Napadi umetanjem SQL programskog koda

Bedeković, Nenad

Sveučilište
Sjever

Završni rad br. 750/MM/2021

Napadi umetanjem SQL programskog koda

Nenad Bedeković, 1654/601

Varaždin, rujan 2021. godine

Sveučilište
Sjever

Odjel za multimediju

Završni rad br. 750/MM/2021

Napadi umetanjem SQL programskog koda

Student

Nenad Bedeković, 1654/601

Mentor

Doc. dr. sc. Ladislav Havaš, dipl.ing.el.

Varaždin, rujan 2021. godine

UNIVERSITY NORTH

Prijava završnog rada

Definiranje teme završnog rada i povjerenstva

ODJEL Odjel za multimediju

STUDIJ preddiplomski stru ni studij Multimedija, oblikovanje i primjena

PRISTUPNIK Nenad Bedeković | JMBAG 0231011848

DATUM 06.09.2021 | KOLEGIJ Baze podataka i SQL

NASLOV RADA Napadi umetanjem SQL programskog koda

NASLOV RADA NA ENGL. JEZIKU SQL injection attacks

MENTOR doc. dr. sc. Ladislav Havaš | ZVANJE docent

ČLANOVI POVJERENSTVA

1. dr. sc. Tomislav Horvat, predavač - predsjednik
2. Dražen Crčić, predavač - član
3. doc. dr. sc. Ladislav Havaš - mentor
4. mr. sc. Ivan Šumiga, viši predavač - zamjenski član
5.

Zadatak završnog rada

BROJ 750/MM/2021

OPIS

Web aplikacija se sastoji od programskog koda koji se izvršava na poslužiteljskog strani koji je u direktnoj, praktički stalnoj, komunikaciji s bazom podataka. Ukoliko Web aplikacije nisu dobro zaštićene, omogućavaju zlonamjernim korisnicima izvođenje neželjinih operacija na bazi podataka. Jedan od najčešćih napada na bazu podataka su napadi umetanjem SQL programskog koda. U radu je potrebno ukratko opisati SQL upitni jezik, definirati osnove arhitekture Web aplikacije te opisati osnove sustava za upravljanje bazom podataka.

U radu je potrebno detaljnije opisati i na konkretnim primjerima prikazati najčešće načine napada umetanjem SQL programskog koda. Također je potrebno navesti i metode zaštite od navedenih napada.

ZADATAK URUČEN 8.9.2021. | POTPIS MENTORA

REPUBLIKA HRVATSKA 5 SVEUČILIŠTE SJEVER KNJIŽNICA SVEUČILIŠTE SJEVER

Predgovor

Ideja za odabir ove teme proizlazi iz znatiželje da neko opremljen samo običnim internet preglednikom, osnovnim znanjem baza podataka i s par znakova može počiniti štetu iskorištavajući sigurnosne propuste sustava. Nakon više od dvadesetak godina postojanja, nevjerojatno je da postoji tehnika zlonamjernog napada koja se i dalje uspješno koristi. Cilj rada je čitatelje neovisno o iskustvu programiranja i na razumljiv način upoznati s posljedicama koje mogu nastati lošim programskim praksama u razvoju web aplikacija.

Ovim putem zahvaljujem mentoru doc. dr. sc. Ladislavu Havašu na iskazanom povjerenju, potpori i smjernicama u pisanju ovog rada.

Sažetak

Završni rad upoznaje čitatelje s glavnim aspektima napada umetanjem SQL koda. Prolazi se kroz temelje kao osnovni način rada web aplikacija, SQL jezik i baze podataka. Internetom kruži velik broj namjerno ranjivih web aplikacija koje služe kao legalni testni teren za prakticiranje napada umetanjem SQL koda. U radu će biti prikazane tehnike napada na Acunetix testnu web aplikaciju napisanu korištenjem skriptnog jezika PHP i relacijske baze podataka MySQL.

U praktičnom dijelu sama izvedba napada uvelike ovise o sustavu za upravljanje bazama podataka pa je prikazana sintaksa namijenjena samo za MySQL sustav. Za primjer automatizirane izvedbe napada je korišten program SQLmap u Linux Kali virtualiziranom okruženju. U sklopu rada se nalaze i smjernice oko sigurnosti za sve programere koji pokušavaju zaštititi svoje baze podataka.

Ključne riječi: baze podataka, MySQL, napad umetanjem SQL koda, SQL, SQLmap

Abstract

The final paper introduces readers to the main aspects of SQL injection attacks. It briefly touches the basics of web application architecture, SQL language and databases. A large number of intentionally vulnerable web applications are circulating on the Internet that serve as a legal test ground for practicing SQL injection attacks. For demonstration purposes the attacks will target an Acunetix test web application created using PHP backend language that uses MySQL as the relational database management system.

In the practical part, the execution of the attack itself largely depends on the database management system, so the displayed syntax is intended only for the MySQL database management system. Example of an automated attack will be executed on SQLmap in a Linux Kali virtualized environment. The paper also includes security guidelines for all developers trying to protect their databases.

Keywords: SQL, SQL injection attack, MySQL, relational database management system, SQLmap, databases

Popis korištenih kratica

HTML	HyperText Markup Language	
CSS	Cascading Style Sheets	
URL	Uniform Resource Locator	Usklađeni lokator sadržaja
SQL	Structured Query Language	Strukturni upitni jezik
DBMS	Database Management System	Sustav upravljanja bazama podataka
CRUD	Create, read, update and delete	Kreiranje, čitanje, ažuriranje i brisanje
PHP	Hypertext Preprocessor	

Sadržaj

1. Uvod

Danas se baze podataka koristi gotovo za sve sustave i web stranice jer podaci moraju biti negdje pohranjeni. Prikupljanje masivne količine podataka su revolucionizirale današnji način života. Kako se osjetljivi podaci pohranjuju u bazu podataka sve je veći rizik uključen u radu sigurnosnih sustava. Napad umetanjem SQL koda barem na prvi pogled radi na vrlo jednostavan način. Napadač šalje zlonamjerni SQL upit u polje koje se može ispuniti i iskorištava ranjivost u samoj SQL implementaciji aplikacije. Iako se ovaj vektor može koristiti za napad na bilo koju bazu podataka, najčešće mete su ipak web aplikacije.

U radu će biti demonstrirana tehnika napada na relacijski sustav baze podataka i prevencije. Organizacije ovise o njima jer koristeći baze podataka mogu analizirati prikupljene informacije i učinkovitije upravljati svojim resursima, a sve u svrhu donošenja kvalitetnih poslovnih odluka. Kako bi zaštitile svoje podatke, organizacije koriste različite mjere jer posljedice sigurnosnog propusta mogu biti kobne. Baze podataka često podržavaju vrlo složene upite i očekuje se da pružaju informacije u stvarnom vremenu (engl. *real time*). Funkcioniraju u tandemu s aplikacijama gdje se obično koristi sučelje na strani klijenta koje služi za otvaranje veze s bazom podataka.

Većina današnjih web aplikacija omogućuje posjetiteljima unos i dohvat podatka, a to ujedno i pruža jedinstven ulaz u bazu podataka. Pristup posjetiteljima stranice uključuje pretraživanje, korištenje obrasca za prijavu ili kontakta, a upravo takvi prozori se koriste za pristup bazi podataka. Neispravno programirane aplikacije omogućit će napadaču da koristi ranjivosti kao ulaznu točku i pristupi bazi podatka, a nakon toga može po želji manipulirati podacima. Otkrivanje o kojem se sustavu za upravljanje bazom podataka radi je ključno za daljnje iskorištavanje ranjivosti. Bez toga ne bi bilo moguće odrediti koje tablice tražiti, o kojim ugrađenim funkcijama sustava se radi i koje načine detekcije izbjeći. Napad umetanjem SQL koda (engl. *SQL injection)* trenutno je jedan od popularnijih vektora napada pošto ovaj oblik ne zahtjeva puno resursa i znanja. Vektor napada definira put koji napadač koristi u svrhu iskorištavanja sigurnosnog propusta.

Još od 1998. godine kada je prvi put evidentiran pa do danas, napad umetanjem SQL koda još uvijek drži visoko mjesto prema OWASP (engl. *Open Web Application Security Project*) organizaciji, usmjerenoj prema poboljšanju sigurnosti softvera. Zanimljivo je da se kroz svoj životni vijek napadi umetanjem SQL koda nisu puno promijenili. Uobičajena je ranjivost koja unatoč tome što se relativno lako može otkloniti, nastavlja stvarati probleme i ako se ne otkrije prije same implementacije pruža prozor mogućnosti potencijalnim napadačima.

Kroz rad će se prezentirati direktan i automatiziran pristup pronalaženja ranjivosti te izvedbe napada umetanjem SQL koda. Koristit će se namjerno ranjiva PHP aplikacija na stranici "Acunetix.com"[1] koja nudi besplatan i legalan teren za testiranje kako bi uz praksu bolje razumjeli greške u programiranju.

Acunetix[2] je osnovan za borbu protiv alarmantnog porasta web napada. Donosi opsežan skup opcija automatiziranih i ručnih alata za testiranje penetracije, omogućujući sigurnosnim stručnjacima da izvrše procjene ranjivosti i poprave otkrivene prijetnje samo s jednim proizvodom. Web aplikacija upotrebljava MySQL kao sustav za upravljanje bazom podataka, tako da ćemo obradit napade koji su namijenjeni specifično tome sustavu.

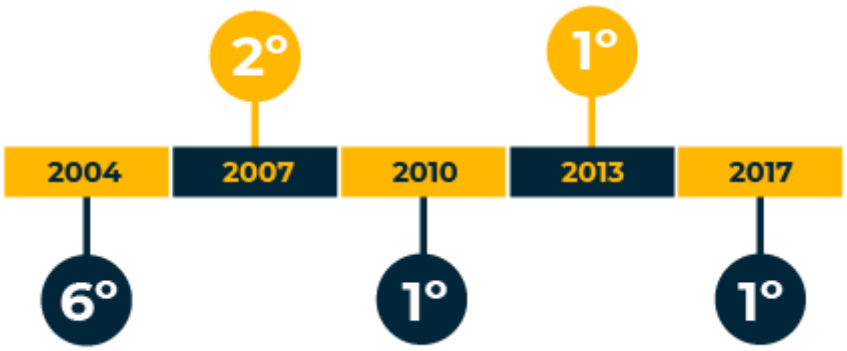
[1] http://testphp.vulnweb.com - ranjiva Acunetix PHP web aplikacija
[2] https://www.acunetix.com - Acunetix web stranica

2. Napadi umetanjem SQL koda

Godine 1998. Jeff Forristal, istraživač za sigurnost pod pseudo imenom "Rain Forest Puppy", objavljuje članak u časopisu Phrack gdje opisuje probleme s tadašnjom verzijom Microsoft SQL servera. [1] Opisao je način kako osoba s osnovnim vještinama programiranja može s neovlaštenim SQL naredbama na nezaštićenoj web stranici, dobiti osjetljive informacije iz baze podataka. Kada je obavijestio Microsoft o tome nisu to doživljavali kao neki problem.

Tek nakon četiri godine, 2002. napadi umetanjem SQL koda privlače pozornost u računalnoj sigurnosti. Pojava velikog broja računalnih virusa i crva u kratkom vremenskom periodu, alarmira sigurnosne stručnjake da je potrebno krenuti u razvoj softverske tehnologije za obranu od različitih načina napada. Bill Gates svojim zaposlenicima šalje dopis "Microsoft toward Trustworthy Computing" o sigurnosti i povjerenju svojeg softvera. [2] Ova inicijativa je pokrenula razvoj sigurnosne tehnologije te se počinju ozbiljnije shvaćati propusti koji nastaju lošim programiranjem. Od tada se pojavljuje bezbroj članaka o napadima umetanjem SQL koda.

Prvi sigurnosni alat kojem je glavni zadatak prevencija i detekcija napada umetanjem SQL koda, pojavljuje se 2005. godine pod nazivom AMNESIA (engl. *Analysis and Monitoring for Neutralizing SQL Injection Attacks*). Alat radi model tablice mogućeg SQL koda koji bi se upotrijebio za napad i tada analizira svaku točku gdje bi se mogao umetnuti kod. [3] OWASP predstavlja neprofitnu zajednicu na internetu kojoj je cilj poboljšati sigurnost softvera uz redovitu objavu dokumentacija, alata i tehnologija kako bi se zaštitili od potencijalnih sigurnosnih rizika. Svake tri do četiri godine objavljuje se OWASP Top 10 rangiranje i savjeti za zaštitu od deset najkritičnijih sigurnosnih rizika web aplikacija, temeljen na konsenzusu sigurnosnih stručnjaka. [4] Još od prvih objava napadi umetanjem SQL koda se redovito razvijaju i zauzimaju visoko mjesto u ranjivosti web aplikacija.



