Zahvaljujem se doc. dr. sc. Stjepanu Grošu i asistentu
Karlu Slovenecu na stručnom vodstvu i pomoći
u izradi ovog završnog rada

Sadržaj

1. Uvod

Sa sve većom uporabom računarstva [1], napadači pokušavaju na razne načine provaliti na poslužitelje baze podataka. Kako bi se to onemogućilo javlja se potreba uvođenja odgovarajućih zaštita.

Velika količina podaka se sprema u baze podataka te ih je zbog toga potrebno pravilno zaštititi [2]. Tema ovog rada su slabosti kriptografske zaštite u sustavu za upravljanje bazom podataka pod nazivom MariaDB [3]. Cilj je istražiti koje su mogućnosti MariaDB, koje prednosti i slabosti ima kriptografska zaštita podržana u MariaDB. U ovom radu je prikazano što napadač može učiniti ako uspije dobiti podatke koji su šifrirani.

Rad se sastoji od tri poglavlja. U prvom poglavlju se opisuje MariaDB sustav za upravljanje bazom podataka. Ukratko se prikazuje arhitektura te način korištenja baze podataka. Osim toga, prikazano je i na koji način se šifriraju podaci u MariaDB. Objašnjen je AES (*Advanced Encryption Standard*) [4] algoritam te njegove slabosti i nedostaci koji se pojavljuju u MariaDB. Nadalje, objašnjeno je na koji način koristiti zaštitu podataka pomoću sustava za pohranu podataka u bazi podataka (engl. *Storage engine*) pod nazivom InnoDB.

Nadalje, opisan je model prijetnje te je objašnjeno kako pravilno definirati model prijetnje za konkretan napad. Kako je riječ o bazama podataka, definirana su dva modela prijetnje.

U drugom poglavlju je definiran model prijetnje napadača na bazi podataka. Pretpostavljeno je da je napadač provalio na bazu podataka koja sadrži podatke kriptirane AES algoritmom te kroz primjere je pokazano što napadač može saznati te kako može kompromitirati bazu podataka koju je napao.

U trećem poglavlju definiran je model prijetnje napadača na aplikaciji. Prikazano je kako napadač može kroz aplikaciju doznati podatke iz baze podataka. Objašnjen je pojam SQL-injekcije [5] te kako se najbolje zaštititi.

2. MariaDB baza podataka

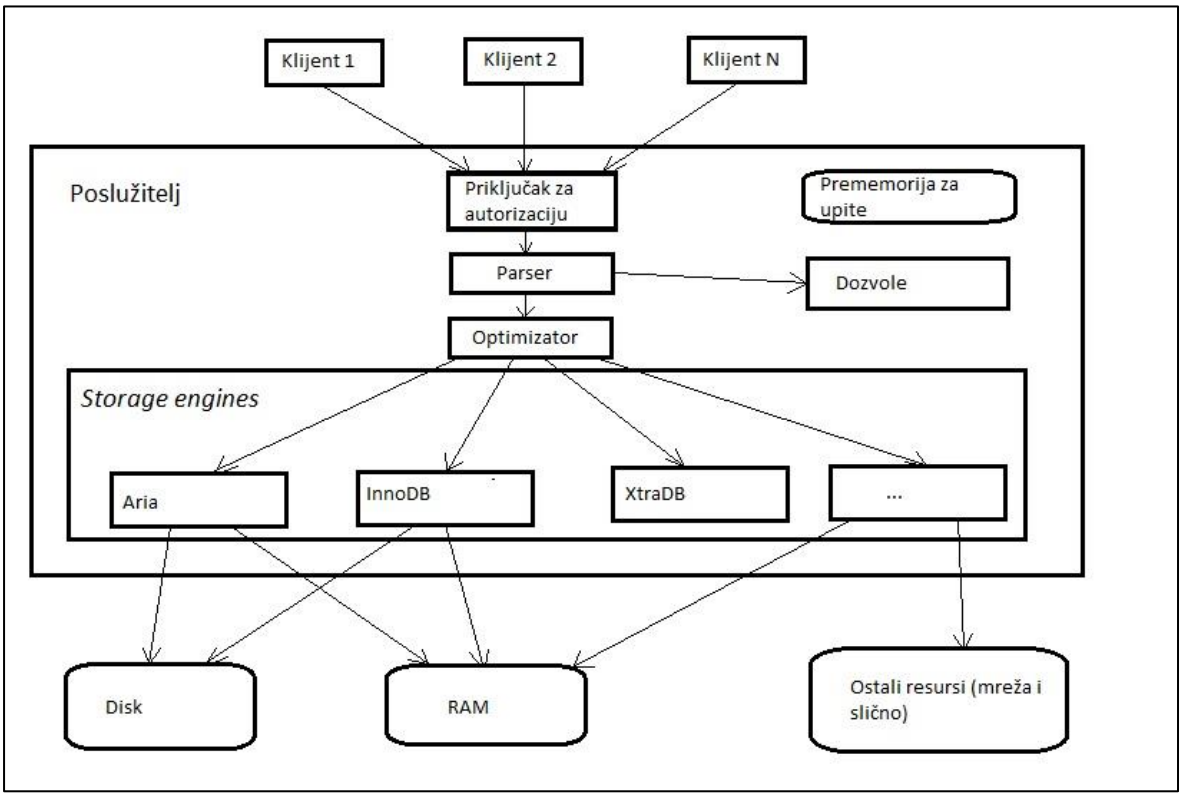
Baza podataka je skup podataka koji su pohranjeni i organizirani tako da mogu zadovoljiti zahtjeve korisnika [6]. MariaDB je jedan od sustava za upravljanje bazom podataka (engl. *Database Managment System*, DBMS) koji je besplatan za korištenje i otvorenog je koda (engl. *Open source*) [7].

2.1. Arhitektura MariaDB baze podataka

MariaDB je nastala iz izvornog koda koji se koristio za MySQL [8], 2008. godine. U početku rada na MariaDB DBMS-u, većina funkcionalnosti je bila ista kao i u MySQL-u, ali s godinama su se funkcionalnosti počele razlikovati[9]. Jedna od funkcionalnosti koja je svojstvena MariaDB je mogućnost korištenja različitih priključaka [10]. Na taj način se DBMS može prilagoditi zahtjevima korisnika, a to ostvaruje instaliranjem odgovarajućih priključaka. Jedni od bitnijih priključaka su priključci za upravljanje spremanjem podataka na disk (engl. *Storage engine*). Pretpostavljeni su InnoDB [11] te Aria [12], a osim njih se mogu koristiti MyISAM te XtraDB [13]. InnoDB se koristi za spremanje podataka na disk, a Aria se koristi za stvaranje sustavskih tablica te privremenih tablica. Jedan od dodatnih priključaka je, primjerice, *userstat* [14] koji prikazuje statističke podatke o resursima i upotrebama tablica. Osim pretpostavljenih priključaka postoje i dodatni, a kako bi se dodali novi priključci koristi se naredba `INSTALL PLUGIN`.

