UNIVERZITET U SARAJEVU
ELEKTROTEHNIČKI FAKULTET
ODSJEK ZA RAČUNARSTVO I INFORMATIKU

Sigurnosni aspekti e-commerce aplikacije

ZAVRŠNI RAD

- PRVI CIKLUS STUDIJA -

Student:
Eldar Panjeta

Mentor:
Red. prof. dr. Dženana Đonko

Sarajevo,
juli 2022.

Sažetak

U ovom radu opisani su najčešći sigurnosni propusti e-commerce aplikacija. Predstavljena je historija sigurnosnih propusta, razlozi nastanka te njihove posljedice. Odabrani su propusti koji se smatraju najčešćim i čije posljedice mogu biti velike, te su oni detaljno analizirani. Za potrebe rada, razvijena je e-commerce aplikacija WatchMe kako bi se pokazala implementacija zaštite u Python okruženju. U sklopu ove aplikacije, tretirani su svi opisani sigurnosni propusti. Na kraju, dat je kratak osvrt na važnost zaštite e-commerce aplikacija.

Abstract

This project describes the most common security issues in e-commerce applications. The history of security failures, the reasons for their occurrence and their consequences are presented. Failures that are considered to be the most common and whose consequences can be great were selected and analyzed in detail. For the purposes of this project, an e-commerce application WatchMe was developed to demonstrate the implementation of protection in Python environment. As part of this application, all described security issues have been treated. Finally, a brief overview of the importance of protecting e-commerce applications is given.

Elektrotehnički fakultet, Univerzitet u Sarajevu
Odsjek za Računarsvo i informatiku
Red. prof. dr. Dženana Đonko

Postavka zadatka završnog rada I ciklusa:
Sigurnosni aspekti e-commerce aplikacije

U okviru rada je potrebno:

- navesti i analizirati najčešće sigurnosne propuste e-commerce aplikacija
- implementirati e-commerce aplikaciju u Python programskom jeziku
- objasniti implementaciju zaštite od navedenih sigurnosnih propusta na primjeru razvijene aplikacije

Polazna literatura:

1. Bishop, M., Introduction to computer security. Addison-Wesley Boston, 2005.
2. Hoffman, A., Web Application Security: Exploitation and Countermeasures for Modern Web Applications. O'Reilly Media, 2020.

Red. prof. dr. Dženana Đonko

Univerzitet u Sarajevu
Elektrotehnički fakultet
Odsjek za Računarstvo i informatiku

Izjava o autentičnosti radova

Završni rad
I ciklusa studija

Ime i prezime: Eldar Panejta
Naslov rada: Sigurnosni aspekti e-commerce aplikacije
Vrsta rada: Završni rad Prvog ciklusa studija
Broj stranica: 50

Potvrđujem:

- da sam pročitao dokumente koji se odnose na plagijarizam, kako je to definirano Statutom Univerziteta u Sarajevu, Etičkim kodeksom Univerziteta u Sarajevu i pravilima studiranja koja se odnose na I i II ciklus studija, integrirani studijski program I i II ciklusa i III ciklus studija na Univerzitetu u Sarajevu, kao i uputama o plagijarizmu navedenim na web stranici Univerziteta u Sarajevu;
- da sam svjestan univerzitetskih disciplinskih pravila koja se tiču plagijarizma;
- da je rad koji predajem potpuno moj, samostalni rad, osim u dijelovima gdje je to naznačeno;
- da rad nije predat, u cjelini ili djelimično, za stjecanje zvanja na Univerzitetu u Sarajevu ili nekoj drugoj visokoškolskoj ustanovi;
- da sam jasno naznačio prisustvo citiranog ili parafraziranog materijala i da sam se referirao na sve izvore;
- da sam dosljedno naveo korištene i citirane izvore ili bibliografiju po nekom od preporučenih stilova citiranja, sa navođenjem potpune reference koja obuhvata potpuni bibliografski opis korištenog i citiranog izvora;
- da sam odgovarajuće naznačio svaku pomoć koju sam dobio pored pomoći mentora i akademskih tutora/ica.

Sarajevo, juli 2022

Potpis:

_____

Eldar Panjeta

Sadržaj

Popis slika

Popis tabela

Popis isječaka koda

Poglavlje 1

Uvod

Dolaskom interneta, način na koji ljudi obavljaju svoje poslove se značajno promijenio. Ovo se odnosi na svaku granu privrede a posebno na trgovinu. Kompanije širom svijeta koje se bave prodajom ili nude određene usluge koriste softver, najčešće e-commerce aplikacije, u svim aspketima svog poslovanja. Softver je postao jedan od najbitnijih fakora uspjeha svake kompanije i jako je važno da on, kao i svaki proizvod, bude siguran za korištenje. Ova sigurnost znači da je potrebno osigurati pouzdanost, integritet i privatnost podataka kako korisnika softvera tako i samog softvera.

1.1 Cilj rada

Cilj rada je opis i analiza najčešćih sigurnosnih propusta e-commerce aplikacija i njihova implementacija unutar aplikacije razvijene koristeći Python okruženje.

1.2 Struktura rada

Rad se sastoji od 6 poglavlja, a to su:

1. Uvod
2. Sigurnosni propusti
3. Specifikacija aplikacije WatchMe
4. Analiza i implementacija najčešćih sigurnosnih propusta
5. Analiza i implementacija identifikacijskih i autentifikacijskih propusta
6. Zaključak

Prvo poglavalje predstavlja uvod u rad i u temu rada. Sastoji se iz prezentacije teme, ciljeva rada te metodologija koje su primjenjene.

U drugom poglavlju predstavlja se pojam sigurnosni propust. U njemu su definisani najvažniji pojmovi važni za sam rad, te je napravljen pregled historijskog razvoja sigurnosnih propusta

softvera, opisani su razlozi zbog čega nastaju i data je osnovna klasifikacija sigurnosnih propusta.

U trećem poglavlju je data specifikacija aplikacije WatchMe koja se razvijala za potrebe ovog rada, to jeste za implementaciju zaštite sigurnosnih propusta. Unutar ovog poglavlja je dat opis aplikacije, njeni korisnici, namjena, funkcionalni i nefunkcionalni zahtjevi, kao i dijagram slučajeva upotrebe. Poslovni procesi su opisani kroz scenarije, te je također predstavljeno okruženje za implementaciju kao i arhitektura aplikacije.

U četvrtom poglavlju predstavljena su 3 velika propusta: SQL Injection, Cross-site scripting i Cross-site request forgery. Dat je njihov opis, kada i zbog čega se javljaju, koje su njihove posljedice te je opisano kako se zaštiti od njih prilikom razvoja aplikacije koristeći Python okruženje.

U petom poglavlju predstavljeni su propusti vezani za nedovoljnu zaštitu lozinke, kontrole pristupa i sesija. Također, opisano je kako implementirati zaštitu koristeći Python okruženje.

Šesto poglavlje predstavlja zaključak rada, odnosno rezime onog što je urađeno te diskusiju rezultata predstavljenih u prethodnim poglavljima.

Osim ovih poglavlja, rad sadrži i jedan prilog u kojem se nalaze scenariji za svaku funkcionalnost aplikacije.

1.3 Metodologije rada

U svrhu pisanja ovog rada, bilo je neophodno istražiti široku oblast sigurnosnih propusta te izdvojiti one koji se smatraju najvažnijima odnosno one koji se najčešće javljaju. Odabrani sigurnosni propusti su detaljno opisani. Potom je bilo neophodno izabrati tehnologiju za razvoj e-commerce aplikacije i upoznati se sa novim programskim jezikom i okruženjem. Aplikacija je razvijena na način da budu pokriveni svi odabrani sigurnosni propusti. Nakon toga, detaljno je opisana implementacija zaštite u samoj aplikaciji.

1.4 Napomene o stilu pisanja

U cijelom radu korišten je font *Times New Roman*, sa veličinom slova 12 pt. Naslovi poglavlja su veličine 26 pt. Programski kodovi su pisani koristeći font Courier, sa veličinom slova 10pt. Korišten je IEEE stil referenciranja. Svi stručni termini čiji prijevodi na bosanski jezik ne postoje označeni su *italic* stilom.

Poglavlje 2

Sigurnosni propusti

Kada neko provali u računarski sistem, ta osoba koristi grešake u procedurama, tehnologiji ili menadžmentu sistema, čime sebi omogućava neovlašteni pristup ili akcije. Konkretan neuspjeh sistema se naziva sigurnosna slabost ili propust. Korištenje tog propusta za narušavanje sigurnosne politike stranice naziva se iskorištavanje ranjivosti, dok osoba koja to radi se naziva napadač ili haker. [1] Korisno je znati kako su se sigurnosni propusti mijenjali kroz historiju, šta su uzroci njihovog nastanka te kako ih možemo podijeliti, pa će se u ovom poglavlju obraditi navedene teme.

2.1 Historijski razvoj

Možemo reći da sami počeci sigurnosti računara potiču iu 1967. godine kada je Willis Ware govorio o sigurnosti na *Joint Computer* konferenciji. Međutim 1970-tih i 1980-ih godina nisu postojale ozbiljne računarske prijetnje s obzirom da su računari i internet još uvijek bili u razvoju. Najčešće su prijetnje dolazile od zlonamjernih zaposlenika koji su dobili neovlašteni pristup osjetljivim dokumentima. [1]

1971. godine, Bob Thomas je kreirao eksperimentalni program *Creeper* koji se smatra prvim računarskim crvom. Računarski crv je vrsta zlonamjernog softvera čija je primarna funkcija da se samoreplicira i inficira druge računare dok u međuvremenu ostaje aktivan na zaraženim sistemima. Godinu dana kasnije, Ray Tomlinson je razvio prvi anti-virus, softver za prevenciju, otkrivanje i uklanjanje štetnog softvera, pod nazivom *Reaper* koji se kretao kroz ARPANET (*Advanced Research Projects Agency Network*) i uspješno brisao *Creeper* virus.

Morris, prvi virus na Internetu, je razvijen 1988. godine te je zadobio značajnu medijsku pažnju. Iskorištavao je povjerenje osoba koje nisu koristile lozinku za prijavu na račun kao i slabe lozinke. [2] Vlada SAD-a procijenjuje da je šteta koju je nanio ovaj virus oko 10 miliona američkih dolara. Njegov kreator, Robert Morris, je osuđen na 3 godine zatvora i ovo je prva presuda za računarske prevare u SAD-u. Izvorni kod *Morris* virusa se nalazi na floppy disku i čuva se kao eksponat u Muzeju kompjuterske historije u Kaliforniji. [3]

Početkom 1990-tih godina, razvijen je *World Wide Web* i prve web stranice. U početku, nije bio previše popularan. Uglavnom je bio korišten za razmjenu dokumenata zapisanih u HTML-u (*HyperText Markup Language*). Samo nekolicina stranica je omogućavala korisnicima da šalju podatke nazad na server kako bi na neki način izmijenili sadržaj stranice. Početkom 2000-tih godina, stranice su počele čuvati podatke koje je unosio korisnik i mijenjati određene funkcionalnosti stranice na osnovu unesenih podataka. Ubrzo su se počele razvijati prve društvene mreže, blogovi, web aplikacije (aplikacijski softver koji radi na web serveru) i slično. Hakeri su iskoristili ovu veliku promjenu i pronašli nove načine za izvesti napade. Njihova glavna meta

bili su korisnici stranica i njihovi podaci. Sve više bitnih informacija kao što su vojna komunikacija i bankovni transferi su prešli na web aplikacije. Nažalost, sigurnost na tim stranicama je bila loša. Sredinom 2000-ih godina, izvedeni su prvi veliki napadi koji su rezultirali privremenim gašenjem poznatih stranica kao što su *Yahoo!*, *Amazon* i *eBay*. Kako su web aplikacije napredovale, sa porastom kompleksnosti aplikacije rastao je i broj sigurnosnih propusta. [3]

2.2 Razlog nastanka

2.2.1 Ljudski faktor

Različiti ljudi, različitih profesija dolaze u kontakt sa računarskim sistemom. Oni imaju različita shvatanja o tome kako sistem funkcioniše, šta je sposoban da radi i šta je potrebno da se promijeni. Bez adekvatnih pravila u organizaciji, dolazi do problema u komunikaciji između programera, njihovih nadređenih i korisnika sistema. Ovo duboko utiče na slabost softvera. Određene komponente sistema, kao što su web serveri i baze podataka, mogu ostati pogrešno konfigurisane ako ne postoji kvalitetna komunikacija. Učestale su i greške samih programera. One se mogu desiti kao rezultat umora, neinformisanosti, nedostatka koncentracije i slično. [4]

