Alati za testiranje sigurnosti u okruženju otvorenog koda

Vukušić, Ivan

VELEUČILIŠTE U BJELOVARU
STRUČNI PRIJEDIPLOMSKI STUDIJ RAČUNARSTVO

Alati za testiranje sigurnosti u okruženju otvorenog koda

Završni rad br. 02/RAČ/2024

Ivan Vukušić

Bjelovar, svibanj 2024.

obrazac ZR - 001

Veleučilište u Bjelovaru

Trg E. Kvaternika 4, Bjelovar

1. DEFINIRANJE TEME ZAVRŠNOG RADA I POVJERENSTVA

Student: **Ivan Vukušić** JMBAG: **0314020694**

Naslov rada (tema): **Alati za testiranje sigurnosti u okruženju otvorenog koda**

Područje: **Tehničke znanosti** Polje: **Računarstvo**

Grana: **Informacijski sustavi**

Mentor: **Dario Vidić, mag. ing. el. techn. inf.** zvanje: **viši predavač**

Članovi Povjerenstva za ocjenjivanje i obranu završnog rada:

1. **Krunoslav Husak, dipl. ing. rač., predsjednik**
2. **Dario Vidić, mag. ing. el. techn. inf., mentor**
3. **Ivan Sekovanić, mag. ing. inf. et comm. techn., član**

2. ZADATAK ZAVRŠNOG RADA BROJ: 02/RAČ/2024

U sklopu završnog rada potrebno je:

1. Predložiti i opisati načine zaštite od napada SQL umetanjem alatom otvorenog koda SQLMap
2. Opisati implementaciju sigurnosnih zakrpa
3. Opisati postupak otkrivanja ranjivosti web aplikacija alatom otvorenog koda OWASP ZAP
4. Pokazati načine prevencija izvršavanja napadačkog koda (engl. Cross-site Scripting) alatom otvorenog koda XSStrike
5. Analizirati sigurnost mrežne infrastrukture

Datum: 27 ožujka 2024. godine

Mentor: **Dario Vidić, mag. ing. el. techn. inf.**

REPUBLIKA HRVATSKA VELEUČILIŠTE U BJELOVARU BJELOVAR

Sadržaj

1. UVOD

Dnevno se u svijetu dogodi oko 2200 napada na računalne sustave i aplikacije. Napredovanjem tehnologije napreduju alati i sustavi koji su potrebni za izvršavanje određenih napada, no isto tako napreduju i alati koji služe za detektiranje sigurnosnih propusta kako bi se ti nedostatci u sustavu sanirali. Tijekom osmišljavanja i implementiranja aplikacije, teško je predvidjeti sve moguće napade, a u slučaju da do istih dođe, potrebno ih je brzo detektirati i efikasno pristupiti tom problemu jer šteta koja nastaje napadima može biti velika. U sklopu ovog završnog rada osmišljena je i implementirana ranjiva aplikacija, čiji je cilj simulirati napade i isprobati rad alata za testiranje i otkrivanje ranjivosti. Korišten je operacijski sustav Linux, točnije Kali Linux jer je opremljen alatima potrebnima za sigurnosno testiranje. Dodatna prednost ovog operacijskog sustava je što je on otvorenog koda. To omogućuje širokoj zajednici da pomogne u otkrivanju i ispravljanju raznih propusta, što ga čini vrlo sigurnim i pouzdanim. Alati koji su korišteni i isprobani na spomenutoj aplikaciji su OWASP ZAP, SQLMap i XSStrike te su predstavljene njihove karakteristike. Na početku rada definirani su općeniti pojmovi vezani za web sigurnost. U drugom dijelu opisane su ranjivosti web aplikacija, njihove posljedice i prevencija. Zatim su opisani koraci testiranja, alati za testiranje sigurnosti aplikacija, njihove prednosti i nedostaci. U sljedećem dijelu opisana je ranjiva aplikacija, tijek napada izvršenih pomoću nje te korištenje alata koji su otkrili te ranjivosti. Na kraju slijedi zaključak i sažetak rada.

2. UVOD U *WEB* SIGURNOST I OTVORENO OKRUŽENJE

U doba digitalnih inovacija i široke upotrebe interneta, zaštita internetskog okruženja postala je ključni element digitalnog krajolika. Web aplikacije, ključne za digitalno iskustvo, nude brojne prednosti i mogućnosti, uz prepreke i rizike. Neophodno je dublje proniknuti u temeljne koncepte i načela web sigurnosti kako bi se naglasilo njezino značenje.

2.1 Web aplikacije: temelj digitalnog doba

Web aplikacije djeluju kao poveznica između digitalnog svijeta i osobe koja ju koristi odnosno korisnika. Bez obzira na to jesu li to društvene mreže, bankarski portali, online trgovine ili neka druga web aplikacija, one su neizostavni dio svakodnevnog života. Ove digitalne platforme omogućuju korisnicima da pristupaju neograničenoj količini informacija, komuniciraju, kupuju i obavljaju ostale usluge na jednostavan način putem web preglednika ili mobilnih uređaja. Pojava web aplikacija promijenila je način na koji doživljavamo informacije, olakšala razmjenu podataka i označila važan korak naprijed u digitalnom dobu. Može se s uvjerenjem tvrditi da su web aplikacije temelj moderne digitalne interakcije.

2.2 Izazovi web sigurnosti

Web aplikacije, unatoč svojim prednostima, osjetljive su na niz prijetnji i ranjivosti. Sigurnosni rizici poput hakerskih napada i zlonamjernih aktivnosti često su usmjereni na web okruženja s ciljem dobivanja neovlaštenog pristupa, uzrokovanja gubitka podataka ili prekida rada. Prepoznavanje ovih prijetnji ključno je za održavanje sigurnosti i stabilnosti online prostora.

2.3 Ciljevi web sigurnosti

Ciljevi web sigurnosti su očuvanje integriteta, njihove povjerljivosti i dostupnosti informacija i mrežnih izvora netaknutima. Osigurava čvrsto povjerenje korisnika u web aplikacije, štiti osjetljive podatke kako bi spriječio neodobreni pristup i štetu od prijetnji. Kako bi se temeljni koncepti i ciljevi web sigurnosti postavili kao početna točka, proučit će se i provesti u djelo sigurni koraci istraženi u ovom radu. Istraživanje koje je u tijeku razmotrit

će alate za sigurnosno testiranje Linuxa za bolju obranu web mjesta od mogućih opasnosti. [1]

2.4 Otvoreno okruženje

Otvoreni softver je softver čiji izvorni kod može pregledati, mijenjati i poboljšavati bilo tko. "Izvorni kod" je dio softvera koji većina korisnika računala nikada ne vidi; to je kod koji programeri mogu manipulirati kako bi promijenili način rada softvera - "programa" ili "aplikacije". Programeri koji imaju pristup izvornom kodu programa mogu poboljšati taj program dodavanjem značajki ili ispravljanjem dijelova koji ne rade ispravno. Neki softver ima izvorni kod koji može mijenjati samo osoba, tim ili organizacija koja ga je stvorila i ima ekskluzivnu kontrolu nad njim. Ljudi nazivaju ovakav softver "vlasničkim" ili "zatvorenim izvorom" softvera. Samo su originalni autori vlasničkog softvera zakonski ovlašteni kopirati, pregledavati i mijenjati taj softver. Kako bi koristili vlasnički softver, korisnici računala moraju se složiti (obično potpisivanjem licence pri prvom pokretanju tog softvera) da neće raditi ništa s softverom što autori softvera nisu izričito dopustili. Microsoft Office i Adobe Photoshop su primjeri vlasničkog softvera. [14]

3. RANJIVOSTI WEB APLIKACIJA

Aplikacije igraju ključnu ulogu u našim svakodnevnim rutinama. Unatoč sve većoj popularnosti, oni također predstavljaju izazove zbog ranjivosti koje napadači mogu iskoristiti kako bi oštetili podatke ili dobili neovlašteni pristup. Ovo poglavlje bavi se specifičnostima i ispitivanjem ranjivosti web aplikacija kako bi se poboljšalo razumijevanje njihovih karakteristika i posljedica.

Internet nije otporan na razne ranjivosti koje često kategoriziraju kao napadi ili specifične ranjivosti. Među najčešćim slabostima ističu se sljedeće slabosti koje su opisane u sljedećim podpoglavljima.

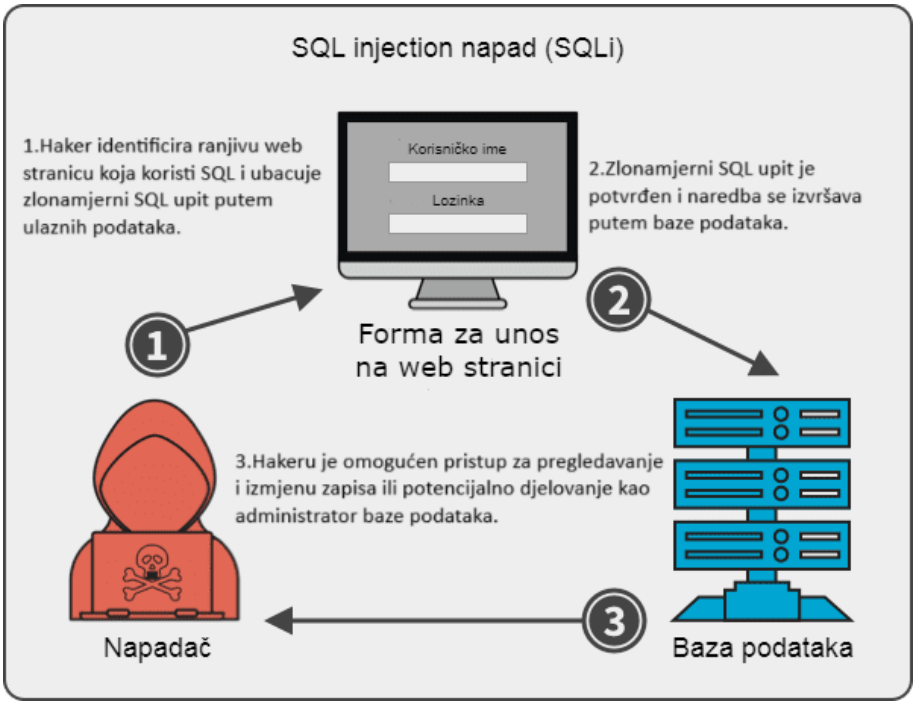
3.1 SQL *injection*

SQL *injection* (SQLi) ozbiljna je ranjivost koja prijeti sigurnosti web aplikacija.

Ove moderne metode hakiranja omogućuju napadačima neovlašteni pristup pohrani web aplikacija i mogu dovesti do ozbiljnih posljedica ako nisu osigurane. [2]

Nije uvijek lako otkriti i popraviti ranjivosti u web aplikacijama. Brz razvoj, različite tehnologije i promjenjivi uvjeti napada otežavaju ovaj proces. Slabe točke u struji pojavljuju se nakon uporabe, povećavajući rizik i ozbiljnost rizika. Nadolazeći odjeljci će pokazati korištenje alata za sigurnosno testiranje Linuxa za otkrivanje i rješavanje ranjivosti u web aplikacijama. Proučavajući primjere iz stvarnog života i analize, dobit će se uvid u sigurnosne prakse i strategije za prevladavanje povezanih izazova.

Posljedice SQL *injectiona* mogu biti značajne. Haker može doći do osobnih podataka poput korisničkih imena, lozinki, financijskih podataka i drugih privatnih podataka koji se čuvaju u bazi podataka. Štoviše, upad može rezultirati izmjenom ili brisanjem podataka u bazi podataka, što dovodi do nereda i značajno narušava integritet aplikacije.



Slika 3.1: Prikaz SQLi napada

Izvor: Spanning.com [7]

3.1.1 **Varijacije SQL *injection* napada**

Iz izvora [15] saznaje se da napadi SQL injekcijom dolaze u mnogim oblicima, kao što su:

- ***Union-based* SQL *injection***: Događa se kada napadač koristi operaciju UNION za kombiniranje podataka iz više tablica unutar baze podataka.
- ***Blind* SQL *injection***: U scenariju ove SQL injekcije napadači ne mogu izravno vidjeti ishode svog SQL upita, ali mogu zaključiti informacije iz načina na koji aplikacija odgovara.
- ***Time-based blind* SQL *injection***: Haker čeka određena vremenska razdoblja da prikupi detalje baze podataka
- ***Out-of-band* SQL *injection***: Događa se kada haker razgovara s vanjskim poslužiteljem radi slanja podataka.

3.1.2 **Prevencija i zaštita**

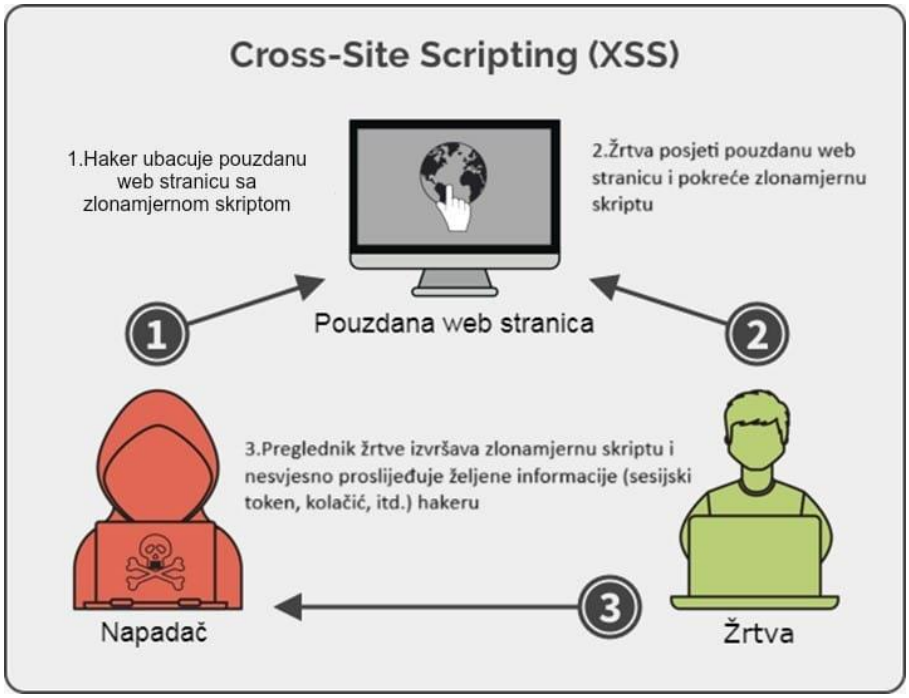
Zaštita od slabosti SQL *injection* je ključna, iz izvora [16] preventivne mjere su:

- **Korištenje gotovih izjava:** Umjesto da se ručno postavljaju SQL upiti s onim što unose korisnici koriste se unaprijed pripremljeni parametrizirani upiti.
- **Test unosa i provjere:** Detaljna provjera i zaslon korisničkog unosa koji sprječavaju da se loši znakovi ili naredbe ušuljaju.
- **Načelo minimalnih privilegija:** Ograničavanje prava pristupa bazi podataka na potreban minimum za pokretanje aplikacije.
- **Ažuriranje i testiranje sigurnosti:** Držanje web aplikacije ažurnom i provođenje sigurnosnih provjera za pronalaženje i ispravljanje slabih točaka.

Suočavanje sa SQL injekcijom zahtijeva marljiv pristup zaštiti sigurnosti web aplikacija. Ulaganje u preventivne mjere i edukacija o ovoj ranjivosti može uvelike poboljšati sigurnost aplikacija i održati povjerenje korisnika.

3.2 *Cross-Site Scripting* (XSS)

Cross-Site Scripting (XSS) je prevladavajuća i ozbiljna ranjivost pronađena u web aplikacijama koja napadačima omogućuje umetanje štetnog JavaScript koda na web stranicu, koja se izvodi u korisničkom pregledniku. Ova ranjivost može dovesti do ugrožavanja sigurnosti i povjerljivosti podataka.



Slika 3.2: Primjer Cross-Site Scripting (XSS) napada

Izvor: Spanning.com [8]

3.2.1 **Kako XSS funkcionira?**

XSS napad događa se kada web aplikacija ne provodi provjeru ili ne očisti korisnički unos prije nego što ga prikaže na web stranici. To omogućuje napadaču da postavi loš JavaScript kod koji se pokreće kada korisnik posjeti zaraženu stranicu. Napad se može pohraniti, reflektirati ili temeljiti na DOM-u, ovisno o tome kako napadač ubacuje i koristi zlonamjerni kod. [3]

3.2.2 **Potencijalne posljedice**

Posljedice XSS ranjivosti mogu biti ozbiljne, a najopasnija je krađa podataka s bankovne kartice. Prema izvoru [17] posljedice XSS ranjivosti su:

- **Krađa kolačića (*Cookie hijacking*)**: Haker se može domoći nečijih kolačića koji sadrže podatke o prijavi, a zatim provaljuje u korisnički račun.
- **Preusmjeravanje na zlonamjerne stranice**: Korisnici mogu biti poslani na loše ili lažne stranice gdje njihovi privatni podaci mogu biti ukradeni.
- **Manipulacija sadržajem stranice**: Napadač može promijeniti sadržaj web stranice, može staviti pogrešne podatke ili izmisliti lažne obrasce podataka za prikupljanje informacija.
- **Napadi na korisnike**: XSS dopušta napadačima da izvršavaju različite neželjene poteze kao što su slanje neugodnih poruka, objavljivanje privatnih podataka ili čak obavljanje bankovnih transakcija bez znanja korisnika.