Svi priključci se koriste uz model klijent-poslužitelj. Za rad s klijentima se koristi tzv. *bazen dretvi* (engl. *Thread pool*). To znači da se svakom klijentu koji se spoji na poslužitelj baze podataka dodijeli jedna dretva koja je prethodno bila u stanju čekanja. Na taj način se postiže konkurentnost, tj. mogućnost posluživanja više klijenata istovremeno. Rad jedne dretve, te način na koji se izvode i vraćaju upiti, prikazan je na slici 2.1. Prvo se klijent spaja na poslužitelj te mu se dodjeljuje dretva. Nakon toga se izvodi autentifikacija klijenta na temelju njegovog imena računala (engl. *hostname*), korisničkog imena i lozinke. Autentifikacija se može, ali i ne mora, delegirati nekom od dostupnih priključaka [15]. Ako je autentifikacija uspješna, klijent šalje SQL upite poslužitelju. Upiti prolaze kroz parser

koji prepoznaje SQL sintaksu te provjerava ima li klijent dozvole za zatraženi upit. Ako nema, klijentu se vraća poruka o nedozovoljenoj radnji. Ako klijent ima dozvole, te ako je uključena opcija za provjeru postojećih upita, provjerava se postoji li već sličan upit u predmemoriji za upite (engl. *Query cache*) [16]. Ako odgovor postoji, odmah se vraća klijentu. Međutim, ako upit ne postoji u predmemoriji za upite, optimizator će pokušati naći najbržu strategiju za izvršavanje upita. Pri tome će optimizator izabrati najpovoljniji priključak za spremanje podataka na disk i pomoću njega pristupiti memoriji i izvesti klijentov upit.



Slika 2.1. Izvođenje upita uz model klijent-poslužitelj

Svi upiti koji su izvedeni se spremaju u dnevnik upita (engl. *query log*). Na taj način se mogu pratiti promjene koje se mogu dogoditi u bazi podataka. Većina *log-ova* je opcionalna [17], ali postoje i oni koji su administrativno potrebni, primjerice, *binary log*. On čuva rezervu baze podataka (engl. *backup*) u slučaju da se dogodi greška koja bi mogla kompromitirati cijelu bazu podataka.

2.2. Način korištenja MariaDB baze podataka

Jednom kada je baza podataka instalirana i podešena, može se početi s upotrebom iste. Kada se korisnik spoji na MariaDB bazu podataka nije automatski spojen na neku od baza podataka, nego mu se nudi mogućnost da izabere jednu od ponuđenih baza podataka za koje ima određena dopuštenja. Potrebno je otvoriti komandnu liniju (engl. *Command Line*), *Powershell* ili nešto slično i upisati naredbu:

```
mysql –u username –p
```

Nakon toga je potrebno upisati lozinku koja je bila postavljena prilikom instalacije.

Ako se želi vidjeti koje baze podataka su trenutno dostupne, dovoljno je upisati naredbu `show databases;`. Ako do sada nije napravljena niti jedna korisnička baza podataka, prikazat će se baze podataka koje su tvornički dostupne nakon instalacije. Nakon toga se može izabrati baza podataka naredbom `use ime_baze_podataka;`. Pokretanje MariaDB te odabir željene baze podataka prikazan je u ispisu 2.1.

```
C:Users/Windows 10> mysql –u root –p lozinka
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 3
Server version: 10.5.3-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.

MariaDB [(none)]> use ime_baze_podataka
Database changed
MariaDB [ime_baze_podataka]>
```

Ispis 2.1. Način pristupa MariaDB bazi podataka i odabir željene baze podataka

Prikaz dostupnih tablica u odabranoj bazi podataka se dobiva naredbom `show tables;`. Rad sa odabranom tablicom ili tablicama radi se pomoću standardne SQL sintakse.

Kao i većina DBMS, tako se i MariaDB može koristiti uz programske jezike za izgradnju aplikacija. U kôdu 2.1. je prikazano na koji način se može spojiti na bazu podataka u Python-u [18] te izvesti ubacivanje podataka u bazu podataka. U kôdu 2.2. je prikazano na koji način je moguće izvršiti čitanje podataka iz baze podataka.

```
1    import mysql.connector as mariadb
2    mariadb_connection = mariadb.connect(user='some_user', /
     password='some_pswd',database='database_name') # spajanje na bazu
3    cursor = mariadb_connection.cursor()
4    cursor.execute("INSERT INTO tablename(data) /
     VALUES(AES_ENCRYPT('text', 'this is a key123'))")
5    mariadb_connection.commit() # potvrda umetanja podatka
```

Kôd 2.1. Spajanje baze podataka s Python-om te ubacivanje podataka u bazu podataka

```
1    import mysql.connector as mariadb
2    mariadb_connection = mariadb.connect(user='some_user', /
     password='some_pswd',database='database_name') # spajanje na bazu
3    cursor = mariadb_connection.cursor()
4    cursor.execute("SELECT data FROM tablename")
5    for data in cursor:
6          print(data)
```

Kôd 2.2. Spajanje baze podataka s Python-om te čitanje podataka iz baze podataka

Kako bi se spojilo na MariaDB bazu podataka, potreban je *mysql.connector*. Ako nije prethodno instaliran, može se dohvatiti izvršavanjem sljedeće naredbe u terminalu:

```
pip install mysql-connector-python
```

Nakon toga je potrebno dodati korisničko ime, lozinku i ime baze podataka na koju se spaja. Kako bi se mogla uspostaviti interakcija s bazom podataka, tj. izvršavanje upita, potrebno je dodati `cursor`. Za pravilno izvođenje upita pomoću programskoj jezika, koristi

se naredba `cursor.execute("query")`. Nakon obavljanja upita za spremanje podataka u bazu podataka potrebno je pozvati metodu `commit()`.

2.3. Slabosti MariaDB i kako se zaštititi

Podaci na tvrdom disku su vidljivi u obliku čistog teksta. Napadač ne mora znati koja je lozinka za pristup MariaDB sustavu upravljanja bazom podataka. Razni su načini na koje napadač može pristupiti računalu, a jednom kad to uspije, dovoljno je da se pozicionira u datoteku u kojoj je instalirana MariaDB. Put do datoteke, na primjer na operacijskom sustavu Windows 10, može biti: „*C:\Program Files\MariaDB 10.5*“ . Primjer čitanja podataka iz korisničke baze podataka koristeći *Powershell* je prikazan u ispisu 2.2. Unutar direktorija, koji ima isto ime kao baza podataka, se nalaze datoteke iz kojih je moguće iščitati podatke iz baze podataka. Te datoteke su *Format* i *InnoDB* datoteke (nastavci *.frm* i *.ibd*) te za svaku tablicu u bazi podataka one postoje. Iz *Format* datoteka se može iščitati koji su atributi odabrane tablice, a iz *InnoDB* datoteka se mogu iščitati koje su vrijednosti tih atributa. Tako dobiveni podaci nisu uredni, to jest sadrže podatke koji nisu čitljivi, ali se iz njih može pročitati ono što napadača zanima.

```
C:\Program Files\MariaDB 10.5\data\ime_baze_podataka> cat .\tablica.frm
      PRIMARY˙
      InnoDB                                                  ‰         ‰
      7              3     %                                            P
      ˙ID˙ime˙prezime˙placa
C:\Program Files\MariaDB 10.5\data\ime_baze_podataka> cat .\tablica.ibd
      ÖiŇ·····················································
      µÚź "□          ·····          »NEż                    ¦€        €
      ň                                    2                  infimum
        supremum
        ˙đ€         €      MarijoCvitanovic10000
```

Ispis 2.2. Ispis načina na koji možemo pročitati podatke baze podataka bez direktnog pristupa bazi podataka

Za zaštitu podataka na tvrdom disku se može iskoristiti navedeni *InnoDB* ili podatke koje unosimo u bazu podataka možemo kriptirati algoritmima za kriptiranje. Dva najpoznatija algoritma za kriptiranje su AES (*Advanced Encryption Standard*) i DES (*Data Encryption Standard*).
DES se nikako ne preporuča za korištenje pri kriptiranju podataka [19]. Stoga kriptiranje podataka na disku je najbolje napraviti pomoću *InnoDB* ili AES algoritmom. Oba načina zaštite su opisana u kasnijim poglavljima.

