REPUBLIKA E SHQIPËRISË

UNIVERSITETI POLITEKNIK I TIRANËS

FAKULTETI I TEKNOLOGJISË SË INFORMACIONIT

DEPARTAMENTI I INXHINIERISË INFORMATIKE

BOJKEN SHEHU

Për marrjen e gradës
"Doktor"
në "Teknologjitë e Informacionit dhe Komunikimit"
drejtimi "Inxhinieri Informatike"

DISERTACION

NJË SHTRESË EFIÇENTE QË MBRON TË DHËNAT DUKE IDENTIFIKUAR DHE PARANDALUAR SULMET ME INJEKTIM SQL

Udhëheqës Shkencor
PROF.DR. ALEKSANDËR XHUVANI

Tiranë, 2015

NJË SHTRESË EFIÇENTE QË MBRON TË DHËNAT DUKE IDENTIFIKUAR DHE PARANDALUAR SULMET ME INJEKTIM SQL

Disertacioni
i paraqitur në Universitetin Politeknik të Tiranës
Për marrjen e gradës
"Doktor"
në
"Teknologjitë e Informacionit dhe Komunikimit"
drejtimi Inxhinieri Informatike

nga
Z. Bojken Shehu
2015

Disertacioni i shkruar nga

Z. Bojken Shehu

DIND, Master i Shkencave, Universiteti Shtetëror Teknik Bauman i Moskës, Federata Ruse, 2010

I aprovuar nga

Juria

_____Kryetari, Juria e doktoratës

_____Anëtar, Juria e doktoratës

_____ Anëtar, Juria e doktoratës

_____ Anëtar, Juria e doktoratës

_____ Anëtar, Juria e doktoratës

I pranuar nga

_____Dekan, Fakulteti i Teknologjisë së Informacionit

Miratuar nga

Akademik Jorgaq Kaçani, Rektor i UPT

_____, Këshilli i Profesorëve, FTI

Abstrakt

NJË SHTRESË EFIÇENTE QË MBRON TË DHËNAT DUKE IDENTIFIKUAR DHE PARANDALUAR SULMET ME INJEKTIM SQL

Zhvillimet intensive të Teknologjisë së Informacionit dhe Komunikimit në dakadat e fundit, kanë bërë që informacioni dhe të dhënat të jenë asetet më të rëndësishme të një biznesi në ditët e sotme, pra duke i kushtuar një rëndësi të madhe informacionit, ruajtjes së tij, ne pa dyshim kemi bërë një hap të madh përpara në raport me zhvillimin e biznesit por edhe me konkurentët tanë. Web aplikacionet dhe bazat e të dhënave janë të ndjeshme ndaj një numri të madh sulmesh ndaj tyre. Një ndër sulmet më të rrezikshme dhe më të shpeshta që i ndodhin një web aplikacioni dhe një baze të dhënash në një websajt, janë sulmet me injektim SQL.

Problemi i sulmeve me injektim SQL është rritur për shkak të web aplikacioneve të shumta, të cilat pranojnë në hyrje, të dhëna nga përdorues të ndryshëm, të cilat i drejtohen bazës së të dhënave në formën e një instruksioni SQL, të ekzekutuara nga aplikacioni. Sulmuesi mund të manipulojë kërkesën që i shkon bazës së të dhënave dhe të përfitojë kalimin e pa autorizuar në bazën e të dhënave. Nëse sulmuesi kalon pengesën e parë, ai ka mundësi që të modifikojë dhe të menaxhojë të dhënat e bazës së të dhënave në favorin e tij. Por problem i madh vazhdon të mbetet siguria në shtresën e aplikacionit, e cila vazhdon të ngelet një nga pikat më kritike, ku përfshihet edhe kontrolli i saktësisë së të dhënave në hyrje, si muri i parë mbrojtës i çdo kërkese që i drejtohet bazës së të dhënave.

Mungesa e një shtrese sigurie çon në rrjedhjen e informacionit e cila vjen si pasojë e ekzekutimit të komandave të gabuara, të cilat dëmtojnë në mënyrë indirekte përfitimet e kompanisë dhe shtojnë sulmet mbi të. Mungesa e një shtrese mbrojtëse ndaj të dhënave në hyrje është një problem që edhe ne e hasëm në pyetësorin qe i kemi drejtuar disa kompanive në realitetin tonë, të trajtuar në kapitullin e tretë. Sulmet me injektim SQL janë lehtësisht të zbatueshme, prandaj për t'u mbrojtur nga to është një detyrë e vështirë, edhe për zhvilluesit me eksperiencë të aplikacioneve.

Zhvilluesit dhe sulmuesit janë në një garë të përjetshme midis tyre. Zhvilluesit përpiqen të zhvillojnë web aplikacione të sigurta nga sulmet e ndryshme dhe sulmuesit gjithmonë përpiqen të gjejnë një dobësi, nga ku mund të marrin ose të dëmtojnë aplikacionin ose të dhënat në një bazë të dhënash. Këto sulme kanë si qëllim kryesor të marrin informacionin konfidencial ose të dhënat personale, për të shkaktuar një dëmtim të bazës së të dhënave ose aplikacionit, për të treguar aftësitë e sulmuesve ose thjesht për të marrë kënaqësi e të bërit diçka që të tjerët nuk mund ta bëjnë.

Gjatë disertacionit u trajtuan në mënyrë të detajuar sulmet me injektim SQL, të cilat janë sulme që i ndihmojnë sulmuesit të hyjnë në mënyrë indirekte në bazën e të dhënave, në mënyrë të pa-

autorizuar, duke nxjerrë ose modifikuar informacionet me ndjeshmëri të lartë, nga bazat e të dhënave të organizatave të ndryshme. Këto sulme janë specifike, që do të thotë që secili sulm i përdorur për të sulmuar web aplikacionin dhe bazën e të dhënave ka karakteristikat e tij. Sulmuesit mund të përdorin metoda ose teknika të ndryshme për të arritur qëllimet e tyre.

U realizua një pasqyrim i realitetit të sulmeve me injektim SQL në Shqiperi, mbi bazën e grafikëve. Ky studim investigoi në realitetin tone rolin e sulmeve me injektim SQL. Strategjia e këtij kërkimi që është përdorur për të marrë informacionin e nevojshëm për këtë studim eshte ajo e pyetësorit.

Gjithashtu u realizua një studim mbi teknikat e identifikimit dhe parandalimit të sulmeve me injektim SQL, duke nxjerrë në mënyrë të qartë të metat dhe veçoritë e këtyre teknikave dhe gjithashtu u bë një studim, duke krahasuar teknikat identifikuese dhe parandaluese, në raport me sulmet me injektim SQL. Për ta kryer këtë studim, në fillim u përcaktuan llojet e ndryshme të sulmeve me injektim SQL dhe u investiguan teknikat e identifikimit dhe parandalimit të sulmeve me injektim SQL. Më pas, u krahasuan këto teknika, duke u bazuar në kriteret e zhvillimit dhe vlerësimit të tyre.

Pasi analizuam mirë sulmet me injektim SQL dhe një pjesë të madhe të teknikave të identifikimit dhe të parandalimit të këtyre sulmeve, u zhvillua një mekanizëm mbrojtës kundër sulmeve me injektim SQL. Metodologjia jonë e përdorur për t'u mbrojtur nga sulmet me injektim SQL, është e bazuar në krijimin e një shtrese e cila kontrollon dhe analizon kërkesat e dërguara nga klientet ose përdoruesit në drejtim të bazave të të dhënave, nëse janë të infektuara nga sulmet me injektim SQL apo jo. Shtresa mbrojtëse është lehtësisht e manovrueshme dhe tepër praktike, që do të thotë, që ajo mund të përdoret me çdo web aplikacion. Shtresa mbrojtëse analizon çdo kërkesë duke qëndruar midis serverit të aplikacionit dhe bazës së të dhënave, por ajo mund të veprojë edhe duke qënë pjesë e aplikacionit.

Eksperimentet e zhvilluara tregojnë që shtresa mbrojtëse është mjaft efiçente. Gjatë përdorimit të tre skanerave me shtresën tonë mbrojtëse, na rezulton që shtresa mbrojtëse është teper efiçente në parandalimin e sulmeve me injektim SQL në masën mbi 94%, duke përfshirë këtu edhe gabimet pozitive të secilit rast. Ndërkohë nga koha në sekonda, me dhe pa shtresën mbrojtëse e skanerit S1, e skanerit S2 dhe e skanerit S3, na rezulton që nuk kemi diferenca të mëdha në kohën që i duhet skanerave për të kontrolluar kërkesat për secilin nga të katërt testet e realizuara.

Fjalë kyçe: *Sulmet me injektim SQL, Siguria e web aplikacionit, Parandalimi i sulmeve me injektim SQL, Identifikimi i sulmeve me injektim SQL, Siguria e Bazës së të dhënave. Sulmet me injektim SQL në Shqipëri.*

Dedikim

Ky punim i dedikohet dy mbesave dhe dy nipave të mi……Ama, Amelia, Darien dhe Noel.

Falenderime

Në mënyrë të veçantë falenderoj familjen time, prindërit dhe dy vëllezërit e mi, bashkë me familjet e Tyre, për përkrahjen e vazhdueshme në të tre ciklet e studimit, një udhëtim aq sa i vështirë por edhe shumë i bukur. Pa mbështetjen e Tyre, gjithçka do të ishte e pamundur.

Dua të falenderoj përzemërsisht dhe të shpreh mirënjohjen time për udhëheqësin shkencor, Prof.Dr. Aleksander Xhuvani, për ndihmën e madhe dhe mbështetjen e çmuar që më ofroi përgjatë gjithë studimit tim, produkt i shumë orëve konsultimi, këshillimi dhe mbështetje nga ana e Tij.

Falenderoj përzemërsisht dekanen e Fakultetit të Teknologjisë së Informacionit, Prof.Dr. Rozeta Miho, për mbështetjen e vazhdueshme të Saj dhe ndjekjen me përkushtim të mbarëvajtjes së ciklit të studimeve të doktoraturës.

Gjithashtu dua të falenderoj të gjithë kolegët e Departamentit të Inxhinierisë Informatike, dhe në veçanti përgjegjësen e departamentit, Prof.Asoc Elinda Mece, për pozitivitetin, komunikimin dhe mbështetjen e vazhdueshme.

Dëshiroj të falënderojë edhe Profesorët e Këshillit të Fakultetit të Teknologjisë së Informacionit, për kohën që më kushtuan dhe për ndihmën e dhënë. Dy referimet shkencore të mbajtura para Tyre më kanë ndihmuar në thellimin dhe zgjerimin e të menduarit tim shkencor në drejtim të Teknologjisë së Informacionit dhe Komunikimit. Pa vërejtjet dhe sugjerimet e Tyre, teza nuk do të ishte e nivelit të duhur.

Shpreh mirënjohjen time për rektorin e Universitetit Politeknik të Tiranës, Akad.Prof.Dr. Jorgaq Kaçani për përkrahjen dhe mbështetjen që ka dhënë, për zhvillimin e stafit akademik.

Bojken Shehu
Tiranë, 2015

TABELA E PËRMBAJTJES

Lista e Figurave

Lista e Tabelave

Fjalor Terminologjik

SQL (angl. Structured Query Language) – është një gjuhë programimi që është e dizenjuar për menaxhimin e të dhënave në sistemet e menaxhimit të bazave të të dhënave.

XSS (angl. Cross-site Scripting) – sulmi i dytë më i rrezikshëm që i ndodh një web aplikacioni ose një baze të dhënash pas sulmeve me injektim SQL.

Query – kërkesë për informacion nga një bazë të dhënash

Union Query – është një nga sulmet me injektim SQL, që i ndodhin një web aplikacioni dhe do të thotë bashkimi i kërkesës origjinale me kërkesën e injektuar ose të infektuar.

Inference - është një nga sulmet me injektim SQL, që i ndodhin një web aplikacioni dhe qëllimi kryesor i sulmuesit, në sulmin me injektim SQL, bazuar në "Inference" është të ndryshojë sjelljen e bazës së të dhënave ose aplikacionit.

Piggy-backed Query - është një nga sulmet me injektim SQL, që i ndodhin një web aplikacioni dhe sulmuesi nëpërmjet sulmit me injektim SQL, bazuar në "Piggy-backed Query" nuk synon të bëj ndryshime në kuerin origjinal por kërkon të shtojë kuerin nëpërmjet injektimit.

HTTPS (angl. Hypertext Transfer Protocol Secure) – është një protokoll i komunikimit, për një komunikim të sigurt**ë** në rrjet, nëpërmjet internetit.

Gabimi pozitiv (angl. False Positive) - Me gabim pozitiv do të kuptojmë ato raste kur skanerat gjatë testimeve të tyre nxjerrin një numër të caktuar kërkesash të sulmuara, në rastin tonë nga sulmet me injektim SQL, por që në fakt këto kërkesa nuk janë të sulmuara nga këto sulme.

Gabimi negativ (angl. False Negative) - Me gabim negativ do të kuptojmë ato raste kur skanerat gjatë testimeve të tyre nuk identifikojnë një numër të caktuar kërkesash të sulmuara, në rastin tonë nga sulmet me injektim SQL, por që në fakt këto kërkesa janë të sulmuara nga këto sulme.

1. Hyrje

Në vitet e fundit, pjesa më e madhe e punëve tona të përditëshme në internet, janë të varura nga web aplikacionet, të lidhura me bazat e të dhënave, si për shëmbull veprimet në bankë, rezervimet apo blerja e produkteve të ndryshme nëpërmjet internetit. Pra, sot interneti eshtë infrastruktura më e përhapur e informacionit, duke u bëre një vënd grumbullimi i të dhënave.

Informacioni dhe të dhënat janë asetet më të rëndësishme të një biznesi në ditët e sotme, dhe duke i kushtuar një rëndësi të madhe informacionit, ruajtjes së tij, ne pa dyshim kemi bëre një hap të madh përpara në raport me zhvillimin e biznesit por edhe me konkurentët tanë.

Keto aplikacione të lidhura me bazat e të dhënave vijnë tek përdoruesit nëpërmjet websajteve të ndryshme. Këto websajte ndihmojnë përdoruesit për të krijuar llogari personale për veprimet e tyre të mëvonshme onlajnë [46], ose thjesht për të mbledhur te dhënat e tyre personale. Përsa kohë përdorimi i internetit është rritur, përdorimi i veprimeve onlajnë dhe proceseve të automatizuara është rritur gjithashtu. Dhe sot ne jemi të vetëdijshëm, që një numër i madh të dhënash që janë të ndjeshme dhe kritike kalojnë nëpër këto web aplikacione. Duke qëne se shkalla e informacionit dhe të dhënave të ruajtura nga portale të ndryshme eshtë rritur, po aq eshtë rritur edhe mënyra e sulmit për të marrë informacionin nga sulmuesit.

Zhvilluesit dhe sulmuesit janë në një garë të përjetshme midis tyre. Zhvilluesit përpiqen të zhvillojnë web aplikacione të sigurta nga sulmet e ndryshme dhe sulmuesit gjithmonë perpiqen të gjejnë një dobësi, nga ku mund të marrin ose të dëmtojnë aplikacionin ose të dhënat në një bazë të dhënash. Këto sulme kanë si qëllim për të marrë informacionin konfidencial ose të dhënat personale, për të shkaktuar një dëmtim të bazës së të dhënave ose aplikacionit, për të treguar aftësitë e sulmuesve ose thjesht për të marrë kënaqësi e të bërit diçka që të tjerët nuk mund ta bejnë.

Web aplikacionet janë të ndjeshme ndaj një numri të madh sulmesh. Një ndër sulmet më të zakonshme dhe më të shpeshta që i ndodhin një web aplikacioni dhe një baze të dhënash në një websajt, janë sulmet me injektim SQL. Duke u nisur nga projekti i sigurisë së web aplikacioneve (OWASP) [47], sulmet me injektim SQL qëndrojne në krye të listës prej 10 sulmeve më të shpeshta që i ndodhin një web aplikacioni ose një bazë të dhënash, për vitin 2013.

Për shembull, mashtrimet financiare, veprimet onlajnë në banka, blerjet onlajnë dhe shumë aktivitete të tjera shtetërore po pse jo edhe terrorismi kibernetik mund të jenë të kapshme nga sulmuesit nëpërmjet sulmeve me injektim SQL. Këto web aplikacione të cilat janë të mundshme për t'u prekur nga sulmet me injektim SQL, mund t'i lejojnë sulmuesit të marrë komplet kontrollin e bazës kryesore të të dhënave.

Nëpërmjet sulmeve me injektim SQL, sulmuesit kibernetikë mund të marrin kontrollin e plotë të bazës së të dhënave, që do të thotë, që ata janë në gjëndje të manipulojnë bazën e të dhënave për të bërë gjithçka që ata duan, ku përfshihen:

- ✓ Ta nxjerrin bazën e të dhënave jashtë përdorimit.
- ✓ Të shkarkojnë ose të shtojnë dokumente.
- ✓ Nëpërmjet një kërkimi në të kundërt, ata mund të kënë shumë shpejt adresat IP, dhe të sulmojnë sërish kompjuterat veç e veç.
- ✓ Mund të fshijnë, të ndryshojnë apo të shtojnë dokumente dhe në bashkepunim me sistemin operativ ata mund të lexojnë ose të shkruajnë në fajle të ndryshme.
- ✓ Vjedhjet onlajnë të websajteve në internet, duke ndryshuar çmimin e produktit ose të shërbimit, duke e bërë atë të negociushëm ose falas per klientët.
- ✓ Duke vendosur një emër të gabuar në një kartë krediti, dhe duke e përdorur atë në një moment tjetër ose duke e shitur atë.

Pra, duhen gjetur disa mënyra ose rregulla që duhen vendosur në çdo websajt ose aplikacion për ta bërë atë të sigurtë nga sulmet me injektim SQL. Shumë web aplikacione mund të sulmohen sepse të dhënat e përdoruesve në hyrje kanë ardhur në një menyrë jo të sigurtë. Pjesa më e madhe e web sajteve kërkojnë të jenë në gjendje të ruajnë ose të magazinojnë informacionin në formën e të dhënave, me qëllim menaxhimin e tyre. Për ta realizuar këtë është e nevojshme prania e një baze të dhënash, pjesë e së cilës janë disa funksione kryesore si, krijimi, leximi, rinovimi dhe fshirja e të dhënave, të cilat ndhmojnë në menaxhimin e të dhënave. Por çdo websajt do të përdorë instruksionet SQL, për t'i vënë në lëvizje këto funksione me qëllim për të shtuar, për të paraqitur, për të rinovuar dhe fshirë informacionin.

1.1 Sulmi me Injektim SQL

Sulmi me injektim SQL, ndodh kur sulmuesi manipulon kërkesën ose në gjuhën teknike kuerin që i drejtohet një baze të dhënash, duke e modifikuar llogjikën e saj. Me fjalë të tjera, në bazën e të dhënave do të kalojë një komandë SQL, e ndryshuar, e cila do t'i krijojë mundësi sulmuesit të modifikojë të dhënat e një baze të dhënash. Për t'i ndaluar sulmet me injektim SQL, zhvilluesit duhet të jenë të vetëdijshëm të dijnë se si të kontrollojnë të dhënat në hyrje, për të rritur sigurinë në Teknologjinë e Informacionit dhe Komunikimit.

Sulmet me injektim SQL, janë një nga mekanizmat e shumtë që një sulmues mund të përdorë për të marrë të dhënat nga kompani ose organizata të ndryshme private ose publike. Nëse kjo gjë do të ndodhë kundër sistemeve të informacionit të një spitali, ku në bazën e të dhënave gjënden të dhënat e pacientëve, kjo do të bëjë që informacioni konfidencial [48] mund të dalë jashtë dyerve të spitalit dhe të thyejë privatësinë e pacientit duke dëmtuar rëndë reputacionin e tij. Sulmuesi nëpërmjet sulmeve me injektim SQL nuk thyen vetëm sigurinë dhe gjithashtu merr

informacionin e nevojshëm nga baza e të dhënave, por gjithashtu mund të ndryshojë skemën e bazës së të dhënave dhe përmbajtjen e saj.

I pari që e ka hasur dhe dokumentuar ekzistencën e sulmeve me injektim SQL, është Jeff Forritsal, në vitin 1998, i cili në intervistën e dhënë [1] në nëntor të vitit 2013, shpjegon se si ai u përball me diçka të pazakontë, me problemin e sulmeve me injektim SQL, ndërkohë që ishte duke punuar dhe kërkuar se si të ndërhynte në një Server. Duhet theksuar që në vitet 1990, shumë pak kompani në fushën e teknologjisë së informacionit kishin filluar të përdorinin bazat e të dhënave me Microsoft SQL Server, sepse pjesa më e madhe në atë kohë përdornin bazat e të dhënave bazuar në Microsoft Access. Në intervistën e dhëne për revistën SecurityPlanet [50], ai nuk habitet që edhe pse kanë kaluar 16 vjet, sulmet me injektim SQL, vazhdojnë të jenë shqetësimi më i madh në fushën e sigurisë.

Problemi i sulmeve me injektim SQL është rritur për shkak të web aplikacioneve të shumta të cilat pranojnë në hyrje, të dhënat e përdoruesve, të cilat i shkojnë bazës së të dhënave në formën e një instruksioni SQL, të ekzekutuara nga aplikacioni. Sulmuesi mund të manipulojë kërkesën që i shkon bazës së të dhënave dhe të përfitojë kalimin e pa autorizuar në bazën e të dhënave, dhe nëse sulmuesi e arrin këtë, më pas do të jetë e mundur që ai të modifikojë dhe të menaxhojë të dhënat e bazës së të dhënave në favorin e tij. Por problem i madh vazhdon të mbetet dhe siguria në shtresën e aplikacionit, e cila vazhdon të ngelet një nga pikat kritike, ku përfshihet dhe kontrolli i saktësisë së të dhënave në hyrje, si muri i parë mbrojtës i çdo kërkese që i drejtohet bazës së të dhënave.

Mungesa e një shtrese sigurie do të çonte në rrjedhjen e informacionit e cila do të vinte si pasojë e ekzekutimit të komandave të gabuara, të cilat do të demtonin në menyrë indirekte përfitimet e kompanisë dhe do të shtonin sulmet mbi të. Mungesa e një shtrese mbrojtëse të të dhënave në hyrje është një problem që edhe ne e hasëm në pyetësorin qe i kemi drejtuar disa kompanive në realitetin tonë, të trajtuar në kapitullin e tretë. Sulmet me injektim SQL janë lehtësisht të zbatueshme, prandaj për t'u mbrojtur nga to është një detyrë e veshtirë, edhe për zhvilluesit me eksperiencë të aplikacioneve.

Në vitet e fundit ka pasur një rritje të vrullshme të web sajteve, dhe siç u permënd më lart cilësia e sigurisë gjithashtu është rritur. Zhvilluesit e aplikacioneve e kanë të nevojshme të ndërhynë në kodin burim për ta ekzaminuar atë, me qëllim zgjidhjen e problemit, edhe pse herë pas here kjo mund të ketë kosto të lartë.

Ka shumë faktorë që mund të ndikojnë vëndimet për të rritur sigurinë e një aplikacioni, si koha, kostot financiare dhe aftesitë që zhvilluesit e këtyre aplikacioneve kanë. Kufizimet financiare janë një faktor i rëndësishëm, që shpesh herë janë edhe problemi më i madh sidomos në realitetin shqipëtar ku kompanitë e ndryshme nuk mund ta përballojnë nga ana financiare të bëjnë një pasqyrë të detajuar të sigurisë. Për të ulur koston financiare, që vjen nga harxhimi i kohës së punës dhe nga kontolli i gabimeve në formë manuale, proçesi duhet të jetë pjesërisht ose

plotësisht i automatizuar duke përdorur teknikën e analizës statike, një mjet testimi që në mënyre automatike skanon dhe shfaq dobësite e shkaktuara nga të dhënat në hyrje të përdoruesve në një websajt. Sulmet me injektim SQL janë ndër sulmet më të rrezikshme që mund t'i ndodhin një aplikacioni dhe një baze të dhënash dhe vazhdojne të qëndrojnë në krye të listës si një ndër sulmet më të përdorshme edhe në vitin 2014.

Sulmet me injektim SQL, nuk janë vetëm një prej sulmeve më kritike, por edhe si një nga mënyrat më popullore për të nxjerrë të dhënat nga një bazë të dhënash. Kërkimet e deri tanishme që kanë të bëjnë me sulmet me injektim SQL janë fokusuar në proçedurat e automatizimit të gjetjes së dobësive të një web aplikacioni duke përdorur mënyra dhe teknika kodimi të ndryshme për mbrojtjen nga sulmet. Ekziston gjithashtu një studim i zgjeruar dhe që përshkruan në mënyrë të detajuar se si kodi në gjuhë të ndryshme programimi duhet të përdoret, dhe si ta shkruajmë kodin me qëllim që të parandalojmë sulmet me injektim SQL dhe sulmet e tjera që i drejtohen një web aplikacioni me qëllim sulmin e të dhënave në bazën e të dhënave [49].

1.2 Diskutimi i Problemit dhe Objektivat e Studimit

Duke ju referuar të dhënave deri tani, është e qartë se numri i sulmeve me injektim SQL është gjithmonë në rritje, prandaj tema e studimit është shumë aktuale, është me interes të diskutohet dhe të hidhet dritë mbi këtë problem. Sulmet me injektim SQL janë shfaqur kur websajtet u lidhën me bazën e të dhënave, dhe sigurisht këto sulme vështirë se do të zhduken përfundimisht, për sa kohë kemi një aplikacion te lidhur me bazën e të dhënave, do të kemi dhe një sulmues që do ta sulmojë atë. Ka shumë gjëra që duhen bërë me qëllim që t'i parandalosh apo t'i mbrosh websajtet dhe bazat e të dhënave nga sulmet me injektim SQL. Më poshtë është bërë një studim për teknikat e parandalimit dhe identifikimit të sulmeve me injektim SQL.

Qellimi kryesor i kësaj teme studimi është të studiojë dhe t'i parandalojë sulmet me injektim SQL, pse këto sulme janë akoma një nga problemet më të mëdha të sigurisë që nga viti 1998, pavarsisht nga fakti që kompanitë apo organizatat e dijnë që ky problem ekziston. Individët po përdorin shërbime të ndryshme të ofruara nga kompani dhe organizata të ndryshme, dhe pjesa më e madhe e njerëzve i besojnë kompanive perpunimin e të dhënave të tyre konfidenciale. Kur ka një rrjedhje informacioni, pra kur dikush ka sulmuar, siguria e kompanisë është në pikëpyetje, dhe rrjedhimisht emri i kësaj kompanie do të bjerë ne tregun ku ajo vepron. Problemi i sulmeve me injektim SQL nuk është një problem i vogël dhe duhet gjithsesi të merret shumë seriozisht.

Ku studim synon të kontribuojë në fushën kërkimore të sigurisë në Teknologjinë e Informacionit dhe Komunikimit, dhe të angazhojë të gjitha burimet e mundshme për të investiguar dhe të kërkoj një përgjigje pse sulmet me injektim SQL, mbeten akoma një nga problemet më të mëdha të sigurisë në Teknologjine e Informacionit dhe Komunikimit. Kjo tezë studimore do të kontribuojë gjithashtu si një pikë referimi për sulmet me injektim SQL, një thirrje për t'u zgjuar për zvilluesit e aplikacioneve, për krijuesit e bazave të të dhënave, por gjithashtu edhe për kërkuesit shkencorë në fushën e sigurisë, pedagogët, dhe sigurisht për studentët që dizenjojnë dhe programojnë bazat e të dhënave dhe kërkojnë të ndërtojnë websajte të sigurta.

Qëllimi i këtij studimi është krijimi i një shtrese efiçente që mbron të dhënat nga sulmet me injektim SQL.

Objektivat kryesorë të studimit janë:

- ✓ Të investigojmë mbi sulmet me injektim SQL, duke shfrytëzuar të gjithë literaturën dhe burimet e mundshme.

- ✓ Cili është realiteti i këtyre sulmeve në vëndin tonë, sa mbrohen kompanitë ose organizatat nga këto sulme.

- ✓ Cilat janë sulmet me injektim SQL dhe karakteristikat e tyre.

- ✓ Studim i literaturës mbi teknikat e identifikimit dhe parandalimit të sulmeve me injektim SQL dhe karakteristikat e tyre. Analiza krahasuese e këtyre teknikave në raport me sulmet me injektim SQL.

- ✓ Krijimi i një shtrese efiçente që mbron të dhënat duke parandaluar sulmet me injektim SQL.

- ✓ Vlerësimi i performancës se arritur nga zgjidhja e propozuar dhe vlerësimi i pergjithshëm i rezultateve të arritura dhe diskutimi kritik i çështjeve dhe problematikave që mbeten për t'u zgjidhur në të ardhmen.

1.3 Kufizimet e Disertacionit

Ekzistojnë dy tipe sulmesh më të përhapura, ndër shumë sulme, që janë më të shpeshta dhe më të dëmshme në lidhje me informacionin, dhe me të dhënat konfidenciale. Ashtu si u përmënd më lart, i pari është sulmi me injektim SQL dhe tjetri është sulmi XSS (Cross Site Scripting). Sulmet me injektim SQL ekzekutojnë një instruksion në bazën e të dhënave dhe sulmi XSS ekzekuton një kod në makinën e viktimës.

Ngjashmëria midis këtyre dy sulmeve është që të dy këto sulme shfrytëzojne dizenjimin e dobët të kontrollit të të dhenave, për të ekzekutuar kodin e modifikuar. Diferenca midis tyre konsiston në atë që, ç'pjesë e të dhënave të shumta që rrjedhin manipulohen dhe ku është ekzekutuar kodi. Në sulmet me injektim SQL të dhënat kalojnë nga përdoruesit në bazën e të dhënave dhe në sulmet XSS të dhënat kalojnë midis viktimës dhe sulmuesit. Sulmi i shpeshtë me injektim SQL, ndodh kur kodi është i ruajtur në serverin e synuar, në rastin tonë në bazën e të dhënave.

Kjo tezë do të studjojë dhe vlerësojë vetëm sulmet me injektim SQL dhe jo sulmet XSS ose ndonjë sulm tjetër që mund të ekzekutohet.

Shumë gjuhë të ndryshme programimi janë në pëdorim dhe kjo tezë nuk do të vlerësojë se sa një gjuhë specifike programimi ndikon mbi sulmet me injektim SQL.

1.4 Përfituesit e Këtij Studimi

Të gjithë zhvilluesit që punojnë në zhvillimin e aplikacioneve të ndryshme të cilët janë të lidhur me bazat e të dhënave do të përfitojnë nga ky studim, veçanërisht zhvilluesit të cilët nuk kanë dijeni për problemet që lidhen me sigurinë e të dhënave. Pjesa akademike, kërkuesit dhe studentët të cilët janë të përfshirë në fushën Teknologjisë së Informacionit dhe Komunikimit mund t'i ndihmojë ky studim për të kuptuar më mirë sulmet me injektim SQL.

1.5 Struktura e Disertacionit

Struktura e organizimit të këtij disertacioni është si më poshtë:

Kapitulli i parë përshkruan në terma të përgjithshëm disa nga elementët kryesore të disertacionit. Percakton bashkësinë të cilit ky studim i drejtohet. Hidhen bazat shkencore mbi të cilat bazohet ky studim, të cilat përmënden shkurt në terma të thjeshtuara. Këtu jepet një tablo e përgjithshme e sulmeve me injektim SQL, trajtohet ne vija të përgjithshme diskutimi i problemit dhe objektivat kryesore te studimit. Gjithashtu bëhet i qartë fokusimi i studimit vetëm mbi sulmet me injektim SQL, dhe jo mbi sulmet e tjera.

Kapitulli i dytë përmban një trajtim të thelluar të sulmeve me injektim SQL, duke trajtuar me shëmbuj secilin sulm me injektim SQL. Këto sulme, të cilat do të ndërthuren në punën e këtij disertacioni janë: Tautologji, kueri i ndërtuar në mënyrë jokorekte llogjikisht, kueri union, kueri piggy-backed, procedurat e ruajtura, inferencë, injektimi i verbër, sulmi i kohës dhe kodimi alternativ.

Kapitulli i tretë përmban një studim mbi raportin që kanë kompanitë dhe organizatat publike ose private në vendin tonë, mbi sulmet me injektim SQL. Në fund të kapitullit bëhet një analizë e detajuar e rezultateve.

Kapitulli i katërt trajton gjerësisht literaturën e përdorur për parandalimin dhe identifikimin e sulmeve me injektim SQL, duke u bazuar në mbi 95 artikuj të botuar në konferenca dhe revista nga autorë të ndryshëm në këtë fushë. Në këtë kapitull analizohen teknikat e parandalimit dhe ato të identifikimit të sulmeve me injektim SQL. Bëhët krahasimi i këtyre teknikave në raport me karakteristikat e tyre dhe gjithashtu në raport me sulmet me injektim SQL.

Kapitulli i pestë paraqet në mënyrë të detajuar skemën e propozuar për ndërtimin e shtresës mbrojtëse, implementimin dhe zbatimin e saj.

Kapitulli i gjashtë paraqet një vlerësim të gjërë eksperimental të shtresës së propozuar për parandalimin e sulmeve me injektim SQL. Eksperimentet janë zhvilluar me ndihmën e tre skanerave që testojnë efiçencën e shtresës tonë mbrojtëse në raport me parandalimin e sulmeve me injektim SQL.

Kapitulli i shtatë paraqet një analizë kritike te punës së zhvilluar duke nxjerrë në pah pikat e forta dhe të dobëta të punimit. Gjithashtu, përmban dhe një sërë drejtimesh për punimet e ardhshme në përmisimin e shtresës mbrojtëse të paraqitur.

2. Llojet e Sulmeve me Injektim SQL

Në këtë kapitull do të trajtohet një studim [41], [45] i kryer nga ne, mbi sulmet me injektim SQL, të cilat janë teknika që i ndihmojnë sulmuesit të hyjnë në mënyrë indirekte në bazën e të dhënave, në mënyrë të pa autorizuar, duke nxjerrë ose modifikuar informacionet me ndjeshmëri të lartë, nga bazat e të dhënave të organizatave të ndryshme. Këto sulme janë specifike, që do të thotë që secili sulm i përdorur për të sulmuar web aplikacionin, bazën e të dhënave ka karakteristikat e tij. Sulmuesit mund të përdorin metoda ose teknika të ndryshme për të arritur qëllimet e tyre. Në këtë kapitull ne do të trajtojmë llojet e sulmeve me injektim SQL, duke trajtuar dhe shëmbuj për secilin sulm.