3.2.3 **Prevencija i zaštita**

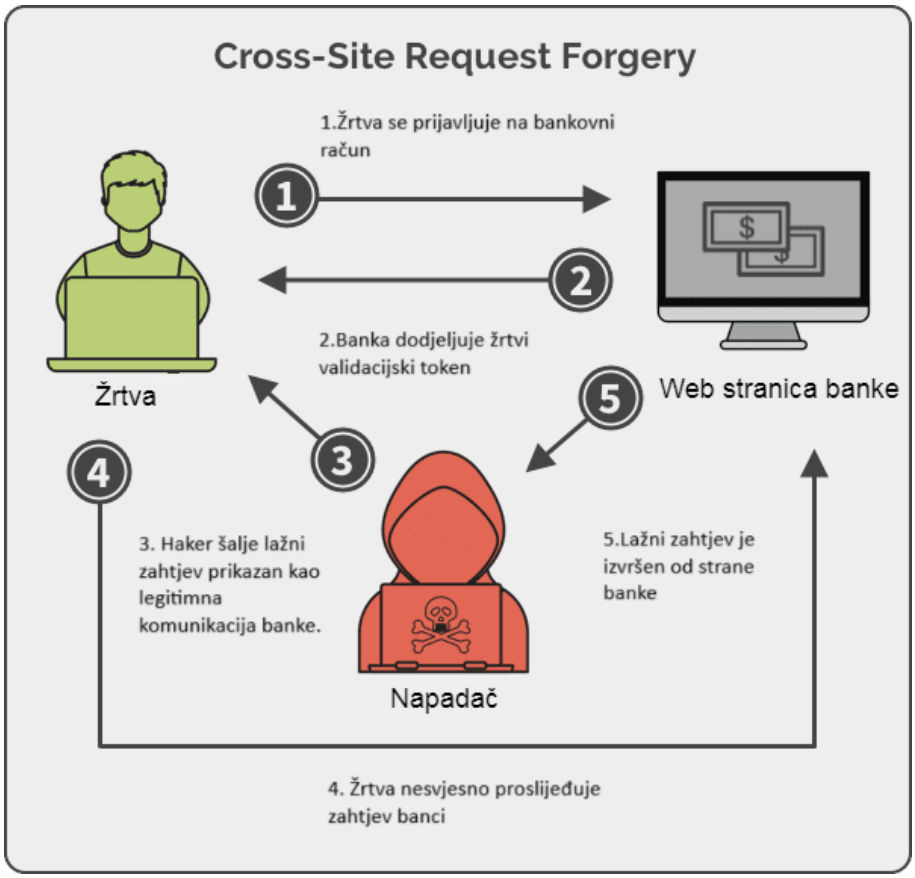
Za zaustavljanje XSS napada potreban je sustavan način za izradu sigurnih web aplikacija. Izvor [17] sadrži načine zaštite i prevencije, a to su:

- **Korištenje *encodinga***: U svrhu sprječavanja pokretanja zlonamjernih kodaova treba voditi brigu o pravilnom postupanju prilikom kodiranja i pravilno koristiti konverziju specijalnih znakova u njihove sigurne ekvivalente prije nego što se korisnički unos prikaže u web aplikaciji

- **Implementacija sigurnosnih *headera***: Sigurnosna HTTP zaglavlja kao što su *Content-Security-Policy* (CSP) i *X-XSS-Protection* pomažu u zaustavljanju XSS napada.
- **Upotreba sigurnosnih biblioteka**: Iskorištavanje postojećih sigurnosnih biblioteka za dodatnu prevenciju i zaštitu podataka.
- **Obrazovanje razvojnog tima**: Održavanje sigurnosti se temelji i na edukaciji tima koji razvija sustav. Vrlo je bitno održavati redovne edukacije i provoditi sigurnosna testiranja aplikacije.

3.3 *Cross-site Request Forgery* (CSRF)

Cross-Site Request Forgery (CSRF) događa se kada napadač prevari prijavljenog korisnika da nesvjesno izvede neželjenu radnju na web aplikaciji. Koristeći podatke o autentifikaciji korisnika, poput kolačića sesije, napadač šalje zlonamjerni zahtjev aplikaciji.



Slika 3.3: Primjer Cross-Site Request Forgery (CSRF) napada

Izvor: Spanning.com [9]

3.3.1 **Funkcioniranje CSRF-a**

CSRF napadi obično započinju kada korisnik stupi u interakciju sa zlonamjernim web mjestom ili vezom putem e-pošte koja automatski izvršava određene radnje na ciljanoj web aplikaciji, poput promjene lozinke ili obavljanja financijske transakcije. Napadač koristi taktike društvenog inženjeringa kako bi prevario korisnika da pristupi zlonamjernom sadržaju. Iskorištavanjem korisničkih pojedinosti o autentifikaciji pohranjenih u kolačićima sesije, napadač može natjerati web aplikaciju da nesvjesno izvrši njihove nedopuštene upute. CSRF zaštita uključuje uključivanje CSRF tokena, koji su jedinstveni identifikatori generirani za svaki zahtjev kako bi se potvrdila legitimnost korisnika i spriječili nelegitimni zahtjevi. [6]

3.3.2 **Potencijalne posljedice CSRF napada**

CSRF napadi mogu imati ozbiljne posljedice kako po korisnike, tako i po same web aplikacije. U izvoru [18] nalaze se moguće posljedice CSRF napada:

- **Gubitak povjerenja korisnika:** Može se dogoditi kada korisnici otkriju da se njihove interakcije na web aplikaciji mogu koristiti bez njihovog pristanka, što dovodi do sumnje u sigurnost i privatnost aplikacije.
- **Financijski gubici**: CSRF napadi mogu dovesti do financijskih gubitaka za korisnike, osobito kada uključuju financijske transakcije ili mijenjanje osjetljivih podataka poput lozinki ili adresa za dostavu.
- **Povrede privatnosti**: Sigurnosne povrede koje napadaču omogućuju dobivanje osobnih ili osjetljivih korisničkih podataka, uključujući adrese, podatke za kontakt ili medicinske podatke, mogu dovesti do značajnih povreda privatnosti.
- **Oštećenje reputacije**: Web aplikacije koje često postaju žrtve CSRF napada mogu pretrpjeti reputaciju korisnika i javnosti, što dovodi do trajnih posljedica na njihov tržišni uspjeh i konkurentnost.
- **Pravne posljedice**: Web aplikacije i njihovi vlasnici mogu se suočiti s pravnim posljedicama, poput tužbi ili regulatornih kazni, ako CSRF napad uzrokuje štetu ili krši privatnost korisnika.
- **Održavanje integriteta podataka**: CSRF napadi mogu ugroziti integritet podataka sustava mijenjanjem ili brisanjem korisničkih podataka ili neovlaštenim izmjenama web stranica.

- **Smanjenje korisničkog angažmana**: Ako korisnici primijete CSRF napad na web aplikaciju, manja je vjerojatnost da će koristiti aplikaciju ili će ju potpuno prestati koristiti, što dovodi do manjeg angažmana korisnika i mogućeg gubitka prihoda od oglašavanja ili prodaje.

3.3.3 **Prevencija i zaštita**

Prevencija CSRF napada ključna je za održavanje sigurnosti web aplikacija i zaštitu korisnika od potencijalnih prijetnji. Iz izvora [18] slijedi nekoliko preporučenih praksi i tehnika za sprječavanje CSRF napada:

- **Upotreba CSRF tokena**: Implementacija CSRF tokena jedna je od najčešćih i najučinkovitijih metoda prevencije. CSRF tokeni su jedinstveni identifikatori koji se generiraju prilikom svakog zahtjeva i šalju korisniku kao skriveni parametar u formama ili kao dio zaglavlja HTTP zahtjeva. Web aplikacija zatim provjerava valjanost CSRF tokena prije obrade zahtjeva.
- **Provjera *referera***: Provjera HTTP *referera* (*referer header*) može se koristiti kao dodatni sloj sigurnosti za otkrivanje CSRF napada. Web aplikacija provjerava je li polje *referer* u zahtjevu odgovarajuće, tj. dolazi li s iste domene kao i web aplikacija.
- **Implementacija *SameSite* atributa za kolačiće**: *SameSite* atribut za kolačiće omogućava web aplikaciji da ograniči slanje kolačića samo na zahtjeve koji dolaze s iste domene. Postavljanjem *SameSite* atributa na "*Strict*" ili "*Lax*" može se smanjiti rizik od CSRF napada.
- **Korištenje HTTP metoda s ograničenjima**: Korištenje HTTP metoda poput POST i PUT za osjetljive operacije umjesto GET može smanjiti rizik od CSRF napada. Osim toga, potrebno je ograničiti izvršavanje osjetljivih operacija samo na zahtjeve koji dolaze s odgovarajućim CSRF tokenom.
- **Redovito ažuriranje sigurnosnih praksi**: Web programeri i administratori trebaju redovito pratiti najnovije sigurnosne smjernice i prakse te ažurirati web aplikacije i radne okvire kako bi održali visoke standarde sigurnosti i spriječili nove vrste napada.
- **Korištenje sigurnosnih alata**: Korištenje alata za skeniranje ranjivosti poput OWASP ZAP ili Burp Suite može pomoći u otkrivanju potencijalnih CSRF ranjivosti u web aplikacijama te u identifikaciji i ispravljanju sigurnosnih propusta.

4. UVOD U LINUX ALATE ZA SIGURNOSNO TESTIRANJE

Alati za testiranje sigurnosti Linuxa stvoreni su za pronalaženje i proučavanje mogućih slabih točaka u aplikacijama web mjesta i sustavima na kojima rade kako bi sigurnosni stručnjaci i administratori mogli djelovati prije nego što se nešto loše dogodi.

4.1 Identifikacija mete

Za početak rada s alatom za sigurnosno testiranje Linuxa, korisnici trebaju točno odrediti što testiraju, bilo da se radi o određenoj mreži web aplikacija ili o cijeloj infrastrukturi. To je ključno jer im omogućuje da usmjere svoj test na određeni resurs. U izvoru [19] nalaze se koraci pri identifikaciji mete:

- **Analiza površinskih informacija o meti*:*** U ovom koraku istraživanja, fokusira se na prikupljanje površinskih informacija o cilju testiranja. To može uključivati pregled web stranica, društvenih mreža, javnih baza podataka ili bilo kojeg drugog javno dostupnog izvora informacija o meti.
- **Skupljanje tehničkih podataka o infrastrukturi:** Ovaj korak obuhvaća prikupljanje tehničkih podataka o infrastrukturi cilja, kao što su IP adrese, DNS zapisi, informacije o web poslužitelju, otvoreni portovi i slično. Analizom ovih tehničkih podataka može se bolje razumjeti arhitektura i osnovna sigurnosna postavka ciljanog sustava.
- **Identifikacija ranjivih točaka na web aplikaciji:** U ovom dijelu fokus je na identifikaciji potencijalnih ranjivih točaka na web aplikaciji koja se nalazi unutar mete. To može uključivati pregled izvornog koda, analizu HTTP zahtjeva i odgovora, provjeru konfiguracije poslužitelja i druge tehnike koje pomažu u otkrivanju potencijalnih sigurnosnih propusta.
- **Procjena razine sigurnosti i ranjivosti:** Nakon prikupljanja informacija i identifikacije ranjivih točaka, provodi se procjena razine sigurnosti i ranjivosti ciljanog sustava. Ova procjena uključuje analizu prikupljenih podataka kako bi se utvrdila ozbiljnost otkrivenih ranjivosti i potencijalnih sigurnosnih prijetnji.

4.2 Skeniranje i analiza

Nakon što se identificira cilj, koriste se alati za skeniranje i analizu ranjivosti:

- Otkrivanje otvorenih mrežnih priključaka i usluga.

- Identificiranje ranjivih aplikacija i njihovih verzija.
- Provjera postavki sustava u svrhu identifikacije mogućih sigurnosnih ranjivosti.

U ovim koracima objašnjava se kako funkcionira skeniranje i što je potrebno poduzeti nakon identifikacije. U izvoru [19] nalaze se koraci:

- **Automatizirano skeniranje mrežnih resursa:** U ovom koraku korišteni su alati za automatizirano skeniranje mrežnih resursa kako bi se identificirali dostupni sustavi, otvoreni portovi, servisi i drugi resursi unutar ciljane mreže. Ovi alati omogućuju brzo otkrivanje potencijalnih mete za daljnju analizu sigurnosti.
- **Identifikacija aktivnih servisa i ranjivosti*:*** Nakon što su mrežni resursi identificirani, usredotočuje se na identifikaciju aktivnih servisa i potencijalnih ranjivosti koji bi mogli biti prisutni. Ovo uključuje korištenje alata za otkrivanje ranjivosti i analizu ranjivih servisa kako bi se utvrdile moguće sigurnosne prijetnje.
- **Analiza otvorenih portova i konfiguracije poslužitelja*:*** Daljnji korak u procesu skeniranja i analize je detaljna analiza otvorenih portova i konfiguracije poslužitelja. Pregledavaju se dostupni servisi na otvorenim portovima, provjeravaju se postavke sigurnosti i identificiramo potencijalne ranjivosti koje proizlaze iz loše konfiguracije poslužitelja.
- **Skeniranje ranjivosti web aplikacija*:*** Osim skeniranja mrežne infrastrukture, važno je provesti i skeniranje ranjivosti web aplikacija koje se nalaze unutar ciljane mreže. Koriste se alati za otkrivanje ranjivosti web aplikacija kako bismo identificirali moguće ranjivosti poput SQL injekcija, XSS-a, CSRF-a i drugih sigurnosnih propusta.
- **Analiza rezultata skeniranja i priprema izvještaja*:*** Konačno, nakon što se provede skeniranje i analiza, analiziraju se rezultati kako bi se razumjeli sigurnosni rizici i pripremili detaljni izvještaji o otkrivenim ranjivostima i preporučenim koracima za ispravak. Ovi izvještaji ključni su za informiranje donositelja odluka i timova za sigurnost o potencijalnim prijetnjama i koracima za poboljšanje sigurnosti sustava.

4.3 Testiranje ranjivosti

Za testiranje ranjivosti koriste se alati koji su napravljeni za precizan pronalazak sigurnosnih ranjivosti i nedostataka. To uključuje:

- Pokušaj napada SQL injekcijom.
- Testiranje ranjivosti *Cross-Site Scripting* (XSS).
- Procjenu koliko su podaci osjetljivi i tko ih smije dobiti.

Kako bi ispravno odradili testiranje ranjivosti, potrebno je proći kroz određene korake. U izvoru [19] nalaze se koraci:

- **Aktivno ispitivanje sigurnosnih propusta:** Tijekom ove faze koristi se niz tehnika i alata za aktivno ispitivanje sigurnosnih slabosti s određenim ciljevima. To uključuje provođenje ručnih testova na aplikacijama kako bi se odredile i iskoristile potencijalne ranjivosti kao što su SQL *injection*, XSS, CSRF, između ostalih.
- **Provjera otpornosti na napade:** Osim identifikacije ranjivosti, provjeravaju se i otpornosti sustava na stvarne napade. Koriste se različite tehnike napada kako bi se testirali obrambeni mehanizmi i sigurnosne politike te provjerili jesu li sustavi u stanju zaštititi se od potencijalnih prijetnji.
- **Testiranje autentikacije i autorizacije:** Ispitivanje mehanizama provjere autentičnosti i autorizacije naglašeno je kako bi se osiguralo da samo ovlašteni korisnici mogu pristupiti osjetljivim resursima i funkcijama sustava. To uključuje pregled sigurnosnih pravila, procjenu složenosti lozinke i procjenu zaštite sesije.
- **Ispitivanje otpornosti na DDoS napade:** Uz tradicionalno testiranje ranjivosti, provjerava se i otpornost sustava na distribuirane napade usluge (DDoS). Koriste se različite tehnike simuliranja DDoS napada kako bi se testirale performanse i stabilnost sustava pod velikim opterećenjem.
- **Analiza rezultata testiranja i priprema izvještaja:** Nakon što se provede testiranje ranjivosti, analiziraju se rezultati kako bi se razumjeli sigurnosni rizici i pripremili detaljni izvještaji o otkrivenim propustima i preporučenim koracima za ispravak. Ovi izvještaji ključni su za informiranje donositelja odluka i timova za sigurnost o potencijalnim prijetnjama i koracima za poboljšanje sigurnosti sustava.

4.4 Izvještavanje

Nakon otkrivanja ranjivosti, Linux alati generiraju izvješća koja ocrtavaju identificirane probleme, omogućujući stručnjacima za sigurnost da shvate prirodu ranjivosti i implementiraju potrebna rješenja. Izvor [19] sadrži korake u izvještavanju:

- **Struktura izvještaja:** Prvi korak u izvještavanju je definiranje strukture izvještaja. Ovdje se određuje koje informacije trebaju biti uključene u izvještaj, kako će biti organizirane i koja će biti njihova međusobna povezanost. Strukturirani izvještaji olakšavaju razumijevanje sigurnosnih rizika i preporučenih koraka za ispravak.
- **Detalji o otkrivenim ranjivostima:** U ovom dijelu izvještaja detaljno se opisuju sve otkrivene ranjivosti tijekom testiranja. To uključuje informacije o vrsti ranjivosti, njezinoj ozbiljnosti, lokaciji u aplikaciji i potencijalnim posljedicama. Dodatno, mogu se priložiti i koraci za reprodukciju ranjivosti kako bi razvojni timovi mogli brže i preciznije riješiti problem.
- **Preporučeni koraci za ispravak:** U ovom odjeljku izvješća navedene su radnje za rješavanje identificiranih ranjivosti. Te radnje sastoje se od detaljnih uputa ili preporuka za rješavanje ranjivosti i povećanje sigurnosti sustava kako bi se spriječili budući napadi. Mogu uključivati primjenu zakrpa, podešavanje konfiguracija ili promjenu arhitekture aplikacije.
- **Prioritetizacija sigurnosnih rizika:** Uz opis ranjivosti i preporučene korake za ispravak, također je važno prioritetizirati sigurnosne rizike prema njihovoj ozbiljnosti i utjecaju na sustav. Ovdje se ocjenjuje svaka ranjivost prema različitim kriterijima, poput vjerojatnosti iskorištavanja, potencijalnih posljedica i dostupnosti zakrpe, kako bi se odredio redoslijed u kojem bi ranjivosti trebale biti riješene.
- **Prezentacija izvještaja i komunikacija s dionicima:** Konačni korak u procesu izvještavanja je prezentacija izvještaja dionicima. To može uključivati sastanke s timovima za razvoj, vlasnicima aplikacija, menadžmentom ili drugim relevantnim stranama kako bismo podijelili rezultate testiranja, raspravili preporučene korake za ispravak i dogovorili se o daljnjim koracima za poboljšanje sigurnosti sustava. Komunikacija s dionicima ključna je za osiguranje razumijevanja i podrške za sigurnosne inicijative.

