neautentificat – TEST PASSED.

✓ Din punctul de vedere al tehnicii de atac cibernetic XSS aplicația nu interpretează cod javascript în interiorul adresei URL sau în interiorul unui input field – TEST PASSED.

✓ Din punctul de vedere al vulnerabilității de tip *Directory Traversal*, nu a fost posibilă accesarea unor fișiere păstrate pe server – TEST PASSED.

✓ Din punctul de vedere al tehnicii de atac cibernetic CSRF – aplicația nu reacționează într-un mod alarmant la expedierea request-urilor modificate (în cazul de față request de modificare a unor date personale ale unui utilizator neautentificat (ex.: Nume, Prenume, Email).

Însă în acelasi timp s-au depistat unele puncte vulnerabile ce țin de asigurarea securității utilizatorilor de sistem, cum ar fi:

o Informații confidențiale transmise în URL, prin intermediul GET request-uri (*sesionId, userId, session tokens*).

o Informații confidențiale transmise și afișate în răspunsurile aplicației (*sessioKey* alături de timpul de expirare a sesiunii).

Recomandări:

✓ A evita transmiterea valorilor, precum: sessionId-ului, userId prin intermediul GET URL.

✓ Utilizarea protocolului securizat HTTPS în cazul transmiterii unor parametri privați.

✓ Utilizarea POST request-urilor pentru transmiterea informațiilor confidențiale.

Recomandat
Alisa CURLICOVSCHI, magistru, asistent univ.