2.4. Načini šifriranja u MariaDB

Podatke, koji se nalaze na disku ili u bazi podataka, moguće je zaštititi uporabom AES algoritma. AES algoritmi za šifriranje se mogu podijeliti prema načinu (engl. *Mode*) na koji šifriraju podatke. Dva najpoznatija načina su CBC (*Cipher Block Chaining*) Mode i ECB (*Electronic Code Book*) Mode. Osim njih postoje još i CFB (*Cipher Feedback*), OFB (*Output Feedback*) i CTR (*Counter*) [20]. Od svih navedenih, u MariaDB bazi podataka je implementiran samo ECB.

Za šifriranje je potreban podatak koji će se šifrirati i ključ kojim će se šifrirati podatak. Podatak koji će se šifrirati u MariaDB mora isključivo biti tekstualna vrijednost (engl. *string*), a ključ mora biti *hash* vrijednost, tj. 128-bitna vrijednost koja se može, po potrebi, povećati i na 256-bitnu vrijednost. Povećanje na 256-bitnu vrijednost se može postići modificiranjem tekstualne vrijednosti. To se može učiniti tako da se doda punjenje (engl. *Padding*) prije tekstualnog podatka ili nakon njega. Šifriranje se obavlja prilikom dodavanja podatka u bazu podataka.

Prototip funkcije koja implementira AES u MariaDB:

```
AES_ENCRYPT(str, key)
```

Na ispisu 2.3. se može vidjeti primjer šifriranja s navedenom funkcijom. Parametar '`text`' predstavlja vrijednost koja će se šifrirati, a parametar '`key`' predstavlja ključ koji se koristi za šifriranje.

```
MariaDB [(database)]> INSERT INTO table VALUES (AES_ENCRYPT('text',
SHA2('key', 256)));
```

Ispis 2.3. Primjer korištenja funkcije AES_ENCRYPT

Funkcija SHA2 (*Secure Hash Algorithm 2*) [21] je funkcija za izračun sažetka (engl. *hash function*), a to znači da bilo koji tekstualni podatak pretvori u *hash* vrijednost.

Prototip funkcije SHA2 je:

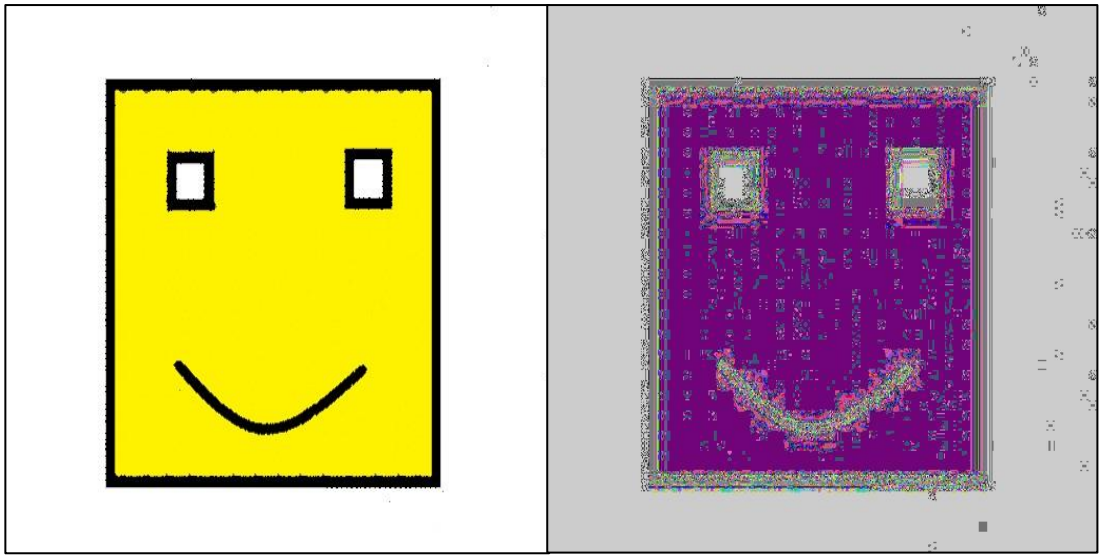
```
SHA2(str, hash_len)
```

gdje parametar `str` predstavlja podatak od kojeg se želi izračunati sažetak, a `hash_len` može biti jedna od vrijednosti (224, 256, 384, 512). Te vrijednosti predstavljaju različite načine izračuna sažetka.

2.5. ECB Mode

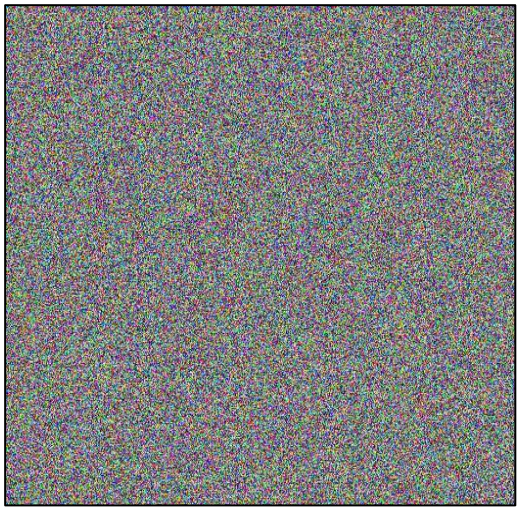
ECB je jedan od najjednostavnijih načina šifriranja koji može biti implementiran AES algoritmom [22]. Radi na način da se tekstualni podaci raspodijele na blokove od po 32 okteta i svaki blok se šifrira zasebno, ali s istim ključem. Ako se dogodi da dva bloka imaju jednake vrijednosti, šifrat će izgledati isto za oba bloka.

Glavni nedostatak koji se javlja kod ECB načina šifriranja je nedostatak raspršenja (difuzije). Raspršenje predstavlja mjeru promjene kôda. Točnije, ako bi se promijenio samo jedan bit čistog teksta koji se šifrira, šifrat koji bi se dobio bi se trebao drastično promijeniti. Na ilustraciji 2.1. b) se može vidjeti utjecaj nedostatka raspršenja, a na ilustraciji 2.2. se vidi kako bi izgledalo da se ista ilustracija šifrirala nekim drugim načinom (engl. *Mode*). Za šifriranje ilustracije 2.1. a) se koristio isti AES algoritam kakav se koristi u MariaDB sustavu za upravljanje bazom podataka.



a) Prije šifriranja b) Poslije šifriranja

Ilustracija 2.1. Grafički prikaz nedostatka raspršenja kod ECB načina šifriranja



Ilustracija 2.2. Šifriranje bilo kojim drugim načinom

MariaDB koristi ECB Mode, a podatak o tome ne piše na službenim stranicama MariaDB, ali se može eksperimentalno pokazati. Na ispisu 2.4. je prikaz podatka šifriranog u bazi podataka, a na ispisu 2.5. je dekripcija tog podatka u Pythonu. Heksadekadska vrijednost, koja se dobije kao šifrat poruke, iskorištava se kako bi se dobio čisti tekst u Pythonu.