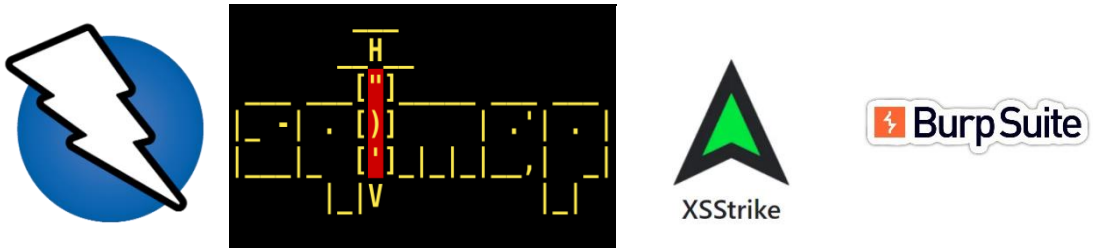
4.5 Ispravljanje i praćenje

Po primitku obavijesti o ranjivostima, administratori i razvojni timovi trebali bi odmah riješiti probleme. Ključno je pratiti i ponovno kontinuirano testirati nakon popravka kako bi se potvrdilo da su ranjivosti riješene. Izvor [19] sadrži neke od koraka pri ispravljanju i praćenju:

- **Implementacija zakrpa i promjena:** Nakon što su ranjivosti identificirane i dokumentirane, sljedeći korak je implementacija zakrpa i promjena u sustavu kako bi se otklonili sigurnosni nedostaci. Ovo uključuje primjenu sigurnosnih zakrpa, konfiguracijskih promjena, ažuriranja softvera ili čak arhitektonskih prepravki. Tim za razvoj i IT odjel surađuju u provedbi tih promjena uz poštivanje najboljih praksi sigurnosti i minimiziranje utjecaja na proizvodni sustav.
- **Testiranje zakrpa**: Nakon primjene zakrpa i promjena, ključno je provesti temeljito testiranje kako bi se potvrdila uspješna implementacija popravaka i spriječile bilo kakve neželjene posljedice ili novi sigurnosni rizici. Testiranje bi trebalo obuhvatiti funkcionalnost aplikacije, performanse sustava i sigurnost kako bi se potvrdilo da su ranjivosti učinkovito riješene i da postojeće zaštite rade kako je predviđeno.
- **Praćenje i nadzor:** Nakon ispravljanja ranjivosti, ključno je provesti kontinuirano praćenje i nadzor sustava kako bi se osiguralo da su sigurnosni propusti uspješno otklonjeni te da nema novih sigurnosnih prijetnji. Ovo uključuje praćenje logova, nadzor mrežnog prometa te redovito testiranje sigurnosti kako bi se identificirali i riješili eventualni novi rizici. Tim za sigurnost i IT odjel kontinuirano surađuju u održavanju visokih standarda sigurnosti i reaktivnom odgovoru na nove prijetnje.
- **Evaluacija učinkovitosti*:*** Kako bi se osiguralo da su poduzete sigurnosne mjere učinkovite i adekvatne, važno je redovito provoditi evaluaciju učinkovitosti sigurnosnih praksi i procesa. Ovo uključuje analizu sigurnosnih incidenta, rezultata testiranja i nadzora te povratne informacije korisnika kako bi se identificirale prilike za poboljšanje i optimizaciju sigurnosnih postupaka. Rezultati evaluacije koriste se za iterativno poboljšanje sigurnosti sustava i jačanje obrambenih mehanizama protiv budućih prijetnji.

5. ALATI ZA SIGURNOSNO TESTIRANJE

Ključno je provesti sigurnosno testiranje kako bi se otkrile i riješile ranjivosti u web aplikacijama. Korištenje odgovarajućih alata može uvelike pojednostaviti otkrivanje potencijalnih prijetnji. U nastavku su navedeni neki često korišteni alati za testiranje sigurnosti:



Slika 5.1: Vizualni identiteti korištenih alata

Izvori: Zaproxy.org [10], Wikipedia.org [11], Medium.com [12], PortSwigger.com [13]

5.1 OWASP ZAP

OWASP *Zed Attack Proxy* (ZAP) je alat za testiranje sigurnosti web aplikacija koji je otvorenog koda i razvijen od strane OWASP (*Open Web Application Security Project*). ZAP nudi brojne značajke za prepoznavanje ranjivosti i analizu sigurnosti web aplikacija. Ovaj alat omogućuje korisnicima da automatski otkriju i iskoriste različite ranjivosti, uključujući SQL *injection*, XSS, CSRF i mnoge druge, te generiraju detaljne izvještaje o pronađenim problemima. OWASP ZAP također nudi mogućnosti za ručno testiranje sigurnosti, omogućujući stručnjacima za sigurnost da precizno analiziraju web aplikacije i provjere njihovu sigurnost.

5.1.1 **Identifikacija sustava za upravljanje bazama podataka(DBMS)**

Identifikacija sustava za upravljanje bazama podataka (DBMS) u OWASP ZAP-u obično se provodi kroz niz koraka koji uključuju aktivno skeniranje web aplikacije radi pronalaženja pokazatelja koji bi mogli ukazivati na korištenje određenog DBMS-a. Alat koristi različite tehnike za ovo, uključujući analizu HTTP odgovora i upita koji se šalju prema aplikaciji, tražeći karakteristične obrasce ili podatke koji ukazuju na specifične DBMS-ove. Primjerice, prilikom skeniranja aplikacije, OWASP ZAP može analizirati HTTP odgovore kako bi identificirao greške ili karakteristične poruke koje DBMS-i često vraćaju kao odgovor na neispravne upite ili pokušaje napada. Također, alat može ispitivati upite koje

korisnik šalje aplikaciji kako bi otkrio specifične sintakse koje ukazuju na korištenje određenog DBMS-a. Dodatno, OWASP ZAP može iskoristiti ranije poznate informacije o poznatim ranjivostima ili specifičnim ponašanjima DBMS-ova kako bi bolje razumio kako se ponašaju aplikacije i na temelju toga identificirao korišteni sustav za upravljanje bazama podataka. Kroz ovaj proces, OWASP ZAP omogućuje korisnicima da dobiju jasniju sliku o tehnologijama koje se koriste u njihovim aplikacijama i potencijalnim ranjivostima povezanim s tim sustavima za upravljanje bazama podataka.

5.1.2 **Prednosti OWASP ZAP-a**

OWASP ZAP sadrži puno prednosti, jer osim što je besplatan, prema izvoru [20] on sadrži i sljedeće:

- **Automatizacija otkrivanja ranjivosti:** OWASP ZAP automatski identificira širok spektar ranjivosti u web aplikacijama, olakšavajući proces testiranja sigurnosti i štedeći vrijeme korisnicima.
- **Svestranost za različite ranjivosti:** Alat podržava identifikaciju i analizu različitih ranjivosti poput SQL *injection*a, XSS-a, CSRF-a i drugih, pružajući korisnicima sveobuhvatno rješenje za testiranje sigurnosti.
- **Dubinsko testiranje ranjivosti:** OWASP ZAP omogućuje detaljnu analizu otkrivenih ranjivosti kako bi se razumjela njihova priroda i ozbiljnost, pružajući korisnicima dublji uvid u sigurnosne propuste njihovih aplikacija.
- **Napredne opcije konfiguracije sigurnosnih politika:** Alat pruža napredne opcije za konfiguriranje sigurnosnih politika i pravila, omogućujući korisnicima da precizno definiraju parametre testiranja prema svojim potrebama.
- **Real-time upozorenja i obavijesti:** Alat pruža *real-time* upozorenja i obavijesti o otkrivenim ranjivostima, omogućujući korisnicima brzu reakciju na sigurnosne prijetnje i ispravak pronađenih propusta.

5.1.3 **Nedostaci OWASP ZAP-a**

Osim problema s dokumentacijom koju većina korisnika ima, odnosno snalaženje u istoj, postoje i drugi problemi tj. nedostaci. U izvoru [20] nalaze se nedostaci OWASP ZAP alata, a to su:

- **Potrebno iskustvo u korištenju alata:** Korištenje OWASP ZAP-a zahtijeva određeno iskustvo u području sigurnosti web aplikacija i razumijevanje njegovih funkcionalnosti kako bi se iskoristile sve njegove mogućnosti.
- **Potencijal za zloupotrebu:** Budući da alat omogućuje otkrivanje i iskorištavanje ranjivosti, postoji potencijal za zloupotrebu ako se koristi na neetički način ili bez pristanka vlasnika web aplikacije.
- **Potreba za pristankom vlasnika web aplikacije:** Prije upotrebe OWASP ZAP-a, preporučuje se dobivanje pristanka vlasnika web aplikacije kako bi se osiguralo da se testiranje sigurnosti provodi na odgovarajući način.
- **Nisu sve ranjivosti detektirane:** Unatoč širokom spektru ranjivosti koje OWASP ZAP može identificirati, mogu postojati neke ranjivosti koje alat neće otkriti, što može ograničiti njegovu učinkovitost u nekim scenarijima.
- **Mogući lažno pozitivni rezultati:** OWASP ZAP može generirati lažno pozitivne rezultate, odnosno identificirati ranjivosti koje zapravo ne postoje, što može rezultirati gubicima vremena i resursa u ispravljanju nepostojećih problema.

5.1.4 **Mogućnosti OWASP ZAP-a**

OWASP ZAP je široko korišten alat otvorenog koda koji nudi niz mogućnosti za testiranje sigurnosti web aplikacija. Kombinacija automatizacije i ručnog testiranja, integracija s drugim alatima te generiranje detaljnih izvješća čine ga dragocjenim resursom za stručnjake za sigurnost i programere. [4]

1. Proxy snimanje prometa:

- OWASP ZAP funkcionira kao posrednik između web preglednika i web poslužitelja kako bi se olakšalo hvatanje svih HTTP zahtjeva i odgovora. Ovo snimanje prometa ključno je za detaljnu analizu sigurnosnih rizika i identifikaciju potencijalnih ranjivosti, pružajući korisnicima dublji uvid u interakciju aplikacija.

2. Aktivno i pasivno skeniranje:

- Alat provodi aktivno skeniranje za automatsku provjeru ranjivosti aplikacije, dok pasivno skeniranje prati promet bez aktivnog testiranja, omogućujući otkrivanje ranjivosti analizom prometa.

3. Automatizacija i skriptiranje:

- OWASP ZAP omogućuje automatizaciju testiranja dopuštajući korisnicima da razviju prilagođene skripte ili iskoriste već postojeće za provođenje raznih sigurnosnih testova, čime se pomaže u ponovljivosti testova i besprijekornoj integraciji u radni tijek razvoja.

4. Identifikacija ranjivosti:

- Alat može otkriti različite ranjivosti poput SQL injekcija, XSS, CSRF i više, nudeći korisnicima detaljan pregled sigurnosnih rizika aplikacije.

5. Interaktivno ručno testiranje:

- OWASP ZAP stručnjacima za sigurnost pruža mogućnost ručnog testiranja aplikacija mijenjanjem zahtjeva i prometa kako bi testirali kako aplikacija reagira na različite situacije, što ga čini ključnim za sveobuhvatne sigurnosne procjene.

6. Integracija i izvješća:

- Alat se besprijekorno integrira s različitim alatima i razvojnim okruženjima, omogućujući stvaranje temeljitog testiranja i izvješća o ranjivostima. Ova značajka promovira timski rad i pomaže u donošenju informiranih sigurnosnih odluka.

5.2 SQLMap

SQLMap je neophodan za otkrivanje i popravljanje ranjivosti u web aplikacijama i povećanje njihove sigurnosti. To je moćan i fleksibilan alat za testiranje sigurnosti posebno dizajniran za web aplikacije koje koriste SQL baze podataka, omogućujući stručnjacima za sigurnost da precizno utvrde i isprave ranjivosti za bolju sigurnost web aplikacija. [5]

5.2.1 **Identifikacija sustava za upravljanje bazama podataka(DBMS)**

Kada se koristi SQLMap za identifikaciju i analizu ranjivosti, prvo se korisnik bavi postavljanjem alata. Nakon toga, specifičira se ciljani URL web aplikacije koju želi testirati. Alat automatski istražuje taj URL kako bi identificirao potencijalne ranjivosti SQL *injectiona*. Otkrivene ranjivosti zatim se detaljno analiziraju kako bi se potvrdilo njihovo postojanje i odredila njihova razina opasnosti. Ova analiza uključuje testiranje različitih tehnika SQL *injectiona* kako bi se osiguralo da su ranjivosti autentične i mogu se iskoristiti. Kada su ranjivosti potvrđene, SQLMap generira detaljne izvještaje o pronađenim ranjivostima. Ti izvještaji sadrže informacije o otkrivenim ranjivostima, tipu ranjivosti, pogođenim URL-ovima, parametrima koje je moguće iskoristiti, te preporučenim koracima

za ispravljanje. SQLMap omogućuje i prilagođavanje postavki skeniranja kako bi se usredotočio na određene dijelove web aplikacije ili kako bi se izbjegle nepotrebne provjere, čime olakšava brže otkrivanje ranjivosti i generiranje relevantnih izvještaja.

5.2.2 **Dohvat informacija i enumeracija**

Prvo se identificira ciljni sustav za upravljanje bazama podataka (DBMS) te uspostavlja komunikacija s njim. Alat zatim izvršava niz upita kako bi dohvatio relevantne metapodatke o bazi podataka i korisnicima. Enumeracija korisnika i baza podataka provodi se kroz detaljnu analizu dobivenih informacija. SQLMap koristi različite tehnike, poput izvršavanja sistemskih upita i analize metapodataka, kako bi identificirao korisničke profile, baze podataka te njihove karakteristike i privilegije. Ove informacije omogućuju stručnjacima za sigurnost bolje razumijevanje strukture baze podataka i korisničkih profila, što je ključno za učinkovito upravljanje sigurnošću.

5.2.3 **Prednosti SQLMapa**

SQLMap nudi razne mogućnosti i ovaj alat je moćan za korisnike koji se bave zaštitom. Prema izvoru [5] ovo su neke od prednosti:

- **Automatizacija otkrivanja ranjivosti:** SQLMap nudi automatizirani pristup otkrivanju ranjivosti SQL *injectiona* u web aplikacijama, olakšavajući sigurnosnim stručnjacima proces identifikacije i analize potencijalnih slabosti.
- **Svestranost za različite baze podataka:** Ovaj alat pruža podršku za različite vrste baza podataka, omogućujući stručnjacima za sigurnost da testiraju ranjivosti na različitim platformama.
- **Dubinsko testiranje ranjivosti:** SQLMap omogućuje detaljno ispitivanje i provjeru identificiranih ranjivosti, uključujući provjeru njihove ozbiljnosti i potencijalnih posljedica.
- **Prilagodljive opcije testiranja:** SQLMap nudi prilagodljive opcije za testiranje, uključujući mogućnost konfiguriranja različitih parametara skeniranja kako bi se postigao najbolji rezultat u identifikaciji ranjivosti.
- **Automatsko generiranje testnih skripti:** Alat omogućuje automatsko generiranje testnih skripti za različite scenarije SQL injekcije, što korisnicima štedi vrijeme i olakšava provođenje detaljnih testiranja.

5.2.4 **Nedostaci SQLMapa**

Iako SQLMap nema neke velike nedostatke koji će odvratiti korisnika od njegovog korištenja, no i dalje postoje bitni nedostaci. U izvoru [22] nalaze se nedostaci SQLMapa, a to su:

- **Potrebno iskustvo u SQL-u i ranjivostima:** Korištenje SQLMapa zahtjeva od korisnika određeno predznanje i iskustvo u radu s SQL jezikom te poznavanje ranjivosti SQL *injectiona* kako bi se alat koristio na učinkovit način i interpretirali rezultati testiranja.
- **Potencijal za zloupotrebu:** Kao i svaki alat za testiranje sigurnosti, postoji potencijalna opasnost od zloupotrebe SQLMapa od strane neovlaštenih korisnika za napade na web stranice i aplikacije, stoga je važno koristiti ga odgovorno i etički.
- **Potreba za pristankom vlasnika web aplikacije:** Prije nego što se SQLMap koristi za testiranje sigurnosti web aplikacija, važno je dobiti pristanak vlasnika ili administratora web stranice kako bi se osiguralo da se testiranje provodi na legitimne svrhe i bez kršenja prava.
- **Nisu sve ranjivosti detektirane:** Iako je SQLMap moćan alat za otkrivanje SQL ranjivosti, važno je imati na umu da ne može otkriti sve moguće ranjivosti u web aplikacijama, što znači da se može dogoditi da neki propusti prođu nezapaženo.
- **Lažno pozitivni rezultati:** Ponekad SQLMap može generirati lažno pozitivne rezultate, tj. prikazati ranjivosti koje zapravo ne postoje. To može dovesti do gubitka vremena i resursa prilikom analize i ispravljanja problema koji zapravo nisu prisutni u aplikaciji. Stoga je važno provesti dodatne provjere kako bi se potvrdila autentičnost otkrivenih ranjivosti.

