

*Capstone Project*

*Faculty of Contemporary Sciences and Technologies (CST)*

*Study Programme: Computer Science*

*Titulli:*

**Application of data mining in public procurements**

Mentori: Studenti:

Assit.PhD.Visar Shehu Kreshnik Abdiju \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tetore,2018

*Kjo faqe eshte lene qellimishte boshe*

**Abstrakt**

Projektin qe e kam krijuar eshte nje Web CRUD Application e cila eshte krijuar duke perdorur Nodejs dhe framework-et e tij. Node js eshte nje platforme open-source shume e perdorur nga zhvilluesit e web-it , e cila mundesone qe te zhvillojme pjesen e serverit (back end) duke perdorur gjuhen me te famshme dhe me te perdorur javascript . Cdo web aplikacion ose aplikacionet e telefonit e kan te detyrueshem qe te perdorin pjesen e serverit (back end) , kompanite gjigjande ne pergjithesi perdorin platformen e famshe Nodejs , kjo ishte nje nder arsyet pse kam zgjedhur node js.

CRUD aplikacioni e ka kuptimin e Create(krijo) , Read(lexo) , Update(Ndrysho) dhe Delete(fshij) operacionet qe mund te perdoren nga perdoruesit .Ne aplikacionin perdoruesit mund te regjistrojne veten e tyre dhe te mund te logohen nepermjet dhenjen e username dhe password. Nje perdorues nuk mund te regjistrohet me nje emer (username) i cili eshte perdorur , pra username eshte unik . Perdoruesit gjithashtu kan mundesi qe te shkycen(log out) nga aplikacioni ne qoftese ato jan te loguar.

Perdoruesit anonim te cilat nuk jane te loguar jane ne gjendjen qe vetem te shohin pyetjet , pergjigje dhe komentet nga perdoruesit tjere ndersa perdoruesit te autorizuar(Authenticated) , pervec qe mund te i shikojne ata mund te krijojne , fshine , ndryshojne krijimin e tyre .

**Falenderime**

Falenderoj familjen time e cila ka qene me mua qe nga fillimi e deri ne funde te ketij rrugetimi duke e bere me te lehte cdo hap timin.

Falenderoje te gjithe shoqerine per bashkpunimin , perkrahjen dhe kujtime e mira gjate kohes se studimeve.

Nje falenderim te posacem kam per profesorin Visar Shehu , per ndihmen e pa kursyer qe me ka dhene gjate studimeve si dhe per ndihmen dhe gatishmerine qe me ofroi gjate gjithe punes sime drejt perfundimit te temes se diplomes . Profesori ishte gjithmone i gatshem per konsultime , keshilla dhe perkrahje.

Falenderoj gjithashtu gjithe profesoret dhe asistentet per perkrahjen e tyre te vazhdueshme .

Ju falenderoj te gjitheve.

Permbajtja

[Abstract II](#_Toc492381819)

[Acknowledgement III](#_Toc492381820)

[1 Hyrje 3](#_Toc492381822)

[1.1 Falenderim 3](#_Toc492381823)

[1.2 Data visualization 4](#_Toc492381824)

[2.3 Data-driven Journalism 5](#_Toc492381825)

[3 ProductionServer 10](#_Toc492381827)

[3.1 Krijimi i SSH celsave 10](#_Toc492381828)

[3.2 Krijimi i Digital Ocean droplet 12](#_Toc492381829)

[3.3 Lidhja me serverin 12](#_Toc492381829)

[3.4 Vendosja dhe rregullimi i Git 12](#_Toc492381829)

[3.5 Kalimi i projektit prej makines lokale ne server 12](#_Toc492381829)

[3.6 Vendosja e HTTPS protokollit 12](#_Toc492381829)

[3.7 Nginx 12](#_Toc492381829)

[3.8 PM2 12](#_Toc492381829)

4 [Siguria 13](#_Toc492381830)

[4.1 Cross Site Scripting 13](#_Toc492381831)

[4.2 SQL Injection 13](#_Toc492381832)

[4.3 Cross-Site Request Forgery 17](#_Toc492381833)

[4.4 Mbrojtja e SnipetApp 18](#_Toc492381834)

[4.4.1 Pse CSRF duhet të merret parasysh? 18](#_Toc492381834)

**Lista e Figurave**

[Figura 1. Vendosja dhe rregullimi i git ne pjesen e serverit 3](#_Toc492381838)

[Figure 2. Kalimi i file-ve nga makina lokale ne server 4](#_Toc492381839)

[Figure 3. Konfigurimi i HTTPS protokollit 6](#_Toc492381840)