U ovom slučaju je ključ poznat pa je to moguće, no u općem slučaju ključ neće biti poznat te je dekriptiranje „na silu“ (engl. *brute force*) neučinkovito.

```
MariaDB [ime_baze_podataka]> select hex(AES_ENCRYPT('The answer is
noThe answer is no', 'This is a key123'));
+------------------------------------------------------------------+
 hex(AES_ENCRYPT('The answer is noThe answer is no', 'This is a key'))
+------------------------------------------------------------------+
9AE8725FB9FE252B0DD7620019EF2C059AE8725FB9FE252B0DD7620019EF2C0510579C
DC19F687EAD13913872B76CAEF
+------------------------------------------------------------------+
1 row in set (0.001 sec)
```

Ispis 2.4. Kriptiranje i prikaz heksadekadske vrijednosti šifrata od poruke 'The answer is noThe answer is no'

```
1     from Crypto.Cipher import AES
2     obj = AES.new(b'This is a key123', AES.MODE_ECB)
3     obj.decrypt(bytes.fromhex("9AE8725FB9FE252B0DD7620019EF2C05 /
      9AE8725FB9FE252B0DD7620019EF2C0510579CDC19F687EAD13913872B7 /
      6CAEF"))

Ispis:
      b'The answer is noThe answer is no\x10\x10\x10  /
      \x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10'
```

Ispis 2.5. Dešifriranje podatka iz MariaDB uz pomoć Python-a

Za dešifriranje primjera s ispisa 2.5. je korišten modul *pycryptodome* [23]. Iz navedenog eksperimenta je vidljivo da je veličina bloka koji se šifrira upravo 32 okteta i da se ECB način šifriranja koristi prilikom šifriranja.

Nedostatak raspršenja se može uočiti na ispisu 2.4., jer poruka 'The answer is noThe answer is no' se podijelila na dva jednaka bloka veličine 32 okteta. I jedani i drugi blok su oblika 'The answer is no'. Iz tog razloga, oba bloka su se jednako šifrirala, a treći blok je punjenje koje se dodaje kako bi se zadovoljilo blokovsko šifriranje podatka. U ovom slučaju se dodaje treći blok kako bi se osiguralo da je cijeli tekstualni podatak kriptiran. Taj blok je oblika '`\x10`' što znači da je punjenje broj okteta koji je potreban da se zadovolji veličina jednog bloka.

2.6. Zaštita podataka na disku uz pomoć InnoDB

InnoDB je komponenta MariaDB (engl. *Storage Engine*) koja je zaslužna za spremanje podataka baze podataka na tvrdi disk. Jedan je od pretpostavljenih priključaka u MariaDB. Uz uključenje dodatnih opcija, moguće je kriptirati podatke koji se spremaju na tvrdi disk te na taj način onemogućiti pristup podacima ako tvrdi disk bude kompromitiran.

Kako bi se ostvarila zaštita podataka na disku, potrebno je pronaći datoteku *my.ini* koja se na Windows operacijskim sustavima može nalaziti na lokaciji „*C:\Program Files\MariaDB 10.5\data“*. Unutar *my.ini* datoteke se nalaze parametri pomoću kojih se MariaDB koristi [24]. Primjerice, postavljaju se vrata (engl. *port*) koja će se koristiti prilikom komunikacije klijenta i poslužitelja ili se postavlja direktorij koji će se koristiti za instaliranje dodatnih priključaka, i slično.

Kako bi se zaštitili podaci na disku, potrebno je urediti tu datoteku. Prvo se dodaje linija koja glasi:

```
plugin_load_add = file_key_managment
```

te nakon toga je potrebno stvoriti datoteku s ključevima koji će se koristiti prilikom enkripcije. Datoteka s ključevima treba sadržavati linije koje su oblika:

```
<encryption_key_id1>;<hex-encoded_encryption_key1>
<encryption_key_id2>;<hex-encoded_encryption_key2>
...
<encryption_key_idn>;<hex-encoded_encryption_keyn>
```

Svaka linija sadrži 32-bitni cijeli broj koji je identifikator ključa i točkom sa zarezom odvojenim ključem. Linija može biti proizvoljno mnogo, ali minimalno jedna čiji identifikator ključa mora biti vrijednost `1` [25].

Za generiranje ključa može se koristiti naredba `openssl rand -hex 32` koja je uobičajena za Unixoidne operacijske sustave. Na Windows operacijskim sustavima je također moguće korištenje te naredbe, ali uz prethodnu instalaciju *OpenSSL* naredbe [26]. Unutar *Powershell-a* potrebno je pokrenuti naredbu:

```
C:\Users\Windows 10> openssl rand -hex 32
```

koja će izgenerirati slučajni ključ za šifriranje duljine 256 bita:

```
09e0a68f074f67475cac1fdf8ca31303e007e4847ba8ae29ce60071c8f88bfa8
```

Nakon toga datoteku s ključevima je potrebno spremiti na neku lokaciju, poželjno unutar direktorija gdje se nalazi MariaDB instalacija, primjerice, „*C:\Program Files\MariaDB 10.5\encryption*“, i unutar tog direktorija stvoriti datoteku pod nekim nazivom, primjerice „*keyfile*“, koja će sadržavati ključeve i njihov identifikacijski broj.

Primjer konfiguracijske datoteke *my.ini* je prikazan na ispisu 2.6., a primjer datoteke s ključevima i njihovim identifikatorima je prikazan na ispisu 2.7.

```
[mysqld]
datadir=C:/Program Files/MariaDB 10.5/data
port=3306
innodb_buffer_pool_size=767M
plugin_load_add = file_key_management
file_key_management_filename=C:/Program Files/MariaDB
10.5/encryption/keyfile
[client]
port=3306
plugin-dir=C:/Program Files/MariaDB 10.5/lib/plugin
```

Ispis 2.6. Primjer konfiguracijske datoteke nakon dodatka za InnoDB šifriranje prilikom spremanja podataka na tvrdi disk

```
1;09e0a68f074f67475cac1fdf8ca31303e007e4847ba8ae29ce60071c8f88bfa8
2;49c16acc2dffe616710c9ba9a10b94944a737de1beccb52dc1560abfdd67388b
100;8db1ee74580e7e93ab8cf157f02656d356c2f437d548d5bf16bf2a56932954a3
```

Ispis 2.7. Primjer datoteke s ključevima

Da bi zaštita bila potpuna, poželjno bi bilo kriptirati i datoteku koja sadrži ključeve koje MariaDB koristi prilikom kriptiranja podataka. Sljedeće naredbe to i omogućuju:

```
C:\Users\Windows 10> openssl rand -hex 128 > "C:\Program Files\MariaDB
      10.5\encryption\keyfile.key"
```

Ova naredba omogućava stvaranje slučajne lozinke za enkripciju, a da bi se kriptirala datoteka koja sadrži ključeve, pomoću prethodno generirane datoteke, potrebno je izvršiti sljedeću naredbu:

```
C:\Users\Windows 10> openssl enc -aes-256-cbc -md -pbkdf2 -pass \
      file:"C:\Program Files\MariaDB 10.5\encryption\keyfile.key" -in \
      "C:\Program Files\MariaDB 10.5\encryption\keyfile" -out "C:\Program
      Files\MariaDB 10.5\encryption\keyfile.enc"
```

Ova naredba će pročitati datoteku „*keyfile*“ i stvoriti novu datoteku „*keyfile.enc*“ koja je kriptirana datoteka stvorena pomoću datoteke „*keyfile*“ i datoteke u kojoj se nalazi ključ za šifriranje, „*keyfile.key*“. Nakon toga, datoteka „*keyfil*e“ se može izbrisati.

Nakon što se kriptira datoteka koja sadrži ključeve za kriptiranje podataka baze podataka prilikom spremanja podataka na disk, potrebno je dodatno podesiti parametre datoteke *my.ini*. Konačan izgled datoteke *my.ini* prikazan je na ispisu 2.8.

```
[mysqld]
datadir=C:/Program Files/MariaDB 10.5/data
port=3306
innodb_buffer_pool_size=767M
plugin_load_add = file_key_management
file_key_management_filename=C:/Program Files/MariaDB
10.5/encryption/keyfile.enc
loose_file_key_management_filekey = FILE:C:/Program Files/MariaDB
10.5/encryption/keyfile.key
[client]
port=3306
plugin-dir=C:/Program Files/MariaDB 10.5/lib/plugin
```

Ispis 2.8. Konačan izgled datoteke *my.ini*

Na ovaj način su u potpunosti zaštićeni podaci na tvrdom disku. Atribute tablica je i dalje moguće pročitati, ali čiste podatke ne. To jest, ispis datoteka s nastavkom *.ibd* neće dati nikakve korisne informacije.