Slika 2.1 - OWASP Top 10 rangiranje napada umetanjem SQL koda po godinama objave [4]

Može se zaključiti da skok u popularnosti leži u samoj jednostavnosti korištenja. Na ranjivoj web aplikaciji uz samo nekoliko linija koda na aplikacijskom sloju moguće je poslati upit prema bazi podataka. Za razliku od drugih vektora napada resursi koji su potrebni za samu izvedbu su krajnje minimalni i svode se na jedno računalo. Drugi napadi zahtijevaju ogromne resurse i vrijeme koji uključuju tisuće računala kao kroz angažman Botneta u izvršavanju napada. Neki od poznatijih napada umetanjem SQL koda redovito dospjevaju u medije:

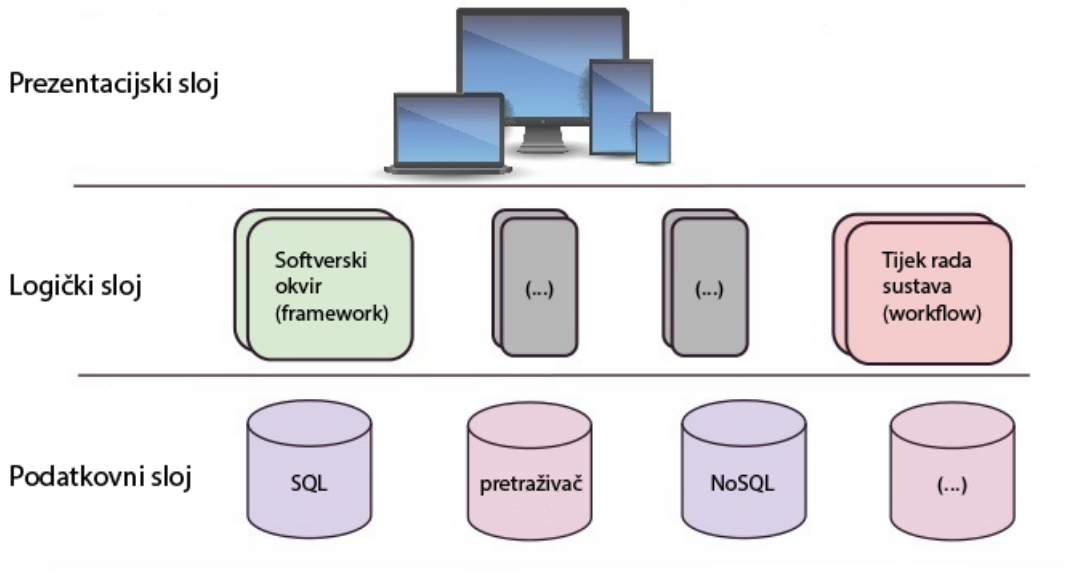
Godina	Reference	Opis
2009.	[5]	ukradeno 130 milijuna brojeva kreditnih kartica, tada prozvano najvećom krađom identiteta u Američkoj povijesti
2009.	[6]	RockYou stranica koja se bavila razvijanjem web aplikacija za socijalne medije ima curenje 32 milijuna nekriptirana korisnička računa i lozinka, spremljenih u bazi podataka s tekstualnim oblikom
2011.	[7]	MySQL službena stranica SUBP sustava ima curenje baze podataka s korisničkim računima i lozinkama vodećih ljudi u kompaniji
2011.	[8]	Sony Pictures, službena stranica poznate kompanije Sony ima curenje baze podataka s milijunima nekriptiranih korisničkih računa i pripadajućim informacijama spremljenih u tekstualnom obliku
2012.	[9]	Yahoo ima curenje baze podataka s 450 tisuća korisničkih računa i pripadajućim lozinkama spremljenih u tekstualnom obliku
2015.	[10]	Vtech globalni dobavljač električnih proizvoda ima curenje baze podataka od pet milijuna korisničkih računa
2016.	[11]	Američka državna stranica EAC (*U.S. Election Assistance Commission*) zadužena za pomoć kod provođenja lokalnih i državnih izbora ima curenje baze podataka administratorskih korisničkih računa
2019.	[12]	Georgia Institute of Technology jedan od vodećih Američkih tehničkih fakulteta ima curenje baze podataka od 1,3 milijuna korisničkih podataka

Tablica 2. Prikazani napadi umetanjem SQL koda, godina i opis

2.1. Arhitektura web aplikacije

Da bi shvatili način rada napada umetanjem SQL koda potrebno je poznavati način rada web aplikacija s kojima se susrećemo. Osnovna ideja načina rada aplikacije je da postoji prednje sučelje (engl. *front-end*) koje komunicira sa stražnjim slojem (engl. *back-end*) na poslužiteljskom dijelu aplikacije. Aplikacije su rađene slojevito pa postoji nekoliko tipova arhitektura, a odabir arhitekture ovisi o namjeni.

Kao primjer se uzima web aplikacija napravljena korištenjem troslojne arhitekture koja se sastoji od prezentacijskog, logičkog i podatkovnog sloja. Često se koristi u aplikacijama kao specifična vrsta sustava na relaciji klijent/poslužitelj, pružajući brojne pogodnosti za razvojno okruženje modularizacijom korisničkog sučelja, logike i slojeva za pohranu podataka. Uvijek je moguće fleksibilno razvijati, nadograđivati ili seliti bilo koji sloj neovisno od drugog. Temeljno pravilo ovog modela je da prezentacijski sloj nikada ne komunicira izravno s podatkovnim slojem, sva komunikacija mora proći kroz logički sloj.



Slika 2.2 - Model troslojne arhitekture

Prezentacijski sloj (engl. *presentation tier*) spada u najviši sloj arhitekture aplikacije. Smatra se komunikacijskim slojem gdje klijent putem korisničkog sučelja komunicira s aplikacijom. Prikazuje informacije povezane s uslugama kao što su sadržaj, kupnja, pregledavanje artikla, itd. Komunicira s drugim slojevima iznošenjem rezultata na sloju web preglednik/korisnik te obično se razvija kroz HTML, CSS i JavaScript.

Logički sloj (engl. *logical tier*) kontrolira funkcionalnost aplikacije i obrađuje dobivene podatke. Smatra se srcem aplikacije gdje se provodi obrada dobivenih podataka s prezentacijskog sloja. Razvoj ovog sloja se najčešće radi kroz PHP, Java, Python, C++, Ruby, .NET, itd.

Podatkovni sloj (engl. *data tier*) se sastoji od baze podataka i sustava za upravljanje bazom podataka. Ovdje se informacije pohranjuju i vraćaju na poslužitelju te drže odvojeno od aplikacijskog i logičkog sloja. Primjeri takvih sustava su neki od relacijskih modela baze podataka kao MySQL, Oracle, Microsoft SQL Server, PostgreSQL, itd.

2.2. SQL

Kada se u računalnim sustavima govori o spremanju podataka tada se oslanjamo na baze podataka. Mogu se opisati kao spremnici objekata gdje su informacije smještene na organiziran i strukturiran način. Baze podataka često svoj ciklus započinju u programu za obradu teksta ili proračunskim tablicama, ali kako postaju sve veće, zbog lakše manipulacije ih je korisno prenijeti u sustav za upravljanjem baza podataka. SQL (engl. *Structured Query Language*) je programski jezik stvoren točno za taj zadatak. Dizajniran da komunicira s bazama podataka, omogućuje unos upita i povratnih rezultata. Za razliku od drugih programskih jezika nije kompliciran, već je razumljiv i lagan, a sposoban je izvršavati različite vrste operacija. Prati jedinstvena pravila i smjernice koja ne razlikuju velika i mala slova, a sintaksa ovisi o relacijskoj algebri. Kao i drugi programski jezici SQL je utemeljen na naredbama za tumačenje niza znakova. Oni su umetnuti sa željenom sintaksom gdje određeni izrazi odgovaraju samo jednoj mogućoj operaciji izvedbe. SQL naredbe koristimo za kreiranje, čitanje, ažuriranje i brisanje podataka u tablici. Bitno je poznavati neke od glavnih naredbi kako bi bolje razumjeli mogućnosti napada umetanjem SQL koda:

- `SELECT` - najčešća naredba u SQL-u, njezina svrha je da omogući pretraživanje baze podataka gdje prikazuje rezultat u obliku tablice koja zadovoljava uvjete upita
- `UPDATE` - koristi se za izmjenu postojećih zapisa u tablici
- `CREATE` - upotrebljava se za stvaranje baze podataka i tablica unutar njih
- `INSERT` - dodaje nove vrijednosti u bazu podataka
- `DROP` - briše tablicu ili cijelu bazu podataka, korisna za izvršavanje napada jer može onemogućit funkcionalnosti aplikacije

U sintaksi SQL-a postoje određeni znakovi koji imaju specifične funkcije i često se ponavljaju kada ih zlonamjerno koristimo kod napada umetanja koda. Neki od popularnijih su:

- jednostruki (`'`) i dvostruki (`"`) navodnik se najčešće prvi upotrebljavaju u testiranju ranjivosti baze podataka. U SQL jeziku se koriste kao razdjelnici (engl. *delimiter*) unutar upita. Pojedinačni navodnik ukazuje na početak i kraj nekog niza znakova, dvostruki se može upotrebljavati za isto, ali to ovisi o tipu baze podataka gdje postoje određene razlike u sintaksi.
- dupla crtica (`--`) predstavlja početak komentara koji odgovara slijedu teksta do kraja retka. Umetne li se u dio koda, sve iza njega će sustav smatrati komentarom i biti će ignorirano.

2.3. Sustavi za upravljanje bazom podataka

Bez obzira gdje su podaci fizički pohranjeni, baza podataka djeluje kao jedinstvena cjelina za sustav koji ih koristi. Da bi mogli koristiti bazu podataka potreban nam je način izvođenja takozvanih CRUD (engl. *Create, Read, Update and Delete*) zadataka, odnosno način za stvaranje novih, čitanje i pronalaženje postojećih te ažuriranje i brisanje zastarjelih podataka. Za to je odgovoran sustav za upravljanje bazom podataka koji služi kao sučelje između krajnjeg korisnika i baze podataka. Sustav optimizira obradu podataka te sastoji se od strukturne sheme koja omogućuje lakšu organizaciju i manipulaciju podataka. DBMS (engl. *Database management system*) će definirati pravila i manipulirati formatom podataka, imenima polja, strukturom zapisa i samom strukturom datoteke. Možemo ih podijeliti na dvije velike skupine, relacijske i nerelacijske baze podataka.

Slika 2.3 Relacijske i nerelacijske baze podataka

Relacijska baza podataka je vrsta baze podataka koja pohranjuje i omogućuje pristup skupu podataka koji su međusobno povezani. Sprema podatke u tablicama, pružajući fleksibilan i učinkovit način za pohranu i pristup strukturiranim informacijama. Svaki red u tablici predstavlja zapis s jedinstvenim načinom identifikacije opisan kao atribut ključ (engl. *key*). Stupci u tablicama drže podatke i svaki zapis ima vrijednost za svaki atribut.

Nerelacijske ili NoSQL baze podataka omogućuje mehanizme za pohranu i pronalaženje podataka koje su bazirane na bilo koji drugi način osim tabličnih odnosa koji se koriste u relacijskim bazama podataka. Probila su se četiri glavna tipa NoSQL baza podataka gdje svaka ima svoje karakteristike i koriste se za specifičan zadatak ovisno o tipu:

- dokument baze (engl. *document databases*)
- model ključ-vrijednost (engl. *key-value databases*)
- model stupaca (engl. *column oriented databases*)
- graf baze (engl. *graph databases*)

Svi sustavi koriste SQL jezik, ali u svakom postoje određene varijacije iste naredbe. Izviđanje je uvijek prvi korak u traženju ranjivosti aplikacija. Informacije o kojem se sustavu za upravljanje bazom podataka radi, uvelike olakšavaju napadaču formiranje pristupa s kojim će pokušati izvršiti napad. Neki od poznatijih komercijalnih sustava za upravljanje bazom podataka su:

- MySQL - najpoznatiji relacijski sustav otvorenog koda (engl. *open source*). Brz, fleksibilan i jednostavan za korištenje, često korišten kao referenca za cjelokupnu SQL sintaksu, najčešće se koristi s web aplikacijama temeljenima na PHP jeziku.
- OracleDB - modularan i prvi izbor većih tvrtki zbog učinkovitog upravljanja bazama podataka i sigurnosnih opcija. Visoka kompatibilnost s mnogim tehnologijama, programskim jezicima i modelima baze podataka, zbog čega je i skuplji izbor.
- PostgreSQL - relacijsko-objektno orijentiran sustav otvorenog koda, napredan i visoko modularan.
- MongoDB - najpopularniji NoSQL sustav. Fleksibilan i visoko kompatibilan, funkcionira na principu kolekcije dokumenata u JSON formatu.
- Microsoft SQL - relacijski sustav koji je najčešće rješenja u poslovnom svijetu jer je optimiziran za rad s Windows sustavima.

