Sigurnosno očvršćivanje GNU/Linux poslužitelja

Matić, Antonio

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Antonio Matić

Sigurnosno očvršćivanje GNU/Linux poslužitelja

ZAVRŠNI RAD

Varaždin, 2022

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Antonio Matić

Matični broj: 0016143409

Studij: Informacijski sustavi

Sigurnosno očvršćivanje GNU/Linux poslužitelja

ZAVRŠNI RAD

Mentor:

Izv. prof. dr. sc. Ivan Magdalenić

Varaždin, rujan 2022.

Antonio Matić

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sažetak

U ovome će radu biti obrađena tema sigurnosnog očvršćivanja GNU/Linux poslužitelja u različitim okolnostima. Pojam sigurnosti općenito, a posebno informacijske sigurnosti s obzirom na sve brži razvoj tehnologije postaje sve popularniji i shvaćanje osnova informacijske sigurnosti postaje sve važnije zbog raznolikosti kibernetičkih napada i njihovih posljedica na pojedinca i društvo. Zbog toga je ovdje iznesena osnovna teorija o informacijskoj sigurnosti, njenim principima, općenitim načinima obrane od kibernetičkih napada te pojam smjernica odnosno uputa za sigurnosno očvršćivanje koje prikazuju osnovu procesa sigurnosnog očvršćivanja. Spomenuti pojmovi u teorijskom dijelu detaljnije su razrađeni u praktičnom dijelu u kojemu su osim tih objašnjenja prikazani i načine ostvarivanja nekih od metoda sigurnosnog očvršćivanja zajedno sa snimkama zaslona koje prikazuju izravan utjecaj i ponašanje primijenjenih postavki na poslužitelju, odnosno njihov rezultat. Tim primjerima nastoji se potaknuti svijest o informacijskoj sigurnosti kao takvoj, posljedicama ne upravljanja sigurnošću, pravilnoj provedbi procesa sigurnosnog očvršćivanja te prikazati praktični primjeri obrađenih metoda kojima se postižu glavna tri principa informacijske sigurnosti: povjerljivost, integritet te dostupnost podataka i sustava.

Principi informacijske sigurnosti postižu se različitim metodama i tehnikama koje se mogu svrstati u nekoliko kategorija koje predstavljaju njihove ciljeve kako slijedi:

1. Upravljanje pristupom (eng. *access management*)
2. Upravljanje mrežnim prometom (eng. *network management*)
3. Zakrpa ranjivosti (eng. *vulnerability patching*)
4. Uklanjanje nepotrebnog softvera (eng. *removing unneeded software*)
5. Nadgledanje sustava (eng. *system monitoring*)
6. Osiguravanje komunikacija (eng. *securing communications*)
7. Provedba sigurnosnih pohrana (eng. *security backups*)
8. Očvršćivanje udaljenih sesija (eng. *remote session hardening*)

Ključne riječi: Informacijska sigurnost; Sigurnosno očvršćivanje; Linux; Baze podataka; Web poslužitelj; Vatroštit;

Sadržaj

1. Uvod

Budući da se posljednjih godina svijet susreće sa sve bržim razvojem tehnologije, sve što može digitalizira se odnosno obavlja putem digitalnih uređaja, a sustavi postaju sve umreženiji.

Sve usluge koje korisnici svakodnevno konzumiraju na Internetu poput različitih web trgovina, državnih web usluga poput e-Građani u Republici Hrvatskoj, društvene mreže, računalne igrice s mogućnosti umrežavanja s drugim igračima i sl. zahtijevaju infrastrukturu koja ih pokreće, drugim riječima, poslužuje uslugu krajnjim korisnicima. Upravo tome služe poslužitelji. Linux je kao operacijski sustav jako konfigurabilan i daje visoku razinu fleksibilnosti u konfiguraciji samog operacijskog sustava, pa čak i jezgre. Osim toga, u poslužiteljskim verzijama bez grafičkog sučelja većinom zahtijevi resursa za rad vrlo mali što čini održavanje poslužitelja jeftinijim i povoljnijim, a njegov rad bržim i stabilnijim. Zbog toga Linux operacijski sustavi prevladavaju u području poslužitelja u odnosu na Windows Server operacijske sustave. Zato se ovaj rad bavi sigurnosnim očvršćivanjem GNU/Linux poslužitelja.

Operacijski je sustav kao temeljna komponenta svih ostalih servisa i poslužitelja koji su instalirani na njega vrlo važan. Kao i svaki drugi sustav, on ima određene propuste, odnosno ranjivosti koje bi mogle biti zloupotrijebljene. Zato je važno provesti ne samo sigurnosno očvršćivanje mreže, instaliranih poslužitelja baza podataka i sl., nego i sigurnosno očvršćivanje samog operacijskog sustava. Moguće je provesti takav proces i nad jezgrom Linux sustava, međutim to u ovome radu neće biti obrađeno.

Svaki sustav zahtijeva neku vrstu upravljanja kako bi uvijek funkcionirao što je bolje moguće. Zbog prirode današnje tehnologije, često su poslužitelji na udaljenim lokacijama pa administratori sustava njima upravljaju iz različitih udaljenih lokacija. Takvo upravljanje mora biti sigurno i enkriptirano jer u suprotnom otkriva mnoge podatke i omogućuje relativno lak neautoriziran pristup poslužitelju i njegovim resursima. Potrebno je osigurati takve sesije, pa se u radu obrađuje i očvršćivanje protokola za udaljeno upravljanje sustavom spominjući također prednost i obvezu korištenja šifriranih protokola za takve aktivnosti.

Općim umrežavanjem sustava pregršt povjerljivih i osjetljivih podataka čuva se na raznim poslužiteljima i raznim mjestima na Zemlji, a još uvijek su dostupni bilo gdje, sve dok su uređaji povezani na Internet. Takvi su podaci *nafta* današnjice, odnosno vrijede jako puno. Upravo je zbog toga danas teško posjetiti neku web stranicu bez obavijesti o prikupljanju korisničkih podataka u razne svrhe. Međutim, tako osjetljivi podaci ne smiju biti nezaštićeni. Spremišta podataka, u informatici poznatija kao baze podataka trebaju biti što sigurniji od

različitih napada i krađa tih podataka koje na kraju mogu dovesti do različitih ilegalnih aktivnosti poput krađe identiteta i drugih zlonamjernih radnji. Zbog toga se u ovome radu obrađuju neke od metoda sigurnosnog očvršćivanja poslužitelja baza podataka, s primjerima na poslužiteljima Postgresql i MySQL.

Možda krajnjim korisnicima najpoznatiji i slikovito *najbliži*, web poslužitelji koji poslužuju različite web stranice također su važna stavka prilikom provedbe procesa očvršćivanja. Iako se ovdje velik dio sigurnosti oslanja na sigurno programiranje, implementirajući određene principe kao princip nultog povjerenja u kojemu se nalaže da se svaki korisnički unos provjerava bez ikakve pretpostavke visoke sigurnosti sadržaja, sigurna komunikacija s poslužiteljem baze podataka i slično, moguće je provesti sigurnosno očvršćivanje i s infrastrukturnog pogleda. To se može postiđu ostalog i kreirajući certifikate koji će se koristiti za šifriranje komunikacije s klijentima, osiguravanje identifikacije i autentifikacije poslužitelja i dr.

U sljedećem će poglavlju biti objašnjen postupak izrade ovog rada, uključujući metode i tehnike koje su korištene.