[Figure 4. Inicializimi i serverit duke perdorur HTTPS protokollin 13](#_Toc492381841)

[Figure 5. Startimi i aplikacionit duke perdorur pm2 14](#_Toc492381842)

[Figure 6. Perdorimi i paketes helmet 15](#_Toc492381843)

[Figure 7. Konfigurimi i paketes csp 18](#_Toc492381844)

[Figure 8. Perdorimi i paketes xss protection 19](#_Toc492381845)

# 3. Production Server

Ky aplikacion perdor Ubuntu Server (MEAN Droplet ) prejt Digital Ocean dhe mund te e perdorim edhe online ne : https://104.248.136.214 . DigitalOcean ofron zhvilluesve (developers) shërbime cloud që ndihmojnë në vendosjen dhe shkallëzimin e aplikacioneve që te funksionojnë dhe te perdoren në të njëjtën kohë në disa kompjuterë . MEAN nenkupton perdorimin e disa programeve sic jane MongoDB , Express , AngularJS dhe Nodejs . Gjate krijimit te droplet-it tuaj juve mund te zgjedhni programet qe deshironi per projektin tuaj.

## 3.1 Krijimi i SSH celsave

Ne fillim per krijimin e serverit neve na duhet qe te krijojme SSH celsa(keys) . SSH eshte nje protokoll rrjeti kriptografik qe perdoret per menaxhuar rrjete , sisteme operative , konfigurime dhe per tu kycur sigurte neper sherbime te rrjetit . Neve na nevoitet per lidhjen me serverin tone dhe per arsye sigurie.

Me ndihmen e ssh celsave ne nuk kemi nevoje per fjalekalime gjate lidhjes me serverin . Krijimi i ssh celsave :

* Hapim git bash
* Shkruajme kodin :

ssh-keygen -t rsa -b 4096 -C "[your\_email@example.com](mailto:your_email@example.com)" japim emailin e github account-it sepse na nevoitet te i kalojme(push) file-t tek serveri.

* Pastaj kopjojme celsin private (private key) dhe e shtojme tek ssh keys ne github (per arsye se do i kalojm filet tek serveri) dhe gjithashtu e vendosim edhe tek Digital Ocean qe do te na nevoitet per krijimin e serverit.

**3.2 Krijimi i Digital Ocean droplet**

Gjate krijimit ne Digital Ocean ne zgjedhim vendin ku de ndodhet serveri , ram-in , cpu-n , ssh celsat dhe madhesine e ruajtjes pra ne pergjithesi te gjithe performancat e serverit . Lokacioni i serverit eshte me efikase nese ndodhet sa me afer perdoruesve .

**3.3 Lidhja me serverin**

Lidhja me serverin behet duke hapur git bash dhe shkruajme komanden :

$ ssh root@"[your.ipaddress]"

Pastaj installojme nodejs , mongoDB dhe npm duke perdorur komanden:

$ apt-get install nodejs , $ apt -get install npm , $ apt -get install mongodb

# 3.4 Vendosja dhe rregullimi i Git

Ne server kam krijuar nji folder me emrin repo , ne te cilen krijojme nje file e cila inicializon git repository qe do e perdorim per dergimin e file-ve ne server prej ne github. Ne file-in e krijuar neve e konfigurojme qe file-t qe do te pranohen do te transportohen ne folder-in app , ne te cilen do te ndhodhet filet dhe do perdoret per inicializimin dhe punimin e projektit. Kodi i cili do ti transportoj te dhenat tek folderi app eshte :

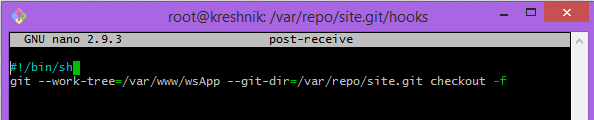


Figura 1. Vendosja dhe rregullimi i git ne pjesen e serverit

# 3.5 Kalimi i fileve prej makines lokale ne server

Hapim git bash dhe kalojme tek folderi(repository) ku kemi projektin tone , krijojme nje remote url qe te i dergojme filet tek serveri. Kodi per krijimin e nje remote url :

$ git remote add production ssh://'secret' @'yourip'/var/repo/site.git

Dergimi i fileve behet duke perdorur komanden :

