VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

SQL injekcijų aptikimo sistema

SQL injection detection system

Bakalauro darbas

Atliko: Vytautas Vargonas

Darbo vadovas: lekt. dr. Vytautas Valaitis

Darbo recenzentas: doc. dr. Saulius Minkevičius

Vilnius, 2017

Santrauka

Programuotojai ne visada pasirūpina vystomų programų saugumu. Todėl ieškoma sprendimų kaip apsiginti nuo programų puolėjų.

Šiame darbe pristatoma SQL injekcijų aptikimo sistema. Sistema analizuoja HTTP užklausų parametrus ir atskiria puolimus. Ji veikia neprižiūrimo mašininio mokymosi pagrindu - apmokyta įprastų programos užklausų duomenimis atskiria neįprastus vartotojų parametrus. Kadangi apmokymas nėra priklausomas nuo žinių apie SQL injekcijas, sistema gali aptikti dar nežinomus SQL injekcijų puolimus.

Eksperimentui atlikti buvo sukurta žiniatinklio programa. Eksperimentas atliktas naudojant populiariausias SQL injekcijas. Rezultatai parodė, jog siūlomos sistemos veiksmingumas priklauso nuo jos apmokymo duomenų įvairovės bei ypatybių.

Raktiniai žodžiai: SQL injekcijos, įsilaužimų aptikimo sistema, atraminių vektorių mašinos, neprižiūrimas mašininis mokymasis.

Summary

Programmers do not always ensure security of developed systems. That is why it is important to look for solutions outside being reliant on developers.

In this work SQL injection detection system is proposed. The system analyzes HTTP request parameters and detects intrusions. It is based on unsupervised machine learning. Trained by regular request data system detects outlier user parameters. Since training is not reliant on previous knowledge of SQL injections, the system is able to detect unknown attacks.

Experiments show that effectiveness of proposed system relies on diversity and features of training data.

Keywords: SQL injection attacks, intrusion detection system, support vector machines, unsupervised machine learning.

Turinys

Santrauka **2**

Summary **3**

Turinys **4**

Įvadas **5**

Tikslas 7

Uždaviniai 7

1. Įsilaužimų aptikimo sistemos **8**

2. SQL injekcija **10**

2.1. SQL injekcijos veikimo principas 10

2.2. Klaidomis pagrįstos SQL injekcijos 11

2.3. Užklausų sąjungų SQL injekcijos 12

2.4. Aklosios SQL injekcijos 13

3. Mašininis mokymasis **15**

3.1. Atraminių vektorių mašinos 15

3.2. Scikit-learn 16

4. Ypatybių atranka **18**

4.1. Simbolių kiekis 18

4.2. Specialių simbolių kiekis 18

4.3. SQL raktažodžių kiekis 18

5. Eksperimentas **20**

5.1. Siūloma sistema 20

5.2. Realizuota sistema 21

5.3. Įrankiai 21

5.4. Tiriamosios žiniatinklio funkcijos 21

5.5. Klasifikatorių apmokymas 24

5.6. Testavimas 25

Rezultatai ir Išvados **28**

Šaltiniai **29**

Įvadas

Pasaulis dideliu tempu juda e-komercijos kryptimi kur verslas vykdomas per žiniatinklį ar internetu. Šioje srityje saugumas yra pagrindinis rūpestis dėl HTTP prigimties, kuri sukelia tam tikras saugumo problemas. Vystant pilnai nuo įsilaužimų neapsaugotą programinę įrangą rizikuojama netyčiniu būdu atskleisti privačią vartotojų informaciją arba patirti nuostolius leidžiant brangius programinės įrangos atnaujinimus, kai ji jau išleista rinkai. Tai gali turėti neigiamas pasekmes verslui - nuo klientų pasitikėjimo praradimo iki verslo žlugimo.

Žiniatinklio programoms ypatingai rimtą pavojų kelia SQL[1] injekcijų puolimai, kurių metu nepatikimi duomenys perduodami interpretatoriui kaip dalis užklausos. Interpretatorius, gavęs priešiškus puolėjo nurodymus, gali įvykdyti neplanuotus veiksmus ar parodyti neleistinus duomenis [CC13a]. Neteisėtų užklausų atlikimas duomenų bazėje gali sukelti rimtų problemų, nuo prisijungimo kito vartotojo paskyra, priėjimo prie jautrios informacijos, iki įrašų ištrynimo [BCA09].

Yra sukurta svetainė pavadinimu „SQL injekcijų gėdos salė”[2] [TCC17], kur išvardinta virš 160 SQL injekcijų puolimų atvejų didelių organizacijų programose. Tarp šių puolimų yra gausu atvejų, kuomet buvo nutekinta milijonų vartotojų informacija, kaip „Ubuntu Linux” ar „Sony Pictures” atvejais, arba rasta spragų žinomose turinio valdymo sistemose kaip „Drupal”, „Joomla” ar „Magento”. Iš to galima spręsti, kad programinės įrangos kūrėjams trūksta patirties saugumo srityje arba jie tikisi, jog saugumu pasirūpins kažkas kitas.

Įprastas būdas ginti modernias žiniatinklio programas yra įdiegiant įsilaužimų aptikimo modelius žiniatinklio programų ugniasienėse [BCA09]. Tokios sistemos dažnai naudoja netinkamo naudojimo aptikimo būdą lygindamos HTTP užklausą su turimu puolimų šablonų sąrašu [BCA09]. Labai populiari sistema veikianti užklausų atitikmens pagrindu yra „Snort IDS” [NM04] ir „Cloud flare” žiniatinklio programų ugniasienės naudojamas „ModSecurity”. Šie įrankiai dažnai naudojami praktikoje dėl jų įgyvendinimo paprastumo, tačiau tokiu pagrindu veikiančios sistemos yra neefektyvios aptinkant nežinomus puolimus, kurių nėra šablonų sąrašuose.

[1] Structured Query Language
[2] angl. SQL injection Hall-of-Shame

Dar vienas būdas ginti modernias žiniatinklio programas yra neįprastų užklausų aptikimo sistemos. Viena jų yra *SOFIA* [CMN16], kuri tikrina užklausas duomenų bazių sluoksnyje. Jai pateikiama pridėtinė informacija, nes užklausa tikrinama neatsižvelgiant ar buvo naudojami kliento parametrai bei tikrinama visa užklausa duomenų bazei, nors injekcijai aptikti užtenka nagrinėti kliento parametrus. Kitoje sistemoje buvo realizuotas žiniatinklio servisas [UOB17], kuriam pateikus užklausos duomenis gaunamas atsakymas ar užklausa yra potencialus SQL injekcijos puolimas. Ši sistema buvo apmokyta dviejų klasių klasifikatoriumi - gerų užklausų ir puolimų, tačiau toks sprendimas neapsaugo nuo nežinomų puolimų, kuriems nebuvo iš anksto pasiruošta. Pastarosios sistemos nėra tokios populiarios kaip anksčiau minėtos dėl įgyvendinimo sudėtingumo. Vis dėlto neįprastų užklausų aptikimo programos gali apsaugoti ir nuo nežinomų puolimų.

Kadangi neįprastų užklausų aptikimo sistemos turi didesnį potencialą apsaugoti nuo galimų grėsmių, verta skirti daugiau dėmesio jų pritaikymo analizei. Todėl šiame darbe siekiama patikrinti sistemos, veikiančios mašininio mokymosi[3] pagrindu veiksmingumą. Ši sistema nagrinėtų kliento parametrus ir gebėtų atskirti net ir tas SQL injekcijas, kurių nėra šablonų sąrašuose.