2.1 Tautologji : Sulmi me Injektim SQL

Qëllimi kryesor i përdorimit të sulmit me injektim SQL, bazuar në tautologji nga sulmuesi është injektimi i kodit në një ose disa instruksione të kushtëzuara, duke i bërë këto instruksione që të jenë gjithmonë të vërteta. Me anë të tautologjisë, sulmuesi kërkon të thyej fjalëkalimin e përdoruesve duke vendosur parametra të injektuar, me qëllim nxjerrjen e të dhënave nga baza e të dhënave. Thyerja e fjalëkalimit është shëmbulli më i zakonshëm i këtij lloj sulmi me injektim SQL. Ndryshimi i kushtit në tautologji, pra në vërtetësinë e tij mund të shkaktojë kthimin e të dhënave në favor të sulmuesit. Pasojat e këtij sulmi varen se sa përdoren brënda aplikacionit rezultatet e një kueri.

Për shëmbull:

Select *mbiemri* ***From*** *perdoruesit* ***Where*** *mbiemri='$mbiemri'* ***And*** *fjalekalimi='$fjalekalimi';*

Në rastin kur përdoruesi është legjitim, me mbiemrin si 'shehu' dhe me fjalëkalimin si 'passbojken', kueri do të marri këtë formë:

Select *mbiemri* ***From*** *perdoruesit* ***Where*** *mbiemri='shehu'* ***And*** *fjalekalimi='passbojkenn';* (nuk kemi injektim)

Për shëmbull:

Marrim rastin kur kemi injektim bazuar në sulmin me injektim SQL tautologji, kur sulmuesi vendos një kod të injektuar [' OR '1'='1] si input për fushën e përdoruesit dhe fjalëkalimit, në këtë rast kueri do të marri formën e mëposhtme:

Select *mbiemri* ***From*** *perdoruesit* ***Where*** *mbiemri= ' 'OR '1'='1'* ***And*** *fjalekalimi = ' 'OR '1'='1';*

Kodi i injektuar në kushtin [' OR '1'='1] transformon në mënyrë të plotë kushtin 'Where' në një tautologji. Për sa kohë kushti është një tautologji, kueri do të vlerësohet si i vërtetë për cdo rresht në tabelë dhe do ti kthejë të gjithë këto rreshta. Si përfundim sulmuesi do të futet ne sistem me identitetin e parë të rekordit të kthyer nga kueri SQL.

Por gjithashtu një sulmues mund ta perdorë kodin e injektuar duke përdorur linkun e mëposhtem:

http://toolsmarket-al.com/?PerdoruesiId=1 ‘ OR ‘1’=’1

Operatori i kushtëzuar OR do ti bashkëngjitet instruksioneve SQL dhe kueri do të jetë gjithmonë i vërtetë, kështu që do të kemi instruksionin e mëposhtëm:

***Select** InfoPerdoruesi **From** TabelaPerdoruesi **Where** PerdoruesiId=’1’ OR ‘1’=’1’*

Në rastin e manipulimit të kushtit “Where”, sulmuesit manipulojnë kodet SQL, duke ndryshuar kushtin “Where” të kodit SQL, për të përfituar rezultate të ndryshme në krahasim me ato të një kodi normal SQL. Sulmuesi manipulon kushtin “Where” duke futur një tautologji dhe duke ç’aktivizuar kushtin për vendosjen e fjalëkalimit (password-it) me anë të përdorimit të një komenti.

Në SQL, për të treguar një koment, pëdoren dy vizat horizontale “--“. Sulmuesi mund të përdorë simbolin e komentit për të shkurtuar kuerin. Por çdo gjë që vjen pas dy vijave “--“ do të trajtohet si koment dhe nuk do të ekzekutohet nga serveri i bazës së të dhënave.

Për shëmbull:

Le të marrim një kueri. Supozojmë se kemi një formë autentifikimi ku një përdoruesi i kërkohet të vendosë kredencialet e tij për të fituar akses në një faqe të caktuar. Ai vendos emrin e përdoruesit në fushën “përdoruesi” dhe fjalëkalimin e tij në fushën “fjalëkalimi”. Në këtë mënyrë, në bazën e të dhënave ku ruhet informacioni mbi këtë përdorues formohet në mënyrë dinamike një kueri.

Kueri që gjenerohet dinamikisht është:

***Select** * **From** EmriTabeles **Where** Perdoruesi=‘bojken’ **And** Fjalekalimi= ‘shehu;’*

Në këtë kueri “bojken” dhe “shehu;” janë respektivisht përdoruesi dhe fjalëkalimi që futen nga përdoruesi për të parë të dhënat e llogarisë personale. Sulmuesi vendos ['OR 1=1 --] në fushën përdoruesi dhe lë fushën fjalëkalimi bosh.

Në këtë rast kueri do të bëhej:

***Select** * **From** EmriTabeles **Where** Perdoruesi="OR 1= –“ **And** Fjalekalimi=’’ ‘’*

Thonjëza teke e vendosur nga përdoruesi ç’aktivizon thonjëzën hapëse, që është pjesë e kuerit që gjenerohet dinamikisht. OR 1=1, që është një tautologji, kënaq kushtin “Where” për çdo rekord. Kështu që, kur ky kueri të ekzekutohet do të na kthejë informacion për të gjithë përdoruesit e tabelës mbi të cilën është bërë kueri.

Nga sa pamë më lartë, vemë re që sulmuesi që sulmon bazuar në sulmin me injektim SQL tautologji, kodi i injektuar i tij gjithmonë do të fillojë me apostrof [‘], i ndjekur nga operatori i kushtëzuar OR. Operatori i kushtëzuar OR do të ndiqet nga instruksione të tjera që gjithmonë do

të shkojnë drejt të vërtetës. Pra në rastin tonë për të parandaluar sulmin me injektim SQL, bazuar në tautologji është e domosdoshme qe në modelin tonë mbrojtës, ne të quajmë elementë të padëshiruar apostrofin [‘] dhe operatorin e kushtëzuar OR, në këtë mënyrë sulmuesi nuk do të ketë mundësi sulmi, pra nuk do të rrezikojë aplikacionin dhe bazën e të dhënave.

2.2 Kueri i Ndërtuar Llogjikisht Gabim: Sulmi me Injektim SQL ose Ndryshe Përdorimi i Mesazheve të Gabimit për të Marrë Emrin e Bazës së të Dhënave dhe Kolonave

Në një web aplikacion të sigurtë, nëse baza e të dhënave gjeneron ndonjë mesazh gabimi atëherë aplikacioni duhet t’i menaxhojë këto gabime dhe nuk duhet t’i shfaqë ato për përdoruesin. Sidoqoftë, në shumicën e aplikacioneve gabimet e web-it të bazave të të dhënave shfaqen për përdoruesin. Ky lloj sulmi me injektim SQL, i lejon sulmuesit te mbledhi informacion të rëndesishëm për tipin dhe strukturën e një baze të dhënash të një web aplikacioni.

Sulmi me injektim SQL “logically Incorrect Query” është konsideruar si një hap bazë për të mbledhur informacionin edhe për sulmet e tjera, nga të cilat mund të sulmohet sërish. Ky sulm ndodh nga kthimi i mesazheve të gabuara nga baza e të dhënave si pasojë e një kueri jokorektë.

Këto mesazhe të gabuara të bazës së të dhënave shpesh herë përmbajnë informacion të nevojshëm dhe të dobishëm që e lejon sulmuesin të identifikojë parametrat vulnerabël të një web aplikacioni dhe të skemës së bazës së të dhënave. Sulmuesi gjatë sulmimit me sulmin me injektim SQL “logically Incorrect Query” përpiqet të injektojë instruksionet nëpërmjet të cilave shkakton një gabim në sintaksë ose një gabim llogjik në bazën e të dhënave sipas qëllimit të tij.

Kjo sjellje mund të përdoret në mënyrë efektive nga sulmuesit për të gjeneruar mesazhe gabimi nga serveri i bazës së të dhënave, të cilat do të ishin shumë ndihmuese për sulmuesin, që të kuptonte strukturën e bazës së të dhënave. Me këto mesazhe sulmuesi më vonë mund të formulojë kueri, për të përfituar informacion të pa autorizuar.

Në shëmbullin e mëposhtëm sulmuesi shkakton një gabim duke vepruar si mëposhte:

- ✓ Hapi i pare.

Linku origjinal:

http://www.toolsmarket-al.com/veglat/?id_nav=2234

- ✓ Hapi i dytë.

Injektimi i sulmit SQL:

http://www.toolsmarket-al/veglat/?id_nav=2234’

- ✓ Hapi i tretë.

Mesazhi i gabuar:

***Select** loginid **From** perdoruesit **Where** id=2234\'*

Nëpërmjet mesazhit të gabuar sulmuesi mund të kuptojë fare thjeshtë emrin e tabelës dhe fushat e saj në bazën e të dhënave, që në rastin konkret jane: *loginid*; *perdoruesit* dhe *id*. Nga informacioni i marrë sulmuesi mund të sulmojë akoma më tepër bazën e të dhënave, duke përdorur dhe sulmet e tjera.

Më poshtë marrim nje shëmbull ku kueri i injektuar përpiqet të nxjerrë përdoruesin e parë te tabelës [id='u'] nga tabelat e shumta të bazës së të dhënave, dhe më pas përpiqet ta konvertojë emrin e tabelës në një numër të plotë.

***Select** KlientId **From** KlientTabela **Where** LoginId='' **And** Fjalekalimi=''**And** Id=convert (int,(**Select** top 1 emri **from** sysobjects **where** Id='u'))'.*

Supozohet se aplikacioni është duke përdorur Microsoft SQL Server, për të cilin tabela e të dhënave është sysobjects. Keshtu që sulmuesi ka marrë informacion të nevojshëm përsa i përket bazës së të dhënave.

Ka disa rrugë për të performuar në mënyrë ilegale ose jo korrekte një kueri.

Për shëmbull:

Vendosja në mënyrë jokorekte të apostrofit ['], ose duke përdorur operatorin AND për të kryer një llogjikë jokorekte, gjithashtu edhe ORDER BY etj. Pra në rastin tonë për të parandaluar sulmin me injektim SQL, bazuar në "Logically Incorrect Query" është e domosdoshme që në modelin tonë mbrojtës ne të quajmë elementë të padëshiruar apostrofin ['] dhe operatorin e kushtëzuar AND, ORDER BY etj.

2.3 Kueri Union : Sulmi me Injektim SQL

"Union" është një komand SQL e cila punon duke kombinuar dy kueri. Nëpërmjet sulmit me injektim SQL, bazuar në "Union Query" sulmuesi duke përdorur komandën UNION bën bashkimin e kuerit të krijuar nga ai, me kuerin origjinal më qëllim marrjen nga baza e të dhënave një grup të dhënash, që janë rezultat i bashkimit të kuerit origjinal me kuerin e injektuar. Sulmuesi mund të perdorë sulmin me injektim SQL, bazuar në "Union Query" duke përdorur linkun.

Për shëmbull:

http://toolsmarket-al.com/?PerdoruesiId=' UNION <SQL statement>

Linku i mësipërm do të paraqesë instruksionet SQL të mëposhtme:

***Select** Perdoruesit **From** PerdoruesitTabela **Where** PerdoruesiId = "UNION < SQL statement>*

Tabela *"PerdoruesiTabela"* me sa duket, nuk do të ketë një të dhënë për *"PerdoruesiId"* të barabartë me ", prandaj ajo do të kthej vlerën "null" por instruksioni i dytë SQL do të ekzekutohet.

Në shëmbullin e mëposhtëm një sulmues mund të injektojë tekstin e mëposhtëm në një fushë për t'u loguar:

'UNION ***Select*** *Perdoruesi, Fjalekalimi* ***From*** *PerdoruesiInfo* ***Where*** *PerdoruesiEmri='fti'--*

e cila do të japi si rezultat kuerin e mëposhtëm:

Select *** ***From*** *LlogariaPerdoruesit* ***Where*** *Llogaria= '' UNION* ***Select*** *Fjalekalimi* ***From*** *PerdoruesiInfo* ***Where*** *PerdoruesiEmri='fti'—***And** pass="*

Duke supozuar që nuk gjëndet një login i barabartë me ", ashtu si dhe me lart kueri i parë kthen bashkësine "null", ndërsa kueri i dytë kthen të dhënat nga tabela *"PerdoruesiInfo"*. Në këtë rast baza e të dhënave do të kthejë kolonën *"Fjalekalimi*" për perdoruesin *"fti"*. Baza e të dhënave merr rezultatin e të dy kuereve, i bashkon ato dhe i kthen përsëri tek aplikacioni. Ne shumë aplikacione, efekti i këtij operacioni është që vlera për *"Fjalekalimi"* është shfaqur së bashku me informacionin për përdoruesin.

Në shëmbullin e mëposhtëm kodi është ekzekutuar nga serveri:

Select *Emri, Tel* ***From*** *PerdoruesiTabela* ***Where*** *Perdoruesiid=$id*

Më poshtë pasi sulmuesi injekton në vlerën Id kodin e mëposhtëm:

$id= 1 UNION ALL ***Select*** *NumriKartes, 1* ***From*** *TabelaKartes*

do të kemi kuerin:

Select *Emrin, Tel* ***From*** *PerdoruesiTabela* ***Where*** *Perdoruesiid= 1 UNION ALL* ***Select*** *NumriKartes, 1* ***From*** *Tabela Kartes.*

Kueri do ti bashkohet kuerit origjinal me të dhënat e kartave të kreditit të të gjithë përdoruesve.

Nga sa pamë më lart vemë re që sulmuesi që sulmon me sulmin me injektim SQL, bazuar ne "Union Query", kodi i injektuar prej tij gjithmonë do të ketë në përbërje të tij UNION, i ndjekur nga një istruksion SQL.

Pra në rastin tonë per të parandaluar sulmin me injektim SQL, bazuar në "Union kueri" është e domosdoshme ndër të tjera, që në modelin tonë mbrojtës ne të quajmë elementë të padëshiruar karakterin UNION, në këtë mënyrë sulmuesi nuk do të ketë mundësi sulmi, pra nuk do të rrezikojë aplikacionin dhe bazën e të dhënave.

2.4 Piggy-backed Kueri: Sulmi me Injektim SQL

Sulmuesi nëpërmjet sulmit me injektim SQL, bazuar në "Piggy-backed Kueri" nuk synon të bëj ndryshime në kuerin origjinal por kërkon të shtojë kuerin nëpërmjet injektimit. Ky është edhe ndryshimi me sulmet e tjera me injektim SQL, sepse sulmuesit nuk përpiqen te modifikojnë kuerin origjinal të propozuar, por në vënd të kësaj, ata perpiqen të përfshijnë në kuerin origjinal, një kueri të ri. Duke qënë se baza e të dhënave merr kërkesa të shumta SQL, kueri i parë është kueri i propozuar nga aplikacioni, i cili performon në mënyrë normale, më pas vinë kuerit e injektuar, të cilët performojnë si shtesë e të parit. Nëse gjithcka shkon me sukses, sulmuesit në mënyrë virtuale mund të vendosin çdo komandë SQL dhe t'i ekzekutojnë ato bashkë me kuerin origjinal. Nga lloj i bazës së të dhënave varet shumë dhe se sa vulnerabël eshte ky lloj sulmi [52].

Për Shëmbull:

Nëse sulmuesi vendos *[';drop tabela klientet --]* në fushën e fjalëkalimit, aplikacioni do të gjenerojë kuerin e mëposhtëm:

Select *LoginId* ***From*** *KlientetId* ***Where*** *LoginId='bojken'* ***And*** *Fjalekalimi=''; Drop Tabela Klientet--'****And*** *Id=3345*

Pas ekzekutimit të kuerit të parë siç e përmëndëm më lart, baza e të dhënave do të ndeshet me (;) dhe do të ekzekutohet direkt kueri i dytë. Rezultati i ekzekutimit të kuerit të dytë, do të sjellë tek sulmuesi të dhënat e përdoruesve, pra mund të shkaterojë të dhënat e rëndësishme të një kompanie.

Për Shembull:

Duke përdorur sulmet me injektim SQL, të diskutuara me shëmbuj më lart, sulmuesi mund të jetë në gjëndje të sigurojë emrat e përdoruesve të autorizuar. Për më tepër edhe në rastin e sulmit me injektim SQL, të bazuar në "Piggy-backed Kueri", sulmuesi përdor emrat e përdoruesve të autorizuar si input në fushën e përdoruesit, dhe mund të perdori kodin e mëposhtëm në fushën e fjalëkalimit për të sulmuar.

Për shëmbull:

pass= ' Or (Select Count(*) From perdoruesi)=34 And ''='

Kodi më i plotë do të jetë si mëposhtë:

Select *Emri* ***From*** *Perdoruesit* ***Where*** *Emri='Perdoruesi1'* ***And*** *pass= '= '* ***Or(Select Count(*) From*** *Perdoruesi)=34* ***And*** *''='';*

Nëse ky kueri do të shkojë gjithmonë drejt së vertetës atëherë sulmuesi do të krijojë një ide që në sistem jane ekzaktësisht 34 përdorues. Nëse kueri do të shkojë drejt së gabuarës atëherë kushti do të jetë jokorekt. Nëse në rastin e mësipërm do të përdornim *'Insert into'* dhe nëse kushti do të

shkonte gjithmonë drejt të vërtetës atëherë sulmuesi mund të vendosë të dhëna në mënyrë të suksesshme brënda një baze të dhënash.

Për shëmbull:

Vëmë re që kur sulmuesi përdor linkun për të sulmuar, kodi vepron si një skript në anën e serverit, duke i lejuar përdoruesve të marrin informacionin për punonjësit, të grumbulluar në një tabelë me emrin *PerdoruesiTabela*, në një bazë të dhënash MySQL. Tre fragmente të kodit janë paraqitur më poshtë. Marrja e të dhënave në hyrje nga përdoruesit:

```
var PerdoruesiId
PerdoruesiId=Request.form("PerdoruesiId")
```

Krijimi i instruksioneve SQL dhe ekzekutimi i tyre:

```
var sqlstmt= "Select PerdoruesIinfo From PerdoruesiTabela Where PerdoruesiId=' '+PerdoruesiId+" ''';
$result= mysql_query(sqlstmt)
```

Dhe paraqitjen e rezultatit:

```
while ($row=mysql_fetch_assoc($result)){
echo $row['PerdoruesInfo';]
```

Që klienti të përdori kodin e mësipërm duhet të plotësojë një dritare dhe informacioni komunikon me serverin e aplikacionit nëpërmjet linkut, i cili do të ketë këtë formë:

http://toolsmarket-al.com/?PerdoruesiId=21

Ky input ose e dhënë në hyrje do të shkaktojë paraqitjen e informacionit të përdoruesve, atë të përdoruesit me Id të barabartë me 21, gjithmonë do të shfaqet diçka vetëm nëse ka të dhëna për këtë përdorues në tabelën e përdoruesve, pra në rast se ky përdorues ekziston.

Por shikojmë rastin kur përdoruesi përdor linkun e mëposhtem për të sulmuar:

http://toolsmarket-al.com/?PerdoruesiId=21'; Drop Tabela PerdoruesiTabela --"

Në këtë rast, për të komunikuar me serverin e aplikacionit, do të kemi nje kueri që i drejtohet bazës së të dhënave i cili do të shfaqet në formën e mëposhtme:

Select *PerdoruesiInfo* ***From*** *PerdoruesiTabela* ***Where*** *PerdoruesiId= '21'; Drop Tabela PerdoruesiTabela --.*

Kjo mund të shkaktojë fshirjen e tabelës së përdoruesve nga sulmuesi.

Pra në rastin tonë për të parandaluar sulmin me injektim SQL, bazuar ne "Piggy-backed Kueri" është e domosdoshme që në modelin tonë mbrojtës ndër të tjera, ne të quajmë elementë të padeshiruar pikëpresjen [;] e cila sigurisht me aq sa pamë më lart do të ndiqet gjithmonë nga instruksionet SQL.

2.5 Procedurat e Ruajtura: Sulmi me Injektim SQL

Sulmuesit nëpërmjet sulmeve me injektim SQL, bazuar në procedurat e ruajtura, përpiqen ti ekzekutojnë këto procedura, prezente në një bazë të dhënash. Një procedurë e ruajtur është një grup komandash SQL, që kompilohen dhe ruhen në serverin e bazës së të dhënave. Bazat e të dhënave kanë të vendosura një numër të caktuar procedurash të ruajtura që rrisin funksionalitetin e bazës së të dhënave dhe lejojnë bashkëveprimin me sistemet operative. Aplikacionet klienteliste mund ti ekzekutojnë procedurat e ruajtura pa pushim, pa i dërguar ato në serverin e bazës së të dhënave ose pa i kompiluar ato.

Për sa kohë procedurat e ruajtura përmbajnë instruksione SQL, kjo shfrytëzohet nga sulmuesit për të sulmuar kodin e tyre. Këto procedura kthejnë vlera të vërteta ose të gabuara duke treguar nëse kredencialet e përdoruesit janë korekte.

Gjatë një sulmi me injektim SQL, sulmuesi injekton [' ; Shutdown; --] në fushën e përdoruesit ose të fjalëkalimit. Ky injektim bën që procedurat e ruajtura të gjenerojnë kuerin e mëposhtem:

Select *llogarite* ***From*** *perdoruesit* ***Where*** *login='bojken'* ***And*** *Fjalekalimi=' ';Shutdown;--*

Nga shëmbulli i mesipërm vëmë re, se sulmuesi sulmon në mënyrë të njejtë, si në sulmin me injektim SQL, bazuar në "Piggy-backed Kueri". Në fillim kueri është ekzekutuar normalisht, dhe më pas ekzekutohet pjesa tjetër e injektuar e kuerit, e cila shkakton rënien e bazës së të dhënave.

Shëmbulli i mësipërm tregon se procedurat e ruajtura mund të jenë vulnerabël po aq sa dhe sulmet në kodin e një aplikacioni të zakonshëm.

2.6 Inference ose Ndërhyrja: Sulmi me Injektim SQL

Qëllimi kryesor i sulmuesit, në sulmin me injektim SQL, bazuar në "Inference" është të ndryshojë sjelljen e bazës së të dhënave ose aplikacionit. Kemi dy sulme me injektim SQL, të njohura tashmë të bazuar në sulmin me injektim SQL, "Inference": Injektimi i verbër dhe sulmet ndaj kohës.

2.6.1 Injektimi i Verbër: Sulmi me Injektim SQL

Sulmi me injektim SQL, bazuar në injektimin e verbër ndodh kur programisti gjatë zhvillimit të një aplikacioni harron të fshehi një gabim. Ky sulm është pothuajse i njëjtë me sulmet e tjera, ndryshimi është se, shpesh herë zhvilluesit e web aplikacioneve i fshehin gabimet, të cilat do të ndihmonin sulmuesit për të sulmuar bazën e të dhënave. Në këtë situatë sulmuesi do të jetë përballë një faqeje të gjeneruar nga programisti, në vënd të një mesazhi të gabuar. Kjo gjë e veshtirëson sulmin por nuk e bën atë të pamundur.

Duke përdorur sulmin me injektim SQL, bazuar në injektimin e verbër, sulmuesi mund të verifikojë në disa mënyra, nëse një kërkesë e dërguar në serverin e bazës së të dhënave do të shkojë drejt së vertetës apo të gabuares. Nëse sulmuesi do të ketë një faqe teke, e cila paraqet një informacion me Id-në e dhënë si parameter, ai mund të performojë disa teste të thjeshta për të kuptuar nëse kjo faqe është vulnerabël ndaj sulmeve me injektim SQL.

Supozojmë se kemi linkun e mëposhtëm:

http://toolsmarket-al.com/produkti.php?id=2

dhe kueri i gjeneruar me id=2 do të jetë:

***Select** * **From** Produkti **Where** id=2*

Sulmuesi do të përpiqet të injektojë ndonjë kueri, e cila me siguri nuk do të kthejë diçka mbrapsht.

Për shëmbull:

http://toolsmarket-al.com/produkti.php?id=2 and 1=2

Kueri i gjeneruar nga linku që përmëndëm më lart do të jetë:

***Select** * **From** Produkti **Where** id=2 **And** 1=2*

që do të thotë, që kueri nuk do të kthejë asgjë. Në shëmbullin e mëposhtëm, le të supozojmë se kemi një formë autentifikimi, ku një përdoruesi i kërkohet të vendosë kredencialet e tij për të fituar akses në një faqe të caktuar.

Pra sulmuesi ka dy mundësi injektimi në fushën "përdoruesi" dhe në fushën "fjalëkalimi". Në këtë mënyrë, në bazën e të dhënave ku ruhet informacioni mbi këtë përdorues formohet në mënyrë dinamike një kueri:

***Select** llogaria **From** perdoruesit **Where** id= '2' and 1 =0 -- AND pass = AND pin=0 **Select** llogaria **From** perdoruesit **Where** login= 'bojken' and 1 = 1 -- AND pass = AND pin=0*

Nëse aplikacioni do të jetë i sigurtë, pra do të jetë vulnerabël ndaj sulmeve me injektim SQL, me siguri të dy kuerit do të rezultojnë të pasuksesshëm, për shkak të kontrollit të inputit ose të dhënave në hyrje. Por nëse nuk kemi një mekanizëm kontrolli të të dhënave në hyrje, pra të inputit nga zhvilluesi i aplikacionit atëherë, sulmuesi mund të ketë shanse të mira për sulm.

Në këtë rast sulmuesi do të tentojë me kuerin e parë dhe do të marrë mbrapsht një mesazh gabimi sepse 1=0. Në këto kushte sulmuesi nuk arrin ta kuptojë nëse gabimi ishte nga mungesa e kontrollit të të dhënave në hyrje të përdoruesve ose inputit apo nga një gabim llogjik në kueri. Në këto kushte sulmuesi tenton me kuerin e dytë i cili gjithmonë kthen kushtin në të vertetë. Nëse

më pas nuk do të shfaqet një gabim llogjik atëherë ai do të ketë mundësi sulmi në fushat e logimit.

Nga sa pamë më lart, vihet re që sulmuesi që sulmon, bazuar në injektimin e verbër, kodi i injektuar prej tij gjithmonë do të ketë në perbërje të tij operatorin AND, por shumë sulmues përdorin edhe operatorët e tjerë të kushtëzuar.

Pra, në rastin tonë për të parandaluar sulmin me injektim SQL, bazuar ne Injektimin e verbër, është e domosdoshme që në modelin tonë mbrojtës ndër të tjera, ne t'i quajmë operatorët e kushtëzuar të padëshiruar, në këtë mënyrë sulmuesi do të ketë pak mundësi sulmi, pra nuk do të rrezikojë aplikacionin dhe bazën e të dhënave.

2.6.2 Sulmet ndaj Kohës: Sulmi me Injektim SQL

Sulmi me injektim SQL, bazuar në sulmin ndaj kohës, i mundëson sulmuesit të mbledhë informacion për bazën e të dhënave duke u nisur nga monitorimi i vonesave kohore të pergjigjeve të bazave të të dhënave. Gjithashtu sulmuesi përdor instruksionet e kushtëzuara *If-Then-Else* për të injektuar kuerin. Gjithashtu përdoret fjala çelës *"Waitfor"* e cila i shkakton bazës së të dhënave një vonesë me kohë të specifikuar.

Si për shëmbull në kuerin e mëposhtëm:

declare @ varchar (7000) select @ = db_bojken () if (ascii (substring (@s, 1, 1)) & (power (2, 0))) > 0 waitfor delay '0:0:7'

Nga sa shikojmë më lart, baza e të dhënave do të krijojë një pausë për shtatë sekonda. Gjithashtu për të vonuar përgjigjet e serverit nëse shprehja është e vërtetë përdoret *Benchmark(),* si në shëmbullin e mëposhtëm:

Benchmark(4000000,Encode('MSG','by 7 seconds'))

Funksioni *Encode* do të ekzekutohet 4000000 herë. Do të na duhet vetëm një moment që procesi të mbarojë, kjo gjë do të varet nga performanca dhe ngarkesa e serverit të bazës së të dhënave. Duke u nisur nga mënyra se si mund ta shikoj sulmuesi, një gjë shumë e rëndësishme është specifikimi i një numri sa më të lartë përsëritjesh në funksionin *Benchmark(),* e cila duhet të ndikojë dukshëm në kohën e përgjigjes së serverit.

Pra, për të parandaluar sulmin me injektim SQL, bazuar në sulmin ndaj kohës, është e domosdoshme ndër të tjera, që në modelin tonë mbrojtës, *Waitfor, If, Else, Benchmarch, etj*, të mënjanohen, në këtë mënyrë sulmuesi do të ketë pak mundësi sulmi, pra nuk do të rrezikojë aplikacionin dhe bazën e të dhënave.

2.7 Kodimi Alternativ: Sulmi me Injektim SQL

Sulmet me injektim SQL, bazuar në kodimin alternativ kanë si objektiv kryesor shmangjen e identifikimit. Nuk krijojnë ndonjë mënyrë të re sulmi mbi web aplikacionin, por thjesht është një

teknikë që u lejon sulmuesve të shmangin teknikat e identifikimit dhe parandalimit dhe të shfrytëzojnë dobësitë që mbase nuk janë të eksploruara. Pra teksti i injektuar është i modifikuar në mënyrë të tillë që të shmangi identifikimin nga kodimi mbrojtës ose nga teknikat parandaluse të automatizuara.

Në sulmin me injektim SQL, bazuar në kodimin alternativ sulmuesit modifikojnë kuerin e injektuar duke përdorur kodimin alternativ si: heksadecimal, ASCII, dhe Unicod-i. Sepse në këtë mënyrë ata kanë mundësi ti shpëtojne zhvilluesve të teknikave mbrojtesë, të cilët skanojnë të dhënat në hyrje, inputin për karaktere speciale [53].

Për shëmbull, në rastin e mëposhtëm sulmuesi përdor ”char (44)” në vënd që të përdori një karakter.

***Select** * **From** Emri **Where** Login= “ **And** Fjalekalimi =’ ‘; exec(char(0x73567574545k889r))*

Ky shëmbull përdor funksionin Char () dhe kodimin heksadecimal ASCII. Funksioni Char () pranon karakterin ose karakteret e kodimit heksadecimal dhe kthen të njëjtin karakter ose karaktere. Pjesa e pafundme e numrave në pjesën e dytë të injektimit është kodimi heksadecimal ASCII i stringut. Ky string i koduar transformohet në një komand shutdown nga baza e të dhënave, kur ai ekzekutohet.

Në shëmbullin e mëposhtëm për t’u shmangur nga mbrojtja, sulmuesit kanë përdorur metoda alternative për të koduar sulmet e tyre.

$login = mysql_query (“Select * From perdoruesit Where (Perdoruesit_Id=’” .mysql_real_escape_string ($_Post [‘Perdoruesit_Id’]) . “’) and (pass_word=’” .mysql_real_escape_string ($_Post [‘pass_word’]) . “’)”);

Pra, nga sa vumë re me lart, skanimi i zakonshëm dhe teknikat identifikuse nuk arrijnë të kontrollojnë të gjitha stringjet e koduara, kështu që këto sulme kalojnë të paidentifikuara.

Pra, për të parandaluar sulmin me injektim SQL, bazuar në kodimin alternativ është e domosdoshme ndër të tjera, që në modelin tonë mbrojtës, *Char(), ASCII(), HEX(), UNHEX*, etj, që përdoren për të koduar stringun, të mënjanohen, në këtë mënyrë sulmuesi do të këtë më pak mundësi sulmi, pra nuk do të rrezikojë aplikacionin dhe bazën e të dhënave.

2.8 Analiza dhe Konkluzionet për Kapitullin e Dytë

Në këtë kapitull, u trajtua një studim i thelluar i sulmeve me injektim SQL, duke i trajtuar me shëmbuj secilin sulm me injektim SQL, duke u vënë herë pas here dhe në pozitat e një sulmuesi. Sulmet me injektim SQL, të cilat do të ndërthuren në punën e këtij disertacioni janë: Tautologji, kueri i ndërtuar në mënyrë jo korekte llogjikisht, kueri union, kueri piggy-backed, procedurat e ruajtura, Inference, injektimi i verbër, sulmi ndaj kohës dhe kodimi alternativ.

Ndër konkluzionet kryesore të kapitullit dhe gjatë studimit për teknikat e ndryshme parandaluese dhe identifikuese SQL, ne vumë re, që shumica e sulmeve nuk janë të izoluara, që do të thotë, ato janë akoma më të suksesshme nëse përdoren nga sulmuesit në kombinim, gjithmonë në varësi të qëllimit që ka sulmuesi.

3. Studimi mbi Sulmet me Injektim SQL në Shqipëri

Ky studim investigon në realitetin tonë, pse sulmet me injektim SQL vazhdojnë të mbeten një nga problemet më të mëdha të kompanive dhe organizatave. Strategjia e këtij kërkimi që është përdorur për të marrë informacionin e nevojshëm për këtë studim eshte ajo e pyetësorit. Qëllimi i këtij kapitulli është të pasqyrojë mbi bazën e grafikëve realitetin e sulmeve me injektim SQL në Shqipëri

3.1 Strategjia e Kërkimit

Ky kërkim studimor është realizuar nëpërmjet një pyetësori i cili përmban 9 pyetje të llojeve të ndryshme, më saktë në dy drejtime. Drejtimi i parë i pyetjeve ka të bëjë me marrjen e një informacioni të përgjithshëm për kompaninë apo organizatën që po i përgjigjet pyetësorit tonë. Qëllimi kryesor i pytjeve tona në këtë drejtim është që të marrim informacion nëse është kompani relativisht e madhe apo e vogël, pra se sa është numri i punonjësve që punojnë në kompani, nëse kjo kompani bën pjesë në sektorin publik apo atë privat. Drejtimi i dytë i pyetjeve ka të bëjë me menyrën se si këto kompani apo organizata e njohin apo e zbatojnë Teknologjinë e Informacionit në ambjentin e tyre. Qëllimi kryesor i këtij grupi pyetjesh në pyetësorin tonë është që të marrim informacionin bazë, në lidhje me mënyrën se si këto kompani apo organizata i kanë ndërtuar websajtet e tyre, mbi sigurinë e tyre, dhe mbi të gjitha se sa këto kompani apo organizata rrezikohen nga sulmet me injektim SQL, dhe nëse janë rrezikuar, çfarë masash kanë marrë për ti parandaluar këto sulme me injektim SQL. Janë mbledhur të dhëna nga një grup shumë i gjërë përgjigjesh. Ky është i pari studim, mbi sulmet me injektim SQL, në kompanitë dhe organizatat shqiptare. Të gjitha kompanitë dhe organizatat që morrën pjesë në këtë pyetësor zhvillojnë aktivitetin e tyre brënda territorit shqipëtar. Pyetjet që janë formuluar në këtë pyetësor kërkonin përgjigje të formave të ndryshme.

3.2 Përmbajtja e Pyetësorit

Ky pyetësor i cili ka si qëllim për të kuptuar gjëndjen e sulmeve me injektim SQL në vëndin tonë, është dërguar në 36 kompani dhe institucione, ku është plotësuar nga vetëm 89 % prej tyre, pra vetëm 32 prej tyre i janë përgjigjur pyetësorit tonë. Të gjitha kompanitë dhe institucionet që plotësuan pyetësorin e paraqitur prej nesh, deklaruan që kane një websajt. Pothuajse gjysma, rreth 66% e atyre që ishin pjesë në pyetësor ishin të vetëdijshëm për sulmet me injektim SQL, dhe rreth 34% e tyre nuk kishin dijeni për to. Nga 32 kompanitë dhe institucionet që janë përfshirë në studimin tonë, 23 prej tyre, pra 72% i përkasin sektorit privat dhe 9 prej tyre, pra 28% në atë publik, figura 3.1.