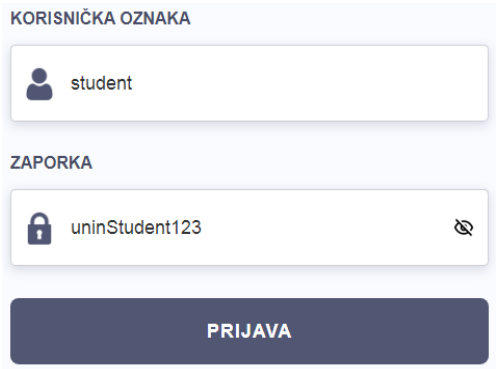
3. Osnovni koncept napada umetanjem SQL koda

Ovisno o cilju, napad umetanjem SQL koda se može koristiti na više načina, ali najčešće da se dođe do nedostupnih informacija slanjem upita prema bazi podataka samom manipulacijom programske logike aplikacije. Tada je napadač u mogućnosti zaobići autentifikaciju, dobiti pristup bazi podataka, kopirati, modificirati i izbrisati podatke po želji. Izviđanje i prikupljanje informacija je uvijek prvi korak svakog uspješnog plana napada, a identifikacije i testiranje konfiguracije sustava uvelike pomaže kod biranja načina na koji će se izvršiti napad.

Napad umetanjem SQL koda je tehnika umetanja programskog koda namijenjena aplikacijama koje koriste bazu podataka gdje se posebno izveden SQL upit ubacuje u polja za unos podataka i tada prosljeđuje na izvedbu. To može biti bilo koje polje za unos u aplikaciji ili URL (engl. *Uniform Resource Locator*) adresa. Iskorištava se sigurnosna ranjivost gdje se uneseni podaci ne provjeravaju od strane poslužitelja jer uspjeh napada ovisi o sintaksi ispravno napisanog SQL upita. Ako kroz poslan upit aplikacija odgovori s greškom u bazi podataka tada postoji mogućnost da je aplikacija ranjiva. Pomoću generiranih informacija o grešci, napadač je u mogućnosti razumjeti na koji način funkcionira baza podataka.

Važno je vizualizirat programsku logiku koja je ugrađena u sustav aplikacije pa tako napadač pokušava modificirat poslan upit zbog kojeg je dobio prvobitnu grešku i testirat daljnje ponašanje baze podataka. Na dobivene podatke od strane korisnika, aplikacijski poslužitelj se pokušava spojiti s bazom podataka i prosljeđuje upit na izvedbu. Za napad i sakupljanje informacija se koristi isti kanal komunikacije pa je moguće samo kroz jedno korisničko polje za unos dobiti uvid u cijelu bazu podataka. Napadač nakon uspješnog napada može čitati i manipulirat postojećim podacima kroz brisanje ili kreiranje novih, pa tako postati i administrator baze podataka. Najviše se koristi kao vektor napad na web aplikacije, ali može se koristiti za napad bilo koje SQL baze podataka.

Uzmimo za primjer aplikacijsku formu za autentifikaciju s kojom se korisnici često susreću. Uobičajeni obrazac za prijavu se sastoji od unosa teksta korisničkog imena i lozinke, nakon unosa aplikacija traži podudaranje s umetnutim podacima kako bi provjerila da li navedeni par korisničkog imena i lozinke postoji unutar istog zapisa baze podataka, tj. da je povezan s jednim korisnikom.

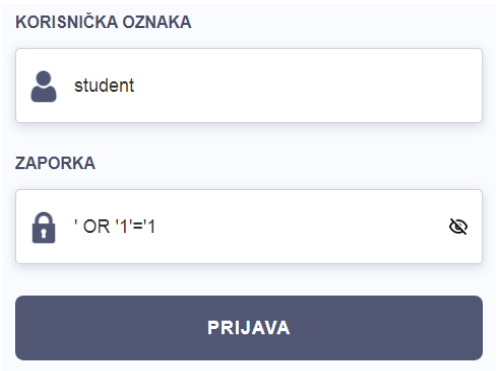


Slika 3.1 Obrazac za prijavu na web stranicu[3]

U našem slučaju upisi u polju korisnička oznaka `student` i lozinka `uninStudent123` su jedine stvari pod našom kontrolom pa na dobivene unose prvo će zahtjev proći kroz aplikaciju na logičkom sloju koja će ga proslijediti u bazu podataka na provjeru gdje bi SQL upit izgledao kao:

```
SELECT * FROM korisnik
WHERE ime='student' AND zaporka='uninStudent123'
```

Ako je dobiveni unos prošao provjeru i podaci se podudaraju s onima u bazi podataka, upit je istinit te se korisnik uspješno prijavljuje (faza autentifikacije) , a ako se ne podudara onda je neistinit i pristup će biti onemogućen. Napad se događa kada aplikacija obrađuje posebno napisan korisnički unos na način koji omogućuje da izađe iz konteksta podataka i uđe u kontekst naredbe. Na taj način se mijenja struktura SQL izraza koja se izvršava, a krajnji cilj svakog SQL upita je da bude istinit, napravljen od dobivenog unosa (engl. *string*) i naredbe (engl. *statement*) koje kontroliraju na koji način će se izvršiti upit.



Slika 3.2 Obrazac za prijavu na web stranicu sa zlonamjernim kodom

[3] https://www.isvu.hr/studomat/hr/prijava - Studomat obrazac za prijavu

```
SELECT * FROM korisnik
WHERE ime='student' AND zaporka='' OR '1'='1'
```

To je jedan od najosnovnijih upita s kojim bi napadač mogao zaobići autentifikaciju. Unese li se zlonamjerni upit (engl. *payload*) `' OR '1'='1` u polje lozinke na aplikacijskoj formi, SQL upit koji se provede će biti izvršen u bazi podataka bez obzira na naveden unos, a razlog je promijena logike upita pomoću naredbe `OR (ILI)` već na sloju aplikacije. Stavljanje bilo kojeg logičkog izraza (engl. *boolean*) s istinitim iskazom s operatorom `OR` koji vraća istinu ako je bilo koji od dva uvjeta istinit, znači da će se uvijek imati istiniti rezultat. S navedenim zlonamjernim upitom i matematičkom logikom jedan će uvijek biti jednak jedan. Na ovaj način ako točka ulaza nije sanirana i ne provjerava uneseni podatak napadač se uspješno prijavljuje na stranicu, neovisno koja je lozinka. Također moguće je izvršiti napad i putem poveznice na internet pregledniku. Uzmimo za primjer sljedeću poveznicu:

```
http://www.primjer.hr/proizvod.php?id=1
```

Poveznica prikazuje proizvod koja ima `id=1` , vizualizira se u bazi podataka kao parametar proizvoda s brojem identifikatora i predstavlja vrijednost u nekoj tablici, a zamišljen SQL upit bi izgledao kao:

```
SELECT id, ime, opisProizvoda FROM proizvod WHERE id=1
```

Ako stranica ne provjerava unesene izraze napadač bi i ovdje mogao manipulirati bazom podataka s nastavkom unosa u poveznici internet preglednika.

```
http://www.primjer.hr/proizvod.php?id=1 OR 1=1
```

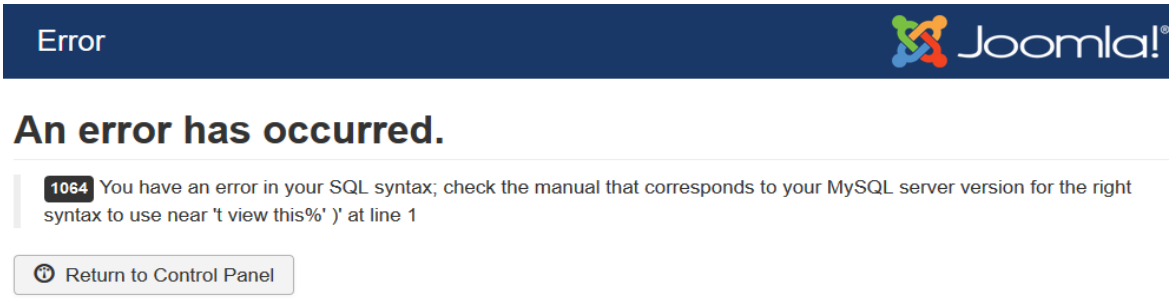
Tada bi zamišljen SQL upit u bazi podataka izgledao kao:

```
SELECT id, ime, opisProizvoda FROM proizvod WHERE id=1 OR 1=1
```

Pošto je upit istinit po `1=1` logici, dobit će se prikazani svi proizvodi koji su popularizirani u bazi podataka. Uvid u sve proizvode ne izgleda kao efektivan napad, ali ako se može na bilo koji način manipulirat unosima, otvorene su mogućnosti gdje sve ovisi o znanju i namjerama napadača.

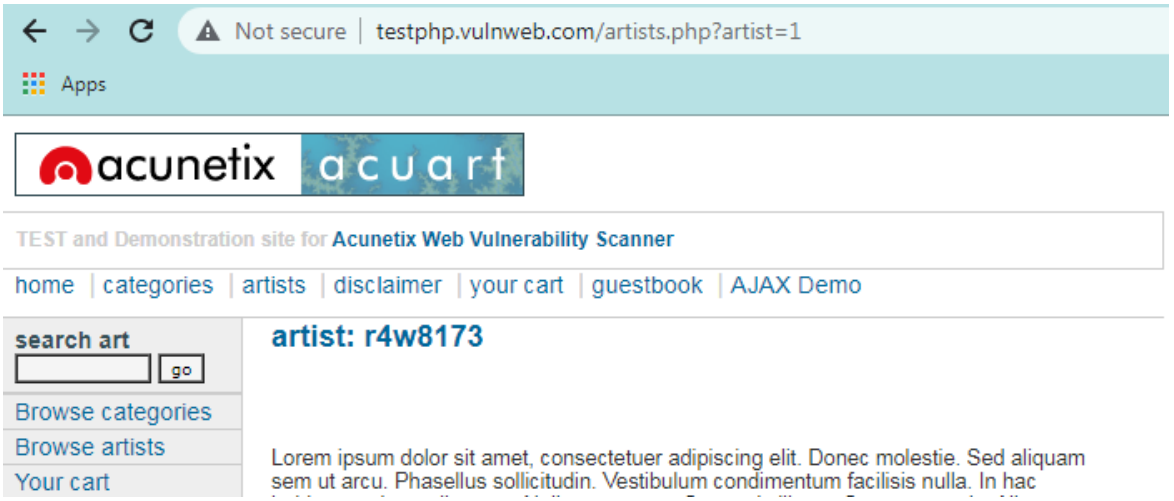
3.1. Napad temeljen na greškama - (engl. *Error based*)

Kada prosječan korisnik vidi neku zbunjujuću grešku aplikacije, često ignorira i ne razmišljamo puno o tome, ali u pravim rukama prikazane informacije mogu puno otkriti o načina rada sustava. Lov na greške i informacije prikazane u njima, uvelike pomažu napadaču u biranju vektora napada i razumijevanju upita koji se šalje prema bazi.