Pirmiausia šiame darbe bus pristatomos įsilaužimų aptikimo sistemų klasifikacijos, jų privalumai bei trūkumai. Ši informacija parodo, jog kelių klasifikacijų derinimas gali panaikinti kai kuriuos trūkumus. Toliau darbe nagrinėjami SQL injekcijos puolimų principai bei trys dažniausiai aptinkami jų tipai. Jų analizė yra reikalinga siekiant išsiaiškinti kokios ypatybės yra svarbiausios atskiriant SQL injekcijas nuo įprastų užklausų. Sekančioje dalyje pristatomas pasirinktas mašininio mokymosi algoritmas, kuris bus naudojamas apmokyti klasifikatorius[4], atskiriančius įprastas ir neteisėtas užklausas. Kitoje dalyje atrenkamos ypatybės, naudojamos aptikti SQL injekcijas. Pristatyta teorinė darbo dalis pasitelkiama eksperimentui atlikti. Šiuo eksperimentu tikrinama galimybė naudoti neprižiūrimą mašininį mokymąsi[5] SQL injekcijų puolimų atpažinimo sistemose. Galiausiai bus pateikti eksperimento rezultatai ir gautos išvados.

[3] angl. machine learning
[4] angl. classifier
[5] angl. unsupervised learning

Tikslas

Pasiūlyti mašininio mokymosi pagrindu veikiančią SQL injekcijų puolimų aptikimo sistemos architektūrą bei sukurti jos prototipą.

Uždaviniai

1. Pasirinkti mašininio mokymosi algoritmą.
2. Atrinkti ypatybes, pagal kurias bus atskirtos SQL injekcijos.
3. Paruošti testinę žiniatinklio aplikaciją.
4. Paruošti įprastų kliento užklausų duomenų bazę.
5. Apmokyti klasifikatorius.
6. Ištestuoti sistemos veiksmingumą.

1. Įsilaužimų aptikimo sistemos

Įsilaužimų aptikimo sistema yra programinė įranga, kuri kontroliuoja tinklą ar sistemą aptikdama neteisėtą veiklą. Įsilaužimų aptikimo sistemos klasifikuojamos į parašo-pagrindo, anomalijos-pagrindo ir būsenų protokolo analizės.

Parašo-pagrindo - tikrina paketus tinkle ir lygina juos su iš anksto apsibrėžtais puolimų modeliais [LHL13]. Šiame darbe bus tikrinami http užklausomis siunčiami vartotojo parametrai, ir jie lyginami su SQL užklausų kalbos raktažodžiais.

Anomalijos-pagrindo - anomalija tai nukrypimas nuo įprastos elgsenos. Įprasta elgsena yra nustatoma renkant per kurį laiką surenkant duomenis apie vartotojo veiksmus [LHL13]. Šiame darbe bus įrašomi įprastos elgsenos vartotojo užklausų duomenys į duomenų bazę, Vėliau pagal šiuos duomenis naudojant mašininį mokymąsi bus bandoma aptikti anomalijas.

Būsenų protokolo analizė - įsilaužimų aptikimo sistema žino ir seka protokolų būsenas ir jų nukrypimus [LHL13]. Šis darbas tokios kategorijos nenaudos, nes ji yra labiau tinkama tinklo saugumo priežiūrai, o darbe tiriamas žiniatinklio programos saugumas.

Lentelėje 2 yra lyginamos įsilaužimų aptikimo kategorijos [LHL13, JS00, SPS10, XPS11].

1 Lentelė. Įsilaužimų aptikimo sistemų kategorijų privalumai ir trūkumai

	Parašo-pagrindo	**Anomalijos-pagrindo**	**Būsenų protokolo analizė**
Privalumai	Paprastas ir efektyvus metodas aptikti žinomus puolimus Detali konteksto analizė	Efektyvus metodas aptikti naujas, nenumatytas spragas Mažiau priklausoma nuo operacinės sistemos Palengvina privilegijų išnaudojimo aptikimą	Žino ir seka protokolų būsenas Atskiria netikėtas komandų sekas

Trūkumai	Neefektyvus aptikti nežinomus puolimus, žinomų puolimų variacijas	Prastas tikslumas dėl stebimų įvykių nuolatinės kaitos	Reikalauja daug resursų
	Nereikalauja supratimo apie būsenas ir protokolus	Neprieinama atstatinėjant elgsenos profilius	Neatskiriami puolimai atrodantys kaip gera protokolo elgsena
	Sudėtinga palaikyti atnaujintus šablonus	Sudėtinga paleisti įspėjimą tinkamu metu	Gali netikti skirtai operacinei sistemai ar prieigos taškui
	Reikalauja daug laiko palaikyti žinias atnaujintas		

Parašo-pagrindo ir anomalijos-pagrindo metodai papildo vienas kitą: pirmasis tikrina konkrečius puolimus, o antrasis susitelkia ties nežinomais [LHL13]. Todėl autoriaus kuriama sistema naudos šiuos abu įsilaužimų aptikimo sistemų principus.

2. SQL injekcija

SQL[6] injekcija yra saugumo išnaudojimo tipas, kuriame puolėjas prideda SQL užklausą į formos įvesties langelį ar tiesioginę nuorodą, kad gautų prieigą prie išteklių arba keistų duomenis [Mar].

2.1. SQL injekcijos veikimo principas

Puolėjas įterpia paruoštą užklausą į siunčiamus serveriui parametrus. Serveris pagal ją sukonstruoja užklausą duombazei. Nors duombazę saugo ugniasienė, SQL injekcijoms ji nekliudo, nes puolėjas nebando įsilaužti į duombazę pats. Užklausos siunčiamos per žiniatinklio programą, nuo kurios ugniasienė nesaugo. Duombazė ištraukus duomenis juos siunčia serveriui, o pastarasis duoda atsakymą klientui.

1 pav. parodytas plačiai naudojamas pavyzdys, kaip prisijungiama prie neapsaugotos aplikacijos SQL injekcijos būdu.

Turime prisijungimo formą, kuri reikalauja prisijungimo vardo ir slaptažodžio. Žiniatinklio serveris siunčia užklausą duombazei, įrašęs atitinkamus kintamuosius[7]:

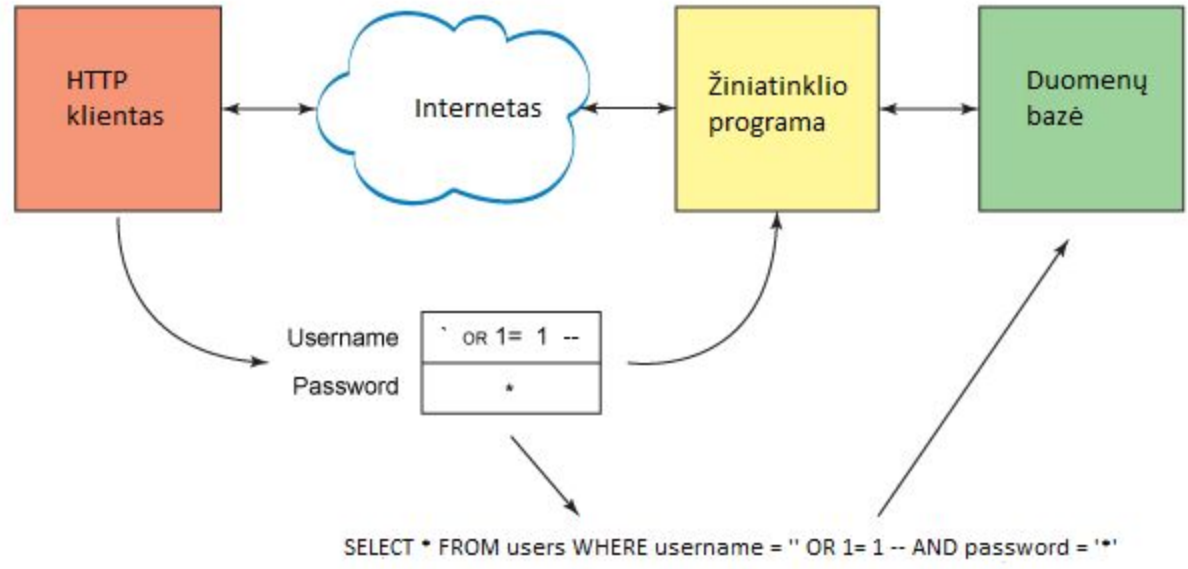
*SELECT * FROM users WHERE username = 'username' AND password = 'password';*