2.7. Model prijetnje

Kako bi se nešto zaštitilo potrebno je definirati model prijetnje [27]. Model prijetnje se može definirati za aplikacije, baze podataka ili za bilo kakvu programsku podršku (engl. *Software*) koja je podložna napadu. Svaki model prijetnje sadrži izvore prijetnje te koliku štetu ta prijetnja može načiniti.

Da bi se otkrile slabosti kriptografske zaštite u MariaDB, definirani su modeli prijetnje nad bazom podataka. Počinje se od definiranja mjesta na kojem će se napadač nalaziti, a to može biti napadač na poslužitelju na kojem se nalazi baza podataka, napadač na klijentskoj strani ili napadač koji prisluškuje podatke koji se šalju između poslužitelja i klijenta (tzv. „*Man in the Middle*“). Nakon što se pravilno definira model prijetnje, moguće prijetnje je lakše prepoznati i raditi na njihovu uklanjanju ili smanjivanju opsega.

Kada se definira model prijetnje, pojavit će se mnogo sigurnosnih prijetnji, ali pokušaj rješavanja svih nije prikladan. Glavni razlog tomu je taj što rješavanje svih sigurnosnih problema zahtijeva veliku količinu resursa.

Jedan od glavnih načina definiranja modela prijetnje je Microsoft-ov SDL (*Security Development Lifecycle*) [28]. Definiranje se izvodi u 5 koraka:

1. Definiranje sigurnosnih zahtijeva
2. Stvaranje aplikacijskog dijagrama
3. Identifikacija prijetnji
4. Ublažavanje prijetnji
5. Provjera valjanosti ublažavanja prijetnji

Nadalje, postoje i četiri različita pristupa modeliranju prijetnji:

1. pristup usredotočen na napadača
2. pristup usredotočen na programsko rješenje
3. pristup usredotočen na resurs
4. pristup usredotočen na obranu

Za potrebe definiranja sigurnosnih prijetnji za bazu podataka, iskorišten je pristup usredotočen na napadača. Isprva je poželjno raščlaniti aktore i moguće situacije, odgovoriti na pitanja zbog čega netko napada bazu podataka i koji mu je krajnji cilj.

Napadači se dijele na aktivne i pasivne [29]. Pasivni napadač je vrsta napadača koja može vidjeti podatke, ali ne može ili ne želi utjecati na njihovu izmjenu. Aktivni napadač ima mogućnost izmjene podataka. Opseg ljudi koji mogu biti aktivni ili pasivni napadači je velik. Pasivan napadač može biti neki korisnik koji prisluškuje mrežni promet na ili sa poslužitelja te iskorištava informacije koje nisu kriptirane. Aktivan napadač može biti neki maliciozni korisnik koji želi zaobići dana mu prava za korištenje aplikacije ili baze podataka, ili napadač koji je kompromitirao poslužitelj te pokušava saznati informacije koje se nalaze na njemu. Osim navedenih, napadači mogu postati i ljudi koji su radili na dizajnu aplikacije, a samim time i baze podataka.

U obzir se može uzeti napadač koji se nalazi na poslužitelju. Svi podaci koji su zapisani na poslužitelju, napadaču su dostupni u obliku u kojem su zapisani. Podaci mogu biti nešifrirani ili šifrirani. Druga vrsta napadača je napadač koji se nalazi na aplikaciji. Aplikacije koriste baze podataka kako bi spremale podatke te napadač može pokušati napasti aplikaciju kako bi došao do podataka u bazi podataka. Jedna od najpoznatijih metoda je *SQL-injekcija* [30]. Treća vrsta napadača je napadač koji se nalazi direktno na bazi podataka. Ako su podaci unutar baze šifrirani, napadač može pokušati napraviti rekonstrukcijski napad [31].

3. Model prijetnje: Napadač na bazi podataka

2014. godine se dogodilo jedno od najvećih curenja podataka iz baze podataka u povijesti [32]. Napadači su napali troje zaposlenika tvrtke *eBay* te pomoću njihovih korisničkih imena i lozinki su ukrali podatke od preko 140 milijuna korisnika. Iz tog razloga je potrebno pravilno definirati model prijetnje.

Pri definiranju modela prijetnje s obzirom na pristup modeliranju, u obzir je uzet pristup usredotočen na napadača. Prvi korak koji je potrebno napraviti je definirati sigurnosne zahtijeve. Ako se pojavi problem koji je obuhvaćen sigurnosnim zahtijevima i nije riješen, krajnji korisnik proizvoda može podnijeti tužbu protiv razvojnog tima. Za početak, jedan od zahtijeva može biti potpuna sigurnost zaštite podataka koji se nalaze na bazi podataka.

Jedan od glavnih nedostataka koje MariaDB ima je upravo taj. To jest, u obzir treba uzeti da se podaci šifriraju AES algoritmom, ali nedostatak je u činjenici da je za način (engl. *Mode*) tog algoritma implementiran ECB.

Uzmimo u obzir napadača koji je na bazi podataka. Jedan od načina na koji je provalio na bazu podataka može biti „na silu" (engl. *brute force*), tj. uzastopnim pokušajima je pogodio lozinku. Sada, napadač na bazi podataka može čitati podatke koji mogu biti šifrirani. Ako su šifrirani, bez ključa, napadač ne može doći do čistih podataka, ali može iskoristiti ranjivost koja je upravo nedostatak raspršenosti ECB načina šifriranja. Ta ranjivost se može iskoristiti rekonstrukcijom podataka.

Kod pokušaja rekonstrukcije podataka, javno dostupne informacije su od velike pomoći. Uzmimo za primjer bazu podataka neke aplikacije u kojoj se korisnici prijavljuju u sustav korištenjem korisničkog imena i lozinke. U toj bazi podataka se nalaze podaci o korisničkim imenima, lozinkama koje su kriptirane AES algoritmom s ECB načinom šifriranja i podatak koji je pomoć za lozinku. Obično se za zaštitu lozinki uzima sažetak lozinke, ali kako je cilj pojašnjenje nedostatka raspršenja ECB načina šifriranja AES algoritma u MariaDB DBMS-u, lozinke su zaštićene korištenjem ECB načina šifriranja AES algoritma.

Podatak za pomoć se često nudi korisnicima da upišu kako bi se lakše prisjetili lozinke. Da bi rekonstrukcija podataka bila efikasnija, to je potreban veći broj podataka. Ako postoje

milijuni podataka, broj podataka koje je moguće saznati time je veći čak i bez poznavanja ključa za dekripciju. Primjer takve tablice dan je u tablici 3.1.

Username	Password	Passwd hint
username_1	7566FFDFCC056A6227849E71A6CBE4020ED894999441C3D73C2C039873793EDE	It's hard password
username_2	7566FFDFCC056A6227849E71A6CBE402A0D827A71EBF9E904455A1821854B73C	I have double password
username_3	F1B3AD1F195FD24DC00A71F683444A3A	9
username_n	C82678F1926873A913D3340A9F2BC63E	1+1+1+1+1+1+1+1
username_n+1	C82678F1926873A913D3340A9F2BC63E	81
username_n+2	87D42179993F4B23C2CE486D0FE707AB	Dog's name

Tablica 3.1. Isječak iz tablice s podacima kriptiranih ECB načinom šifriranja

Passwd hint je pomoć koju je korisnik postavio kako bi se sjetio lozinke te ta pomoć ne mora biti zadana za sve podatke koji se nalaze u tablici. Neke pomoći nam neće biti od koristi, jer na primjer ne možemo zaključiti ime psa osobe s korisničkim imenom username_n+2, ali može se primijetiti dosta sličnosti u pojedinim heksadecimalnim vrijednostima lozinki.