5.2.5 **Mogućnosti SQLMapa**

Alati poput SQLMap-a su relativno automatizirani i nude puno više mogućnosti napadaču i iskorištavaju doslovno svaku slabu točku sustava. Neke od mogućnosti
izvoru [5] su:

- **Napredno mapiranje baze podataka:** SQLMap nudi napredno mapiranje strukture baze podataka, omogućujući korisnicima detaljnu analizu i izvlačenje podataka iz

baza podataka različitih tipova. Ova funkcionalnost pomaže sigurnosnim stručnjacima da razumiju arhitekturu baze podataka i identificiraju potencijalne ranjivosti.

- **Fleksibilnost u izvozu rezultata:** Alat pruža fleksibilne opcije za izvoz rezultata testiranja, uključujući formatiranje izvještaja prema potrebama korisnika. Korisnici mogu izvesti rezultate u različite formate poput CSV, XML ili HTML, olakšavajući dijeljenje i analizu rezultata testiranja.
- **Automatsko prepoznavanje i korištenje dodatnih parametara:** SQLMap automatski prepoznaje dodatne parametre i postavke aplikacije te ih koristi u procesu testiranja. Ova funkcionalnost pomaže korisnicima da osiguraju sveobuhvatan i detaljan test sigurnosti aplikacije.
- **Integracija s drugim alatima:** Alat se lako integrira s drugim alatima za sigurnosno testiranje i analizu, pružajući korisnicima mogućnost kombiniranja različitih alata i tehnika za bolje rezultate testiranja.
- **Napredne opcije za obmanu filtriranja:** SQLMap nudi napredne opcije za obmanu filtriranja i detekciju sigurnosnih mehanizama poput WAF-a (*Web Application Firewall*). Ova funkcionalnost omogućuje korisnicima da testiraju aplikacije koje koriste različite sigurnosne slojeve i mehanizme zaštite.

5.3 XSStrike

XSStrike je vitalan za prepoznavanje ranjivosti i povećanje sigurnosti web aplikacija otkrivanjem i iskorištavanjem ranjivosti temeljenih na XSS-u, koje napadačima omogućuju ubacivanje zlonamjernih skripti u preglednik krajnjeg korisnika.

5.3.1 **Identifikacija i iskorištavanje XSS ranjivosti**

XSStrike ne samo da identificira ranjivosti *Cross-Site Scripting* (XSS), već omogućuje i njihovo iskorištavanje kako bi korisnici dublje istražili potencijalne prijetnje. Kroz aktivno skeniranje, alat detaljno pretražuje web aplikacije u potrazi za XSS ranjivim točkama, omogućujući korisnicima da automatski otkriju i iskoriste te ranjivosti. XSStrike pruža korisnicima alate i tehnike potrebne za ispitivanje i manipulaciju pronađenim XSS ranjivostima, što omogućuje bolje razumijevanje njihove ozbiljnosti i potencijalnih posljedica.

5.3.2 **Prikupljanje podataka i demonstracija prijetnji**

Uz identifikaciju ranjivosti, XSStrike korisnicima omogućuje prikupljanje osjetljivih podataka i informacija s ciljanih web stranica. To uključuje ne samo sesije korisnika, već i kolačiće i druge povjerljive informacije koje bi mogli iskoristiti napadači. Nadalje, alat omogućuje korisnicima da demonstriraju prijetnje i potencijalne ranjivosti web aplikacija putem izmjene ponašanja. Ova funkcionalnost omogućuje simuliranje napada i prikaz posljedica XSS ranjivosti na web aplikacije, pružajući korisnicima dublji uvid u sigurnosne rizike koje takve ranjivosti mogu predstavljati.

5.3.3 **Prednosti XSStrikea**

Osim što nudi sveobuhvatan set metoda skeniranja XSS-a i korisnih *payloada*, čineći ga vrijednim alatom za sigurnosne stručnjake i etičke hakere u zaštiti web aplikacija. Izvor [24] sadrži prednosti ovog alata, a neke od prednosti su:

- **Automatizacija otkrivanja XSS ranjivosti:** XSStrike omogućuje automatizirano otkrivanje *Cross-Site Scripting* (XSS) ranjivosti, što značajno ubrzava proces sigurnosnog testiranja.
- **Dubinska analiza ranjivosti:** Alat pruža detaljnu analizu otkrivenih XSS ranjivosti, omogućujući korisnicima da dublje istraže njihovu ozbiljnost i potencijalne posljedice.
- **Prilagodljive opcije testiranja:** Korisnici mogu prilagoditi postavke skeniranja prema svojim potrebama i specifičnostima web aplikacija koje testiraju.
- **Napredno praćenje aktivnosti:** XSStrike omogućuje praćenje aktivnosti tijekom testiranja, uključujući registraciju svih HTTP zahtjeva i odgovora, što omogućuje detaljno analiziranje interakcije između aplikacije i korisnika.
- **Integracija s drugim alatima:** Korisnici mogu jednostavno kombinirati različite alate i tehnike za poboljšane rezultate testiranja jer se alat neprimjetno integrira s drugim alatima za sigurnosno testiranje i razvojnim okruženjima.

5.3.4 **Nedostaci XSStrikea**

Alat ima vrlo važne prednosti no i neke nedostatke koji utječu na korištenje ukoliko se ne radi samo o XSS ranjivosti. Prema izvoru [24] ovo su neki od nedostataka:

- **Ograničena pokrivenost:** XSStrike se uglavnom usredotočuje na ranjivosti *Cross-Site Scriptinga*, dok možda ne pokriva sve druge vrste ranjivosti koje mogu biti prisutne u web aplikacijama.
- **Složenost uporabe:** Alat može zahtijevati određeno vrijeme i učenje kako bi se u potpunosti iskoristio, posebno za korisnike koji nisu upoznati s tehničkim pojmovima ili metodama testiranja sigurnosti.
- **Mogućnost lažno pozitivnih rezultata**: XSStrike može generirati lažno pozitivne rezultate ili previdjeti neke stvarne ranjivosti, što može dovesti do gubitka vremena i resursa prilikom analize sigurnosti web aplikacija.
- **Ovisnost o ažuriranju:** Kao i svaki alat za sigurnosno testiranje, XSStrike zahtijeva redovito ažuriranje kako bi ostao učinkovit u otkrivanju najnovijih sigurnosnih prijetnji i tehnika napada.
- **Potreba za naprednijim znanjem:** Za potpuno iskorištavanje XSStrikea može biti potrebno naprednije znanje o web sigurnosti i sigurnosnom testiranju, što može otežati upotrebu za početnike ili manje iskusne korisnike.

5.3.5 **Mogućnosti XSStrikea**

XSStrike dolazi u paketu sa raznim alatima i mogućnostima, od *database fingerprinta*, do pristupa podatkovnom sistemu i izvršavanju naredbi na udaljenim operacijskim sustavima. Prvi primjer SQL *injectiona* bio je vrlo pojednostavljen zbog lakšeg shvaćanja principa i samog cilja takvog napada. Osim automatizacije koja je vrlo bitna stvar u XSStrikeu, postoje i druge mogućnosti. Prema izvoru [24] ovo su mogućnosti:

- **XSStrike automatizacija:** prepoznavanje i iskorištavanje ranjivosti *Cross-Site Scripting* u web aplikacijama dopuštajući korisnicima da učinkovito skeniraju web stranice u potrazi za XSS ranjivostima. Time se pojednostavljuje proces prepoznavanja sigurnosnih rizika.
- **Napredne tehnike skeniranja:** Alat nudi različite napredne tehnike skeniranja koje omogućuju temeljitu analizu web stranica radi otkrivanja XSS ranjivosti. Osim pasivnog skeniranja prometa, XSStrike pruža i aktivno skeniranje koje koristi različite tehnike kako bi otkrio skrivenije ranjivosti.
- **Detaljno iskorištavanje ranjivosti:** XSStrike omogućuje korisnicima da detaljno iskoriste otkrivene XSS ranjivosti radi demonstracije potencijalnih napada i

posljedica. To uključuje manipulaciju zahtjevima i odgovorima kako bi se prikazale različite mogućnosti napada i otkrile potencijalne sigurnosne prijetnje.

- **Prilagodljivost i skriptiranje:** Alat podržava prilagodljive skripte koji omogućuju korisnicima da prilagode postavke skeniranja i iskoriste dodatne funkcionalnosti. To uključuje stvaranje vlastitih skripti za testiranje sigurnosti ili korištenje postojećih skripti za izvođenje različitih testova.
- **Integracija i izvješća:** XSStrike se može integrirati s raznim sigurnosnim alatima i razvojnim okruženjima kako bi se pojednostavilo testiranje sigurnosti web aplikacija. Generira detaljna izvješća o otkrivenim ranjivostima i aktivnostima, nudeći korisnicima sveobuhvatan pregled rezultata testiranja i preporučenih sigurnosnih mjera.

5.4 Burp Suite

Burp Suite, razvijen od strane renomirane tvrtke PortSwigger, ističe se kao sveobuhvatan alat za testiranje sigurnosti web aplikacija koji je postao ključan dio arsenala sigurnosnih stručnjaka diljem svijeta. Njegova reputacija izgrađena je na temelju svestranosti, pouzdanosti i naprednih značajki koje pruža.

Jedna od ključnih karakteristika Burp Suite-a je njegova sposobnost da korisnicima omogući sveobuhvatan uvid u sigurnosne propuste njihovih aplikacija. Kroz niz alata za skeniranje, analizu i manipulaciju HTTP prometa, korisnici mogu otkriti širok spektar ranjivosti, uključujući SQL injekcije, XSS napade, CSRF prijetnje i mnoge druge. [21]

5.4.1 **Identifikacija ranjivosti web aplikacija**

Identifikacija ranjivosti web aplikacija ključna je faza u procesu sigurnosnog testiranja, a Burp Suite pruža moćne alate i tehnike za ovu svrhu. Kroz svoje napredne mogućnosti skeniranja i analize, Burp Suite omogućuje korisnicima da detaljno istraže web aplikacije u potrazi za različitim ranjivostima. Burp Suite koristi kombinaciju pasivnih i aktivnih tehnika skeniranja kako bi identificirao potencijalne sigurnosne prijetnje. Pasivno skeniranje omogućuje alatu da prati i analizira promet između preglednika krajnjeg korisnika i web servera, identificirajući potencijalne ranjivosti poput neispravne upotrebe sesija, izlaganja osjetljivih podataka ili upotrebe nezaštićenih HTTP veza. S druge strane, aktivno skeniranje uključuje aktivno slanje zahtjeva web aplikacijama s ciljem identificiranja ranjivosti poput SQL injekcija, XSS napada, CSRF prijetnji i drugih. Burp Suite omogućuje korisnicima da

prilagode postavke skeniranja, kao što su dubina i opseg skeniranja, kako bi se bolje usredotočili na određene dijelove web aplikacija ili specifične vrste ranjivosti. Nakon što se identificiraju potencijalne ranjivosti, Burp Suite pruža korisnicima detaljne izvještaje i preporuke za ispravak pronađenih propusta. Ovi izvještaji sadrže informacije o otkrivenim ranjivostima, njihovoj ozbiljnosti i preporukama za njihovo rješavanje, pružajući tako jasan put za poboljšanje sigurnosti web aplikacija.

5.4.2 **Automatizacija skeniranja ranjivosti**

Koristeći Burp Suite's Scanner, korisnici mogu automatizirati skeniranje web aplikacija radi identifikacije širokog spektra sigurnosnih propusta. Burp Suite Scanner omogućuje korisnicima da definiraju skup parametara i kriterija za skeniranje, uključujući opseg skeniranja, dubinu skeniranja, tehnike skeniranja i ciljeve skeniranja. Ova prilagodljivost omogućuje korisnicima da usmjere svoje skeniranje prema određenim dijelovima web aplikacija ili specifičnim ranjivostima, što rezultira u učinkovitijem procesu identifikacije sigurnosnih prijetnji. Jedna od ključnih značajki Burp Suite Scannera je mogućnost pasivnog skeniranja, koje omogućuje alatu da prati i analizira promet između preglednika krajnjeg korisnika i web servera kako bi identificirao potencijalne ranjivosti. Osim toga, Burp Suite Scanner provodi i aktivno skeniranje, koje uključuje aktivno slanje zahtjeva web aplikacijama radi otkrivanja ranjivosti poput SQL injekcija, XSS napada, CSRF prijetnji i drugih. Burp Suite Scanner također pruža napredne mogućnosti za praćenje napretka skeniranja, kao što su detaljni izvještaji, statistike skeniranja i upravljanje ranjivostima. Ovi alati omogućuju korisnicima da učinkovito upravljaju skeniranjem, analiziraju rezultate skeniranja i prioritiziraju ispravke ranjivosti kako bi osigurali visoku razinu sigurnosti svojih web aplikacija.

5.4.3 **Detaljna analiza ranjivosti**

Jedna od ključnih značajki Burp Suite je mogućnost ručnog testiranja, koja korisnicima omogućuje interaktivno istraživanje otkrivenih ranjivosti i provjeru njihove stvarne ranjivosti. Kroz Burp Suite's Proxy alat, korisnici mogu pregledavati, modificirati i ponovno slati HTTP zahtjeve i odgovore kako bi testirali različite scenarije napada i provjerili osjetljivost web aplikacija na različite vrste napada. Osim ručnog testiranja, Burp Suite pruža i napredne alate za analizu ranjivosti, kao što su skener ranjivosti, pasivno skeniranje i aktivno skeniranje.

Kroz ove alate, korisnici mogu provesti detaljnu analizu otkrivenih ranjivosti i identificirati potencijalne *exploite* i ranjivosti koje bi mogle biti iskorištene napadačima. Burp Suite također omogućuje korisnicima da generiraju detaljne izvještaje o otkrivenim ranjivostima, uključujući opis ranjivosti, preporučene korake za ispravak, razinu ozbiljnosti i potencijalne posljedice.

5.4.4 **Prednosti Burp Suite-a**

Jedna od ključnih prednosti Burp Suite-a je njegova svestranost. Nudi različite značajke za analizu i ispitivanje web aplikacija, bez obzira na vrstu ranjivosti ili složenost aplikacije. Korisnici mogu pouzdano osloniti se na Burp Suite za detaljnu analizu ranjivosti, koristeći napredne tehnike skeniranja koje omogućuju identifikaciju čak i najmanjih sigurnosnih prijetnji. Osim toga, Burp Suite se ističe svojom efikasnošću i pouzdanošću. Dugogodišnje iskustvo u razvoju ovog alata rezultiralo je pouzdanim i efikasnim rješenjem za testiranje sigurnosti web aplikacija. Korisnici mogu biti sigurni da će Burp Suite precizno identificirati ranjivosti i pružiti detaljna izvješća o njima.

5.4.5 **Nedostaci Burp Suite-a**

Jedan od nedostataka je visoka cijena licence, što može biti prepreka za pojedince ili manje organizacije s ograničenim budžetom. Uz to, korisničko sučelje alata može biti složeno za nove korisnike, zahtijevajući vrijeme i trud kako bi se osposobili za puni potencijal. Burp Suite također može zahtijevati značajne resurse sustava, posebno kod skeniranja kompleksnih web aplikacija, što može rezultirati sporijim performansama ili potrebom za dodatnim resursima. Iako su ovi nedostaci važni, Burp Suite i dalje ostaje moćan alat za identifikaciju i rješavanje sigurnosnih ranjivosti web aplikacija, pružajući vrijedne značajke koje su ključne za sigurnost online okruženja.

6. RANJIVA APLIKACIJA ZA TESTIRANJE ALATA

U ovom radu stvorena je osnovna aplikacija koja korisnicima omogućuje iskorištavanje sigurnosnih propusta web aplikacije ili simulaciju napada. Aplikacija je dizajnirana posebno za testiranje alata za skeniranje ranjivosti i podijeljena je u tri glavna odjeljka: prvi uvodi korisnike u SQL *Injection*, dok drugi pokriva *Cross-site request forgery* (CSRF) napade, a treći se bavi *Cross-Site Scripting* (XSS). Interakcijom s tim ranjivostima i napadima cilj je podići svijest o važnosti sigurnosti web aplikacija i motivirati korisnike da steknu potrebne vještine za očuvanje integriteta podataka i sigurnosti na internetu.

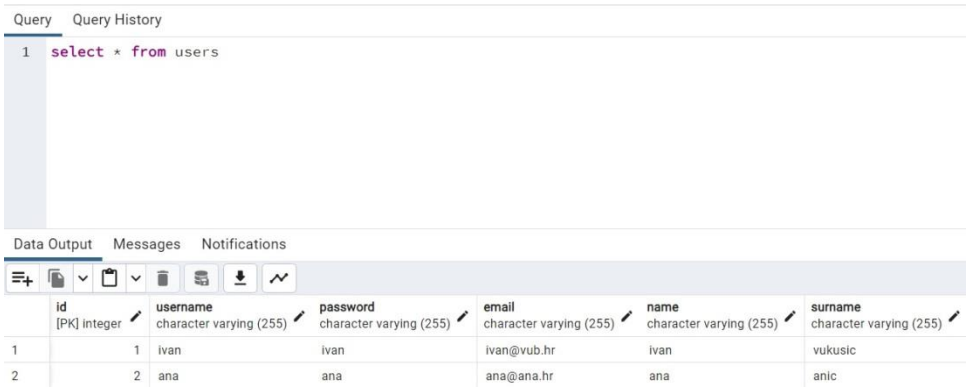
6.1 Primjer SQL *injectiona* u aplikaciji i testiranje alata za skeniranje

Programski kod 6.1 prikazuje funkciju aplikacije na strani poslužitelja, posebno u vezi sa sustavom prijave korisnika. Ova funkcija komunicira s PostgreSQL bazom podataka koja pohranjuje korisničke informacije. Slika 6.1 ilustrira odnos i uključene podatke. Zbog izravne upotrebe parametara na strani klijenta u URL upitu koji se šalje bazi podataka, aplikacija je osjetljiva na SQL *injection*.