2. Metode i tehnike rada

U popisu literature navedena je knjiga o sigurnosti Linux poslužitelja kojoj je autor Michael D. Bauer. Budući da je knjiga napisana 2005. godine, služila je kao temelj za pronalazak ažurnijih informacija o praktičnim postupcima sigurnosnog očvršćivanja na Internetu. Osim te knjige, temelj za pronalazak detaljnijih informacija je i dokument preporuka Nacionalnog instituta za standarde i tehnologiju (NIST - eng. *National Institute of Standards and Technology*), odnosno dokument pod šifrom SP 800-123, vodič za opću sigurnost poslužitelja (eng. *Guide to general server security*). Temeljem informacija u prethodno navedenim izvorima, u izradi ovog završnog rada provedena su istraživanja putem Interneta u svrhu pronalaska najboljih praksi iz ovog područja na kojima se temelje primjeri i metode sigurnosnog očvršćivanja GNU/Linux poslužitelja prikazane u radu. Korišteni su web materijali odnosno dokumenti nekih od vodećih tvrtki u području informacijske sigurnosti i drugih autora kao i web stranice s tutorijalima koje su bile dodatna tehnička potpora u smislu praktičnih izvedbi, tj. realizacije preporuka iz spomenutih dokumenata.

Praktični dio sastoji se od niza naredbi te njihovih rezultata uz komentare kojima je cilj olakšati razumijevanje tijeka naredbi i njihovog utjecaja na određeni dio sustava ili funkcionalnosti. Proces sigurnosnog očvršćivanja proveden je u lokalnom virtualnom okruženju odnosno na virtualnom Debian Linux poslužitelju. Nakon provedbe svakog primjera, naredbe su pohranjene te prikazane u ovome radu kao praktični primjeri. Pri tome su korišteni različiti alati koji se koriste u područjima administracije sustava te informacijske sigurnosti, ali i u drugim područjima informacijskih tehnologija. Korišteni su alati poput upravitelja paketa za instalaciju i drugu manipulaciju paketima, iptables za konfiguraciju pravila vatroštita, rsyslog za upravljanje zapisima (eng. *logs*) i dr. Za testiranje povezanosti prema kreiranom Debian poslužitelju nakon provedenih procesa očvršćivanja koji se tiču vatroštita i sličnih pravila povezanih s mrežom korišten je dodatni klijent. Budući da većina korisničkih uređaja (stolnih i prijenosnih računala) pokreće Microsoft Windows operacijski sustav, spomenuti klijent također pokreće Microsoft Windows 10 operacijski sustav. Primjeri u kojima se prikazuje promet prema poslužitelju praćeni su snimkama zaslona koje približavaju sliku toga, npr. povezivanje na web poslužitelj Apache putem web preglednika na klijentu, pokretanje ping naredbe na klijentu prema poslužitelju i sl.

3. O sigurnosti i upravljanju rizikom

Svijest o sigurnosti informacijskih sustava počinje se sve snažnije razvijati s ubrzanim razvojem tehnologije i općenite umreženosti sustava. Podaci su vrjedniji nego ikada, potreba za povezanošću s Internetom zbog velikog udjela online usluga sve veća. Samim time, sigurnost takvih sustava i informacija kojima oni raspolažu sve je važnija. To potvrđuje i Sigurnosno-obavještajna agencija Republike Hrvatske na svojim stranicama, a dio toga iznesen je u sljedećem ulomku.

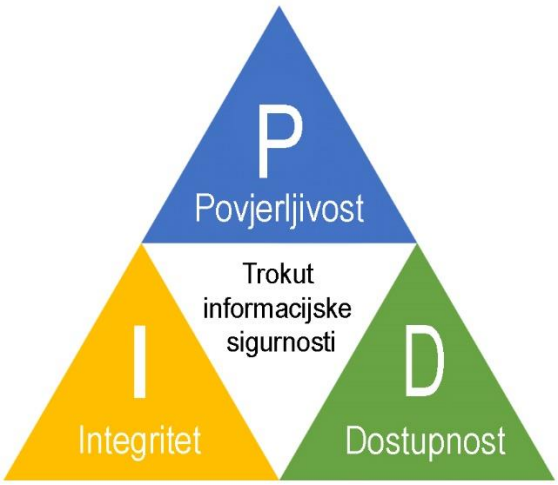
Razvojem tehnologije sve se više podataka važnih za nacionalnu sigurnost pohranjuje u informacijskim sustavima tijela državne uprave ili se razmjenjuju informacijsko-komunikacijskim kanalima. Elektronički napadi i ugrožavanja informacijske sigurnosti sve su složeniji te je u odgovoru na njih potrebno stalno učenje, praćenje trendova i inovativnost u rješenjima [1].

Velik dio u povećanju sigurnosti danas za organizacije ima proces koji se zove upravljanje rizikom. Već su na prvi pogled ta dva pojma vrlo povezana i to je s pravom tako. Rizik je svima poznat, nešto što intuitivno svaka osoba shvaća na više ili manje jednak način. Međutim, rizik bi se mogao definirati kao svojevrsna mjera kojom se predstavlja količina učinka nekog problema odnosno njegove posljedice. Za bolju ilustraciju te definicije može se uzeti sljedeći primjer. U slučaju baza podataka, mnoštvo osjetljivih podataka čuva se u određenom obliku. Ako su dakle podaci u bazi u čitljivom obliku, bez ikakvih sigurnosnih mjera koje bi omogućile njihovu zaštitu, rizik od kompromitiranja takvog sustava je ogroman jer nosi za posljedicu krađu osjetljivih podataka u čitljivom obliku koji odmah mogu biti zloupotrijebljeni. U drugom slučaju, ako su podaci u bazi šifrirani jakim kriptografskim algoritmom, a postoje algoritmi za čije bi dešifriranje i superračunalima trebala stoljeća i tisućljeća, rizik od napada na sustav i krađe podataka je manji, nije nepostojan, ali je relativno malen. Razlog tomu je što bi u slučaju krađe takvih podataka napadač imao neupotrebljive podatke koje ne može iskoristiti i najvjerojatnije ni ne bi mogao iskoristiti u dogledno vrijeme. S druge strane, svaki napad za sobom povlači rizik od gubitka povjerenja i reputacije. Mnoštvo je slučaja u kojemu su tvrtke propale zbog jednog većeg ili manjeg, ali uspješnog napada na njihove sustave nakon čega su njihovi klijenti izgubili povjerenje i prestali surađivati.

Upravljanje rizikom u ovom kontekstu ima za cilj smanjiti rizik napada, a uz to dodatno pojačati otpornost svojih sustava na napade. Svime se time pojačava sigurnost zbog čega je upravljanje rizikom, kao što je ranije spomenuto, značajan dio procesa ojačavanja sigurnosti. Upravljanje rizikom zapravo je temeljni korak u pojačavanju sigurnosti jer on pokreće sve ostale procese sigurnosnog očvršćivanja.

3.1. Principi informacijske sigurnosti

Sigurnost kao disciplina ima svoje ciljeve odnosno principe. U skorije se vrijeme ti principi svode na tri glavna pojma koji su povezani jedni s drugima i zajedno čine cjelinu. Ta tri pojma su povjerljivost, integritet te dostupnost. Često ih se spominje kao CIA trokut prema engleskom izvorniku (eng. *CIA Triangle*). C predstavlja povjerljivost (eng. *confidentiality*), I je integritet (eng. *integrity*), A predstavlja dostupnost (eng. *availability*). Jedna od ilustrativnih interpretacija tog pojma nalazi se na sljedećoj slici prevedenoj na hrvatski jezik.



Slika 1 - CIA Trokut (Prema: J. Nikander, O. Manninen, M. Laajalahti, 2020)

Ovakav prikaz dobar je jer na slikovit i jednostavan način predstavlja složenu strukturu ciljeva sigurnosti. Zbog toga ga je lakše zapamtiti i shvatiti, iako na višoj razini apstrakcije.

Ovaj princip uključen je i u upravljanje rizikom. Naime, tijekom upravljanja rizikom, organizacije mjere stopu rizika, prijetnji i ranjivosti koje bi mogle kompromitirati neki od spomenutih: povjerljivost, integritet ili dostupnost svojih sustava i podataka. Implementirajući sigurnosne kontrole kojima žele smanjiti rizik ili ga u potpunosti izbjeći, organizacije implementiraju upravo kontrole kojima zadovoljavaju neki od ova 3 glavna principa sigurnosnog CIA trokuta [2].