Moguće su i greške korisnika sistema. 52% preduzeća priznaje da su zaposlenici njihova najveća slabost u IT sigurnosti. Radnik može napraviti greške koje dovode podatke ili sisteme njihove kompanije u opasnost. To se može desiti zato što je nepažljiv i slučajno napravi grešku ili zato što mu nije osigurana potrebna obuka za rad na tim sistemima. [5]

2.2.2 Ekonomski faktor

Ekonomisti su oduvijek bili svjesni povezanosti između ekonomije i sigurnosti. Recimo, bogatije nacije i države su mogle priuštiti velike vojske. Jasno je da ovo vrijedi i za sigurnost računarskih sistema. Kao primjer možemo navesti firmu koja ne želi uložiti veću sumu novca u sigurnost njihove web stranice. Ta web stranica će postati ranjiva, te podložna namjernim pa čak i nenamjernim napadima. Nenamjerni napadi se mogu javiti kada firma očekuje od svojih korisnika da imaju instaliran anti-virus. Korisnik bez anti-virusa može lahko zaraziti nezaštićen računarski sistem i nanijeti mu veliku štetu. Mnogi sigurnosni propusti na Windows operativnom sistemu su nastali kao razlog nedovoljnog ulaganja u sigurnost. [6]

2.2.3 Nedostatak testiranja

Testiranje softvera je jedan od načina kojim se osigurava da softver nema sigurnosnih propusta. Međutim, programeri kao i drugi ljudi, više vole raditi stvari koje su im zanimljive nego neke dosadne zadatke. Pa tako, često izbjegavaju raditi na testiranju softvera jer im to nije zanimljivo. [6]

Programeri, kao i QA inžinjeri su fokusirani da testiraju funkcionalnosti softvera i da li on radi kako je klijent zamislio. Za testiranje sigurnosti zaduženi su takozvani *White Hat* hakeri odnosno etički hakeri. Oni imaju za cilj da učine da softver ne funkcioniše onako kako bi trebao u cilju pronalaska načina da izvrše određene napade na softver. Najviše su fokusirani da od softvera dobiju odgovore koji sadrže skrivene i osjetljive informacije ili da pošalju informacije na server koje program ne bi trebao primiti od običnog korisnika. [7]

2.3 Klasifikacija

CWE lista (*Common Weakness Enumeration*) najčešćih i najuticajnijih propusta, razdvaja sigurnosne propuste u tri kategorije: slabe odbrane (*porous defenses*), rizično upravljanje resursima i nesigurna interakcija između komponenti.

U slabe odbrane spadaju propusti koji bi mogli omogućiti korisnicima da zaobiđu ili lažiraju procese autentifikacije i autorizacije. Autentifikacija provjerava identitet nekoga ko pokušava pristupiti sistemu, dok je autorizacija skup dozvola dodijeljenih korisniku za pristup i korištenje sistema. Tipični predstavnici ove kategorije sigurnosnih propusta su slabo kodiranje lozinke, nedovoljno zaštićeni lični dokumenti i sesije koje nemaju vremensko ograničenje.

Mnogi propusti spadaju u kategoriju rizičnog upravljanja resursima kao što su memorija, funkcije i *open-source frameworks* (besplatni šabloni za razvoj softvera). Tokom faza dizajna i razvoja web aplikacija i web stranica, važno je da sve komponente preuzete od neke treće strane, koje treba uključiti u arhitekturu, budu skenirane u potrazi za ranjivostima. Primjer takvih komponenti su biblioteke. Biblioteka je skup podataka i programskog koda koji se koristi da programeru olakša razvoj softverskih aplikacija.

Još jedna kategorija propusta je nesigurna interkacija između komponenti. Današnje aplikacije šalju i primaju podatke kroz širok spektar servisa i procesa. Web aplikacije i web stranice moraju implementirati *zero-trust* pristup koji podrazumijeva da je svaki unos sumnjiv dok se ne potvrdi da dolazi iz pouzdanog izvora i da ima adekvatnu svrhu. [8]

Neke organizacije poput *The Open Web Application Security Project* (OWASP), nemaju direktenu klasifikaciju sigurnosnih propusta. Međutim, OWASP svake godine objavljuje aktuelnu listu 10 najutjecajnih propusta web aplikacija pod nazivom *OWASP TOP 10*. OWASP TOP 10 je standardni dokument za podizanje svijesti o sigurnosti web aplikacija. Predstavlja širok konsenzus o najkritičnijim sigurnosnim rizicima za web aplikacije. U ovom radu će se analizirati i tretirati propusti sa te liste.

2.4 Rezime

U ovom poglavlju definisani su pojmovi sigurnosni propust, napad i haker. Prikazan je kratak pregled razvoja sigurnosnih propusta kroz historiju te opisani su razlozi njihovog nastanka. Naposlijetku je predstavljena njihova klasifikacija.

U sljedećem poglavlju će biti predstavljena specifikacija aplikacije WatchMe koja će se razvijati za potrebe ovog rada.

Poglavlje 3

Specifikacija aplikacije WatchMe

U ovom poglavlju će biti prikazana specifikacija aplikacije koja će biti implementirana u cilju demonstriranja i tretiranja određenih sigurnosnih propusta kroz Python okruženje. Predstavit će se korisnici aplikacije, funkcionalni i nefunkcionalni zahtjevi, pojedini dijagrami te će biti opisano okruženje za implementaciju i arhitektura aplikacije.

3.1 Potreba za aplikacijom WatchMe

Širenjem Interneta kao najpopularnije računarske mreže mnoge stvari u svijetu trgovine su se počele mijenjati. Kupovina proizvoda ali i njihova prodaja je značajno olakšana. Kupac sada može iz udobnosti svog doma izvršiti kupovinu skoro bilo čega, bez da troši vrijeme na odlazak u prodavnicu i čekanje u redu. Osim kupovine, može dobiti i određene usluge, kao što su online bankarstvo, platiti režije i slično. Prodavač sada može prodati više proizvoda ili ponuditi više usluga i nije više ograničen na svoju lokaciju, može poslovati sa ljudima širom svijeta. Veliki broj kompanija počeo je prodavati ili nuditi svoje usluge putem Interneta. Za ovu potrebu, počele su se razvijati web prodavnice odnosno e-commerce aplikacije.

Danas, gotovo da ne možemo zamisliti neku kompaniju koja prodaje ili nudi usluge a da ne radi to i putem Interneta. Pa tako i lokalna prodavnica satova želi da proširi svoj posao tako što će omogućiti upravo to.

3.2 Namjena aplikacije WatchMe

Namjena e-commerce aplikacije WatchMe je da obezbjedi online prodaju satova ove lokalne prodavnice. Aplikacija će pomoći korisnicima da vrše pregled svih dostupnih satova u prodavnici, izaberu svoj idealni sat te da izvrše kupovinu online plaćanjem. Također, aplikacija će zasigurno imati i značajnu marketinšku ulogu.

3.3 Korisnici aplikacije WatchMe

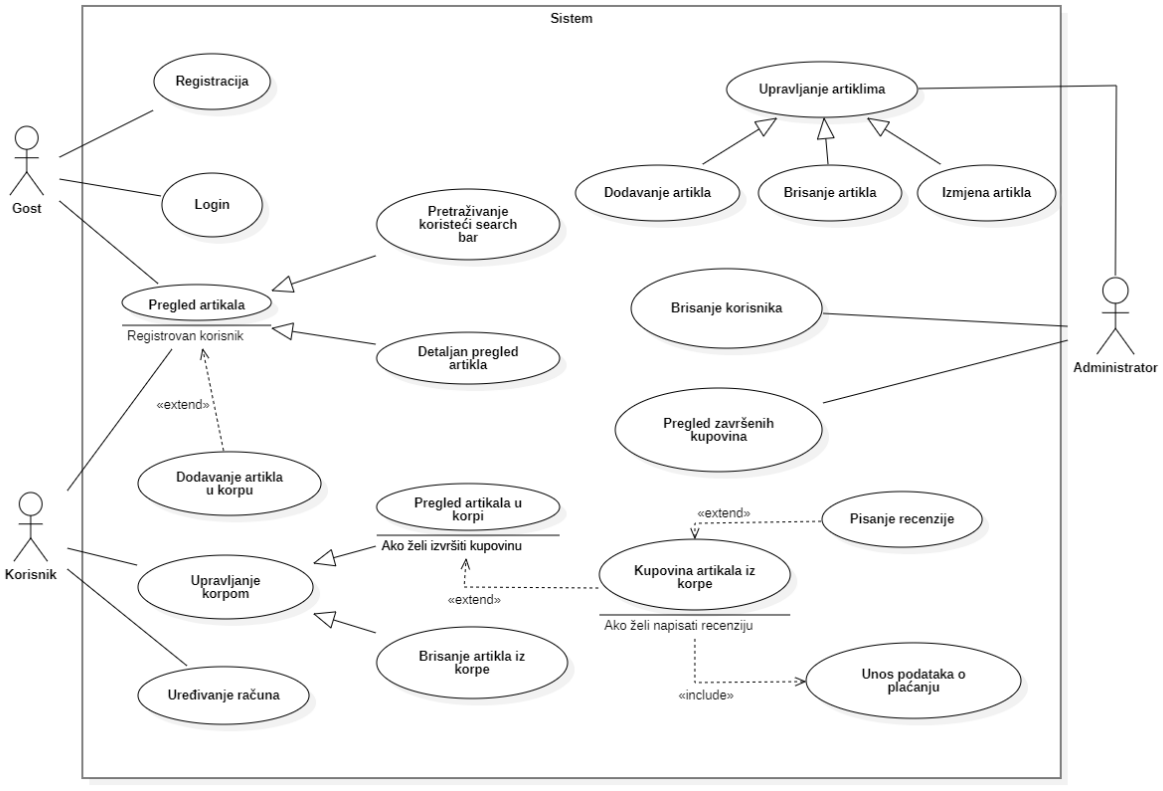
Aplikacija WatchMe će biti dostupna svim korisnicima, s tim da korisnik mora biti registrovan da bi mogao izvršiti kupovinu dok u suprotnom, on može samo pregledati artikle koji se nalaze na stranici. Također, registrovani korisnik ima mogućnost ostavljanja recenzija na satove koje je kupio. Administrator će imati mogućnost upravljanja bazom podataka aplikacije, što podrazumijeva upravljanje satovima u ponudi, korisnicima, recenzijama i slično.

3.4 Funkcionalnosti aplikacije

Funkcionalnosti koje ova aplikacija treba da pruži su:

- Gost ima mogućnost registracije i logina
- Gost ima mogućnost pregleda svih artikala
- Gost ima mogućnost detaljnog pregleda artikla
- Gost ima mogućnost pretraživanja artikala
- Korisnik ima sve mogućnosti kao i gost
- Korisnik ima mogućnost dodavanja artikla u korpu
- Korisnik ima mogućnost pregleda artikala u korpi
- Korisnik ima mogućnost brisanja artikala iz korpe
- Korisnik ima mogućnost izmjene ličnih podataka
- Korisnik ima mogućnost kupovine artikala iz korpe
- Korisnik ima mogućnost ostavljanja recenzija
- Administrator ima mogućnost dodavanja novih artikala, izmjene i brisanja postojećih
- Administrator ima mogućnost brisanja korisnika
- Administrator ima mogućnost pregleda završenih kupovina

Na slici 3.1, uz pomoć dijagrama slučajeva upotrebe, date su osnovne funkcionalnosti aplikacije WatchMe



Slika 3.1: Dijagram slučajeva upotrebe za WatchMe prodavnicu

Većina funkcionalnosti aplikacije je opisana u obliku scenarija. U tabelama koje slijede prikazani su scenariji i tokovi događaja za ključne procese aplikacije.

U tabeli 3.1 je prikazan scenarij za aktivnost pregled artikala dok se u tabelama 3.2, 3.3, 3.4 i 3.5 nalaze tokovi događaja za uspješne i alternativne uspješne završetke.