Programski kod 6.1: Prikaz funkcije na serverskoj strani aplikacije

Izvor: Autor

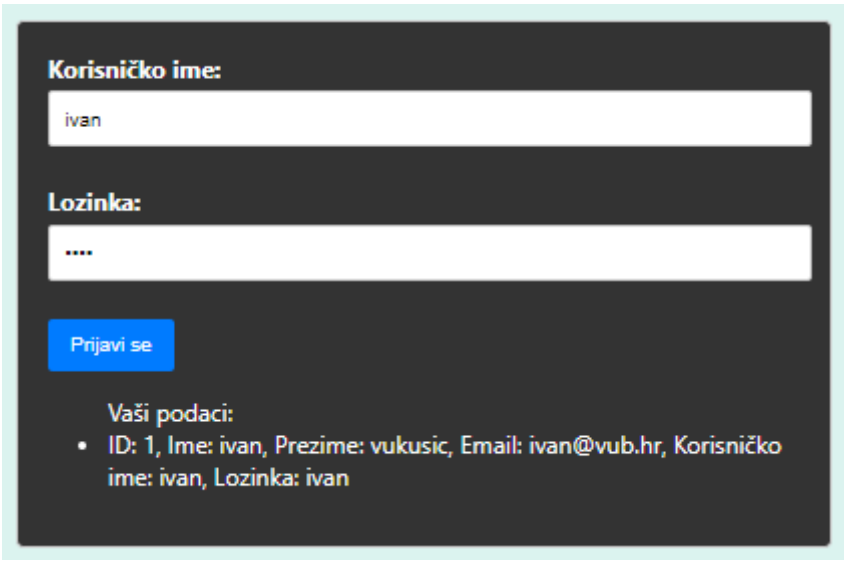
```
exports.login = (req, res) => {
    const sqlQuery = `SELECT * FROM users WHERE username =
'${req.query.username}' and password = '${req.query.password}'`;

    db.query(sqlQuery, (error, results) => {
        console.log(results);
        if (error) {
            return res.status(500).json({ error: 'Greška u bazi podataka'
}));
        }
        return res.json(results);
    });
};
```



Slika 6.1: Prikaz podataka u bazi podataka

Izvor: Autor

Na strani klijenta aplikacije nalazi se obrazac koji šalje podatke funkciji na strani poslužitelja. Registrirani korisnik nakon uspješne prijave može se prijaviti i pregledati vlastite podatke iz baze.



Slika 6.2: Prikaz sučelja za registraciju

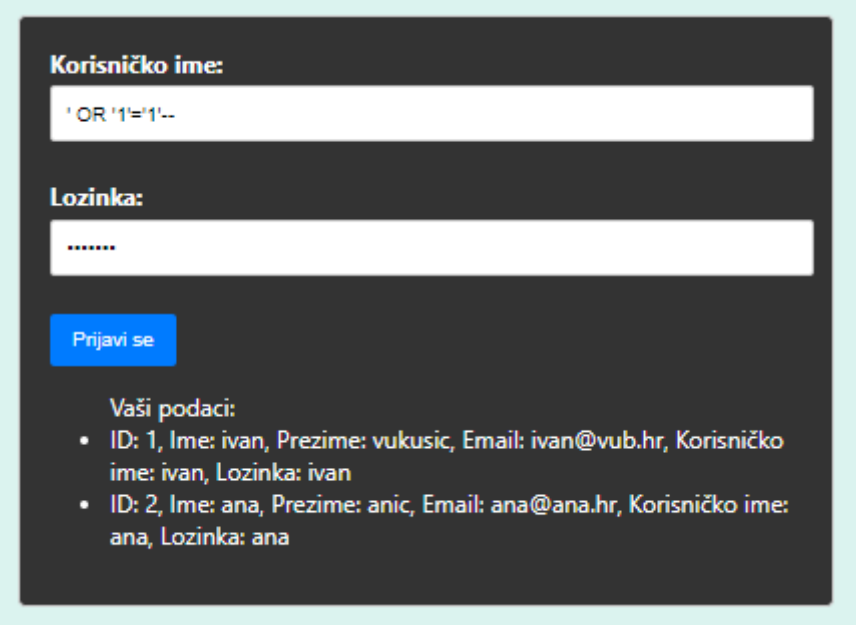
Izvor: Autor

Zbog ranjivosti prisutne u funkciji prijave, korisnik može pokušati unijeti, na primjer, umjesto korisničkog imena:

' OR '1'='1'—

Korisnici će unijeti proizvoljan niz znakova za lozinku. Svaki tekst nakon "--" smatrat će se komentarom i zanemariti. Korisnik je ovim unosom postigao da se izvrši:

SELECT * FROM users WHERE username = '' OR '1'='1'-- AND password = 'proizvoljan tekst koji je zanemaren', upit će biti potvrđen kao istinit u bazi podataka i dohvatit će sve podatke iz tablice na koju se odnosi. Ishod ovog upita prikazan je na slici ispod.



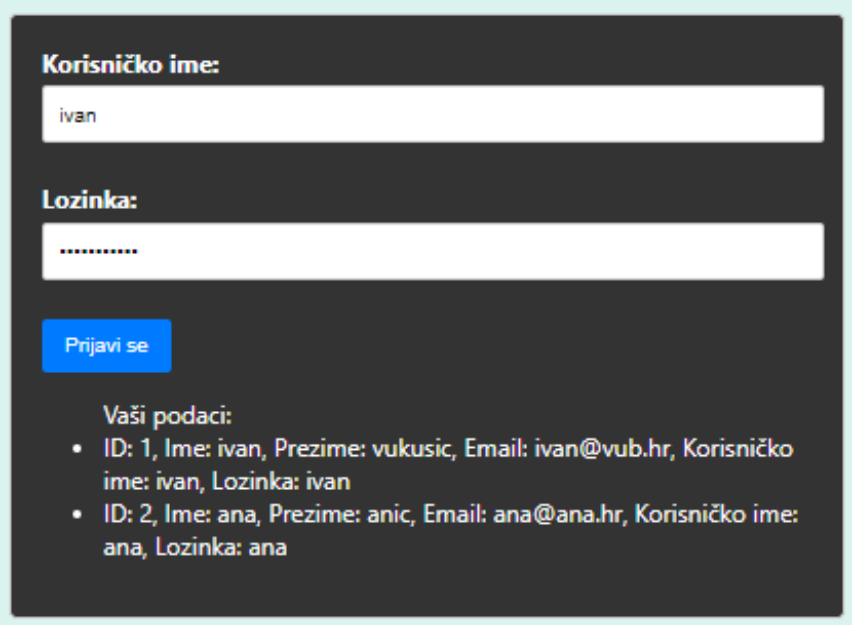
Slika 6.3: Prikaz rezultata

Izvor: Autor

U ovom primjeru, slika 6.4 prikazuje primjer iskorištavanja ranjivosti gdje je dopušten proizvoljan tekst za korisničko ime, a umjesto lozinke koristi se drugačiji unos:

' OR '1'='1

Ovo će uvijek biti istinito i dohvaćati sve zapise iz navedene tablice u bazi podataka.



Slika 6.4: Prikaz rezultata

Izvor: Autor

U prethodnim slučajevima ovaj obrazac i njegova funkcija pokazali su loše ponašanje što se može iskoristiti za napade. Kada netko koristi alat za skeniranje ranjivosti kao što je SQLMap ili OWASP ZAP, može otkriti svoje slabe točke.

6.1.1 **Korištenje alata SQLMap**

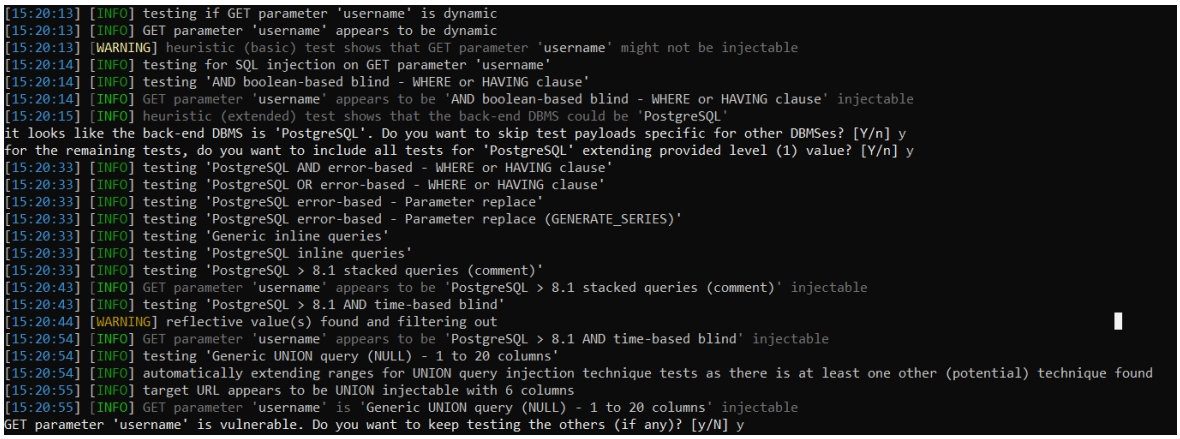
U poglavlju 5.2 predstavljen je SQLMap, čiji je rad prikazan u nastavku.

Nakon instalacije alata, pokrenuta je naredba kojom započinje skeniranje zadanog URL-a:

```
sqlmap.py -u "http://localhost:8080/sqlInjection/login?username=ivan&password=ivan" --risk=3 --tamper=space2comment
```

Moguće je postaviti razne parametre, a korišteni su:

- --risk=3: Određuje razinu rizika za testiranje SQL ubacivanja, u rasponu od 1 do 3, pri čemu 3 označava najvišu razinu rizika.
- --tamper=space2comment: U upitima baze podataka razmaci se zamjenjuju komentarima kako bi se spriječilo otkrivanje tijekom testiranja.
- -u : Naveden je URL koji se testira, s poslužiteljskim dijelom aplikacije na portu 8080.

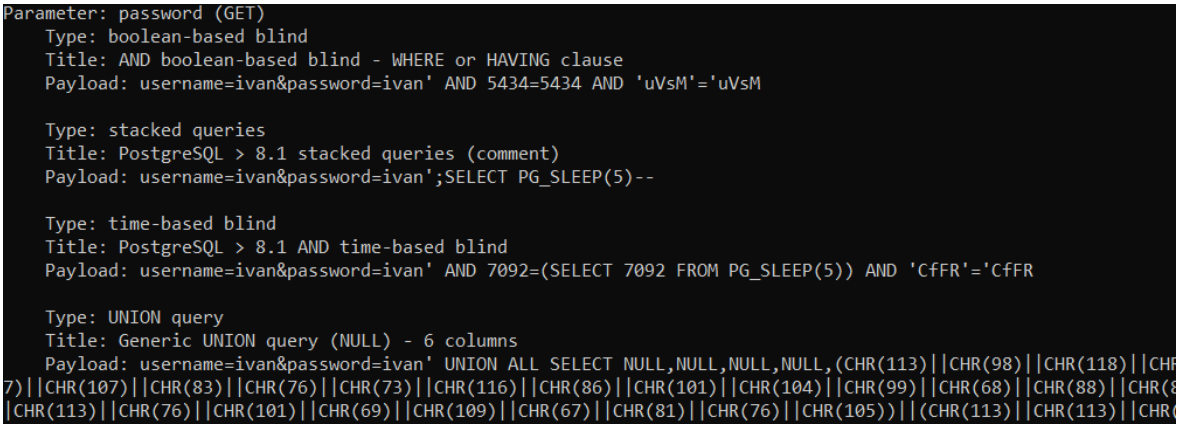


Slika 6.5: Prikaz ispisa u terminalu tijekom testiranja

Izvor: Autor

Na slici 6.5 prikazan je izlaz terminala tijekom testiranja. SQLMap prepoznaje GET parametar 'korisničko ime' kao dinamički, sugerirajući moguću ranjivost SQL injekcije. Iako je identificiran kao dinamičan, SQLMap još uvijek provodi heuristički test koji sugerira potencijalnu odsutnost ranjivosti, dajući početni uvid u situaciju. Nakon toga, SQLMap opsežno testira različite metode SQL ubacivanja koje su specifične za parametar 'username', kao što su slijepi upiti temeljeni na booleovim vrijednostima, naslagani upiti, slijepi upiti temeljeni na vremenu i UNION upiti. S pozadinskim DBMS-om prepoznatim kao PostgreSQL, SQLMap prilagođava testiranje tehnikama relevantnim za ovaj određeni sustav. Kroz više testova, od kojih su neki uspješni, potvrđena je ranjivost parametra 'username' na SQL injekciju. Zatim se provodi sličan postupak testiranja na parametru 'lozinka', što dovodi do istog zaključka:

sqlmap identified the following injection point(s) with a total of 112 HTTP(s) requests



Slika 6.7: Testiranje parametra password

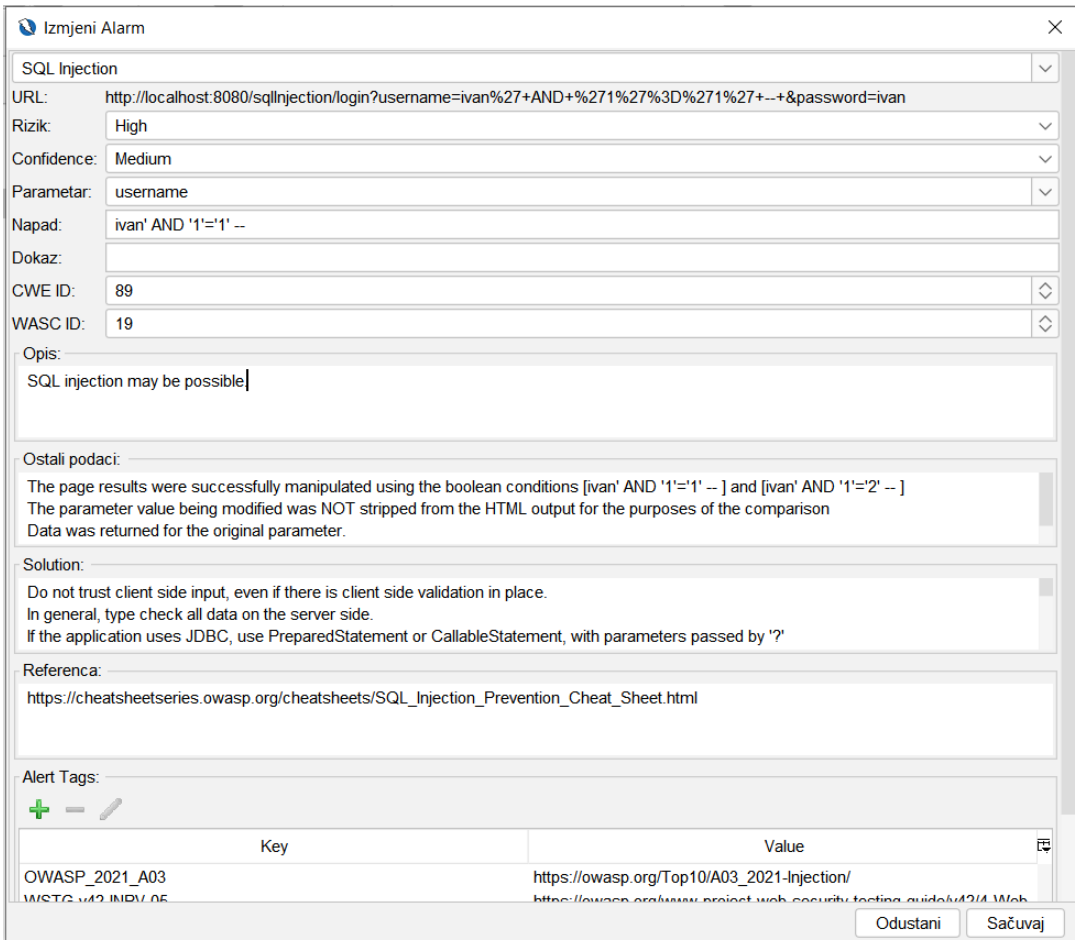
Izvor: Autor

Oba parametra su ranjiva, s prikupljenim primjerima korisnih opterećenja za iskorištavanje ovih ranjivosti. Postoje brojni drugi parametri naredbi osim ovih za testiranje raznih vrsta napada:

- --union-cols=2: Označava količinu stupaca za provođenje testiranja SQL ubacivanja temeljenog na UNION-u
- --technique=B: Opisuje metodu testiranja SQL ubrizgavanja, gdje "B" označava ubacivanje temeljeno na Booleovim vrijednostima
- --technique=E: Tehnika temeljena na greškama koristi se za identificiranje grešaka u SQL upitu
- --technique=T: Slijepo ubrizgavanje temeljeno na vremenu koristi se kada je vrijeme odgovora poslužitelja značajno odgođeno kada je ispunjen određeni uvjet
- --dump: Podaci se moraju izvesti iz baze podataka kada se identificira ranjivost
- --batch: SQLMap je dizajniran za automatiziranje određivanja svih parametara, te će pokušati različite vrste injekcija bez razmatranja rizika i sigurnosti.