U nastavku će biti objašnjena sva tri pojma s nekoliko metoda kojim se ti principi zadovoljavaju u postupku sigurnosnog očvršćivanja, a neke od tih metoda detaljnije su opisane u praktičnim primjerima gdje se istovremeno jasno može vidjeti njihova implementacija, upotreba i rezultat.

3.1.1. Povjerljivost

Povjerljivost podataka podrazumijeva tajnost podataka. Drugim riječima, podrazumijeva sprječavanje neautoriziranog pristupa osjetljivim informacijama. Takav pristup može biti namjeran, poput napadača koji pristupa informacijama, a može biti i nenamjeran u slučaju kada radnik jednostavno ne mari o svojim postupcima ili nije obučen za sigurno upravljanje informacijama. To se može osigurati korištenjem različitih metoda, a neke od njih su kriptografija, maskiranje podataka te kontrola pristupa [2].

Podaci trebaju biti sigurni i u prometu odnosno razmjeni kroz mrežu te u mirovanju, odnosno u bazi podataka ili bilo kakvom drugom obliku u kojemu se mogu čuvati podaci na papiru, tvrdom disku, oblaku i sl. Maskiranjem i kriptografijom podataka osigurava se njihova povjerljivost jer se u slučaju kriptografije od šifriranog oblika podatka ne može doći do njegovog izvornog oblika, tj. dešifrirati ili dekriptirati ga bez poznavanja ključa kojim je taj podatak šifriran odnosno enkriptiran. Postoje simetrični algoritmi koji koriste isti ključ za enkripciju i dekripciju podataka te asimetrični kriptografski algoritmi koji koriste različite ključeve, većinom poznati kao par ključeva, javni i privatni ključ. Simetrični algoritmi su manje sigurni, ali brži od asimetričnih koji su pak sigurniji. Maskiranjem podataka je dostupan samo dio izvornog podatka, dok su ostali dijelovi maskirani posebnim znakovima, npr. zvjezdicama i sl. Zbog toga je teže doći do izvornog oblika u slučaju napada i krađe podataka ili eventualne slučajne objave tih podataka. Kontrolom pristupa osigurava se slojevitost o kojoj će više riječi biti u poglavlju o načinima obrane. Kontrolom pristupa smanjuje se površina napada, odnosno manje ljudi može pristupiti osjetljivijim podacima. Drugim riječima, što su podaci osjetljiviji, to osoba ili sustav treba imati više privilegije kako bi im pristupila. Zbog toga, u slučaju da napadač kompromitira neki sustav ili korisnički račun nižih privilegija koji je dobro izoliran od ostatka mreže, napadač ne može pristupiti osjetljivim podacima sve dok ne uspije povisiti svoje privilegije u mreži. Više o kriptografiji napisano je u ovome radu u poglavlju o sigurnosti baza podataka te poglavlju o sigurnosti web poslužitelja.

3.1.2. Integritet

Prema autoru R. Brooks [2] integritet ima 3 cilja u ostvarivanju sigurnosti podataka, a to su sprječavanje neautorizirane ili slučajne izmjene podataka od neautoriziranih korisnika, sprječavanje neautorizirane ili slučajne izmjene podataka od autoriziranih korisnika te očuvanje konzistentnosti, odnosno točnosti podataka. Integritet se također može očuvati kriptografijom, međutim često se koristi i sažimanje (eng. *hashing*). Algoritmi sažimanja (eng. *hashing algorithms*) kreiraju sažetak poruke koji je uvijek jedinstven za tu poruku i proces sažimanja nije reverzibilan. To bi značilo da se iz sažetka podatka u pravilu ne može dobiti

njegov izvornik, što također djelomično podupire i povjerljivost podataka jer se izvorni oblik ne može vidjeti. Druge dvije značajke algoritama sažimanja, a to su da ulaz promjenjive duljine uvijek daje izlaz fiksne duljine. Npr. algoritam SHA-256 daje uvijek izlaz duljine 256 bita. Treća je značajka da algoritam treba imati jako malo ili nikako kolizija.

Kolizija sažetaka (eng. *hash collision*) je pojava u kojoj dva ili više različitih ulaza u isti algoritam sažimanja daju isti sažetak, zbog čega se javljaju poteškoće i različiti napadi poput rođendanskog napada (eng. *birthday attack*) koji se zasniva na sljedećem principu. Nije potrebno znati točnu lozinku korisnika nego poznavati kombinaciju slova, brojeva i znakova koja će za korišteni algoritam sažimanja dati isti sažetak kao korisnikova lozinka. To se može izbjeći korištenjem sigurnijih algoritama koji imaju dovoljno veliku duljinu sažetka. Više o sažimanju napisano je u ovom radu u poglavlju o sigurnosti baza podataka jer se algoritmi sažimanja najčešće koriste za korisničke lozinke.

Integritet se također veže i za integritet sustava. Prema autoru M. D. Bauer [3, str. 4] on garantira da sustav nije kompromitiran i da služi svrsi za koju je prvotno i namijenjen. Također objašnjava da se podaci na kompromitiranom sustavu ne mogu smatrati povjerljivim i nepromijenjenim te da u pozadini može biti skrivenih procesa koji ostaju neprimjetni. Zbog toga se u oporavku takvih sustava većinom koristi potpuna obnova, odnosno potpuno brisanje tvrdih diskova, reinstalacija operacijskog sustava „s nule“ te obnova podataka iz pohrane s datumom pohrane prije datuma napada.

3.1.3. Dostupnost

Dostupnost osigurava da autorizirani korisnici sustava imaju neometan i osiguran pristup informacijama u sustavu i na mreži. Sasvim je razumljivo da su podaci kojima korisnik ne može pristupiti, kao i poslužitelji koji poslužuju te podatke beskorisni za krajnje korisnike sve dok su nedostupni. Zbog toga je ovo također važan princip informacijske sigurnosti. Metode koje podupiru dostupnost sustava i informacija prema autoru R. Brooks [2] jesu balansiranje opterećenja (eng. *load balancing*), visoka dostupnost (eng. *high availability*) te redundancija (eng. *redundancy*).

Balansiranje opterećenja podrazumijeva preusmjeravanje. Za primjer se može uzeti web trgovina koju poslužuje više poslužitelja na više različitih IP adresa. Tvrtka koja održava stranicu za web trgovinu primijeti da je promet na stranici veći u podne nego u ponoć. Zbog toga može koristiti *load balancer* uređaje ili softver koji će u vremenu veće posjećenosti prebacivati posjetitelje na različite IP adrese u ovisnosti o opterećenosti, tj. posjećenosti samih poslužitelja. Ako je recimo opterećenost poslužitelja 1 95%, a drugog 50%, sljedeći posjetitelj

bit će automatski prebačen na drugi poslužitelj kako se prvi ne bi preopteretio i zbog toga ugasio ili stvarao druge negativne utjecaje na iskustvo korisnika.

Visoka dostupnost odnosi se na mjere koje se koriste da bi se usluge i informacijski sustavi održali aktivnim i operativnim u slučaju nestanka energije. Cilj visoke dostupnosti je da ključne usluge budu dostupne 99.999% vremena te uključuje redundanciju [2].

Redundancija je pojam koji podrazumijeva da postoji više jednoličnih sustava, tako reći svojih identičnih kopija na koje se sustav može preusmjeriti u slučaju kvara i nedostupnosti glavnog poslužitelja. Ponovno je za primjer tvrtka koja održava web stranicu za web trgovinu. U slučaju otkazivanja radne memorije na jednom poslužitelju, on postaje nedostupan. Ako ne postoji identičan sustav koji se može odmah aktivirati po prestanku rada prvog poslužitelja, poslovanje trpi gubitke. Ako tvrtka ima zamjenski poslužitelj, on će se aktivirati i raditi uobičajene aktivnosti glavnog poslužitelja sve dok se dio ne zamijeni i poslužitelj aktivira.