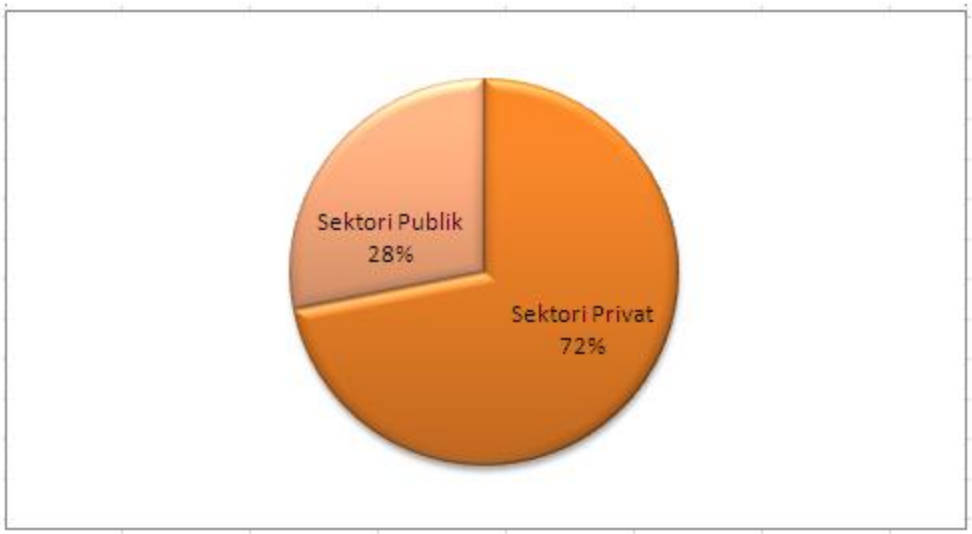

Figura 3.1. Skema në përqindje sipas sektoreve.

Numri i punonjësve në kompanitë dhe organizatat varionte nga 4 punonjës deri në 100 punonjës, figura 3.2.

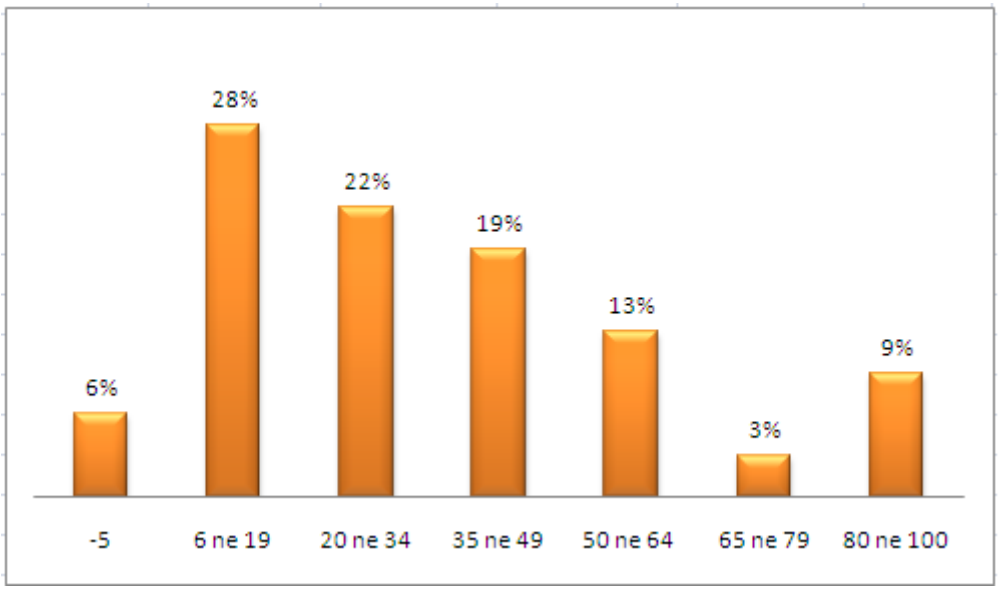

Figura 3.2. Numri i punonjësve në përqindje

Një pjesë e madhe e kompanive dhe organizatave rreth 81% e tyre, kanë një departament të Teknologjisë së Informacionit dhe 19% prej tyre, nuk kanë një departament të IT. Nga sa përmendëm më lart të gjitha kompanitë ose organizatat kanë një websajt dhe nga pyetësori rezulton që 67%, e kanë zhvilluar vetë websajtin e tyre dhe 33% kanë kërkuar ndihmë nga jashtë, figura 3.3.

Figura 3.3. Zhvillimi i websajtit

Diferenca midis kompanive që e kishin krijuar vetë websajtin e tyre dhe atyre që kishin kërkuar ndihmë nga jashtë qëndronte në dijeninë që ato kishin për sulmet me injektim SQL. Në rastin kur e kishin krijuar vetë websajtin e tyre, 13% e tyre, nuk ishin në dijeni të sulmeve me injektim SQL dhe në rastin kur ishte e krijuar nga një kompani tjetër vlera ishte 21%.

Pyetjes se kur e keni krijuar websajtin tuaj, nga sa tregohet edhe në figuren 3.4, 19% e kompanive iu pergjigj se kanë më pak se 1 vit që e kanë krijuar, 66% prej tyre u përgjigjën nga 1 ne 4 vjet dhe 13% para katër vitesh, Një numër i vogël kompanish nuk kishin dijeni se kur e kishin krijuar websajtin e tyre.

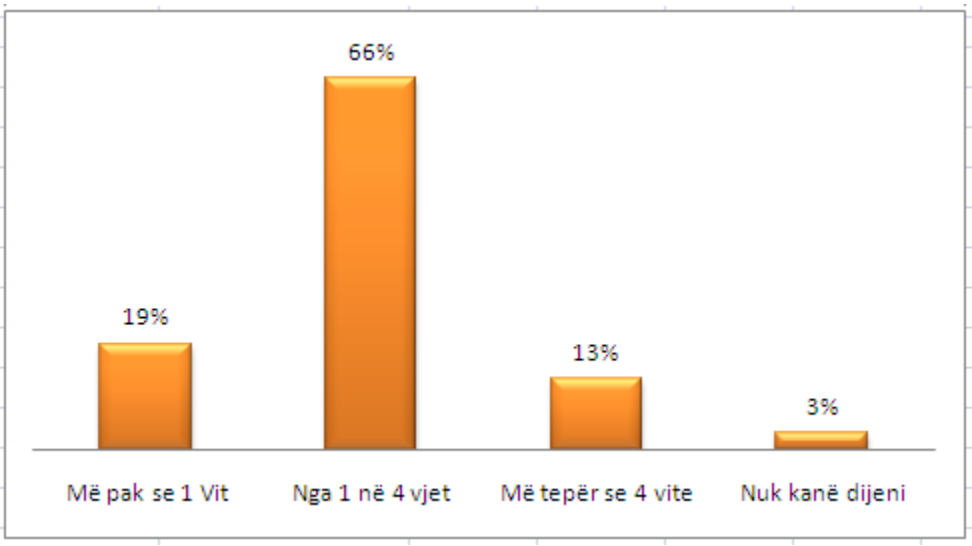

Figura 3.4. Koha e krijimi të webit

Nga sa e përmendëm në fillim te studimit 66% e kompanive apo organizatave kishin dijeni për sulmet me injektim SQL dhe 34% prej tyre nuk kishin dijeni për to figura 3.5.

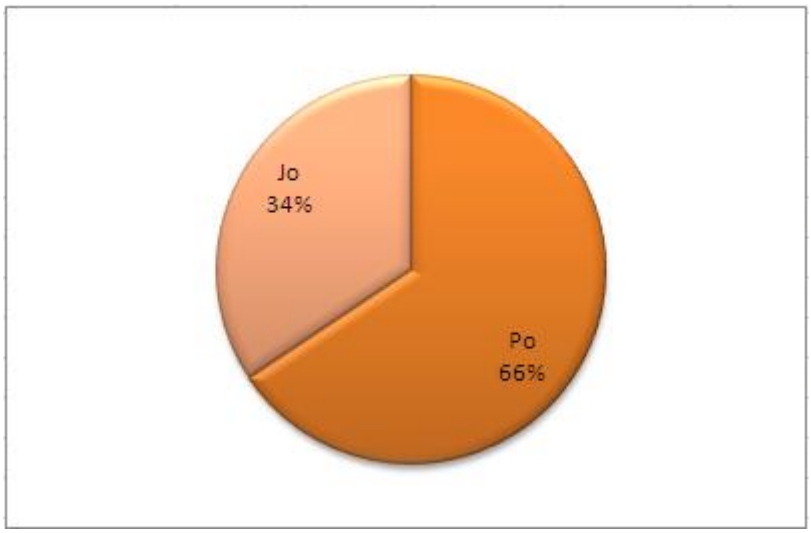

Figura 3.5. Dijeni për sulmet me injektim SQL

Kur kompanitë apo organizatat u pyetën për masat që ato merrnin për të parandaluar sulmet me injektim SQL, figura 3.6, 59% prej tyre u përgjigjën që e kontrollonin inputin ose të dhënat në hyrje, në server. 47% prej tyre u përgjigjën që e kontrollonin inputin në web, dhe 13% prej tyre, nuk mernin asnjë masë për t'i parandaluar sulmet me injektim SQL. 16% u përgjigjen se e kanë lënë parandalimin e sulmeve me injektim SQL, në dorë të një kompanie tjetër në fushën e Teknologjisë së Informacionit.

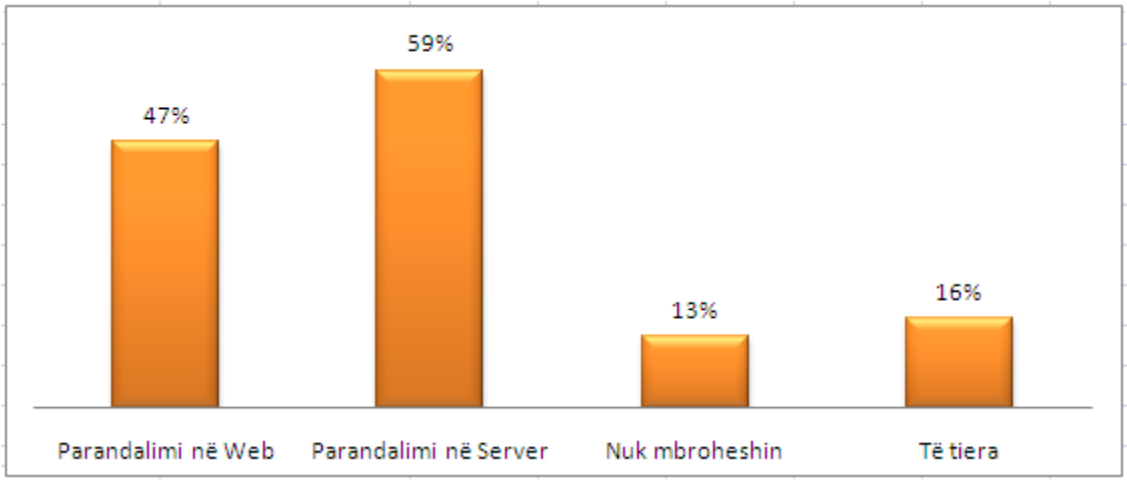

Figura 3.6. Menyra e parandalimit te sulmeve me injetkim SQL

Gjithashtu kompanitë dhe organizatat u pyetën se si ato i mbronin të dhënat personale që ndodheshin në bazën e tyre të të dhënave, figura 3.7. Nga përgjigjet e dhëna, ku këto kompani

apo organizata, mund të zgjidhnin disa variante, rezultoi që pothuajse rreth 69% e tyre përdornin enkriptimin dhe 9% prej tyre nuk bënin asgje. 31% deklaroi në pyetësor që përdornin një mënyrë tjetër.

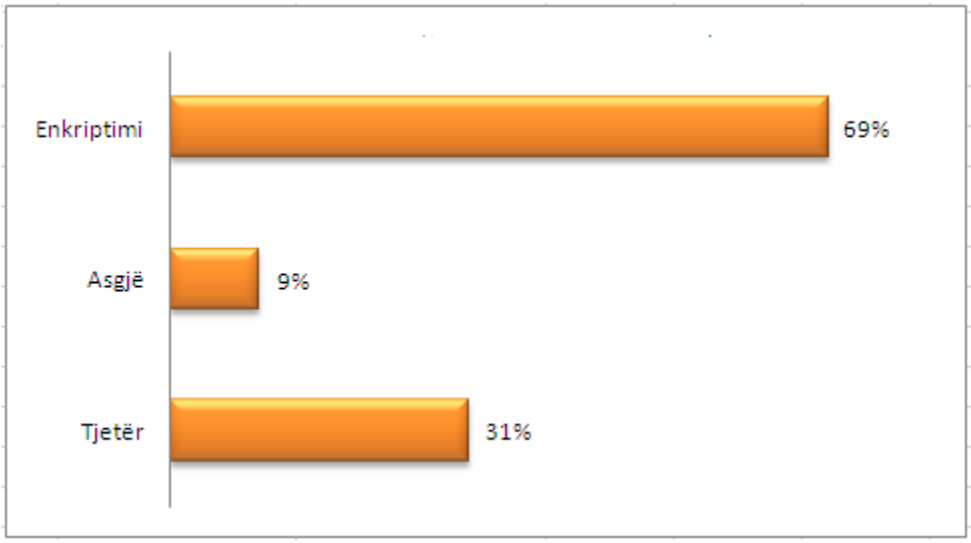

Figura 3.7. Mbrojtja e të dhënave personale

Nga të gjitha kompanitë dhe organizatat që iu pergjigjën pyetjes nëse dikush kishte tentuar të sulmonte webin e tyre duke përdorur sulmet me injektim SQL, figura 3.8, 53% e kishin kuptuar sulmin me injektim SQL mbi webin e tyre, 25% nuk e kishin vënë re sulmin me injektim SQL, dhe 22% nuk kishin dijeni nëse dicka e tillë ka ndodhur apo jo.

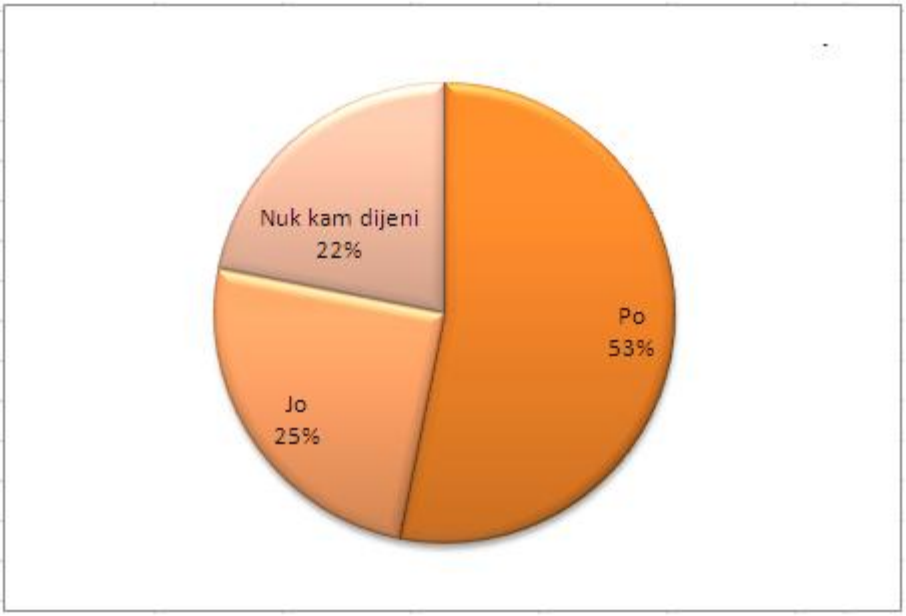

Figura 3.8. Përpjekja e sulmeve me injektim SQL ndaj kompanive

Figura 3.9. tregon identifikimin e sulmeve me injektim SQL, nga kompanitë apo organizatat, pjesëmarrëse në pyetësor të ekspozuara ndaj sulmeve me injektim SQL.

Rreth 72% prej tyre janë përgjigjur se mënyra më e zakonshme për të identifikuar një sulm është që në sistemet e logimit, më pas rreth 31% nëpërmjet sistemeve të identifikimit, dhe rreth 13% e kishin identifikuar sepse websajti i tyre kishte ndryshuar. Dhe 38% prej tyre deklaruan se kompania e identifikon sulmin në një mënyrë tjetër.

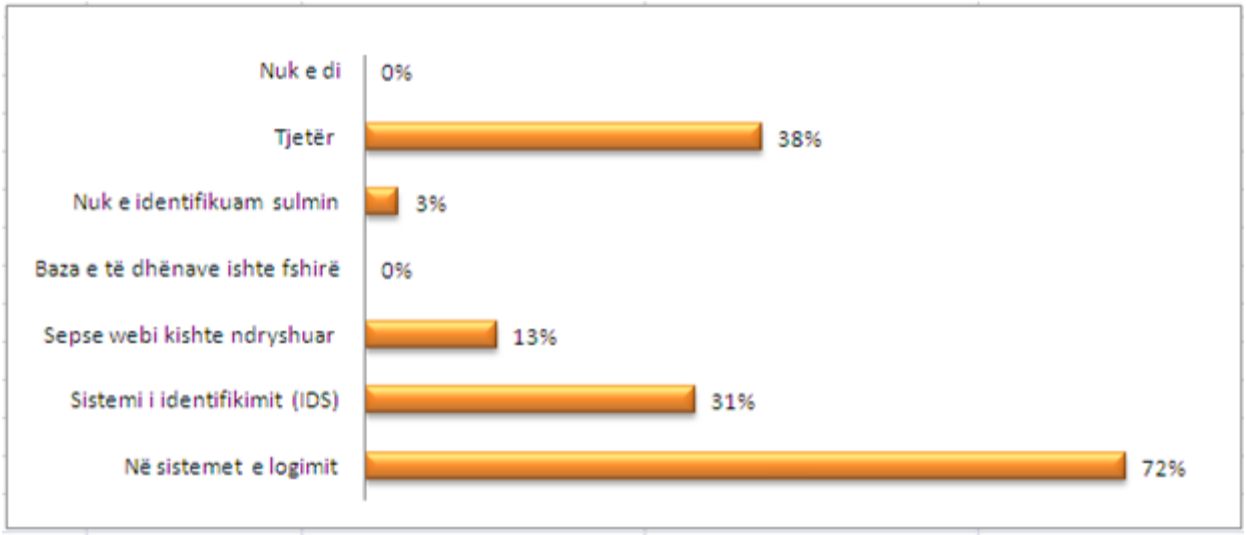

Figura. 3.9. Identifikimi i sulmeve

3.3 Analizimi i Pyetësorit mbi Sulmet me Injektim SQL në Shqipëri dhe Konkluzionet për Kapitullin e Tretë

Një numër prej 32 kompanish dhe organizatash i janë përgjigjur këtij studimi, pra shkalla në përqindje e përgjigjeve është 89%. Qëllimi ishte që pyetësorit tonë ti përgjigjeshin më tepër se sa 30 kompani, me qëllim që të bënim një analizë statistikore të pranueshme, për të marrë një shkallë përgjigjeje mbi 30%, gjë e cila u arrit.

Kompanitë dhe organizatat që morën pjesë në këtë studim vijnë nga sektori privat dhe ai publik, figura 3.1. dhe të gjitha këto kompani kishin nga një websajt funksional. Këto kompani ose organizata ishin të madhësive të ndryshme duke u nisur nga numri i punonjësve, i cili varionte, nga 10 deri në 100 punonjës, figura 3.2.

Duke u nisur nga rezultatet e kërkimit vihet re se pjesa më e madhe e kompanive dhe organizatave janë të vetëdijshëm për sulmet me injektim SQL, por fakti interesant është që pothuajse gjysma e të gjitha kompanive dhe organizatave, rreth 34% prej tyre, nuk janë të vetëdijshme për sulmet me injektim SQL, figura 3.5. Kjo tregon që shumë prej tyre, nuk po bëjnë mjaftueshëm për të parandaluar dhe mbrojtur websajtet e tyre nga sulmet me injektim SQL.

Pjesa më e madhe e kompanive dhe organizatave i kane identifikuar sulmet në sistemet e logimit dhe shume prej tyre gjithashtu i kanë zbuluar sulmet me injektim SQL, duke përdorur një sistem identifikues IDS dhe më pas vijnë mënyra të tjera parandalimi, figura 3.9.

Pas sulmit me injektim SQL, pjesa më e madhe e kompanive dhe e organizatave kanë bërë ndryshime në websajtet dhe bazat e të dhënave të tyre, ndërsa kishte dhe kompani ose organizata që nuk bënë ndonjë ndryshim të të dhënave aktuale, sepse nuk ishte nevoja.

Ndryshimet që kompanite ose organizatat bënë pas sulmit me injektim SQL, ishin kontrolli i të dhënave në hyrje, në shtresën e serverit dhe në websajtin e tyre, dhe ndryshime të tjera që kishin të bënin me permiresimin e sistemit.

Fakt interesant është që rreth 25% e kompanive ose organizatave të sulmuara, nuk e kishin vënë re sulmin me injektim SQL dhe 22% prej tyre, nuk kishin dijeni nëse dicka e tillë ka ndodhur apo jo, figura 3.8.

Më shumë se gjysma e kompanive dhe organizatave i kanë zhvilluar vetë websajtet e tyre, ndërsa pjesa tjetër ka përdorur mënyra të tjera zgjidhjesh jashtë kompanive ose organizatave**.** Në bazë të pergjigjeve të pyetësorit nga këto kompani dhe organizata u studiua lidhja midis rrezikut nga sulmet me injektim SQL dhe mënyrës së zhvillimit të websajtit, dhe rezultoi se kompanitë dhe organizatat që i kishin besuar websajtin e tyre në një kompani tjetër ishin shume më pak të rezikuar nga sulmet me injektim SQL, se sa kompanitë që e menaxhonin vetë websajtin e tyre.

Kompanitë dhe organizatat me pak punonjës nuk e dinin nëse ato kanë qënë të ekspozuara ndaj sulmeve me injektim SQL apo jo. Kompanitë dhe organizatat më të mëdha theksojnë që ato nuk kanë qënë të ekspozuara ndaj sulmeve me injektim SQL. Kjo tregon faktin se kompanitë dhe organizatat e vogla, të cilat nuk kanë një departament IT mbikqyrës, nuk janë të sigurtë, nëse janë ose jo objekt i sulmeve me injektim SQL.

Nga sa përmendëm dhe më parë, është bërë një krahasim, kur websajtet e kompanive dhe organizatave janë zhvilluar dhe nëse ato kanë qënë të ekspozuara ndaj sulmeve me injektim SQL, figura.3.4. Në këtë krahasim, bëhet e qartë që kompanitë dhe organizatat që i kanë pasur websajtet e tyre të zhvilluara më afër në kohë janë më të vetëdijshme ndaj përpjekjeve të sulmeve me injektim SQL. Ishte e qartë që kompanitë dhe organizatat që i kanë zhvilluar websajtet e tyre kohët e fundit, ishin në gjëndje të kuptonin nëse ato ishin sulmuar apo jo.

4. Teknikat që Identifikojnë dhe Parandalojnë Sulmet me Injektim SQL

Në këtë kapitull do të trajtohet një studim [41], [45] i kryer nga ne, mbi teknikat e identifikimit dhe parandalimit të sulmeve me injektim SQL, të metat dhe veçoritë e këtyre teknikave dhe gjithashtu do të realizohet një studim mbi teknikat identifikuese dhe parandaluese, në raport me sulmet me injektim SQL, të trajtuara në kapitullin e dytë.

Për ta kryer këtë studim, ne fillim ne do të përcaktojme llojet e ndryshme të sulmeve me injektim SQL [41]. Gjithashtu, do të investigojme teknikat e identifikimit dhe parandalimit të sulmeve me injektim SQL. Me pas, do të krahasojmë këto teknika, duke u bazuar në kriteret e zhvillimit dhe vlerësimit të tyre.

4.1 Teknikat që Identifikojnë Sulmet me Injektim SQL

Më poshtë janë analizuar teknikat që identifikojne sulmet me injektim SQL. Secila teknikë identifikuese do të studiohet në raport me sulmet me injektim SQL dhe mbi kriteret e zhvillimit dhe vlerësimit te secilës teknikë.

4.1.1 SQLRand

Autorët në artikullin [50], kanë propozuar teknikën SQLRand, e cila përdor një bashkësi instruksionesh, në mënyre të rastësishme, nga instruksionet SQL, për të kontrolluar sulmet me injektim SQL, pra kanë përdorur konceptin e një manuali instruktiv mbi mangësitë e gjuhës SQL. Fjalët kyçe të SQL, ndryshohen duk ju bashkëngjitur atyre, në mënyrë të rastësishme, një numër të plotë, i cili bën që sulmuesi të ketë veshtirësi në parashikim.

Këto fjalë kyçe nuk njihen nga sulmuesi, kështu që, kodi i injektuar nga sulmuesi trajtohet si fjalë kyçe e papërcaktuar dhe kërkesa nuk i dërgohet bazës së të dhënave. Prandaj sulmuesi që tenton sulmin me injeksion SQL, do të evitohet.

Hartimi i projektit të SQLRand-it, përbëhet nga një proxy server i cili është vendosur midis klientit dhe serverit të bazës së të dhënave. Detyra kryesore e serverit është që të sistemojë kërkesat e dyshimta ose të papërcaktuara SQL, të cilat kanë ardhur në mënyrë të rastësishme, dhe më pas ti transmetojë këto istruksione SQL, drejt serverit të bazës së të dhënave për ekzekutim. Nëse ndonjë përdorues tenton sulmin me injektim SQL, mekanizmi kontrollues do ta kundërshtojë atë.

Për ta vënë në jetë këtë teknikë, zhvilluesi ka nevojë të modifikojë libraritë e klientit. Autorët kanë publikuar një raport vlerësimi konkret të rritjes së performancës dhe kanë provuar që kjo gjë nuk e dëmton performancën.

Disavantazhi i këtij sistemi është konfiguracioni kompleks dhe siguria e fjaleve kyçe. Nëse fjalët kyçe ekspozohen, sulmuesi mund të formulojë kërkesa për një sulm të suksesshëm.

Mbi bazën e studimeve nga autorë të ndryshëm që kanë testuar dhe publikuar artikuj për teknikën SQLRand, kemi dalë në përfundimin, se teknika SQLRand i identifikon të gjitha sulmet

me injektim SQL, përveç sulmeve me injektim SQL që ndodhin në procedurat e ruajtura (Tabela 4.1). Kjo teknikë paraqet një gabim pozitiv dhe negativ të nënkuptuar.

Tabela 4.1. Teknika SQLRand në raport me sulmet me injektim SQL

Sulmet me injektim SQL	SQLRand
Tautologji	E identifikon
Piggy-backed	E identifikon
Kueri llogjikisht gabim	E identifikon
Union kueri	E identifikon
Procedurat e ruajtura	Nuk e identifikon
Inference	E identifikon
Kodimi Alternativ	E identifikon

Gjithashtu mbi bazën këtyre studimeve në tabelën 4.2 janë paraqitur disa karakteristika të kësaj teknike.

Tabela 4.2. Disa karakteristika të teknikës SQLRand

Teknika	Koha e identifikimit	Vëndi i identifikimit	Modifikimi kodit	Infrastrukturë shtesë
SQLRand [50]	Në kohën e ekzekutimit	Proksi server	Po	Nuk kërkohet

4.1.2 SQLGuard: Përdorimi i Përqasjes së Vlerësimit të Trungut të të Dhënave Gjuhësore për të Parandaluar Sulmet me Injeksion SQL

Autorët në artikullin [6], kanë propozuar SQLGuard, e cila mbron aplikacionet web në internet, duke krahasuar strukturën e skemës gramatikore të një kërkese SQL, përpara futjes së të dhënave nga perdoruesi dhe të kërkesës që i drejtohet bazes së të dhënave, pas futjes së të dhënave nga përdoruesi, pra duke i krahasuar këto dy kërkesa me njëra-tjetrën.

Kjo teknikë e kontrollon kërkesën që i bëhet bazës së të dhënave në kohën e ekzekutimit duke e vëzhguar atë, nëse modeli i kërkesës është i sulmuar ose jo. Ky model paraqitet si një formë gramatikore e cila pranon vetëm kërkesat e ligjshme dhe gjenerohet në kohën e punës, duke kontrolluar strukturën e kërkesave, përpara dhe pas futjes së të dhënave nga përdoruesit. Një çelës sekret është përdorur nga një mekanizëm kontrolli për gjatë gjithë kohës së punës, për të kufizuar futjen e të dhënave në hyrje nga përdorues të ndryshëm. Në këtë mënyrë, siguria e kësaj teknike është plotësisht e varur nga siguria e këtij çelësi sekret.

Autorët e kanë zbatuar atë në platformën J2EE. Kur zhvilluesi të ketë qëllimin e përdorimit të kësaj teknike, ai duhet ose të vendosë indikatorë të veçantë në formë manuale në kodin burim ku

përdoruesi ka vendosur të dhënat mbi bazën e një kërkese të krijuar në formë dinamike ose ta rishkruaj kodin nga e para. Kështu që një perpjekje shtesë në modifikimin e kodit është e nevojshme për ta vënë atë në punë.

Mbi bazën e studimeve nga autorë të ndryshëm që kanë testuar dhe publikuar artikuj për teknikën SQLGuard, kemi dalë në përfundimin, se teknika SQLGuard i identifikon të gjitha sulmet me injektim SQL, përveç sulmeve me injektim SQL që ndodhin në procedurat e ruajtura (Tabela 4.3). Kjo teknikë paraqet një gabim pozitiv dhe negativ të nënkuptuar.

Tabela 4.3. Teknika e SQLGuard në raport me sulmet me injektim SQL

Sulmet me injektim SQL	SQLGuard
Tautologji	E identifikon
Piggy-backed kueri	E identifikon
Kueri llogjikisht gabim	E identifikon
Union kueri	E identifikon
Procedurat e ruajtura	Nuk e identifikon
Inference	E identifikon
Kodimi alternativ	E identifikon

Gjithashtu mbi bazën këtyre studimeve në tabelën 4.4 janë paraqitur disa karakteristika të kësaj teknike.

Tabela 4.4. Disa karakteristika të teknikës SQLGuard

Teknika	Koha e identifikimit	Vëndi i identifikimit	Modifikimi kodit	Infrastrukturë shtesë
SQLGuard [6]	Në kohën e ekzekutimit	Në serverin e aplikacionit	Po	Kërkohet

4.1.3 Analiza Dinamike dhe Amnesia: Analiza e Kombinuar Statike dhe Dinamike, dhe Monitorimi per Neutralizimin e Sulmeve me Injektim SQL

Analiza dinamike ndryshe nga ajo statike, mund të lokalizojë dobësitë e sulmeve me injektim SQL pa bërë ndonjë rregullim apo përshtatje me web aplikacionet. Ka disa programe ose modele të cilat kanë kodin e hapur, që skanojnë jo vetëm dobësitë e sulmeve me injektim SQL, por edhe dobësitë e tjera që web aplikacionet kanë.

Modele të tilla, nuk janë perfekte sepse përdorin kode sulmi të paracaktuara për të skanuar dhe perdorur përgjigjet HTTP, për të rritur shkallën e suksesshmërisë së sulmit. Sania [28], kërkon dhe mbledh dobësitë e sulmeve me injektim SQL, midis web aplikacioneve dhe bazës së të dhënave. Më pas ajo vazhdon të gjenerojë kodet e sulmeve me injektim SQL. Mbas sulmit me

kodin e gjeneruar, ajo mbledh kërkesat SQL nga ky sulm. Pas krahasimit të një kërkese normale SQL dhe analizimit me kërkesën SQL të ruajtur, duke përdorur mekanizimin e kontrollit nëpërmjet pemës gramatikore, verifikohet shkalla e suksesit të sulmit me injektim SQL. Për sa kohë teknika kundër sulmeve me injektim SQL e quajtur Sania, përdor këtë mekanizëm kontrolli dhe krahasimi, është më i saktë se sa përdorimi i verifikimit të përgjigjes HTTP.

Autorët në artikullin [29], kanë propozuar të përdorej analiza e rrjedhjes dhe vlerësimit të të dhënave në hyrje, për të gjeneruar teste të të dhënave në hyrje, të cilat do të përcaktonin dobësitë e sulmeve me injektim SQL. Metoda e analizës dinamike ka përparësi sepse nuk kërkon ndonjë përshtatje të veçantë, të web aplokacioneve, sidoqoftë dobësitë që gjënden në web aplikacione, mund të korrigjohen dhe në mënyrë manuale nga zhvilluesit e këtyre web aplikacioneve dhe jo të gjitha prej tyre mund të gjënden nga sulmet me injektim SQL të paracaktuara.

Autorët në artikullin [7], kanë prezantuar një model për identifikimin dhe parandalimin e sulmeve me injektim SQL, në aplikacionet që janë ndërtuar me Java. Autorët e kanë quajtur këtë teknikë të tyre AMNESIA dhe e kanë testuar dhe zhvilluar këtë teknikë në aplikacione të ndryshme në kohë reale.

Amnesia është një teknikë, modeli i së cilës kombinon analizën statike dhe monitorimin në kohën e punës [7]. Në fazën e saj statike, Amnesia duke analizuar kodin e aplikacionit, përdor analizën statike për të ndërtuar modele te kërkesave të llojeve të ndryshme, që një aplikacion në mënyrë të ligjshme gjeneron, në secilën pikë të aksesimit të bazës së të dhënave. Në fazën e saj dinamike, Amnesia kontrollon të gjitha kërkesat e gjeneruara në mënyrë dinamike në kohën e punës dhe i krahason ato me kërkesat legjitime të ndërtuara nga modeli statik, pra kap të gjitha kërkesat përpara se ato të dërgohen në bazën e të dhënave dhe kontrollon secilën kërkesë përkundrejt modeleve të ndërtuara nga ana statistike. Kërkesat që dëmtojnë modelin janë përcaktuar si sulme me injektim SQL dhe ndalohen të ekzekutohen në bazen e të dhënave.

Kjo teknikë konsiston në tre hapa:

- ✓ Të identifikojë momentet kritike:

  Të skanojë kodin, të gjejë pikat kritike të cilat prodhojnë më pas sulmet me injektim SQL.

- ✓ Ndërtimi i modelit të një kërkese SQL:

  Për çdo pikë kritike të identifikuar, ndërtohet një model kërkese SQL, i cili paraqet të gjitha kërkesat SQL të mundshme, që do të ndërtohen në këtë pikë kritike.