Jeigu buvo rastas vartotojas su atitinkamu prisijungimo vardu ir slaptažodiu – vartotojas prisijungia prie sistemos. Paveiksle 1 parodyta, kaip galima pakeisti žiniatinklio serverio generuojamą užklausą. Į reikalaujamą prisijungimo vardo lauką įrašius „ ' OR 1 = 1 -- “, į slaptažodžio lauką įrašius „*“, serverio užklausa atrodys taip:

*SELECT * FROM users WHERE username = '' OR 1 = 1 -- AND password = '*';*

Šios užklausos loginė dalis „ username = '' OR 1 = 1“ visiems duomenų bazės lentelės users įrašams visuomet bus tiesa, dviejų brūkšnelių simbolis „--“ užkomentuoja toliau einančią užklausos dalį. Taigi ši užklausa grąžins visus users lentelės elementus.

[6] struktūrinių užklausų kalbos
[7] *username* ir *password*

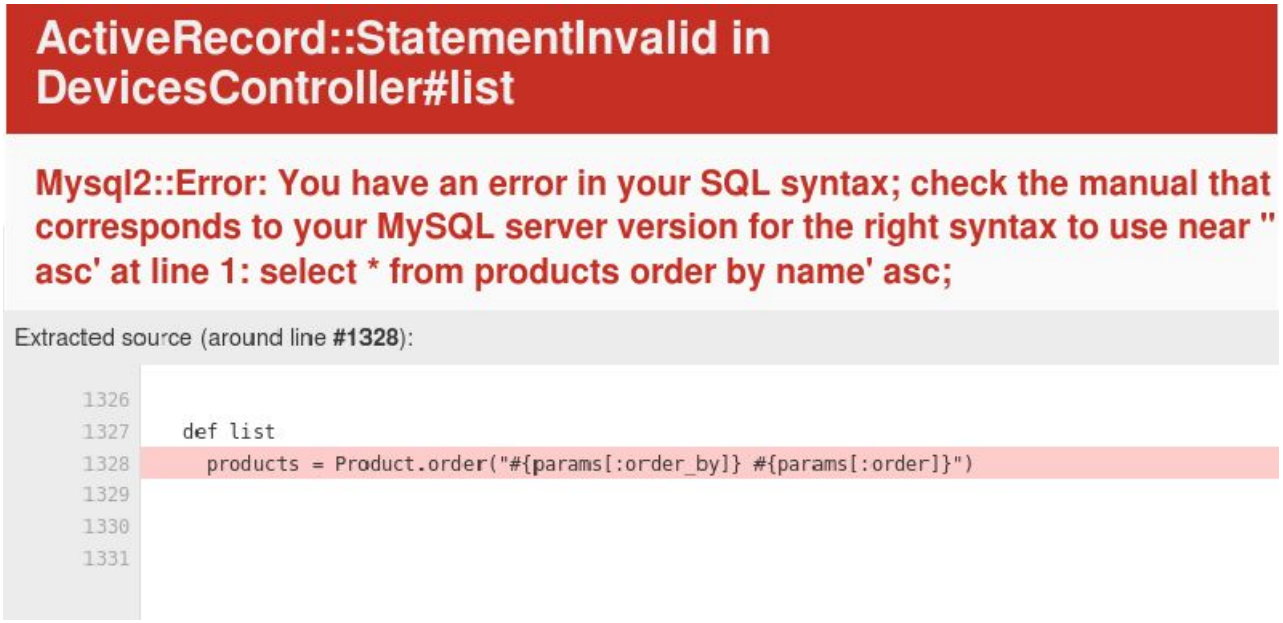


1 pav. Įsilaužimas į aplikaciją per prisijungimo formą [Tim14]

2.2. Klaidomis pagrįstos SQL injekcijos

Klaidomis pagrįstos injekcijos yra vienas dažniausiai pasitaikančių ir paprasčiausiai išnaudojamų SQL injekcijos tipų. Jos galimos, kai aplikacija grąžina duomenų bazės klaidų pranešimus. Šio tipo puolimas yra ypač paprastas, nes iš klaidos pranešimų galime išgauti įvairios informacijos apie duombazės vidinę struktūrą ir net apie pačius duomenis.

Nesudėtingas būdas gauti klaidą aplikacijoje yra surasti vietą, kur tiesiogine nuoroda yra perduodami parametrai ir prie vieno iš jų prirašyti kabutę. Pavyzdžiui turime internetinę parduotuvę, */products/list* puslapyje yra išvardinti visi produktai. Yra galimybė rikiuoti pagal produkto vardą arba kainą. Paspaudus rikiuoti pagal vardą siunčiama užklausa su parametrais *order_by* ir *order* */products/list?order_by=name&order=asc*. Prirašius kabutę prie *order_by* parametro, t.y. atidarant tiesioginę nuorodą */products/list?order_by=name'&order=asc* turėtų įvykti klaida. Antrame paveiksle matoma aplikacijos grąžinta klaida įvykusi duomenų bazėje. Vartotojui neturi būti rodomi informacijos duodantys klaidos pranešimai. Paveiksle 3 yra geras pavyzdys kaip turi būti valdomos sistemos klaidos.



2 pav. Aplikacijos grąžinamas klaidos pranešimas pateikiantis per daug informacijos



3 pav. Neduodantis informacijos klaidos pranešimas [AI14]

2.3. Užklausų sąjungų SQL injekcijos

Užklausų sąjungų injekcija remiasi papildomos užklausos įrašymu perduodant parametrus aplikacijai. Duomenų bazė grąžina apjungtus kelių užklausų duomenis. Taigi įvykdžius šį puolimą aplikacija vartotojui atvaizduos pagrindinės užklausos bei prijungtus duomenis.