Najčešći napad na dostupnost je napad uskraćivanja usluge (eng. Denial of Service). Dobra vijest o napadima na dostupnost sustava je ta da se nakon završetka napada, sustav ili mreža uobičajeno vrlo brzo mogu oporaviti. Pored toga, osim ako je napad uskraćivanja usluge kombiniran s drugim napadima, rijetko utječu na povjerljivost i integritet sustava te informacija [3, str. 5].

3.2. Principi sigurnosti poslužitelja prema NIST-u

Kroz prethodno je poglavlje predstavljen slikoviti prikaz 3 glavna principa informacijske sigurnosti. Takav je prikaz danas vrlo čest radi svoje jednostavnosti, ali treba imati na umu da ta tri principa nisu jednostavni, nego su složeni i postižu se u konačnici prateći mnoštvo detaljnijih principa koji na posljetku osiguravaju upravo povjerljivost, integritet te dostupnost. Takav prikaz principa informacijske sigurnosti na nižoj razini apstrakcije od CIA trokuta nalazi se u NIST-ovoj dokumentaciji [4]. Prema tom dokumentu, opće principe informacijske sigurnosti čine:

- jednostavnost – sigurnosni mehanizmi i informacijski sustavi općenito trebali bi biti što jednostavniji jer složenost stoji iza mnoštva sigurnosnih problema.
- sigurnost pri pogrešci (eng. *fail-safe*) - ako se dogodi greška, sustav bi se trebao isključiti na siguran način, npr. sigurnosne kontrole i postavke ostaju u aktivnom stanju i primjenjuju se. Uobičajeno je bolje izgubiti funkcionalnost nego sigurnost.
- potpuna medijacija (eng. *complete mediation*) – bolje je koristiti dodatne slojeve nego davati izravan pristup informacijama. Primjer toga mogu biti dozvole nad datotečnim sustavima te *proxy*, *firewall* ili *mail gateway*.

- Otvoreni dizajn – sigurnost sustava ne bi smjela ovisiti o tajnosti implementacije ili njegovih komponenata
- Razdvajanje privilegija – funkcije bi trebale biti odvojene i davati što je moguće detaljniju kontrolu nad istima. Npr. kod sustava bi funkcije čitanja, uređivanja, zapisivanja i izvršavanja trebale biti odvojene, a kod ljudi, uloga administratora sustava bi trebala biti odvojena od uloge administratora baza podataka, ako je to moguće.
- Najmanje privilegije – svaki zadatak, proces ili korisnik treba imati minimalna prava koja su mu dovoljna za izvršavanje njegovih dužnosti.
- Psihološka prihvatljivost – korisnici bi trebali razumjeti nužnost sigurnosti. To se može osigurati edukacijom i uvježbavanjem korisnika.
- Najmanje zajednički mehanizam (eng. *Least Common Mechanism*) – Kada se značajka implementira u sustavu, najbolje je imati jedan proces ili servis koji može dobiti neku funkciju bez da tu istu funkciju dozvoljava koristiti nekome drugome. Npr. mogućnost da proces web poslužitelja može komunicirati s bazom podataka ne bi smjela omogućiti drugim aplikacijama na sustavu pristup do baze podataka.
- Obrana u dubinu (eng. *Defense-in-Depth*) – organizacije bi trebale razumjeti da jedan sigurnosni mehanizam je općenito nedovoljan. Više njih bi se trebalo postavljati u slojeve tako da se u slučaju jednog kompromitiranog sigurnosnog mehanizma ne može kompromitirati čitav poslužitelj ili mreža.
- Čimbenik rada – organizacije bi trebale razumjeti što je potrebno da se provali u sustav ili mrežu i njege sigurnosne značajke. Ta količina rada koja je potrebna napadaču da provali u sustav ili mrežu trebala bi biti veća od vrijednosti koju bi napadač dobio uspješnim napadom.
- Zapisivanje kompromitiranosti (eng. *Compromise Recording*) – Zapisi (*logovi*) bi se trebali održavati tako da se u slučaju napada sačuvaju dokazi napada vidljivi organizaciji. Te informacije mogu pomoći u sigurnosnom očvršćivanju mreže i sustava nakon napada te pomoći u identificiranju metoda korištenih u napadu. Osim toga, mogu pomoći i u identificiranju napadača i poduzimanju drugih pravnih radnji.

Iz ovoga se može vidjeti da je CIA trokut zapravo sažeti prikaz principa sigurnosti, a kroz primjere i nastavak ovog rada može se vidjeti da svaki od primjera i korištenih metoda ima svrhu ostvarivanja barem jednog od navedenih principa informacijske sigurnosti, pa je tako npr. posvećeno poglavlje zapisivanju aktivnosti na operacijskom sustavu te zapisivanje aktivnosti na poslužiteljima baza podataka i web poslužiteljima, razne metode kojima se postiže odvajanje privilegija te princip najmanjih privilegija i sl.

3.3. Načini obrane

U ovome poglavlju objašnjena su tri načina obrane od napada koji stručnjacima informacijske sigurnosti služe kao nit vodilja prema uspješnom ojačavanju sigurnosti sustava, mreže, informacija i drugih vrijednosti. Napadi se mogu izbjeći ili u određenoj mjeri smanjiti vodeći se sljedećim ciljevima prema autoru M. D. Baueru [3, str. 18] : obezvređivanje imovine, ublažavanje ranjivosti te ublažavanje napada.

Obezvređivanje imovine može se činiti kao besmislen cilj, ali ključno je smanjiti vrijednost imovine (većinom informacija) napadačima, a ne njenim vlasnicima i korisnicima. Najbolji primjer ovoga je enkripcija. Ako je email učinkovito enkriptiran, napadač ga ne može pročitati, ako je digitalno potpisan, napadač ga ne može promijeniti bez primateljevog zapažanja [3, str. 19].

Ublažavanje ili uklanjanje ranjivosti još je jedna strategija obrane informacijske imovine. Ranjivost predstavlja propust koji se može iskoristiti u svrhu neautoriziranog pristupa sustavu, informacijama i sl. ili u svrhu postizanja da sustav radi ono za što prvotno nije bio namijenjen. Ovakvo što postiže se redovnim ažuriranjem softvera i instaliranjem zakrpa. Bitno je napomenuti da ažuriranja i zakrpe prvo treba testirati na odgovarajućim testnim platformama u testnim okruženjima. To se treba uraditi jer je moguće da određena zakrpa nakon instalacije negativno utječe na rad određenog dijela sustava čiji je rad kritičan i može nastati kaos u čitavom sustavu. Testiranjem se takve situacije izbjegavaju jer se na vrijeme uoči nepravilan rad. Unatoč tomu, moguće je da se nakon primjene testirane zakrpe ipak pojave poteškoće. Razlog tomu je što često testna okruženja nisu u potpunosti konfigurirana kao produkcijska okruženja i ta razlika igra ulogu u radu zakrpe koja onda negativno utječe na rad sustava.

Primjer ublažavanja ranjivosti je također sigurno programiranje, odnosno programiranje u kojemu programski kod prolazi kroz razne filtere koji provjeravaju je li kod ranjiv, npr. na *buffer-overflow* napade [3, str. 19].

Prema autoru M. D. Baueru [3, str. 19] ublažavanje ili izbjegavanje napada je način kojemu se danas pridodaje najviše pažnje implementacijom skenera za viruse, vatroštitovima i sl. Osim toga, napominje da je njihova svrha smanjenje napada, u slučaju vatroštita, smanjenje mrežnih napada te u slučaju skenera za viruse napada na sustave. Ovdje također spadaju i kontrole pristupa jer je njihova svrha da odrede je li neki korisnik od povjerenja ili ne, autoriziran za određenu radnju ili ne.