Naziv	Pregled artikala
Opis	Korisnik i gost imaju mogućnost pregleda svih artikala u sistemu
Vezani zahtjevi	/
Preduslovi	/
Posljedice – uspješan završetak	Izlistavaju se svi dostupni artikli
Posljedice – neuspješan završetak	/
Primarni akteri	Korisnik i gost
Ostali akteri	/
Glavni tok	Korisnik ili gost mogu da vrše pregled jednog ili više artikala koji su dostupni. Kada zatraže pregled, svi artikli se izlistavaju u vidu liste.
Proširenja/alternative	Pretraga putem search bar-a Detaljan pregled jednog artikla Dodavanje artikla u korpu

Tabela 3.1: Pregled artikala - scenarij

Korisnik	Sistem
1. Odabir opcije za pregled artikala	
	2. Prikaz svih dostupnih artikala

Tabela 3.2: Pregled artikala - tok događaja, uspješan završetak

Korisnik	Sistem
1. Odabir opcije za pregled artikala uz upis nekih ključnih riječi u search bar	
	2. Provjera unosa u search bar
	3. Prikaz svih dostupnih artikala koji na neki način zadovoljavaju korisnički unos u search bar-u

Tabela 3.3: Pregled artikala - tok događaja, uspješan (alternativni) završetak 1

Korisnik	Sistem
1. Odabir opcije za pregled artikala	
	2. Prikaz svih dostupnih artikala
3. Odabir opcije za detaljan prikaz konkretnog artikla	
	4. Prikaz detaljnih informacija o konkretnom artiklu

Tabela 3.4: Pregled artikala - tok događaja, uspješan (alternativni) završetak 2

Korisnik	Sistem
1. Odabir opcije za pregled artikala	
	2. Prikaz svih dostupnih artikala
3. Odabir opcije za detaljan prikaz konkretnog artikla	
	4. Prikaz detaljnih informacija o konkretnom artiklu
5. Odabir opcije za dodavanje artikla u korisničku korpu	
	6. Ažuriranje stanja korisničke korpe sa novim artiklom

Tabela 3.5: Pregled artikala - tok događaja, uspješan (alternativni) završetak 3

U tabeli 3.6 je prikazan scenarij za aktivnost registracija dok se u tabelama 3.7 i 3.8 nalaze tokovi događaja za uspješni i neuspješni završetak.

Naziv	Registracija računa
Opis	Gost ima mogućnost registracije vlastitog računa
Vezani zahtjevi	/
Preduslovi	/
Posljedice – uspješan završetak	Kreira se račun
Posljedice – neuspješan završetak	Uneseni podaci nisu validni i registracija nije završena
Primarni akteri	Gost
Ostali akteri	/
Glavni tok	Gost bira mogućnost registracije. Prikazuje se forma sa ličnim podacima o korisniku. Gost upisuje podatke. Sistem validira podatke te ukoliko su validni, registruje novog korisnika.
Proširenja/alternative	/

Tabela 3.6: Registracija računa - scenarij

Korisnik	Sistem
1. Odabir opcije za registracijom	
	2. Prikaz forme za registraciju
3. Popunjavanje forme	
	4. Validacija unosa
	5. Dodavanje novog korisnika u bazu
	6. Poruka o uspješnoj registraciji

Tabela 3.7: Registracija računa - tok događaja, uspješan završetak

Korisnik	Sistem
1. Odabir opcije za registracijom	
	2. Prikaz forme za registraciju
3. Popunjavanje forme	
	4. Validacija unosa
	5. Poruka o neuspješnoj registraciji zbog nepravilnog unosa

Tabela 3.8: Registracija računa - tok događaja, neuspješan završetak

U tabeli 3.9 je prikazan scenarij za aktivnost kupovina artikala iz korpe dok se u tabelama 3.10 i 3.11 nalaze tokovi događaja za uspješni i neuspješni završetak.

Naziv	Kupovina artikala iz korpe
Opis	Korisnik ima mogućnost kupovine artikala iz korpe
Vezani zahtjevi	/
Preduslovi	Kreiran račun Prijavljen
Posljedice – uspješan završetak	Uspješna kupovina artikala iz korpe
Posljedice – neuspješan završetak	Neuspješna kupovina zbog nevalidnih podataka o plaćanju
Primarni akteri	Korisnik
Ostali akteri	/
Glavni tok	Korisnik bira opciju za preged svoje korpe. Izlistavaju se svi artikli koji se nalaze u korpi. Prikazuje se ukupna cijena svih artikala. Korisnik bira opciju za kupovinu. Popunjava formu sa podacima o plaćanju. Sistem validira podatke i bilježi formu.
Proširenja/alternative	/

Tabela 3.9: Kupovina artikala iz korpe - scenarij

Korisnik	Sistem
1. Odabir opcije za prikaz korpe	
	2. Prikaz korisničke korpe
3. Odabir opcije za kupovinu artikala iz korpe	
	4. Prikaz forme za upis podataka o plaćanju
5. Popunjavanje podataka o plaćanju	
	6. Validacija unesenih podataka
	7. Spašavanje kupovine
	7. Prikaz poruke o završenoj kupovini

Tabela 3.10: Kupovina artikala iz korpe - tok događaja, uspješan završetak

Korisnik	Sistem
1. Odabir opcije za prikaz korpe	
	2. Prikaz korisničke korpe
3. Odabir opcije za kupovinu artikala iz korpe	
	4. Prikaz forme za upis podataka o plaćanju
5. Popunjavanje podataka o plaćanju	
	6. Validacija unesenih podataka
	7. Prikaz poruke o neuspješnoj kupovini zbog nevalidnih podataka o plaćanju

Tabela 3.11: Kupovina artikala iz korpe - tok događaja, neuspješan završetak

Scenariji i tokovi događaja za ostale funkcionalnosti su dati u prilogu A.

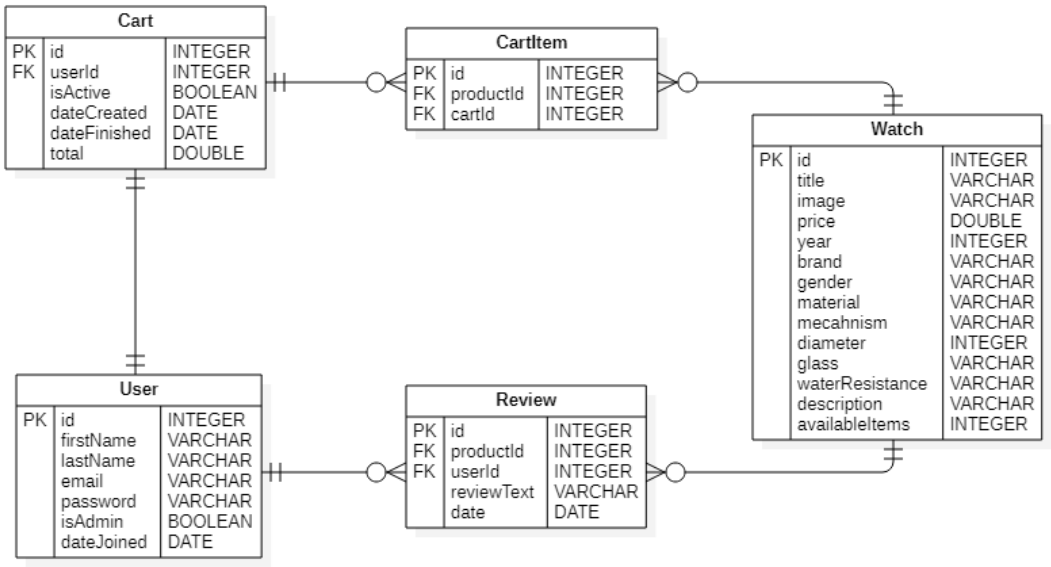
3.5 Nefunkcionalni zahtjevi

- Sistem mora biti otporan na zlonamjerne napade kao što su SQL Injection, Cross site scripting i Cross-Site Request Forgery
- Sistem mora vršiti provjeru svakog korisničkog unosa na zlonamjerni sadržaj.
- Sistem mora zaštiti korisničke informacije o plaćanju
- Sistem će dozvoliti korisniku pristup samo onim funkcionalnostima i podacima za koje ima privilegije
- Korisnici ne smiju imati pristup informacijama nečijeg korisničkog računa, niti ih smiju mijenjati.
- Korisnička lozinka mora sadržavati barem 8 karaktera
- Korisnička lozinka ne smije biti neka od često korištenih lozinki na Internetu
- Korisnička lozinka mora biti enkriptovana prije upisa u bazu podataka

Svi nefunkcionalni zahtjevi moraju biti ispunjeni. U narednim poglavljima će se obaviti osvrt na implementaciju nefunkcionalnih zahtjeva.

3.6 Dijagram baze podataka

Na slici 3.2 je prikazan dijagram baze podataka. Prikazane su tabele: User, Watch, Cart, CartItem i Review te veze između njih. Ove tabele predstavljaju modele klasa u aplikaciji.



Slika 3.2: ER dijagram za aplikaciju WatchMe

3.7 Okruženje za implementaciju

Odabrano okruženje za implementaciju ove aplikacije je Django. Django je besplatni open-source framework za razvoj web aplikacija, napisan u Pythonu. Njegov glavni cilj je olakšati programerima razvoj složenih aplikacija kroz naglasak na ponovnu upotrebu koda i komponenti, nisku povezanost među komponentama i brz razvoj. Nudi veliku kolekciju modula koje možemo koristi na različitim projektima.

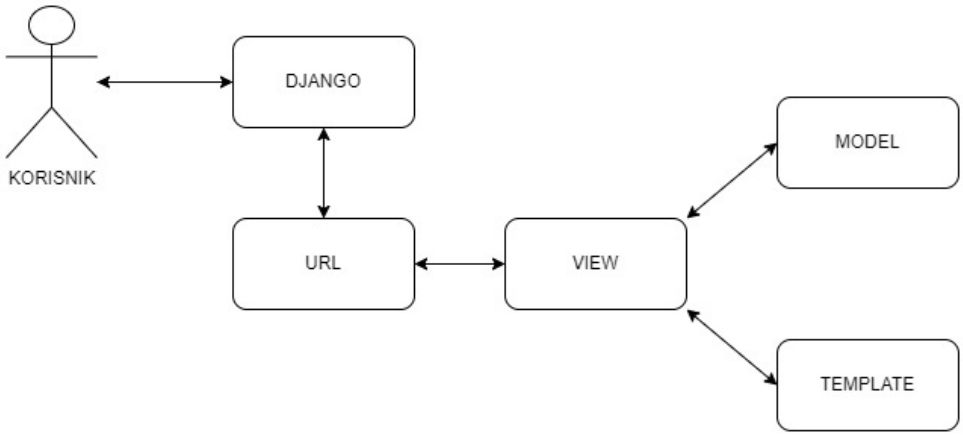
Karakterističan je po tome što je veoma jednostavan te da pomoću njega možemo razvijati aplikacije bilo koje kompleksnosti. Neke od poznatih stranica koje su izgrađne korištenjem Django frameworka su: Mozilla, Instagram, BitBucket i Spotify.

3.8 Arhitektura aplikacije

Jedna od najpoznatijih i najkorištenijih arhitektura je Model – View – Controller (MVC) arhitektura. MVC je slojevita arhitektura koja se u početku koristila za razvoj desktop aplikacija sa korisničkim interfejsom ali danas se koristi i za web i mobilne aplikacije. Osnovna ideja ove arhitekture jeste da odvoji korisnički interfejs od poslovne logike. [9]

Odabrana arhitekura za razvoj aplikacije WatchMe je dosta slična MVC modelu i naziva se Model – View – Template (MVT). Ovaj arhitekturalni patern je karakterističan za Django framework i u suštini predstavlja prilagođeni MVC model. U pitanju je slojevita arhitektura koja se sastoji od tri važne komponente:

- Model – sloj zadužen za strukturiranje i manipulaciju podacima web aplikacije. U suštini, pomoću njega pohranjujemo podatke te pristupamo i vršimo obradu podataka koji se nalaze u bazi podataka.
- View – sloj koji enkapsulira logiku odgovornu za obradu zahtjeva korisnika i vraća odgovor. To je korisnički interfejs za izvršavanje logike i interakciju sa modelima. Odgovoran je za prikaz svih ili dijela podataka korisniku.
- Template – prezentacijski sloj koji pruža sintaksu prilagođenu dizajnerima za prikazivanje informacija koje će biti predstavljene korisniku. Sastoji se od statičkog HTML koda kao i od određenih dinamičkih informacija koje View proslijeđuje prilikom odgovara na zahtjev. [9]



Slika 3.3: MVT patern

Korisnik šalje URL zahtjev za određenom stranicom. Nakon toga, Django traži odgovarajući pogled koji odgovara URL zahtjevu korisnika. Ukoliko je pronađen, View potom komunicira sa Modelom i Template-om kako bi napravio kompletan odgovor i predstavio ga nazad korisniku. Na slici 3.3. je ugrubo skicirano kako funkcioniše MVT patern.