6.1.2 **Korištenje alata OWASP ZAP**

Poglavlje 6.1.2 objašnjava rad OWASP ZAP-a. Prije pokretanja skeniranja pomoću alata, bitno je konfigurirati *proxy* poslužitelj za presretanje prometa. *Proxy* poslužitelj preusmjerava zahtjeve s ciljanog URL-a na sebe, a zatim na odredište za snimanje. *Proxy* je postavljen na port 8081 zbog pozadine aplikacije na portu 8080. Korisnici mogu pristupiti ovom alatu putem korisničkog sučelja i terminala. Nakon što je *proxy* postavljen, skeniranje može započeti, prikazujući rezultate poput onih prikazanih na slici broj sedam. Izvješće o skeniranju sadrži tehničke podatke o provedenom napadu, detaljan opis podataka o napadu (npr. parametri) i identificirane ranjivosti. Izvješće, poznato kao upozorenje, daje prijedloge za rješavanje ranjivosti i uključuje poveznice na detaljna objašnjenja i upute.



Slika 6.8: Rezultati testiranja alatom OWASP ZAP

Izvor: Autor

6.2 Primjer XSS ranjivosti u aplikaciji i otkrivanje pomoću alata

Konstruirana forma u razvijenoj aplikaciji, kao što je prikazano na slici broj 6.9, podložna je XSS (*Cross-site scripting*) napadima. Korisnici unose svoje ime i komentar u obrazac koji se zatim prikazuje na stranici. Komentari pohranjeni u bazi podataka od prethodnih korisnika dohvaćaju se kada se web stranica učita, što potencijalno dovodi do pohranjenog XSS napada.

Slika 6.9: Prikaz forme koja je ranjiva na XSS

Izvor: Autor

PHP kod u ovom obliku generira HTML div element za svaki komentar pohranjen u bazi podataka, prikazujući komentar zajedno s imenom korisnika. Ovaj proces može biti sigurnosni rizik jer se podaci izravno prikazuju korisniku bez prethodne obrade.

Programski kod 6.2: Prikaz funkcije za dohvaćanje i prikazivanje komentara

Izvor: Autor

```
try {
    $conn = new
PDO("pgsql:host=$host;dbname=$database;user=$user;password=$password");

    $conn-> setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    $query = "SELECT * FROM comments";
    $stmt = $conn-> query($query);

    if($stmt){
        while ($comment = $stmt -> fetch (PDO::FETCH_ASSOC)){
            echo "<div class='comment'><p>Name:
{$comment['name']}</p><p>Comment: {$comment['comment']}</p></div>";
        }
    }  else{
```

```
        echo json_encode(['error'=>'Greška prilikom dohvata komentara:
' .$conn->errorInfo()[2]]);
    }
}  catch(PDOException $e) {
        echo json_encode(['error' => 'Greška prilikom dohvata
komentara: ' . $e->getMessage()]);
    }
```

Podaci koje korisnik unese šalju se izravno u bazu nakon podnošenja novog komentara. Ovo je namjerno učinjeno kako bi se korisnicima aplikacije pokazale potencijalne posljedice i kako bi se procijenila učinkovitost alata u identificiranju XSS ranjivosti u web aplikacijama.

Programski kod 6.3: Funkcija za spremanje komentara

Izvor: Autor

```
try{
    $conn = new
PDO("pgsql:host=$host;dbname=$database;user=$user;password=$password");
    $conn-> setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $conn->exec('SET NAMES \'UTF8\'');

    //upit za unos novog komentara
    $query = "INSERT INTO comments (name, comment) VALUES (:name,
:comment)";
    $stmt = $conn ->prepare($query);
    $stmt ->bindParam(':name, $name);
    $stmt ->bindParam(':comment', $comment);
    $stmt ->execute();
} catch(PDOException $e){
    echo json_encode(['error' => 'Greška prilikom unosa komentara: '
.$e ->getMessage()]);
}
```

Prikaz komentara pohranjenih u bazi podataka vidi se na slici broj 6.10

Query Query History

```
1  select * from comments
```

Data Output Messages Notifications

	id [PK] integer	name character varying (20)	comment character varying (200)
1	1	ivan	ovo je komentar za probu
2	2	test	test
3	3	admin	dobrodosli na blog
4	4	ana	Meni se svida
5	5	test2	test2
6	6	test3	test3
7	7	napadac	<script>alert("ovo je napad")</script>
8	8	haker	tekst

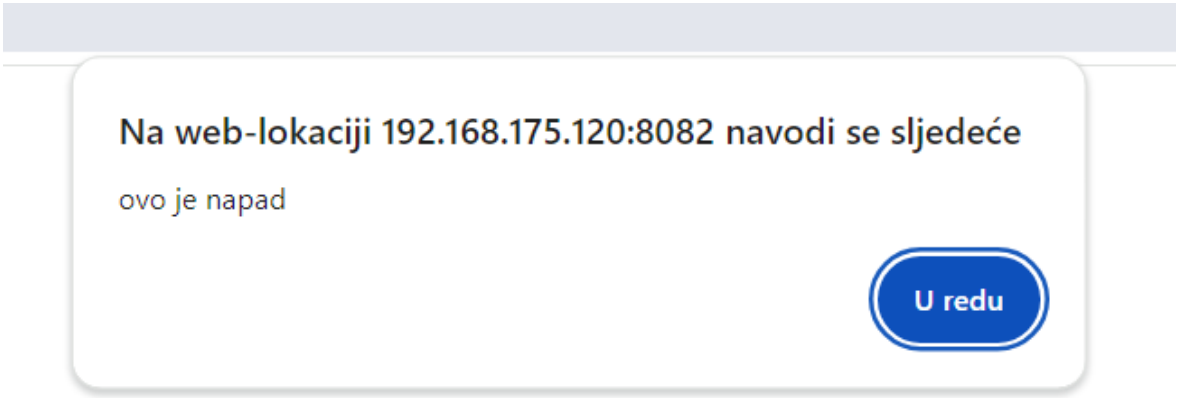
Slika 6.10: Pohranjeni komentari u bazi podataka

Izvor: Autor

Nakon što se stranica učita, ako se komentar koji ima štetnu skriptu pohrani u bazu podataka, pokrenut će se u vašem web pregledniku. Recimo, na primjer, komentar ima:

<script>alert('ovo je napad')</script>

pojavit će se skočni prozor, kao što je prikazano na slici 6.11.



Slika 6.11: Prikaz skočnog prozora nakon učitavanja pohranjenog komentara sa zlonamjernom skriptom

Izvor: Autor

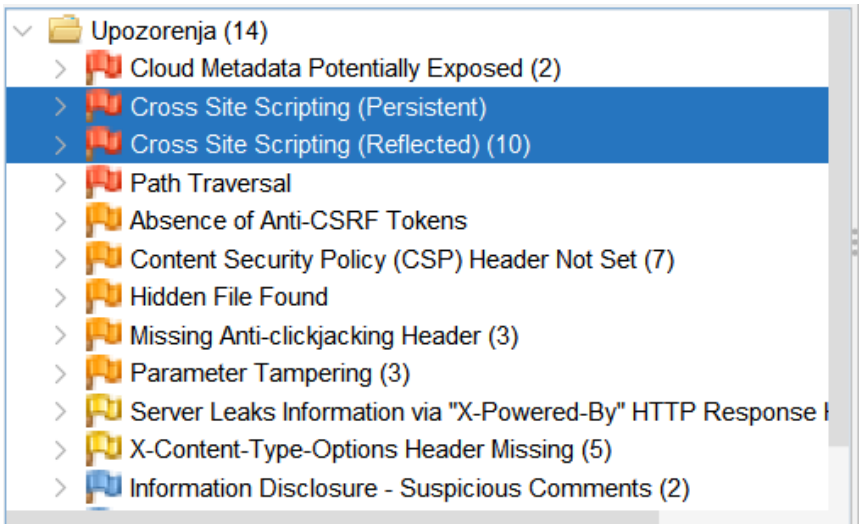
6.2.1 **Korištenje alata OWASP ZAP**

Nakon automatiziranog skeniranja web aplikacije stvorene za konačni projekt, OWASP ZAP otkrio je potencijalno ranjivi POST zahtjev i njegove parametre na stranicama aplikacije. Detaljan pregled alata uključuje parametre koji su korišteni za slanje zahtjeva, a koji su naknadno korišteni za aktivno skeniranje komunikacijskog načina obrasca za slanje POST zahtjeva.

Slika 6.12: Ranjivi post zahtjev i njegovi parametri

Izvor: Autor

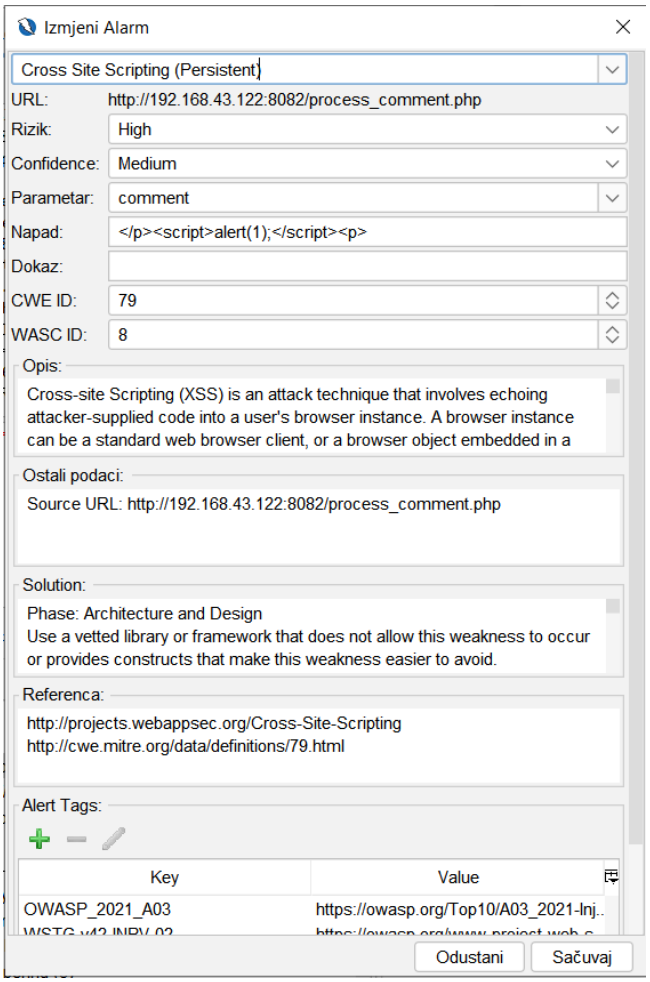
OWASP ZAP pronašao je 14 upozorenja tijekom skeniranja za ranjivosti, gdje su 2 upozorenja bila povezana s *Cross Site Scripting*. Alat je prepoznao pohranjeni i reflektirani XSS kao različite vrste ove ranjivosti.



Slika 6.13: Upozorenja nakon skeniranja aplikacije

Izvor: Autor

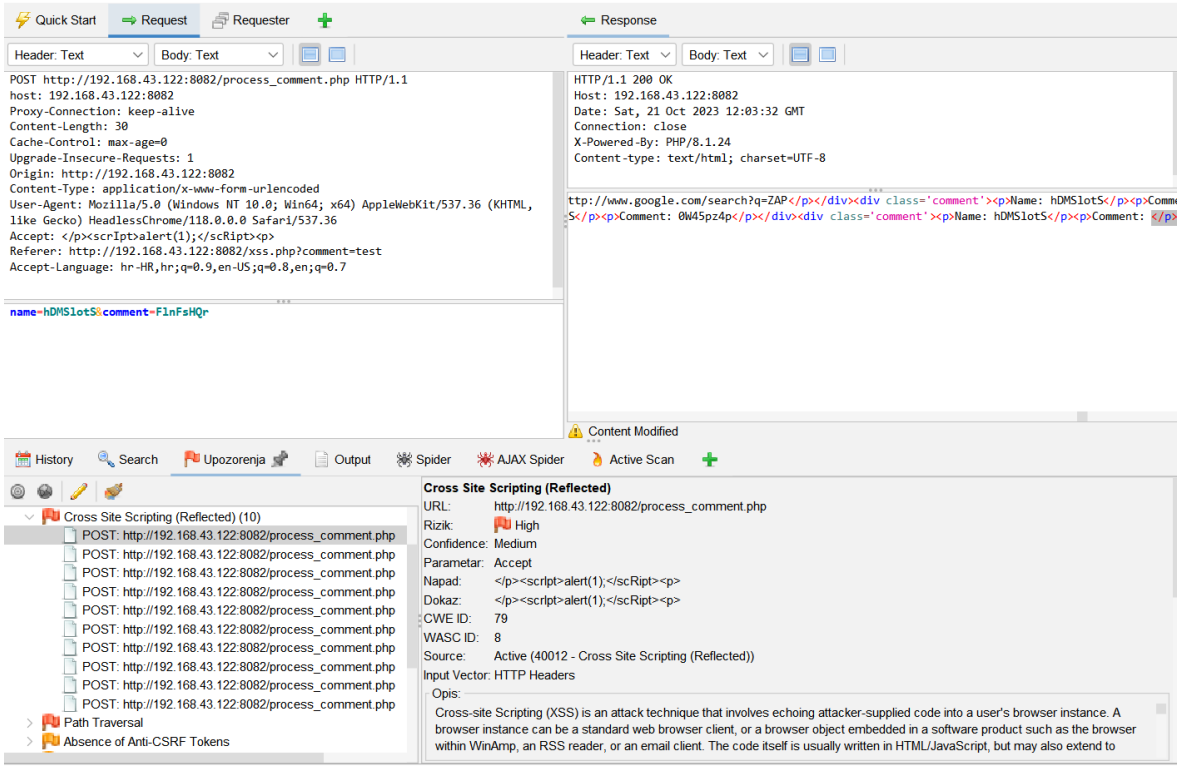
Slika 6.14 pruža opsežniji pogled na pohranjeno XSS upozorenje. Baš kao i kod SQL *injection* u poglavlju 6.1, upozorenje uključuje tehničke detalje poput URL-a, iscrpno objašnjenje ranjivosti i preporuke za prevenciju.



Slika 6.14: Detaljni prikaz upozorenja o pohranjenom XSS-u

Izvor: Autor

10 POST zahtjeva u reflektiranim XSS upozorenjima razlikuje se prema parametru zaglavlja koji se koristi za prijenos zlonamjernog komentara ili skripte. Na slici 6.15 zlonamjerni sadržaj se prenosi kroz parametar *Accept*, dok ostali parametri uključuju *Accept-Language*, *Content-Type*, *Host*, *Origin*, *Referer* i *User-Agent*.



Slika 6.15: Prikaz upozorenja o reflektiranom XSS-u

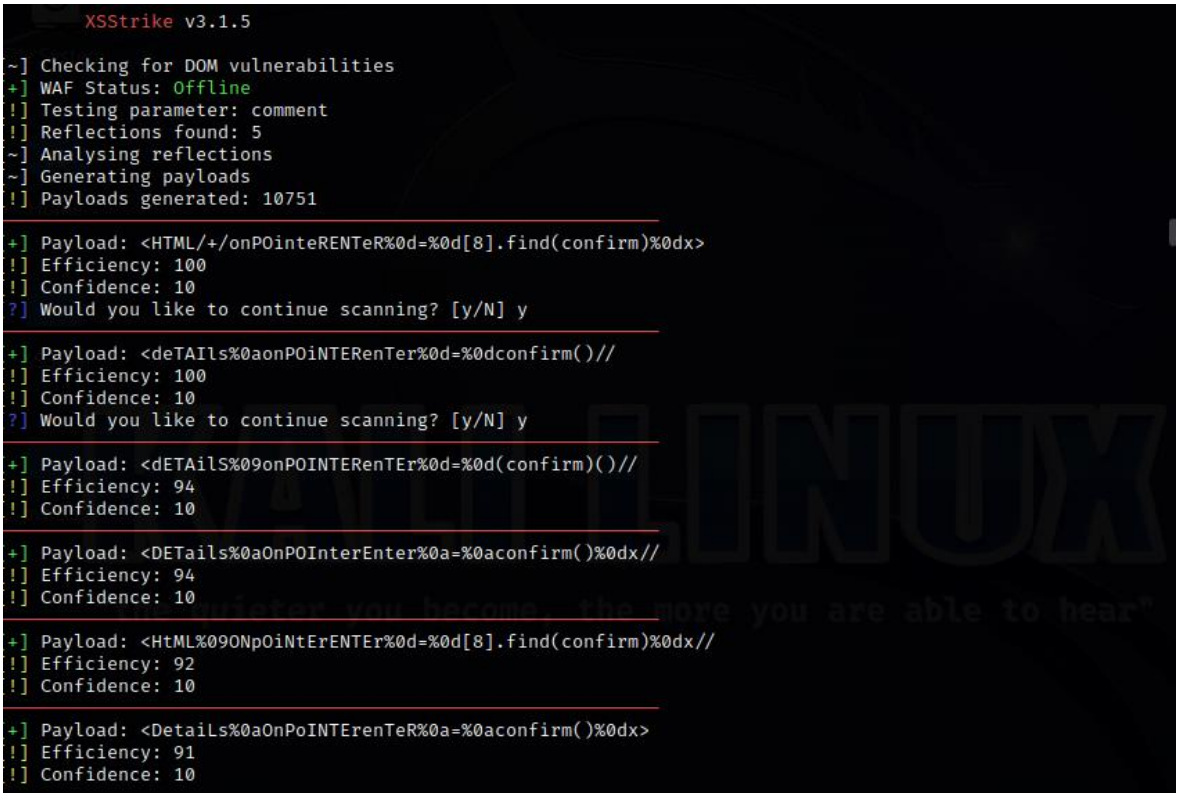
Izvor: Autor

6.2.2 **Korištenje alata XSStrike**

Nakon instalacije i postavljanja alata XSStrike, naredbom

```
python xsstrike.py –u "http://192.168.43.120:8082/process_comments.php" --data "comment=<script>alert('xss napad")</script>
```

Funkcija skeniranja za rukovanje korisničkim komentarima pokreće se putem datoteke process_comments.php na lokalnom poslužitelju na portu 8082. Alat pokazuje da je Vatrozid web aplikacije (WAF) trenutno izvan mreže, što može dovesti do potencijalnih ranjivosti parametra komentara. XSStrike je proveo 10751 automatizirani test kako bi identificirao moguće sigurnosne slabosti, a svaki test koristi jedinstveni zlonamjerni JavaScript korisni teret. Ovi opsežni testovi imaju za cilj otkriti ranjivosti koje se mogu pojaviti u različitim scenarijima ili kombinacijama koda. Svaki test se ocjenjuje za njegovu učinkovitost i pouzdanost.