Slika 3.3 Primjer generirane greške web aplikacije

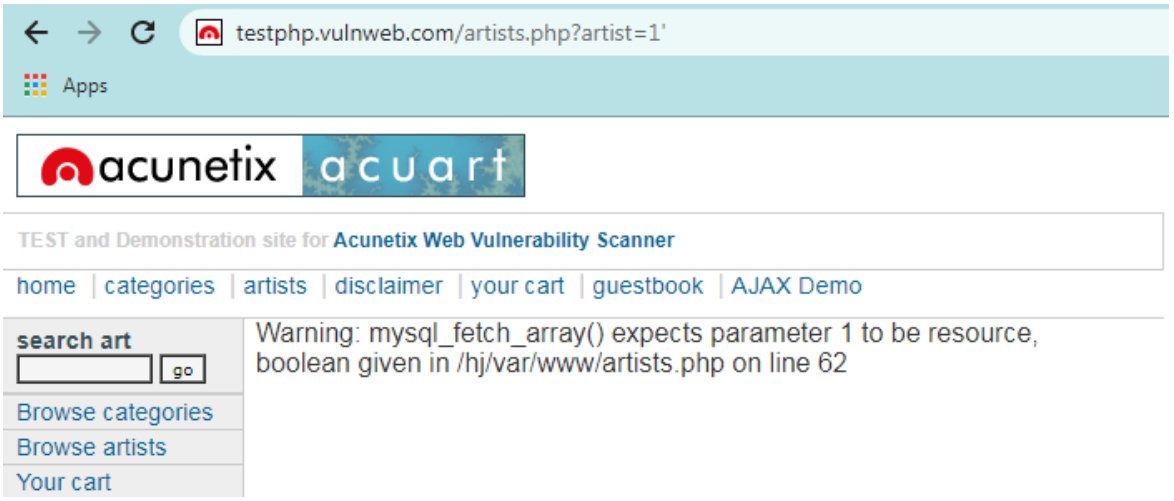
Kako bi se pronašao način izvršavanja napada umetanjem SQL koda, započinje se s ubacivanjem specijalnih znakova namijenjenih samo za bazu podataka na sučelje korisničkog unosa aplikacije ili izravno kroz URL. Prati se komunikacija između aplikacije i baze podataka te gleda kako će se ona ponašati. U slučaju da se u bazi podataka desila greška, aplikacija može generirati grešku kao povratnu informaciju i odati korisne informacije o bazi podataka.



Slika 3.4 Prikaz Acunetix web aplikacije

Pokušava se pronaći način izlazaka iz postojećeg SQL upita gdje se često upotrebljavaju znakovi kao jednostruki navodnici (`'`), dvostruki navodnici (`"`), točka-zarez (`;`) ili dupla crtica (`--`) uneseni u ranjiva polja za unos kako bi se pokušao omesti postojeći upit.

```
http://testphp.vulnweb.com/artists.php?artist=1'
```



Slika 3.5 Greška web aplikacije nakon unesene zlonamjerne vrijednosti

U ovom slučaju ako se ciljano u internet preglednik unese po sintaksi krivi unos kao jednostruki navodnik (`'`) na kraju poveznice, dobit će se informacije o grešci koja govori o funkciji i kojem se sustavu za upravljanje bazama podataka radi. Funkcija `mysql_fetch_array()` očekuje resurs, a umjesto toga prima `boolean` vrijednost.

Ova greška se često pojavljuje ako se nije provjeravalo da li se MySQL upit uspješno izvršio. Prikazana PHP funkcija omogućuje pristup pohranjenim podacima u povratnom rezultatu ako je logički upit istiniti. Namjerno krivo unesena varijabla, dala je logički neistinitu vrijednost umjesto resursa pa se povratno vratila greška. Pogledom na poveznicu može se vizualizirati kako bi izgledao upit prema bazi podataka koji je generirao grešku, upitnik (`?`) predstavlja još nepoznatu tablicu.

```
SELECT * FROM ? WHERE artist=1'
```

Pošto je jasno da se radi o MySQL poslužitelju, napadaču je ovo bitna informacija, ne samo da se štedi vrijeme već sužava broj vektora napada koji se mogu isprobati. Može se ići i korak dalje te saznati koliko stupaca sadrži tablica kroz seriju postavljenih upita prema bazi podataka. Pratimo ponašanje aplikacije:

Poslan upit	Odgovor
`http://testphp.vulnweb.com/artists.php?artist=1 order by 1`	nema greške
`http://testphp.vulnweb.com/artists.php?artist=1 order by 2`	nema greške
`http://testphp.vulnweb.com/artists.php?artist=1 order by 3`	nema greške
`http://testphp.vulnweb.com/artists.php?artist=1 order by 4`	greška

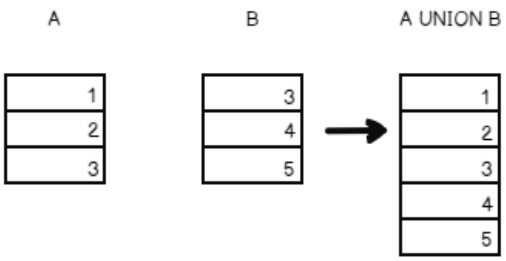
Tablica 3.1 Prikazano je ponašanje aplikacije tijekom napada temeljen na greškama `ORDER BY` naredbom

Ponašanje aplikacije je otkrilo da se radi o tri stupca u tablici. Naredba `ORDER BY` sortira po redoslijedu zapis iz rezultata upita, nije čak ni potrebno poznavati ime stupca u tablici. Umjesto ciljanog naziva, s brojem se može jednostavno specificirat njegov redni položaj. Postepenim povećavanjem broja se pokušava izazvati greška i tada se sa sigurnošću može pretpostaviti koliki broj stupaca postoji u tablici. Polako se dobiva mentalna slika tablice pa kroz sljedeći primjer pomoću `UNION` operatora dobit će se uvid u bazu podataka.

3.2. Napad pomoću UNION operatora - (engl. *UNION-based*)

U SQL-u operator unije (engl. `UNION`) se koristi za spajanje rezultata dva ili više `SELECT` izraza gdje je potrebno pratit određena pravila:

- svaki `SELECT` upit mora imati isti broj stupaca
- izrazi moraju imati isti redosljed
- stupci moraju sadržavati sličan tip podataka



Slika 3.6 Primjena UNION operatora

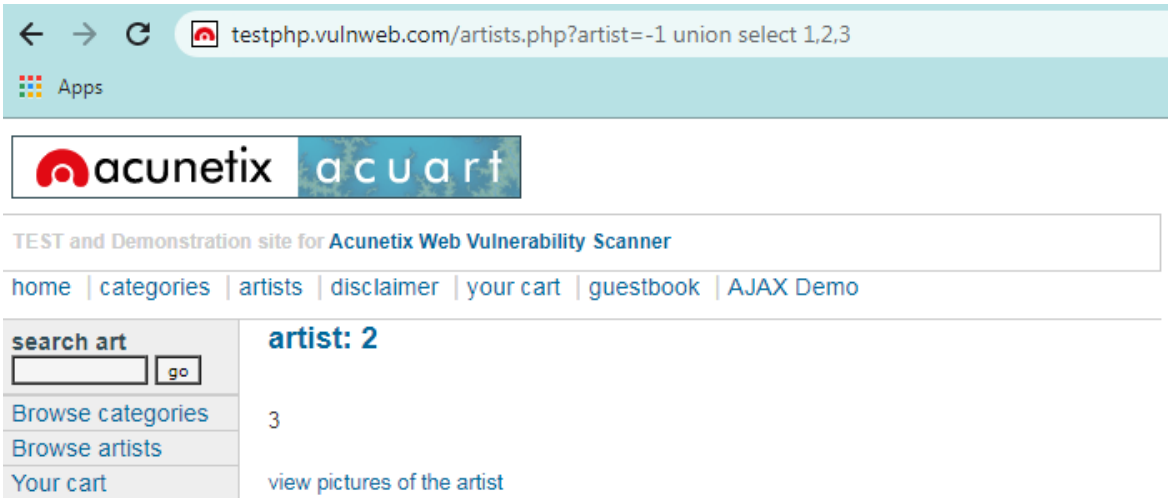
Kada je aplikacija ranjiva na napad, a rezultati upita se prikazuju unutar odgovora aplikacije, operator `UNION` može se koristiti za dohvat podataka iz drugih tablica koje nisu vidljive u bazi podataka. Operator `UNION` često se koristi u napadima gdje se otkrivaju i objavljuju baze podataka. Mora zadovoljiti nekoliko uvjeta, točan broj stupaca koji se koristi u ranjivom upitu i koji stupci vraćeni iz prvobitnog upita će zadovoljit kompatibilnost tipa podataka kada se ubaci zlonamjerni dio upita.

Iz predhodnog primjera je vidljivo da postoje tri stupca u tablici, a možemo logikom pretpostaviti da se radi o znakovnom tipu pošto prikazan tekst sadrži slova. Ovisno o sustavu, u SQL-u se mogu `SELECT` naredbom odabrati konstante bez pozivanja tablice u upitu, ako se to spoji i zadovolji uvjet unije moguće je dobiti ispis upita kroz sučelje ranjivog parametra aplikacije.

```
SELECT * FROM ? WHERE artist=1
UNION
SELECT 1,2,3
```

Vidljivo je da u parametru ‘`artist`’ postoji ranjiva vrijednost, pa se može promijeniti izvorni upit i napraviti neispravnim kako bi se vidio ispis zlonamjernog upita. Negativne vrijednosti rijetko služe kao identifikatori u bazi podataka. Postavi li se vrijednost u poveznici upita na `-1` ili bilo koju drugu negativnu vrijednosti koja nije prisutna u bazi podataka, izvorni upit će tražit podatke koji ne postoje.

```
http://testphp.vulnweb.com/artists.php?artist=-1 union select 1,2,3
```



Slika 3.7 Uspješan napad `UNION` operatorom

Operatorom `UNION` uspješno se spojio zlonamjerni SQL upit s izvornim upitom kojeg treba izvesti web aplikacija. Rezultat unesenog upita spojio se s rezultatom izvornog upita što je omogućilo dobivanje vrijednosti stupaca iz druge ili iste tablice. Vidljivo je da su drugi i treći stupac prikazani u sučelju popularizirani konstantama umetnutog SQL koda. Na ovaj način se saznalo da drugi i treći stupac pružaju ispis iz baze podataka. SQL poslužitelji posjeduje veliki broj ugrađenih funkcija pa se mogu iskoristiti neke od njih da se sazna više informacija.

```
http://testphp.vulnweb.com/artists.php?artist=-1 union select
1,database(),version()
```

Slika 3.8 Ime baze podataka i verzija poslužitelja

3.3. Slijepi SQL napad - (engl. Blind based)

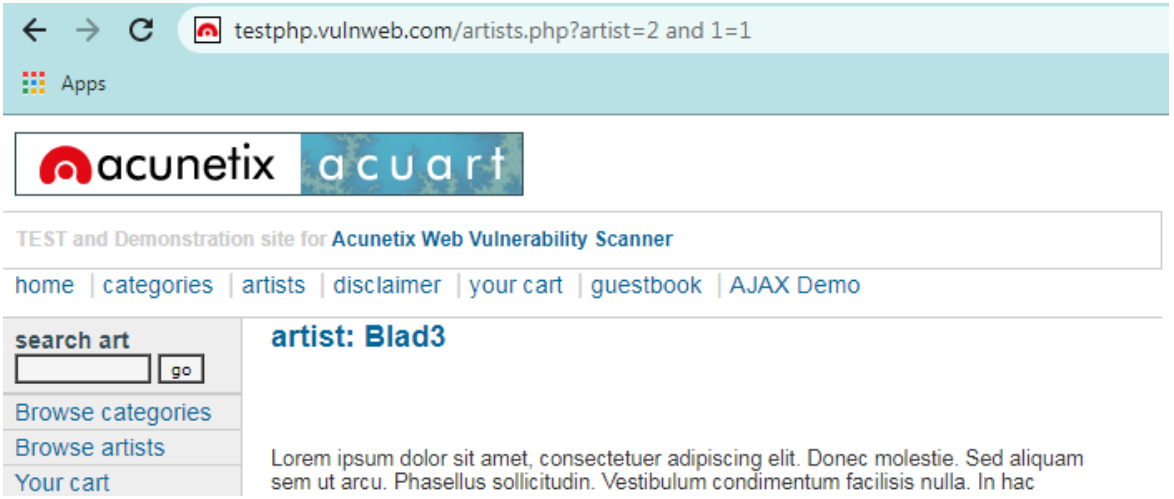
Slijepi napad umetanjem SQL koda vuče ime od interakcije s bazom podataka gdje napadač manipulira bazom bez da izravno vidi rezultate svojih zlonamjernih radnji. Za razliku od napada temeljenih na greškama koja se svodi na dobivanje poruke od aplikacije, slijepi SQL napad ne zahtijeva nikakav vizualni dokaz da se nešto desilo s bazom podataka.

U prijašnjim tipovima napada radili su se upiti s predvidljivim rezultatima logičkog izraza, uvjet je bio uvijek istiniti ili neistiniti. Ovdje se koristi uvjet koji bi mogao biti istiniti i na taj način otkriti neke informacije o bazi podataka bez prikazanih povratnih informacija.