3.4. Upute za sigurnosno očvršćivanje

Za provedbu procesa sigurnosnog očvršćivanja često se preporučuje koristiti upute za sigurnosno očvršćivanje (eng. *security hardening guidelines*) jer se tako smanjuje moguća pogreška u smislu previđanja implementacije određenih sigurnosnih mjera. Mnoge vodeće tvrtke u području informacijske sigurnosti imaju svoje upute za sigurnosno očvršćivanje koje redovito ažuriraju kako bi bili u korak s novim tehnologijama, napadima, metodama, alatima itd. Skladno s tim postoje i upute za sigurnosno očvršćivanje pojedinih operacijskih sustava, određenih verzija operacijskih sustava, poslužitelja, klijenata, mreža i sl. Takve upute mogu biti detaljne u slučaju uputa za pojedine softvere, poslužitelje i sl., a mogu biti i općenite. Na takvim općenitim uputama temelje se detaljnije i često velike tvrtke kreiraju svoje upute koje njihovi administratori sustava koriste pri ojačavanju sigurnosti njihovih sustava i postavljene su kao standard za određenu tvrtku. Takve općenite upute vrlo su slične jer u konačnici imaju isti cilj i vođeni su istim principima informacijske sigurnosti. Radi usporedbe, u nastavku će biti spomenute 2 upute iz različitih izvora.

Upute prema CERT Hrvatska [5]:

- Ispravna instalacija i ažuriranje softvera na poslužitelju
- Administracija korisnika
- Kontrola izloženosti osjetljivih podataka
- Uklanjanje nepotrebnih usluga
- Zatvaranje nepotrebno otvorenih priključaka (eng. *ports*)
- Vatroštit

Upute prema tvrtci Netwrix [6]:

- Uklanjanje nepotrebnih funkcionalnosti
- Uklanjanje otvorenih priključaka i drugih mrežnih ranjivosti
- Upravljanje korisničkim računima i autentifikacijom
- Upravljanje servisnim računima

Već iz ova dva primjera može se vidjeti velika sličnost pokrivenog sadržaja. U sljedećem primjeru prikazane su još jedne općenite upute. Međutim, na ovome primjeru bit će ukratko objašnjene neke od metoda koje spadaju pod određenu stavku uputa. Dakle, prema autoru Oleg Zlotnik [7], ovo su upute s 8 stavki za sigurnosno očvršćivanje:

9. Upravljanje pristupom
10. Upravljanje mrežnim prometom
11. Zakrpa ranjivosti

12. Uklanjanje nepotrebnog softvera
13. Nadgledanje sustava (eng. *monitoring*)
14. Osiguravanje komunikacija
15. Provedba sigurnosnih pohrana
16. Očvršćivanje udaljenih sesija

Upravljanje pristupom podrazumijeva ne samo pristup unutar mreže i sustava, nego i fizički pristup. Primjer toga bio bi korištenje RFID kartica s kojima zaposlenici mogu ući u prostore tvrtke. Osim toga, upravljanje pristupom podrazumijeva i slojevitost sigurnosti, pojam poznatiji kao *security in depth*. Praktična izvedba ovoga je autorizacija korisnika, bilo za pristup informacijama, sustavima ili određenim prostorijama unutar tvrtke. Dakle, kontrolom pristupa omogućava se smanjena površina napada, a time i ranjivost cjelokupnog sustava jer je težina pristupa do osjetljivih podataka proporcionalna njihovoj osjetljivosti. Ove metode također počivaju na principu najmanjih privilegija prema kojemu određeni korisnik ima samo pristup informacijama i privilegije koje su mu potrebne za izvršavanje njegovih zaduženja i ništa više od toga. Taj je princip vezan i za princip nultog povjerenja, a to je princip koji zahtijeva dodatnu provjeru svega i svakoga jer ne postoji povjerenje. Čest primjer nultog povjerenja je u aplikacijama koje zahtijevaju korisnički unos. Takvom unosu nikad ne treba slijepo vjerovati jer se u tom slučaju mogu izvršiti injekcijski napadi koji mogu imati drastične posljedice.

Upravljanje mrežnim prometom također se zasniva na već spomenutim principima slojevitosti sigurnosti. Osim toga, javlja se i segmentacija odnosno izoliranje određenih dijelova mreže u podmreže (eng. *subnets*). Tim načinom kreiraju se izolirani sustavi i mreže čije kompromitiranje ne može naštetiti ostatku uređaja na toj mreži ili na drugoj podmreži u slučaju izoliranja podmreže. To je posebno važno implementirati zbog smanjivanja površine napada, ali i zbog lakšeg oporavka sustava i informacija na njemu. Često se u ovom kontekstu spominju demilitarizirane zone, pojam koji je preuzet iz vojne terminologije. Pojam demilitarizirane zone označava podmrežu kojoj se može pristupiti iz vanjske mreže, odnosno Interneta, a izolirana je od interne mreže kojoj se ne može pristupiti iz vanjske mreže. U tu zonu najčešće se postavljaju web poslužitelji.

Zakrpe ranjivosti već su spomenute u prijašnjim poglavljima pa ih ovdje nije potrebno detaljnije razmatrati.

Uklanjanje nepotrebnog softvera velik je dio u procesu sigurnosnog očvršćivanja jer znatno smanjuje površinu napada.

Nadgledanje sustava također je važno jer osim zaštite odnosno prevencije, bitna je i detekcija, odnosno na vrijeme uvidjeti napad da bi ga se moglo pravovremeno zaustaviti i poduzeti sve potrebne radnje.

Osiguravanje komunikacija odnosi se na korištenje enkriptiranih protokola poput TLS (eng. *Transport Layer Security*) protokola, SSH protokola umjesto Telnet protokola za udaljenu administraciju sustava itd.

Sigurnosne pohrane također igraju veliku ulogu u ovome procesu, a posebno su važne u slučaju napada, kvara u kojemu su podaci izgubljeni i sličnih negativnih događaja. Imajući kvalitetnu sigurnosnu pohranu moguće je brže i s manjim gubitcima vratiti sustav u operativno stanje.

Očvršćivanje udaljenih sesija odnosi se na osiguranje protokola i konfiguraciju alata koji se koriste za udaljenu administraciju sustava. Npr. koristeći kriptirani protokol SSH umjesto nekriptiranog protokola telnet omogućuje se sigurnija administracija sustava jer napadač koji eventualno prisluškuje promet ne može dešifrirati komunikaciju i saznati podatke o strukturi sustava, lozinke korisnika i sl. Također se odnosi na konfiguraciju postavki SSH poslužitelja, onemogućavanje spajanja bez autentifikacije korisnika i sl.

Ovo su temeljne metode na kojima se temelji proces sigurnosnog očvršćivanja s ciljem osiguranja 3 principa informacijske sigurnosti. Zbog toga se kroz čitav ovaj rad spominju i detaljnije objašnjava većina spomenutih metoda. Kroz praktični dio rada u kojemu su primjeri procesa sigurnosnog očvršćivanja Linux poslužitelja, web poslužitelja i poslužitelja baza podataka na njemu, može se vidjeti da su obrađene upravo neke od ovih metoda koje su u tim poglavljima detaljnije objašnjene. Na taj se način može praktično vidjeti njihova svrha, implementacija te rezultat njihove implementacije.

4. Očvršćivanje Linux poslužitelja na razini servisa i operacijskog sustava

Svaki poslužitelj ima operacijski sustav koji ga pokreće. Na tom sustavu se nalaze ostali servisi koji pružaju usluge raznim korisnicima. Na operacijskom sustavu se pokreću poslužitelji baza podataka, web poslužitelji, servisi za udaljeno upravljanje sustavom i sl. Budući da se sve to nalazi slikovito „na operacijskom sustavu“, to čini operacijski sustav temeljem za sve druge servise koji se instaliraju na poslužitelj. Naravno, operacijski sustav dalje ovisi o arhitekturi računala i drugim stvarima. Međutim, kao temelj za sve ostalo, potrebno je nakon instalacije prvo sigurnosno očvrstiti operacijski sustav, a nakon toga instalirati servise po želji i potrebi te ih očvršćivati pojedinačno. Ovakav se pristup naziva pristupom „odozdo prema gore“, a naravno postoji i suprotan pristup koji se bavi prvo ojačavanjem sigurnosti mreže, servisa pa operacijskog sustava i njegove jezgre.