Za potrebe aplikacije WatchMe, Model sloj će sadržavati implementaciju neophodnih klasa objekata (User, Watch, Cart, CartItem i Review), Template sloj HTML datoteke za prikaz korisničkog interfejsa dok će View sloj sadržavati funkcije koje služe za interakciju između pogleda i klasa.

3.9 Rezime

U ovom poglavlju je prikazana specifikacija aplikacije koja je implementirana u cilju demonstriranja implementacije sigurnosnih propusta. Predstavljeni su njeni korisnici, funkcionalni i nefunkcionalni zahtjevi. Objašnjena je arhitektura aplikacije i MTV patern, te je navedeno i okruženje za njen razvoj.

U sljedećem poglavlju će biti predstavljeni sigurnosni propusti i njihova implementacija kroz funkcionalnosti aplikacije.

Poglavlje 4

Analiza i implementacija najčešćih sigurnosnih propusta

U ovom poglavlju će se predstaviti tri česta sigurnosna propusta: *SQL Injection*, *Cross-site Scripting* (XSS) i *Cross-site request forgery* (CSRF). Navedeni propusti se nalaze na OWASP TOP 10 listi sigurnosnih propusta.

Propusti će biti tretirani kroz određene funkcionalnosti aplikacije WatchMe. Ove funkcionalnosti su definisane u prethodnom poglavlju. U tabeli 4.1 se nalaze propusti i funkcionalnosti koje su potencijalno ranjive na njih. Razlozi zašto su ranjive će biti opisani u nastavku ovog poglavlja.

Sigurnosni propust	Funkcionalnosti
SQL Injection	• Pregled artikala • Pretraga putem search bar-a • Ostavljanje recenzija • Pregled i izmjena korisničkih informacija
Cross Site Scripting (XSS)	• Ostavljanje recenzija • Izmjena korisničkih informacija
Cross-site Request Forgery (CSRF)	• Registracija/login • Izmjena korisničkih informacija • Ostavljanje recenzija • Kupovina artikala

Tabela 4.1: Pregled propusta i funkcionalnosti gdje se mogu javiti

4.1 SQL Injection

Napad SQL Injection sastoji se od umetanja ili "injekcije" SQL upita preko ulaznih podataka od strane klijenta najčešće koristeći korisnički interfejs. U pitanju je vrsta napada ubrizgavanjem,

u kojem se SQL naredbe ubrizgavaju u unos podataka kako bi se utjecacalo na izvršavanje unaprijed definiranih SQL naredbi.

Uspješan SQL Injection napad može čitati osjetljive podatke iz baze podataka, modificirati ih, izvršiti administrativne operacije na bazi podataka (kao što je gašenje *Data Base Management System-a*) i u nekim slučajevima dati komande operativnom sistemu. [10]

Glavne posljedice su:

1. Povjerljivost: Pošto SQL baze podataka općenito sadrže osjetljive podatke, gubitak povjerljivosti je čest problem.

2. Autentifikacija: Ako se koriste loše SQL komande za provjeru korisničkih imena i lozinki, možda će biti moguće povezati se na sistem kao drugi korisnik bez prethodnog znanja o lozinki.

3. Autorizacija: Ako se informacije o autorizaciji drže u SQL bazi podataka, moguće je promijeniti te informacije kroz uspješnu eksploataciju propusta.

4. Integritet: Baš kao što je moguće pročitati osjetljive informacije, moguće je napraviti promjene ili čak izbrisati ove informacije. [10]

Glavne dvije kategorije SQL Injection napada su: *SQL Manipulation* i *Code Injection*

- SQL Manipulation - podrazumjeva modifikaciju SQL naredbe kroz skup operacija (npr. UNION) ili izmjenom WHERE klauzule da vraća drugačiji rezultat. Mnogi dokumentovani SQL injection napadi su ovog tipa. Najpoznatiji napad je izmjena WHERE klauzule naredbe za autentifikaciju korisnika pa WHERE klauzula uvijek vraća rezultat TRUE

- Code Injection – je napad kada napadač ubacuje novu SQL naredbu ili komandu baze podataka u SQL naredbu. Klasični "Code Injection" napad je kada se doda SQL Server EXECUTE komanda u ranjivu SQL naredbu. "Code Injection" funkcioniše jedino kada su podržane višestruke SQL naredbe po jednom zahtjevu baze podataka.

U slučaju e-commerce aplikacija, ukoliko stranica nije dobro zaštićena, posljedice mogu biti katastrofalne. Napadač je u stanju pristupiti svim podacima iz baze podataka i može ih mijenjati. Recimo, može mijenjati atribute produkata ili usluga kao što su cijena ili dostupnost. Također, napadač može brisati produkte iz baze podataka - jedan, više ili čak sve produkte čime bi stranicu učinio praktički neupotrebljivom. Osim produkata i usluga, može mijenjati i brisati korisničke informacije i pristupati podacima kojima samo administrator ima pristup kao što su završene transkacije.

Najveći SQL Injection napad do danas bio je na *Heartland Payment Systems* 2008. godine. Korišten je za pristup sistemima za obradu kreditnih kartica. Napad je počeo u martu 2008. godine, ali je otkriven tek u januaru 2009. godine. Ukradeno je oko 130 miliona brojeva kreditnih kartica te se ovaj slučaj smatra najvećom krađom identiteta u američkoj historiji. Kompanija nije mogla poslovati 5 mjeseci nakon toga i kažnjena je sa 145 miliona dolara. [11]

4.1.1 Implementacija zaštite

Prije svega će biti opisano kako izvršiti SQL injection napad kroz jedan jednostavan primjer. Recimo da na korisničkom interfejsu imamo polje za unos gdje je potrebno unijeti ID korisnika (broj) i da je unesen broj 105. Nakon unosa aplikacija će izvršiti SQL naredbu prikazanu u isječku koda 4.1:

```
1 SELECT * FROM Users WHERE UserId = 105;
```

Isječak koda 4.1: Primjer jednostavne SQL naredbne

Međutim, ukoliko je korisnik zlonamjeran i želi pristupiti, mijenjati ili brisati podatke iz baze podataka, u polje za unos će napisati "105 OR 1=1", čime će aplikacija izvršiti upit prikazan u isječku koda 4.2:

```
1 SELECT * FROM Users WHERE UserId = 105 OR 1=1;
```

Isječak koda 4.2: Primjer jednostavnog SQL Injection napada

Nakon čega će server vratiti listu svih korisnika. Razlog ovoga je što je dodan uslov "1=1" koji je uvijek tačan.

Django, kao i ostala moderna razvojna okruženja, nudi kvalitetnu zaštitu od SQL injection napada. Django-ovi upiti su zaštićeni od SQL Injection jer su konstruirani korištenjem parametrizacije upita. Parametrizovani SQL upiti omogućavaju postavljanje parametara u SQL upit umjesto konstantnih vrijednosti. Parametar uzima vrijednost samo kada se upit izvrši, što omogućava da se upit ponovo koristi s različitim vrijednostima i za različite svrhe. SQL kod se definira odvojeno od parametara upita. Budući da parametri mogu biti dostavljeni od strane korisnika i stoga nesigurni, osnovni upravljački program baze podataka ih izbjegava.

Django nudi *Object-relational mapping* (ORM) pomoću kojeg možemo pisati intuitivne i sigurne upite. ORM je tehnika koja omogućuje programerima da koriste podatke iz baze kao objekte, tj. da im pristupaju kao objektima. Django QuerySet predstavlja kolekciju objekata iz baze podataka koji se mogu filtrirati po raznim filterima.

U nastavku će kroz određene funkcionalnosti aplikacije WatchMe biti objašnjeno kako ORM radi i koje su mu prednosti.

Pretraga artikala

Pretraga artikala koristeći search bar je jedna od funkcionalnosti koja je ranjiva na određene propuste uključujući i SQL injection. Razlog je zato što korisnik može unijeti bilo kakvu informaciju u search bar, koju će sistem kasnije koristiti za dohvaćanje i obradu podataka. Zbog potencijalnih zlonamjernih korisnika, potrebno je obratiti dosta pažnje pri implementaciji ove funkcionalnosti.

Na sreću, ovdje nam je Django od velike pomoći. Django ORM nam omogućava da se ovakvi napadi izbjegnu. U isječku koda 4.3 je predstavljeno kako možemo koristeći Django ORM dobiti listu svih artikala odnosno satova iz tabele Watch. Ovi artikli će biti spašeni u varijablu *products* koja je tipa QuerySet.

```
1 products = Watch.objects.all()
```

Isječak koda 4.3: Dohvaćanje svih artikala iz baze podataka

Naravno, da bi ovo bilo moguće potrebno je kreirati model Watch, kao i ostale modele koju su navedeni u poglavlju 3 u dijagramu baze podataka. U isječku koda 4.4 je prikazano kako izgleda implementacija modela Watch.

```
class Watch(models.Model):
    title = models.CharField(max_length=30)
    image = models.ImageField(upload_to='photos/watches', blank=True
        )
    price = models.DecimalField(decimal_places=2, max_digits=12)
    year = models.IntegerField(validators=[MaxValueValidator(2022)])
    gender = models.CharField(max_length=30, choices=GENDER_CHOICES)
    brand = models.CharField(max_length=30, choices=BRAND_CHOICES)
    material = models.CharField(max_length=30, choices=
        MATERIAL_CHOICES)
    mechanism = models.CharField(max_length=30, choices=
        MECHANISM_CHOICES)
    diameter = models.IntegerField(validators=[MinValueValidator(24)
        , MaxValueValidator(56)])
    glass = models.CharField(max_length=30, choices=GLASS_CHOICES)
    waterResistantType = models.IntegerField(choices=
        WATER_RESISTANCE_CHOICES)
    description = models.TextField(max_length=255, blank=True)
    availableItems = models.IntegerField(validators=[
        MinValueValidator(0), MaxValueValidator(25)])
```

Isječak koda 4.4: Implementacija modela Watch

Koristeći Django ORM možemo dobiti i listu artikala odnosno satova koji na neki način zadovoljavaju korisnički unos u search bar-u. Za potrebe ovog rada, pretraga se vrši po nazivu sata odnosno atributu *title*. Aplikacija čita korisnički unos i spašava ga u varijablu *titleSearch* te nekaon toga pomoću funkcije filter pronalazi sve artikle u bazi podataka koji u svom nazivu sadrže ono što je korisnik unio. Ovo je prikazano u isječku koda 4.5:

```
titleSearch = request.POST['search']
products = Watch.objects.filter(title__icontains=titleSearch)
```

Isječak koda 4.5: Dohvaćanje artikala iz baze podataka po specifičnom imenu

Na ovaj način, funkcija filter uz pomoć parametrizacije onemogućava zlonamjernim korisnicima da izvrše napade umetanjem SQL koda. Ukoliko bi neko pokušao izvršiti napad sličan kao u isječku koda 4.2, tako što će u search bar upisati "Rolex OR 1=1", Django ORM će ovo pretvoriti u SQL upit koji je prikazan u isječku koda 4.6:

```
SELECT * FROM Watch WHERE title LIKE '%Rolex␣OR␣1=1%';
```

Isječak koda 4.6: Primjer neuspješnog pokušaja SQL injection napada

Odnosno, pretraživati će artikle koji u svom naslovu sadrže string "Rolex OR 1=1".