Pomoću logičke varijable zadatak je utvrditi da li se upit može smatrati istinitim (engl. *true*) ili nestinitim (engl. *false*), a dostupne vrijednosti su normalne logičke operacije (`i, ili, nije`). Slanjem logičkog upita prema bazi podataka ranjive aplikacije, napadač može ovisno o ponašanju aplikacije zaključiti da li je moguće izvesti napad umetanjem SQL koda.

Povratni odgovor omogućuje napadaču da procijeni kako se tretiraju logički upiti, tj. vraća li aplikacija istiniti ili neistiniti odgovor na poslan zlonamjerni upit, iako se podaci iz baze podataka ne prikazuju.

```
https://testphp.vulnweb.com/artists.php?artist=2 and 1=1
```

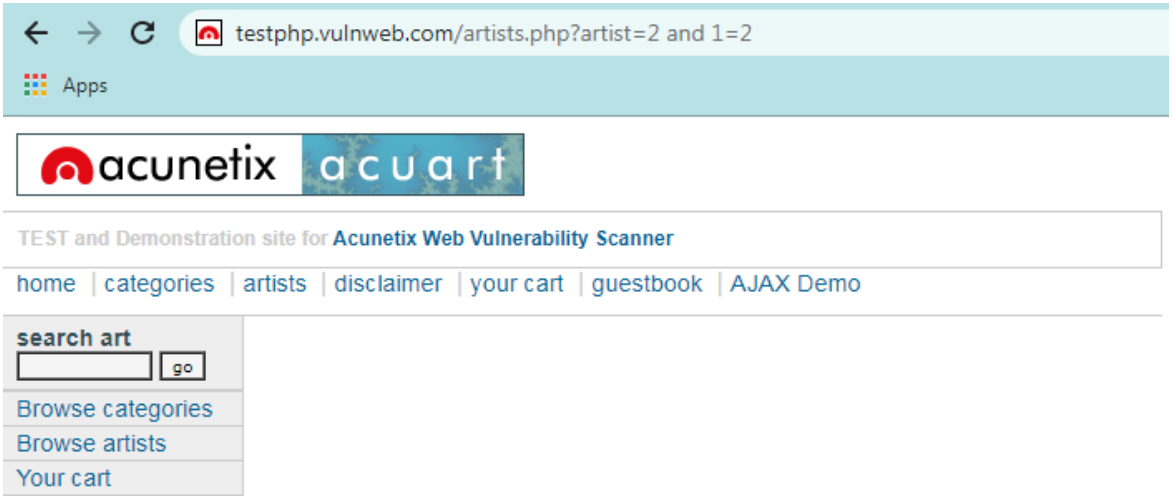


Slika 3.9 Prikaz aplikacije nakon unesenog istinitog upita

Ubaci li se zlonamjerni uvijek istiniti upit `AND 1=1`, a nakon toga kontradiktoran uvijek neistiniti upit `AND 1=2` postoji mogućnost da se vide drugačiji rezultati u odgovoru aplikacije. Ako se sadržaj stranice razlikuje od stranice koja se vratila tijekom neistinitog upita, napadač može zaključiti da napad logičkim izrazima funkcionira..

Kod rezultata baze podataka istiniti upit osigurava da ako se uspješno izvrši, rezultati koji se vraćaju će biti uvijek istiniti pošto je uvjet zadovoljen, dok obrnuto s neistinitim uvjetom vraćeni će biti prazni rezultati. Razliku između istinitih i lažnih povratnih rezultata napadač potvrđuje upotrebom logičkog “ `i` “ operatora, pošto upit vraća istinu ako su svi uvjeti odvojeni `AND` operatorom istiniti.

```
http://testphp.vulnweb.com/artists.php?artist=2 and 1=2
```



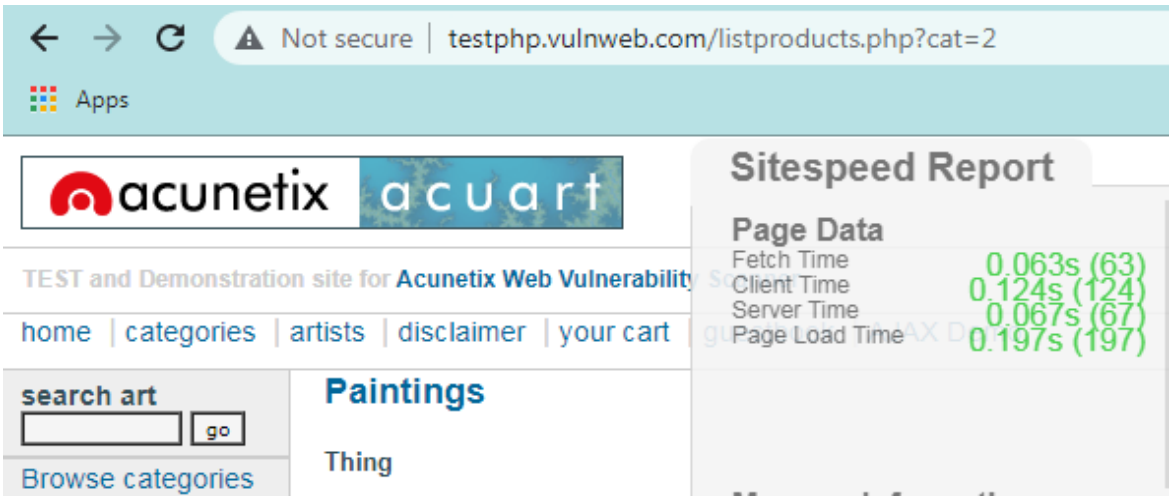
Slika 3.10 Prikaz aplikacije nakon unesenog neistinitog upita

U oba primjera aplikacija se ponašala drugačije. Smatra se sporijim pristupom, ali napadač može s malo truda kroz zlonamjerne upite prisilit bazu podataka da mu aplikacija svojim ponašanjem prilikom testiranja odgovara s da ili ne i na taj način rekonstruira informacije iz baze podataka. Napadač često neće biti te sreće da mu se ispišu detaljne poruke o pogreškama, ponekad je i vraćena prazna stranica indikator da je na aplikaciji prisutna ranjivost.

3.4. Slijepi SQL vremenski napad - (engl. *Time-based blind*)

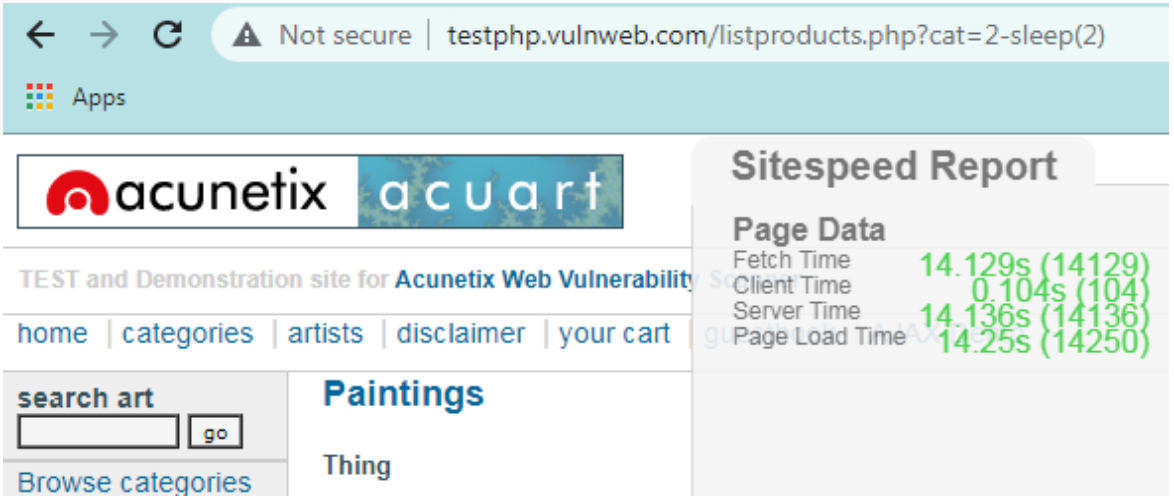
Još jedan način izvođenja koji spada u slijepe SQL napade su takozvani vremenski napadi. Postoje određene varijacije, ali većina sustava za upravljanje bazom podataka podržava funkcije za vremensku manipulaciju upita. Temelji se na slanju SQL upita prema bazi podataka gdje će baza pričekati određeno vrijeme ili kasniti prije nego što odgovori s povratnom informacijom. Vrijeme odziva u odgovoru će otkriti napadaču da li je rezultat zlonamjernog upita istinit ili neistinit jer ovisno o tome je li uvjet potvrđen ili ne, vremensko kašnjenje će se izvršiti, a odgovor će vremenski biti duži od normalnog odziva.

Ponekad se dobiveni rezultati zlonamjernog logičkog upita puno ne razlikuju pa napadač na ovaj način stvara neku dodatnu razliku u odgovoru kojeg dobije iz baze podataka. U MySQL poslužitelju neke od funkcija koje to omogućuju su `SLEEP()` i `BENCHMARK()`.



Slika 3.11 Primjer normalnog vremenskog odaziva web aplikacije

`http://testphp.vulnweb.com/listproducts.php?cat=2-sleep(2)`



Slika 3.12 Vremenski odziv aplikacije nakon umetnutog zlonamjernog koda

U ovom slučaju SQL upit se izvršava sedam puta na strani baze podataka jer je vrijeme odziva uvijek sedam puta veće od ulazne vrijednosti `SLEEP()` izraza, tj. 2 x 7 = 14 sekundi. Stvari mogu biti i kompleksnije, ovisno o umetnutom upitu moguće je sustavno testirat jedan po jedan znak u tablicama baze podataka. Ako bi tražio prvi znak riječi uz specifično pisan vremenski upit, moglo bi se logičkim istinitim ili neistinitim upitom zaključiti po vremenskom odzivu da li je traženo slovo prvo u redoslijedu ili nije.

4. Automatizirani napad umetanjem SQL koda

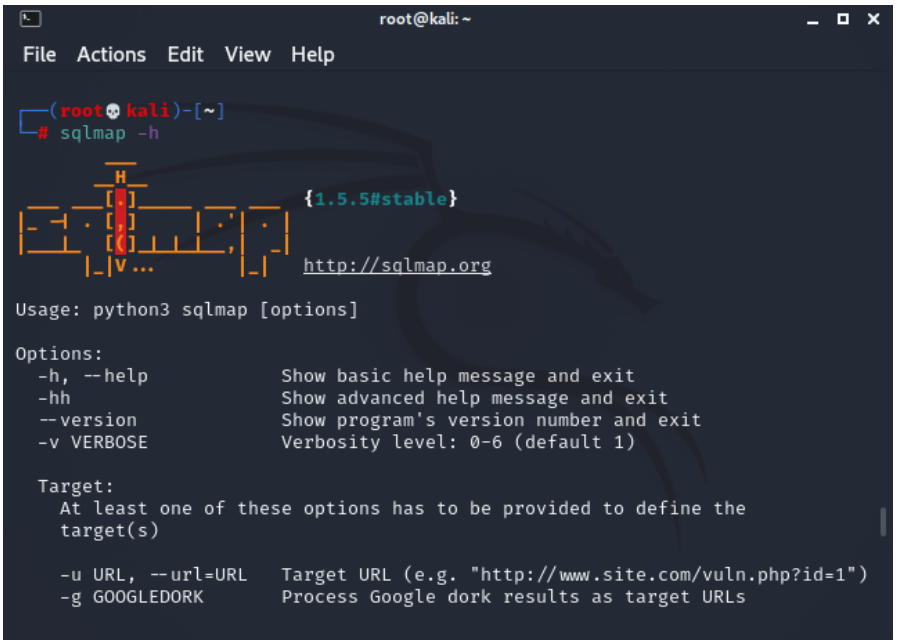
Osim direktnog pristupa, napad umetanjem SQL koda moguće je izvesti kroz specijalizirane softverske alate i skripte koji mogu puno brže odraditi neke od ključnih zadataka potrebnih za uspješnu izvedbu napada. Neki sustavi ili web aplikacije mogu biti poprilično komplicirani pa ručno testiranje zahtijeva visoku razinu stručnosti i sposobnost praćenja velike količine programskog koda, uz to oduzima puno vremena. Ove automatizirane alate ne koriste samo napadači, nego i sigurnosni stručnjaci jer konstantnom optimizacijom pojednostavljuju i pomažu uštedjeti vrijeme u pronalaženju ranjivosti ciljanje aplikacije. Treba napomenuti da čak i ako automatizirani alati uštede vrijeme ne smatraju se uvijek pouzdanima jer postoji mogućnost da nešto preskoče ili zbog tehničkih mogućnosti ne mogu izvesti nešto što se moglo direktnim pristupom.