Od velike pomoći će biti podatak o najčešće korištenim lozinkama koje ljudi koriste, a taj podatak se može naći na raznim stranicama [33]. Zbog velike količine podataka, ako se uspije saznati lozinka jednog korisnika, saznat će se sve lozinke jednake ili slične toj. Jedan od načina da se sazna jedna lozinka je taj da se ode na aplikaciju i pokuša upisati korisničko ime i lozinka te kada je ulazak u aplikaciju uspješan, jedna lozinka je otkrivena. Slične lozinke, na primjer, imaju korisnik username_1 i korisnik username_2. Uzrok sličnosti lozinki je upravo taj što ECB način šifriranja podatke grupira u blokove od 128 bita. Iz toga se može zaključiti da je prvih 128 bita lozinke korisnika username_1 jednako prvih 128 bita lozinke korisnika username_2. 128 bita je ekvivalent 16-slovnoj riječi ili čistom tekstu.

Druga vrsta prijetnje koja se može pojaviti kod napadača koji je na bazi podataka je umetanje vlastitih podataka ili kompromitiranje postojećih. I dalje je pretpostavka da je napadač na bazi podataka i da ima pristup podacima koji su šifrirani. Nakon šifriranja podaci koji se nalaze u bazi podataka su veličine 128 bita, na taj način, ako napadač promijeni bilo koji bit, podaci će se promijeniti ili izgubiti. Kompromitiranje podataka se događa ako napadač promijeni jedan bit koji predstavlja punjenje. Ako se nakon promjene bita šifriranog podatka izvede upit koji će vratiti taj podatak, rezultat koji baza podataka vrati je NULL. Nadalje, ako napadač promijeni jedan bit na šifriranom podatku koji ne predstavlja punjenje, kao rezultat upita koji bi se izveo nad tim promijenjenim podatkom se dobije podatak koji nije čitljiv. Promjena heksadekadske vrijednosti dijela koji predstavlja punjenje je prikazana na ispisu 3.1. Prvi dio ispisa prikazuje originalnu vrijednost, drugi dio ispisa prikazuje vrijednost nakon što se promijeni jedan bit na dijelu koji je punjenje.

```
MariaDB [ime_baze_podataka]> select aes_decrypt(unhex
 ('6EC5FA56513EBF3CB1E06846A71AB3B296F9B2CC373363B8601D1824D97F6C33'),
'this is a key123');    # prikaz originalnog podatka
+------------------------------------------------------------------+
moj podatak#####podatak
+------------------------------------------------------------------+
1 row in set (0.000 sec)
MariaDB [ime_baze_podataka]> select aes_decrypt(unhex
('6EC5FA56513EBF3CB1E06846A71AB3B296F9B2DC373363B8601D1824D97F6C33'),
'this is a key123');     # promjena bita na dijelu koji je punjenje
+------------------------------------------------------------------+
NULL
+------------------------------------------------------------------+
1 row in set (0.001 sec)
```

Ispis 3.1. Prikaz originalnog podatka te promjene heksadekadske vrijednosti na dijelu za punjenje

Na ispisu 3.2. je prikazano što bi se dogodilo kad bi se promijenila heksadekadska vrijednost na dijelu koji nije punjenje, tj. dio je podatka. Ako je podatak veći od 128 bita, dio podatka će ostati vidljiv i čitljiv.

```
MariaDB [ime_baze_podataka]> select aes_decrypt(unhex

('7EC5FA56513EBF3CB1E06846A71AB3B296F9B2CC373363B8601D1824D97F6C33'),

'this is a key123');    # promjena bita koji je dio podatka

+------------------------------------------------------------------+

k╚ÜdŘčGF=##ąHŐbpodatak

+------------------------------------------------------------------+

1 row in set (0.000 sec)
```

Ispis 3.2. Prikaz promjene heksadekadske vrijednosti na dijelu koji nije punjenje

Način umetanja napadačevih podataka u bazu podataka je prikazan na ispisu 3.3. Napadač može šifrirati neke svoje podatke i samo ih naljepiti na već postojeće.

```
MariaDB [ime_baze_podataka]> update tablename set data=concat(data,

unhex('96F9B2CC373363B8601D1824D97F6C33')) where id=1;

Query OK, 1 row affected (0.034 sec)

Rows matched: 1  Changed: 1  Warnings: 0
```

Ispis 3.3. Prikaz umetanja napadačevih podataka

4. Model prijetnje: Napadač na aplikaciji

Osim na bazi podataka, napadač može biti i na samoj aplikaciji. Jedna od najpoznatijih vrsta napada koju napadač može izvesti na aplikaciji kako bi dobio podatke iz baze podataka je *SQL-injekcija*. SQL-injekcijom se ignorira upit definiran u aplikaciji, a umjesto njega se izvršava napadačev upit. Kako bi došao do informacija, SQL upite upisuje na mjesto korisničkog imena ili lozinke neke aplikacije. Pretpostavka je da baza podataka nije kriptirana, te da je aplikacija podložna SQL-injekcijama.

Postoji nekoliko različitih vrsta *SQL-injekcija* [34]:

a. Napadi bazirani na tautologiji (engl. *Tautology Based Attacks*)

b. Napadi temeljeni na uniji upita (engl. *Union Queries Based Attacks*)

c. Napadi koji trajno štete bazi podataka (engl. *Piggybacked Queries*)

Ako je riječ o napadima baziranima na tautologiji, napadač ubacuje (injektira) dijelove upita (engl. *SQL tokens*) koji su uvijek istiniti. Na taj način, kao odgovor aplikacije dobije sve podatke koji se nalaze u tablici koju je napao. Primjer napada je prikazan na ispisu 4.1. Dio upita koji predstavlja tautologiju je oblika `'1'='1'`.

```
MariaDB [dbname]> select * from unamepasswd where username = 'a' or
                  '1'='1';

+----+-----------+----------------------------------+
| id | username  | password                         |
+----+-----------+----------------------------------+
|  1 | username1 | password1                        |
|  2 | username2 | password2                        |
+----+-----------+----------------------------------+
```

Ispis 4.1. *Tautology Based Attack*

Druga vrsta SQL-injekcije su napadi bazirani na uniji upita. Napadač nadoda, na upit koji se izvršava, kôd „UNION upit_koji_ga_zanima“. Na ovaj način može saznati informacije o bazi podataka koju koristi, broj stupaca u tablici i slično. Primjer napada je prikazan u ispisu 4.2.

```
MariaDB [db_name]> select * from unamepasswd where id = '1' union all
                   select NULL, NULL, version() from DUAL;

+------+-----------+----------------------------------+
| id   | username  | password                         |
+------+-----------+----------------------------------+
|    1 | username1 | password1                        |
| NULL | NULL      | 10.4.12-MariaDB                  |
+------+-----------+----------------------------------+
```

Ispis 4.2. *Union Based Attack*

Dio koji je u ispisu 4.2. dodao napadač izgleda ovako „`[aplikacijski dio upita] union all select NULL, NULL, version() from DUAL`“ te u ovom slučaju predstavlja ispis verzije DBMS-a koji je napadnut. DUAL je virtualna tablica koja služi za vježbu (engl. *Dummy table name*) [35]. U praksi, prvi dio upita nije vidljiv te napadač prvo mora saznati koliki je broj stupaca tablice koju napada te nakon toga može početi s *Union Based Attack* [36].