- ✓ Monitorimi në kohën e ekzekutimit:

Kontrollon kërkesën e gjeneruar në mënyrë dinamike përkundrejt modelit statik të një kërkese SQL, dhe bllokon kërkesat që dëmtojnë modelin.

Për të vënë në jetë këtë teknikë, për zhvilluesin nuk është e nevojshme të modifikojë kodin e aplikacionit. Autorët kanë prezantuar gjithashtu rezultate empirike të saktesisë së modelit të tyre dhe një performancë të lartë të këtij modeli. Në vlerësimin dhe punën e tyre, autorët kanë treguar se kjo teknikë funksionon mirë kundër sulmeve me injektim SQL.

Një ndër mangësite kryesore të kësaj teknike duke u bazuar në artikullin [8], është se suksesi i saj varet nga saktësia e analizës së saj statike në ndërtimin e modeleve të kërkesave që i drejtohen një baze të dhënash.

Mbi bazën e studimeve nga autorë të ndryshëm që kanë testuar dhe publikuar artikuj për teknikën Amnesia, kemi dalë në përfundimin, se teknika Amnesia i identifikon të gjitha sulmet me injektim SQL, me përjashtim të sulmeve me injektim SQL që ndodhin në procedurat e ruajtura (Tabela 4.5). Kjo teknikë paraqet një gabim pozitiv dhe negativ konkret.

Tabela 4.5. Amnesia në raport me sulmet me injektim SQL

Sulmet me injektim SQL	Amnesia
Tautologji	E identifikon
Piggy-backed kueri	E identifikon
Kueri llogjikisht gabim	E identifikon
Union kueri	E identifikon
Procedurat e ruajtura	Nuk e identifikon
Inference	E identifikon
Kodimi alternativ	E identifikon

Gjithashtu mbi bazën këtyre studimeve në tabelën 4.6 janë paraqitur disa karakteristika të kësaj teknike.

Tabela 4.6. Disa karakteristika të teknikës Amnesia

Teknika	Koha e identifikimit	Vëndi i identifikimit	Modifikimi kodit	Infrastrukturë shtesë
Amnesia [7]	Në kohën e ekzekutimit	Në serverin e aplikacionit	Jo	Nuk kërkohet

4.1.4 SQLCheck

Autorët në artikullin [9], e kanë quajtur këtë teknikë SQLCheck, ata kanë zbatuar algoritmin e tyre me SQLCheck në një mejdis real. Një mekanizëm mbrojtës është përdorur për kufizimin e të dhënave në hyrje nga përdorues të ndryshëm. Në artikullin e tyre ata kanë siguruar përkufizimin e parë formal të sulmeve me injektim SQL dhe të parandalimit të tyre. Kjo teknikë parandaluese është e bazuar në gramatikën me kontekst të lirë dhe krahasimin e skemës shpjeguese gramatikore më të dhënat në hyrje të përdoruesit në kohën e ekzekutimit.

Autorët e kanë testuar dhe vlerësuar teknikën e tyre në aplikacione reale të shkruara në PHP dhe JSP. Për të parë funksionimin e kësaj teknike, një numër i madh sulmesh me injektim SQL, janë përdorur si të dhëna në hyrje.

SQLCheck është ndërtuar me idenë e bllokimit të kërkesave, ku të dhënat në hyrje të përdoruesve mund të ndryshojnë strukturën sintaktike të kërkesës drejtuar bazës së të dhënave. Për të gjurmuar të dhënat hyrëse nga përdoruesit, autorët kanë përdorur të dhëna të ndërmjetme, për të shënuar fillimin dhe mbarimin e çdo vargu karakteresh në hyrje. SQLCheck është zhvilluar duke përdorur të dhënat gramatikore në dalje dhe një vijueshmëri që përcakton format e vlefshme gjuhësore të secilës kërkesë.

SQLCheck qëndron në web server. Më pas web aplikacioni gjeneron kërkesa në rritje në bazë të detyrave që kryen, të cilat kontrollohen nga SQLCheck. Nëse kërkesa është kontrolluar në mënyrë të suksesshme, atëherë ajo është një kërkesë legjitime dhe SQLCheck e dërgon atë në bazën e të dhënave për ekzekutim, në të kundërt ajo do ta bllokojë atë.

Për ta vënë në zbatim këtë teknikë, zhvilluesve u duhet të modifikojnë kodin burim duke përdorur librari speciale ose në mënyrë manuale duke bërë ndryshime brënda kodit, ku të dhënave në hyrje të përdoruesve u shtohet një kërkesë e gjeneruar në mënyrë dinamike.

Autorët kanë publikuar një raport të saktësisë dhe performancës. Analiza e tyre nuk paraqet gabimet pozitive dhe ato negative, gjithashtu ritmi i kohës së punës është i ulët dhe mund të zbatohet drejtpërsëdrejti në shumë aplikacione duke përdorur gjuhë të ndryshme.

Mbi bazën e studimeve nga autorë të ndryshëm që kanë testuar dhe publikuar artikuj për teknikën SQLCheck, kemi dalë në përfundimin, se teknika SQLCheck i identifikon të gjitha sulmet me injektim SQL, përveç sulmeve me injektim SQL që ndodhin në procedurat e ruajtura (Tabela 4.7). Kjo teknikë paraqet një gabim pozitiv dhe negativ te nënkuptuar.

Tabela 4.7. Teknika e SQLCheck në raport me sulmet me injektim SQL

Sulmet me injektim SQL	SQLCheck
Tautologji	E identifikon
Piggy-backed	E identifikon

Ilegal incorrect	E identifikon
Union	E identifikon
Procedurat e ruajtura	Nuk e identifikon
Inference	E identifikon
Alternate Encondings	E identifikon

Gjithashtu mbi bazën këtyre studimeve në tabelën 4.8 janë paraqitur disa karakteristika të kësaj teknike.

Tabela 4.8. Disa karakteristika të teknikës SQLCheck

Teknika	Koha e identifikimit	Vëndi i identifikimit	Modifikimi kodit	Infrastrukturë shtesë
SQLCheck [9]	Në kohën e ekzekutimit	Proksi server	Po	Kërkohet

4.1.5 Swaddler: Një Model për Identifikimin mbi Bazën e Anomalive të Pasaktësive në Web Aplikacionet

Swaddler [10], analizon gjëndjen e brëndshme të një web aplikacioni. Kjo teknikë punon duke u bazuar në variablat teke dhe të shumëfishta, dhe paraqet një rrugë shumë të qartë kundër sulmeve në përgjithësi, që i ndodhin web aplikacioneve të ndryshme.

Kjo teknikë ndalon sulmet me injektim SQL në shtresën e bazës së të dhënave, ndërkohë mesa kemi parë deri tani nga kërkimi shkencor i kryer, pjesa më e madhe e teknikave i lufton këto sulme në shtresën e aplikacionit. Kjo teknikë mbron pjesërisht të gjitha sulmet me injektim SQL, dhe gjithashtu sulmet ndaj procedurave te ruajtura, që është një nga llojet e sulmeve me injektim SQL më pak të zbuluara deri sot nga teknikat identifikuese dhe parandaluese.

Kjo teknikë funksionon me anë të kombinimit të analizimit të kodit statik dhe kohës së ekzekutimit të kërkesave. Në pjesën statike, një mekanizem kontrolli është dizenjuar pikërisht në procedurat e ruajtura dhe për cdo kërkesë SQL, e cila është mbeshtetur në të dhënat hyrëse të përdoruesve, bëhet kontrolli nga ky mekanizëm për të parandaluar kërkesat e pasakta drejt bazës së të dhënave, pra bëhet një krahasim me strukturën e kërkesës origjinale SQL, me atë të të dhënave në hyrje të përdoruesve.

Kjo teknikë mund të përdoret në mënyrë të pavarur, automatikisht, dhe vetëm atëherë kur është e nevojshme.

Mbi bazën e studimeve nga autorë të ndryshëm që kanë testuar dhe publikuar artikuj për teknikën Swaddler, kemi dalë në përfundimin, se teknika Swaddler i identifikon pjesërisht të gjitha sulmet me injektim SQL (Tabela 4.9). Mbi bazën e këtyre studimeve rezulton se gabimi pozitiv në këtë teknikë është konkret dhe gabimi negativ është i nënkuptuar.

Tabela 4.9. Teknika e Swaddler së raport me sulmet me injektim SQL

Sulmet me injektim SQL	Swaddler
Tautologji	Pjesërisht
Piggy-backed kueri	Pjesërisht
Kueri llogjikisht gabim	Pjesërisht
Union kueri	Pjesërisht
Procedurat e ruajtura	Pjesërisht
Inference	Pjesërisht
Kodimi alternativ	Pjesërisht

Gjithashtu mbi bazën këtyre studimeve në tabelën 4.8 janë paraqitur disa karakteristika të kësaj teknike.

Tabela 4.10. Disa karakteristika të teknikës SQLCheck

Teknika	Koha e identifikimit	Vëndi i identifikimit	Modifikimi kodit	Infrastrukturë shtesë
Swaddler [10]	Në kohën e ekzekutimit	Në serverin e aplikacionit	Jo	Kërkohet

4.1.6 Sania: Analiza Sintaktike dhe Semantike për Testimet e Autorizuara Kundër Sulmeve me Injektim SQL

Autorët në artikullin [28], kanë propozuar një teknikë që identifikon dobësitë e shkaktuara nga sulmet me injektim SQL, në web aplikacionet, gjatë fazës së zhvillimit dhe testimit të tyre. Kjo teknikë intercepton kërkesat SQL që kalojnë nga aplikacioni në bazën e të dhënave dhe sulmet me injektim SQL gjenerohen në mënyrë të pavarur sipas semantikës dhe sintaksës të pozicionit të kerkesave SQL.

Për të përcaktuar pozicionet e papërshtatshme, kjo teknikë analizon kërkesat e dala perkundrejt kërkesave HTTP dhe përcakton pozicionet e papërshtatshme në kërkesat SQL, në të cilat sulmuesi mund të ndërhyje me karaktere të dyshimta. Autorët e kanë zbatuar këtë teknikë në një tuls të quajtur Sania.

Mbi bazën e studimeve nga autorë të ndryshëm që kanë testuar dhe publikuar artikuj për teknikën Sania, kemi dalë në përfundimin, se teknika Sania i identifikon pjesërisht të gjitha sulmet me injektim SQL (Tabela 4.11). Mbi bazën e këtyre studimeve rezulton se gabimi pozitiv në këtë teknikë është konkret dhe gabimi negativ është i nënkuptuar.

Tabela 4.11. Teknika Sania në raport me sulmet me injektim SQL

Sulmet me injetkim SQL	Sania[28]
Tautologji	Pjesërisht
Piggy-backed kueri	Pjesërisht
Kueri llogjikisht gabim	Pjesërisht
Union kueri	Pjesërisht
Procedurat e ruajtura	Pjesërisht
Inference	Pjesërisht
Kodimi alternativ	Pjesërisht

Gjithashtu mbi bazën këtyre studimeve në tabelën 4.12 janë paraqitur disa karakteristika të kësaj teknike.

Tabela 4.12. Disa karakteristika të teknikës Sania

Teknika	Koha e identifikimit	Vëndi i identifikimit	Modifikimi kodit	Infrastrukturë shtesë
Sania[28]	Gjatë testimit	Klient dhe Server	Jo	Nuk kërkohet

4.1.7 SAFELI: Skema e Analizës Statike për Zbulimin ose Identifikimin e Sulmeve me Injektim SQL

Autorët në artikullin [11], kanë propozuar një strukturë pune për analizën statike me qëllim që të identifikojë dobësite e sulmeve me injektim SQL, në web aplikacionet. Safeli, që e përshtatur ne gjuhën shqipe do të thotë "Skema e analizës statike për identifikimin e dobësive të sulmeve me injektim SQL".

Qëllimi i teknikës Safeli, është të përcaktojë ose identifikojë sulmet me injektim SQL, gjatë kohës së kompilimit. Skema e përdorur merret me kodin e web aplikacioneve, të ndërtuar në java dhe përfshin një zbatim simbolik, për të kontrolluar nga ana statike dobësitë në sistemin e sigurise. Në çdo pikë kritike në të cilën kalon një kërkesë SQL, gjenerehohet një varg karakteresh, për të zbuluar vlerat origjinale të këtyre kërkesave, të cilat mund të çojnë deri në demtimin përfundimtar të bazës së të dhënave.

Kjo mënyrë e përdorimit të analizës statike ka dy përparësi kryesore: Së pari ai bën një analizë statike të gjenerimit të testeve dhe së dyti ai përdor një varg karakteresh hibride. Për analizën statike të gjenerimit të testeve, modeli i propozuar nga autorët vlerëson kodin ose merret kryesisht me vargjet e karaktereve. Në përparësinë e dytë, kjo metodë përdor një mjet efiçent, të analizës së një vargu karakteresh, i cili është i aftë të merret me numrat e plotë dhe me një grup karakteresh. Zgjidhja e siguruar nga një varg karakteresh hibride, përdoret për të gjeneruar raste testimi. Nëse skema ndeshet me fjale kyçe të keqpërdorura SQL, shkakton përjashtimin e tyre dhe e ndalon proçesin e kërkesës drejt bazës së të dhënave.

Skema SAFELI ka katër komponentë kryesore:

- ✓ Aparatura e kryerjes së simbolikave JAVA
- ✓ Libraritë e modeleve të sulmeve
- ✓ Vargu i karaktereve hibride
- ✓ Përsëritësi i rasteve të testimeve.

Autorët në artikullin e tyre nuk kanë përmëndur rezultatet e vlerësimit. Kjo teknikë paraqet një gabim pozitiv konkret dhe një gabim negativ të nënkuptuar.

Mbi bazën e studimeve nga autorë të ndryshëm që kanë testuar dhe publikuar artikuj për teknikën Safeli, kemi dalë në përfundimin, se teknika Safeli i identifikon të gjitha sulmet me injektim SQL, përveç sulmit të parë me injektim SQL tautologji (Tabela 4.13). Mbi bazën e këtyre studimeve rezulton se gabimi pozitiv dhe gabimi negativ në këtë teknikë janë konkret.

Tabela 4.13. Teknika Safeli në raport me sulmet me injektim SQL

Sulmet me injektim SQL	SAFELI
Tautologji	Nuk e identifikon
Piggy-backed kueri	E identifikon
Kueri llogjikisht gabim	E identifikon
Union kueri	E identifikon
Procedurat e ruajtura	E identifikon
Inference	E identifikon
Kodimi alternativ	E identifikon

Gjithashtu mbi bazën këtyre studimeve në tabelën 4.14 janë paraqitur disa karakteristika të kësaj teknike.

Tabela 4.14. Disa karakteristika të teknikës Safeli

Teknika	Koha e identifikimit	Vëndi i identifikimit	Modifikimi kodit	Infrastrukturë shtesë
SAFELI [11]	Në kohën e kompilimit	Në serverin e aplikacionit	Jo	Nuk kërkohet

4.1.8 SQL-IDS: Sistemet e Identifikimit të Ndërhyrjeve SQL. Një model Specifikisht i Bazuar në Zbulimin e Sulmeve me Injektim SQL

Autorët në artikullin [12], kanë përdorur një sistem për identifikimin e ndërhyrjeve nga jashtë IDS, për të identifikuar sulmet me injektim SQL. Ky sistem është i bazuar në makinat virtuale, të cilat janë të përgatitura për të identifikuar ose të mësuara për të njohur një grup kërkesash tipike që i drejtohen një aplikacioni. Kjo teknikë, ndërton disa modele të kërkesave tipike dhe më pas vëzhgon aplikacionin në kohën e ekzekutimit, për të identifikuar kërkesat që nuk i përshtaten modelit të krijuar.

Në vlerësimin për punën e tyre, autorët kanë treguar që sistemi i tyre është i aftë të zbulojë sulmet me injektim SQL, në një përqindje shumë të lartë. Megjithatë, një ndër mangësitë kryesore në teknikat që ato kanë përdorur për ta ndërtuar këtë model është që autorët nuk mund të japin garanci për cilësine e identifikimit ose zbulimit të sulmeve me injektim SQL, sepse suksesi i tyre, pra i zbulimit të sulmeve me injektim SQL, është i varur nga grupi i kërkesave tipike që ato kanë zgjedhur ose që mendojnë se do të sulmohet një aplikacion dhe si rrjedhim të dhënat e një baze të dhënash. Një bashkësi e varfër do të bënte që modeli i tyre të gjeneronte një numër të madh gabimesh pozitive dhe negative.

Në një artikull tjetër [13], autorët kanë ndjekur një metodologji specifike për të identifikuar dobësitë e sulmeve me injektim SQL. Në teknikën e tyre këta autorë kanë përdorur specifikime që karakterizojnë strukturën sintaktike të kërkesave, gjatë programimit, të cilat janë të gjeneruara dhe ekzekutohen nga aplikacioni.

Autorët gjithashtu nëpërmjet teknikës së tyre, vëzhgojnë aplikacionin gjatë ekzekutimeve të kërkesave SQL, për të anashkaluar ose mënjanuar ato kërkesa që çojnë në prishjen e specifikimeve të tyre.

Kjo teknikë kontrollon trafikun midis serverit të aplikacionit dhe serverit të bazës së të dhënave, ku çdo kërkesë SQL, kalon përmes një vlerësimi ose kontrolli, me qëllim për të identifikuar ose zbuluar ekzistencen potenciale të sulmeve me injektim SQL. Kjo teknikë është e pavarur nga çdo sistem tjetër, nga sistemet e menaxhimit të bazave të të dhënave, apo nga kushtet në të cilat ndodhet aplikacioni.

Autorët e kanë zbatuar modelin e tyre në SQL-IDS, që e përshtatur në gjuhën shqipe nënkupton sistemin e zbulimit të sulmeve me injektim SQL, i cili vëzhgon aplikacionet e ndërtuara me Java. Nuk është i nevojshëm modifikimi i kodit burim në aplikacionet ekzistuese për ta përdorur këtë teknikë mbrojtëse.

Mbi bazën e studimeve nga autorë të ndryshëm që kanë testuar dhe publikuar artikuj për teknikën SQL-IDS, kemi dalë në përfundimin, se teknika SQL-IDS, i identifikon të gjitha sulmet me injektim SQL (Tabela 4.15). Kjo teknike paraqet një gabim pozitiv dhe negativ konkret.

Tabela 4.15. Teknika SQL-IDS në raport me sulmet me injektim SQL

Sulmet me injektim SQL	SQL-IDS
Tautologji	E identifikon
Piggy-backed kueri	E identifikon
Kueri llogjikisht gabim	E identifikon
Union kueri	E identifikon
Procedurat e ruajtura	E identifikon
Inference	E identifikon
Kodimi alternativ	E identifikon

Gjithashtu mbi bazën këtyre studimeve në tabelën 4.16 janë paraqitur disa karakteristika të kësaj teknike.

Tabela 4.16. Disa karakteristika të teknikës SQL-IDS

Teknika	Koha e identifikimit	Vëndi i identifikimit	Modifikimi kodit	Infrastrukturë shtesë
SQL-IDS [12]	Në kohën e ekzekutimit	Në serverin e aplikacionit	Jo	Nuk kërkohet

4.1.9 CANDID: Vlerësimi Dinamik për Parandalimin në Mënyrë Automatike të Sulmeve me Injektim SQL

Autorët në artikullin [14], kanë propozuar Candid, një metodë e vlerësimit dinamik, për parandalimin në mënyrë automatike, të sulmeve me injektim SQL. Kjo teknikë funksionon duke krahasuar strukturën e një kërkese SQL të krijuar në mënyrë dinamike, përkundrejt kërkesës origjinale SQL. Për çdo të dhënë në hyrje nga përdorues të ndryshëm, kjo teknikë krijon disa inpute të thjeshta të njohura si të dhëna kandidate ose inpute kandidate dhe programi zbatohet njeherazi, mbi të dhënat aktuale në hyrje dhe të dhënave kandidate në hyrje. Nëse kontrolli i strukturës së dy kërkesave në hyrje nuk përkon, atëherë kërkesa aktuale refuzohet.

Mbi bazën e studimeve nga autorë të ndryshëm që kanë testuar dhe publikuar artikuj për teknikën Candid, kemi dalë në përfundimin, se teknika Candid i identifikon pjesërisht të gjitha sulmet me injektim SQL (Tabela 4.17). Mbi bazën e këtyre studimeve rezulton se gabimi pozitiv në këtë teknikë është konkret dhe gabimi negativ është i nënkuptuar.

Tabela 4.17. Teknika Candid në raport me sulmet me injektim SQL

Sulmet me injektim SQL	Candid
Tautologji	Pjesërisht
Piggy-backed kueri	Pjesërisht
Kueri llogjikisht gabim	Pjesërisht
Union kueri	Pjesërisht
Procedurat e ruajtura	Pjesërisht
Inference	Pjesërisht
Kodimi alternativ	Pjesërisht

Gjithashtu mbi bazën këtyre studimeve në tabelën 4.18 janë paraqitur disa karakteristika të kësaj teknike.

Tabela 4.18. Disa karakteristika të teknikës Candid

Teknika	Koha e identifikimit	Vëndi i identifikimit	Modifikimi kodit	Infrastrukturë shtesë
Candid[14]	Në kohën e ekzekutimit	Në serverin e aplikacionit	Jo	Nuk kërkohet

4.1.10 Anashkalimi ose Mënjanimi i Sulmeve me Injektim SQL

Autorët në artikullin [15], kanë propozuar një model për të identifikuar sulmet me injektim SQL, të bazuar në analizën statike dhe dinamike. Kjo metodë mënjanon ose anashkalon vlerat e atributeve të kërkesave SQL, gjatë kohës së ekzekutimit, sipas metodës dinamike, dhe i krahason ato me kërkesat SQL të analizuara më përpara, sipas metodës statike për të identifikuar sulmet me injektim SQL. Kur aplikacioni është në ekzekutim, secila kërkese e gjeneruar në mënyrë dinamike krahasohet me kërkesën e fiksuar më pare. Nëse rezulton zero, kjo kërkesë e veçantë lejohet të kalojë në bazën e të dhënave. Nëse nuk rezulton zero, atëherë kjo kërkesë raportohet si jo normale, dhe nuk kalon për ekzekutim në bazën e të dhënave.

Mbi bazën e studimeve nga autorë të ndryshëm që kanë testuar dhe publikuar artikuj për mënjanimin ose anashkalimin te kërkesës SQL, kemi dalë në përfundimin, se kjo teknikë i identifikon të gjitha sulmet me injektim SQL (Tabela 4.19).

Tabela 4.19. Teknika e mënjanimit të kërkesës SQL, në raport me sulmet me injektim SQL

Sulmet me injektim SQL	Menjanimi i kerkeses SQL [15]
Tautologji	E identifikon
Piggy-backed kueri	E identifikon

Kueri llogjikisht gabim	E identifikon
Union kueri	E identifikon
Procedurat e ruajtura	E identifikon
Inference	E identifikon
Kodimi alternativ	E identifikon

Gjithashtu mbi bazën këtyre studimeve në tabelën 4.20 janë paraqitur disa karakteristika të kësaj teknike.

Tabela 4.20. Disa karakteristika të teknikës së mënjanimit të kërkesës SQL

Teknika	Koha e identifikimit	Vëndi i identifikimit	Modifikimi kodit	Infrastrukturë shtesë
Mënjanimi i kerkeses SQL [15]	Në kohën e ekzekutimit	Në server	Jo	Nuk kërkohet

4.1.11 Parandalimi SQL

Autori në tezën e studimit të tij [16], ka paraqitur një teknikë që konsiston në sinjalizimin e kërkesave HTTP. Skema origjinale e kalimit të të dhënave modifikohet kur teknika e përdorur nga autori është vendosur në një web server. Kërkesat HTTP, ruhen në vëndin e përcaktuar për këtë qëllim. Më pas, sinjalizuesi SQL i kap instruksionet SQL, që janë ekzekutuar nga web aplikacioni dhe më pas i kalon ato në modulin zbulues ose kontrollues të sulmeve me injektim SQL. Për pasojë, kërkesa HTTP, thirret nga vëndi ku qëndrontë dhe kontrollohet për të përcaktuar nëse ajo përmban ndonjë nga sulmet me injektim SQL. Instruksionet e gabuara dhe të infektuara do të ndalohen të dërgohen në bazën e të dhënave, për sa kohë ekziston dyshimi se kjo kërkesë është një sulm me injektim SQL.

Mbi bazën e studimeve nga autorë të ndryshëm që kanë testuar dhe publikuar artikuj për teknikën e parandalimit SQL, kemi dalë në përfundimin, se kjo teknikë i identifikon të gjitha sulmet me injektim SQL (Tabela 4.21).

Tabela 4.21. Teknika e "Parandalimit SQL" në raport me sulmet me injektim SQL

Sulmet me injektim SQL	Parandalimi SQL[16]
Tautologji	E identifikon
Piggy-backed kueri	E identifikon
Kueri llogjikisht gabim	E identifikon
Union kueri	E identifikon
Procedurat e ruajtura	E identifikon
Inference	E identifikon

Kodimi alternativ	E identifikon

Gjithashtu mbi bazën këtyre studimeve në tabelën 4.22 janë paraqitur disa karakteristika të kësaj teknike.

Tabela 4.22. Disa karakteristika të teknikës së Parandalimit SQL

Teknika	Koha e identifikimit	Vëndi i identifikimit	Modifikimi kodit	Infrastrukturë shtesë
Parandalimi SQL[16]	Në kohën e ekzekutimit	Në serverin e aplikacionit	Jo	Nuk kërkohet

4.1.12 Modelet e Automatizuara dhe Manuale Kundër Sulmeve me Injektim SQL

Përpara se të analizojmë të dy modelet veç e veç, autorët në artikullin [30], kanë bërë një kombinim të dy modeleve, ata kanë propozuar një teknike të bazuar në disa modele kundër sulmeve me injektim SQL.

Në këtë teknikë, ata përdorin tre module për të arritur sigurinë sipas tyre. Modulin e monitorimit, i cili i merr të dhënat në hyrje nga web aplikacioni dhe i dërgon në modulin e analizimit. Moduli i analizimit, i cili zbulon të gjitha pikat kritike të web aplikacionit duke përdorur algoritmin që autorët kanë krijuar, i cili është një algoritëm krahasues i një vargu karakteresh, i cili funksionon në bazë të rregullit përça dhe sundo. Ky algoritëm i ruan të gjitha fjalët kyçe në modulin përfundimtar.

Autorët në artikullin [17], kanë përdorur të dy modelet kundër sulmeve me injektim SQL. Ata theksojnë përdorimin e modelit manual, me qëllim parandalimin e të dhënave në hyrje, të dëmtuara SQL. Në artikullin e tyre, në modelin manual është aplikuar një mënyrë programimi me tendencë mbrojtëse dhe shpesh herë teknika e përsëritjes se kodit. Në programimin me tendencë mbrojtjen, autorët kanë përdorur një mekanizem, që kontrollon të dhënat në hyrje të përdoruesve nga sulmet me injektim SQL, duke i penguar përdoruesit të shtojnë fjalë të tjera kyçe apo karaktere në kërkesën që i drejtohet bazës së të dhënave. Këtë autorët e arrijnë duke përdorur listat e bardha ose ato të zeza. Përsa i përket teknikës së përsëritjes së kodit [18], kjo teknikë është nje mënyrë me kosto të ulët që shërben për zbulimin e dëmeve të vogla, por sidoqoftë kjo kërkon njohuri të thella mbi sulmet me injektim SQL.

Siç e përmendëm edhe më lart në artikullin e tyre [17] të kërkimit shkencor, autorët nuk kanë përdorur vetem modelet manuale por theksojnë edhe përdorimin e atyre automatike. Në artikullin e tyre ata theksojnë se dy skemat kryesore janë:

- ✓ Analiza statike e gjetjes së gabimeve të vogla, të vendosura nga sulmues të ndryshëm në hyrje.
- ✓ Skanimi i sigurisë së webit.

E para zbulon gabimet e vogla të sulmeve me injektim SQL, skema e paralajmëron kur shikon diçka të tillë. Ndërkohë që për skemën e dytë autorët përdorin agjentët softuerik per të skanuar aplikacionet web, dhe për të identifikuar dobësitë, duke vëzhguar ose monitoruar sjelljen e tyre ndaj sulmeve me injektim SQL.

Mbi bazën e studimeve nga autorë të ndryshëm që kanë testuar dhe publikuar artikuj për modelet e automatizuara dhe manuale, kemi dalë në përfundimin, se kjo teknikë i identifikon të gjitha sulmet me injektim SQL, përveç sulmit me injektim SQL, në procedurat e ruajtura, të cilën këto modele, nuk e identifikojnë (Tabela 4.23).

Tabela 4.23. Modelet e automatizuara dhe manuale në raport me sulmet me sumet me injektim SQL

Sulmet me injektim SQL	Modelet e automatizuara dhe manuale[17]
Tautologji	E identifikon
Piggy-backed kueri	E identifikon
Kueri llogjikisht gabim	E identifikon
Union kueri	E identifikon
Procedurat e ruajtura	Nuk e identifikon
Inference	E identifikon
Kodimi alternativ	E identifikon

Gjithashtu mbi bazën këtyre studimeve në tabelën 4.24 janë paraqitur disa karakteristika të kësaj teknike.

Tabela 4.24. Disa karakteristika të modeleve të automatizuara dhe manuale

Teknika	Koha e identifikimit	Vëndi i identifikimit	Modifikimi kodit	Infrastrukturë shtesë
Modelet e automatizuara dhe manuale[17]	Në kohën e ekzekutimit	Në serverin e aplikacionit	Jo	Nuk kërkohet

4.1.13 DIWDB: Identifikimi i Sulmeve me Injektim SQL në Bazat e të Dhënave në Web

Autorët në artikullin [19], propozojnë një sistem për identifikimin e ndërhyrjeve në bazat e të dhënave në web. Ata përdorin modelin e tyre, i cili më tepër vepron në bazën e të dhënave se sa në instruksionet SQL, ose në fazën e veprimeve, për të identifikuar ndërhyrjet në aplikacionet web. Kjo metodë e tyre është efikase dhe mund të identifikojë sulmet me injektim SQL, dhe njëkohësisht dëmtimet llogjike që mund ti shkaktohen aplikacioneve.

Mbi bazën e studimeve nga autorë të ndryshëm që kanë testuar dhe publikuar artikuj për tekniken DIWDB, kemi dalë në përfundimin, se kjo teknikë identifikon vetëm sulmin me injektim SQL, Inference, në të cilën përfshihen dy sulme me injektim SQL, sulmet ndaj kohëvonesës së përgjigjes pas çdo ekzekutimi dhe injektimit të verbër. (Tabela 4.25).

Tabela 4.25. Teknika DIWDB në raport me sulmet me injektim SQL

Sulmet me injektim SQL	DIWDB[19]
Tautologji	Nuk e identifikon
Piggy-backed kueri	Nuk e identifikon
Kueri llogjikisht gabim	Nuk e identifikon
Union kueri	Nuk e identifikon
Procedurat e ruajtura	Nuk e identifikon
Inference	E identifikon
Kodimi Alternativ	Nuk e identifikon

Gjithashtu mbi bazën këtyre studimeve në tabelën 4.26 janë paraqitur disa karakteristika të kësaj teknike.

Tabela 4.26. Disa karakteristika të teknikës DIWDB

Teknika	Koha e identifikimit	Vëndi i identifikimit	Modifikimi kodit	Infrastrukturë shtesë
DIWDB[19]	Në kohën e ekzekutimit	Në serverin e aplikacionit	Jo	Nuk kërkohet

4.1.14 SQLIPA: Një Mekanizëm Autentifikimi Kundër Sulmeve me Injektim SQL

Autorët në artikullin [20], përdorin modelin e vlerave hash që të përmisojnë më tej sigurinë e autentifikimit të përdoruesve. Autorët përdorin emrin e përdoruesit dhe fjalëkalimin si vlera hash. SQLIPA, që i përshtatur në gjuhën shqipe nënkupton (Mbrojtësi i sulmeve me injektim SQL për autenticitetin) u zhvillua për të testuar modelin e tyre. Vlerat hash të përdoruesit dhe të fjalëkalimit u krijuan dhe u llogaritën në kohën e punës dhe autorët kanë krijuar një llogari të veçantë të përdoruesit.

Mbi bazën e studimeve nga autorë të ndryshëm që kanë testuar dhe publikuar artikuj për tekniken SQLIPA, kemi dalë në përfundimin, se kjo teknikë identifikon vetëm sulmin me injektim SQL, tautologji, dhe sulmet e tjera me injektim SQL nuk i identifikon (Tabela 4.27).

Tabela 4.27. Teknika SQLIPA në raport me sulmet me injektim SQL

Sulmet me injektim SQL	SQLIPA[20]
Tautologji	E identifikon
Piggy-backed kueri	Nuk e identifikon
Kueri llogjikisht gabim	Nuk e identifikon
Union kueri	Nuk e identifikon
Procedurat e ruajtura	Nuk e identifikon
Inference	Nuk e identifikon
Kodimi alternativ	Nuk e identifikon

Gjithashtu mbi bazën këtyre studimeve në tabelën 4.28 janë paraqitur disa karakteristika të kësaj teknike.

Tabela 4.28. Disa karakteristika të teknikës SQLIPA

Teknika	Koha e identifikimit	Vëndi i identifikimit	Modifikimi kodit	Infrastrukturë shtesë
SQLIPA[20]	Në kohën e ekzekutimit	Në serverin e aplikacionit	Jo	Nuk kërkohet

4.1.15 CSSE: Mbrojtja nga Sulmet me Injektim SQL përmes Vlerësimit Kontekstual, të një Vargu Karakteresh me Ndjeshmëri të Lartë

Ideja kryesore që kjo teknikë ka, është që të konstatojë shkakun kryesor të sulmeve me injektim SQL. Autorët në artikullin [21], kanë prezantuar një teknikë mbrojtëse dhe zbuluese të cilën ata e kanë emërtuar CSSE [21], e cila funksionon me anë të kombinimit të shtimit të të dhënave, të dhëna që ruajnë proçesin e vargëzimit, për t'u siguruar që këto të dhëna janë të ruajtura dhe aktuale, gjithashtu edhe me metodat e vlerësimit të vargëzimit të karaktereve në kontekstin e ndjeshmërisë së të dhënave, pra në atë sensitiv. Kjo teknikë nuk kërkon as modifikim të kodit as ndërhyrjen e zhvilluesit për ndryshime.

Autorët e kanë zbatuar këtë teknikë me PHP. Përpara ekzekutimit, kjo teknikë perdor analizën e kontekstit sensitiv për të identifikuar dhe kundërshtuar formulimet SQL, në kontrollin që bën nëse kërkesa SQL, që i drejtohet bazës së të dhënave, është ndërtuar me qëllim sulmin me injektim SQL ose jo.

Ideja kryesore e kësaj teknike është të modifikojë modelin e gjuhës, të diferencojë kërkesën origjinale të përdoruesit që vjen në formën e një vargu karaketeresh nga kërkesa statike.