Danas postoji široki spektar alata za napad umetanjem SQL koda, jedan od neophodnih je SQLmap. SQLmap pretražuje i indeksira stranice na web aplikaciji, traži prozore unosa i obrasce kako bi prema njima slao zlonamjerne SQL upite koji bi mogli uzrokovati grešku u sintaksi sustava. Sadržava različite mogućnosti prilagođavanja koje omogućuju implementaciju različitih tehnika napada pa tako ima funkcionalnosti za dešifriranje raspršene lozinke (engl. *hashing*) ako se dohvati iz baze podataka. Za potrebe testiranja će se također koristiti ranjiva Acunetix[4] web aplikacija.

4.1. SQLmap

SQLmap je jedan od popularnijih i močnijih alata za automatizaciju napada umetanjem SQL koda. Open source, razvijen u Python programskom jeziku i usavršavan kroz godine, SQLmap služi za ispitivanje mogućnosti proboja web aplikacije. [13] Automatizira proces detekcije i ako postoje, iskorištava moguće sigurnosne propuste kroz napad umetanjem SQL koda. Uzme li se kroz HTTP ranjiva poveznica, SQLmap može skenirati, povući podatke iz baze podataka pa i pod određenim uvjetima unosit iste u bazu podataka. Nema grafičko sučelje pa se sve radi upisivanjem naredbi putem konzole. Radi na bilo kojem operativnom sustavu koji ima instaliran Python, a za potrebe ovog rada upotrijebit ćemo SQLmap 1.5.5 inačicu na virtualiziranom okruženju Linux Kali verzije. Linux je sustav otvorenog koda poznat po različitim verzijama svojeg operativnog sustava. Popularan je zbog mogućnosti moduliranja i da odgovara okruženju za koji ga korisnik želi koristiti. Tako je nastala distribucija pod nazivom Kali koja sadrži veliki broj alata za penetracijsko testiranje računalnih mreža i sustava.

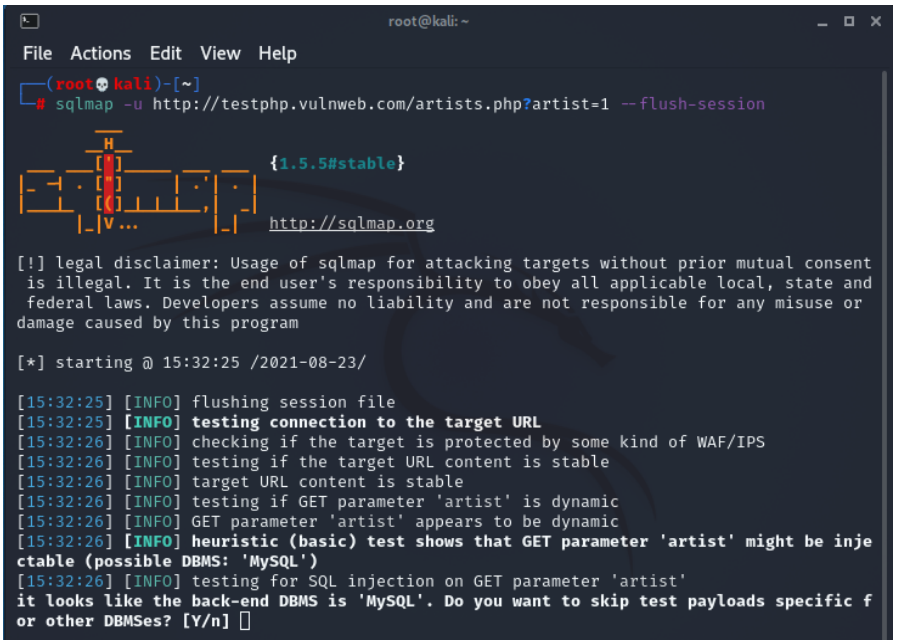
[4] http://testphp.vulnweb.com - ranjiva PHP web aplikacija



Slika 4.1 Prikaz SQLmap konzole

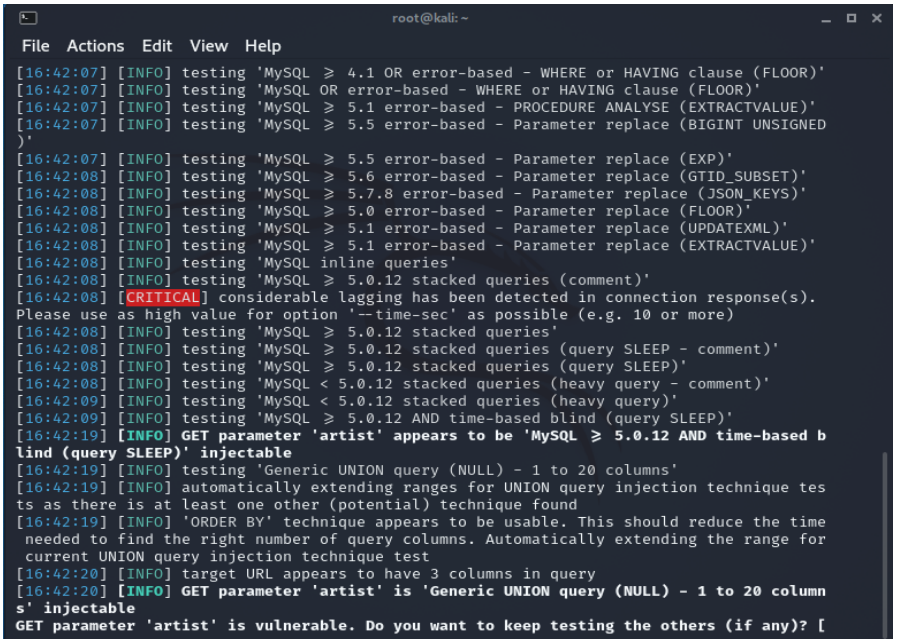
Kroz prijašnje primjere se već zna da je vrijednost u parametru `artist` ranjiva, dinamična je i može se promijeniti s varijablama SQL izraza. U konzoli je dovoljno samo navesti poveznicu koja je opisana kao meta (engl. *target*) `-u` parametrom. Kroz seriju pitanja SQLmap će skenirati i isprobati različite tehnike napada tijekom izvršavanja. Budući da se pretražuju aspekti web aplikacije i ovisno o rezultatima, SQLmap će pitat što točno želimo i koji napadi su mogući ako se pronađu ranjivosti.

`sqlmap -u` `http://testphp.vulnweb.com/artists.php?artist=1`



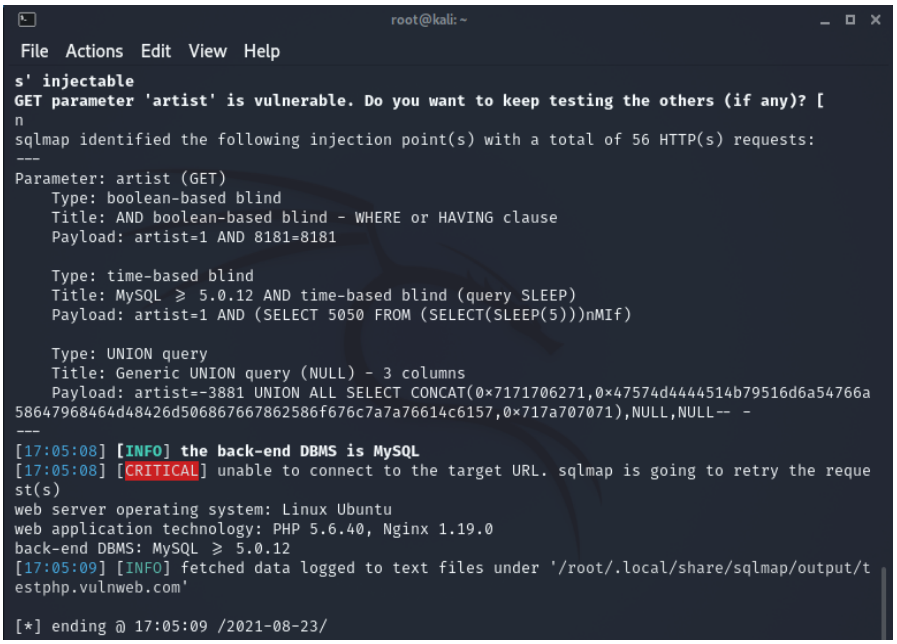
Slika 4.2 SQLmap pokretanje napada

SQLmap podržava napade pomoću HTTP GET metode. GET metode koriste se kada klijent poput preglednika, zahtijeva određene resurse od poslužitelja pomoću HTTP protokola te je ovo uobičajena metoda za dohvaćanje podataka. Sagledamo li poveznice u pregledniku, uvijek počinju upitnikom (?), nakon toga dolazi naziv varijable i vrijednost, odvojene sa znakom jednakosti (=). Ako poveznica sadrži više parametara tada su odvojene sa simbolom ampersand (&).



Slika 4.3 SQLmap automatizirano testiranje parametara

SQLmap je potvrdio da je aplikacija ranjiva na napad umetanjem SQL koda pomoću slijepog napada, vremenskog napada i UNION operatora. Testirao je različite varijacije zlonamjernog koda i detaljno naveo koji SQL upit će koristiti u napadu. Rezultati se automatski spremaju u zapisnik na sistemu za eventualnu buduću analizu.



Slika 4.4 SQLmap rezultati skeniranja

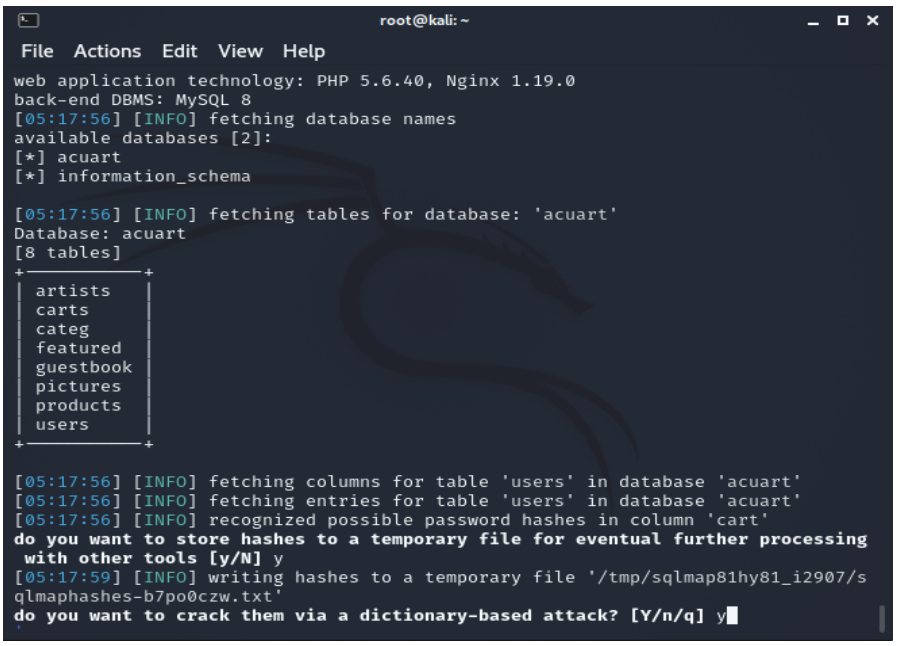
Uz to SQLmap je otkrio da aplikacija radi na operativnom sustavu Linux Ubuntu, web poslužitelj PHP i sustav baze podataka MySQL, zajedno s pripadajućim podacima o inačicama. Sada možemo ići i dublje, sljedeći korak je saznati imena baze podataka i izvući informacije. SQLmap dokumentacija[5] sadrži velik broj podesivih parametara, ali za potrebe rada demonstrirat ćemo samo neke:

Parametar	Opis
`--dbs`	pronalazi imena baze podataka
`--tables`	pronalazi imena tablica
`-dump`	izvuče strukturu i podatke iz baze gdje ih konvertira u jednu datoteku
`-D`	navodi se ime ciljane baze podataka
`-T`	navodi se ime ciljane tablice