Mnoge sigurnosne poteškoće mogu se izbjeći ako je operacijski sustav na poslužitelju prikladno konfiguriran. Budući da proizvođači nisu upućeni u potrebe svake organizacije glede sigurnosti, poslužiteljski administratori trebaju konfigurirati nove poslužitelje prema sigurnosnim standardima i uputama svoje organizacije te ih po potrebi ponovno konfigurirati u slučaju promjene standarda ili zahtjeva. [4]

U nastavku su detaljnije objašnjene neke od metoda sigurnosnog očvršćivanja Linux operacijskog sustava s primjerima.

4.1. Onemogućivanje nepotrebnog softvera (servisa)

Onemogućivanjem servisa se naravno smanjuje površina napada koju potencijalni napadač može istraživati i eventualno zlouporabiti. Time je smanjena mogućnost upada u sustav. Međutim, preporuka je servise koji se sigurno neće koristiti ili eventualno u daljoj budućnosti hoće, a ne trebaju zahtjevnu konfiguraciju, potpuno izbrisati iz sustava čime se štede resursi, ali i mogućnost eksploatacije ranjivosti koje takvi servisi mogu imati, a ugašeni i onemogućeni ne služe svrsi dulje vrijeme. Oni se uvijek mogu ponovno po potrebi instalirati i koristiti. Ako je pak za koristan rad potrebna složena konfiguracija, npr. Apache web poslužitelja, moguće je servis privremeno onemogućiti sve dok se ne javi potreba za njegovim ponovnim korištenjem.

Za upravljanje servisima na SystemD distribucijama najčešće se koristi alat systemctl koji je zapravo dio systemd upravitelja. Systemd je upravitelj sustava i servisa na Linux

distribucijama [8]. Drugi je način koristeći alat update-rc.d koji se zasniva na manipulaciji virtualnim poveznicama prema skriptama koje se pokreću prilikom pokretanja sustava, a nalaze se u /etc/init.d direktoriju. Update-rc.d također omogućava upravljanje razinama na kojima se određeni servisi hoće ili neće pokrenuti, a korišten je u Debian distribucijama. Treći način bio bi ručno uklanjanje logičkih poveznica prema skriptama u init.d direktoriju što se ne preporučuje zbog veće mogućnosti pogreške. U ovome radu prikazan je primjer rada sa systemctl alatom koji predstavlja sučelje za upravljanje systemd upraviteljem kako i samo ime sugerira.

Sintaksa korištenja alata: `systemctl [OPTIONS...] COMMAND [UNIT...]`

Primjer:

Prvo je potrebno provjeriti koji su sve servisi pokrenuti, kao što je prikazano ispod. Budući da je *service* alat namijenjen za upravljanje *startup skriptama*, odnosno skriptama koje se pokreću pri pokretanju sustava, dolje prikazanom naredbom dobit ćemo status svih servisa koji su za to namijenjeni. Mnoštvo je servisa koji su pokrenuti u pozadini i o njima ovisi rad sustava zbog čega nije preporuka zaustavljati ih ili pokretati ručno, osim ako korisnik zna točno što radi.

```
debian@debian10:~$ sudo service --status-all
 [ - ]  alsa-utils
 [ + ]  anacron
 [ + ]  apparmor
 [ + ]  avahi-daemon
 [ - ]  bluetooth
 [ - ]  console-setup.sh
 [ + ]  cron
 [ + ]  dbus
 [ + ]  gdm3
 [ - ]  hwclock.sh
 [ - ]  keyboard-setup.sh
 [ + ]  kmod
 [ - ]  lvm2
 [ - ]  lvm2-lvmpolld
 [ + ]  network-manager
 [ + ]  networking
 [ - ]  plymouth
 [ - ]  plymouth-log
 [ - ]  pppd-dns
 [ + ]  procps
```

```
 [ + ]  rsyslog
 [ - ]  saned
 [ - ]  speech-dispatcher
 [ + ]  ssh
 [ - ]  sudo
 [ + ]  udev
 [ + ]  ufw
 [ + ]  unattended-upgrades
 [ - ]  x11-common
```

U dobivenom ispisu znak plusa označava servis koji je pokrenut i koji se pokreće pri pokretanju sustava. Znak minusa pak označava da je servis instaliran, ali onemogućen što bi značilo da se ne pokreće pri pokretanju sustava.
Kao što je ranije spomenuto, ovi servisi nalaze se u /etc/init.d direktoriju. Pokrene li se naredba izlistavanja tog direktorija, očito je da su u ispisu iznad svi servisi iz tog direktorija.

```
debian@debian10:~$ ls /etc/init.d
alsa-utils         cron                lvm2              pppd-dns
sudo
anacron             dbus                  lvm2-lvmpolld     procps
udev
apparmor            gdm3                  networking        rsyslog
ufw
avahi-daemon         hwclock.sh             network-manager   saned
unattended-upgrades
bluetooth            keyboard-setup.sh   plymouth           speech-
dispatcher  x11-common
console-setup.sh  kmod               plymouth-log     ssh
```

Prethodna naredba ne prikazuje popis svih servisa na poslužitelju. Naredba koja to omogućuje jest sljedeća:

```
systemctl list-units --type=service
```

Napomena: *ispis prethodne naredbe je opširan te zbog toga nije prikazan*.

Da bi se određeni servis onemogućio, potrebno je unijeti sljedeće naredbe.

```
debian@debian10:~$ sudo systemctl disable ssh
Synchronizing state of ssh.service with SysV service script with
/lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install disable ssh
Removed /etc/systemd/system/multi-user.target.wants/ssh.service.
Removed /etc/systemd/system/sshd.service.
```

Trenutno je servis ssh onemogućen i neće se pokrenuti kada se sustav sljedeći put pokrene. Međutim, servis je još uvijek pokrenut i moguće ga je trenutno onemogućiti.

```
debian@debian10:~$ sudo systemctl stop ssh
debian@debian10:~$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: inactive (dead) since Mon 2022-08-08 06:33:15 CDT; 4s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Process: 2899 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
  Process: 2900 ExecStart=/usr/sbin/sshd -D $SSHD_OPTS (code=exited, status=0/SUCCESS)
 Main PID: 2900 (code=exited, status=0/SUCCESS)
```

Svi se instalirani servisi mogu naknadno omogućiti naredbom:

```
debian@debian10:~$ sudo systemctl enable ssh
```

4.2. Ažuriranje sustava i softvera

Kako je ranije spomenuto, ažuriranje sustava i softvera te primjena sigurnosnih zakrpa važan je korak u procesu sigurnosnog očvršćivanja sustava. Međutim, potrebno je staviti naglasak na prethodno testiranje svih zakrpa i ažuriranja na testnim okružjima da bi se otkrile eventualni konflikti zakrpa u radu sa softverom koji je nužan za funkcioniranje određenog sustava.

Na Debian sustavima ažuriranje sustava i softvera svodi se na korištenje apt-get alata. Ukratko, ažuriranje Linux sustava se u osnovi svodi na provjeravanje lokalnih popisa dostupnih paketa. Ako postoje novije verzije, provodi se postupak ažuriranja. U nastavku je prikazan slijed naredbi i njihovih ispisa za ažuriranje na Debian poslužitelju.

```
debian@debian10:~$ sudo apt-get update
Hit:1 http://security.debian.org/debian-security buster/updates InRelease
Hit:2 http://deb.debian.org/debian buster InRelease
Hit:3 http://deb.debian.org/debian buster-updates InRelease
Reading package lists... Done
debian@debian10:~$ sudo apt-get -u upgrade
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages have been kept back:
  linux-headers-amd64 linux-image-amd64
  …
```

Naravno, ispis naredbe je skraćen da ne bi bespotrebno zauzimao prostor. Međutim, može se vidjeti osnovna funkcionalnost koja je prethodno objašnjena. Prvom naredbom ažurirane su lokalni popisi dostupnih paketa koji se nalaze u direktoriju /var/lib/apt/lists, a drugom naredbom se pokreće proces ažuriranja. Zastavicom -u omogućuje se prikaz popisa paketa koji će se ažurirati [3, str. 67]. Zatim je pokrenuta naredba za čišćenje dodatnih bespotrebnih dokumenata koji se kreiraju u /var direktoriju [9]**.**

4.3. Upravljanje računima u sustavu

Potrebno je pregledati i ukloniti sve lokalne korisničke račune koji nisu potrebni. Za lokalne korisničke račune koji trebaju ostati aktivni, potrebno je postaviti stroge zahtjeve za složenost, duljinu, istek, ponovno korištenje i čestu promjenu lozinke. Također treba koristiti jake algoritme sažimanja za spremljene lozinke [6].