Pavyzdžiui turime internetinės parduotuvės puslapį kuriame išvardinti produktai. Produktų kiekiui riboti puslapyje yra produktų puslapiavimas pagal produkto pavadinimo pirmą raidę. Puslapyje kiekvienam produktui matome išvardintus pavadinimus ir kainas. Tiesioginė nuoroda naršyklėje atrodo taip: /products/list?letter=A. Parametras letter nurodo kokia raide turi prasidėti produkto pavadinimas. Iš to galima numanyti, jog aplikacijos siunčiama užklausa duomenų bazei

yra:

*SELECT * FROM products WHERE name LIKE 'a%';*

Dėl to, kad aplikacijos atsakyme gauname tik produkto vardą ir kainą, tikėtina, jog užklausa duombazei prašo tik trijų stulpelių: id, name, price. Rašant sąjungos užklausą yra svarbu, jog jos traukiamų stulpelių skaičius sutaptų su pagrindinės užklausos gaunamų stulpelių skaičiumi. Sakykime norime sužinoti visų vartotojų prisijungimo vardus ir slaptažodžius. Sąjungos užklausos dalis turės būti:

SELECT id, username, password FROM users;

Įdėjus šią užklausą į tiesioginę nuorodą, pastaroji bus */products/list?letter=A%' UNION SELECT id, username, password FROM users; --*. Aplikacijos siunčiama užklausa duomenų bazei:

*SELECT * FROM products WHERE name LIKE 'a%' UNION SELECT id, username, password FROM users; --%';*

Sėkmingai įvykdžius užklausą, aplikacijos atsakyme bus grąžinami ne tik produktai prasidedantys raide a, bet ir visų vartotojų prisijungimo duomenys, kuriuos aplikacija traktuos kaip duomenis apie produktą.

Šiame pavyzdyje duomenų bazės lentelės ir stulpeliai buvo žinomi, tačiau galima juos atspėti bandymo būdu.

2.4. Aklosios SQL injekcijos

Aklosios SQL injekcijos ataka užklausoje užduoda duombazei klausimus, ir priklausomai ar atsakymas yra tiesa, tikimasi skirtingo aplikacijos atsako [CC13b]. Aklosios SQL injekcijos gali būti kelių tipų: pagrįstos turiniu ir pagrįstos laiku.

Turiniu pagrįsta akloji SQL injekcija atliekama vietoj perduodamo tiesiogine nuoroda parametro įrašius *IF* sąlygą nuo kurios priklausys gaunamas atsakymas. Pavyzdžiui, internetinėje parduotuvėje, kurioje yra išrikiuoti produktai pagal jų vardą, turime tiesioginę nuorodą */products/list?order_by=name&order=asc*. Generuojama užklausa duomenų bazei:

*SELECT * FROM products ORDER BY name ASC;*

Pakeičiant perduodamą *order_by* parametrą į *IF* sakinį, kuris grąžintų *name* arba *price* atitinkamai jei atsakymas yra *TRUE*[8] arba *FALSE*[9]. Sakykime, mus domina ar pirmo vartotojo prisijungimo vardas duomenų bazėje yra „admin“. Tuomet rašome užklausą:

IF((SELECT username FROM users LIMIT 1) = 'admin', name, price).

Ją įterpiame į tiesioginę nuorodą:

/products/list?order_by=IF((SELECT username FROM users LIMIT 1) = 'admin', name, price)&order=asc

Siunčiama užklausa duomenų bazei tuomet bus:

*SELECT * FROM products ORDER BY IF((SELECT username FROM users LIMIT 1) = 'admin', name, price) ASC;*

Iš aplikacijos gautame atsakyme matoma ar produktai surikiuoti pagal pavadinimus ar pagal kainą. Iš to išplaukia ar *IF* sąlyga yra teisinga ar klaidinga.

Kai kuriais atvejais įvykdžius užklausą aplikacija vartotojui negrąžina jokių duomenų turinio pavidalu, taigi neįmanoma panaudoti užklausų sąjungos ar aklosios pagrįstos turiniu SQL injekcijos. Tokiais atvejais naudojama laiku pagrįsta akloji SQL injekcijos ataka. Į užklausą įterpiama *IF* sąlygą, kuri priklausomai nuo sąlygos teisingumo iškviečia duomenų bazės funkciją sukeliančią atsako užlaikymą. Iš atsako užlaikymo trukmės sprendžiama apie įterptos sąlygos teisingumą.

[8] tiesa
[9] netiesa

3. Mašininis mokymasis

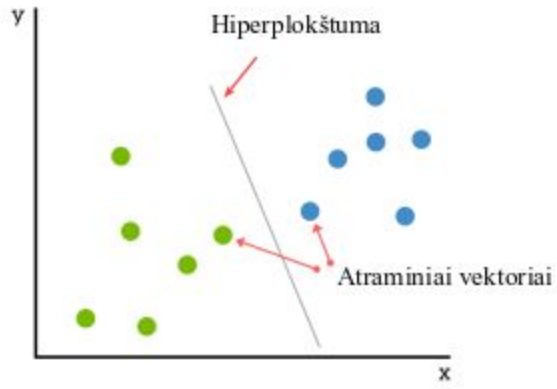
Mašininis mokymasis yra algoritmų konstrukcija kuri gali mokytis iš pateiktų duomenų ir pateikti prognozes jiems [KP98]. Mašininis mokymasis yra naudojamas įvairiose skaičiavimo užduotyse, kur suprogramuoti aiškius, gerai veikiančius algoritmus yra sudėtinga ar neįmanoma.

Daugiausiai praktikoje naudojami mašininio mokymosi algoritmai yra daugiasluoksnis perceptronas[10], saviorganizuojantis žemėlapis[11], k-vidurkiu klasterizavimas[12] bei atraminių vektorių mašinos[13]. Daugelis minėtų algoritmų reikalauja pažymėtų[14] duomenų, tačiau tokiu atveju būtų neįmanoma automatizuoti kuriamos įsilaužimų aptikimo sistemos. Šiame darbe norima naudoti nereikalaujantį pažymėtų duomenų mašininio mokymosi algoritmą. Egzistuoja ne vienas algoritmas nereikalaujantis pažymėtų duomenų, tačiau šiame darbe pasirinktos atraminių vektorių mašinos.

3.1. Atraminių vektorių mašinos

Atraminių vektorių mašinos yra mašininio mokymosi metodų rinkinys naudojamas klasifikavimui, regresijai ir pašalinių duomenų aptikimui [SL16c]. Atraminių vektorių mašinų esmė yra rasti hiperplokštumą, kuri padalintų duomenis į dvi grupes. Atraminiai vektoriai - tai duomenų vienetai esantys arčiausiai hiperplokštumos, kuriuos pašalinus ji keistųsi, todėl jie yra esminiai duomenų aibėje. 4 pav. pavaizduoti duomenys esantys dviejose dimensijose, juos skirianti hiperplokštuma ir atraminiai vektoriai.

[10] angl. Multilayer perceptron – MLP
[11] angl. Self-organizing map – SOM
[12] angl. k-mean clustering
[13] angl. Support Vector Machines – SVM
[14] angl. labeled



4 pav. Hiperplokštuma ir atraminiai vektoriai

Jeigu kiekvienas iš turimų duomenų yra pažymėtas kaip priklausantis vienai iš dviejų klasių, atraminių vektorių mašinos algoritmas gali sukurti modelį, kuris galėtų atpažinti naujus duomenis ir priskirti juos atitinkamai klasei. Atraminių vektorių mašinos modelis yra atitikmuo taškų, išdėstytų taip, kad marža būtų kuo didesnė. Nauji duomenys tuomet pridedami į tą pačią erdvę ir priskiriami tai klasei į kurios pusę yra labiau pasislinkę nuo hiperplokštumos.

Jeigu turimi duomenys nėra pažymėti kuriai klasei priklauso, prižiūrimas mokymasis yra neįmanomas, taigi naudojamas neprižiūrimas mokymasis. Neprižiūrimas mokymasis klasterizuoja duomenis į grupes, ir naujus duomenis priskiria suformuotoms grupėms. Klasterizavimo algoritmas patobulina atraminių vektorių mašinas ir yra vadinamas atraminių vektorių klasterizavimu [BHS01]. Atraminių vektorių klasterizavimas naudojamas išskirtinių duomenų radimui. Vykdant SQL injekcijas kliento parametrai yra neįprasti, todėl pasiekti darbo tikslą bus naudojamas atraminių vektorių klasterizavimas.