Autorët në studimin e tyre kanë zbatuar dhe vlerësuar teknikën e tyre me phpBB. Ata kanë demostruar rezultate konkrete të performancës pozitive që kjo teknikë ka. Ndërkohë rezulton që gabimi pozitiv dhe ai negativ janë të nënkuptuar nga studimi.

Mbi bazën e studimeve nga artikulli [21] por dhe nga autorë të tjere që kanë testuar dhe publikuar artikuj për tekniken CSSE, kemi dalë në përfundimin, se kjo teknikë nuk i identifikon vetëm dy sulme me injektim SQL, sulmet me injektim SQL në procedurat e ruajtura dhe sulmet me injektim SQL, të kodimit alternativ (Tabela 4.29).

Tabela 4.29. Teknika CSSE në raport me sulmet me injektim SQL

Sulmet me injektim SQL	CSSE[21]
Tautologji	E identifikon
Piggy-backed kueri	E identifikon
Kueri llogjikisht gabim	E identifikon
Union kueri	E identifikon
Procedurat e ruajtura	Nuk e identifikon
Inference	E identifikon
Kodimi Alternativ	Nuk e identifikon

Gjithashtu mbi bazën këtyre studimeve në tabelën 4.30 janë paraqitur disa karakteristika të kësaj teknike.

Tabela 4.30. Disa karakteristika të teknikës CSSE

Teknika	Koha e identifikimit	Vendi i identifikimit	Modifikimi kodit	Infrastrukturë shtesë
CSSE[21]	Në kohën e ekzekutimit	Në serverin e aplikacionit	Jo	Kërkohet

4.2 Teknikat që Parandalojnë Sulmet me Injektim SQL

Më poshtë janë analizuar teknikat që parandalojnë sulmet me injektim SQL. Secila teknikë parandaluese do të studiohet në raport mbi sulmet me injektim SQL dhe mbi kriteret e zhvillimit dhe të vlerësimit në raport me veten e saj.

4.2.1 Waves: Nje Teknikë për Testimin e Sulmeve me Injektim SQL në Aplikacionet Web

Autorët në artikullin [22], propozojnë WAVES, një teknikë e kutisë së zezë, për testimin nga sulmet me injektim SQL, të aplikacioneve web. Kjo teknikë përcakton të gjitha pikat kritike në një aplikacion web, të cilat mund të përdoren për injektimin e sulmeve SQL. Më pas ajo ndërton disa modele sulmesh me injektim SQL dhe teknikash, të cilat kanë si objektiv kryesor këto pika kritike.

Kjo teknikë më pas vëzhgon përgjigjen e aplikacionit ndaj sulmeve me injektim SQL dhe përdor teknikat e makinave virtuale për të përmisuar metodologjinë e sulmit. Duke përdorur modelet e

makinave virtuale si model të testeve të saj, kjo teknikë përmison teknikat e krijimit dhe kalimit të testeve. Sidoqoftë, ashtu si të gjitha kutitë e zeza dhe teknikat e krijimit të testeve, nuk mund të japin siguri në rezultate.

Mbi bazën e studimeve nga artikulli [22] por edhe nga autorë të tjerë që kanë testuar dhe publikuar artikuj për tekniken Waves, kemi dalë në përfundimin, se kjo teknikë i parandalon pjesërisht të gjitha sulmet me injektim SQL (Tabela 4.31).

Tabela 4.31. Teknika Waves në raport me sulmet me injektim SQL

Sulmet me injektim SQL	Waves[22]
Tautologji	Pjesërisht
Piggy-backed kueri	Pjesërisht
Kueri llogjikisht gabim	Pjesërisht
Union kueri	Pjesërisht
Procedurat e ruajtura	Pjesërisht
Inference	Pjesërisht
Kodimi alternativ	Pjesërisht

Gjithashtu mbi bazën këtyre studimeve në tabelën 4.32 janë paraqitur disa karakteristika të kësaj teknike. Mbi bazën e studimit për këtë teknikë rezulton që gabimi pozitiv është konkret kurse gabimi negativ i nënkuptuar.

Tabela 4.32. Disa karakteristika të teknikës Waves

Teknika	Koha e identifikimit	Vëndi i identifikimit	Modifikimi kodit	Infrastrukturë shtesë
Waves[22]	Gjatë testimit	Në pjesën e klientit të aplikacionit	Jo	Nuk kërkohet

4.2.2 Kontrolluesit e Kodit Statik, Kontrolluesi JDBC (Java Database Connectivity)

Autorët në artikullin [2] [3], kanë propozuar kontrolluesin JDBC - një mjet për analizimin statik, që kontrollon plotësimin dhe saktësinë e një stringu të ndërtuar ose gjeneruar në mënyrë dinamike.

Kjo teknikë nuk është zhvilluar me qëllim parësor identifikimin dhe parandalimin e sulmeve me injektim SQL, por pavarsisht nga kjo, mund të përdoret për të penguar sulmet me injektim SQL, të cilat shfrytëzojnë mospërputhjen ose papërshtatjen e një stringu që gjenerehohet në mënyrë dinamike.

Kontrolluesi JDBC është në gjëndje të identifikojë një prej shkaqeve kryesore, nga ndodhin sulmet me injektim SQL, atë të kontrollit të të dhënave që në hyrje. Kontrolluesi-JDBC, përdoret për të përcaktuar gabimet në programimin e aplikacioneve ndërtuar në Java/JDBC.

Kjo teknikë përdoret për të zvogëluar gabimet potenciale të mundshme, ose për të përcaktuar mungesën e gabimeve në një kërkesë SQL të ndërtuar në mënyrë dinamike. Analiza që kjo teknikë bën, konsiston në përcaktimin e gabimeve semantike, si gabime të llojit SQL, të cilat arrihen me anë të aplikimit të një gjuhe të lirë dhe të efektshme për të provuar kontrollin semantik. Nëse gjëndet një gabim semantik ai raportohet menjëherë.

Sidoqoftë kjo teknikë, nuk do të mbulojë plotësisht sulmet me injektim SQL, sepse pjesa më e madhe e këtyre sulmeve rezultojnë korrekte gjatë ndërhyrjes nga ana sintaksore. Megjithatë kërkuesit shkencore kanë zhvilluar paketa të veçanta të cilat mund të aplikohen për ta bërë të sigurtë, një kërkesë që vjen nëpërmjet instruksioneve SQL[4].

Autorët në artikullin [5], propozojnë një përqasje e cila përdor analizën statike të kombinuar me arsyetimin e automatizuar për të verifikuar që kërkesat SQL të gjeneruara në shtresën e aplikimit nuk mund të përmbajnë sulmin me injektim SQL, tautologji. Një ndër mangësite kryesore te kësaj teknike është që hapësira e përdorimit të saj është e kufizuar për sa i takon identifikimit dhe parandalimit të sulmeve me injektim SQL të llojit tautologji dhe gjithashtu nuk mund të zbulojë llojet e tjera të sulmeve.

Autorët e kanë vënë në provë teknikën e tyre me një numër të larmishëm kodifikimesh të përshtatshme, të gjetura nga burime të ndryshme si nga interneti apo dhe në bashkëpunim me universitetet.

Autorët kanë argumentuar që kontrolluesi JDBC, është gjetur i suksesshëm në konstatimin e gabimeve të njohura dhe të panjohura në program.

Mbi bazën e studimeve nga artikulli [2][3] por dhe nga autorë të tjere që kanë testuar dhe publikuar artikuj për kontrolluesin JDBC, kemi dalë në përfundimin, se kjo teknikë i parandalon pjesërisht të gjitha sulmet me injektim SQL, në një formë apo në një tjetër (Tabela 4.33).

Tabela 4.33. Teknika e Kontrolluesit JDBC në raport me sulmet me injektim SQL

Sulmet me injektim SQL	Kontrolluesi JDBC
Tautologji	Pjesërisht
Piggy-backed kueri	Pjesërisht
Kueri llogjikisht gabim	Pjesërisht
Union kueri	Pjesërisht
Procedurat e ruajtura	Pjesërisht
Inference	Pjesërisht
Kodimi alternativ	Pjesërisht

Gjithashtu mbi bazën këtyre studimeve në tabelën 4.34 janë paraqitur disa karakteristika të kësaj teknike. Mbi bazën e studimit për këtë teknikë rezulton që gabimi pozitiv dhe gabimi negativ janë të nënkuptuar.

Tabela 4.34. Disa karakteristika të teknikës JDBC

Teknika	Koha e identifikimit	Vëndi i identifikimit	Modifikimi kodit	Infrastrukturë shtesë
Kontrolluesi JDBC[2][3]	Në kohën e kodimit	Në serverin e aplikacionit	Jo	Nuk kërkohet

4.2.3 SQL DOM: Kontrolli i Instruksioneve Dinamike SQL, në Kohën e Kompilimit

Autorët në artikullin [23], e kanë quajtur zbatimin e tyre, SQLDOM. Koncepti bazë i kësaj metode është inkurajimi i zhvilluesve të aplikacioneve për të përdorur një bashkësi klasash të cilat janë fuqimisht të rekomanduara dhe të përcaktuara në skemën e bazës së të dhënave. Në vënd të përdorimit të manipulimit të stringjeve për të ndërtuar kërkesa dinamike SQL, të programuara nga zhvilluesit, ky model përdor gjenerimin e instruksioneve SQL, të cilat do të rrisin nivelin e sigurisë. Ajo i eviton karakteret e gabuara ose të infektuara nga sulmet me injetkim SQL, duke parandaluar kështu sulmet me injektim SQL.

Autorët e kanë testuar SQLDOM-in dhe kanë prezantuar të dhëna konkrete, për të kontrolluar saktësinë dhe performancën. Mbi bazën e studimit të artikullit [23], por dhe nga autorë të tjerë që kanë testuar dhe publikuar artikuj për SQLDOM-in, kemi dalë në përfundimin, se kjo teknikë i parandalon të gjitha sulmet me injektim SQL, në një formë apo në një tjetër, me përjashtim të sulmeve me injektim SQL, në procedurat e ruajtura (Tabela 4.35).

Tabela 4.35. Teknika SQL DOM në raport me sulmet me injektim SQL

Sulmet me injektim SQL	SQL DOM[23]
Tautologji	E parandalon
Piggy-backed kueri	E parandalon
Kueri llogjikisht gabim	E parandalon
Union kueri	E parandalon
Procedurat e ruajtura	Nuk e parandalon
Inference	E parandalon
Kodimi alternativ	E parandalon

Gjithashtu mbi bazën këtyre studimeve në tabelën 4.36 janë paraqitur disa karakteristika të kësaj teknike. Mbi bazën e studimit për këtë teknikë rezulton që gabimi pozitiv dhe gabimi negativ janë të nënkuptuar.

Tabela 4.36. Disa karakteristika të teknikës SQL DOM

Teknika	Koha e identifikimit	Vëndi i identifikimit	Modifikimi kodit	Infrastrukturë shtesë
SQLDOM[23]	Gjatë kompilimit	Në serverin e aplikacionit	Po	Kërkohet

4.2.4 WebSSARI

Autorët në artikullin [24], kanë përdorur analizën statike, kundër parakushteve, për funksionet sensitive. Kjo teknikë është e bazuar në një mekanizëm kontrolli të të dhënave që në hyrje, të cilat kanë kaluar më parë në një numër të paracaktuar filtrash. Kufizimi i ketij modeli është sepse këto funksione të ndjeshme ose sensitive nuk mund të shprehen siç duhët, prandaj mund të eleminohen disa filtra në proçesin e kalimit të të dhënave.

Autorët e kanë testuar WebSSARI-n dhe kanë prezantuar të dhëna konkrete, për të kontrolluar saktesinë dhe performancën. Mbi bazën e studimit te artikullit [24], por dhe nga autorë të tjerë që kanë testuar dhe publikuar artikuj për WebSSARI-in, kemi dalë në përfundimin, se kjo teknikë i parandalon të gjitha sulmet me injektim SQL, në një formë apo në një tjetër (Tabela 4.37).

Tabela 4.37. Teknika WebSSARI në raport me sulmet me injektim SQL

Sulmet me injektim SQL	WebSSARI[24]
Tautologji	E parandalon
Piggy-backed kueri	E parandalon
Kueri llogjikisht gabim	E parandalon
Union kueri	E parandalon
Procedurat e ruajtura	E parandalon
Inference	E parandalon
Kodimi alternativ	E parandalon

Gjithashtu mbi bazën këtyre studimeve në tabelën 4.38 janë paraqitur disa karakteristika të kësaj teknike. Mbi bazën e studimit për këtë teknikë rezulton që saktësia e paraqitur nga gabimi pozitiv është konkrete dhe gabimi negativ i nënkuptuar.

Tabela 4.38. Disa karakteristika të teknikës WebSSARI

Teknika	Koha e	Vëndi i	Modifikimi	Infrastrukturë

	identifikimit	identifikimit	kodit	shtesë
WebSSARI[24]	Gjatë kompilimit	Në serverin e aplikacionit	Jo	Kërkohet

4.2.5 SecuriFLY: Zbulimi të Metave në Sigurinë e Aplikacioneve duke Perdorur PQL

Autorët në artikullin [25], kanë propozuar SecuriFLY, nje sistem sigurie që punon në kohën e ekzekutimit për aplikacionet web, duke përdorur PQL. Kjo teknikë identifikon dhe parandalon dobësitë e sulmeve me injektim SQL, po aq mirë sa dhe dobësite e disa sulmeve të tjera. Kjo teknikë funksionon duke ndjekur të dhënat në mënyrë të vazhdueshme.

Pavarsisht modeleve të tjera, kjo teknike ndjek një varg karakteresh, në vënd të një karakteri, për informacionin e sulmuar dhe përpiqet të kontrollojë kërkesën në formë stringu, por fatkeqësisht injektimi në fushat numerike nuk mund të ndalohet nga ky model.

Pra kjo teknikë ka veshtirësi në identifikimin e të gjitha burimeve që vinë nëpërmjet të dhënave në hyrje nga përdoruesit, që është dhe e meta kryesore e këtij modeli. Kjo teknikë mund të integrohet lehtësisht me web serverin, kështu që çdo aplikacion i ri mund të bashkohet automatikisht, sa herë që kalon në server.

Mbi bazën e studimit te artikullit [25], por dhe nga autorë të tjere që kanë testuar dhe publikuar artikuj për tekniken SecuriFLY, kemi dalë në përfundimin, se kjo teknikë i parandalon pjesërisht, për arsyet që i përmëndëm pak më lartë, të gjitha sulmet me injektim SQL, në një formë apo në një tjetër (Tabela 4.39).

Tabela 4.39. Teknika SecuriFLY në raport me sulmet me injektim SQL

Sulmet me injektim SQL	Securi FLY[25]
Tautologji	Pjesërisht
Piggy-backed	Pjesërisht
Ilegal incorrect	Pjesërisht
Union	Pjesërisht
Procedurat e ruajtura	Pjesërisht
Inference	Pjesërisht
Alternate Encondings	Pjesërisht

Gjithashtu mbi bazën këtyre studimeve në tabelën 4.40, janë paraqitur disa karakteristika të kësaj teknike.

Tabela 4.40. Disa karakteristika të teknikës SecuriFLY

Teknika	Koha e	Vëndi i	Modifikimi	Infrastrukturë

	identifikimit	identifikimit	kodit	shtesë
SecuriFLY[25]	Gjatë kompilimit	Në serverin e aplikacionit	Jo	Nuk kërkohet

4.2.6 Portat e Sigurisë

Autorët në artikullin [26], kanë paraqitur një sistem filtrimi të të dhënave, i cili përforcon rregullat e vlerësimit të të dhënave në hyrje gjatë kalimit në aplikacionet web. Duke përdorur gjuhën përshkruese të sigurisë SPDL, zhvilluesit sjellin pengesa dhe përcaktojnë transformimet që do të aplikohen në parametrat e aplikacionit, gjatë kalimit të tyre nga faqja web në serverin e aplikacionit. Zhvilluesit kanë liri të plotë në ndërtimin e politikave dhe për të shprehur veprimet e tyre, falë gjuhës SPDL. Kjo teknikë është e bazuar në programuesin e sistemit.

Autorët e kanë testuar teknikën e "portave të sigurisë" dhe kanë prezantuar të dhëna konkrete, për të kontrolluar saktesinë dhe performancën. Mbi bazën e studimit te artikullit [26], por dhe nga autorë të tjerë që kanë testuar dhe publikuar artikuj për portat e sigurisë, kemi dalë në përfundimin, se kjo teknikë i parandalon pjesërisht të gjitha sulmet me injektim SQL, në një formë apo në një tjetër (Tabela 4.41).

Tabela 4.41. Portat e sigurise në raport me sulmet me injektim SQL

Sulmet me injektim SQL	Portat e sigurise[26]
Tautologji	Pjesërisht
Piggy-backed kueri	Pjesërisht
Kueri llogjikisht gabim	Pjesërisht
Union kueri	Pjesërisht
Procedurat e ruajtura	Pjesërisht
Inference	Pjesërisht
Kodimi alternativ	Pjesërisht

Gjithashtu mbi bazën këtyre studimeve në tabelën 4.42 janë paraqitur disa karakteristika të kësaj teknike. Mbi bazën e studimit për këtë teknikë rezulton që saktësia e paraqitur nga gabimi pozitiv dhe gabimi negativ janë të nënkuptuara.

Tabela 4.42. Disa karakteristika të portave të sigurisë

Teknika	Koha e identifikimit	Vëndi i identifikimit	Modifikimi kodit	Infrastrukturë shtesë
Portat e sigurisë [26]	Në kohën e ekzekutimit	Në proksi server	Jo	Kërkohet

4.2.7 Shmangiet Pozitive dhe Vlerësimet e Sakta Gjuhësore

Në artikullin [27], autorët kanë prezantuar një model për parandalimin e sulmeve me injektim SQL, duke kontrolluar të dhënat në hyrje si vargje karakteresh. Në kohën reale, modeli i kategorizon stringjet në hyrje dhe shpërndan simbolet jo të besuara. Pas kësaj performohet një vlerësim nga ana e sintaksës, për evolimin e stringjeve të propaganduara. Kështu, bazuar në këtë vlerësim, nëse stringje jo të besuara, pra të infektuara, identifikohen, atëherë kërkesa të tilla kufizohen të kalojnë në serverin e bazës së të dhënave.

Gjatë fazës fillestare, të stringjeve të besuara, modeli kryen identifikimin bazuar në të dhënat në hyrje. Stringjet kategorizohen si stringje të koduara mire, stringje të krijuara nga Java, dhe stringje të pandryshuara, që burojnë nga burime të jashtme. Në rastin e modelit, që autorët kanë paraqitur, pra atë të vlerësimit sintaksor, ajo e kryen këtë analizë në pikën e ndërveprimit në bazën e të dhënave.

Sintaksa përcakton rrugët e duhura, pra të besueshme, të cilat jepen nëpërmjet funksioneve të përcaktuara nga zhvilluesi i webit. Funksionet kryejnë përshtatjen e modeleve, dhe nëse rezultati i përshtatjes jep të dhëna pozitive, modeli lejon kërkesën të ekzekutohet në serverin e bazës së të dhënave.

Mbi bazën e studimit te artikullit [27], por dhe nga autorë të tjerë që kanë testuar dhe publikuar artikuj për shmangiet pozitive, kemi dalë në përfundimin, se kjo teknikë i parandalon plotësisht, për arsyet që i përmëndëm pak më lartë, të gjitha sulmet me injektim SQL, në një formë apo në një tjetër (Tabela 4.43).

Tabela 4.43. Shmangiet pozitive në raport me sulmet me injektim SQL

Sulmet me injektim SQL	Shmangiet pozitive[27]
Tautologji	E parandalon
Piggy-backed kueri	E parandalon
Kueri llogjikisht gabim	E parandalon
Union kueri	E parandalon
Procedurat e ruajtura	E parandalon
Inference	E parandalon
Kodimi alternativ	E parandalon

Gjithashtu mbi bazën këtyre studimeve në tabelën 4.44, janë paraqitur disa karakteristika të kësaj teknike.

Tabela 4.44. Shmangiet pozitive, disa karakteristika

Teknika	Koha e identifikimit	Vëndi i identifikimit	Modifikimi kodit	Infrastrukturë shtesë
Shmangiet	Në kohën e	Në serverin e	Jo	Nuk kërkohet

pozitive [27]	ekzekutimit	aplikacionit		

4.3 Disa Zgjidhje për Identifikimin dhe Parandalimin e Sulmeve me Injektim SQL

Më poshtë do të trajtojmë disa zgjidhje për identifikimin dhe parandalimin e sulmeve me injektim SQL [41].

Autorët në artikullin [31], trajtojnë një algoritem që merret me instruksionet e përgatitura SQL, të cilat ndajnë vlerën e një kërkese nga struktura e SQL. Zhvilluesi përcakton strukturën e një kërkese SQL dhe më pas e plotëson këtë strukurë në kohën e punës ose të ekzekutimit. Instruksionet e përgatitura e bëjnë të veshtirë injektimin e kërkesave SQL, sepse struktura SQL nuk mund të ndryshojë.

Autoret në po këtë artikulli [31], kanë propozuar një algoritëm që gjeneron instruksionet e përgatitura në mënyrë automatike, për të c'vendosur dobësitë me injektim SQL. Autorët e kanë zbatuar punën e tyre duke përdorur disa projekte me kod të hapur. Duke u bazuar në rezultatet eksperimentale kodi i tyre i instruksioneve të përgatitura ishte në gjëndje të zëvëndësonte me sukses 94% të sulmeve me injektim SQL, në të gjitha projektet e tyre të përdorura. Për të përdorur instruksionet e përgatitura, ne duhet të modifikojmë web aplikacionin, ta rishkruajmë atë, për të ulur mundësinë e sulmeve me injektim SQL.

Autorët në artikullin [32], kanë propozuar një teknikë që përdor një skemë që gjeneron testet në mënyrë automatike për të identifikuar sulmet me injektim SQL. Ideja kryesore e kësaj teknike, bazohet në krijimin e një modeli specifik, që trajton kërkesat SQL në mënyrë automatike. Përveç kësaj, kjo përqasje përcakton lidhjen ose varësinë midis kërkesave dytësore. Bazuar në rezultatet e nxjerra nga autorët, kjo metodologji duket se është në gjëndje të identifikojë në mënyrë specifike sulmet me injektim SQL.

Autorët në artikullin [33], kanë propozuar një skemë që teston sigurinë e një baze të dhënash, të web aplikacioneve. Autorët në këtë artikull sygjerojnë disa momente kryesore, si identifikimi i pikave kritike të sulmeve me injektim SQL, të të dhënave në hyrje, së dyti, gjenerimin e rasteve të testimit në mënyre automatike, dhe në perfundim, zbulojnë dobësite e bazës së të dhënave, duke vënë në punë testet, për të simuluar sulmet në një aplikacion. Metodologjia e propozuar paraqitet si efiçente nga autorët, për sa kohë ajo është në gjëndje të identifikojë pikat kritike të sulmeve me injektim SQL, të të dhënave në hyrje, dhe ekzaktësisht në kohë. Sidoqoftë pas analizimit të skemës, ne kemi vënë re, që nuk është një zgjidhje e plotë dhe ka nevojë për përmisime në dy aspekte kryesore siç janë, aftësia e identifikimit dhe krijimin e një moduli më të gjërë për krijimin e sulmeve.

Autorët në artikullin [34], propozojnë një mekanizëm për tokëzimin e stringjeve, i cili krahason kërkesën origjinale me kërkesën e injektuar SQL, dhe krijon një grup të të dy argumentave të

kërkesës origjinale dhe asaj të injektuar SQL, nëse gjatësia e të dy kërkesave është e njejtë, do të thotë që nuk kemi injektim SQL, në të kundërt do të kemi injektim SQL.

Autorët në artikullin [35], kanë propozuar një mekanizëm që të mbron nga sulmet me injetkim SQL në procedurat e ruajtura. Procedurat e ruajtura janë ndër pjesët kryesore të një baze të dhënash, ku gjatë kërkesave të shumta që i drejtohen një baze të dhënash, aplikacionet shpesh herë i thërrasin këto procedura. Parandalimi në këto procedura të ruajtura, realizohet me anë të kombinimit të analizës statike dhe analizës së kohës së ekzekutimit. Analiza statike e përdorur për identifikimin e komandave arrihet nëpërmjet mekanizmit të kontrollit të vendosur në këto procedura dhe analizës së kohës së ekzekutimit duke përdorur SQLChecker për identifikimin e të dhënave në hyrje. Në disa modele propozohet një kombinim i analizës statike dhe monitorimit në kohën e ekzekutimit për të përforcuar sigurinë nga dobësite e mundshme.

Autorët në artikullin [36], kanë krijuar një algoritëm, i cili identifikon dhe parandalon sulmet me injektim SQL. Efiçenca e këtij algoritmi është krahasuar me teknikat e tjera, dhe performaca e tij kishte rezultate pozitive, dhe pse siguria nga sulmet me injektim SQL ishte në nivele të larta, prapë ishte e nevojshme për të rritur efiçencen në mbrojtjen e të dhënave të përdoruesve, në një aplikacion, për sa kohe të dhënat janë shume konfidenciale dhe të ndjeshme.

Autorët në artikullin [37], kanë përdorur SVM (Support Vector Machine) për klasifikimin dhe parashikimin e sulmeve me injektim SQL. Në algoritmin e propozuar nga autorët, saktësia e identifikimit të sulmeve me injektim SQL shkonte në 96%, e cila në bazë të studimit që ne po kryejmë është përqindja më e lartë midis teknikave të tjera identifikuese për identifikimin e sulmeve me injektim SQL.

Autorët në artikullin [38], kanë propozuar një sistem për identifikimin edhe të sulmeve me injektim SQL. Në punën e tyre autorët kanë vlerësuar performancën e këtij sistemi në një aplikacion web. Rezultatet e testimet tregojnë që teknika e përdorur është efektive në identifikimin e sulmeve dhe e realizon këtë gjë duke pasur një performancë të mirë.

Autorët në artikullin [39], kanë përdorur një metodologji që punon në mënyrë të pavarur, e cila ka për qëllim të pergjithsoj strukturën sintaktike të një kërkese SQL dhe të kontrolloj të dhënat në hyrje të përdoruesve. Kërkesat SQL, që vijnë nga të dhënat e përdoruesve në hyrje, kalojnë në një mekanizëm kontrolli të pavarur e cila kontrollon strukturën sintaktike të kërkesës. Një ndër avantazhet e kësaj teknike është që mesazhi i gabimit që gjenerohet nuk përmban ndonjë informacion të rëndësishëm për bazën e të dhënave, i cili do të ndihmonte sulmuesin. Për sa kohë kjo teknikë nuk është e integruar me web aplikacionin, çdo ndryshim ose modifikim që do ti bëhej sistemit, do ti bëhej në atë formë që të pranohej dhe nga teknika e përdorur.

Në artikullin [40] autorët kanë propozuar një model bazuar në përkthimin dhe kontrollin e kërkesave, për identifikimin dhe parandalimin e sulmeve me injektim SQL. Ideja bazë e këtij modeli bazohet në mënyrën se si baza të dhënash të ndryshme interpretojnë kërkesat SQL dhe kërkesat e injektuara SQL. Pas një analizimi të detajuar se si baza të dhënash të ndryshme

interpretojnë kërkesat e ndryshme SQL, autorët kanë propozuar një zgjidhje efektive TransSQL, nëpërmjet së cilës kërkesat SQL janë ekzekutuar në dy baza të ndryshme të dhënash duke analizuar përgjigjet e gjeneruara.

4.4 Analiza Krahasuese e Teknikave të Identifikimit dhe Parandalimit të Sulmeve me Injektim SQL në Raport me Llojet e Sulmeve me Injektim SQL

Në këtë pikë të studimit, teknikat që identifikojnë dhe parandalojnë sulmet me injektim SQL, të trajtuara në kapitujt më lart, do të krahasohen në raport me sulmet e ndryshme me injektim SQL [41].

Eshtë e vështirë për të dhënë në mënyrë të saktë, secila skemë ose model është më e mirë se tjetra, meqë secila prej tyre ka përfitime specifike për sisteme specifike. Në këtë pjesë të studimit ne do të analizojmë se si skemat ose modelet e ndryshme funksionojnë ndaj sulmeve me injektim SQL. Për arsye të studimit ne i kemi ndarë këto teknika në dy kategori:

- ✓ Teknikat që identifikojnë sulmet me injektim SQL
- ✓ Teknikat që parandalojnë sulmet me injektim SQL

Duhet theksuar se ky studim, që përfshin krahasimin e këtyre teknikave është i bazuar në artikujt e publikuar dhe jo në eksperiencën empirike. Së fundmi, ne vlerësojmë kërkesat që nevojiten për zhvillimin e secilës teknikë.

4.4.1 Krahasimi i Teknikave të Identifikimit të Sulmeve me Injektim SQL në Raport me Sulmet me Injektim SQL

Teknikat e identifikimit të sulmeve me injektim SQL, janë teknika që i identifikojnë ose zbulojnë sulmet me injektim SQL përgjithsisht në kohën e ekzekutimit. Tabela 4.45, tregon raportin ose krahasimin e skemave ose teknikave që identifikojnë sulmet me injektim SQL, në raport pikërisht me këto sulme me injektim SQL [41].

Simboli √, është përdorur për teknikat që mund të identifikojnë në mënyrë të suksesshme, të gjitha sulmet me injektim SQL të atij lloji.

Simboli ×**,** është përdorur për teknikat të cilat nuk janë në gjëndje të identifikojnë asnjë lloj sulmi me injektim SQL të atij lloji.

Simboli □, i referohet të gjitha atyre teknikave që i identifikojnë pjesërisht të gjitha sulmet me injektim SQL, të atij lloji, për shkak të kufizimeve natyrale që ato kanë.

Tabela 4.45. Krahasimi i teknikave të identifikimit të sulmeve me injektim SQL në raport me sulmet me injektim SQL [41]

Teknikat e identifikimit në raport me sulmet me injektim SQL	SAFELI [11]	SQL-IDS [12]	Swaddler [10]	Parandalimi SQL [16]	SQLrand [1]	SQLIPA [20]	AMNESIA [7]	Modelet automatike dhe manuale[17]	CANDID [14]	DIWeDa [19]	Kontrolluesi tautologji [35]	Mënjanimi I kërkesës SQL [15]	SQLCheck [9]	SQL Guard [6]
Tautologji	×	√	□	√	√	√	√	√	□	×	√	√	√	√
Piggy-backed kueri	√	√	□	√	√	×	√	√	□	×	×	√	√	√
Kueri llogjikisht gabim	√	√	□	√	√	×	√	√	□	×	×	√	√	√
Union kueri	√	√	□	√	√	×	√	√	□	×	×	√	√	√
Procedurat e ruajtura	√	√	□	√	×	×	×	×	□	×	×	√	×	×
Inference	√	√	□	√	√	×	√	√	□	√	×	√	√	√
Kodimi alternativ	√	√	□	√	√	×	√	×	□	×	×	√	√	√

Tabela 4.46, paraqet përqindjen e identifikimit të sulmeve me injektim SQL, në raport me teknikat e identifikimit të ketyre sulmeve me injektim SQL [41]. Përqindja e teknikave që identifikojnë një nga sulmet me injektim SQL, është llogaritur duke marrë numrin e teknikave që e kanë identifikuar sulmin e parë në raport me numrin total të të gjitha teknikave.

Tabela 4.46. Krahasimi në përqindje i teknikave të identifikimit të sulmeve me injektim SQL në raport me sulmet me injektim SQL [41]

Krahasimi në %	*Teknikat që i identifikojnë plotësisht të gjitha sulmet e këtij lloji (√)*	*Teknikat që i identifikojnë pjesërisht të gjitha sulmet e këtij lloji (□)*	*Teknikat që nuk janë në gjëndje të identifikojnë sulmet e këtij lloji (×)*
Tautologji	72%	14%	14%
Piggy-backed kueri	64%	14%	22%

Kueri llogjikisht gabim	64%	14%	22%
Union kueri	64%	14%	22%
Procedurat e ruajtura	29%	14%	57%
Inference	72%	14%	14%
Kodimi alternativ	57%	14%	29%

4.4.2 Krahasimi i Teknikave të Parandalimit të Sulmeve me Injektim SQL, në Raport me Llojet e Sulmeve me Injektim SQL

Teknikat e parandalimit janë teknika që identifikojnë dhe parandalojnë dobësitë në mënyrë statistike në kodin burim. Tabela 4.47, tregon raportin ose krahasimin e skemave ose teknikave që parandalojnë sulmet me injektim SQL, në raport pikërisht me llojet e sulmeve me injektim SQL [41].

Simboli √, është përdorur për teknikat që mund të parandalojnë në mënyrë të suksesshme, të gjitha sulmet me injektim SQL të këtij lloji.

Simboli ×**,** është përdorur për teknikat të cilat nuk janë në gjëndje të parandalojnë asnjë nga sulmet me injektim SQL, të secilit lloj sulmi veç e veç.

Simboli □, i referohet të gjitha atyre teknikave që i parandalojnë pjesërisht të gjitha sulmet me injektim SQL, të atij lloji përkatës, për shkak te kufizimeve natyrale që ato kanë.

Tabela 4.47. Krahasimi i teknikave të parandalimit të sulmeve me injektim SQL në raport me sulmet me injektim SQL [41]

Teknikat e parandalimit në raport me sulmet me injektim SQL	Kontrolluesi JDBC [2][3]	Shmangiet pozitive [27]	SecuriFly [25]	Portat e sigurisë [26]	SQLDOM [23]	WAVES [22]	WebSSARI [24]
Tautologji	□	√	□	□	√	□	√
Piggy-backed kueri	□	√	□	□	√	□	√
Kueri llogjikisht gabim	□	√	□	□	√	□	√

Union kueri	□	√	□	□	√	□	√
Procedurat e ruajtura	□	√	□	□	×	□	√
Inference	□	√	□	□	√	□	√
Kodimi alternativ	□	√	□	□	√	□	√

Tabela 4.48, paraqet përqindjen e parandalimit të sulmeve me injektim SQL, në raport me teknikat e parandalimit të këtyre sulmeve me injektim SQL [41].

Përqindja e teknikave që parandalojnë një ndër sulmet me injektim SQL, është llogaritur duke marrë numrin e teknikave që e kanë parandaluar sulmin e parë në raport me numrin total të të gjitha teknikave.

Tabela 4.48. Krahasimi në përqindje i teknikave të parandalimit të sulmeve me injektim SQL në raport me sulmet me injektim SQL [41]

Krahasimi në %	*Teknikat që i parandalojnë plotësisht të gjitha sulmet e këtij lloji (√)*	*Teknikat që i parandalojnë pjesërisht të gjitha sulmet e këtij lloji (□)*	*Teknikat që nuk janë në gjëndje të parandalojnë sulmet e këtij lloji (×)*
Tautologji	43%	57%	0%
Piggy-backed kueri	43%	57%	0%
Kueri llogjikisht gabim	43%	57%	0%
Union kueri	43%	57%	0%
Procedurat e ruajtura	29%	57%	11%
Inference	43%	57%	0%
Kodimi alternativ	43%	57%	0%

4.4.3 Krahasimi i Teknikave të Parandalimit dhe Identifikimit të Sulmeve me Injektim SQL, Duke u Bazuar në Kriteret e Vlerësimit dhe Zhvillimit të Tyre

Mbi bazën e studimeve të trajtuara më lartë nga autorë të ndryshem mbi teknikat e parandalimit dhe identifikimit të sulmeve me injektim SQL, ne kemi bërë një krahasim të këtyre teknikave, duke u bazuar në kriteret e vlerësimit dhe zhvillimit të tyre.