Tablica 4.1 Prikazani neki od parametara i opis njihove funkcije

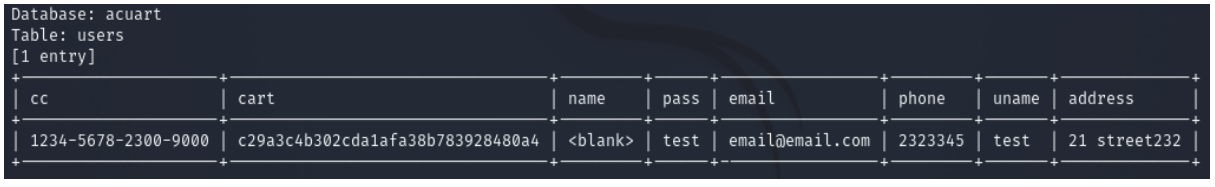
[5] https://usermanual.wiki/Document/sqlmapmanual.1597701365 - SQLmap dokumentacija

```
sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 --dbs
--tables -dump -D acuart -T users
```



Slika 4.5 SQLmap imena baze podataka i tablica

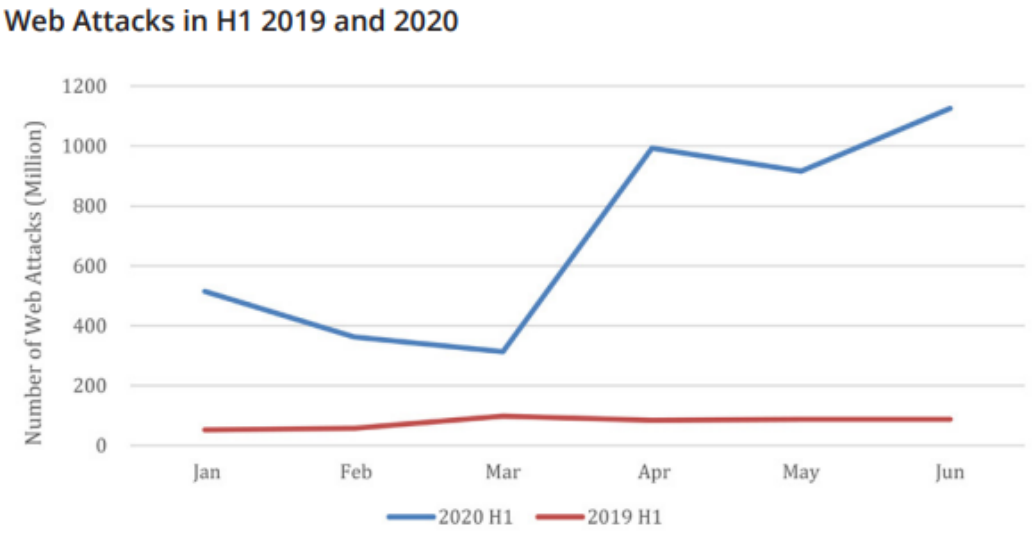
Kroz samo par linija koda SQLmap je uspio prikupiti dosta korisnih informacija iz baze podataka i omogućio gotovo izravan pristup. U stvarnom scenariju napadač bi pokušao steći višu razinu pristupa sustavu. U tu svrhu pokušao bi dešifrirati raspršene lozinke i pokušao se prijaviti kao administrator sustava gdje bi mogao pokretati naredbe izravno na poslužitelju. SQLmap podržava i napad pomoću rječnika (eng. *dictionary based attack*) gdje odma pokušava prepoznati i dešifrirati pronađene raspršene lozinke.



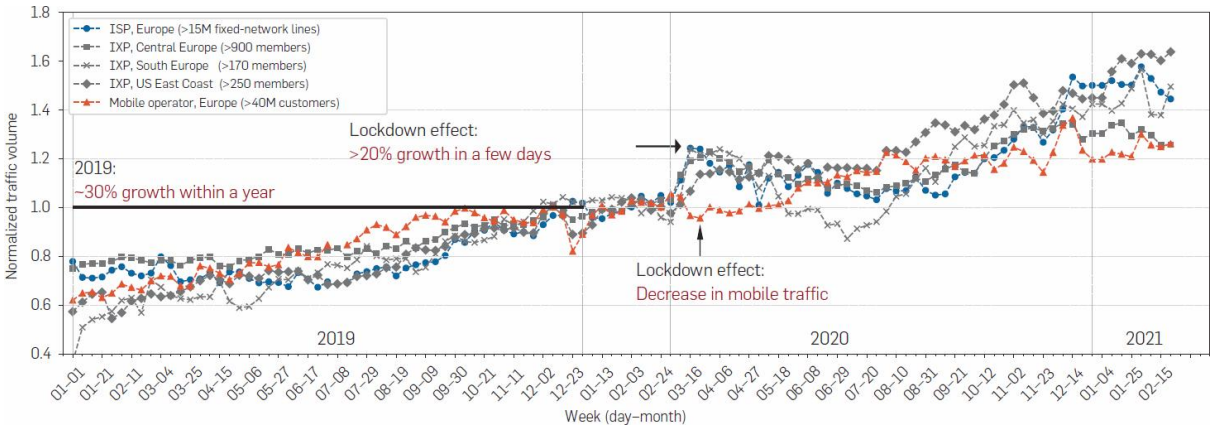
Slika 4.6 SQLmap prikaz dohvaćenih informacija u tablici

5. Analiza

CDNetworks[6] kompanija koja operira na globalnoj razini pružanjem usluga mrežne infrastrukture CDN sadržaja (engl. *Content Delivery Network*) kroz usluge u oblaku (engl. *cloud*) i sigurnost, u studenom 2020. objavljuje izvješće o stanju web sigurnosti za prvu polovicu 2020. godine. [14] U izvješću navode da su se napadi na web aplikacije povećali za više od 800% nego za isto razdoblje 2019. godine. Kao glavni razlog naglog skoka se smatra trenutna COVID pandemija koja je imala ozbiljan utjecaj na porast kibernetičkih napada, što je s jedne strane i logično. Ovaj globalni fenomen zatvorio je stanovništvo svijeta i prisilio nas da za većinu životnih socio-ekonomskih aktivnosti koristimo internet. [15]



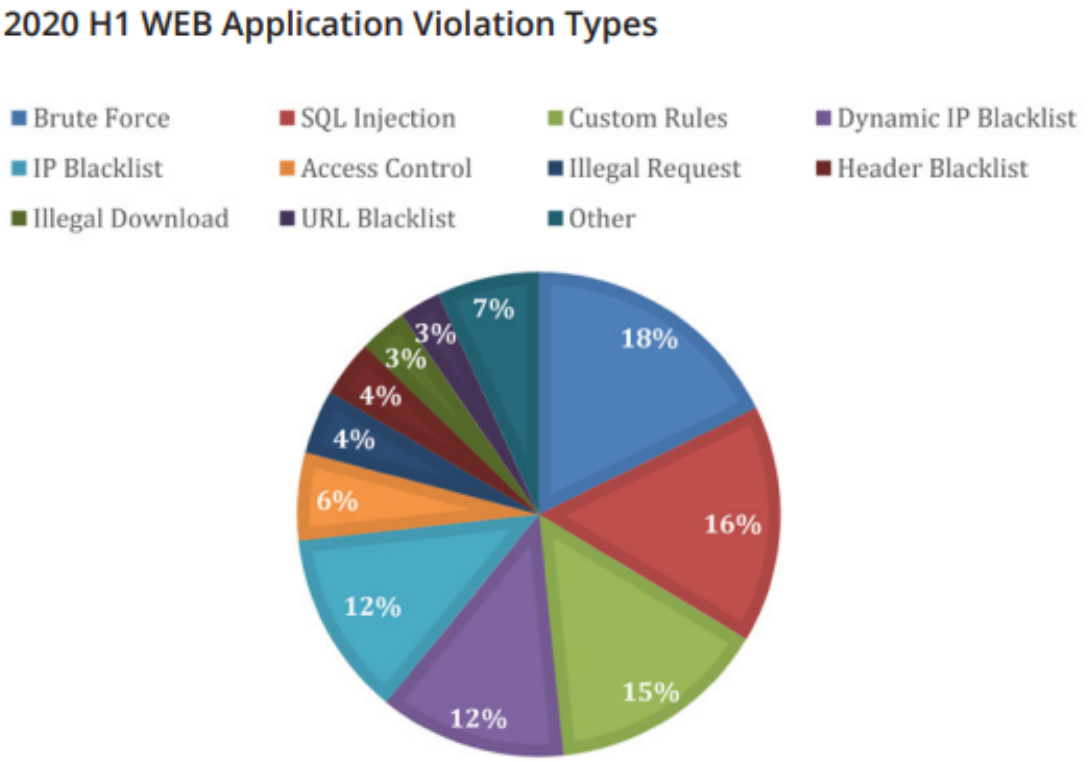
Slika 5.1 Usporedba blokiranih napada na web aplikacije za prvu polovicu 2019. i 2020. godine [14]



Slika 5.2 Mrežni promet Europskog ISP-a, IXP-a i mobilnog operatera u razdoblju COVID pandemije [15]

[6] https://www.cdnetworks.com - CDNetworks web stranica

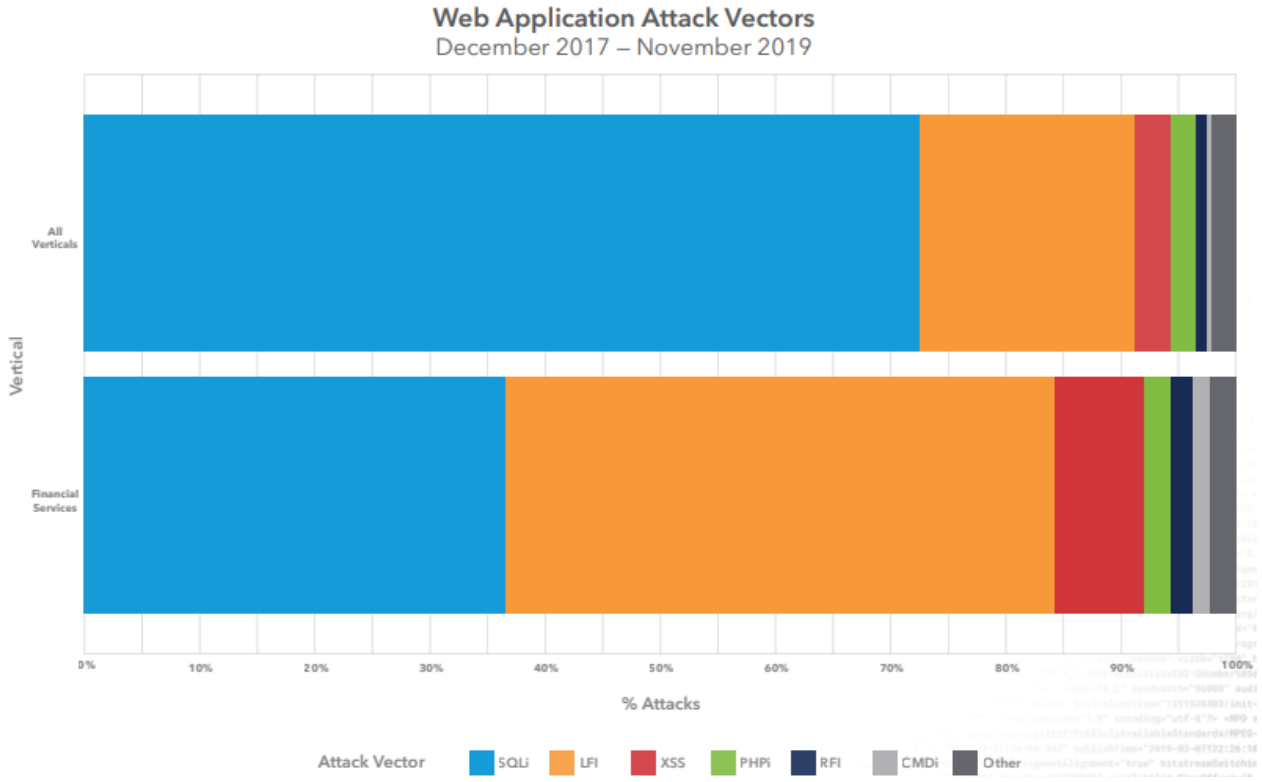
Navodi se da su za prvu polovicu 2020. godine napadi umetanjem SQL koda na web aplikacije zauzeli visokih 16%. Ukazuje da čak i nakon dvadesetak godina postojanja, ova ranjivost je i dalje jedan od omiljenih vektora napada.