Izmjena korisničkih informacija

Kada se radi o izmjeni korisničkih informacija kao jednoj od funkcionalnosti aplikacije mora se biti posebno pažljiv jer se mijenja stanje u bazi podataka. Ali i ovdje uz pomoć Django ORM-a se lahko zaobilaze potencijalni propusti.
Prvo se uzima trenutni korisnik aplikacije i pretražuje se baza podataka koristeći funkciju get kako bi se dobio objekat koji predstavlja korisnika. Zatim se čitaju podaci sa polja za unos i spašavaju u određene varijable. Nakon toga vrši se kratka provjera na autentičnost nove e-mail adrese tj. da li već postoji neki korisnik sa tom e-mail adresom. Ukoliko postoji, ispisuje se greška i korisnik se vraća da ponovo unese infromacije. Ako je sve uredu, ažuriraju se nove

informacije nad objektom i spašavaju uz pomoć funkcije save. Na ovaj način, SQL injection napadi su nemogući. U isječku koda 4.7 prikazana je implementacija ovog procesa:

```
current_user = request.user
userDb = User.objects.get(username=current_user.username)
first_name = request.POST.get('first_name', current_user.first_name)
last_name = request.POST.get('last_name', current_user.last_name)
email = request.POST.get('email', current_user.email)
if current_user.email != email:
    if User.objects.filter(email=email).exists():
        messages.error(request, 'That email is being used!')
        return redirect('edit_profile')
userDb.first_name = first_name
userDb.last_name = last_name
userDb.email = email
userDb.save()
```

Isječak koda 4.7: Implementacija izmjena korisničkih informacija

Alternativni način rada sa bazom podataka

Važno je napomenuti da ovo nije jedini način rada sa bazom podataka u Django-u. Django osim ORM-a, nudi i opciju pisanja "sirovih" (eng. raw) SQL upita. Ovakav način se ne preporučuje, jer se onda mora paziti na potencijalne napade za koje se ORM sam brine. U isječku koda 4.8 prikazano je korištenje "sirovih" upita na primjeru pretrage artikala:

```
titleSearch = request.POST['search']
with connection.cursor() as cursor:
    cursor.execute("SELECT * FROM Watch WHERE title LIKE %s", [
       titleSearch])
    row = cursor.fetchone()
return row
```

Isječak koda 4.8: Primjer implementacije raw SQL upita

4.2 Cross-site scripting (XSS)

XSS napadi omogućavaju napadaču da ubaci skripte u pretraživače drugih korisnika. To se obično postiže pohranjivanjem zlonamjernih skripti u bazu podataka gdje će biti preuzete i prikazane drugim korisnicima, ili navođenjem korisnika da kliknu na link koji će uzrokovati izvršavanje JavaScript-a napadača. [10]

Međutim, XSS napadi mogu poticati iz bilo kojeg nepouzdanog izvora podataka, kao što su kolačići ili web servisi, kad god podaci nisu dovoljno filtrirani prije nego što se prikažu na stranici.

Glavne posljedice XSS napada su:

1. Krađa osjetljivih informacija
2. Krađa identiteta
3. Pokretanje zlonamjernog koda

4. Instaliranje virusa [10]

Postoje dvije kategorije XSS napada:

- Reflektovani (eng. reflected) XSS - napadi kod kojih se umetnuta skripta reflektuje sa web servera, kao što je poruka o grešci, rezultat pretrage ili bilo koji drugi odgovor koji uključuje dio ili sav unos poslat serveru. Reflektovani napadi se dostavljaju žrtvama putem druge rute, na primjer putem e-mail poruke ili na nekoj drugoj web lokaciji. Kada je korisnik prevaren da klikne na zlonamjernu vezu ili čak samo pregleda zlonamjernu stranicu, ubrizgani kod putuje na ranjivu web stranicu, što reflektuje napad nazad na korisnikov pretraživač. Pretraživač tada izvršava kod jer je došao sa "pouzdanog" servera. Reflektovani XSS se ponekad naziva i netrajnim ili XSS tipa 2.
- Pohranjeni (eng. stored) XSS - napadi kod kojih je umetnuta skripta trajno pohranjena na ciljnim serverima, kao što su baze podataka ili u poljima za komentare. Žrtva zatim preuzima zlonamjernu skriptu sa servera. Pohranjeni XSS se također ponekad naziva i trajnim ili XSS tipa 1.

Jedan od najpoznatijih XSS napada desio se 2005. godine. Poznat je pod nazivom *Samy*. Proširio se putem društvene mreže *MySpace*. U roku od 20 sati, preko milion korisnika je postalo žrtva ranjivosti, čineći Samy-a najbrže širećim virusom svih vremena. [10]

4.2.1 Implementacija zaštite

Ostavljanje recenzija

Ostavljanje recenzija za određeni artikal je odličan primjer funkcionalnosti koja lahko može biti ranjiva na XSS napad. Nakon što je kupio artikal, korisnik može ostaviti recenziju koja se spašava u bazu podataka. Ta recenzija će se prikazivati svima koji otvore detaljan prikaz tog artikla. Zlonamjerni korisnik može umjesto recenzije napisati HTML ili JavaScript kod koji će se spasiti u bazu i izvršiti kada drugi korisnik otvori taj artikal.

Django ovdje pruža kvalitetnu zaštitu. Posjeduje ugrađene sigurnosne protokole protiv XSS napada te pretpostavlja da su svi unosi podataka nesigurni osim ako nije naglašeno drugačije. Važno je reći da ovu zaštitu imaju samo pogledi kreirani koristeći Django Templates koji su opisani u prethodnom poglavlju.

Ukoliko pokušamo u polje za ostavljanje recenzija upisati kod prikazan u isječku 4.9:

```
1 <img src="https://media.giphy.com/media/I3eVhMpz8hns4/giphy.gif"/>
```

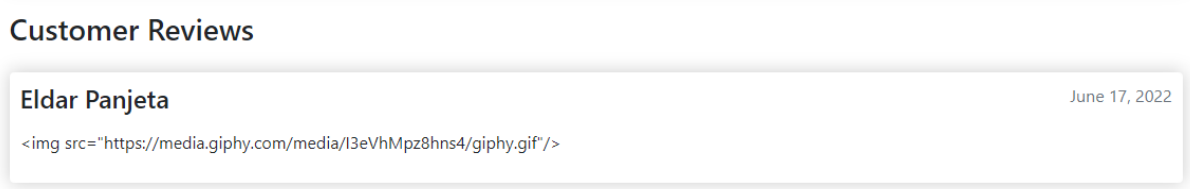
Isječak koda 4.9: Pokušaj XSS napada

Django će automatski izbjeći potencijalno opasni unos tako što će nad unesenim stringom pozvati funkciju html.escape(). Ova funkcija je zadužena da:

- Zamjeni svaki karakter **&** sa **&**
- Zamjeni svaki karatker **<** sa **<**
- Zamjeni svaki karatker **>** sa **>**
- Zamjeni svaki karatker **"** sa **\"**

- Zamjeni svaki karatker ’ sa \’

Nakon ovoga, zlonamjerni unos se neće tretirati kao HTML ili JavaScript kod koji se treba izvršiti, već kao obični sadržaj na stranici i neće promijeniti njenu strukturu. Na slici 4.1 je prikazano kako izgleda ostavljena recenzija sa XSS zaštitom.



Slika 4.1: Prikaz pokušaja XSS napada

Situacije kada Django XSS zaštita nije dovoljna

Postoje situacije kada Django ne može pomoći pri zaštiti od XSS napada. Kao što je već rečeno, ova zaštita je aktivna samo kada se koriste Templates. Ukoliko aplikacija koristi neki JavaScript framework kao što je React ili Angular onda Django zaštita neće biti od pomoći.

Također, programer ima mogućnost da samostalno isključi zaštitu. Ovo se obično nikada ne radi osim u sitacijama kada se razvija aplikacija za potrebe malog broja ljudi ili neku organizaciju, odnosno kada je apsolutno sigurno da korisnički unos neće biti zlonamjeran.

Ostale situacije kada Django zaštita nije dovoljna:

- Ako stranica koristi neke posebne ili prilagođene HTML atribute
- Ako stranica koristi HTML atribute koje prihvataju JavaScript kod

Najbolja praksa za zaštitu od XSS napada jeste da nikad ne treba vjerovati korisničkom unosu te da treba biti izuzetno pažljiv kada se HTML kod spašava u bazu podataka pogotovo ukoliko se taj kod izvršava i prikazuje.

4.3 Cross-site request forgery (CSRF)

Cross-site request forgery (CSRF) se može prevesti kao napad falsfikovanim zahtjevima. CSRF je napad koji prisiljava krajnjeg korisnika da izvrši neželjene radnje u web aplikaciji u kojoj je trenutno prijavljen. Napadači se uglavnom služe društvenin inženjeringom koje predstavlja psihološku manipulaciju nad ljudima u cilju izvođenju zlonamjernih radnji ili odavanju povjerljivih informacija. Koristeći razne tehnike, kao što je slanje veze putem e-mail, napadač može lahko prevariti korisnika i natjerati ga da izvrši određene akcije.

Ako je žrtva običan korisnik, uspješan CSRF napad može natjerati korisnika da izvrši zahtjeve za promjenom stanja kao što su prijenos sredstava, promjena e-mail adrese i slično. Ako je žrtva administrator na stranici, CSRF može ugroziti cijelu web aplikaciju.

Da bi napad imao učinka koriste se HTTP zahtjevi koji izazivaju određene promjene podataka i sadržaja. Tipični zahtjevi su GET i POST. GET metoda koristi se za dohvat podataka

i prenosi se putem URL zahtjeva. Kod POST metode podaci se šalju u tijelu poruke. Ona se koristi za različite transakcije i izmjene podataka. [10]

Glavne posljedice CSRF napada koje napadač može učiniti u ime drugog korisnika su:

1. Objavljivanje sadržaja na stranici i slanje poruka drugim korisnicima
2. Transfer novčanih sredstava ili online kupovina
3. Promjena e-mail adrese ili drugih korisničkih informacija
4. Promjena lozinke

CSRF napadi su mogu izvoditi na različite načine:

- Napadi HTML objektima - CSRF napadi se mogu izvesti na više načina pomoću HTTP GET metode, što je klasičan primjer CSRF napada korištenjem HTML objekata. Primjer takvog objekta je IMG koji se koristi za oblikovanje slikovnog web sadržaja. IMG objekat sadrži atribut "src" u kojem je pohranjen izvor s kojeg se preuzima slika. Podmetanjem zlonamjernog koda u taj atribut moguće je poslati zahtjev drugom web odredištu.
- Napadi skriptnim kodom - Posebno su poznate ranjivosti JavaScript programskog koda. Jednostavan primjer zloupotrebe je pomoću objekta Image(). Ukoliko je omogućeno izvođenje skriptnog koda u web pregledniku on se pokreće automatski prilikom učitavanja stranice.
- Napadi XMLHttpRequest objektima - XMLHttpRequest objekti su osnovni objekti AJAX programa. AJAX (eng. Asynchronous Javascript And XML) je skupina tehnologija koja omogućuje slanje i obradu HTTP zahtjeva u pozadinskom radu aplikacije. HTTP zahtjevi se automatski kreirairu i podložni su CSRF napadima. [12]

CSRF napad je poznat još od 90-tih godina prošlog stoljeća. Najpoznatiji slučajevi CSRF napada su oni koji su zahvatili vrlo velik broj korisnika. Tako je u napadu na web stranicu *Auction.co.kr*, korejsko sjedište organizacije *eBay*, 18 miliona ljudi izgubilo svoje lične podatke. [12]

4.3.1 Implementacija zaštite

E-commerce stranice su česta meta za CSRF napade jer su orijentisane na transakciju novca i online kupovinu. Aplikacija WatchMe ima nekoliko funkcionalnosti koje bez adekvatne zaštite mogu postati ranjive na CSRF napade. Neke od tih funkcionalnosti su: login, izmjena korisničkih informacija, ostavljanje recenzija, kupovina artikala iz korpe i admin funkcionalnosti. U ovom radu će se obraditi zaštita na primjeru kupovine artikala. Način zaštite za ostale funkcionalnosti je identičan.

Kupovina artikala iz korpe

Nakon što je korisnik odabrao opciju kupovine, aplikacija ga vodi na popunjavanje obrasca za dostavu i plaćanje. Korisnik popunjava obrazac sa informacijama koje uključuju i broj kreditne kartice. Bez dobre zaštite, napadač može izvršiti kupovinu u ime drugog korisnika.

Baš kao i za prethodna dva propusta, i ovdje Django nudi kvalitetnu zaštitu. Ta zaštita je bazirana na korištenju CSRF tokena. CSRF token je poput alfanumeričkog koda ili nasumične tajne vrijednosti koja je svojstvena određenoj stranici. Dakle, nijedna druga stranica nema isti kod.