Rezultatet e këtij krahasimi [41], janë paraqitur si më poshte në tabelën 4.49. dhe tabelën 4.50.

Tabela 4.49. Krahasimi i teknikave të identifikimit të sulmeve me injektim SQL duke u bazuar ne kriteret e vleresimit dhe zhvillimit te tyre [41]

Teknikat	Koha e identifikimit	Vëndi i identifikimit	Modifikimi kodit	Infrastrukturë shtesë
SQLRand [1]	Në kohën e ekzekutimit	Proksi server	Po	Nuk kërkohet
SQLGuard [6]	Në kohën e ekzekutimit	Në serverin e aplikacionit	Po	Kërkohet
Amnesia [7]	Në kohën e ekzekutimit	Në serverin e aplikacionit	Jo	Nuk kërkohet
SQLCheck [9]	Në kohën e ekzekutimit	Proksi server	Po	Kërkohet
Swaddler [10]	Në kohën e ekzekutimit	Në serverin e aplikacionit	Jo	Kërkohet
Sania[28]	Gjatë testimit	Klient dhe Server	Jo	Nuk kërkohet
SAFELI [11]	Në kohën e kompilimit	Në serverin e aplikacionit	Jo	Nuk kërkohet
SQL-IDS [12]	Në kohën e ekzekutimit	Në serverin e aplikacionit	Jo	Nuk kërkohet
Candid[14]	Në kohën e ekzekutimit	Në serverin e aplikacionit	Jo	Nuk kërkohet
Mënjanimi i kerkeses SQL [15]	Në kohën e ekzekutimit	Në server	Jo	Nuk kërkohet
Parandalimi SQL[16]	Në kohën e ekzekutimit	Në serverin e aplikacionit	Jo	Nuk kërkohet
Modelet e automatizuara dhe manuale[17]	Në kohën e ekzekutimit	Në serverin e aplikacionit	Jo	Nuk kërkohet
DIWDB[19]	Në kohën e ekzekutimit	Në serverin e aplikacionit	Jo	Nuk kërkohet
SQLIPA[20]	Në kohën e ekzekutimit	Në serverin e aplikacionit	Jo	Nuk kërkohet
CSSE[21]	Në kohën e ekzekutimit	Në serverin e aplikacionit	Jo	Kërkohet

Tabela 4.50. Krahasimi i teknikave të parandalimit të sulmeve me injektim SQL duke u bazuar në kriteret e vlerësimit dhe zhvillimit të tyre [41]

Teknikat	Koha e identifikimit	Vëndi i identifikimit	Modifikimi kodit	Infrastrukturë shtesë

Kontrolluesi JDBC[2][3]	Në kohën e kodimit	Në serverin e aplikacionit	Jo	Nuk kërkohet
Shmangiet pozitive [27]	Në kohën e ekzekutimit	Në serverin e aplikacionit	Jo	Nuk kërkohet
SecuriFLY[25]	Gjatë kompilimit	Në serverin e aplikacionit	Jo	Nuk kërkohet
Portat e sigurisë [26]	Në kohën e ekzekutimit	Në proksi server	Jo	Kërkohet
SQLDOM[23]	Gjatë kompilimit	Në serverin e aplikacionit	Po	Kërkohet
Waves[22]	Gjatë testimit	Në anen e klientit të aplikacionit	Jo	Nuk kërkohet
WebSSARI[24]	Gjatë kompilimit	Në serverin e aplikacionit	Jo	Kërkohet

4.5 Analiza dhe Konkluzionet për Kapitullin e Katërt

Në këtë kapitull, ne paraqitëm një studim mbi teknikat e identifikimit dhe parandalimit të sulmeve me injektim SQL. Pamë të metat dhe veçoritë e ketyre teknikave dhe gjithashtu i studiuam këto teknika identifikuese dhe parandaluese, në raport me sulmet me injektim SQL, të trajtuara në kapitullin e dytë.

Për ta kryer këtë studim, në fillim ne percaktuam llojet e ndryshme të sulmeve me injektim SQL. Gjithashtu investiguam teknikat e identifikimit dhe parandalimit të sulmeve me injektim SQL. Më pas, krahasuam këto teknika, duke u bazuar në kriteret e zhvillimit dhe vlerësimit të tyre.

Dy sulme me injektim SQL, të cilat jane "Procedurat e ruajtura" dhe "Kodimi alternativ" krijonin më tepër problem për teknikat e identifikimit dhe parandalimit. Nga rezultatet më lartë, duket qartë që pothuajse të gjitha sulmet me injektim SQL, janë adresuar në mënyrë të qëndrueshme nga teknikat e identifikimit dhe parandalimit të sulmeve me injektim SQL, me perjashtim të sulmeve që ndodhin në procedurat e ruajtura, të cilat nuk mund të ndalohen nga disa teknika.

Eshtë evidente që vetem 29% e teknikave, mund të identifikojnë dhe parandalojnë plotësisht sulmin me injektim SQL, që ndodh në procedurat e ruajtura. Nga rezultatet e studimit duket qartë që sulmi me injektim SQL, që ndodh në procedurat e ruajtura nuk identifikohet nga 57% e teknikave të identifikimit të sulmeve me injektim SQL. Ndërkohë duket qartë që sulmet e tjera me injektim SQL, janë të identifikueshme plotësisht dhe të parandalueshme pjesërisht nga afërsisht 60% e teknikave të identifikimit dhe parandalimit të sulmeve me injektim SQL.

Ishte një studim i vështire sepse autorë të ndryshëm i kanë prezantuar punët e tyre në nivele të ndryshme, dhe duke i marrë këto të dhëna nga artikuj të ndryshëm, puna bëhej akoma më e veshtirë.

Puna jonë në të ardhmen për këtë kapitull do të jetë zgjerimi i njohurive për këto teknika përsa i përket kritereve të vlerësimit.

5. Analiza dhe Projektimi i Shtresës Mbrojtëse Kundër Sulmeve me Injektim SQL

Në këtë kapitull, do të trajtojmë mekanizmin tonë mbrojtës kundër sulmeve me injektim SQL [42]. Metodologjia jonë e përdorur për tu mbrojtur nga sulmet me injektim SQL, është e bazuar në krijimin e një shtrese e cila kontrollon dhe analizon kërkesat e dërguara nga klientet ose përdoruesit në drejtim të bazave të të dhënave, nëse janë të infektuara nga sulmet injektim SQL apo jo. Shtresa jonë mbrojtëse është lehtësisht e manovrueshme dhe tepër praktike, që do të thotë, ajo mund të përdoret me cdo web aplikacion. Shtresa jonë mbrojtëse do të analizoje cdo kërkesë duke qëndruar midis përdoruesve dhe serverit të aplikacionit, por ajo mund të veprojë edhe duke qënë pjese e aplikacionit.

5.1 Skema e Projektimit të Shtresës Tonë Mbrojtëse Kundër Sulmeve me Injektim SQL

Skema e projektimit të shtresës tonë mbrojtëse [42], kundër sulmeve me injektim SQL, konsiston në katër pjesë: klientët, shtresa mbrojtëse, serveri i aplikacionit dhe serveri i bazës së të dhënave, figura 5.1. Në këtë skemë shtresa jonë mbrojtëse adreson problemin e sulmeve me injektim SQL.

Me shtresën mbrojtëse ne do të kuptojmë një mekanizëm ose program që vepron midis klientit ose përdoruesve dhe serverit të aplikacionit. Çdo kërkesë ose e dhëne që vjen nga përdoruesit në hyrje do të kontrollohet dhe analizohet nga shtresa jonë mbrojtëse, para se ajo të kalojë në serverin e aplikacionit. Nëse kërkesa eshtë e infektuar ose injektuar nga sulmet me injektim SQL të cilat ne i kemi trajtuar gjerësisht ne artikujt [41] dhe [42], por gjithashtu edhe në kapitullin 2, do të jetë një kërkesë që nuk do të kalojë në bazën e të dhënave. Qëllimi i këtij studimi dhe i kësaj shtrese është të parandalojë sulmet me injektim SQL, që të kalojnë në bazën e të dhënave, pra të ndalojë kerkesat ilegale.

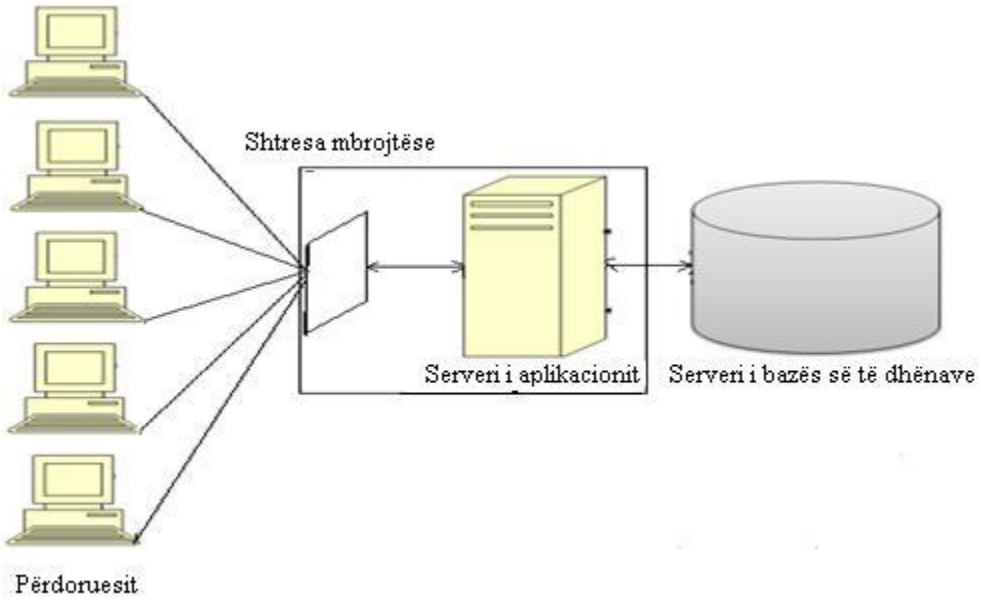

Figura 5.1. Projektimi dhe zbatimi i shtresës tonë mbrojtëse kundër sulmeve me injektim SQL [42]

Siç vihet re edhe nga skema e propozuar më lart, kërkesat SQL gjenerohen nga serveri i aplikacionit. Shtresa jonë mbrojtëse do të kontrollojë nëse ka prezence të sulmeve me injektim SQL në kerkesën që i drejtohet bazës së të dhënave, përpara se të dhënat në hyrje të kalojnë në serverin e aplikacionit. Në skemën e propozuar në figurën.5.1, ne kemi venë në punë shtresën tonë mbrojtëse për të parandaluar sumet me injektim SQL .

Rezultatet tregojnë që është e mjaftueshme vetëm prania e kësaj shtrese mbrojtëse midis klientit ose përdoruesve dhe serverit të aplikacionit të çdo websajti. Me ndihmën e kësaj shtrese ne do të jemi në gjëndje të parandalojmë sulmet me injektim SQL.

5.2 Zbatimi i Shtresës Tonë Mbrojtëse Kundër Sulmeve me Injektim SQL

Zbatimi i skemës ku perfshihet shtresa jonë mbrojtëse, u realizua duke përdorur disa aplikacione në drejtime të ndryshme (Tabela 5.1). Për zbatimin e saj u përdor Visual Studio Ultimate, SQL server dhe tre skanerave që testojnë sigurinë dhe tregojnë efiçencën e skemës tone, duhet theksuar që të tre skanerat që janë zgjedhur për testimin e shtresës tonë mbrojtëse, janë skanera të specializuara për sulmet me injektim SQL. Nëpërmjet këtyre tre skanerave ne do të testojmë marrëdhënien që kanë midis tyre websajti dhe aplikacioni, në raport me rrezikun nga sulmet me injektim SQL.

Për të provuar efiçencën e shtresës tonë mbrojtëse midis klienteve dhe serverit të aplikacionit, siç e përmëndëm edhe më lart, ne e kemi testuar dhe demostruar atë në aplikacione të ndryshme (Tabela 5.1) të lidhur sigurisht me bazën e të dhënave në SQL server.

Tabela 5.1. Aplikacionet e testuara me dhe pa shtresën mbrojtëse

Tematika e websajtit	Publike apo lokale
Agjenci udhëtimi	Publike
Universitet privat	Publike
Kompani veglash ndërtimi	Publike
Restorant	Publike

Projekti që u demostrua nëpërmjet websajtit u testua në dy momente, në një gjëndje të mbrojtur nga shtresa jonë mbrojtëse dhe në një gjëndje të pambrojtur nga shtresa jonë mbrojtëse. Në figuren 5.2 dhe ne figuren 5.3 duket qartë momenti kur ne e testojmë projektin tonë në këto dy momente.

```
</h2>
<p>
<asp:RadioButtonListID="RadioButtonList1"="server">
<asp:ListItemSelected="True">me_shtresën_mbrojtëse</asp:ListItem>
<asp:ListItem>pa_shtresën_ mbrojtëse</asp:ListItem>
</asp:RadioButtonList>
</p>
<p>
```

Figura 5.2. Demonstrimi në dy momente

5.2.1 Zbatimi me shtresën mbrojtëse

Me shtresën mbrojtëse: Në këtë rast websajti vepron nën kontrollin dhe drejtimin e shtresës tonë mbrojtëse, e cila kontrollon çdo kërkesë që vjen nga të dhënat në hyrje dhe është gjithmone gati për të parë nëse ka ndonjë ndërhyrje ose një kodim të gabuar në kërkesën që i drejtohet një baze të dhënash.

Figura 5.3. Zbatimi me shtresën mbrojtëse

Në këtë rast shtresa mbrojtëse kontrollon çdo kërkesë dhe pasi sigurohet që ajo është e pastër, e dërgon atë në serverin e aplikacionit dhe më tej. Një ndër veçantitë e shtresës tonë mbrojtëse dhe e strukturës tonë është se ajo është lehtësisht e zbatueshme, dhe mund të mos varet nga serveri i aplikacionit dhe për më tepër nga baza e të dhënave. Shtresa jonë mbrojtëse nuk krijon ndonjë ngarkesë të bazës së të dhënave apo vonesë në proçesimin e të dhënave.

Shtresa jonë mbrojtëse gjithashtu parandalon modelet e sulmeve me injektim SQL dhe injektimet në pikat kritike të kërkesës që i drejtohet web aplikacionit dhe bazës së të dhënave. Duke u bazuar në studimin mbi sulmet me injektim SQL të zhvilluar më lart, ne kemi krijuar një listë me fjalë kyçe të cilat injektohen në kërkesat SQL, të cilat sigurisht kanë si qëllim të shkaktojnë dëmtimin e bazës së të dhënave.

Zbatimi i kësaj liste në shtresën tonë ka si qëllim identifikimin e ndërhyrjeve në kërkesat SQL dhe dërgimin e një mesazhi tek administratori i bazës së të dhënave, i cili si përgjigje do ti kthejë një mesazh përdoruesit sulmues dhe do të ndalojë të gjitha veprimet e kryera nga ky përdorues duke i bllokuar edhe adresën IP të tij (Figura 5.6). Kjo mënyre parandalon dëmtimin e bazës së të dhënave. Në figurën 5.4, paraqiten modelet e fjalëve kyçe që realizojnë më pas sulmin me injektim SQL nga ana e përdoruesit sulmues, të cilat shtresa jonë mbrojtëse i parandalon.

```
private string[] AttackList =
{"--",
";--",
";",
"/*",
"*/",
"@@",
"@@version",
"@@VERSION",
"char",
"int",
"nchar",
"varchar",
"nvarchar",
"fetch",
"union",
"UNION",
"rename",
"RENAME",
"insert",
"INSERT",
"kill",
"open",
"select",
"SELECT",
"alter",
"ALTER",
"begin",
"cast",
"create",
"CREATE",
```

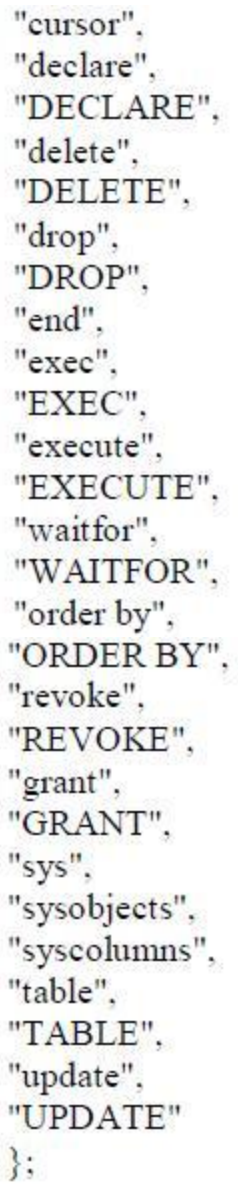

Figura 5.4. Lista e kontrollit të fjalëve kyçe

Në figurën 5.5 është dhënë një pjesë e kodit që ndihmon web aplikacionin të jetë më i sigurtë nga sulmet me injektim SQL, sepse ne pak më parë përmëndëm që shtresa jonë mund të qëndrojë e pavarur por dhe si pjesë e web aplikacionit. Pas analizimit të sulmeve me injektim SQL, shtresa mbrojtëse i parandalon ato.

```
namespace WebApplication.Security
   {
publicpartialclassWebForm : System.Web.UI.Page
   {
privatestring SelectStatment1 = "SELECT Id, UserMessage FROM aspnet_UserMessages
WHERE Id = ";

protectedvoidPage_Load(object sender, EventArgs e)
      {
if (!IsPostBack)
         {
            Label1.Text = SelectStatment1;
         }
if (IsPostBack)
         {
ExecuteSelectStatment();
         }
      }
protectedvoid Button1_Click(object sender, EventArgs e)
      {
      }
protectedvoidExecuteSelectStatment()
      {
StringstrConnect;
StringstrCommand;
strConnect =@"DataSource=.\SQLEXPRESS;AttachDbFileName=|DataDirectory|aspnetdb.mdf;
                     Integrated Security=True;User Instance=True";
SqlConnectionmyConn = newSqlConnection(strConnect);
if (RadioButtonList1.SelectedIndex == 0)
      {
strCommand = SelectStatment1 + ReadSqlInjection();

    }
  else
         {
  strCommand = SelectStatment1 + ProtectiveLayerSqlInjection();
         }
  SqlDataAdaptermyData = newSqlDataAdapter(strCommand, myConn);
  DataSet DataSet1 = newDataSet();
  myData.Fill(DataSet1, "UserMessages");
  DataViewmyView = newDataView(DataSet1.Tables["UserMessages"], "", "Id",
  DataViewRowState.CurrentRows);
  DataGrid1.DataSource = myView;
  DataGrid1.DataBind();
  myConn.Close();
         }
  protectedstring ReadSqlInjection()
         {
  string strMessage = TextBox1.Text;
  if (strMessage.Equals(""))
            {
               Label2.Text = "Debug";
  return"1";
            }
  else
            {
               Label2.Text = "Debug";
  returnstrMessage;
  }
         }
```

```
protectedstring ProtectiveLayerSqlInjection()
      {
string strSqlInjection = TextBox1.Text;
RegexrxNums = newRegex(@"^\d+$");
if (rxNums.IsMatch(strSqlInjection))
        {
          Label2.Text = "Debug";
return strSqlInjection;
        }
else
        {
          Label2.Text = "Sulmi me injektim SQL u bllokua";
return"1";
        }
      }
   }
}
```

Figura 5.5. Si pjesë e aplikacionit

Ne figurën 5.6 është paraqitur një pjesë e kodit të parandalimit të sulmeve me injektim SQL dhe dërgimin e një mesazhi tek administratori i bazës së të dhënave, i cili si përgjigje do ti kthejë një mesazh përdoruesit sulmues dhe do të ndalojë të gjitha veprimet e kryera nga ky përdorues duke i bllokuar edhe adresën IP të tij.

```
if (objds_UserRegistrations.Count > 0)
{
Session["LoginName"] = objds_UserRegistrations[0].Username;
if (objds_UserRegistrations[0].IsAdmin == "Y")
{
Response.Redirect("~/AdminPages/AdminHomePage.aspx");
}
else
{
bs_SQLInjectAttackDetails objbs_SQLInjectAttackDetails =
bs_SQLInjectAttackDetailBase.SelectByField("InjectUserName",
Session["LoginName"].ToString());
if (objbs_SQLInjectAttackDetails.Count > 3)
{
Alert.Show("IP u bllokua nga Administratori");
}
Else
{
Response.Redirect("~/UserPages/UserHomePage.aspx");
}
}
}
else
{
Alert.Show("Gabim ne fjalekalim ose ne ID e perdoruesit ");
}
}
}

private bool isValidSubmitData()
{
bool r = false;
if (!string.IsNullOrEmpty(TxtName.Text))
{
if (!Regex.IsMatch(TxtName.Text, @"^[a-zA-Z"-'\s]{1,40}$"))
{
Alert.Show("Fraza e gabuar");

return r;
}
else if (IDS.CheckInput(TxtName.Text))
{
Alert.Show( "Instruksion i dyshimte / Elemente shtese jane gjetur ");
TraceInject();
return r;
}
}
```

Figura 5.6. Disa pika kritike të parandalimit të sulmeve me injektim SQL

Shtresa jonë mbrojtëse nuk kontrollon vetëm të dhënat në hyrje të përdoruesve që vijnë nga aplikacionet nëpërmjet websajteve të ndryshme, por gjithashtu kontrollon edhe linkun, sepse në vetvete çdo kërkesë vjen në formën e një linku. Dhe ky link merret nga shtresa jonë mbrojtëse nëpërmjet kodit të mëposhtëm e gjitha si një kërkesë e zakonshme që do të përfundojë në bazën e të dhënave dhe do të kontrollohet në formën e një inputi të zakonshëm.

```
HttpRequest request1 = (HttpRequest)request1;
String query = request1.getQueryString();
```

Shtresa e kontrollon linkun dhe nëse është jo i infektuar nga sulmet me injektim SQL e dergon atë në serverin e aplikacionit, në të kundërt e parandalon.

Shtresa jonë nuk do të bllokojë të dhënat në hyrje që janë të rregullta sepse shtresa mbrojtëse konrollon në mënyrë të vazhdueshme të dhënat në hyrje me fjalet kyçe që ndihmojne sulmet me injektim SQL.

```
ResultSet request= executeQuery("select * from protective_ layer");
```

Më poshtë do të shikojmë interpretimin e linkut si një kërkesë e zakonshme që do të përfundojë në bazën e të dhënave dhe do të kontrollohet në formën e një inputi të zakonshëm.

```
while(request.next())
String x=requests.getString("input");
```

Nëse shtresa mbrojtëse gjatë kontrollit të linkut identifikon që instruksione të tjera SQL nuk i janë shtuar kërkesës, e cili në këtë rast vjen në formë linku dhe këto instruksione nuk përkojnë me fjalët kyçe që ndihmojnë sulmet me injektim SQL, ajo i kalon automatikisht sipas kodit të mëposhtëm.

```
if (query.compare(x) == 0)
doProtectiveLayer (request, res);
```

Në rastin e kundërt e raporton këtë kërkesë si jo të rregullt pra si mesazh të gabuar.

```
request1.sendRedirect("/gabimi.html");
```

Shtresa mbrojtëse është plotesisht e aftë të parandalojë sulmet me injektim SQL pa ndonjë infrastrukturë shtesë të nevojshme. Më poshtë do të shikojmë edhe efiçencën e saj e cila tregohet nëpërmjet testimeve me tre skanerat më të specializuar në bazë të studimeve për sulmet me injektim SQL.

5.2.2 Zbatimi pa Shtresën Mbrojtëse

Pa shtresën mbrojtëse: Në këtë rast websajti vepron pa ndonjë strategji në mbrojtjen e të dhënave, në parandalimin dhe identifikimin e sulmeve me injektim SQL. Të dhënat kalojnë thjesht nga ana e klientit në atë të bazës së të dhënave.

5.3 Përgatitja e Testimeve për të Treguar Efiçencën e Shtresës Tonë Mbrojtëse Kundër Sulmeve me Injektim SQL

Skanerat virtualë që testojnë sigurinë e një web aplikacioni janë ndërtuar ose dizenjuar për të testuar, mekanizmin e sigurisë të zbatuar në një web aplikacion, që në rastin tonë është shtresa jonë mbrojtese.

Këto skanera veprojnë duke zbuluar pikat kritike që vijnë si pasojë e kërkesave ose të dhënave në hyrje nga përdorues të ndryshëm, ku sulmuesi mund të injektojë një kod të gabuar SQL për të sulmuar bazën e të dhënave dhe rrjedhimisht informacionin që ndodhet në të.

Më pas këto skanera verifikojnë dhe tregojnë suksesin e sulmit dhe e raportojnë tek rezultatet e tyre, në rastin e studimit tonë neve na duhen rezultatet që skanerat do të nxjerin mbi sulmet me injektim SQL. Këto skanera janë të dizenjuara në mënyrë të tillë, që të performojnë të njëjtin sulm, si sulmi që do të behej në mënyrë manuale nga një sulmues i zakonshëm, prandaj ato sigurojnë një mjedis praktik dhe real për të testuar shtresën tonë mbrojtëse kundër sulmeve me injektim SQL.

Në këtë studim ne e kemi testuar shtresën tonë mbrojtëse, mbi bazën e tre skanerave më të përdorur në ditët e sotme, të specializuar për sulmet me injektim SQL, të cilët sigurisht nuk do ti përmëndim as me emra, as sipas versioneve të tyre, për shkak të barazisë por edhe sepse liçencat e tyre të përdorimit nuk e lejojnë publikimin e performancës apo rezultateve të tyre në raport me çdo mekanizem sigurie të përdorur në çdo skeme sigurie ose web aplikacion.

5.3.1 Skanerat S1, S2 dhe S3 që Ndihmuan në Testimin e Shtresës tonë Mbrojtëse

Në këtë mënyrë skanerat, që janë trajtuar më poshtë, dhe që kanë ndihmuar në testimin e shtresës tonë mbrojtëse ndaj sulmeve me injektim SQL, i kemi vendosur shkurtimet S1, S2, S3, duke i rradhitur ato në mënyrë të rastësishme si më poshtë. Duhet theksuar që këto skanera janë zgjedhur nga ne sepse janë të specializuar në lidhje me sulmet me injektim SQL.

- ✓ Skaneri i parë S1
- ✓ Skaneri i dytë S2
- ✓ Skaneri i tretë S3

Skaneri i parë S1, kontrollon në mënyrë automatike web aplikacionet dhe spikat më tepër për sulmet me injektim SQL dhe XSS. Ky skaner zotëron një teknologji që identifikon dobësite e një web aplikacioni që skanerat e tjerë të studiuar nga ne nuk munden. Ky skaner e ndan skanimin e sulmeve të llojeve të ndryshme, sipas ashpërsisë së sulmit, ai e ndan në katër tipe, si më poshtë:

- ✓ Shumë të vështirë
- ✓ Të vështirë
- ✓ Të mesëm
- ✓ Të ulët.

Sulmet me injektim SQL, bëjnë pjesë në sulmet me ashpërsi të lartë, dhe janë në kategorinë "shumë të vështirë" për t'u kapur, dhe pse ky skaner është i specializuar për identifikimin e këtyre sulmeve me injektim SQL. Sulmet e tjera që ky skaner mundet ti identifikojë janë të shpërndara në kategoritë e tjera. Një veçori tjetër e këtij skaneri siç do ta shikojmë dhe në rezultatet e mëposhtëme është, se është shumë i ngadaltë në skanimin e një skeme sigurie në raport me skanerat e tjerë.

Skaneri i dytë S2, përdoret për vlerësimin e sigurisë së web aplikacioneve dhe skemave mbrojtëse. Ky është një skaner i shkruar në Java dhe në pergjithësi zhvilluesit ose menaxhuesit e web aplikacioneve e përdorin këtë skaner për të zhvilluar sigurinë në websajtet ose në aplikacionet që kanë në websajtet e tyre. Ky skaner ashtu si edhe skaneri S1 është pa pagese dhe duke përdorur këtë skaner ne mund të modifikojmë të gjitha të dhënat HTTP dhe HTTPS midis klientit dhe serverit. Duke u nisur nga shkallët e çdo websajti, serveri skanohet nëpërmjet skanerit, i cili kontrollon nëse ka ndonjë konfiguracion të gabuar në server, kjo është një veçori e ketij skaneri, sepse shumë skanera të tjerë nuk janë në gjëndje ta bëjnë këtë.

Pra për ta bërë skanimin sa më funksional, ky skaner e kontrollon websajtin hap pas hapi sipas mënyrës së ndërtimit të websajtit. Dhe sigurisht ky skaner si edhe të tjerët në fund paraqet rezultatet mbi bazën e nivelit të sigurisë që çdo websajt apo web aplikacion ka përdorur.

Skaneri i tretë S3, është programuar gjithashtu në Java. Dhe ky skaner në mënyrë automatike skanon çdo web aplikacion dhe skemë sigurie të zbatuar në çdo websajt për identifikimin e sulmeve me injektim SQL dhe XSS. Ky skaner ka një perparësi nga skanerat e tjerë në lidhje me sulmet me injektim SQL, sepse në raportin perfundimtar i ndan sulmet me injektim SQL në tre kategori si më poshtë:

- ✓ Kërkesa eshtë e sulmuar pak
- ✓ Kërkesa është sulmuar në nivel mesatar
- ✓ Kërkesa është shumë e sulmuar

Ky skaner gjithashtu paraqet në fund një grafik për sulmet me injektim SQL.

5.3.2 Vlerësimi i Shtresës Mbrojtëse sipas Saktësisë, Gabimit Pozitiv dhe Gabimit Negativ

Për të realizuar vlerësimin e shtresës tonë mbrojtëse në lidhje me tre skanerat, ne jemi bazuar në disa elementë që do të na ndihmojnë të shikojmë performancën e këtyre skanerave me ose pa shtresën tonë mbrojtëse.

Këto elementë janë:

a) Saktësia
b) Gabimi pozitiv
c) Gabimi negativ

Saktësia : Saktësia ka të bëjë me korrektësine e rezultateve të gjeneruara nga skaneri. Nëse kemi x dobësi në një aplikacion ose skemë sigurie, të monitoruara nga secili skaner dhe gjënden y nga këto dobësi, të cilat për skanerat janë të rregullta, pra me saktësi do të kuptojmë nëse kërkesa do të jetë e sulmuar nga sulmet me injektim SQL apo jo.

$$1) \text{ Saktesia } (\%) = (Y/X) * 100$$

Saktësia gjithashtu është përcaktuar si numri i të gjitha dobësive që skanerat kanë identifikuar për mbi numrin e pergjithshëm të dobësive prezente gjatë testimit me secilin skaner.

Gabimi pozitiv: Me gabim pozitiv do të kuptojmë ato raste kur skanerat gjatë testimeve të tyre nxjerrin një numër të caktuar kërkesash të sulmuara, në rastin tonë nga sulmet me injektim SQL, por që në fakt këto kërkesa nuk janë të sulmuara nga këto sulme. Pra nëse gjatë testeve me secilin skaner ne kemi krijuar X kërkesa të mundshme, skanerat do të nxjerrin edhe Y kërkesa nga këto X1 kërkesa, si të sulmuara nga sulmet me injektim SQL, por që në fakt nuk janë të tilla.

Gabimi pozitv në përqindje gjëndet me formulën (2) si më poshtë:

$$2) \text{ Gabimi pozitiv } (\%) = (Y/X1) * 100$$

Gabimi negativ: Me gabim negativ do të kuptojmë ato raste kur skanerat gjatë testimeve të tyre nuk identifikojnë një numër të caktuar kërkesash të sulmuara, në rastin tonë nga sulmet me injektim SQL, por që në fakt këto kërkesa janë të sulmuara nga këto sulme. Pra nëse gjatë testeve ne kemi krijuar X kërkesa të mundshme, skanerat nuk do të identifikojnë edhe Y kërkesa nga këto X1 kërkesa, si te sulmuara nga sulmet me injektim SQL, por që në fakt janë të tilla, pra janë sulme me injektim SQL në rastin tonë.

Gabimi negativ në përqindje gjëndet me formulën (3) si më poshtë:

$$3) \text{ Gabimi negativ } (\%) = (Y/X1) * 100$$

Këto formula janë përdorur për të nxjerrë rezultatet e shtresës tonë mbrojtëse në raport me gabimin pozitiv dhe atë negativ.

5.4 Analiza dhe Konkluzionet për Kapitullin e Pestë

Shtresa mbrojtëse është plotesisht e aftë të parandalojë sulmet me injektim SQL pa ndonjë infrastrukturë shtesë të nevojshme. Vlen të theksohet që zhvilluesit i mjafton të fokusohet vetëm në shtresën mbrojtëse për të anashkaluar problemin e sulmeve me injektim SQL. Një ndër avantazhet e tjera që ka shtresa mbrojtëse është se ajo është e vetme dhe e mjaftueshme për disa

faqe nga të cilat vijnë kërkesa të ndryshme. Pra nuk është e nevojshme që zhvilluesi ta përsërisi shtresën për çdo faqe të websajtit.

Shtresa jonë mbrojtëse është lehtësisht e manovrueshme dhe tepër praktike, që do të thotë, ajo mund të përdoret me çdo web aplikacion dhe nuk varet nga serveri i aplikacionit. Shtresa jonë mbrojtëse do të analizoje cdo kërkesë duke qëndruar midis përdoruesve dhe serverit të aplikacionit, por ajo mund të veprojë edhe duke qënë pjese e aplikacionit. Efiçenca e shtresës u testua nga skanerat që testojnë sigurinë e një aplikacioni.

6. Testimet e Realizuara dhe Rezultatet

Në këtë pjesë të studimit, ne do të trajtojmë dhe do të paraqesim rezultatet e shtresës tonë mbrojtëse, të cilat kanë dalë nga testet e realizuara, me ndihmën e tre skanerave S1, S2 dhe S3, veçoritë e të cilëve i kemi trajtuar më lart.

Për të testuar efiçencën e shtresës tonë mbrojtëse ndaj sulmeve me injektim SQL, ne kemi realizuar katër teste për secilin skaner, të cilat janë të mjaftueshme për analizimin e rezultateve të shtresës tonë në lidhje me sulmet me injektim SQL që i ndodhin nje web aplikacioni.