3.2. Scikit-learn

Dėl aukšto lygio interaktyvumo ir brandžios ekosistemos Python programavimo kalba yra viena populiariausių kalbų mokslinei kompiuterijai [PVG11]. *Scikit-learn* yra Python kalbos paketas, kuris išnaudoja šios kalbos turtingą aplinką ir suteikia išplėtotas žinomų mašininio mokymosi algoritmų realizacijas [PVG11]. *Scikit-learn* paketo *OneClassSVM* atraminių vektorių mašina yra neprižiūrimo mokymosi algoritmas, kuris išmoksta *sprendimo funkciją* neįprastumų aptikimui: lyginant *sprendimo funkciją* su slenkstine reikšme nauji duomenys klasifikuojami kaip

įprasti arba išsiskiriantys nuo apmokymo duomenų rinkinio [SL16b]. *OneClassSVM sprendimo funkcija* yra imties atstumas iki skiriančios hiperplokštumos [SL16a].

4. Ypatybių atranka

Šiame skyriuje nagrinėjamos ypatybės pagal kurias apmokomas klasifikatorius, atsjiriantis įprastus kliento parametrus nuo SQL injekcijos puolimų.

Ypatybėms nagrinėti naudojami užklausų sąjungų ir aklųjų SQL injekcijų pavyzdžiai [CC13c]. Pavyzdinis žiniatinklis:

http://www.example.com/product.php?id=10

Užklausų sąjungų injekcija:

http://www.example.com/product.php?id=10 UNION SELECT 1,null,null--

Akloji SQL injekcija:

http://www.example.com/product.php?id=10' AND LENGTH(username)=N AND '1' = '1

4.1. Simbolių kiekis

Vykdant SQL injekcijos puolimą siunčiamo parametro simbolių kiekis žymiai padidėja. Pateiktuose pavyzdžiuose parametro *id* reikšmės simbolių kiekis padidėjo nuo 2 iki 29 ir nuo 2 iki 39.

4.2. Specialių simbolių kiekis

Vykdant SQL injekcijos puolimą gali būti naudojami specialūs simboliai kaip kabutės, skliausteliai, lygybės ir kiti ženklai. Pirmame pavyzdyje matome, kad vykdant SQL injekciją buvo prirašyti du brūkšneliai norint užkomentuoti toliau vykdomą SQL užklausos kodą. Antrame pavyzdyje vykdant SQL injekciją buvo prirašytos keturios kabutės, pora skliaustelių ir lygybės ženklas.

4.3. SQL raktažodžių kiekis

Vykdant SQL injekcijos puolimą gali būti naudojami SQL raktažodžiai. Pirmame pavyzdyje buvo panaudoti SQL raktažodžiai 4 kartus, antrame 3.

Užklausos kelias žiniatinklio programai nurodo į kokį veiksmą kreipiamasi. Veiksmui gali būti perduodamas įvairus kiekis parametrų. SQL injekcija gali būti vykdoma bet kuriame iš jų, todėl

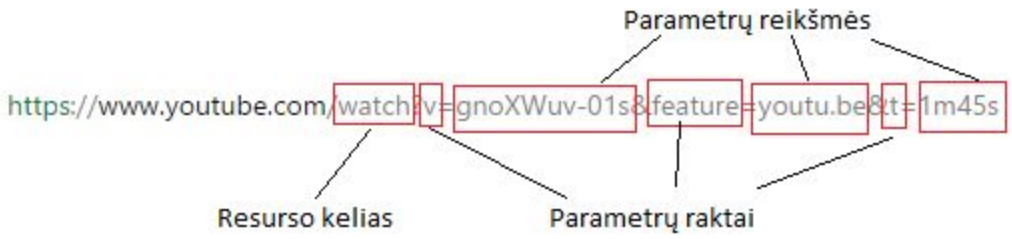
jeigu bent vienas parametras yra atpažįstamas kaip neįprastas - visa užklausa traktuojama kaip puolimas. Skirtingų programos kelių arba to paties kelio parametrų reikšmių ypatybės gali gerokai skirtis, todėl, vietoje kelio atskyrimo kaip ypatybės, nuspręsta apmokyti atskirus klasifikatorius kiekvieno kelio kiekvienam parametrui atskirai. Tai pagerins modelio tikslumą ir sumažins skaičiavimo sudėtingumą. Atraminių vektorių mašinos yra efektyvus įrankis, tačiau jų skaičiavimo ir talpos reikalavimai ženkliai auga didėjant mokymo vektorių skaičiui. Jų esmė yra kvadratinis programavimas. Scikit-learn paketo naudojamas kvadratinio programavimo sudėtingumas yra maždaug tarp $O(n \times m^2)$ ir $O(n \times m^3)$, kur n - ypatybių skaičius ir m - imties dydis. Imtį išskirsčius per k klasifikatorių sudėtingumas tampa tarp $O(n \times (\frac{m}{k})^2 \times k)$ ir $O(n \times (\frac{m}{k})^3 \times k)$.

5. Eksperimentas

Šiame skyriuje siūlomas ir realizuojamas įrankis, aptinkantis SQL injekcijos puolimus žiniatinklio programose, ir sistema, kurioje šis įrankis būtų naudojamas.

5.1. Siūloma sistema

Įrankis aptinkantis SQL injekcijos puolimus žiniatinklio programose yra įsilaužimų aptikimo sistema. Ji remiasi neprižiūrimo mašininio mokymosi būdu įprastų programos užklausų duomenimis ir geba atpažinti neįprastus kliento siunčiamus parametrus. Sistemai paruošti reikalingi duomenys apie HTTP užklausų resursų kelius, parametrų raktus ir jų reikšmes. Kiekvieno kelio kiekvienam parametro raktui sukuriamas atskiras klasifikatorius. Klasifikatoriui apmokyti reikalingas galimų parametro reikšmių rinkinys. Apmokymo metu iš kiekvienos reikšmės išgaunamos trys ypatybės: simbolių kiekį, specialių simbolių kiekį ir SQL raktažodžių kiekį. Turint apmokytą klasifikatorių bus apskaičiuojama kiek tiriamos užklausos parametro reikšmės ypatybės nutolusios nuo apmokymui skirtų. Paruošus sistemą bus vykdomos įprastos užklausos ir SQL injekcijos sistemai testuoti. Pagal gautus įverčius nustatoma ar sistema veiksminga.



5 pav. HTTP užklausa

Įsilaužimų aptikimo sistemai integruoti siūloma dviejų fazių sistema. Pirmoje fazėje išsaugomi įprastų užklausų duomenys ir pagal juos yra apmokomi įsilaužimų aptikimo sistemos klasifikatoriai. Antroje fazėje klientų užklausų duomenys perduodami įsilaužimų aptikimo sistemai, kuri

kiekvienam parametro raktui naudodama reikiamą klasifikatorių nustato ar užklausoje yra vykdomas SQL injekcijos puolimas.