Server ima sopstveni CSRF token te ga on šalje korisniku zajedno sa obrascen. Kada se serveru vrati popunjen obrazac sa njim dolazi i token. Server zatim provjerava da li token odgovara onom koji server prepoznaje. Ukoliko ne odgovara, taj obrazac neće biti prihvaćen.

Da bi koristili ovu zaštitu, potrebno je dodati { % csrf_token % } na svako mjesto u HTML kodu gdje se koristi objekat FORM, odnosno gdje se god šalje HTTP POST zahtjev. U isječku koda 4.10 prikazan je dio obrasca koji šalje informacije o dostavi i plaćanju na server:

```
<form action="{% url 'checkout' %}" method="POST">
    {% csrf_token %}
    {% include 'partials/alerts.html' %}
        <div class="row">
            <div class="form-group col-sm-6">
                <label>City</label>
                        <select name="city" class="form-control">
                            <option value="">Sarajevo</option>
                            <option value="">Tuzla</option>
                            <option value="">Zenica</option>
                            <option value="">Mostar</option>
                        </select>
            </div>
```

Isječak koda 4.10: Dio obrasca koji šalje informacije o dostavi i plaćanju

4.4 Rezime

U ovom poglavlju predstavljeni su propusti: SQL Injection, Cross-site Scripting (XSS) i Cross-site request forgery (CSRF). Dat je kratak opis svakog, njihove posjedice, klasifikacija i kada se pojavljuju. Zatim je objašnjeno kako se zaštiti od potencijalnih napada na primjeru web aplikacije WatchMe.

U sljedećom poglavlju biće predstavljeni određeni identifikacijski i autentifikacijski propusti koji se često pojavljuju te kako se zaštiti od njih.

Poglavlje 5

Analiza i implementacija identifikacijskih i autentifikacijskih propusta

Greške u identifikaciji i autentifikaciji su grupa propusta koje se odnose na korisnički identitet, autentifikaciju ili upravljanje sesijom. Identifikacija je sposobnost sistema ili aplikacije da otkrije identitet korisnika, dok autentifikacija predstavlja mogućnost da se dokaže da je korisnik zaista ono što tvrdi da jeste.

Napadači mogu iskoristiti greške u identifikaciji i autentifikaciji i ukrasti lozinke, ključeve, tokene sesije ili iskoristiti druge propuste u implementaciji da bi preuzeli identitete drugih korisnika, bilo privremeno ili trajno. [10]

U ovom poglavlju, biti će predstavljeni mogući napadi koji se mogu desiti zbog loše zaštite lozinke, kontrole pristupa i sesija, te implemetacija zaštite u Python okruženju.

5.1 Lozinka

Lozinka je niz znakova koji se koristi za provjeru identiteta korisnika tokom procesa autentifikacije. Trebale bi biti poznate samo korisniku koji ih koristi za pristup uređaju, aplikaciji ili web stranici. Nužno je da lozinka bude dobro zaštićena.

Opšti propusti

Poznato je da skoro svaka web stranica ima određena ograničenja pri registraciji, odnosno odabiru lozinke za račun. Različite organizacije imaju drugačija shvatanja o tome koja to ograničenja trebaju biti. OWASP je definisao sljedeća pravila:

- Lozinku ne bi trebalo skraćivati
- Lozinka treba sadržavati najmanje 10 karaktera
- Lozinka u sebi treba da sadrži barem jedno veliko, jedno malo slovo te jedan broj
- Lozinka u sebi ne bi trebala sadržavati lične korisničke informacije kao što su ime, prezime ili e-mail.

Također, neophodno je onemogućiti korisnicima da prilikom registracije odaberu neku od često korištenih lozinki. [10]

Čuvanje korisničkih lozinki je bitna komponenta za svaku web aplikaciju. Pri pohrani lozinke, nepohodno je da bude zaštićena na takav način da ukoliko su podaci baze podataka ugroženi, korisničke lozinke ne budu otkrivene. Najefektivniji način da bi se to osiguralo je *heširanje* lozinke.

Heširanje je kriptografski proces koji se može koristiti za provjeru autentičnosti i integriteta različitih vrsta unosa. Koristi se u sistemima za autentifikaciju kako bi se izbjeglo pohranjivanje lozinki u obliku izvornog teksta, ali se također koristi za provjeru valjanosti datoteka, dokumenata i drugih vrsta podataka. [10]

Pri autentifikaciji, kada korisnici kreiraju novi račun i unesu lozinku koju su odabrali, aplikacija tu lozinku prosljeđuje kroz funkciju heširanja i pohranjuje rezultat u bazu podataka. Kada se korisnik kasnije želi autentifikovati, proces se ponavlja i rezultat se uspoređuje s vrijednošću iz baze podataka. Ako se podudara, korisnik je dao ispravnu lozinku.

Brute-force napadi

Brute-force napad se može prevesti kao napad grubom silom. On predstavlja pokušaj otkrivanja lozinke sistematskim isprobavanjem svake moguće kombinacije slova, brojeva i simbola dok se ne otkrije ona ispravna kombinacija koja radi. Svaka web aplikacija koja zahtjeva autentifikaciju korisnika može biti meta za ovu vrstu napada. [10]

Napadač uvijek može otkriti lozinku na ovaj način ali to može potrajati dugo vremena. Ovisno o dužini i složenosti lozinke mogu postojati milijarde mogućih kombinacija. Da bi se stvari malo ubrzale, napadač može koristiti riječi iz rječnika ili modificirane riječi jer će većina ljudi koristiti njih umjesto potpuno nasumične lozinke.

Brute-force napadi dovode korisničke račune u opasnost i preplavljuju web aplikaciju nepotrebnim prometom.

5.1.1 Implementacija zaštite

Implementacija ograničenja na lozinku je relativno lagana. Pri registraciji, potrebno je provjeriti da li odabrana korisnička lozinka zadovoljava kriterije i pravila opisana u ovom radu. U isječku koda 5.1 prikazana je funkcija *weakPassword* koja provjerava dužinu lozinke, te da li ona ima barem jedno veliko, jedno malo slovo i jedan broj. Ukoliko ova ograničenja nisu zadovoljena, funkcija vraća *True*, u suprotnom vraća *False*. Ova funkcija koristi pomoćne funkcije *containsNumber* i *containsLetter*.

```
1  def weakPassword(password):
2      if len(password) < 10:
3          return True
4      if password.islower():
5          return True
6      if password.isupper():
7          return True
8      if not containsLetter(password):
9          return True
10     if not containsNumber(password):
11         return True
12     return False
```

Isječak koda 5.1: Implementacija funkcije weakPassword

Još jedno pravilo jeste da lozinka u sebi ne bi trebala sadržavati korisničke informacije. U isječku koda 5.2 je prikazana funkcija *badPassword* koja ispituje da li odabrana lozinka u sebi sadrži ime, prezime ili username koji predstavlja dio unesene e-mail adrese (lijevo od znaka @).

```
def badPassword(firstName, lastName, username, password):
    if password != None and firstName.lower() in password.lower():
        return True
    if password != None and lastName.lower() in password.lower():
        return True
    if password != None and username.lower() in password.lower():
        return True
    return False
```

Isječak koda 5.2: Implementacija funkcije badPassword

Za potrebe testiranja da li je odabrana lozinka neka od često korištenih na internetu, implementirana je funkcija *commonPassword* čija je implementacija prikazana u isječku koda 5.3. Ova funkcija učitava statičku tekstualnu datoteku pod nazivom *commonpasswords.txt* koja u sebi sadrži 1000 najčešće korištenih lozinki i provjerava da li je odabrana lozinka jedna od njih. Ukoliko jeste, funkcija vraća *True*, dok u suprotnom vraća *False*.

```
def commonPassword(password):
    file = open('accounts/commonpasswords.txt', 'r')
    common_values = [line.rstrip("\n") for line in file]
    if any(value == password for value in common_values):
        return True
    return False
```

Isječak koda 5.3: Implementacija funkcije commonPassword

Nakon što su izvršene sve neophodne provjere, sistem spašava informacije o novom korisniku u bazu podataka. Ovaj proces podrazumijeva i spašavanje lozinke a prije toga njeno heširanje. Tu nam Django značajno pomaže jer on to radi sam. Django za ove potrebe koristi PBKDF2 algoritam sa SHA256 hešom. PBKDF2 je trenutno jedan od vodećih algoritama za heširanje i nema poznatih sigurnosnih problema. [9]

U isječku koda 5.4 prikazano je kako kreirati novog korisnika i spasiti ga u bazu podataka. Heširanje se dešava automatski.

```
user = User.objects.create_user(username=username, password=password
    ,email=email,first_name=first_name,last_name=last_name)
user.save()
```

Isječak koda 5.4: Spašavanje novog računa u bazu podataka

Implementacijom ovih ograničenja i heširanja kreirana je kvalitetna zaštita i za brute-force napade. Sada će napadaču trebati mnogo više pokušaja da pogodi tačnu lozinku. Međutim, postoji još jedan način da se zaštiti od brute-force napada.

Ideja je da se prati broj uzastopnih neuspjelih pokušaja prijave odnosno login-a za određenog korisnika. Nakon određenog broja neuspjelih pokušaja koji dolaze sa iste IP adrese, korisniku postoje onemogućeno da se prijavi sa te IP adrese na nekoliko minuta.

Za implementaciju ovoga korišten je paket *django-axes*. Nakon instalacije, ovaj paket kreira novu tabelu u bazi podataka koja služi za praćenje neuspjelih pokušaja login-a. Ukoliko broj pokušaja pređe 3, korisnik se neko vrijeme ne može prijaviti, odnosno postaje zaključan. Da bi definisali vrijeme trajanja ove blokade, potrebno je da u *settings.py* datoteci dodamo kod prikazan u isječku koda 5.5:

```
AXES_COOLOFF_TIME = timedelta(minutes = 1)
```

Isječak koda 5.5: Podešavanje trajanja vremena blokade nakon 3 neuspjela pokušaja prijave

U ovom slučaju, vrijeme blokade traje jednu minutu.

5.2 Kontrola pristupa

Kontrola pristupa je sigurnosna tehnika koja reguliše ko ili šta može da vidi ili koristi resurse u računarskom okruženju. U suštini, korisnik mora imati određene permisije odnosno dozvole, da bi pristupio nekoj stranici ili podacima.

Prema najnovijoj OWASP Top 10 listi, neispravna kontrola pristupa predstavlja najčešći sigurnosni propust mnogih web aplikacija. Greške dovode do neovlaštenog otkrivanja informacija, modifikacije ili brisanja svih podataka ili obavljanja poslovne funkcije za koju korisnik nema dozvolu. Napadač može ukrasti korisničke podatke, mijenjati ih te izvesti akcije u ime korisnika. Postoje dvije kategorije neispravne kontrole pristupa:

1. Horiznotalne kontrole pristupa - javljaju se kada napadač može izvršiti radnju ili pristupiti podacima drugog korisnika koji ima iste permisije kao napadač

2. Vertikalne kontrole pristupa - javljaju se kada napadač može izvršiti radnju ili pristupiti podacima koji zahtijevaju veće permisije [10]

Klasičan primjer napada ove vrste je kada zlonamjerni korisnik promjeni parametar unutar URL-a koji ukazuje na ID korisnika. Na taj način može pristupiti njegovim informacijama i mijenjati ih.

5.2.1 Implementacija zaštite

Aplikacija WatchMe ima brojne funkcionalnosti koje moraju imati kontrolu pristupa. Neke od tih funkcionalnosti su: pregled i izmjena ličnih podataka, pregled i brisanje artikala iz korpe i administratorske funkcionalnosti.

U slučaju WatchMe aplikacije, postoje tri moguća scenarija neispravne kontrole pristupa: kada gost pokušava koristiti funkcionalnosti prijavljenog korisnika, kada prijavljen korisnik pokušava koristiti funkcionalnosti drugog korisnika i kada gost ili korisnik pokušava koristiti funkcionalnosti administratora.

