4.1. XSS támadások

4.1.1. Hajtson végre egy reflected XSS támadást! Keressen az alkalmazásban olyan bemenetet, amely tartalma a szerverrel megvalósított kommunikáció során megjelenik a válaszban is. Segítség: A sérülékenység a megrendelések nyomon követése szolgáltatásban van. (Ezt a korábbi rendelések [Order History] oldalon a kis teherautó ikonra kattintva lehet elérni.) Hajtson végre XSS támadást, amely során egy HTML elemet szúr be a weboldalba (lásd korábban részletezett várt eredmény)!

Folyamat:

Eloszor is admin szintu felhasznaloval beleptem SQL injectiont hasznalva a bejelentkezesi feluleten. Felhasznalo: ‘ or 1=1;--

Navigaltam az Order History oldara es rakatintottam a teherautora.

Probalkoztam beszurni egy html elemet az id ertekenek a kovetkezo keppen:

id=<iframe src=”javascript:alert(‘XSS’);”>

Nem mukodik mert backticket kellet hasznalni hogy ne szakadjon meg a string.

Megoldas URL:

http://localhost:3000/#/track-result?id=%3Ciframe%20src=%22javascript:alert(%60xss%60)%22%3E

4.1.2. Hajtson végre egy DOM alapú XSS támadást! Ez a feladat nagyon hasonló az előző feladatban végrehajtotthoz. Ismételje meg a támadást a weboldal keresés funkciója ellen, majd keresse meg a forrás kódban, hogy hol található a sérülékeny kódrészlet. A keresés feldolgozását

megvalósító kódot ezen a linken keresztül éri el:

https://github.com/juice-shop/juice-shop/blob/master/frontend/src/app/search-result/search-result.component.html.

A keresesi mezoben a kevetkezovel jelent meg az alert :

<iframe src=javascript:alert(`XSS`);>

A kodban a serulekenyseg a 13-ik sorban talalhato ahol innerHTML van hasznalva searchValue-t tarolo elemen.

```
10        <div class="heading mat-elevation-z6">
11          <div *ngIf="searchValue">
12            <span>{{"TITLE_SEARCH_RESULTS" | translate}} - </span>
13            <span id="searchValue" [innerHTML]="searchValue"></span>
14          </div>
15          <div *ngIf="!searchValue">{{"TITLE_ALL_PRODUCTS" | translate}}</div>
```

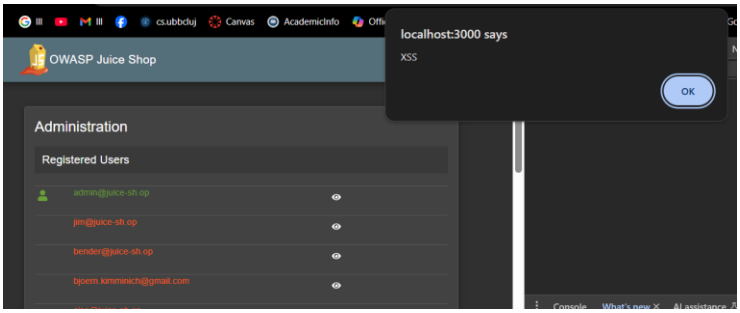
4.1.3. Hajtson végre egy persistent XSS támadást a kliens oldali védelem megkerülés után! Minden új felhasználó adata eltárolásra kerül az adatbázisban. A regisztráció folyamata során minden adat megfelelő ellenőrzése egy kritikus feladat a biztonság szempontjából. A Juice Shop alkalmazásban ezt a védelmet a kliens oldalra implementálta a fejlesztő, ami így a támadó által megkerülhető. A kliens oldalra írt védelmek a weboldal HTML tartalmának a kézzel végrehajtott módosításaival könnyen megkerülhető. Hajtson végre persistent XSS támadást, hogy demonstrálja ezt a sérülékenységet! A sikeres támadás eredményét a .../#/administration URL megnyitása után lehet látni! Ez a kihívás a webes alkalmazások egy nagyon gyakori biztonsági hibáján alapul, ahol a fejlesztők figyelmen kívül hagyták a következő aranyszabályt a bemeneti érvényesítésről: Be aware that any JavaScript input validation performed on the client can be bypassed by an attacker that disables JavaScript or uses a Web Proxy. Ensure that any input validation performed on the client is also performed on the server. Segítség: a sebezhetőség a regisztrációs űrlapon található. Az űrlap DOM-fájának manipulálásával megkerülhető a regisztráció “kliens oldali biztonsága”.

Folyamat:
Probaltam egy script elemet irni az email mezobe de nem jartam sikerrel mivel felismeri hogy nem valos email cim.