Slika 6.16: Testiranje XSStrike alatom

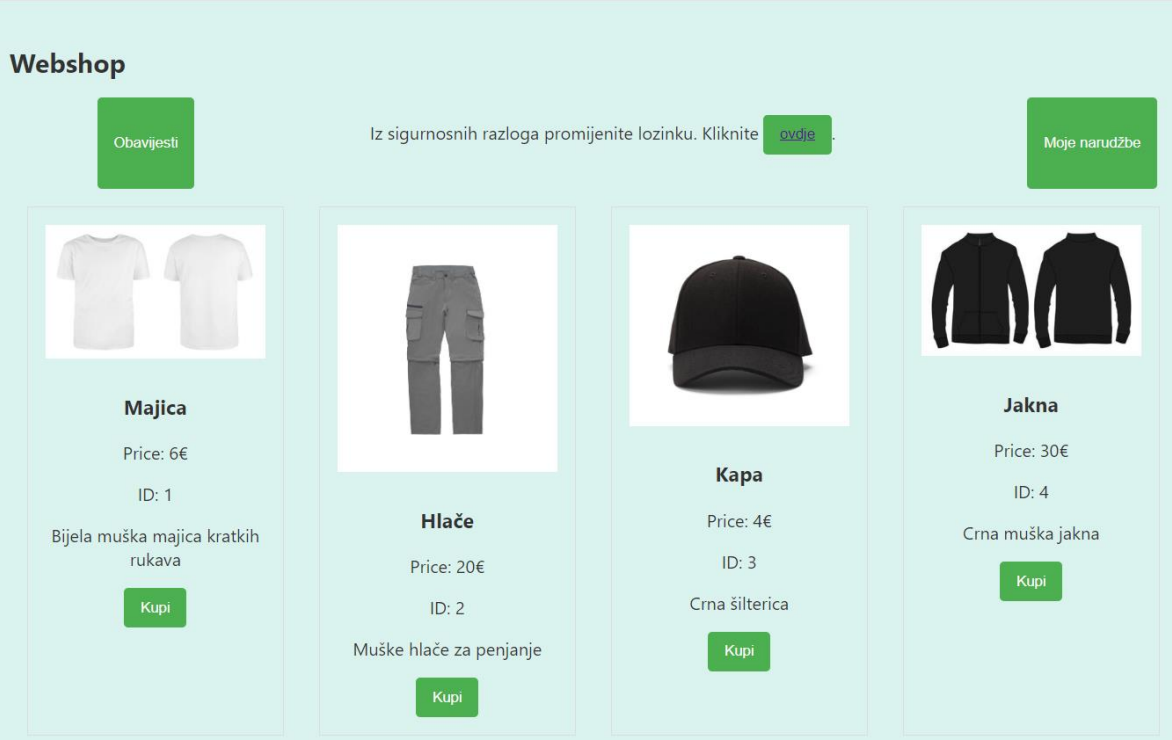
Izvor: Autor

Opcija --data postavlja podatke koji se prenose u POST zahtjevu. No, osim nje dostupne su i druge:

- --crawl sustav automatski pretražuje internet za novim ciljevima
- -- uključuje slijepi XSS sadržaj u svaki parametar HTML obrasca tijekom skeniranja
- --timeout parametar određuje vremensko ograničenje obrade za zahtjev
- --file navodi korisničku datoteku koja će se koristiti u testiranju

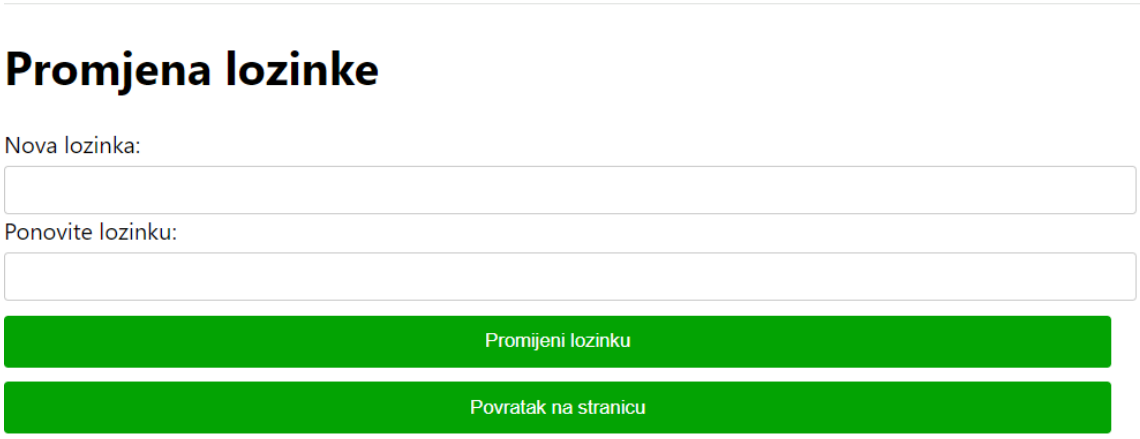
6.3 Primjer CSRF napada u aplikaciji i testiranje alata za skeniranje

Za prikaz CSRF napada u sklopu ranjive aplikacije napravljena je jednostavna Internet trgovina u koju se korisnik najprije mora prijaviti putem forme. Prijavom u aplikaciju, točnije trgovinu, za klijenta je stvorena sjednica koja se koristi za praćenje stanja ili aktivnosti korisnika tijekom njegove interakcije s web aplikacijom. Podaci o sjednici spremljeni su u kolačić i poslani klijentu koji ga kasnije šalje uz svaki zahtjev prema serverskoj strani aplikacije kako bi bio identificiran. Korisnik na svom profilu vidi obavijesti među kojima se nalazi i ona od napadača: predloženo mu je da promijeni svoju lozinku iz sigurnosnih razloga, kao na slici.



Slika 6.17: Korisničko sučelje aplikacije ranjive na CSRF napad
Izvor: Autor

Klikom na link korisnik je prebačen na napadačevu aplikaciju na kojoj se nalazi forma za promjenu lozinke.



Slika 6.18: Ranjiva forma za promjenu lozinke
Izvor: Autor

Korisnik može upisati novu lozinku, no nebitno je što će on upisati jer će napadač postaviti svoju novu lozinku umjesto tog unosa. Dio koda njegove aplikacije izgleda kao na primjeru programskog koda 6.4.

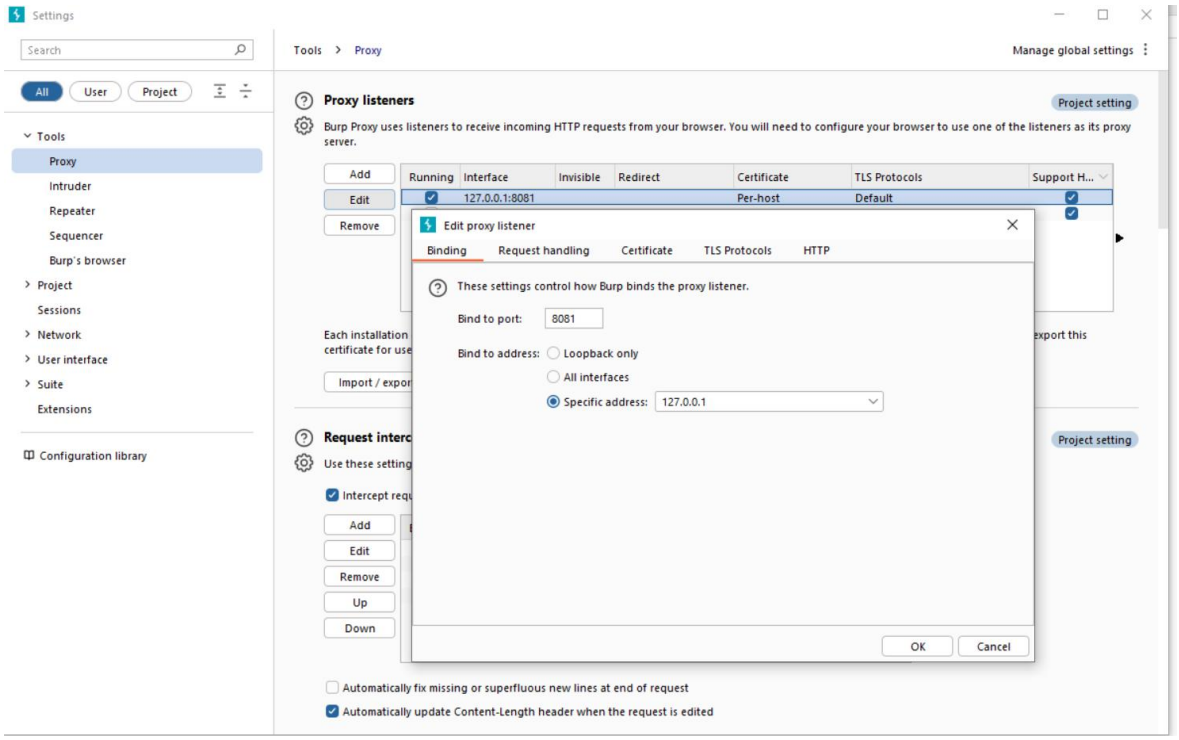
Programski kod 6.4: Dio koda zlonamjerne aplikacije za promjenu lozinke
Izvor: Autor

```
try{
    $conn = new
PDO("pgsql:host=$host;dbname=$database;user=$user;password=$password");
    $conn-> setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $conn->exec('SET NAMES \'UTF8\'');

    //upit za unos novog komentara
    $query = "INSERT INTO comments (name, comment) VALUES (:name,
:comment)";
    $stmt = $conn ->prepare($query);
    $stmt ->bindParam(':name, $name);
    $stmt ->bindParam(':comment', $comment);
    $stmt ->execute();
} catch(PDOException $e){
    echo json_encode(['error' => 'Greška prilikom unosa komentara: '
.$e ->getMessage()]);
}
```

Napadač je najprije morao presresti zahtjeve iz trgovine prema poslužitelju i pogledati njihovu strukturu. Iz tih zahtjeva koji su snimljeni zaključio je da se u ovom slučaju šalje GET zahtjev, a podaci se prenose kao dio URL-a. Dakle, u aplikaciji za promjenu lozinke koju je napadač napravio i podmetnuo korisniku kao obavijest, šalje se GET zahtjev koji će za novu lozinku prijavljenog korisnika postaviti proizvoljan niz, dok će korisnik misliti kako je njegova nova lozinka ono što je upisao u formu.

6.3.1 **Korištenje alata Burp Suite**

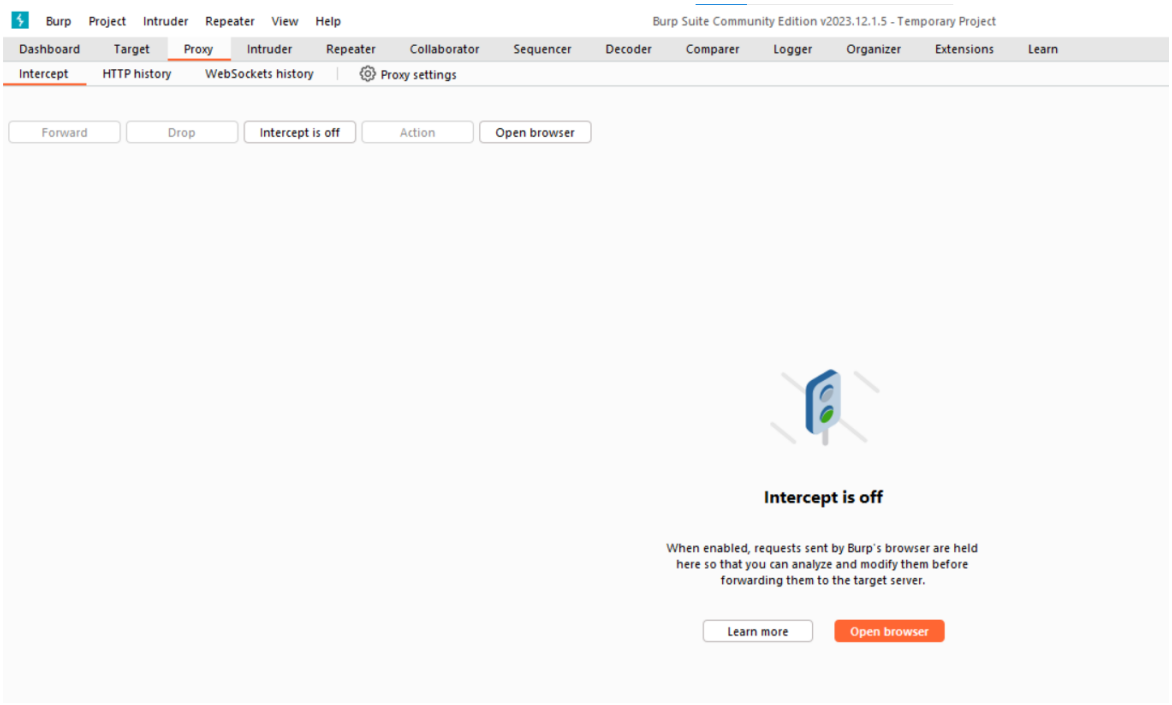
U 5. poglavlju predstavljen je Burp Suite, čiji je rad prikazan u nastavku. Za potrebe skeniranja aplikacije razvijene u sklopu ovog završnog rada korištena je *Community* verzija ovog alata jer su sve potrebne funkcionalnosti dostupne u toj inačici. Cilj korištenja ovog alata bio je provjeriti je li doista funkcija za promjenu lozinke na serverskoj strani aplikacije ranjiva na CSRF napade. Prije početka rada potrebno je konfigurirati posrednički poslužitelj, tj. *proxy* i primijeniti ga u pregledniku. Konfiguracija posrednika u alatu prikazana je na slici dolje. S obzirom da je zadani port 8080, promijenjen je u 8081 zbog serverskog dijela aplikacije. U postavkama računala i preglednika na isti način ručno je postavljen posrednički poslužitelj.



Slika 6.19: Konfiguracija posrednika u alatu Burp Suite

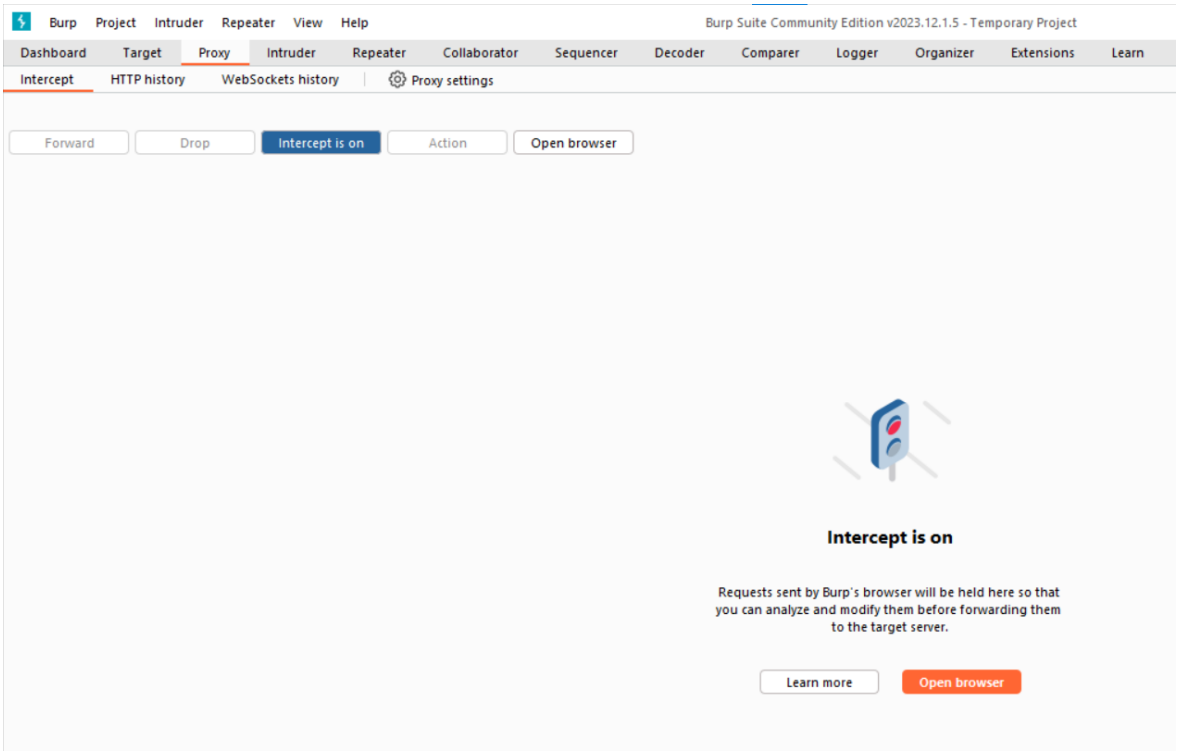
Izvor: Autor

Inicijalno stanje nakon postavljanja posrednika prikazano je na slici broj 6.20. U traci s alatima izabran je *Proxy*, pa *Intercept*. Klikom na gumb *Open browser*, otvara se zadani preglednik pomoću kojeg se pristupa aplikaciji.