Popis svih računa se može vidjeti u datoteci passwd koja se nalazi na putanji /etc/passwd. Većina najčešće korištenih Linux distribucija pri instalaciji stvara priličan broj računa koji nisu nužni za rad sustava. Katkada čak kreira račune za aplikacije koje uopće nisu instalirane [3, str. 68].

Potrebno je pratiti kreiranje i korištenje računa u sustavu. Zbog toga se u organizacijama najčešće koriste centralizirani sustavi upravljanja kao MS Active Directory ili modernim PAM rješenjima (eng. *Privileged Access Management*) koji dopuštaju *Zero Standing Privilege* strategiju koja rješava tradicionalne probleme koji se javljaju s običnim računima i privilegijama koje su trajno dodijeljene i nikad se ne bi trebao koristiti root korisnik, a radnje s root ovlastima bi se trebale vršiti samo po potrebi [6].

Prema autoru M. D. Baueru [3, str. 69] najčešći potrebni računi koji su instalirani tijekom instalacije na Linux sustavu su: *root, bin, daemon, halt, shutdown, man, at.* Prema istom autoru, najčešći nepotrebni računi koji su instalirani tijekom instalacije na Linux sustavu su: *uucp, games, gdm, xfs, rpcuser* te *rpc.*

Međutim, potrebno je napomenuti da i nabrojani „nepotrebni“ računi mogu u određenim slučajevima korištenja biti potrebni za sustav. Zato je važno uzeti u obzir svrhu poslužitelja pri provedbi cjelokupnog procesa sigurnosnog očvršćivanja, pa tako i u odlučivanju o uklanjanju računa.

Preporuka je za servisne račune onemogućiti interaktivnu prijavu, odnosno korištenje naredbene ljuske općenito, a to se postiže postavljanjem teksta /bin/false na kraju retka za željeni račun u datoteci /etc/passwd kao što je prikazano ispod [3] za korisnički račun *games*.

Početno: games:x:5:60:games:/usr/games:/usr/sbin/nologin

Nakon promjene: games:x:5:60:games:/usr/games:/bin/false

Neaktivne račune i one koji se ne koriste ni za što, potrebno je obrisati iz sustava. To se radi naredbom *deluser*. Korištenjem te naredbe se tijekom brisanja računa ne briše i početni (eng. *home*) direktorij za taj račun. Za tu je svrhu potrebno koristiti zastavicu `--remove-home`.

```
debian@debian10:~$ sudo deluser games --remove-home
Looking for files to backup/remove ...
Not backing up/removing `/usr/games', it matches ^/usr/.*.
Removing user `games' ...
Warning: group `games' has no more members.
Done.
```

Preporuka je i onemogućivanje korištenja povišenih privilegija. Naime prefiksom *sudo* naredba se pokreće kao root korisnik odnosno s povišenim privilegijama. Zbog toga je potrebno kontinuirano održavati datoteku *sudoers* na putanji /etc/sudoers u kojoj su postavke za korištenje najviših privilegija odnosno korisnika koji se mogu koristiti naredbama kao root korisnik.

4.3.1. Upravljanje lozinkama

Upravljanje lozinkama također je važna stavka. Napad grubom silom (eng. *brute force attack*) napad je kojim se u teoriji može pogoditi bilo koja lozinka jer je to napad u kojemu računalo generira i testira sve moguće kombinacije slova, brojeva, znakova. Time je logično da će u jednom trenutku doći do kombinacije koju koristi neki račun za lozinku. Međutim, to nije moguće za složene i duge lozinke jer je broj kombinacija ogroman te time oduzima veliku količinu vremena. Zbog toga je napad grubom silom kada se koriste duge i složene lozinke skoro beskoristan jer računalo ne može u doglednom vremenu isprobati sve moguće kombinacije, osim „na sreću“.

Sigurna lozinka trebala bi imati minimalno 10 znakova, i to pod uvjetom da nije riječ o jednostavnoj ili predvidljivoj lozinki. Na DEB-based distribucijama konfiguracijske datoteke vezane uz lozinke i autentifikaciju nalaze se unutar mape /etc/pam.d/ [5].

Postavljanje minimalne duljine lozinke

Za postavljanje pravila za minimalnu duljinu lozinke potrebno je dodati taj uvjet (*minlen=<broj>*) u datoteci /etc/pam.d/common-password na sljedeći način.

Redak koji treba izmijeniti početno izgleda ovako:

```
# here are the per-package modules (the "Primary" block)
password      [success=1 default=ignore]     pam_unix.so    obscure sha512
```

Na kraj tog reda dodaje se uvjet *minlen=10* koji daje uputu da duljina lozinke ne može biti manja od 10.

```
# here are the per-package modules (the "Primary" block)
password      [success=1 default=ignore]     pam_unix.so    obscure sha512 minlen = 10
```

Složenost lozinke

Složenost lozinke podrazumijeva korištenje različitih elemenata u lozinci, u smislu velikih i malih slova, brojeva te posebnih znakova. Da bi korisnici morali postaviti dovoljno složenu lozinku, može se konfigurirati pravilo u sustavu koje će pri svakom kreiranju te promjeni lozinke zahtijevati definiranu složenost.

Na DEB-based sustavima za to je potrebno instalirati knjižnicu libpam-pwquality, koja, provjerava kvalitetu lozinke. Zatim s administratorskim ovlastima otvorimo datoteku /etc/pam.d/common-password u uređivaču teksta. [5].

```
debian@debian10:~$ sudo apt install libpam-pwquality
debian@debian10:~$ sudo nano /etc/pam.d/common-password
```

Redak koji je potrebno gledati u ovom kontekstu izgleda ovako:

```
password        requisite                       pam_pwquality.so retry=3 ucredit=-1 dcredit=-1
```

Kao što se može vidjeti, postoje opcije koje su podrazumijevano prisutne, a njihovo značenje te značenje i naziv dodatnih može se vidjeti ispod prema CERT.hr [5]:

- ucredit – velika slova
- dcredit – mala slova
- ocredit – specijalni znakovi
- minclass – koliko različitih klasa znakova mora imati lozinka
- remember=3 pamte se 3 lozinke i nova ne može biti ista kao neka od prethodne tri
- hashing=sha512 hashing algoritam
- dcredit=-1 -> bar jedno malo slovo

Zaključavanje računa

Još jedna metoda upravljanja lozinkama jest zaključavanje računa. Naime, korisnika treba ograničiti glede broja pogrešnih unosa lozinki prilikom autentifikacije. Time značajno utječemo na uspješnost *brute-force* napada. Zaključavanje računa u osnovi onemogućuje korisniku korištenje računa. To ima za posljedicu i nemogućnost autentifikacije pri čemu se lozinka ni ne provjerava nego se pokušaj odmah odbacuje uz poruku o zaključanom računu. Zbog toga, *brute-force* napadi traju puno dulje i postaju neupotrebljivi jer u slučaju da je zaključavanje računa definirano nakon 3 neuspjela pokušaja prijave i traje 1h vremena, to bi napadaču omogućilo da u nešto više od sat vremena isproba samo 6 mogućih kombinacija, a ima ih pregršt. Zbog toga je važno implementirati zaključavanje računa.