$ git push production

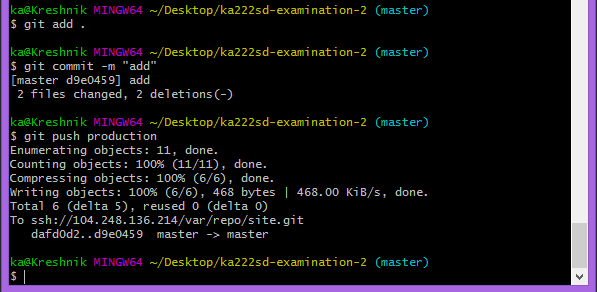


Figura 2. Kalimi i file-ve nga makina lokale ne server

Pas dergimit te projektit ne server , kalojme tek projekti ne server dhe bejme installimin e te gjithe librarive(dependencies) duke perdorur komanden :

> $ npm install --production

# 3.6 Vendosja e HTTPS protokollit

HTTPS (**Hypertext Transfer Protocol Secure)** eshte nje zgjerim i HTTP (**Hyper Transfer Protocol)** qe perdoret per nje kominikim me te sigurte ne rrjete dhe gjithashtu eshjte shum e perdorur . Ne HTTPS protokolli i komunikimit enkriptohet duke perdorur **Transport Layer Security**(TLS) ose **Secure Socket Layer** (SSL). Motivi kryesor i HTTPS eshte mbrojtja e privatsise dhe integritetit te te dhenave te shkembyera gjate tranzitit.Historikisht HTTPS lidhjet perdoreshin gjate pagesave te nje transaksioni , por ne ditet e sotme ajo perdoret me shpesh .

Kodi qe perdoret per gjenerimin e certifikatave të vetë-nënshkruara(self-signed certificates) :

Figura 3. Konfigurimi i HTTPS protokollit



Pas ekzekutimit te ketij kodi krijiohen cert.pem dhe key.pem . Cert.pem eshte certifikata qe kerkon klienti nga serveri kurse key.pem eshte celsi public (public key ) qe perdoret per startuar HTTPS komunikimin . Ne kodin qe e inicializojme serverin e shtojme edhe filet me lokacionet e tyre .

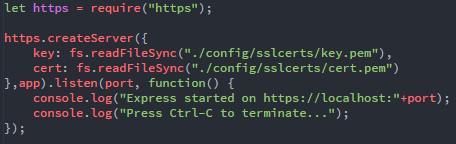


Figura 4. Inicializimi i serverit duke perdorur HTTPS protokollin

# 3.7 Nginx

Nginx eshte nje web server me performanca te larta , stabilitet , konfigurim te thjeshte e cila mund te perdoret nga te gjithe falas. Nginx zakonisht perdoret per shperndarjen e ngarkesave midis disa servereve , shfaqjen e permbajtjes se faqeve te ndryshme , trajtimin e enkriptimin dhe dekriptimin e HTTPs dhe per menaxhimin e kerkesave te dosjeve statike (reverse proxy).

Une Nginx e kam perdorur per te trajtuar enkriptimin dhe dekriptimin e HTTPS dhe per kerkesat e te gjithave dosjeve statike (reverse proxy) .

Ne /etc/nginx/sites-available ndodhet dosja per konfigurimin e nginx dhe kam konfiguruar per trajtimin e https dhe gjithashtu per menaxhimin e kerkesave te dosjeve statike (reverse proxy).

Konfigurimi i nginx :

server {

listen 443;

ssl on;

ssl\_certificate /var/www/wsApp/config/sslcerts/cert.pem;

ssl\_certificate\_key /var/www/wsApp/config/sslcerts/key.pem;

server\_name localhost;

gzip on;

gzip\_comp\_level 6;

gzip\_vary on;

gzip\_min\_length 1000;

gzip\_proxied any;

gzip\_types text/plain text/html text/css application/json application/x-javascript text/xml application/xml application/xml+rss text/javascript;

gzip\_buffers 16 8k;

location ~ ^/(images/|img/|javascript/|js/|css/|stylesheets/|flash/|media/|static/|robots.txt|humans.txt|favicon.ico) {

root /var/www/wsApp/public;

access\_log off;

expires max;

}

location / {

proxy\_pass https://localhost:3000;

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection 'upgrade';

proxy\_set\_header Host $host;

proxy\_cache\_bypass $http\_upgrade;

}

}

## 3.8 PM2 (Menaxhuesi i proceseve)

PM2(Process Manager) sherben per menaxhimin e proceseve te aplikacioneve qe perdorin Node js . PM2 mundeson qe aplikacionet te jen pergjithmone duke u ekzekutuar edhe ne qoftese ndodhe ndonje problem . PM2 mundeson operacione per startimin , ndalimin , rifillimin e aplikacionit por gjjithashtu mundeson eshe shume operacione te ndryshme per menaxhimin e te gjithe proceseve.