Slika 6.20: Prikaz sučelja alata nakon postavljanja posrednika
Izvor: Autor

Nakon prijave u Internet trgovinu, u alatu Burp Suite potrebno je kliknuti gumb *Intercept is off* kako bi zahtjevi bili presretnuti i snimljeni. Alat sada predstavlja napadača.



Slika 6.21: Pokrenut posrednik
Izvor: Autor

Dio koda na klijentskoj strani aplikacije prikazan je na primjeru programskog koda 6.5. Nova lozinka i ponovljena nova lozinka prenose se kao parametri u URL-u.

Programski kod 6.5: Dio koda za promjenu lozinke
Izvor: Autor

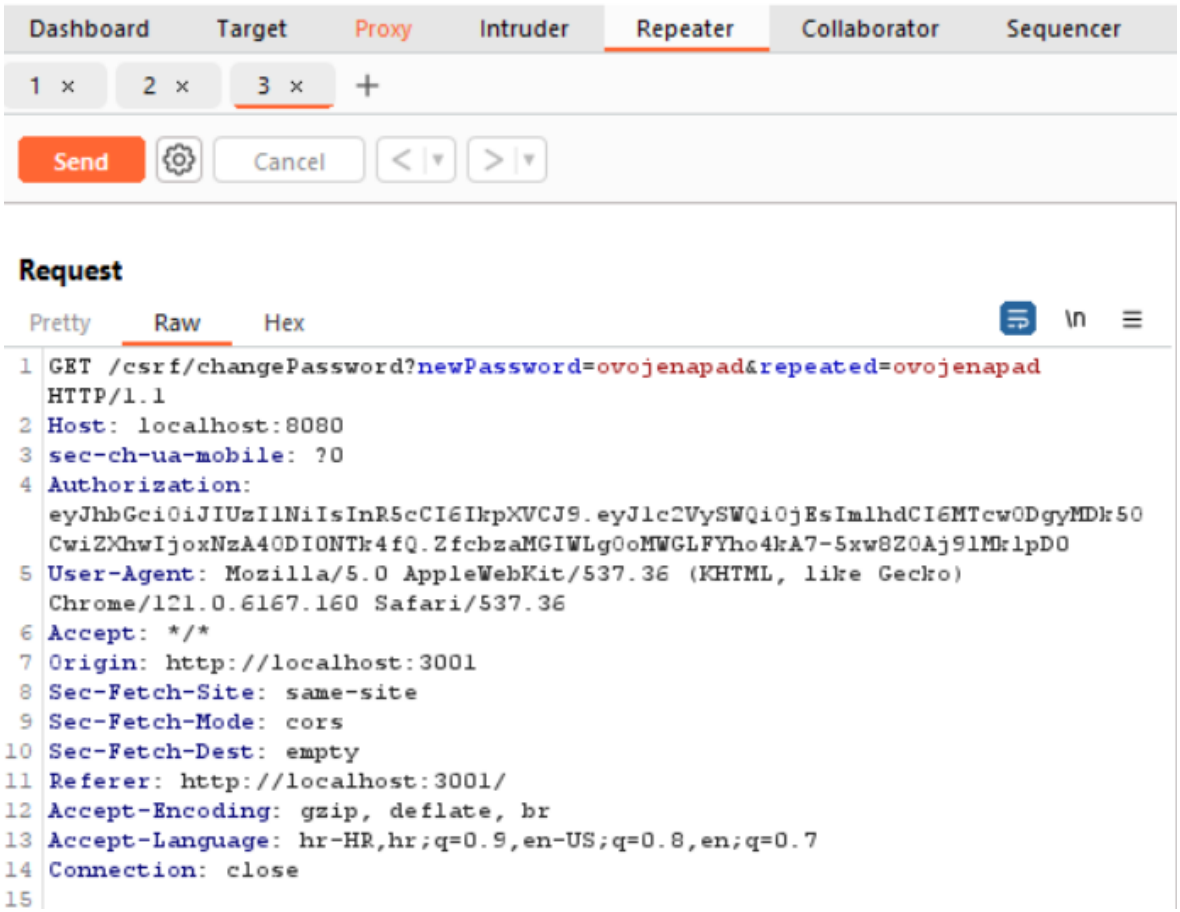
```
const url =
'http://localhost:8080/csrf/changePassword?newPassword=${newPassword}&repeated
=${repeatedPassword};
const res = await fetch(url,{
    headers: {
        'Authorization': '${token}'
    }
});

if(!res.ok){
    throw new Error('Problem sa serverom.');
}

window.alert('Lozinka uspješno promijenjena. Molimo prijavite se ponovno.');
```

Zahtjev koji je klijent poslao kada je korisnik potvrdio podatke iz forme u alatu je prikazan kao na slici broj 6.22.

Slika 6.22: HTTP GET zahtjev s podacima za promjenu lozinke

Izvor: Autor

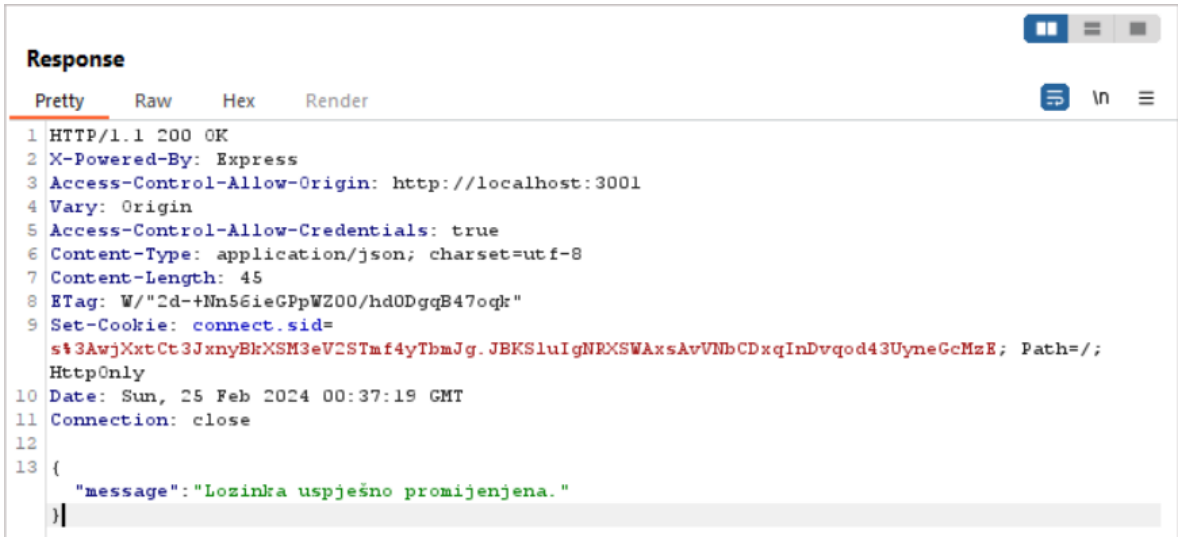
Action gumb nudi opciju slanja zahtjeva *Repeateru*. U repeateru je moguće mijenjati zahtjev, na primjer parametre koji se šalju u URL-u.



Slika 6.23: HTTP GET zahtjev izmijenjen u alatu

Izvor: Autor

Klikom na gumb *Send* izmijenjeni zahtjev poslan je serveru koji je uspješno promijenio lozinku, što se vidi iz odgovora prikazanog na slici broj 6.24.



Slika 6.24: Odgovor servera na izmijenjeni zahtjev

Izvor: Autor

Dakle, zahtjev je moguće presresti i poslati izmijenjen bez da server prepozna napad. Ova ranjivost može se spriječiti korištenjem CSRF tokena koji omogućavaju provjeru autentičnosti korisnika pri svakom zahtjevu.

7. ZAKLJUČAK

U zaključku ovog rada o praktičnoj primjeni Linux alata za sigurnosno testiranje, naglašava se ključna uloga ovih alata u očuvanju sigurnosti web aplikacija. Svaki od ovih prikazanih napada je problematičan na svoj način i predstavlja ozbiljnu opasnost, no ako bi trebalo izdvojiti jedan onda bi to bio SQL *injection* jer direktno može ugroziti sigurnost podataka koji se nalaze u bazi. Kroz analizu alata poput OWASP ZAP, XSStrike, SQLMap i Burp Suite potvrđeno je da oni igraju ključnu ulogu u identifikaciji i sprječavanju SQL *injectiona*, XSS i CSRF napada. Ovaj rad ne samo da potvrđuje važnost Linux alata za sigurnosno testiranje već i naglašava potrebu za stalnim praćenjem najnovijih trendova u sigurnosnom inženjeringu. Daljnje istraživanje i edukacija ključni su za održavanje koraka s evolucijom prijetnji te za osiguravanje dugoročne otpornosti web aplikacija.

U konačnici, implementacija sigurnosnih mjera treba biti agilna i prilagodljiva, a radovi poput ovog pridonose razvoju znanja potrebnog za stalno poboljšanje sigurnosti na internetu. Trebali bi pripaziti da imamo sigurnu komunikaciju s bazom, spriječimo unos zlonamjernog koda kroz forme i ono najvažnije za svakog korisnika na internetu je da ne ulazi na nesigurne linkove ili stranice. Ovaj rad pruža temelje za daljnje istraživanje i praksu u području web sigurnosti, potičući zajednicu na suradnju u stvaranju sigurnijeg digitalnog okoliša.

8. LITERATURA I IZVORI

[1] Bryan Sullivan and Vincent Liu: "Web Application Security: A Beginner's Guide"

[2] OWASP. “SQL Injection” SQL Injection | OWASP Foundation, 11.11.2023. https://owasp.org/www-community/attacks/SQL_Injection

[3] OWASP. “Cross Site Scripting” Cross Site Scripting | OWASP Foundation, 11.11.2023. https://owasp.org/www-community/attacks/xss/

[4] OWASP. “OWASP Zap Documentation” 11.11.2023. https://www.zaproxy.org/docs/

[5] “sqlmap Documentation” 11.11.2023. https://github.com/sqlmapproject/sqlmap/wiki

[6] “CSRF Documentation” 11.11.2023. https://portswigger.net/web-security/csrf

[7] https://spanning.com/blog/sql-injection-attacks-web-based-application-security-part-4/

[8] https://spanning.com/blog/cross-site-scripting-web-based-application-security-part-3/

[9] https://spanning.com/blog/cross-site-forgery-web-based-application-security-part-2/

[10] https://zaproxy.org

[11] https://en.m.wikipedia.org/wiki/File:Sqlmap_logo.png

[12] https://medium.com/@KnightFreak/cross-site-scripting-xss-attack-95bc33ffb56e

[13] https://portswigger.net/web-security/csrf

[14] Opensource “Open source documentation“ https://opensource.com/resources/what-open-source

[15] Acunetix “SQL Injection“ https://www.acunetix.com/websitesecurity/sql-injection2/

[16] OWASP Cheat Sheet Series “SQL Injection Prevention“ https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

[17] CERT “Analiza XSS sigurnosnih propusta“ https://www.cert.hr/NCAnaXssSIGPRO/ccert-pubdoc-2006-05-157/

[18] CERT “CSRF napadi“ https://www.cis.hr/www.edicija/LinkedDocuments/NCERT-PUBDOC-2010-04-297.pdf

[19] Gilberto Najera-Gutierrez, Juned Ahmed Ansari:“Web Penetration Testing with Kali Linux“

[20] Codiga „ OWASP ZAP “ https://www.codiga.io/blog/owasp-zap/

[21] PortSwigger „Burp Suite documentation“ https://portswigger.net/burp/documentation

[22] testsigma „Database testing tools“ https://testsigma.com/blog/database-testing-tools/

[23] Medium „XSStrike and Cypress“ https://medium.com/@elyassinisafouat2/xsstrike-and-cypress-finding-xss-vulnerabilities-testing-and-safe-your-web-apps-84e0cdc51afc

[24] Medium „XSStrike-A tool to detect XSS“ https://medium.com/@aswinchandran274/xsstrike-a-tool-to-detect-xss-e6b54b5f6f5b

9. OZNAKE I KRATICE

CSP - Content Security Policy (Sigurnosne politike sadržaja)

DBMS - Database Management System (Sustav za upravljanje bazom podataka)

DOM - Document Object Model (Objektni model dokumenta)

HTML - Hypertext Markup Language (Hipertekst Markup Jezik)

HTTP - Hyper Text Transfer Protocol (Protokol za prijenos hiperteksta)

OWASP ZAP - Open Worldwide Application Security Project Zed Attack Proxy (Otvoreni svjetski projekt za sigurnost aplikacija - Proxy za napade Zed)

PHP - Hypertext Preprocessor (Hiper tekst Preprocesor)

SQL – Structured Query Language (Strukturirani jezik upita)

SQLi – Structured Query Language Injection (Injektiranje u strukturirani jezik upita)

URL - Uniform Resource Locator (Jedinstveni identifikator resursa)

WAF - Web Application Firewall (Vatrozid za web aplikacije)

XSS – Cross-Site Scripting (Unakrsno ubacivanje skripti)

10. SAŽETAK

Naslov: Alati za testiranje sigurnosti u okruženju otvorenog koda

U ovom diplomskom radu demonstrira se korištenje alata za testiranje sigurnosti Linuxa kroz razvoj osnovne web aplikacije. Fokusiramo se na identifikaciju i ispravku ranjivosti kao što su SQL injection, *Cross-Site Scripting* (XSS) i *Cross-Site Request Forgery* (CSRF) pomoću alata SQLMap, OWASP ZAP, XSStrike i Burp Suite. Rad započinje razvojem osnovne web aplikacije, a zatim se koriste navedeni alati za identifikaciju potencijalnih ranjivosti. Kroz testiranje, otkrivaju se i ispravljaju potencijalne SQL *injection* ranjivosti kako bi se osigurala sigurnost podataka. Zatim se opisuje XSS ranjivost i korištenje alata XSStrike i OWASP ZAP za identifikaciju i ispravak potencijalnih napadačkih točaka. Nakon toga se skenira CSRF ranjivost koristeći Burp Suite alat. Na kraju se analizira rezultat testiranja i generira se izvještaj o ranjivostima. Ovaj rad pruža praktičan uvid u identifikaciju ranjivosti u web aplikacijama te naglašava važnost sigurnosnog testiranja i korištenje Linux alata za osiguravanje sigurnosti i integriteta.

Ključne riječi: Linux, SQLi, XSS, CSRF, testiranje aplikacija.

11. ABSTRACT

Title: Security Testing Tools in Open Source Environment

This thesis explores the practical application of Linux security testing tools using a simple web application development example. We focus on identifying and fixing vulnerabilities such as SQL Injection, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF) using tools like SQLMap, OWASP ZAP, XSStrike, and Burp Suite. The thesis begins with the development of a basic web application, followed by the use of the mentioned tools to identify potential vulnerabilities. Through testing, potential SQL Injection vulnerabilities are discovered and addressed to ensure data security. Then, XSS vulnerability and the use of XSStrike and OWASP ZAP tools to identify and fix potential attack points are described. Subsequently, CSRF vulnerability is scanned using Burp Suite tool. Finally, the test results are analyzed, and a vulnerability report is generated. This thesis provides a practical insight into vulnerability identification in web applications and emphasizes the importance of security testing and the use of Linux tools to ensure the security and integrity.

Keywords: Linux, SQLi, XSS, CSRF, application testing.

IZJAVA O AUTORSTVU ZAVRŠNOG RADA

Pod punom odgovornošću izjavljujem da sam ovaj rad izradio/la samostalno, poštujući načela akademske čestitosti, pravila struke te pravila i norme standardnog hrvatskog jezika. Rad je moje autorsko djelo i svi su preuzeti citati i parafraze u njemu primjereno označeni.

Mjesto i datum	Ime i prezime studenta/ice	Potpis studenta/ice
U Bjelovaru, 18.05.2024.	IVAN VUKUŠIĆ	Ivan Vukušić

U skladu s čl. 58, st. 5 Zakona o visokom obrazovanju i znanstvenoj djelatnosti, Veleučilište u Bjelovaru dužno je u roku od 30 dana od dana obrane završnog rada objaviti elektroničke inačice završnih radova studenata Veleučilišta u Bjelovaru u nacionalnom repozitoriju.

Suglasnost za pravo pristupa elektroničkoj inačici završnog rada u nacionalnom repozitoriju

IVAN VUKUŠIĆ
ime i prezime studenta/ice

Dajem suglasnost da tekst mojeg završnog rada u repozitorij Nacionalne i sveučilišne knjižnice u Zagrebu bude pohranjen s pravom pristupa (zaokružiti jedno od ponuđenog):

a) Rad javno dostupan
b) Rad javno dostupan nakon _____ (upisati datum)
c) Rad dostupan svim korisnicima iz sustava znanosti i visokog obrazovanja RH
d) Rad dostupan samo korisnicima matične ustanove (Veleučilište u Bjelovaru)
e) Rad nije dostupan

Svojim potpisom potvrđujem istovjetnost tiskane i elektroničke inačice završnog rada.

U Bjelovaru, 18. 05. 2024.

Ivan Vukušić
potpis studenta/ice