Treća i najštetnija vrsta napada su *Piggybacked Queries*. Jedan je od najštetnijih napada kojeg napadač može izvršiti. Razlog tomu je što takvi napadi štete bazi podataka koja je napadnuta, primjerice, brisanjem tablice nakon što su pročitani podaci. Podaci se mogu saznati korištenjem napada temeljenih na uniji upita. Primjer napada je prikazan u ispisu 4.3. Pretpostavka je da je napadač saznao ime tablice koju želi obrisati.

```
MariaDB [db_name]> select * from unamepasswd where id = '1' union all /
                   select NULL, NULL, version() from DUAL; drop table /
                   unamepasswd;

+------+-----------+----------------------------------+
| id   | username  | password                         |
+------+-----------+----------------------------------+
|    1 | username1 | password1                        |
| NULL | NULL      | 10.4.12-MariaDB                  |
+------+-----------+----------------------------------+

Query OK, 0 rows affected (0.250 sec)
```

Ispis 4.3. *Piggybacked Query*

Postoji nekoliko načina na koje se moguće zaštititi od SQL-injekcija. Zaštitu je najlakše implementirati u samoj aplikaciji. Jedan od načina na koje se moguće zaštititi od SQL-injekcije je tretiranje svakog unesenog znaka korisnikovog unosa, koji predstavlja korisničko ime ili lozinku, kao tekst, a ne kao dio upita (engl. *string escape*). Još jedan od načina na koje se moguće zaštititi je taj da se u programskom kodu doda regularni izraz koji provjerava korisnikov unos. Primjerice, prilikom unosa korisničkog imena, može se ograničiti broj slova ili brojeva od kojih bi se korisničko ime sastojalo. Osim toga, mogu se ograničiti i znakovi koji se smiju koristiti prilikom unosa korisničkog imena na slova, brojeve i neke posebne znakove kao što je, na primjer, donja crtica. Primjer regularnog izraza koji provjerava korisnikov unos za korisničko ime napisan u programskom jeziku koji podržava regularne izraze bi mogao izgledati ovako:

```
"[a-zA-Z0-9_]{4, 15}"
```

Ovaj regularni izraz bi provjeravao korisničko ime i ako sadrži velika i mala slova, brojeve i donju crticu, bio bi u redu. Osim toga, veličina korisničkog imena bi bila ograničena na najmanje četiri znaka, a najviše petnaest znakova. Ako bi se pri unosu korisničkog imena pojavio neki drugi znak, aplikaciji bi to bio alarm da se pokušava izvršiti SQL-injekcija.

Drugi način na koji se moguće zaštititi od SQL-injekcija je implementacija zaštite unutar baze podataka. Najlakši način je uz pomoć *ClusterControl* sustava za nadgledanje baze podataka [37]. Dio *ClusterControl-a* koji se koristi kako bi se nadgledao promet između baze podataka i aplikacije je *ProxySQL* [38]. Uz pomoć *ProxySQL* se ograniče upiti koji su dostupni korisniku [39][40]. Ograničavanje upita se ostvaruje na način da se generira lista dopuštenih upita (engl. *whitelist*) koje korisnik smije izvršiti. Svi upiti koji su dozvoljeni se izvršavaju, a upiti koji nisu dozvoljeni generiraju grešku. Kako su na listi dopuštenih svi upiti koje korisnik smije izvršiti, SQL injekcija nije moguća.

5. Zaključak

MariaDB kao sustav za upravljanje bazama podataka je prikladan za manje projekte, ali ne i za velike tvrtke i korporacije kojima je zaštita podataka unutar baze podataka od velike važnosti. Samim time što je implementiran ECB način šifriranja AES algoritma, moguće je rekonstruirati podatke upravo zbog nedostatka raspršenja. Za uspješnu rekonstrukciju podataka potrebna je velika količina podataka, javno dostupne informacije i malo logičkog razmišljanja.

U sklopu ovog rada prikazano je kako napadač može dobiti podatke iz šifrata bez poznavanja ključa za šifriranje uz pomoć javno dostupnih informacija. Osim toga prikazano je i kako napadač može ugroziti bazu podataka te pročitati podatke unutar nje ili čak umetnuti svoje podatke. Objašnjen je pojam SQL-injekcije i kako se aplikacijski zaštititi od takve vrste napada.

6. Literatura

[1] U.S. Bureau Of Labor Statistics, *Computer and Information Technology Occupations*, 10. Travanj 2020. Dostupno na: https://www.bls.gov/ooh/computer-and-information-technology/home.htm [Pristupljeno: 10. lipnja 2020.]

[2] B. Marr, *How Much Data Is There In The World?*, 2019. Dostupno na: https://www.bernardmarr.com/default.asp?contentID=1846 [Pristupljeno: 10. lipnja 2020.]

[3] MariaDB Foundation, *MariaDB Server: The open source realtional database*, 2020. Dostupno na: https://mariadb.org/ [Pristupljeno: 2. lipnja 2020.]

[4] J. Lake, *What is AES encryption and how does it work?*, 17. Veljače 2020. Dostupno na: https://www.comparitech.com/blog/information-security/what-is-aes-encryption/ [Pristupljeno: 10. lipnja 2020.]

[5] Acunetix, *What is SQL Injection (SQLi) and How to Prevent It*, 2020. Dostupno na: https://www.acunetix.com/websitesecurity/sql-injection/ [Pristupljeno: 10. lipnja 2020.]

[6] M. Vetter, *Database Design Methodology*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[7] MariaDB, *MariaDB Source Code*, 2020. Dostupno na: https://mariadb.com/kb/en/getting-the-mariadb-source-code/ [Pristupljeno: 2. lipnja 2020.]

[8] Oracle Corporation and/or its affiliates, "MySQL", 2020. Dostupno na: https://www.mysql.com/ [Pristupljeno: 10. lipnja 2020.]

[9] MariaDB, *Understanding MariaDB Architecture*, 2020. Dostupno na: https://mariadb.com/kb/en/understanding-mariadb-architecture/ [Pristupljeno: 2. lipnja 2020.]

[10] MariaDB, *Plugin SQL Statements*, 2020. Dostupno na: https://mariadb.com/kb/en/plugin-sql-statements/ [Pristupljeno: 2. lipnja 2020.]

[11] MariaDB, *InnoDB and XtraDB*, 2020. Dostupno na: https://mariadb.com/kb/en/innodb/ [Pristupljeno: 10. lipnja 2020.]

[12] MariaDB, *Aria Storage Engine*, 2020. Dostupno na: https://mariadb.com/kb/en/aria-storage-engine/ [Pristupljeno: 10. lipnja 2020.]

[13] MariaDB, *Choosing the Right Storage Engine*, 2020. Dostupno na: https://mariadb.com/kb/en/choosing-the-right-storage-engine/ [Pristupljeno: 2. lipnja 2020]

[14] MariaDB, *User Statistics*, 2020. Dostupno na: https://mariadb.com/kb/en/user-statistics/ [Pristupljeno: 10. lipnja 2020.]

[15] MariaDB, *Authentication Plugins*, 2020. Dostupno na: https://mariadb.com/kb/en/authentication-plugins/ [Pristupljeno: 3. lipnja 2020.]

[16] MariaDB, *Query Cache*, 2020. Dostupno na: https://mariadb.com/kb/en/query-cache/ [Pristupljeno: 10. lipnja 2020.]

[17] MariaDB, *Server Monitoring & Logs*, 2020., Dostupno na: https://mariadb.com/kb/en/server-monitoring-logs/ [Pristupljeno: 3. lipnja 2020.]

[18] MariaDB, *How to connect Python programs to MariaDB*, 14. studeni 2014. Dostupno na: https://mariadb.com/resources/blog/how-to-connect-python-programs-to-mariadb/ [Pristupljeno: 2. lipnja 2020.]