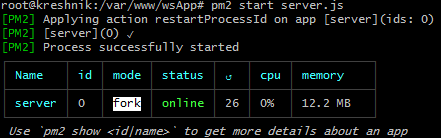


Figura 5. Startimi i aplikacionit duke perdorur pm2

# 4. Siguria

Aplikacionet web jane shpese pjesa me e cenueshme e nje sistemi informacioni . Ne te njejten kohe , per shkak te ekspozimit te tyre ne internet , ata jane zakonisht objektivi i pare ne nje sulm.

Gjithmone e me teper kompanite po kryejne biznesin e tyre ne internet , arsye per te cilen aplikacionet web jane nder asetet e tyre kritike . Per shkak te kompleksitetit ne rritje , keto aplikacione kerkojne me shume perpjekje , dhe testime sigurie profesionale dhe te dedikuar. (https://www.infigo.hr/al/testimet-e-sigurise-ne-aplikacionet-web-s78?t=1)

Është e qartë se çdo ueb aplikacion që mbledhë informata nga shfrytëzuesit është i prekshëm ndaj sulmeve, prandaj siguria është një problem kyç që duhet të merret në konsideratë. Gjatë realizimit të një aplikacioni ne përdorim resurse të ndryshme (protokolle të komunikimit, librari të ndryshme) të cilat na ndihmojnë në shumë aspekte mirëpo mungesa e njohurisë për përdorim të mirëfilltë të këtyre resurseve mund të na shkaktoj probleme të ndryshme sikur në aspektin e efikasitetit po ashtu dhe të sigurisë.

Metodat me te perdorura per te hakuar jane :

* SQL injection
* Cross Site Scripting
* Cross Site Request Forgery
* File inclusion
* Command execution
* Code injection

# 4.1 Cross Site Scripting(XSS)

**Cross Site Scripting (XSS)** eshte menyra me e perdorur nga sulmuesit per vjedhjen e te dhenave, kjo metode perdoret ne pjesen e klientit (client-side) duke injektuar skripta te cilat ekzekutohen tek perdoruesit tjere . Kjo u mundeson qe te vjedhin cookies, session tokens , ose infromacione te tjera te ndjeshme te ruajtura nga browser-i.

Nese vjedhin cookies u mundesohet qe te perdorin web faqe te ndryshe dhe te pretendojn se jane dikush tjeter gjithashtu mund te ju dergojne me ndonje web faqe tjeter dhe te marrin te dhenat bashke me perdoruesin.

Cookies jane te dhena te vogla qe ruhen ne kompjuterin e perdoruesit ose me sakte ne browser-in nderkohe qe perdoruesi shfleton .

**Rreziqet më të përhapura nga XSS :**

* Sulmuesi mund të rrezikoj ose të marrë përsipër llogarinë e viktimës në ueb faqe.
* Marrja e të dhënave nga faqja e targetuar për tu vizituar.
* Modifikimi i përmbajtjes së faqes.
* Ridërgimi i viktimës në faqe të tjera të qëllimit të keq.

# 4.2 SQL Injection

**SQL Injection (SQLi)** është një dobësi e sigurisë së aplikacionit që lejon sulmuesit të kontrollojnë bazën e të dhënave të një aplikacioni, duke u lejuar atyre të insertojnë ose fshijnë të dhëna, të ndryshojnë sjelljet e tyre nga të dhënat e një aplikacioni dhe të bëjnë gjëra të tjera të padëshiruara, duke mashtruar aplikacionin në dërgimin e komandave të papritura SQL. Dobësitë e injektimit SQL ndodhin kur një aplikacion përdor të dhëna të pabesuara, të tilla si të dhënat e futura në fushat e formës së internetit, si pjesë e një query-i të bazës së të dhënave. Kur një aplikacion nuk arrin t’i filtroj si duhet këto të dhëna të pabesuara para se t’i shtoj ato në pyetësorët SQL, një sulmues mund të përfshijë komandat e tij SQL të cilat do të ekzekutohen në databazë. Dobësitë e tilla SQLi janë të lehta për tu parandaluar, megjithatë SQLi mbetet një rrezik kryesor për aplikacionin në ueb dhe shumë organizata mbeten të ndjeshme ndaj shkeljeve të mundshme të të dhënave që rezultojnë nga injektimi SQL.