Pas çdo testimi të suksesshëm, rezultatet e nxjerra nga çdo skaner do të tregojnë të gjitha sulmet qe i ndodhën web aplikacionit të ndara sipas llojeve të tyre. Ne do të fokusohemi vetëm tek rezultatet e nxjerra mbi sulmet me injektim SQL sepse studimi jonë ka të bëjë vetëm me parandalimin e këtyre sulmeve dhe jo të sulmeve të tjera.

6.1 Krahasimi i Performancës së tre Skanerave midis Tyre në Raport me Sulmet dhe me Sistemet e Menaxhimit të Bazave të të Dhënave

Përpara se të punojmë me këto skanera është shumë e rëndësishme të studiojmë karakteristikat e tyre, dhe pse jo, ti krahasojmë ato me njëri tjetrin.

Krahasimi në tabelën 6.1. është bërë duke pasur parasysh dy kritere:

- ✓ Karakteristikat krahasuese të skanerave në raport me sistemet e menaxhimit të bazave të të dhënave
- ✓ Tipet e sulmeve që secili skaner kontrollon.

Tabela 6.1. Karakteristikat midis skanerave në raport me sulmet dhe me sistemet e bazave të të dhënave.

Skanerat e Sigurisë	Sistemet e bazës së të dhënave që suporton secili skaner	Llojet e sulmeve që kontrollojnë
S1	Oracle, MS SQL Server, MySQL ,MS Access	SQLi, XSS
S2	Oracle, MS SQL Server, MySQL	SQLi,XSS
S3	Oracle, MS SQL Server, MySQL	SQLi,XSS

Për të parë efiçencën e shtresës tonë mbrojtëse ne e testuam atë në katër web aplikacione të ndryshme publike, të cilat janë paraqitur në tabelën 6.2.

Emri i këtyre web aplikacioneve dhe websajteve, ku këto aplikacione veprojnë, nuk është përmëndur, për arsye sigurie. Të gjitha këto web aplikacione janë testuar me dhe pa shtresën tonë mbrojtëse, nëpërmjet tre skanerave të përmëndur më lart, për të parë nëse shtresa mbrojtëse është efiçente apo jo.

Tabela 6.2. Aplikacionet e testuara

Tematika e websajtit	Publike apo lokale
Agjenci udhëtimi	Publike
Universitet privat	Publike
Kompani veglash ndërtimi	Publike
Restorant onlinë	Publike

6.2 Testet dhe Rezultatet e Sulmeve me Injektim SQL me Skanerin S1 pa Shtresën Mbrojtëse

Në këtë pjesë të studimit, ne do të përshkruajme dhe do të diskutojme për rezultatet e skanerave duke u nisur nga testet përkatëse. Për të testuar efiçencën e shtresës tonë mbrojtëse ndaj sulmeve me injektim SQL, ne kemi realizuar katër teste për secilin skaner, të cilat janë të mjaftueshme për analizimin e rezultateve të shtresës tonë në lidhje me sulmet me injektim SQL që i ndodhin një web aplikacioni.

Për çdo test të kryer në skanerin përkatës, ne do të paraqesim rezultatet e sulmeve me injektim SQL të evidentuara në raport me numrin e sulmeve me injektim SQL të zbatuara, gjithashtu do të rendisim për çdo test duke u nisur nga sa përmëndëm më lart, gabimin negativ dhe atë pozitiv. Këto testime do të realizohen njeherë me shtresën mbrojtëse dhe në rastin tjeter pa të.

Nga sa vëmë re në tabelën 6.3, skaneri S1 ka identifikuar 66 sulme me injektim SQL, nga 73 sulme me injektim SQL gjithsej të zbatuara, për të katërta testet pa zbatimin e shtresës mbrojtëse. Gjithashtu gabimi negativ korespondues është 7 sulme, si edhe gabimi pozitiv i identifikuar është vetëm 5, për të katërta testet, theksojmë pa shtresën mbrojtëse.

Tabela 6.3. Rezultatet e sulmeve me injektim SQL me skanerin S1 pa shtresën mbrojtëse

Testimet pa shtresën mbrojtëse me skanerin S1			
	Numri i sulmeve me injektim SQL të evidentuara/Numri i sulmeve të zbatuara	*Gabimi Negativ*	*Gabimi Pozitiv*
Test 1	23/26	3	1
Test 2	15/17	2	2
Test 3	18/20	2	1
Test 4	10/10	0	1
Numri Total	**66 (90.4%) të evidentuara nga 73 gjithsej**	**7 (9.5%)**	**5 (6.8%)**

Përqindja e përgjithshme e identifikimit është 90.4%, dhe ajo e gabimit negativ është 9.5%, dhe e gabimit pozitiv është 6.8%. Këto rezultate janë për të katërta testet pa shtresën mbrojtëse.

Figura 6.1, i paraqet rezultatet respektive të skanerit S1 pa shtresën mbrojtëse nëpërmjet grafikut.

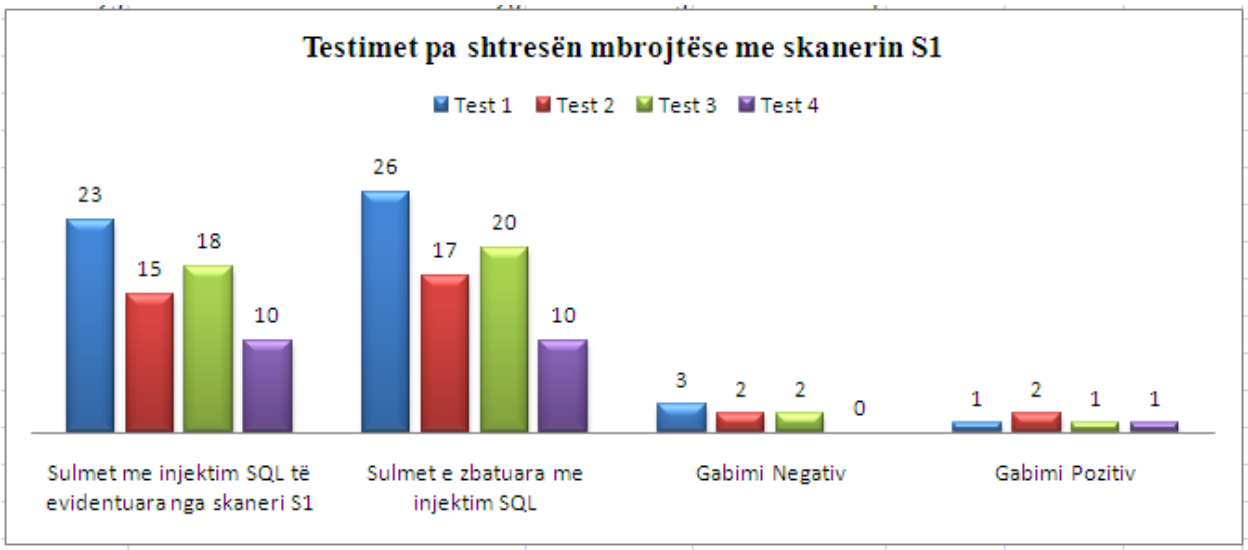

Figura 6.1. Grafiku i rezultateve të sulmeve me injektim SQL me skanerin S1 pa shtresën mbrojtëse

6.3 Testet dhe Rezultatet e Sulmeve me Injektim SQL me Skanerin S1 me Shtresën Mbrojtëse

Nga sa vëmë re në tabelën 6.4, skaneri S1 ka identifikuar 8 sulme me injektim SQL, nga 73 sulme me injektim SQL gjithsej të zbatuara, për të katërta testet me zbatimin e shtresës mbrojtëse. Gjithashtu gabimi negativ korespondues është 4 sulme, si edhe gabimi pozitiv i identifikuar është vetëm 2, për të katërta testet, theksojmë me shtresën mbrojtëse.

Tabela 6.4. Rezultatet e sulmeve me injektim SQL me skanerin S1 me shtresën mbrojtëse

Testimet me shtresën mbrojtëse me skanerin S1			
	Numri i sulmeve me injektim SQL të evidentuara/Numri i sulmeve të zbatuara	*Gabimi Negativ*	*Gabimi Pozitiv*
Test 1	3/26	2	1
Test 2	2/17	1	1
Test 3	2/20	1	0
Test 4	1/10	0	0
Numri Total	**8 (89 %) të evidentuara nga 73 gjithsej**	**4 (5.5%)**	**2 (2.7%)**

Përqindja e përgjithshme e identifikimit nga shtresa jonë mbrojtëse është 89%, gjithsej të identifikuar rreth 65 sulme me injektim SQL, gjithashtu gabimi negativ ka një rënie me zbatimin e shtresës mbrojtëse, ai është 5.5%, dhe gjithashtu gabimi pozitiv, i cili është 2.7%. Këto rezultate janë për të katërta testet me shtresën mbrojtëse, gjë e cila tregon efiçencen e saj.

Figura 6.2, i paraqet rezultatet respektive të skanerit S1 me shtresën mbrojtëse nëpërmjet grafikut.

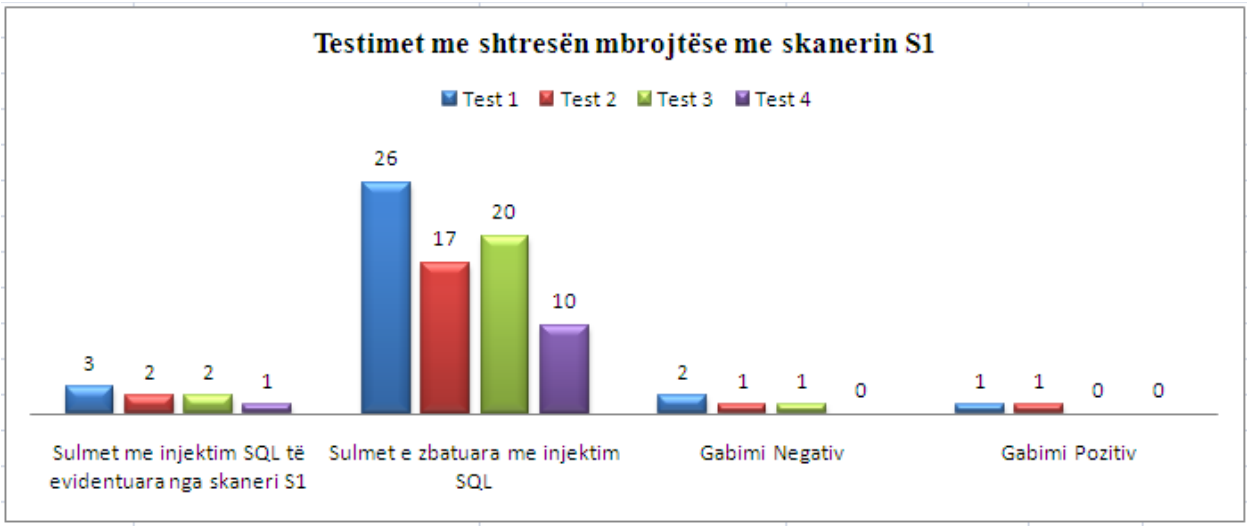

Figura 6.2. Grafiku i rezultateve të sulmeve me injektim SQL me skanerin S1 me shtresën mbrojtëse

6.4 Koha për Skanerin S1 me dhe pa Shtresën Mbrojtëse

Tabela 6.5, tregon kohën që i duhet skanerit të parë S1 për secilin test me shtresën mbrojtëse dhe pa të. Siç shikohet nga kjo tabelë, nuk kemi diferenca të medha në kohën që i duhet skanerit për të kontrolluar kërkesat për secilin test.

Tabela 6.5. Tabela e kohës për skanerin S1 me dhe pa shtresën mbrojtëse

	Skaneri S1		
	Numri i sulmeve me injektim SQL të zbatuara	*Koha pa shtresën mbrojtëse në sekonda*	*Koha me shtresën mbrojtëse në sekonda*
Test 1	26	350	385
Test 2	17	240	267
Test 3	20	295	311
Test 4	10	155	177

Figura 6.3, i paraqet rezultatet respektive të skanerit S1 me shtresën mbrojtëse dhe pa të nëpërmjet grafikut.

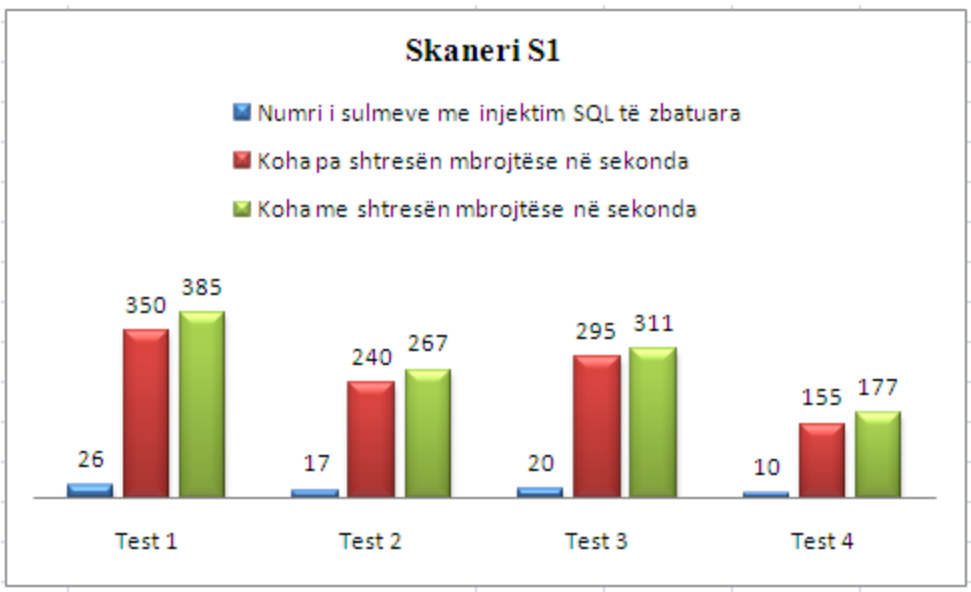

Figura 6.3. Paraqitja grafike e kohës për skanerin S1 me dhe pa shtresën mbrojtëse

6.5 Testet dhe Rezultatet e Sulmeve me Injektim SQL me Skanerin S2 pa Shtresën Mbrojtëse

Nga sa vëmë re në tabelën 6.6, skaneri S2 ka identifikuar 84 sulme me injektim SQL, nga 94 sulme me injektim SQL gjithsej të zbatuara, për të katërta testet pa zbatimin e shtresës mbrojtëse. Gjithashtu gabimi negativ korespondues është 10 sulme, si edhe gabimi pozitiv i identifikuar është vetëm 6, për të katërta testet, theksojmë pa shtresën mbrojtëse.

Tabela 6.6. Rezultatet e sulmeve me injektim SQL me skanerin S2 pa shtresën mbrojtëse

Testimet pa shtresën mbrojtëse me skanerin S2			
	Numri i sulmeve me injektim SQL të evidentuara/Numri i sulmeve të zbatuara	*Gabimi Negativ*	*Gabimi Pozitiv*
Test 1	28/32	4	2
Test 2	19/22	3	2
Test 3	23/25	2	1
Test 4	14/15	1	1
Numri Total	**84 (89.3%) të evidentuara nga 94 gjithsej**	**10 (10.6%)**	**6 (6.4%)**

Përqindja e përgjithshme e identifikimit është 89.3%, dhe ajo e gabimit negativ është 10.6%, dhe e gabimit pozitiv është 6.4%. Këto rezultate janë për të katërta testet pa shtresën mbrojtëse.

Figura 6.4, i paraqet rezultatet respektive të skanerit S2 pa shtresën mbrojtëse nëpërmjet grafikut.

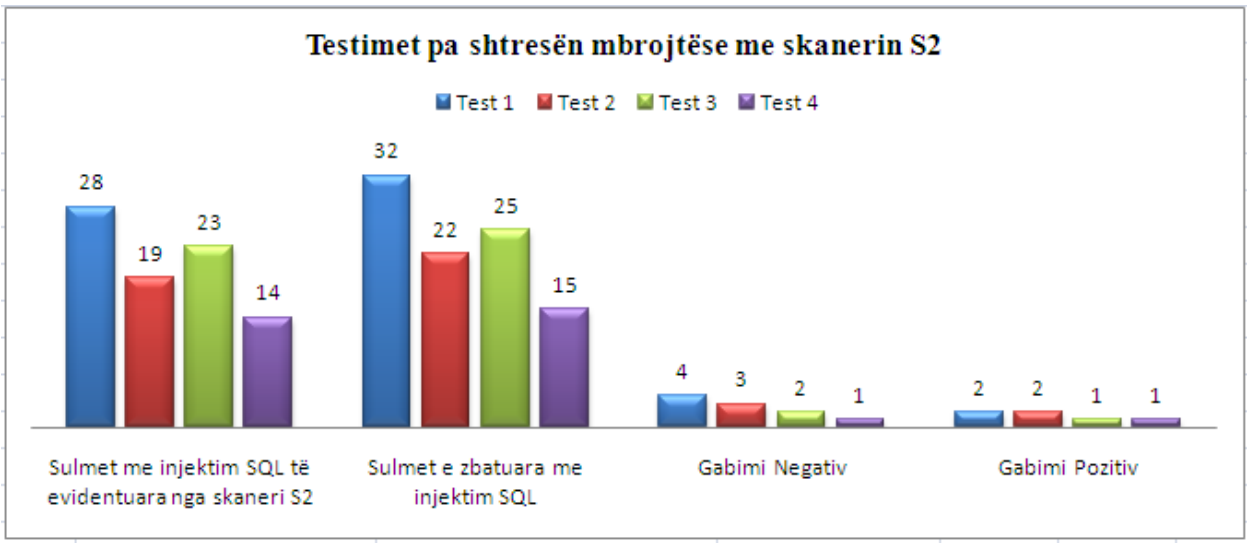

Figura 6.4. Grafiku i rezultateve të sulmeve me injektim SQL me skanerin S2 pa shtresën mbrojtëse

6.6 Testet dhe Rezultatet e Sulmeve me Injektim SQL me Skanerin S2 me Shtresën mbrojtëse

Nga sa vëmë re në tabelën 6.7, skaneri S2 ka identifikuar 9 sulme me injektim SQL, nga 94 sulme me injektim SQL gjithsej të zbatuara, për të katërta testet me zbatimin e shtresës mbrojtëse. Gjithashtu gabimi negativ korespondues është 6 sulme, si edhe gabimi pozitiv i identifikuar është vetëm 4, për të katërta testet, theksojmë pa shtresën mbrojtëse.

Tabela 6.7. Rezultatet e sulmeve me injektim SQL me skanerin S2 me shtresën mbrojtëse

Testimet me shtresën mbrojtëse me skanerin S2			
	Numri i sulmeve me injektim SQL të evidentuara/Numri i sulmeve të zbatuara	*Gabimi Negativ*	*Gabimi Pozitiv*
Test 1	3/32	2	1
Test 2	2/22	1	1
Test 3	2/25	1	1
Test 4	2/15	2	1
Numri Total	**9 (90.4 %) të evidentuara nga 94 gjithsej**	**6 (6.3%)**	**4 (4.2%)**

Përqindja e përgjithshme e identifikimit nga shtresa jonë mbrojtëse është 90.4%, gjithsej të identifikuar rreth 85 sulme me injektim SQL, gjithashtu gabimi negativ ka një rënie me zbatimin e shtresës mbrojtëse, ai është 6.3%, dhe gjithashtu gabimi pozitiv, i cili është 4.2%. Këto rezultate janë për të katërta testet me shtresën mbrojtëse, gjë e cila tregon efiçencen e saj.

Figura 6.5, i paraqet rezultatet respektive të skanerit S2 me shtresën mbrojtëse nëpërmjet grafikut.

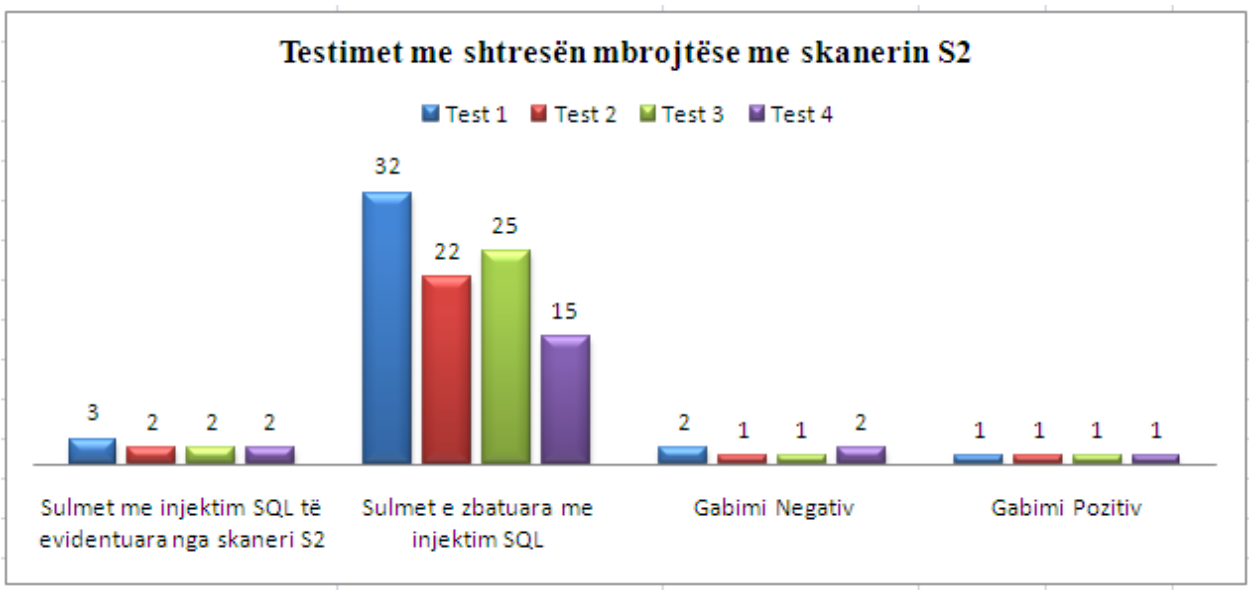

Figura 6.5. Grafiku i rezultateve të sulmeve me injektim SQL me skanerin S2 me shtresën mbrojtëse

6.7 Koha për Skanerin S2 me dhe pa Shtresën Mbrojtëse

Tabela 6.8, tregon kohën që i duhet skanerit të parë S2 për secilin test me shtresën mbrojtëse dhe pa të. Siç vihet re nga kjo tabelë, nuk kemi diferenca të mëdha në kohën që i duhet skanerit për të kontrolluar kërkesat për secilin test.

Tabela 6.8. Tabela e kohës për skanerin S2 me dhe pa shtresën mbrojtëse

	Skaneri S2		
	Numri i sulmeve me injektim SQL te zbatuara	*Koha pa shtresën mbrojtëse në sekonda*	*Koha me shtresën mbrojtëse në sekonda*
Test 1	32	410	428
Test 2	22	313	321
Test 3	25	336	359
Test 4	15	204	222

Figura 6.6, i paraqet rezultatet respektive të skanerit S2 me shtresën mbrojtëse dhe pa të nëpërmjet grafikut.

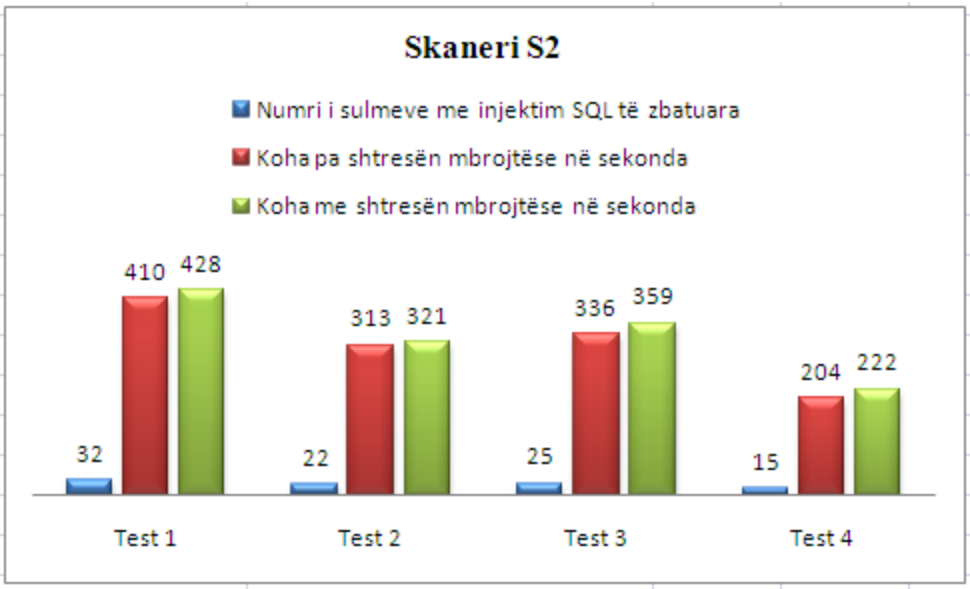

Figura 6.6. Paraqitja grafike e kohës për skanerin S2 me dhe pa shtresën mbrojtëse

6.8 Testet dhe Rezultatet e Sulmeve me Injektim SQL me Skanerin S3 pa Shtresën Mbrojtëse

Nga sa vëmë re në tabelën 6.9, skaneri S3 ka identifikuar 52 sulme me injektim SQL, nga 59 sulme me injektim SQL gjithsej të zbatuara, për të katërta testet pa zbatimin e shtresës mbrojtëse. Gjithashtu gabimi negativ korespondues është 6 sulme, si edhe gabimi pozitiv i identifikuar është vetëm 5, për të katërta testet, e theksojmë pa shtresën mbrojtëse.

Tabela 6.9. Rezultatet e sulmeve me injektim SQL me skanerin S3 pa shtresën mbrojtëse

Testimet pa shtresën mbrojtëse me skanerin S3			
	Numri i sulmeve me injektim SQL të evidentuara/Numri i sulmeve të zbatuara	*Gabimi Negativ*	*Gabimi Pozitiv*
Test 1	17/21	4	1
Test 2	12/13	1	2
Test 3	15/15	0	1
Test 4	9/10	1	1
Numri Total	**52 (88.2%) të evidentuara nga 59 gjithsej**	**6 (10.1%)**	**5 (8.4%)**

Përqindja e përgjithshme e identifikimit është 88.2%, dhe ajo e gabimit negativ është 10.1%, dhe e gabimit pozitiv është 8.4%. Këto rezultate janë për të katërta testet pa shtresën mbrojtëse.

Figura 6.7, i paraqet rezultatet respektive të skanerit S3 pa shtresën mbrojtëse nëpërmjet grafikut.

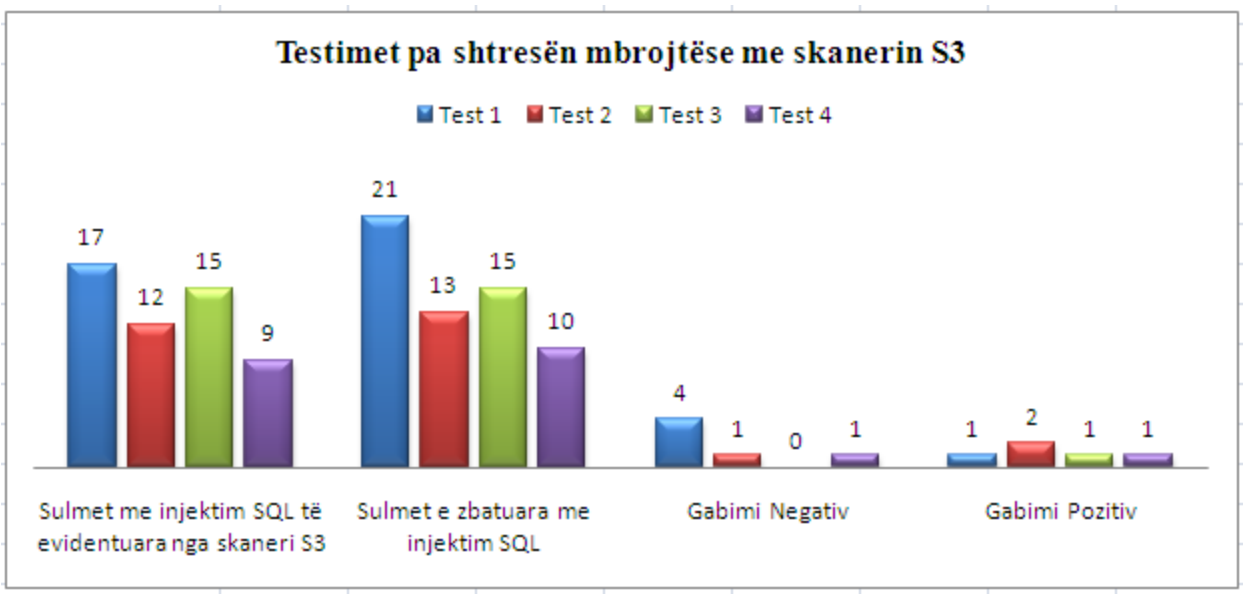

Figura 6.7. Grafiku i rezultateve të sulmeve me injektim SQL me skanerin S3 pa shtresën mbrojtëse

6.9 Testet dhe Rezultatet e Sulmeve me Injektim SQL me Skanerin S3 me Shtresën Mbrojtëse

Nga sa vëmë re në tabelën 6.10, skaneri S3 ka identifikuar 6 sulme me injektim SQL, nga 59 sulme me injektim SQL gjithsej të zbatuara, për të katërta testet me zbatimin e shtresës mbrojtëse. Gjithashtu gabimi negativ korespondues është 4 sulme, si edhe gabimi pozitiv i identifikuar është vetëm 3, për të katërta testet, e theksojmë pa shtresën mbrojtëse.

Tabela 6.10. Rezultatet e sulmeve me injektim SQL me skanerin S3 me shtresën mbrojtëse

Testimet me shtresën mbrojtëse me skanerin S3			
	Numri i sulmeve me injektim SQL të evidentuara/Numri i sulmeve të zbatuara	*Gabimi Negativ*	*Gabimi Pozitiv*
Test 1	1/21	2	1
Test 2	2/13	1	1
Test 3	2/15	1	0
Test 4	1/10	0	1
Numri Total	**6 (89.8 %) të evidentuara nga 59 gjithsej**	**4 (6.7%)**	**3 (5.1%)**

Përqindja e përgjithshme e identifikimit nga shtresa jonë mbrojtëse është 89.8%, gjithsej të identifikuar rreth 53 sulme me injektim SQL, gjithashtu gabimi negativ ka një rënie me zbatimin e shtresës tonë mbrojtëse, ai është 6.7%, dhe gjithashtu gabimi pozitiv, i cili është 5.1%. Këto rezultate janë për të katërta testet me shtresën mbrojtëse, gjë e cila tregon eficencen e saj.

Figura 6.8, i paraqet rezultatet respektive të skanerit S2 me shtresën mbrojtëse nëpërmjet grafikut.

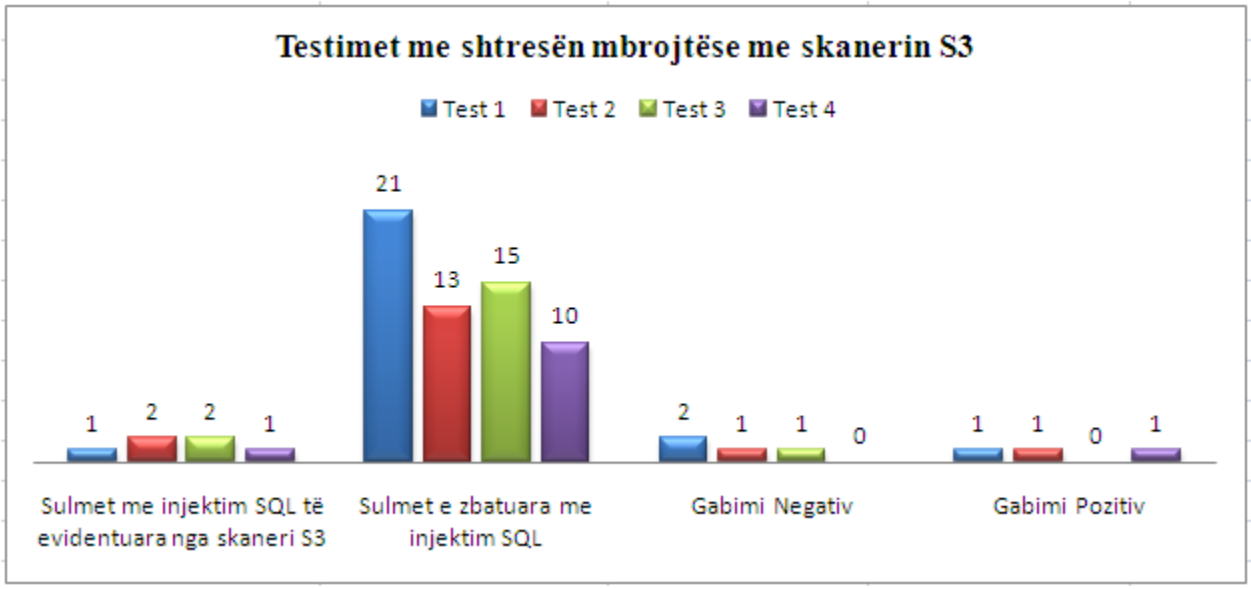

Figura 6.8. Grafiku i rezultateve të sulmeve me injektim SQL me skanerin S3 me shtresën mbrojtëse

6.10 Koha për Skanerin S3 me dhe pa Shtresën Mbrojtëse

Tabela 6.11, tregon kohën që i duhet skanerit të parë S3 për secilin test me shtresën mbrojtëse dhe pa të. Siç shikohet nga kjo tabelë, nuk kemi diferenca të mëdha në kohën që i duhet skanerit për të kontrolluar kërkesat për secilin test.

Tabela 6.11. Tabela e kohës për skanerin S3 me dhe pa shtresën mbrojtëse

	Skaneri S3		
	Numri i sulmeve me injektim SQL të zbatuara	*Koha pa shtresën mbrojtëse në sekonda*	*Koha me shtresën mbrojtëse në sekonda*
Test 1	21	1150	1290
Test 2	13	559	691
Test 3	15	772	834
Test 4	10	443	562

Figura 6.9, i paraqet rezultatet respektive të skanerit S2 me shtresën mbrojtëse dhe pa të nëpërmjet grafikut.