[19] S. Kelly, *Security Implications of Using the Data Encryption Standard (DES)*, RFC4772, Aruba Networks, Prosinac 2006. Dostupno na: https://www.ipa.go.jp/security/rfc/RFC4772EN.html [Pristupljeno: 2. lipnja 2020.]

[20] S. Wang, *The difference in five modes in the AES encryption algorithm*, 8. kolovoz 2019. Dostupno na: https://www.highgo.ca/2019/08/08/the-difference-in-five-modes-in-the-aes-encryption-algorithm/ [Pristupljeno: 2. lipnja 2020.]

[21] Algorithm Hall of Fame, *SHA-2*, 2020. Dostupno na: https://www.algorithmhalloffame.org/algorithms/sha-2/ [Pristupljeno: 2. lipnja 2020.]

[22] A. Nath, *The Modern Cryptography Cookbook: Learn Crypto Principle to Applied Cryptography with example*, Anish Nath, 2018.

[23] Built with Sphinx using a theme provided by Read the Docs, *Welcome to PyCryptodome's documentation*, n. d. Dostupno na: https://pycryptodome.readthedocs.io/en/latest/ [Pristupljeno: 2. lipnja 2020.]

[24] MariaDB, *Configuring MariaDB with Option Files*, 2020. Dostupno na: https://mariadb.com/kb/en/configuring-mariadb-with-option-files/ [Pristupljeno: 10. lipnja 2020.]

[25] MariaDB, *File Key Management Encryption Plugin*, 2020. Dostupno na: https://mariadb.com/kb/en/file-key-management-encryption-plugin/ [Pristupljeno: 10. lipnja 2020.]

[26] Shining Light Productions, *Win32/Win64 OpenSSL*, 2003. Dostupno na: https://slproweb.com/products/Win32OpenSSL.html [Pristupljeno: 2. lipnja 2020.]

[27] Stručni rad, Centar Informacijske Sigurnosti, *Modeliranje sigurnosnih prijetnji Threat Modeling)*, CIS-DOC-2012-05-049, Zagreb: Fakultet Elektrotehnike i Računarstva, svibanj. 2012.
Dostupno na: https://www.cis.hr/files/dokumenti/CIS-DOC-2012-05-049.pdf

[28] Microsoft, *Microsoft Threat Modeling Tool*, 2020. Dostupno na: https://docs.microsoft.com/en-us/azure/security/develop/threat-modeling-tool [Pristupljeno: 2. lipnja 2020.]

[29] Stručni članak, P. Grubbs, T. Ristenpart, V. Shmatikov, „Why Your Encrypted Database Is Not Secure“, 10. svibanj 2017. Dostupno na: https://dl.acm.org/doi/pdf/10.1145/3102980.3103007 [Pristupljeno: 2. lipnja 2020.]

[30] Stručni članak, W. G. J. Halfond, J. Viegas i A. Orso, „A Classification of SQL Injection Attacks and Countermeasures“, 2006. Dostupno na: https://www.cc.gatech.edu/fac/Alex.Orso/papers/halfond.viegas.orso.ISSSE06.pdf [Pristupljeno: 2. lipnja 2020.]

[31] Stručni članak, M. S. Lacharité, „Breaking Encrypted Databases: Generic Attacks on Range Queries“, kolovoz 2019. Dostupno na: https://i.blackhat.com/USA-19/Thursday/us-19-Lacharite-Breaking-Encrypted-Databases-Generic-Attacks-On-Range-Queries-wp.pdf [Pristupljeno: 10. lipnja 2020.]

[32] Članak u novinama, J. Finkle i D. Seetharaman „Cyber Thieves Took Data On 145 Million eBay Customers By Hacking 3 Corporate Employees“, *Business Insider*, 27. svibanj 2014. Dostupno na: https://www.businessinsider.com/cyber-thieves-took-data-on-145-million-ebay-customers-by-hacking-3-corporate-employees-2014-5 [Pristupljeno: 2. lipnja 2020.]

[33] Članak s Wikipedije, „List of the most common passwords“, Dostupno na: https://en.wikipedia.org/wiki/List_of_the_most_common_passwords [Pristupljeno: 2. lipnja 2020.]

[34] P. Sharma, „Database Security: Attacks and Techniques“, Prosinac 2016., Dostupno na: https://www.ijser.org/researchpaper/Database-Security--Attacks-and-Techniques.pdf [Pristupljeno: 3. lipnja 2020.]

[35] MariaDB, *DUAL*, 2020. Dostupno na: https://mariadb.com/kb/en/dual/ [Pristupljeno: 10. lipnja 2020.]

[36] PortSwigger Ltd., „SQL injection UNION attacks“, 2020., Dostupno na: https://portswigger.net/web-security/sql-injection/union-attacks [Pristupljeno: 3. lipnja 2020.]

[37] Severalnines, *ClusterControl*, 2020. Dostupno na: https://severalnines.com/product/clustercontrol [Pristupljeno: 11. lipnja 2020.]

[38] Severalnines, *ClusterControl for ProxySQL*, 2020. Dostupno na: https://severalnines.com/product/clustercontrol/mysql-load-balancing-proxysql [Pristupljeno: 11. lipnja 2020.]

[39] K. Ksiazek, *how to Protect your MySQL or MariaDB Database From SQL Injection: Part One*, 10. veljače 2020. Dostupno na: https://severalnines.com/database-blog/how-protect-your-mysql-or-mariadb-database-sql-injection-part-one [Pristupljeno: 11. lipnja 2020.]

[40] K. Ksiazek, *how to Protect your MySQL or MariaDB Database From SQL Injection: Part Two*, 20. veljače 2020. Dostupno na: https://severalnines.com/database-blog/how-protect-your-mysql-or-mariadb-database-sql-injection-part-two [Pristupljeno: 11. lipnja 2020.]

Sažetak

U ovom radu je opisana upotreba MariaDB sustava za upravljanje bazom podataka te slabosti koje se u tom sustavu pojavljuju. Upotreba sustava je prikazana nizom primjera u obliku ispisa koje je moguće isprobati i implementirati. Opisan je AES način kriptiranja podataka i posebno ECB način šifriranja, njegove slabosti i propusti, jer se kao takav koristi u MariaDB. Osim njega prikazano je i kako uspostaviti zaštitu pomoću InnoDB *Storage Engine*-a. Definiran je model prijetnje napadača na bazi podataka i prikazano je što napadač može saznati uz pomoć javno dostupnih informacija uz primjere. Osim informacija koje može saznati, prikazano je i na koji način napadač može ugroziti bazu podataka. Dodatno je definiran model napadača na aplikaciji. Opisane su SQL-injekcije i kako se obraniti od njih, aplikacijski i unutar same baze podataka.

Skraćenice

AES	*Advanced Encryption Standard*	Napredni standard za kriptiranje
DES	*Data Encryption Standard*	Standard za kriptiranje podataka
DBMS	*Database Managment System*	Sustav za upravljanje bazom podataka
SHA2	*Secure Hash Algorithm 2*	Sigurni algoritam za sažimanje
ECB	*Electronic Code Book*	Najjednostavniji način šifrianja
CBC	*Cipher Block Chaining Mode*	Način povezivanja šifrata u blokove
CFB	*Cipher Feedback Mode*	Način povratne informacije šifre
OFB	*Output Feedback Mode*	Treći način AES algoritma
CTR	*Counter Mode*	Način uz brojač
XOR	*Exclusive OR*	Ekskluzivno ili
IV	*Initialization Vector*	Inicijalizacijski vektor
SDL	*Security Development Lifecycle*	Microsoft-ov način definiranja modela prijetnje