5.2. Realizuota sistema

Žiniatinklio programa yra elektroninė parduotuvė, kuri buvo sukurta naudojant „Spree Commerce" paketą. „Spree Commerce" yra atviro kodo e-komercijos sprendimas su daugybe įrankių ir funkcijų skirtų sukurti ir palaikyti elektroninei parduotuvei. Pirmoje fazėje renkant įprastų užklausų duomenis yra išsaugoma parametro reikšmė - duomenų bazės laukas *parameter*, bei užklausos pavadinimas susidedantis iš programos klasės ir metodo apdorojančio užklausą pavadinimų ir parametro rakto - duomenų bazės laukas *request_name*. Veiksmai rekursiškai atliekami kiekvienam įdėtiniam parametrui. Įdėtiniai parametrai yra parametrai, susieti tarpusavyje tėviniu raktu.

5.3. Įrankiai

Žiniatinklio programai kurti naudoti įrankiai:

- Ubuntu 17.04 - operacinė sistema
- Ruby - programavimo kalba
- Ruby on Rails - žiniatinklio karkasas
- SQLite3 - reliacinė duomenų bazių valdymo sistema

Ypatybių gavyba ir mašininio mokymosi klasifikatorių apmokymas programuoti Python programavimo kalba. Naudoti įrankiai:

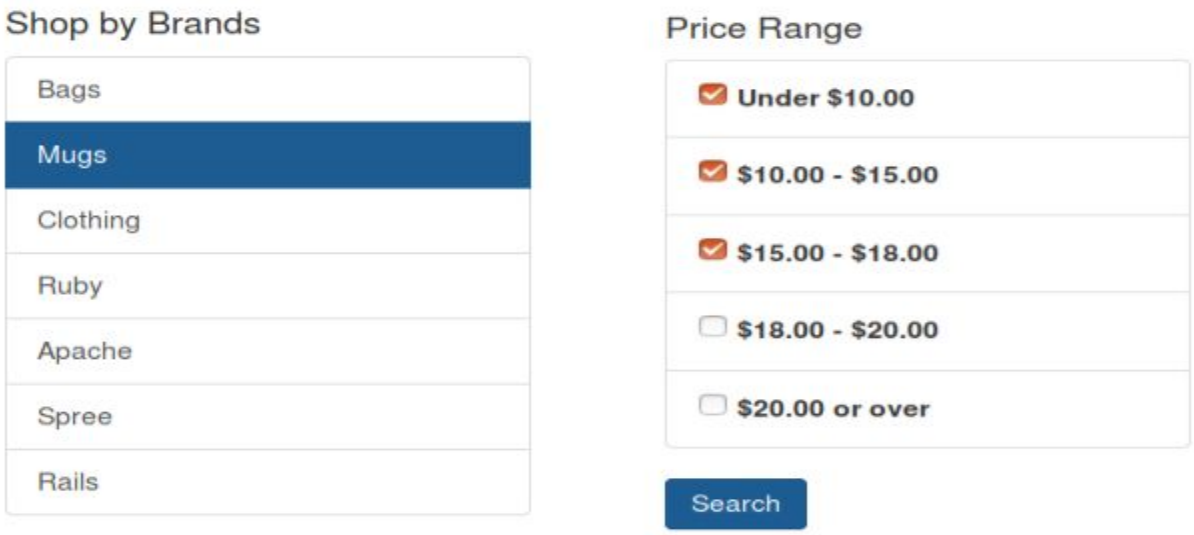
- Ubuntu 17.04 - operacinė sistema
- Python - programavimo kalba
- Scikit-learn - mašininio mokymosi paketas

5.4. Tiriamosios žiniatinklio funkcijos

Toliau apžvelgiamos žiniatinklio programos funkcijos ir pateikiama po pavyzdį kokie duomenys renkami vėlesniam klasifikatorių apmokymui. Pavyzdžiuose nurodomi kliento parametrai bei duombazėje išsaugomi duomenys.

Prekių filtravimas pagal rūšį ir kainos diapazoną

Pasirinkus prekių rūšį ir kainos diapazoną, žiūrėti 6 pav., išfiltruojamos prekės.



6 pav. Prekių filtravimas pagal rūšį ir kainos diapazoną

Pavyzdžiui, pasirinkus rūšį *Mugs* ir kainos diapazonus *mažiau nei $10, $10 - $15, $15 - $18,* serveriui siunčiama GET užklausa keliui /t/ perduodant parametrus[15]:

```
{
        ‘id’: ‘brands/mugs’,
        ‘search’: {
                ‘price_range_any’: [
                        ‘Under $10.00’, ‘$10.00 - $15.00’, ‘$15.00 - $18.00’
                ]
        }
}
```

Programoje užklausa apdorojama *Spree::TaxonsController* klasės *show* metodo. Užklausos parametrų raktai *id* ir *search[price_range_any]*[16]. Užklausai į duomenų bazę kiekvienam užklausos parametrui įrašoma po įrašą:

```
{
        'parameter': 'brands/mugs',
        'request_name': 'spree_taxons-show-id'
},
{
```

[15] pateikiama JSON formatu
[16] įdėtinis parametras

'parameter': '[\"Under $10.00\", \"$10.00 - $15.00\", \"$15.00 - $18.00\"]',
'request_name': 'spree_taxons-show-search-price_range_any'
}

Paieška pagal raktažodį ir prekės rūšį.

Pasirinkus prekių rūšį ir į įvesties lauką įrašius paieškos raktažodį, žiūrėti 7 pav., surandamos prekės.



7 pav. Paieška pagal raktažodį ir prekės rūšį.

Pavyzdžiui, pasirinkus rūšį *Mugs* ir įvedus į paieškos lauką *query*, serveriui siunčiama *GET* užklausa keliui */products* perduodant parametrus:

{
'taxon': 4,
'keywords': 'query'
}

Programoje užklausa apdorojama *Spree::ProductsController* klasės *index* metodo. Užklausos parametrų raktai *taxon* ir *keywords*. Užklausai į duomenų bazę kiekvienam užklausos parametrui įrašoma po įrašą:

{
'parameter': '4',
'request_name': 'spree_products-index-taxon'
},
{
'parameter': 'query',
'request_name': 'spree_products-index-keywords'
}

Prekės įdėjimas į krepšelį

Pridedant prekę į pirkinių krepšelį siunčiama *POST* užklausa keliui */orders/populate.* Pavyzdiniai perduodami parametrai:

{

```
    'variant_id': 13,
    'quantity': '1'
}
```

Programoje užklausa apdorojama *Spree::OrdersController* klasės *populate* metodo. Užklausos parametrų raktai *variant_id* ir *quantity*. Užklausai į duomenų bazę kiekvienam užklausos parametrui įrašoma po įrašą:

```
{
    'parameter': '13',
    'request_name': 'spree_orders-populate-variant_id'
},
{
    'parameter': '1',
    'request_name': 'spree_orders-populate-quantity'
}
```

Klasifikatorių apmokymas

Kliento siunčiamų prekių filtro parametrų reikšmių imtis priklauso nuo rūšių skaičiaus ir kainos apimties pasirinkimų skaičiaus. Generuojami duomenys kiekvienai galimai parametro reikšmei.

Paieškoje vienas kliento parametras taip pat priklauso nuo rūšių skaičiaus, tačiau kitam klientas pats įveda ieškomą raktažodį. Rūšių pasirinkimo duomenys buvo generuojami kiekvienai galimai reikšmei. Paieškos raktažodžių duomenys buvo generuojami atsižvelgiant į ypatybes, pagal kurias apmokomas klasifikatorius, t. y. simbolių kiekį, specialių simbolių kiekį ir panaudotų SQL raktažodžių kiekį. Generuojamų duomenų simbolių kiekis kito nuo 1 iki 15, specialių simbolių kiekis kito nuo 0 iki 2, bei SQL raktažodžių kiekis kito nuo 0 iki 2. Specialių simbolių kiekis nebuvo 0, nes yra galimybė vartotojui naudojantis paieška naudoti simbolį „-”. SQL raktažodžių kiekis nebuvo 0, nes yra galimybė, jog vartotojo paieškos raktažodyje atsitiktinai pasitaikys SQL raktažodis[17].