Slika 5.3 Tipovi blokiranih napada na web aplikacije [14]

Akamai Technologies[7] kompanija koja se također bavi CDN sadržajem, objavljuje 2020. godine slične podatke. Tijekom razdoblja promatranja od 24 mjeseca, gledajući sve vertikale razdoblja napad umetanjem SQL koda čini više od 72% svih napada na web aplikacije. Stopa napada se prepolovi na 36% ako se gledaju napadi koji ciljaju samo sektor pružanja financijskih usluga. [16]

[7] https://www.akamai.com - Akamai web stranica



Slika 5.4 Pregled vrste napada na web aplikacije unutar 24 mjeseca, od prosinca 2017. do studenog 2019. [16]

5.1. Prevencija

Napad umetanjem SQL koda danas je već dobro poznata ranjivost i obično se uzima u obzir još kada je aplikacija u stanju razvoja, ali u većini slučajeva aplikacija vuče ranjivost još od razvojne faze i dizajna. Programeri često obično ne razmatraju sigurnosne aspekte u aplikacijama na kojima rade jer veću važnost posvećuju funkcionalnostima aplikacije. Takva praksa ne dovodi samo do ranjivosti kao što je napad umetanjem SQL koda nego stvara i dodatne probleme koji se uglavnom rješavaju softverskim zakrpama (engl. *patching*).

Postoji nekoliko načina prevencije, a jedan od njih je kroz pripremljene upite (engl. *prepared statements*). Kada se govori o pripremljenim upitima to znači da se SQL kod uključen u upit prethodno definira, što znači da će baza podataka razlikovati kod i korisnički unos. Ako napadač pokušava unijeti zlonamjerni SQL izraz, baza podataka će dobiveni upit tretirati kao podatak, a ne kao kod. Ovaj pristup će preventirati da se dobiven upit pretvori u zlonamjeran. Kod pripreme upita držimo se nekoliko pravila:

- priprema SQL upita s praznim vrijednostima kao rezerviranim mjestima gdje upitnik predstavlja vrijednost
- vezanje varijable s rezerviranim mjestima, navodeći svaku varijablu zajedno s tipom podatka. Dopušteni tipovi podataka:

  i - cijeli brojevi (engl. *integer*)

  d - decimalni broj s pomičnim zarezom (engl. *double*)

  s - niz znakova (engl. *string*)

  b - binarne podatke (engl. *blob - binary large object*)

Primjer pripremljenog MySQL upita u PHP-u:

```
$stmt = $conn->prepare("INSERT INTO Student (ime, prezime, email)
VALUES (?, ?, ?)");
$stmt->bind_param("sss", $ime, $prezime, $email);
```

U osnovi radi se predložak kako će izgledati SQL upit. Funkcija `bind_param()` omogućit će da se varijabla veže na rezervirana mjesta lažne vrijednosti `VALUES (?, ?, ?)` u pripremljenom predlošku. Ovo sprečava bilo kakvu mogućnost ubacivanja zlonamjernog SQL koda. Nakon toga parametar "`sss`" govori bazi podataka tip varijable i navodi da će vrijednost `$ime, $prezime, $email` biti niz znakova.

Još jedan način prevencije je kroz pohranjene procedure gdje je princip sličan kao i kod pripremljenih upita. Razlika je u tome gdje se SQL kod nalazi, dok su pripremljeni upiti u logičkom sloju aplikacije, pohranjene procedure se nalaze u podatkovnom sloju kojeg poziva aplikacija. To je su skup uputa koje izvršavaju predodređen zadatak, a njihova prednost je što može centralizirati logiku pristupa podacima na jedno mjesto i promijeniti dozvole pristupa. Svodi se na načelo najmanje privilegija, što znači da ako napadač ugrozi pohranjenu proceduru koja sadrži SQL naredbe, šteta bi bila ograničena zbog strožeg pristupa podacima. Ovisno o aplikaciji u slučajevima gdje neki SQL kod postane nužan dio korisničkog unosa, bitno je napraviti popis dopuštenih SQL izraza. Izrada popisa samo najnužnijih izraza služi za filtriranje nevažećih izraza, tako da ne završe u poslanom upitu.

Dobra praksa je razmišljati o zaštiti u početnom stadiju razvoja kada se piše kod koji pristupa bazi podataka, a to je najbolje vrijeme za sprečavanje nastanka ranjivosti umjesto da ih kasnije rješava kroz softverske zakrpe. Danas postoje različiti alati koji analiziraju programski kod (engl. *linters*) i pružaju smjernice za najbolje prakse u programiranju, poštivanjem svih standarda kodiranja, a također mogu prepoznati i potencijalne sigurnosne probleme. Proces razvoja bi trebao uključivati testove u različitim oblicima protiv ovog napada jer svakome je u interesu da smanji rizik od budućih posljedica.

6. Zaključak

U ovom radu je kroz praktični dio, koristeći ranjivu web aplikaciju, istraženo nekoliko tehnika napada umetanjem SQL koda. Unatoč tome što je starija i poznata ranjivost, pokazano je da je još uvijek prisutna i danas, naglašavajući važnost primjene preventivnih mjera. Zbog interneta živimo u zlatnom dobu sakupljanja i pohrane ogromne količine podataka, a današnje baze podataka omogućuju tvrtkama da ih analiziraju kako bi poboljšale svoje poslovanje, profit ili da bolje razumiju sklonosti i potrebe svojih klijenata. Njihova sigurnost je poprimila mnogo oblika jer posljedice ranjivosti mogu biti katastrofalne.

Ručnim pristupom prikazan je potencijal napada i način testiranja za ovu ranjivost. Demonstrirano je da SQL kod nije zamišljen za direktnu interakciju s aplikacijama, umjesto toga aplikacija je ta koja s obzirom na primljeni unos priprema SQL kod koji je potrebno poslati prema bazi podataka. Kroz rad je prikazan i softverski alat SQLmap koji pomaže u provjeri ranjivosti aplikacije na automatiziran način, čime se štedi vrijeme tijekom testiranja. Automatizirani alati nisu zamjena za ručno testiranje, ali obično su dovoljni da se utvrdi je li aplikacija ranjiva ili ne.

U smislu obrane od napada umetanjem SQL koda radi se o provjeri i primjeni načina za kontrolu ulaza i izlaza informacija u aplikaciji. Napad je osnovan na slanju zlonamjernog SQL upita i kao što je kroz rad prikazano princip je uvijek isti. Unos koji dolazi od korisnika treba smatrati potencijalno zlonamjernim i tek tada se može razmišljati o sigurnosnim mjerama koje se mogu implementirati.

7. Literatura

[1] Jeff Forristal, rfp, (1998), Phrack Magazine, NT Web Technology Vulnerabilities, http://phrack.org/issues/54/8.html, 20. svibanj 2021.

[2] Bill Gates, (2002), Global Information Assurance Certification, Microsoft toward Trustworthy Computing, https://www.giac.org/paper/gsec/4243/pillars-trustworthy-computing-displayed-patch-management/106837, 20. svibanj 2021.

[3] W. Halfond i A. Orso, (2005), College of Computing Georgia, AMNESIA, https://www.cc.gatech.edu/home/orso/papers/halfond.orso.ICSEDEMO06.pdf, 20. svibanj 2021.

[4] OWASP, (2017), nonprofit organization, OWASP Top 10, https://owasp.org/www-project-top-ten/, 20. svibanj 2021.

[5] Reuters, (2009), Daniel Trotta, Three indicted in largest U.S. identity theft scheme, US Department of Justice, https://www.reuters.com/article/us-crime-identity-idUSTRE57G4GC20090817, 20. svibanj 2021.

[6] NY Times, Jolie o'Dell, (2009), RockYou Hacker: 30% of Sites Store Plain Text Passwords, RockYou breach, https://archive.nytimes.com/www.nytimes.com/external/readwriteweb/2009/12/16/16readwriteweb-rockyou-hacker-30-of-sites-store-plain-text-13200.htm, 20. svibanj 2021.

[7] Pastebin, TinKode@Slacker.Ro & Ne0hDell, (2011), MySQL breach, https://pastebin.com/raw/BayvYdcP , 20. svibanj 2021.

[8] BBC, (2011), LulzSecurity, Sony investigating another hack, https://www.bbc.com/news/business-13636704 , 20. svibanj 2021.

[9] CNN, Doug Gross, (2012), Yahoo hacked 450,000 passwords posted online, https://edition.cnn.com/2012/07/12/tech/web/yahoo-users-hacked/index.html , 20. svibanj 2021.

[10] Vtech, (2015), Data Breach on VTech Learning Lodge, https://www.vtech.com/en/press_release/2015/data-breach-on-vtech-learning-lodge-update/ , 20. svibanj 2021.

[11] Reuters, Joseph Menn, (2016) US Election Assistance Commission breach, https://www.reuters.com/article/us-election-hack-commission-idUSKBN1442VC , 20. svibanj 2021.

[12] Gatech, (2019), Unauthorized Access on Georgia Tech Network Exposes Information for 1.3 Million Individuals, Georgia Tech breach https://www.news.gatech.edu/news/2019/04/02/unauthorized-access-georgia-tech-network-exposes-information-13-million-individuals , 20. svibanj 2021.

[13] GitHub, Miroslav Stampar, (2021), SQLmap History, https://github-wiki-see.page/m/sqlmapproject/sqlmap/wiki/History, 15. kolovoz 2021.

[14] CDNetworks, (2020), State of Web Security H1 2020, https://www.cdnetworks.com/wp-content/uploads/2020/11/CDNetworks-State-Of-web-Security-H1-2020.pdf , 21. kolovoz 2021.

[15] Communications of the Acm, Anja Feldmann, Oliver Gasser, Franziska Lichtblau, Enric Pujol, Ingmar Poese, Christoph Dietzel, Daniel Wagner, Matthias Wichtlhuber, Juan Tapiador, Narseo Vallina-Rodriguez, Oliver Hohlfeld, Georgios Smaragdakis, (2021), A Year in Lockdown: How the Waves of COVID-19 Impact Internet Traffic, https://cacm.acm.org/magazines/2021/7/253468-a-year-in-lockdown/fulltext , 21. kolovoz 2021.

[16] Akamai, (2020), State of the Internet, Financial Services Hostile Takeover Attempts https://www.akamai.com/content/dam/site/en/documents/state-of-the-internet/soti-security-financial-services-hostile-takeover-attempts-report-2020.pdf , 21. kolovoz 2021.

Popis slika

Popis tablica

Sveučilište Sjever

IZJAVA O AUTORSTVU
I
SUGLASNOST ZA JAVNU OBJAVU

Završni/diplomski rad isključivo je autorsko djelo studenta koji je isti izradio te student odgovara za istinitost, izvornost i ispravnost teksta rada. U radu se ne smiju koristiti dijelovi tuđih radova (knjiga, članaka, doktorskih disertacija, magistarskih radova, izvora s interneta, i drugih izvora) bez navođenja izvora i autora navedenih radova. Svi dijelovi tuđih radova moraju biti pravilno navedeni i citirani. Dijelovi tuđih radova koji nisu pravilno citirani, smatraju se plagijatom, odnosno nezakonitim prisvajanjem tuđeg znanstvenog ili stručnoga rada. Sukladno navedenom studenti su dužni potpisati izjavu o autorstvu rada.

Ja, NENAD BEDEKOVIĆ *(ime i prezime)* pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor/~~ica~~ završnog/~~diplomskog~~ *(obrisati nepotrebno)* rada pod naslovom NAPADI UMETANJEM SQL PROGRAMSKOG KODA *(upisati naslov)* te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.

Student/~~ica~~:
(upisati ime i prezime)
Bedeković Nenad
(vlastoručni potpis)

Sukladno Zakonu o znanstvenoj djelatnost i visokom obrazovanju završne/diplomske radove sveučilišta su dužna trajno objaviti na javnoj internetskoj bazi sveučilišne knjižnice u sastavu sveučilišta te kopirati u javnu internetsku bazu završnih/diplomskih radova Nacionalne i sveučilišne knjižnice. Završni radovi istovrsnih umjetničkih studija koji se realiziraju kroz umjetnička ostvarenja objavljuju se na odgovarajući način.

Ja, NENAD BEDEKOVIĆ *(ime i prezime)* neopozivo izjavljujem da sam suglasan/~~na~~ s javnom objavom završnog/~~diplomskog~~ *(obrisati nepotrebno)* rada pod naslovom NAPADI UMETANJEM SQL PROGRAMSKOG KODA *(upisati naslov)* čiji sam autor/ica.

Student/~~ica~~:
(upisati ime i prezime)
Bedeković Nenad
(vlastoručni potpis)