Sikerult megkerulnom a frontend validaciot a Bongeszo API-al.

A kovetkezo scriptet hasznaltam:

Az emailcimem a kovetkezo volt : <iframe src=javascript:alert(`XSS`);>



4.2. SQL injection 4.2.1. Szerezzen adminisztrátori hozzáférést a rendszerhez! Találjon egy sérülékeny bemenetet, ahol a hibát kihasználva képes az adminisztrátor nevében bejelentkezni. Bár a feladat megoldható a jelszó kitalálásával is, a cél egy SQL injection végrehajtása. Segítség: A bejelentkezést végző szerver oldali kód az alábbi linken

A megoldas amit talatam hogy egy ‘ karakter kezdem az email cimet hogy bezarjam a karakterlancot majd egy or <egy igaz kijelentes> bezarva a komment karakterel --

peldaul : ‘ or 1=1;--

4.2.2. Hagyja a szervert végre pihenni egy kicsit! A megoldandó feladat arról szól, hogy a folyamatos terhelés után végre pihenhessen a szerver egy kicsit, ami a felhasználók szempontjából egy leegyszerűsített denial-of-service támadásként érzékelhető. A Juice Shop web alkalmazás egy MongoDB nevű NoSQL adatbázist használ információk tárolására. Ebben található többek között az egyes termékek értékelése. A feladat során először az értékelések betöltéséhez használt hálózati kommunikáció megkeresése szükséges, majd a kérésben található sérülékenység megtalálása után, a támadás végrehajtása. A támadáshoz a MongoDB által használt sleep függvény használata javasolt, melynek dokumentációja megtalálható itt: https://docs.mongodb.com/manual/reference/method/sleep/. Segítség: A szerver oldali kód, amely kiszolgálja a kéréseket megtalálható ezen a linken: https://github.com/juice-shop/juice-shop/blob/master/routes/showProductReviews.ts. Megjegyzés: a szerver elárasztása nagy számú kérésekkel nem fogja megoldani ezt a feladatot. Ezzel valószínűleg csak a teljes környezet túlterhelését lehet elérni, ami nem számít helyes megoldásnak.

A keres amivel az ertekeleseket keri le igy nez ki:

http://localhost:3000/rest/products/:id/reviews

A backend kod az id beleszurja egy mongo scriptbe semmifele ellenorzes nelkul, ebbol ha az id-t atirjuk valamilyen feltetel sorozata meghivhatjuk a sleep fuggveny ahanyszor csak akarjuk. Ugyan a kodban le van limitalva az hogy milyen hosszu ideig tudunk sleep-t hivni.

peldaul az id legyen 1 || sleep(2000) || sleep(2000) || ... || true

http://localhost:3000/rest/products/1%20%7C%7C%20sleep(2000)%20%7C%7C%20sleep(2000)%20%7C%7C%20true/reviews

4.2.3. Jim jelszavának a megváltoztatása az ‘Elfelejtett jelszó’ funkció segítségével. Használja az alkalmazásba épített elfelejtett jelszó funkciót, hogy módosítsa Jim (jim@juicesh.op) jelszavát. A jelszó helyreállítás rendelkezik egy beépített védelemmel, amely az engedély nélküli módosításokat hivatott megállítani: egy biztonsági kérdésre kell válaszolni. Ezt a védelmet azonban social engineering támadás segítségével ki lehet játszani. Jim az egyik lehető legrosszabb biztonsági kérdést választotta a regisztrációja során. Ez, azzal együtt, hogy Jim egy híresség semmilyen védelmet nem nyújt a felhasználói fiókja számára, mivel a kérdésre a választ ki lehet találni. A feladat megoldásához a következő lépések javasoltak: Segítség: • Keresse meg az ‘Elfelejtett jelszó’ funkciót a web alkalmazásban! • Jim e-mail címének a megadása után jegyezze meg Jim biztonsági kérdését! • Keressen olyan Juice Shop termékeket amelyekhez írt Jim értékelést! A második oldalon érdemes keresgélni. Az így talált információ árulkodhat Jim személyéről. • Az első feladatból ismert SQL injection módszerhez hasonlóval jelentkezzen be Jim nevében, majd keresse meg az általa megadott szállítási címeket! • Az értékelésekből és a címekből megszerzett információk alapján derítse ki az interneten, hogy ki Jim valójában. • A Jimről található elérhető információkból válaszolja meg a korábban megtalált biztonsági kérdést!

Ahhoz hogy bejelentkezek jim neveben az email utan hozzafuztem a kovetkezot: ‘--

Jim idenitasa : **James Tiberius Kirk** es a valasz a biztonsagi kerdesre Samuel