Prekės įdėjimo į krepšelį vieno parametro reikšmių imtis priklauso nuo visų prekių skaičiaus, kitas parametras - kiekis yra parenkamas vartotojo. Prekės pasirinkimo duomenys buvo generuojami kiekvienai galimai reikšmei. Kiekio parametro duomenys buvo vienženkliai ir dviženkliai skaičiai.

[17] pvz.: raktažodis *AND* žodyje „brand”

Kiekvienam parametrui surinkta ne mažiau 100 įrašų duomenų rinkiniai. Kiekvienam duomenų rinkiniui apmokomas atskiras klasifikatorius ir išsaugomas į failą vėlesniam naudojimui. Klasifikatoriui apmokyti naudojamos duomenų įrašų lauko parameter reikšmių ypatybės: simbolių kiekis, specialių simbolių kiekis ir pavartotų SQL raktažodžių kiekis. Specialių simbolių kiekiui nustatyti naudojamas String modulis [PSF17]. SQL raktažodžių kiekiui nustatyti buvo surinkti dažniausiai naudojami SQL raktažodžiai [Cod, W3S]. Klasifikatoriui apmokyti duodamas ypatybių grupių sąrašas: *[(y11, y12, y13), ..., (yn1, yn2, yn3)]*, kur y yra įrašo ypatybė, n įrašų kiekis.

Testavimas

Patikrinti užklausos neįprastumą naudojama klasifikatoriaus *sprendimo funkcija*. *Sprendimo funkcija* nurodo skaitiniu pavidalu kiek pateikiami duomenys yra nutolę nuo apmokymui skirtų duomenų. Įvertinus gaunamas *sprendimo funkcijos* reikšmes kiekvienam programos parametrui, bus nustatyta konstanta, kurią viršijanti *sprendimo funkcijos* reikšmė bus laikoma įsilaužimu. Testavimo tikslas patikrinti ar dažniausiai pasitaikančios SQL injekcijos laikomos įsilaužimu ir ar įprastos užklausos - ne.

Sistemai testuoti buvo surinkta SQL injekcijų [FM15] ir jos naudojamos siunčiant parametrus nepriklausomai nuo to ar žiniatinklio programa pažeidžiama SQL injekcijoms. Surinktos injekcijos nepriklausomai nuo testinės žiniatinklio programos duomenų bazių valdymo sistemos, nes norima patikrinti įsilaužimų aptikimo sistemos veiksmingumą.

' UNION SELECT id, username, password FROM users; --

IF((SELECT username FROM users LIMIT 1) = admin', 'ruby', 'rails')&order=asc

DROP sampletable;--

10;DROP members --

' HAVING 1=1 --

' GROUP BY table.columnfromerror1 HAVING 1=1 --

ORDER BY 1--

' UNION SELECT SUM(columntofind) FROM users--

'; insert into users values(1, 'hax0r', 'coolpass', 9) --

'AND ISNULL(ASCII(SUBSTRING((SELECT TOP 1 name FROM sysObjects WHERE xtYpe=0x55 AND name NOT IN(SELECT TOP 0 name FROM sysObjects WHERE xtYpe=0x55)),1,1)),0)>78--

';WAITFOR DELAY '0:0:10'--

') BENCHMARK(1000000000,MD5(1))

SQL injekcijos buvo daromos kiekvienam parametrui ir apdorojamos jam skirtu klasifikatoriumi. 2 lentelėje pavaizduota kaip kinta klasifikatorių *sprendimo funkcijos* darant SQL injekcijas ir paprastas užklausas.

2 Lentelė. Klasifikatorių sprendimo funkcijos reikšmės vykdant SQL injekcijas ir darant įprastas užklausas

Klasifikatorius	**SQL injekcijos**		**Įprastos užklausos**		**Siūloma slenkstinė reikšmė**
	Nuo	**Iki**	**Nuo**	**Iki**	
spree_taxons-show-id	-78.97	-79.49	-4.28	0	-5
spree_taxons-show-search-price_range_any	-18.99	-18.99	0	0	-0,1
spree_products-index-taxon	-38.53	-38.53	-4.11	0	-5
spree_products-index-keywords	-23.75	-23.75	-4.43	0	-10
spree_orders-populate-variant_id	-72.09	-72.09	0	0	-0,1
spree_orders-populate-quantity	-35.19	-35.19	0	0	-0,1

Sprendimo funkcijos reikšmės buvo suapvalintos iki šimtųjų. Jų ekstremumai priklauso nuo klasifikatoriaus apmokymui skirtos imties dydžio ir jos duomenų įvairovės. Vykdant SQL injekcijas klasifikatorių *sprendimo funkcijų* reikšmės artėja prie minimalios. Kiekvienam klasifikatoriui pasiūlytos slenkstinės *sprendimo funkcijos* reikšmės. Bent vieno užklausos parametro reikšmei gavus mažesnę *sprendimo funkcijos* reikšmę už slenkstinę, užklausa bus laikoma įsilaužimu. Visiems klasifikatoriams slenkstinės reikšmės siūlomos šiek tiek mažesnės už minimalią gautą įprastų užklausų *sprendimo funkciją*, išskyrus *spree_products-index-keywords* klasifikatoriui. Šiam

klasifikatoriui tenka paieškos parametro reikšmė, kuri gali būti labai įvairi, nes priklauso nuo vartotojo įvesties.

Klaidingi teiginiai[18] yra galimi kai įprastas kliento parametras per daug skiriasi nuo apmokymo duomenų. Eksperimento atveju *spree_products-index-keywords* klasifikatorius buvo apmokytas duomenimis iki 15 simbolių. Vykdant paiešką su 16 simbolių reikšme gaunama *sprendimo funkcijos* reikšmė mažesnė už slenkstinę. Tokią paiešką sistema atpažintų kaip SQL injekciją. Produkcinėje aplinkoje apmokant sistemą duomenys būtų daug įvairesni. Tokioje sistemoje būtų mažesnė galimybė gauti *klaidingą teiginį.*

Teisingi neiginiai[19] yra galimi kai SQL injekcija turi itin panašias ypatybes kaip klasifikatoriaus apmokymui skirtos reikšmės. Jeigu tarp apmokymo duomenų nėra SQL injekcijų - *teisingi neiginiai* nėra tikėtini. Kadangi eksperimento atveju sugeneruotų testinių duomenų ypatybės stipriai skiriasi nuo SQL injekcijų ypatybių, *teisingi neiginiai* neatsikartoja.

SQL injekcijų aptikimo veiksmingumas priklauso nuo įprastų užklausų programai duomenų. Kuo įvairesni duomenys - tuo *klaidingų teiginių* tikimybė mažesnė. Kuo duomenų ypatybės labiau skiriasi nuo SQL injekcijų ypatybių - tuo *teisingų neiginių* tikimybė mažesnė.

[18] angl. false positive
[19] angl. true negative

Rezultatai ir Išvados

Šiame darbe buvo pasiūlytas ir sukurtas prototipas įsilaužimų aptikimo sistemos, kuri analizuoja žiniatinklio kliento siunčiamus parametrus ir geba nuspręsti ar yra vykdomas SQL injekcijos puolimas. Tikslui pasiekti buvo įgyvendinti išsikelti uždaviniai.