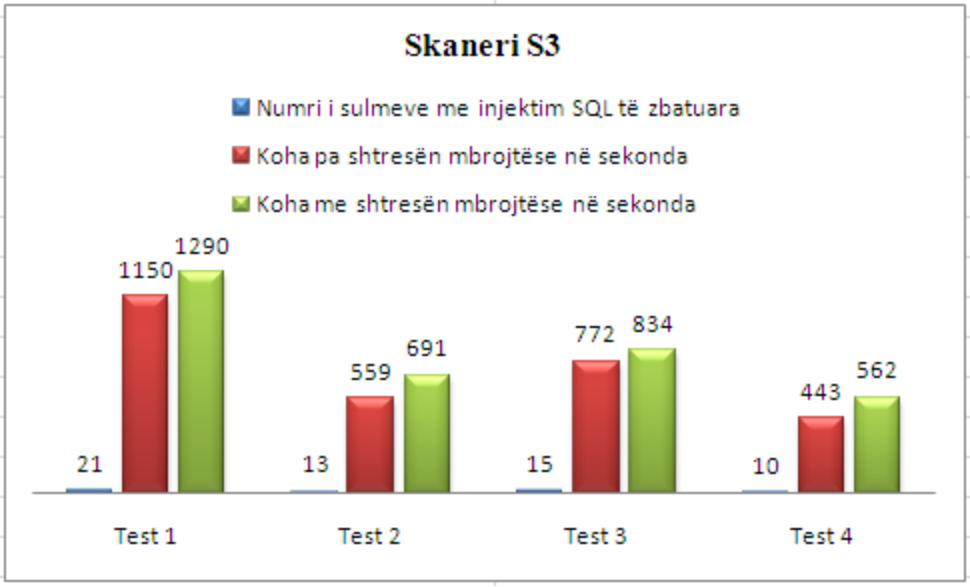

Figura 6.9. Paraqitja grafike e kohës për skanerin S3 me dhe pa shtresën mbrojtëse

6.11 Analiza e Testeve, Rezultateve dhe Konkluzionet për Kapitullin e Gjashtë

Nga ky kapitull rezulton që të tre skanerat, që janë përdorur për të kontrolluar efiçencën e shtresës tone mbrojtëse, janë skanera që janë të specializuar për të kontrolluar sulmet me injektim SQL dhe XSS (Figura 6.1), dhe kjo është mjaft e rëndësishme për të kuptuar efektin pozitiv që ka shtresa mbrojtëse në raport me këta skanera. Nga eksperimentet e zhvilluara me të tre skanerat me dhe pa shtresën tonë mbrojtëse, vihet re qartë rezultatet pozitive që dalin me zbatimin e shtresës tonë mbrojtëse mbi parandalimin e sulmeve me injektim SQL.

Përqindja e përgjithshme e identifikimit nga skaneri S1, pa zbatimin e shtresës tonë mbrojtëse është 90.4% (Tabela 6.3), pra do të thotë që pothuajse të gjitha sulmet me injektim SQL, kanë kaluar dhe janë identifikuar si sulme me injektim SQL nga skaneri S1. Përqindja e gabimit negativ është 9.5%, dhe ajo e gabimit pozitiv është 6.8%. Këto rezultate janë për të katërta testet pa shtresën mbrojtëse me skanerin S1.

Përqindja e përgjithshme e parandalimit të sulmeve me injektim SQL nga shtresa jonë mbrojtëse me skanerin S1 është 89% (Tabela 6.4), gjithsej janë të identifikuar janë rreth 65 sulme me injektim SQL, gjithashtu gabimi negativ ka një rënie me zbatimin e shtresës tonë mbrojtëse, ai është 5.5%, dhe gjithashtu gabimi pozitiv, i cili është 2.7%. Këto rezultate janë për të katërta testet me shtresën mbrojtëse me skanerin S1, gjë e cila tregon efiçencen e saj.

Perqindja e parandalimit nga shtresa jonë mbrojtëse me skanerin S1, është edhe më e madhe se 89%, në rast se ne do të përfshimë edhe gabimin pozitiv, e cila duhet ti shtohet 89%-it, për sa kohë sulmet që kanë kaluar në gabimin pozitiv janë kërkesa të sakta dhe jo të infektuara.

Ndërkohë koha në sekonda e skanerit S1 me dhe pa shtresën tonë mbrojtëse, siç tregohet edhe nga tabela 6.5 dhe nga figura 6.3, rezulton që nuk kemi diferenca të mëdha në kohën që i duhet skanerit S1 për të kontrolluar kërkesat për secilin nga të katërt testet me skanerin S1.

Përqindja e përgjithshme e identifikimit nga skaneri S2, pa zbatimin e shtresës tonë mbrojtëse është 89.3% (Tabela 6.6), pra do të thotë që pothuajse të gjitha sulmet me injektim SQL, kanë kaluar dhe janë identifikuar si sulme me injektim SQL nga skaneri S2. Perqindja e gabimit negativ është 10.6%, dhe ajo e gabimit pozitiv është 6.4%. Këto rezultate janë për të katërta testet pa shtresën mbrojtëse me skanerin S2.

Përqindja e përgjithshme e parandalimit te sulmeve me injektim SQL nga shtresa jonë mbrojtëse me skanerin S2, është 90.4% (Tabela 6.7), gjithsej të identifikuar janë rreth 85 sulme me injektim SQL, gjithashtu gabimi negativ ka një rënie me zbatimin e shtresës mbrojtëse, ai është 6.3%, dhe gjithashtu gabimi pozitiv, i cili është 4.2%. Këto rezultate janë për të katërta testet me shtresën mbrojtëse, gjë e cila tregon efiçencen e saj.

Përqindja e parandalimit nga shtresa jonë mbrojtëse me skanerin S2, është edhe më e madhe se 90.4%, në rast se ne do të përfshimë edhe gabimin pozitiv, e cila duhet ti shtohet 90.4%-it, për sa kohë sulmet që kanë kaluar në gabimin pozitiv janë kërkesa të sakta dhe jo të infektuara.

Ndërkohë koha në sekonda e skanerit S2 me dhe pa shtresën tonë mbrojtëse, siç tregohet edhe nga tabela 6.8 dhe nga figura 6.6, rezulton që nuk kemi diferenca të mëdha në kohën që i duhet skanerit S2 për të kontrolluar kërkesat për secilin nga të katërt testet me skanerin S2.

Përqindja e përgjithshme e identifikimit nga skaneri S3, pa zbatimin e shtreses tone mbrojtëse është 88.2% (Tabela 6.9), pra do të thotë që pothuajse të gjitha sulmet me injektim SQL, kanë kaluar dhe janë identifikuar si sulme me injektim SQL nga skaneri S3. Përqindja e gabimit negativ është 10.1%, dhe e gabimit pozitiv është 8.4%. Këto rezultate janë për të katërta testet pa shtresën mbrojtëse me skanerin S3.

Përqindja e përgjithshme parandalimit te sulmeve me injektim SQL nga shtresa jonë mbrojtëse me skanerin S3, është 89.8% (Tabela 6.10), gjithsej të identifikuar janë rreth 53 sulme me injektim SQL, gjithashtu gabimi negativ ka një rënie me zbatimin e shtresës tonë mbrojtëse, ai është 6.7%, dhe gjithashtu gabimi pozitiv, i cili është 5.1%. Këto rezultate janë për të katërta testet me shtresën mbrojtëse, gjë e cila tregon efiçencen e saj.

Përqindja e parandalimit nga shtresa jonë mbrojtëse me skanerin S3, është edhe më e madhe se 89.8%, në rast se ne do të përfshimë edhe gabimin pozitiv, e cila duhet ti shtohet 89.8%-it, për sa kohë sulmet që kanë kaluar në gabimin pozitiv janë kërkesa të sakta dhe jo të infektuara.

Ndërkohë koha në sekonda e skanerit S3 me dhe pa shtresën tonë mbrojtëse, siç tregohet edhe nga tabela 6.11 dhe nga figura 6.9, rezulton që nuk kemi diferenca të mëdha në kohën që i duhet skanerit S3 për të kontrolluar kërkesat për secilin nga të katërt testet.

Gjatë përdorimit të tre skanerave me shtresën tonë mbrojtëse, na rezulton që shtresa jonë mbrojtëse është efiçente në parandalimin e sulmeve me injektim SQL në masen mbi 94%, duke përfshire këtu edhe gabimet pozitive të secilit rast.

7. Konkluzionet

Në këtë studim u fokusuam në një fushë shumë specifike si ajo e sulmeve me injektim SQL. Duke u nisur nga projekti i sigurisë se web aplikacioneve, OWASP, sulmet me injektim SQL, janë një nga 10 sulmet me kritike që i ndodhin një web aplikacioni. Një ndër qëllimet kryesore të kësaj teme studimi doktorature është krijimi i një shtrese efiçente që mbron të dhënat nga sulmet me injektim SQL, duke u fokusuar në parandalimin e këtyre sulmeve, që i ndodhin kompanive ose organizatave që kanë të dhënat e tyre onlajnë, pra që i lidh puna me web aplikacionet të lidhura me bazën e të dhënave.

Ka shumë mënyra të cilat të lejojnë të kesh akses pa pasur të drejtë të hysh në një bazë të dhënash, nëpërmjet një websajti, dhe sulmet me injektim SQL janë mënyra më e shpeshtë nga sulmuesit për t'u futur në bazën e të dhënave në mënyrë ilegale. Në këtë studim u prezantua një shtresë efiçente që parandalon sulmet me injektim SQL, gjithashtu u realizua një studim i thelluar i sulmeve me injektim SQL, duke i trajtuar me shëmbuj secilin sulm me injektim SQL, duke u vënë herë pas here dhe në pozitat e një sulmuesi. Sulmet me injektim SQL, të cilat u trajtuan me shëmbuj në punën e këtij disertacioni ishin: Tautologji, kueri i ndërtuar në mënyrë jokorekte llogjikisht, kueri union, kueri piggy-backed, procedurat e ruajtura, inference, injektimi i verbër, sulmi ndaj kohës dhe kodimi alternativ.

Gjatë studimit për teknikat e ndryshme parandaluese dhe identifikuese SQL, në kapitullin e katërt, ne vumë re, si pasojë e studimit dhe kërkimit mbi këto teknika, që shumica e sulmeve me injektim SQL, nuk janë të izoluara, që do të thotë, ato janë akoma më të suksesshme nëse përdoren nga sulmuesit në kombinim, gjithmonë në varësi të qëllimit që ka sulmuesi. Dhe pse ideja e kombinimit të këtyre sulmeve nuk përmëndet në mënyrë specifike në literaturë, ajo nënkuptohet nga kërkuesit shkencorë nga studimi në terësi i artikujve, që ne kemi përdorur për studimin tonë.

Gjatë studimit mbi sulmet me injektim në Shqipëri ne kapitullin e tretë, ne dolëm në përfundimin se shumë kompani dhe organizata kanë dijeni, por nuk bëjnë asgjë për ta parandaluar atë, pra në këtë mjedis ku nuk i kushtohet vëmëndje aq sa duhet rrezikut nga sulmet me injektim SQL, propozimi i një shtrese mbrojtëse ndaj këtyre sulmeve është mjaft aktual. Nga studimi rezultoi gjithashtu që kompanitë me një numër të ulët punonjësish dhe pa një departament të Teknologjisë së Informacionit brënda tyre, nuk e dinin nëse kishte patur përpjekje të sulmeve me injektim SQL kundër websajteve të tyre krahasuar me kompanitë dhe organizatat më të mëdha të cilët kishin një sektor të Teknologjisë së Informacionit dhe Komunikimit të zhvilluar. Eshtë e mundur që kompanitë dhe organizatat publike ose private që kanë një numër të ulët punonjësish, nuk janë në gjëndje të caktojnë burime të mjaftueshme për sigurine në Teknologjinë e Informacionit dhe Komunikimit, e cila mund të jetë edhe arsyeja pse ato nuk kanë krijuar një departament të Teknologjise së Informacionit dhe Komunikimit, dhe që sulmohen vazhdimisht nga sulmet e jashtme pa qënë në dijeni që po sulmohen.

Gjithashtu gjatë zhvillimit të disertacionit ne kapitullin e katërt ne trajtuam gjerësisht një studim [41], [45] mbi teknikat e identifikimit dhe parandalimit të sulmeve me injektim SQL. Nga rezultatet e kerkimit evidentuam një numër të madh teknikash që identifikonin dhe parandalonin sulmet me injektim SQL. U trajtuan të metat dhe veçoritë e këtyre teknikave dhe gjithashtu i studiuam këto teknika identifikuese dhe parandaluese, në raport me sulmet me injektim SQL. Për ta kryer këtë studim, në fillim ne përcaktuam llojet e ndryshme të sulmeve me injektim SQL. Gjithashtu u investiguan teknikat e identifikimit dhe parandalimit të sulmeve me injektim SQL. Më pas, u krahasuan këto teknika, duke u bazuar në kriteret e zhvillimit dhe vlerësimit të tyre. Nga studimi (Tabela 4.46 dhe Tabela 4.48) rezulton që dy sulme me injektim SQL, të cilat janë "Procedurat e ruajtura" dhe "Kodimi alternativ" krijonin më tepër probleme për teknikat e identifikimit dhe parandalimit. Nga rezultatet më lartë, në kapitullin e katërt, duket qartë që pothuajse të gjitha sulmet me injektim SQL, janë adresuar në mënyrë të qëndrueshme nga teknikat e identifikimit dhe parandalimit të sulmeve me injektim SQL, me përjashtim të sulmeve që ndodhin në procedurat e ruajtura, të cilat nuk mund të ndalohen nga disa teknika.

Nga studimi në kapitullin e katërt (Tabela 4.46 dhe Tabela 4.48), është evidente që vetëm 29% e teknikave, mund të identifikojnë dhe parandalojnë plotësisht sulmin me injektim SQL, që ndodh në procedurat e ruajtura. Nga rezultatet e studimit në kapitullin e katërt, duket qartë që sulmi me injektim SQL, që ndodh në procedurat e ruajtura nuk identifikohet nga 57% e teknikave të identifikimit të sulmeve me injektim SQL.

Ndërkohë nga studimi në kapitullin e katërt (Tabela 4.46 dhe Tabela 4.48), duket qartë që sulmet e tjera me injektim SQL, janë të identifikueshme plotësisht dhe të parandalueshme pjesërisht nga afërsisht 60% e teknikave të identifikimit dhe parandalimit të sulmeve me injektim SQL.

Ishte shumë e vështirë për të bërë një krahasim të teknikave që parandalonin dhe identifikonin sulmet me injektim SQL, në raport me llojet e sulmeve me injektim SQL, sepse autorë të ndryshëm i kanë prezantuar punët e tyre në nivele të ndryshme, dhe duke i marrë këto të dhëna nga artikuj të ndryshëm, puna bëhej akoma më e veshtirë. Puna jonë në të ardhmen për identifikimin dhe parandalimin e sulmeve me injektim SQL do të jetë zgjerimi i njohurive për këto teknika përsa i përket kritereve të vlerësimit.

Përqindja e përgjithshme e parandalimit të sulmeve me injektim SQL nga shtresa jonë mbrojtëse me skanerin S1 është 89% (Tabela 6.4), gjithsej janë të identifikuar janë rreth 65 sulme me injektim SQL, gjithashtu gabimi negativ ka një rënie me zbatimin e shtresës tonë mbrojtëse, ai është 5.5%, dhe gjithashtu gabimi pozitiv, i cili është 2.7%. Këto rezultate janë për të katërta testet në skanerin S1 me shtresën mbrojtëse, gjë e cila tregon efiçencën e saj.

Përqindja e përgjithshme e parandalimit të sulmeve me injektim SQL nga shtresa jonë mbrojtëse me skanerin S2, është 90.4% (Tabela 6.7), gjithsej të identifikuar janë rreth 85 sulme me injektim SQL, gjithashtu gabimi negativ ka një rënie me zbatimin e shtresës mbrojtëse, ai është

6.3%, dhe gjithashtu gabimi pozitiv, i cili është 4.2%. Këto rezultate janë për të katërta testet në skanerin S2 me shtresën mbrojtëse, gjë e cila tregon efiçencen e saj.

Përqindja e përgjithshme parandalimit të sulmeve me injektim SQL nga shtresa jonë mbrojtëse me skanerin S3, është 89.8% (Tabela 6.10), gjithsej të identifikuar janë rreth 53 sulme me injektim SQL, gjithashtu gabimi negativ ka një rënie me zbatimin e shtresës tonë mbrojtëse, ai është 6.7%, dhe gjithashtu gabimi pozitiv, i cili është 5.1%. Këto rezultate janë për të katërta testet në skanerin S3 me shtresën mbrojtëse, gjë e cila tregon efiçencën e saj.

Gjatë përdorimit të tre skanerave me shtresën tonë mbrojtëse na rezulton që shtresa jonë mbrojtëse është efiçente në parandalimin e sulmeve me injektim SQL në masen mbi 95%, duke përfshirë këtu edhe gabimet pozitive të secilit rast.

Ndërkohë nga koha në sekonda me dhe pa shtresën tonë mbrojtëse e skanerit S1 (Tabela 6.5 dhe Figura 6.3), e skanerit S2 (Tabela 6.8 dhe Figura 6.6) dhe e skanerit S3 (Tabela 6.11 dhe Figura 6.9), rezulton që nuk kemi diferenca të mëdha në kohën që i duhet skanerave për të kontrolluar kërkesat për secilin nga të katërt testet e realizuara.

Shtresa mbrojtëse është lehtësisht e manovrueshme dhe nuk varet nga server i aplikacionit, dhe për më tepër nga shtresa e bazës së të dhënave. Efiçenca e shtresës u testua nga skanerat që testojnë sigurinë e një aplikacioni. Shtresa jonë mbrojtëse nuk krijon ndonjë ngarkesë të bazës së të dhënave apo vonesë në proçesimin e të dhënave.

7.1 Kontributet e Këtij Disertacioni

Ky studim kontribuon në fushën kërkimore të sigurisë në Teknologjinë e Informacionit dhe Komunikimit. U shfrytëzuan dhe u mblodhën të gjitha burimet, literatura e mundshme për të investiguar dhe për të kërkuar informacionin e nevojshëm për sulmet me injektim SQL, për rolin e tyre dhe rëndësine që ato kanë në mjedisin tonë. Mbi bazën e këtij kërkimi dhe analize të zhvilluar, u krijua një shtresë mbrojtëse efiçente për parandalimin e tyre.

Ky studim kontribuon për të njohur realitetit e sulmeve me injektim SQL në Shqipëri. Strategjia e këtij kërkimi që u përdorur për të marrë informacionin e nevojshëm për këtë studim është ajo e pyetësorit.

Gjithashtu ky studim kontribuon për kërkuesit shkencorë, që ata të njohin në mënyrë të detajuar teknikat e identifikimit dhe parandalimit të sulmeve me injektim SQL, sepse studimi ka nxjerrë në mënyrë të qartë të metat dhe veçoritë e këtyre teknikave në kapitullin e katërt dhe gjithashtu ka krahasuar teknikat identifikuese dhe parandaluese, në raport me sulmet me injektim SQL. Për ta kryer këtë studim, në fillim u përcaktuan llojet e ndryshme të sulmeve me injektim SQL dhe u investiguan teknikat e identifikimit dhe parandalimit të sulmeve me injektim SQL. Më pas, u krahasuan këto teknika, duke u bazuar në kriteret e zhvillimit dhe vlerësimit të tyre.

Kjo tezë studimore kontribuon gjithashtu si një pikë referimi për sulmet me injektim SQL, një thirrje për t'u zgjuar për zvilluesit e aplikacioneve, për krijuesit e bazave të të dhënave, por gjithashtu dhe për kërkuesit shkencorë në fushën e sigurisë, pedagogët, dhe sigurisht për studentët që dizenjojnë dhe programojnë bazat e të dhënave dhe websajtet.

Nga studimi ne kapitullin e dytë, ne arritëm të njohim në mënyrë të detajuar sulmet me injektim SQL dhe të kuptojmë rëndësinë e secilit sulm, rezultatet e studimit teorik në kapitullin e katërt treguan qarte që dy sulme me injektim SQL, të cilat janë "Procedurat e ruajtura" dhe "Kodimi alternativ" krijonin më tepër probleme për teknikat e identifikimit dhe parandalimit. Nga rezultatet më lartë, duket qartë që pothuajse të gjitha sulmet me injektim SQL, janë adresuar në mënyrë të qëndrueshme nga teknikat e identifikimit dhe parandalimit të sulmeve me injektim SQL, me përjashtim të sulmeve që ndodhin në procedurat e ruajtura, të cilat nuk mund të ndalohen nga disa teknika.

U krijua një shtresë mbrojtëse [42] kundër sulmeve me injektim SQL. Shtresa mbrojtëse është lehtësisht e manovrueshme dhe nuk varet nga serveri i aplikacionit, dhe për më tepër nga shtresa e bazës së të dhënave. Eksperimentet e zhvilluara tregojnë që shtresa mbrojtëse është mjaft efiçente. Gjatë përdorimit të tre skanerave me shtresën tonë mbrojtëse, na rezulton që shtresa mbrojtëse është tepër efiçente në parandalimin e sulmeve me injektim SQL në masën mbi 90%, duke përfshirë këtu edhe gabimet pozitive të secilit rast. Gjithashtu nga rezultatet e të tre skanerave me shtresën mbrojtëse del qartë që kemi një rënie të gabimit negativ dhe pozitiv, në krahasim me rastet kur shtresa mbrojtëse mungonte.

Ndërkohë nga koha në sekonda, me dhe pa shtresën tonë mbrojtëse e skanerit S1, e skanerit S2 dhe e skanerit S3, na rezulton që nuk kemi diferenca të mëdha në kohën që i duhet skanerave për të kontrolluar kërkesat për secilin nga të katërt testet e realizuara.

7.2 Puna në të Ardhmen

Parandalimi i sulmeve me injektim SQL është një pikë e nxehtë për shumë kërkues shkencorë, ne besojmë që kjo fushë ka nevojë akoma për t'u investiguar në të ardhmen, kryesisht për dy arsye:

- ✓ E para, sepse ne nuk jemi të qartë plotësisht nëse i kemi marrë në konsideratë të gjitha pikat që do të çonin në kufizimin maksimal të sulmeve me injektim SQL.

- ✓ E dyta, sepse sulmet me injektim SQL, zhvillohen në mënyrë shumë të shpejtë, dhe në këtë moment që unë po shkruaj konkluzionet një sulm tjetër me injektim SQL mund të jetë duke u zhvilluar, dhe bashkë me të, një teknikë që do ta luftojë atë.

Puna jonë në të ardhmen për identifikimin dhe parandalimin e sulmeve me injektim SQL do të jetë zgjerimi i modeleve të sulmeve me injektim SQL, në shtresën mbrojtëse dhe zgjerimi i njohurive për teknikat e identifkimit dhe parandalimit të sulmeve me injektim SQL, përsa i përket kritereve të vlerësimit.

Gjithashtu puna në të ardhmen do të perfshijë edhe zgjerimin e sulmeve që shtresa jonë do të parandalojë, siç janë sulmet XSS.

Referencat

[1] Revista EsecurityPlanet http://www.esecurityplanet.com/network-security/how-was-sql-injection-discovered.html , Nentor 2013.

[2] C. Gould, Z. Su, and P. Devanbu, "JDBC checker: A static analysis tool for SQL/JDBC applications," 2004, pp. 697-698.

[3] C. Gould, Z. Su, and P. Devanbu, "Static Checking of Dynamically Generated Queries in Database Applications", in Proceedings of the 26th International Conference on Software Engineering (ICSE 04), pages 645-654,2004..

[4] R. A. McClure and I. H. Krüger, "SQL DOM: compile time checking of dynamic SQL statements," 2005, pp. 88-96

[5] G. Wassermann, Z. Su, "An analysis framework for security in web applications," In: Proceedings of the FSE Workshop on Specification and Verification of Component-Based Systems, SAVCBS, pp. 70–78, 2004.

[6] G. Buehrer, B. Weide, and G. Sivilotti. Using Parse Tree Validation to Prevent SQL Injection Attacks. In International Workshop on Software Engineering and Middleware (SEM), 2005.

[7] W. Halfond and A. Orso, "Preventing SQL injection attacks using AMNESIA," presented at the Proceedings of the 28th international conference on Software engineering, Shanghai, China, 2006.

[8] I. Balasundaram and I. Ramaraj, "An Approach to Detect and Prevent SQL Injection Attacks in Database Using Web Service," International Journal of Computer Science and Network Security, vol. 11, pp. 197-205, 2011.

[9] Z. Su and G. Wassermann. "The Essence of Command Injection Attacks in Web Applications".In The 33rd Annual Symposium on Principles of Programming Languages (POPL 2006), Jan. 2006.

[10]M. Cova, D. Balzarotti. "Swaddler:An Approach for the Anomaly-based Detection of State Violations in Web Applications", Recent Advances in Intrusion Detection, Proceedings, volume: 4637 Pages: 63-86 ,2007.

[11] X. Fu, X. Lu, B. Peltsverger, S. Chen, K. Qian, and L. Tao. "A StaticAnalysis Framework for Detecting SQL Injection Vulnerabilities",COMPSAC 2007, pp.87-96, July 2007.

[12] F. Valeur, D. Mutz, and G. Vigna. "A Learning-Based Approach to the Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA), Vienna, Austria, July 2005.

[13]K. Kemalis, and T. Tzouramanis. "SQL-IDS: A Specification-based Approach for SQL Injection Detection", SAC, 2008, Brazil, ACM, pp.2153-2158.

[14]P. Bisht, P. Madhusudan, and V. N. Venkatakrishnan. "CANDID:Dynamic Candidate Evaluations for Automatic Prevention of SQL Injection Attacks". ACM Trans. Inf. Syst. Secur, 2010.

[15] I. Lee , S. Jeong, S. Yeoc, J. Moond, "A novel method for SQL injection attack detection based on removing SQL query attribute", Journal Of mathematical and computer modeling, Elsevier 2011.

[16]P. Grazie, "SQL Prevent thesis", University of British Columbia (UBC) Vancouver, Canada,2008.

[17] Mei Junjin, "An Approach for SQL Injection Vulnerability Detection," Proceedings. of the 6th Int. Conf. on Information Technology: New Generations, Las Vegas, Nevada, pp. 1411-1414, Apr. 2009.

[18] R. A. Baker, "Code Reviews Enhance Software Quality" In Proceedings. of the 9th Int. Conf. on Software Engineering (ICSE 97), pp. 570-571, Boston, MA, USA 1997.

[19] A. Roichman, E. Gudes, "DIWeDa - Detecting Intrusions in WebDatabases". In: Atluri, V. (ed.) DAS 2008. LNCS, vol. 5094, pp. 313–329. Springer, Heidelberg (2008).

[20]Shaukat Ali, Azhar Rauf, Huma Javed "SQLIPA: An authentication mechanism Against SQL Injection", Europian Journal of Scientific Research, ISSN 1450-216x, vol.38 No.4 (2009), pp 604-611.

[21]Tadeusz Pietraszek and Dhris Vanden Berghe, "Defending against Injection Attacks through Context-Sensitive String Evaluation", Proccedings of Recent Advances in Intrusion Detection (RAID 2005).

[22] Y. Huang, S. Huang, T. Lin, and C. Tsai. Sivilotti."Web Application Security Assessment by Fault Injection and Behavior Monitoring", In Proceedings of the 11th International Word Wide Web Conference, May 2003.

[23]McClure, and I.H. Kruger, "SQL DOM: compile time checking of dynamic SQL statements," Software Engineering, ICSE 2005, Proceedings. 27th International Conference on, pp. 88- 96, 2005.

[24] Y. Huang, F. Yu, C. Yang, C. H. Tsai, D. T. Lee, and S. Y. Ku. "Securing Web Application Code by Static Analysis and Runtime Protection", In Proceedings of the 12th International Word Wide Web Conference, May 2004.

[25] M. Martin, B. Livshits, and M. S. Lam "Finding Application Errors and Security Flaws Using PQL: A Program Query Language", ACM Notices, Volume 40, Issue:10 pages, 2005.

[26] D. Scott and R. Sharp, "Abstracting Application-level Web Security", in Proceedings of the 11th International Conference on the World Wide Web, pages 396–407, 2002.

[27] G. William, J. Halfond, A. Orso, "Using Positive Tainting and Syntax Aware Evaluation to Counter SQL Injection Attacks, 14th ACM SIGSOFT international symposium on Foundations of software engineering, 2006.

[28] Y. Kosuga, K. Kernel, M. Hanaoka, M. Hishiyama and Y. Takahama "Sania: Syntactic and Semantic Analysis for Automated Testing Against SQL Injection", in Proceedings of the Computer Security Applications Conference 2007, pp.107-117.

[29] Y. Shin "Improving the Identification of Actual Input Manipulation Vulnerabilities", in 14th ACM SIGSOFT Symposium on Fundation of Software Engineering ACM, 2006.

[30] R. Ezumalai, "Combinatorial Approach for Preventing SQL Injection Attacks", IEEE International Advance Computing Conference (IACC 2009), pp.1212-1217

[31] S. Thomas, L. Williams, "Using Automated Fix Generation to Source SQL Statements", Third International Workshop on Software Engineering for Secure Systems (SESS'07), pages 9-9, May 2007.

[32] M. Ruse, T. Sarkar and S. Basu, "Analysis and Detection of SQL Injection Vulnerabilities via Automatic Test Case Generation of Programs", 10th International Symposium on Applications and the Internet, pp 31-37, 2010.

[33] Y. Haixia, N. Zhihong, "A Databse Security Testing Scheme of Web Application", Proccedings of 4th International Conference on Computer Science & Education 2009 (ICCSE'09), 25-28 July 2009, 953-955.

[34] N. A. Lambert and K. Song Lin," Use of Query Tokenization to detect and prevent SQL Injection Attacks", IEEE, 2010.

[35] K. Wei, M. Muthuprasanna, S. Kothari, "Preventing SQL Injection Attacks in Stored Procedures". Proc of the 2006 Australian Software Engineering Conference (ASWEC'06).

[36] A. Srinivas, G. Narayan, S. Ram. "Random4: An Application Specific Randomized Encryption Algorithm to prevent SQL injection, in Trust, Security and Privacy in Computing and Communications (TrustCom)",2012 IEEE 11th International Conference, pp.no. 1327 – 133, 25-27 June 2012.

[37] R. Romil, R. Shailendra, "SQL injection attack Detection using SVM", in International Journal of Computer Applications, Volume 42– No.13, March 2012.

[38]E. Kajo, L. Kodra, E. Vrenozaj and B. Shehu "Protection of Web Application Using Aspect Oriented Programming and Performance Evaluation", 5th Balkan Conference in Informatics, 16-20 September 2012. Novi Sad, Serbia.

[39] V. Shanmughaneethi, C. Emilin Shyni and S.Swamynathan, "SBSQLID: Securing Web Applications with Service Based SQL Injection Detection" 2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies, 978-0-7695-3915-7/09, 2009 IEEE.

[40] Kai-Xiang Zhang, Chia-Jun Lin, Shih-Jen Chen, Yanling Hwang, Hao-Lun Huang, and Fu-Hau Hsu, "TransSQL: A Translation and Validation-based Solution for SQL-Injection Attacks", First International Conference on Robot, Vision and Signal Processing, IEEE, 2011.

[41] B. Shehu and A. Xhuvani "A Literature Review on SQL Injection: Vulnerabilities, Attacks and their Prevention and Detection Techniques", International Journal of Computer Science Issues, 2014.

[42] B. Shehu and A. Xhuvani "An Efficient Protective Layer Against SQL Injection Attacks", International Journal of Computer Science Issues, 2014.

[43] B. Shehu "Analyses of SQL Injection Attacks on Web Applications", 4th International Conference on "Information Systems and Technology Innovations: towards a digital Economy ", ISTI 2012, Tirana, Albania.

[44] B. Shehu "Depersonalization of Personal Data in Information Systems", 4th International Conference on "Information Systems and Technology Innovations: towards a digital Economy ", ISTI 2012, Tirana, Albania.

[45] B. Shehu, A. Xhuvani and Sh. Ahmetaj " Methods of Identifying and Preventing SQL Attacks ", International Journal of Computer Science Issues, 2012.

[46] M. Aranitasi, M. Ibro, and B. Shehu "Using PKI to Increase the Security of the Electronic Transactions" 8th Annual South-East European Doctoral Student Conference. September 2013. Greece.

[47]The Open Web Application Security Project,"OWASP TOP Project", https://www.owasp.org/SQL_Injection. 2013.

[48] B. Shehu, Sh. Ahmetaj, M. Aranitasi, and A. Xhuvani, "Protection of Personal Data in Information Systems", International Journal of Computer Science, Vol. 10, No. 2, July 2013, ISSN (Online): 1694-0784.

[49]S. W. Boyd and A. D. Keromytis. "SQLrand: Preventing SQL Injection Attacks", In Proceedings of the 2nd Applied Cryptography and Network Security Conference, pages 292–302, June 2004.

[50] S. W. Boyd and A. D. Keromytis. “SQLrand: Preventing SQL Injection Attacks”, In Proceedings of the 2nd Applied Cryptography and Network Security Conference, pages 292–302, June 2004.

[51] M. Flodstrom and O. Vikkolm, SQL-Injections: A wake-up call for developers, Uppsala University, Sweden 2013.

[52] F. Monticelli, Phd thesis on Prevent SQL attacks. University of British Columbia (UBC) Vancouver, Canada 2008.

[53] S. Thomas, L. Williams, T.Xie, “On Automated Prepared Statement Generation To Remove SQL Injection Vulnerabilities”, Information and Software Technology 51, pp.589-598,2009.

.

Lista e konferencave dhe revistave të botuara nga doktoranti.

Konferenca:

- ✓ **B. Shehu** "Analyses of SQL Injection Attacks on Web Applications", 4th International Conference on "Information Systems and Technology Innovations: towards a digital Economy ", ISTI 2013, Tirana, Albania.
- ✓ E. Mece, L. Kodra, E. Vrenozaj and **B. Shehu** "Protection of Web Application Using Aspect Oriented Programming and Performance Evaluation", 5th Balkan Conference in Informatics, 16-20 September 2012. Novi Sad, Serbia.
- ✓ **B. Shehu** "Depersonalization of Personal Data in Information Systems", 4th International Conference on "Information Systems and Technology Innovations: towards a digital Economy ", ISTI 2012, Tirana, Albania.
- ✓ M. Aranitasi, M. Ibro, and **B. Shehu** "Using PKI to Increase the Security of the Electronic Transactions" 8th Annual South-East European Doctoral Student Conference. September 2013. Greece.

Revista:

- ✓ **B. Shehu**, A. Xhuvani and Sh. Ahmetaj "Methods of Identifying and Preventing SQL Attacks ", International Journal of Computer Science Issues, Vol. 9, Issue 6, No 2, November 2012. ISSN (Print): 1694-0814 | ISSN (Online): 1694-0814. Impact Factor 0.24.
- ✓ **B. Shehu** and A. Xhuvani "A Literature Review on SQL Injection: Vulnerabilities, Attacks and their Prevention and Detection Techniques", International Journal of Computer Science Issues, Vol. 11, Issue 4, No 1, July 2014. ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784. Impact Factor 0.24.
- ✓ **B. Shehu** and A. Xhuvani "An Efficient Protective Layer Against SQL Injection Attacks", International Journal of Computer Science Issues, Vol. 11, Issue 4, No 1, July 2014. ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784. Impact Factor 0.24.
- ✓ **B. Shehu**, Sh. Ahmetaj, M. Aranitasi, and A. Xhuvani, "Protection of Personal Data in Information Systems", International Journal of Computer Science, Vol. 10, No. 2, July 2013, ISSN (Online): 1694-0784. Impact Factor 0.24.