Zahvaljujući Django-u, implementacija kontrole pristupa je jako jednostavna. Da bi zabranili da gost koristi funkcionalnosti prijavljenog korisnika, dovoljno je koristiti ugrađenu funkciju *is_authenticated*. Ova funkcija kada se pozove nad varijablom tipa *User*, vraća *True* ako je korisnik prijavljen dok u suprotnom vraća *False*. U isječku koda 5.6 prikazana je implementacija funkcije koja se poziva na zahtjev prikaza korisničkih informacija. Iz zahtjeva se čita korisnik i provjerava se da li je on autentifikovan, odnosno prijavljen. Ako jeste, funkcija prikazuje njegove informacije. Ako nije, funkcija ga obavještava da se mora prijaviti.

```
def myProfile_view(request):
    current_user = request.user
    if current_user.is_authenticated:
        context = {'user': current_user}
        return render(request, 'accounts/myprofile.html', context)
    else:
        context = {'message': 'Please log in so you can access your
            personal information'}
        return render(request, 'pages/permission_denied.html',
            context)
```

Isječak koda 5.6: Implementacija funkcije za prikaz korisničkih informacija

S obzirom da se u URL-u zahtjeva ne navodi ID korisnika, jer za tim nema potrebe, jedan korisnik ne može pristupiti niti mijenjati informacije drugog korisnika. Ovo isto vrijedi i za prikaz korisničke korpe.

Django se sam brine za pristup admin funkcionalnostima. Ukoliko korisnik nije autorizovan za pristup tim funkcionalnostima, stranica ga obavještava da se mora prijaviti kao administrator. Ali ovo važi samo za one funkcionalnosti koje čitaju, mijenjaju, brišu ili dodaju nove podatke u bazu. Ukoliko aplikacija posjeduje admin funkcionalnost koja radi nešto drugo i treba biti skrivena od drugih korisnika onda će se za kontrolu pristupa koristiti ugrađena funkcija *is_staff* koja provjerava da li je korisnik administrator. Pri razvoju aplikacije WatchMe nije bilo potrebe za korištenjem ove funkcije.

5.3 Sesije

Web sesija predstavlja vrijeme koje korisnik provede pregledajući datu web stranicu. U praksi, sastoji se od podataka ili datoteka koji postoje tijekom korištenja web stranice ili web aplikacije. Ovi resursi su jedinstveno identificirani s ID-om sesije. Korisnički pretraživač dobija ovaj ID na početku nove sesije, i on se razmjenjuje tokom svake naredne komunikacije između pretraživača i servera. Sesije omogućavaju web stranici da ima neki oblik "kratkoročnog pamćenja" o aktivnostima korisnika. [13]

Nedovoljna sigurnost sesija može dovesti do toga da korisnički računi budu podložni neovlaštenom pristupu. Napadači koriste različite vrste sigurnosnih propusta kako bi ukrali ID korisničke sesije i izvršavali razne akcije u njegovo ime.

Najpoznatiji napad na sesije se naziva *Session hijacking* što bi u prevodu značilo otmica sesije. Otmica sesije je čin preuzimanja kontrole nad korisničkom sesijom nakon uspješnog dobivanja ili generiranja ID-a sesije autentifikacije. [13]

5.3.1 Implementacija zaštite

Django pruža punu podršku za sesije. On koristi kolačić koji sadrži poseban ID sesije da identifikuje svaki pretraživač i njegovu pridruženu sesiju sa stranicom. Kolačići su tekstualne datoteke sa malom količinom podataka. Podaci o sesiji se pohranjuju u bazi podataka, što je sigurnije nego pohranjivanje podataka u kolačić gdje su podložniji zlonamjernim korisnicima. Moguće je konfigurisati da Django te podatke pohranjuje na nekom drugom mjestu kao što su keš memorija, datoteke, "sigurni" kolačići ali zadana lokacija je dobra i relativno sigurna opcija.

Da bi sesije bile omogućene potrebno je dodati kod prikazan u isječku koda 5.7 u datoteku *settings.py*, u *Middleware*:

```
'django.contrib.sessions.middleware.SessionMiddleware'
```

Isječak koda 5.7: Omogućavanje korištenja Django sesija

Za potrebe aplikacije WatchMe, određeno je da jedna sesija traje 10 minuta. To znači da će korisnik biti automatski odjavljen sa aplikacije nakon 10 minuta neaktivnosti. Ovo se postiže tako što se doda kod prikazan u isječku koda 5.8 u *settings.py* datoteku:

```
SESSION_COOKIE_AGE = 600
SESSION_SAVE_EVERY_REQUEST = True
```

Isječak koda 5.8: Konfigurisanje sesije da ističe nakon određenog vremena neaktivnosti

5.4 Rezime

U ovom poglavlju predstavljeni su različiti identifikacijski i autentifikacijski propusti. Navedeni propusti se pojavljuju pri lošoj implementaciji zaštite korisničke lozinke, kontrole pristupa i sesija. Opisano je kako implementirati kvalitetnu zaštitu koristeći Python okruženje te kako nam Django pomaže.

Poglavlje 6

Zaključak

U ovom završnom radu detaljno su obrađeni najčešći sigurnosni propusti koji se javljaju u e-commerce aplikacijama. Svi propusti su detaljno objašnjeni, kada se dešavaju i zbog čega te koje posljedice nose sa sobom. Prikazano je kako implementirati odgovorajaću zaštitu na primjeru jedne e-commerce aplikacije koja je razvijena za potrebe ovog rada.

E-commerce aplikacije pohranjuju i obrađuju veliku količinu podataka. Ti podaci mogu biti od njenih korisnika ali i od samih "vlasnika" aplikacije, odnosno vlasnika te virtuelne prodavnice. Ukoliko bi dospjeli u pogrešne ruke, posljedice mogu biti katastrofalne. Zlonamjerni korisnici mogu iskoristite ove podatke na razne načine kako bi nanijeli štetu svakom njenom korisniku ali i samoj aplikaciji. Osim toga, korisnici mogu biti prisiljeni da izvrše određene radnje kao što je transkacija novca u njihovo ime. Dakle, možemo zaključiti da je adekvatna zaštita izuzetno bitna za svaki softver ali posebno za e-commerce aplikacije. Zaštiti se mora posvetiti ogromna pažnja u svakoj fazi razvoja softvera.

Također, možemo zaključiti da nam moderna okruženja za razvoj softvera nude dosta mogućnosti te olakšavaju posao implementacije zaštite. Određeni sigurnosni propusti su čak automatski tretirani od strane okruženja.

Postavlja se pitanje da li je aplikaciju WatchMe moguće dodatno zaštiti. Odgovor je naravno potvrdan. Ukoliko bi se ova aplikacija stavila u stvarnu upotrebu, posebna pažnja bi se trebala obratiti na sistem plaćanja. Korisničke informacije o plaćanju, kao što je broj kartice, moraju biti zaštićeni i ne bi se trebali čuvati u bazi podataka. Neophodno je osigurati da se transakcija obavlja preko sigurnih posrednika.

Također, dodatnu pažnju bi trebalo posvetiti i korisničkim unosima. Recimo, ukoliko korisnik u polju za recenziju ostavi vulgaran komentar, aplikacija bi ga trebala automatski cenzurisati.

Postoje i mnogi drugi sigurnosni propusti i nije lahko sve nabrojati. Ali, važno je istaći da je njihovo prepoznavanje u ranim fazama razvoja softvera od ključne važnosti. Ukoliko smo svjesni da je propust moguć kada analiziramo aplikaciju koju pravimo, pri implementaciji ćemo imati manje problema i njegovo otklanjanje neće zahtjevati previše resursa.

Literatura

[1] Bishop, M., Introduction to computer security. Addison-Wesley Boston, 2005.

[2] Chen, T. M., Robert, J.-M., "The evolution of viruses and worms", in Statistical methods in computer security. CRC press, 2004.

[3] Hoffman, A., Web Application Security: Exploitation and Countermeasures for Modern Web Applications. O'Reilly Media, 2020.

[4] Lundestad, C. V., Hommels, A., "Software vulnerability due to practical drift", Ethics and Information Technology, Vol. 9, No. 2, 2007, str. 89–100.

[5] Metalidou, E., Marinagi, C., Trivellas, P., Eberhagen, N., Skourlas, C., Giannakopoulos, G., "The human factor of information security: Unintentional damage perspective", Procedia-Social and Behavioral Sciences, Vol. 147, 2014, str. 424–428.

[6] Anderson, R., Security engineering: a guide to building dependable distributed systems. John Wiley & Sons, 2020.

[7] Caldwell, T., "Ethical hackers: putting on the white hat", Network Security, Vol. 2011, No. 7, 2011.

[8] Rose, S., Borchert, O., Mitchell, S., Connelly, S., "Zero trust architecture", National Institute of Standards and Technology, Tech. Rep., 2020.

[9] Django Software Foundation, "Django", dostupno na: https://djangoproject.com

[10] "The open web application security project", dostupno na: https://owasp.org

[11] Grbavac, S., "Eliminisanje sql injection napada–mehanizmi odbrane elimination of the sql injection attacks–defense mechanisms", 2011.

[12] Burns, J., "Cross site request forgery", An introduction to a common web application weakness, Information Security Partners, 2005.

[13] Calzavara, S., Focardi, R., Squarcina, M., Tempesta, M., "Surviving the web: A journey into web session security", ACM Computing Surveys (CSUR), Vol. 50, No. 1, 2017.

Prilozi

Prilog A

Scenariji i tokovi događaja

U ovom prilogu su tabelarno prikazani scenariji za sve nabrojane funkcionalnosti aplikacije WatchMe. Osim scenarija, prikazani su i tokovi događaja za uspješno i neuspješno obavljenu aktivnost.

A.1 Pregled artikala

Naziv	Pregled artikala
Opis	Korisnik i gost imaju mogućnost pregleda svih artikala u sistemu
Vezani zahtjevi	/
Preduslovi	/
Posljedice – uspješan završetak	Izlistavaju se svi dostupni artikli
Posljedice – neuspješan završetak	/
Primarni akteri	Korisnik i gost
Ostali akteri	/
Glavni tok	Korisnik ili gost mogu da vrše pregled jednog ili više artikala koji su dostupni. Kada zatraže pregled, svi artikli se izlistavaju u vidu liste.
Proširenja/alternative	Pretraga putem search bar-a Detaljan pregled jednog artikla Dodavanje artikla u korpu

Tabela A.1: Pregled artikala - scenarij

Korisnik	Sistem
1. Odabir opcije za pregled artikala	
	2. Prikaz svih dostupnih artikala

Tabela A.2: Pregled artikala - tok događaja, uspješan završetak

Korisnik	Sistem
1. Odabir opcije za pregled artikala uz upis nekih ključnih riječi u search bar	
	2. Provjera unosa u search bar
	3. Prikaz svih dostupnih artikala koji na neki način zadovoljavaju korisnički unos u search bar-u

Tabela A.3: Pregled artikala - tok događaja, uspješan (alternativni) završetak 1

Korisnik	Sistem
1. Odabir opcije za pregled artikala	
	2. Prikaz svih dostupnih artikala
3. Odabir opcije za detaljan prikaz konkretnog artikla	
	4. Prikaz detaljnih informacija o konkretnom artiklu

Tabela A.4: Pregled artikala - tok događaja, uspješan (alternativni) završetak 2

Korisnik	Sistem
1. Odabir opcije za pregled artikala	
	2. Prikaz svih dostupnih artikala
3. Odabir opcije za detaljan prikaz konkretnog artikla	
	4. Prikaz detaljnih informacija o konkretnom artiklu
5. Odabir opcije za dodavanje artikla u korisničku korpu	
	6. Ažuriranje stanja korisničke korpe sa novim artiklom

Tabela A.5: Pregled artikala - tok događaja, uspješan (alternativni) završetak 3

A.2 Pregled detalja korisničkog računa

Naziv	Pregled detalja korisničkog računa
Opis	Korisnik vrši pregled detalja svog korisničkog računa
Vezani zahtjevi	/
Preduslovi	Kreiranan račun Prijavljen
Posljedice – uspješan završetak	Uspješan pregled detalja korisničkog računa
Posljedice – neuspješan završetak	/
Primarni akteri	Korisnik
Ostali akteri	/
Glavni tok	Korisnik nakon prijave bira opciju za pregled korisničkog računa nakon čega se prikazuju detalji o njegovom korisničkom računu
Proširenja/alternative	Korisnik želi izmijeniti određene podatke

Tabela A.6: Pregled detalja korisničkog računa - scenarij

Korisnik	Sistem
1. Odabir opcije za prikaz detalja korisničkog računa	
	2. Prikaz detalja korisničkog računa