1. Sistemai realizuoti nuspręsta naudoti neprižiūrimo mašininio mokymosi algoritmą.
2. Buvo išskirtos kliento siunčiamų parametrų ypatybės pagal kurias apmokomas mašininio mokymosi klasifikatorius: simbolių kiekis, specialių simbolių kiekis ir SQL raktažodžių kiekis.
3. Prototipo testavimui sukurta žiniatinklio programa - elektroninė parduotuvė.
4. Sugeneruoti žiniatinklio programos užklausų parametrų duomenys skirti testavimui.
5. Pagal testavimo duomenų ypatybes apmokyti klasifikatoriai.
6. Sistemai patikrinti naudotos populiarios SQL injekcijos. Pagal sugeneruotus testavimo duomenis prototipas sėkmingai atskyrė SQL injekcijas nuo įprastų parametrų.

Pritaikant neprižiūrimo mašininio mokymosi būdu veikiančią įsilaužimų aptikimo sistemą produkcinėje aplinkoje, būtų susiduriama su keletu iššūkių. Pirmiausia, klasifikatoriai turėtų būti apmokomi automatiškai, kad išleidus naujų programos galimybių nebūtų reikalinga apmokyti naujus klasifikatorius. Taip pat, *sprendimo funkcijos* slenkstinė reikšmė turėtų kisti dinamiškai kiekvienam klasifikatoriui, priklausomai nuo jo gaunamų įprastų užklausų *sprendimo funkcijos* reikšmių tikslumo. Galiausiai, eksperimentas parodė, kad generuojant dirbtinius testavimo duomenis yra susiduriama su *klaidingais teiginiais*, todėl būtina apmokymui naudoti gerą įprastų užklausų duomenų bazę. Duomenys turi būti įvairūs ir jų ypatybės turi skirtis nuo SQL injekcijų, kad būtų išvengta klaidų. Įveikus šiuos iššūkius siūloma SQL injekcijų aptikimo sistema gali būti ypač veiksminga.

Tolimesniems tyrimams siūloma ištirti SQL injekcijų aptikimo sistemos našumą, t.y. patikrinti klasifikatorių *sprendimo funkcijos* reikšmių gavimo laiką bei atmintyje užimamos vietos dydį. Gauti rezultatai parodytų sistemos resursų reikalavimus. Taip pat būtų naudinga atlikti eksperimentus su produkcinės aplinkos duomenimis. Tai pateiktų tikslesnius rezultatus ir galimus sistemos trūkumus.

Šaltiniai

[AI14] Al's Tech Tips. *SharePoint 2013 OWA: Server Error: We're sorry. An Error has occurred. We've logged the error...* Prieiga per internetą: http://alstechtips.blogspot.lt/2014/07/sharepoint-2013-owa-server-error-were.html. Žiūrėta 2017-05-27.

[BCA09] C. Bockermann, M. Apel, M. Meier. Learning sql for database intrusion detection using context-sensitive modelling. In International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. 2009.

[BHS01] A. Ben-Hur, D. Horn, HT. Siegelmann, V. Vapnik. Support vector clustering. Journal of machine learning research. 2001;2(Dec):125-37.

[CC13a] Creative Commons Attribution-ShareAlike. OWASP Top 10 – 2013. The ten Most Critical Web Application Security Risks. Prieiga per internetą: https://www.owasp.org/index.php/Top_10_2013-Top_10. Žiūrėta 2017-05-27.

[CC13b] Creative Commons Attribution-ShareAlike. The Open Web Application Security Project. Prieiga per internetą: https://www.owasp.org/index.php/Blind_SQL_Injection. Žiūrėta 2017-05-27.

[CC13c] Creative Commons Attribution-ShareAlike. Testing for SQL Injection (OTG-INPVAL-005). [žiūrėta 2017-05-27] Prieiga per internetą: https://www.owasp.org/index.php/Testing_for_SQL_Injection_(OTG-INPVAL-005)

[CMN16] Ceccato, M., Nguyen, C.D., Appelt, D. and Briand, L.C., 2016, August. SOFIA: an automated security oracle for black-box testing of SQL-injection vulnerabilities. In Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering (pp. 167-177). ACM.

[Cod] Codecademy. List of SQL commands.
[žiūrėta 2017-05-27] Prieiga per internetą:
https://www.codecademy.com/articles/sql-commands

[FM15] Ferruh Mavituna. SQL Injection Cheat Sheet. 2015
[žiūrėta 2017-05-27] Prieiga per internetą:
https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/

[JS00] A.K. Jones, R. S. Sielken. Computer system intrusion detection: A survey. Computer Science Technical Report. 2000.

[KP98] R. Kohavi, F. Provost. Glossary of Terms: Special Issue on Applications of Machine Learning and the Knowledge Discovery Process. 1998
[žiūrėta 2017-05-27] Prieiga per internetą: http://ai.stanford.edu/~ronnyk/glossary.html

[LHL13] Liao, Hung-Jen, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. "Intrusion detection system: A comprehensive review." Journal of Network and Computer Applications 36, no. 1 (2013): 16-24.

[Mar] Margaret Rouse. TechTarget. SQL injection definition. Prieiga per internetą: http://searchsoftwarequality.techtarget.com/definition/SQL-injection. Žiūrėta 2017-05-27.

[NM04] Norton, M., 2004. Optimizing pattern matching for intrusion detection. Sourcefire, Inc., Columbia, MD.

[PSF17] Python Software Foundation. Common string operations. 1990-2017
[žiūrėta 2017-05-27] Prieiga per internetą: https://docs.python.org/2/library/string.html

[PVG11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research. 2011;12(Oct):2825-30.

[SL16a] Scikit-learn developers. OneClassSVM. 2010-2016
[žiūrėta 2017-05-27] Prieiga per internetą:
http://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html#sklearn.svm.OneClassSVM.decision_function

[SL16b] Scikit-learn developers. One-class SVM with non-linear kernel (RBF). 2010-2016 [žiūrėta 2017-05-27] Prieiga per internetą: http://scikit-learn.org/stable/auto_examples/svm/plot_oneclass.html

[SL16c] Scikit-learn developers. Support Vector Machines. 2010-2016 [žiūrėta 2017-05-27] Prieiga per internetą: http://scikit-learn.org/stable/modules/svm.html

[SPS10] Stavroulakis, P. and Stamp, M. eds., 2010. Handbook of information and communication security. Springer Science & Business Media.

[TCC17] The Code Curmudgeon. SQLi Hall of shame. 2017 [žiūrėta 2017-05-27] Prieiga per internetą: http://codecurmudgeon.com/wp/sql-injection-hall-of-shame/

[Tim14] M. Tim Jones. Fight against SQL injection attacks. Prieiga per internetą: http://www.ibm.com/developerworks/library/se-sql-injection-attacks. Žiūrėta 2017-05-27.

[UOB17] Uwagbole, S. Ogbomon, W. J. Buchanan, L. Fan. Applied Machine Learning Predictive Analytics to SQL Injection Attack Detection and Prevention. 2017.

[W3S] W3Schools. SQL Quick Reference From W3Schools. [žiūrėta 2017-05-27] Prieiga per internetą: https://www.w3schools.com/sql/sql_quickref.asp

[XPS11] C. Xenakis, C. Panos, I. Stavrakakis. A comparative evaluation of intrusion detection architectures for mobile ad hoc networks. 2011.