# 4.3 Cross-Site Request Forgery (CSPR)

**Cross-Site Request Forgery** aktualisht renditet ne vendin e 8 në tabelën e OWASP të Top 10 Dobësive dhe është një tip shumë i shfrytëzueshëm i cenueshmërisë. Cross-Site Request Forgery është një dobësi e aplikacionit në internet që e bën të mundur që një sulmues të detyrojë një përdorues që të kryejë veprime pa vetëdije kur ata janë të kyçur në një aplikacion. Sulmuesit zakonisht përdorin sulme CSRF kur kanë target faqet si cloud storage, social media, banking dhe online shopping, si pasojë e informatave dhe veprimeve të përdoruesve në dispozicion për këto aplikacione. Përafërsisht 23% e të gjitha aplikacioneve të testuara janë të prekshme ndaj Cross-site Request Forgery.

# 4.4 Mbrojtja e SnippetApp

Node js eshte nje platforme qe mundeson librari te ndryshme per mbrojtjen e aplikacionit. Gjate ekzekutimit te SnippetApp pjesa me e madhe e skriptave ekzekutohet ne pjesen e serverit (mundesin qe ta jep Express.js) dhe kjo mundeson mshefjen e kodit nga perdoruesit.

Pjesa me e madhe e informacioneve senzitive sic jane , Api Token ,Secret key , database URL jane te ruajtura ne variabla mjedisore(Environmental variables) dhe nuk lejon asnje skripte per vjedhjen e ketyre informatave . Per sigurine te projektit kam perdorur disa npm paketa : helmet , xssFilter , csrf dhe csp .

**Helmet** mundeson sigurine e aplikacionit tone duke vendosur HTTP headers qe te i ruaje informatat te cilat mund te perdoren nga sulmuesit.

Screenshot_7.png

Figura 6. Perdorimi i paketes helmet

**CSP(Content Security Policy)** eshte nje mbrojtje mjaft premtuese kunder rrezikut dhe ndikimit te sulmeve XSS(Script Injection). Qellimi kryesor eshte te parandaloje ndonje gje te padeshiruar qe te injektohet ne faqen tone. Neve mund te i caktojme se cilat skripta,style(css filet), foto dhe url dhe shume gjera tjera do mund qe te ekzekutoje.

Konfigurimi i csp :



Figura 7. Konfigurimi i paketes csp

**xssFilter** enkodon dhe shmang cdo karakter te demshem si "<" dhe ">" . Ne kete menyre ne mund te sigurohemi qe asnje kod i demshem (skript) te mos ekzekutohet . Kjo ben ndalesen e gjithave kodeve qe perdoren nepermjet dhenjeve te te dhenave per te sulmuar dhe te fshire ose te modifikojn databazen , keto sulme njihen edhe si SQL Injection.

Screenshot_8.png

Figura 8. Perdorimi i paketes xss protection

**CSRF** perdoret per te krijuar nje varg(token) sekrete qe nevoiten per realizimin e ndonje kerkese , pra krijon nje token unik per te komunikuar perdoruesi me serverin , nese kjo nuk eshte valide atehere kerkesa refuzohet . Pra kjo mundeson ruajtjen nga Cross-Site Request Forgery.

**4.4.1 Pse CSRF duhet të merret parasysh?**

Varësisht nga veprimet që po kryhen, një dobësi e CSRF mund të ketë pasoja serioze për përdoruesin e aplikacionit. Përdoruesit zakonisht nuk janë të vetëdijshëm se po kryhen veprime me qëllim të keq. Kjo do të thotë që mund të bëhen ndryshime në emër të përdoruesit pa dijeninë apo miratimin e tyre. Rreziqet e aplikacioneve të CSRF variojnë nga postimet e turpshme në rrjete sociale deri në humbjen e parave nga llogaritë tuaja online.

**4.4.2 Online banking CSRF – Si arrihet?**

Një sulm CSRF ne xhirollogarinë tuaj bankare fillon si në vijim. Përderisa ju jeni kyçur në aplikacionin e bankës tuaj, ju në të njëjtën kohë vizitoni një faqe e cila përmban një sulm CSRF. Në këtë faqe ju klikoni një buton të çfarëdoshëm në cilin është vendosur një kurthë. Ai buton në prapavijë bënë kërkesën në aplikacionin e bankës për të transferuar para nga një xhirollogari në tjetrën. Meqenëse ju keni qenë i kyçur në aplikacionin e bankës, ky sulm do jetë i suksesshëm. Sulmuesi tashmë ka paratë e përdoruesit edhe asnjë shenjë që diçka e çuditshme të ketë ndodhur.

**Si ndodhi kjo?**

Dy arsye kryesore pse ky sulm ishte i suksesshëm janë:

1) Aplikacioni i bankës lejon të gjenerohen kërkesa nga serverë të tjerë dhe nuk kufizohet vetëm në kërkesat e gjeneruara nga vetja.

2) Nuk ekziston një token unik që lidhet me sesionin e përdoruesit