Tabela A.7: Pregled detalja korisničkog računa - tok događaja, uspješan završetak

Korisnik	Sistem
1. Odabir opcije za prikaz detalja korisničkog računa	
	2. Prikaz detalja korisničkog računa
3. Izmjena određenih korisničkih informacija	
	4. Validiranje unesenih podataka
	5. Spašavanje izmjena

Tabela A.8: Pregled detalja korisničkog računa - tok događaja, uspješan (alternativni) završetak

A.3 Pregled korpe

Naziv	Pregled korpe
Opis	Korisnik ima mogućnost pregleda artikala iz korpe
Vezani zahtjevi	/
Preduslovi	Kreiran račun Prijavljen
Posljedice – uspješan završetak	Uspješna pregled artikala iz korisničke korpe
Posljedice – neuspješan završetak	/
Primarni akteri	Korisnik
Ostali akteri	/
Glavni tok	Korisnik bira opciju za preged svoje korpe. Izlistavaju se svi artikli koji se nalaze u korpi. Prikazuje se ukupna cijena svih artikala.
Proširenja/alternative	Brisanje artikala iz korpe

Tabela A.9: Pregled artikala iz korpe - scenarij

Korisnik	Sistem
1. Odabir opcije za prikaz korisničke korpe	
	2. Prikaz korisničke korpe

Tabela A.10: Pregled korisničke korpe - tok događaja, uspješan završetak

Korisnik	Sistem
1. Odabir opcije za prikaz korisničke korpe	
	2. Prikaz korisničke korpe
3. Odabir opcije za brisanje artikla iz korpe	
	4. Uklanjanje artikla iz korpe
	5. Prikaz ažurirane korisničke korpe

Tabela A.11: Pregled korisničke korpe - tok događaja, uspješan (alternativni) završetak

A.4 Kupovina artikala iz korpe

Naziv	Kupovina artikala iz korpe
Opis	Korisnik ima mogućnost kupovine artikala iz korpe
Vezani zahtjevi	/
Preduslovi	Kreiran račun Prijavljen
Posljedice – uspješan završetak	Uspješna kupovina artikala iz korpe
Posljedice – neuspješan završetak	Neuspješna kupovina zbog nevalidnih podataka o plaćanju
Primarni akteri	Korisnik
Ostali akteri	/
Glavni tok	Korisnik bira opciju za preged svoje korpe. Izlistavaju se svi artikli koji se nalaze u korpi. Prikazuje se ukupna cijena svih artikala. Korisnik bira opciju za kupovinu. Popunjava formu sa podacima o plaćanju. Sistem validira podatke i bilježi formu.
Proširenja/alternative	/

Tabela A.12: Kupovina artikala iz korpe - scenarij

Korisnik	Sistem
1. Odabir opcije za prikaz korpe	
	2. Prikaz korisničke korpe
3. Odabir opcije za kupovinu artikala iz korpe	
	4. Prikaz forme za upis podataka o plaćanju
5. Popunjavanje podataka o plaćanju	
	6. Validacija unesenih podataka
	7. Spašavanje kupovine
	8. Prikaz poruke o završenoj kupovini

Tabela A.13: Kupovina artikala iz korpe - tok događaja, uspješan završetak

Korisnik	Sistem
1. Odabir opcije za prikaz korpe	
	2. Prikaz korisničke korpe
3. Odabir opcije za kupovinu artikala iz korpe	
	4. Prikaz forme za upis podataka o plaćanju
5. Popunjavanje podataka o plaćanju	
	6. Validacija unesenih podataka
	7. Prikaz poruke o neuspješnoj kupovini zbog nevalidnih podataka o plaćanju

Tabela A.14: Kupovina artikala iz korpe - tok događaja, neuspješan završetak

A.5 Brisanje artikala

Naziv	Brisanje artikala
Opis	Administrator ima dostupan prikaz svih artikala u bazi kako bi vršio određene izmjene, želi obrisati određeni artikal
Vezani zahtjevi	/
Preduslovi	Prijavljen kao administrator
Posljedice – uspješan završetak	Uspješno brisanje artikla
Posljedice – neuspješan završetak	/
Primarni akteri	Administrator
Ostali akteri	/
Glavni tok	Administrator bira opciju za pregled svih artikala u bazi. Ovo uključuje i artikle koji nisu dostupni korisniku i gostu za pregled. Izlistavaju se svi artikli. Administrator bira brisanje određenog artikla.
Proširenja/alternative	/

Tabela A.15: Brisanje artikala - scenarij

Korisnik	Sistem
1. Odabir opcije za prikaz svih artikala iz baze	
	2. Prikaz svih artikala iz baze
3. Odabir opcije za brisanje određenog artikla	
	4. Uklanjanje artikla iz baze
	5. Prikaz ažurirane liste artikala

Tabela A.16: Brisanje artikala - tok događaja, uspješan završetak

A.6 Dodavanje novog artikla

Naziv	Dodavanje novog artikala
Opis	Administrator ima dostupan prikaz svih artikala u bazi kako bi vršio određene izmjene, želi dodati novi artikal
Vezani zahtjevi	/
Preduslovi	Prijavljen kao administrator
Posljedice – uspješan završetak	Uspješno dodavanje artikla
Posljedice – neuspješan završetak	/
Primarni akteri	Administrator
Ostali akteri	/
Glavni tok	Administrator bira opciju za pregled svih artikala u bazi. Ovo uključuje i artikle koji nisu dostupni korisniku i gostu za pregled. Izlistavaju se svi artikli. Administrator bira opciju za dodavanje artikla. Popunjava formu za dodavanje. Sistem validira unesene podatke i spašava novi artikal.
Proširenja/alternative	/

Tabela A.17: Dodavanje novog artikala - scenarij

Korisnik	Sistem
1. Odabir opcije za prikaz svih artikala iz baze	
	2. Prikaz svih artikala iz baze
3. Odabir opcije za dodavanje novog artikla	
	4. Prikaz forme za dodavanje artikla
5. Popunjavanje forme sa podacima o artiklu	
	6. Validacija unesenih podataka
	7. Spašavanje artikla
	8. Prikaz poruke o uspješno dodanom artiklu

Tabela A.18: Dodavanje novog artikala - tok događaja, uspješan završetak

A.7 Izmjena informacija o artiklu

Naziv	Izmjena informacija o artiklu
Opis	Administrator ima dostupan prikaz svih artikala u bazi kako bi vršio određene izmjene, želi izmijeniti postojeći artikal.
Vezani zahtjevi	/
Preduslovi	Prijavljen kao administrator
Posljedice – uspješan završetak	Uspješne izmjene artikla
Posljedice – neuspješan završetak	/
Primarni akteri	Administrator
Ostali akteri	/
Glavni tok	Admin bira opciju za pregled svih artikala u bazi. Ovo uključuje i artikle koji nisu dostupni korisniku i gostu za pregled. Izlistavaju se svi artikli. Bira opciju izmjene određenog artikla. Popunjava formu o novim podacima o artiklu. Sistem validira unos i spašava artikal.
Proširenja/alternative	/

Tabela A.19: Izmjena informacija o artiklu - scenarij

Korisnik	Sistem
1. Odabir opcije za prikaz svih artikala iz baze	
	2. Prikaz svih artikala iz baze
3. Odabir opcije za izmjenu postojećeg artikla	
	4. Prikaz forme za izmjenu artikla
5. Popunjavanje forme sa podacima o artiklu	
	6. Validacija unesenih podataka
	7. Spašavanje artikla
	8. Prikaz poruke o uspješno izmijenjenom artiklu

Tabela A.20: Izmjena informacija o artiklu - tok događaja, uspješan završetak

A.8 Brisanje korisnika

Naziv	Brisanje korisnika
Opis	Administrator ima mogućnost brisanja korisnika.
Vezani zahtjevi	/
Preduslovi	Prijavljen kao administrator
Posljedice – uspješan završetak	Uspješno obrisan korisnik
Posljedice – neuspješan završetak	/
Primarni akteri	Administrator
Ostali akteri	/
Glavni tok	Admin bira opciju za pregled svih korisnika u bazi. Nakon čega se prikazuje lista svih korisnika. Administrator bira opciju za brisanje konkretnog korisnika.
Proširenja/alternative	/

Tabela A.21: Brisanje korisnika - scenarij

Korisnik	Sistem
1. Odabir opcije za prikaz svih korisnika iz baze	
	2. Prikaz liste svih korisnika
3. Odabir opcije za brisanje određenog korisnika	
	4. Uklanjanje korisnika iz baze podataka
	5. Prikaz poruke o uspješno obrisanom korisniku

Tabela A.22: Brisanje korisnika - tok događaja, uspješan završetak

A.9 Prikaz završenih kupovina

Naziv	Prikaz završenih kupovina
Opis	Administrator ima mogućnost pregleda završenih kupovina.
Vezani zahtjevi	/
Preduslovi	Prijavljen kao administrator
Posljedice – uspješan završetak	Uspješan prikaz završenih kupovina
Posljedice – neuspješan završetak	/
Primarni akteri	Administrator
Ostali akteri	/
Glavni tok	Administrator bira opciju za prikaz završenih kupovina. Izlistava se lista završenih kupovina.
Proširenja/alternative	/

Tabela A.23: Prikaz završenih kupovina - scenarij

Korisnik	Sistem
1. Odabir opcije za prikaz svih završenih kupovina	
	2. Prikaz završenih kupovina

Tabela A.24: Prikaz završenih kupovina - tok događaja, uspješan završetak

A.10 Registracija

Naziv	Registracija računa
Opis	Gost ima mogućnost registracije vlastitog računa
Vezani zahtjevi	/
Preduslovi	/
Posljedice – uspješan završetak	Kreira se račun
Posljedice – neuspješan završetak	Uneseni podaci nisu validni i registracija nije završena
Primarni akteri	Gost
Ostali akteri	/
Glavni tok	Gost bira mogućnost registracije. Prikazuje se forma sa ličnim podacima o korisniku. Gost upisuje podatke. Sistem validira podatke te ukoliko su validni, registruje novog korisnika.
Proširenja/alternative	/

Tabela A.25: Registracija računa - scenarij

Korisnik	Sistem
1. Odabir opcije za registracijom	
	2. Prikaz forme za registraciju
3. Popunjavanje forme	
	4. Validacija unosa
	5. Dodavanje novog korisnika u bazu
	6. Poruka o uspješnoj registraciji

Tabela A.26: Registracija računa - tok događaja, uspješan završetak

Korisnik	Sistem
1. Odabir opcije za registracijom	
	2. Prikaz forme za registraciju
3. Popunjavanje forme	
	4. Validacija unosa
	5. Poruka o neuspješnoj registraciji zbog nepravilnog unosa

Tabela A.27: Registracija računa - tok događaja, neuspješan završetak

A.11 Login

Naziv	Prijava (Login)
Opis	Korisnik sa izrađenim računom ima mogućnost prijave
Vezani zahtjevi	Korisnik ima izrađen račun
Preduslovi	Prijavljen kao administrator
Posljedice – uspješan završetak	Korisnik se prijavljuje na sistem
Posljedice – neuspješan završetak	Prikazuje se poruka ukoliko nije došlo do prijave zbog nepravilnog unosa
Primarni akteri	Korisnik, Administrator
Ostali akteri	/
Glavni tok	Korisnik ili Administrator bira opciju prijave. Prikazuje se forma za prijavu. Korisnik ili Administrator upisuje neophodne podatke u formu. Sistem validira unesene podatke, vrši autentifikaciju i autorizaciju ako je unos ispravan. Ako unos nije ispravan prikazuje se poruka.
Proširenja/alternative	/

Tabela A.28: Login - scenarij

Korisnik	Sistem
1. Odabir opcije za login	
	2. Prikaz forme za unos podataka neophodnih za autentifikaciju
3. Unos podataka u formu	
	4. Validacija unosa
	5. Prijava korisnika u sistem
	6. Prikaz početne stranice

Tabela A.29: Login - tok događaja, uspješan završetak

Korisnik	Sistem
1. Odabir opcije za login	
	2. Prikaz forme za unos podataka neophodnih za autentifikaciju
3. Unos podataka u formu	
	4. Validacija unosa
	5. Prikaz poruke za neuspješan login, razlog zašto login nije uspio

Tabela A.30: Login - tok događaja, neuspješan završetak