Prema dokumentu čiji je autor CERT.hr [5], potrebno je izmijeniti /etc/pam.d/common-auth datoteku tako da se doda redak naznačen ispod, a u njemu se zaključavanje računa primjenjuje i na root korisnika.

```
debian@debian10:~$ sudo nano /etc/pam.d/common-auth
auth   required                        pam_tally2.so      onerr=fail
deny=3 unlock_time=600 audit even_deny_root root_unlock_time=600
```

Praćenje i upravljanje poviješću lozinki

Upravljanjem poviješću lozinki korisnik za svoj korisnički račun mora s vremena na vrijeme promijeniti lozinku koja ne smije biti slična ili ista određenom broju prethodnih lozinki, a sve to ovisi o parametrima postavljenim u datoteci /etc/login.defs. Tri su najvažnija parametra u ovoj datoteci za ovu svrhu, a prikazani su ispod. Dakle, moguće je definirati koliko dana može maksimalno proći prije nego korisnička lozinka istekne i postane neupotrebljiva, koliko dana mora proći nakon promjene lozinke da bi ju korisnik mogao ponovno upisati te postavljanje upozorenja korisniku o potrebi promjene lozinke nekoliko dana prije isteka. Ta

pravila ponovno usporavaju i sprječavaju neke napade. Npr., ako napadač dođe do lozinke korisnika, ona neće dugo biti aktivna jer ju korisnik mora mijenjati, a osiguravanjem pamćenja lozinke parametrom remember u datoteci /etc/pam.d/common-password omogućuje se da jedna lozinka ne može biti upotrijebljena više puta. Sve to podupire ranije spomenutu metodu obezvređivanja imovine jer lozinke više nisu vrijedne ako su neupotrebljive.

```
debian@debian10:~$ sudo nano /etc/login.defs
PASS_MAX_DAYS   90
PASS_MIN_DAYS   7
PASS_WARN_AGE   14
```

Kada se ova konfiguracija promijeni i spremi, vrijedi samo za novododane korisnike koje će administrator sustava dodati nakon promjene. Kako bi vrijedile i za postojeće korisnike sustava, potrebno je za svakog korisnika unijeti sljedeću naredbu [5]:

```
$ sudo chage –M <days> <username>
```

Njome postavljamo valjanost trenutne lozinke u danima za odabranog korisnika. Nakon što lozinka prestane vrijediti, korisnik ju mora promijeniti poštivajući sva pravila koja su navedena u konfiguraciji lozinki.

4.4. Zapisivanje i praćenje aktivnosti - Rsyslog

Nakon početnog postavljanja poslužitelja, administratori trebaju kontinuirano upravljati njegovom sigurnošću. U tom procesu jedna od vitalnih aktivnosti je i upravljanje i analiza zapisa (eng. *log files*) uz provođenje redovnih sigurnosnih pohrana, oporavljanje od napada, redovno testiranje sigurnosti poslužitelja te sigurno provođenje udaljenog upravljanja poslužiteljem. Zapisi, odnosno *logovi* su često jedini zapis, tj. dokaz sumnjivog ponašanja i zbog toga je skupljanje, nadzor i analiza tih zapisa vitalno [4]. Zbog toga se i u ovome radu obrađuje zapisivanje aktivnosti kroz primjer rsyslog sustava.

Rsyslog je sustav otvorenog koda za procesiranje zapisa s visokim performansama. On je više od običnog zapisnika sustava (eng. *system logger*). To je zapravo alat koji može prihvatiti ulazne zapise od različitih izvora te slati zapise na razna odredišta [10]. Ovo je svojevrsna naprednija verzija *syslog* alata koji se još uvijek može naći na raznim distribucijama Linux sustava. Međutim, mogućnost rsyslog alata da prima zapise od različitih izvora te slanje zapisa na razna odredišta je njegova dodana vrijednost. Time je pogodan za centralizirano skupljanje zapisa i konfiguraciju prema SIEM (eng. *Security Information Event Management*) rješenju koje se danas često koristi u sigurnosnim operativnim centrima većih, a sve češće i manjih tvrtki koje rade s osjetljivim podacima i pokreću sustave koje je potrebno dodatno zaštititi. Takvi centralizirani poslužitelji za prikupljanje zapisa dodatan su sloj u sigurnosti jer u

slučaju naknadne promjene zapisa na klijentu, izvorni zapisi ostaju nepromijenjeni na centraliziranom poslužitelju za prikupljanje zapisa. SIEM rješenja služe za upravljanje sigurnosnim događajima. To su zapravo rješenja koja analiziraju skupljene zapise i njihovom analizom uočavaju potencijalne napade, anomalije u uzorcima ponašanja korisnika ili druge indikatore kompromitiranosti te o tome obavještavaju sigurnosne inženjere u sigurnosnim operativnim centrima koji dalje mogu djelovati prema potrebi. Često su SIEM rješenja ujedno i centralni poslužitelji za skupljanje zapisa pa nema potrebe za konfiguracijom više rješenja, ali ovakva mogućnost rsyslog alata definitivno je prednost nad nekim drugim alatima bez te mogućnosti.

Glavna konfiguracijska datoteka za ovaj alat je: /etc/rsyslog.conf

Ispis dijela konfiguracije (/etc/rsyslog.conf)

```
#
# First some standard log files.  Log by facility.
#
auth,authpriv.*                 /var/log/auth.log
*.*;auth,authpriv.none          -/var/log/syslog
#cron.*                         /var/log/cron.log
daemon.*                        -/var/log/daemon.log
kern.*                          -/var/log/kern.log
lpr.*                           -/var/log/lpr.log
mail.*                          -/var/log/mail.log
user.*                          -/var/log/user.log
```

Izvorna je konfiguracija dobro „dokumentirana“, tj. zakomentirana. Ti komentari prikazuju svrhu pojedinog retka. Isječak konfiguracije prikazuje, kao što je u komentaru i navedeno, neke od uobičajenih zapisa. Pregledom konfiguracije, može se primijetiti shema zapisivanja pravila *logiranja*:

```
Kategorija.prioritet          Mjesto_za_spremanje
```

Kategorija i *prioritet* tvore selektor. On, kako i samo ime sugerira predstavlja ono što je odabrano, a s desne strane je mjesto za spremanje zapisa odabranog (*selektiranog*) alata, programa i sl.

Kategorija (na engleskom i *Facility*) predstavlja kategorije poruka. Kao što je vidljivo i iz prethodnog isječka konfiguracije, podržani selektori su prema Gentoo wiki stranici [10]:

0. Kern
1. User
2. Mail
3. Daemon
4. Auth
5. Syslog
6. Lpr
7. News
8. Uucp
9. Cron
10. Security
11. ftp
12. ntp
13. logaudit
14. logalert
15. clock
16. local0->local7

Kao i za prethodni popis, za sljedeći popis prioriteta, odnosno važnosti poruka vrijedi da prvi nabrojani ima najveću važnost, a zadnje navedeni najmanju važnost.

Važnosti poruka prema Gentoo wiki stranici [10] mogu biti sljedeće:

0. emerg – sustav je neupotrebljiv
1. alert – potrebno je odmah intervenirati
2. crit – kritični uvjeti, ali sustav je još uvijek djelomično upotrebljiv
3. error – javljaju se pogreške u radu sustava
4. warning – upozorenja na nepravilnosti
5. notice – obavještenja, uobičajeno ponašanje, ali značajno za zabilježiti
6. info – informacije
7. debug – informacije na *debug* razini, većinom korištene za testiranje novih funkcionalnosti ili promjena

Pri zapisivanju pravila moguće je koristiti i modifikatore kada se određuje željeni prioritet. To su modifikatori znak jednako (=) i znak uskličnika (!). Ako se oni izostave, onda se pravilo tumači kao da vrijedi za navedeni prioritet i sve iznad. Ako se uključi znak jednako, to daje uputu za zapis samo poruka određene kategorije s točno tim prioritetom,

a znak uskličnika kaže da se zapisuju sve poruke određene kategorije niže od naznačenog prioriteta, isključujući navedeni prioritet. Znak != daje uputu da se zapisuju sve poruke određene kategorije koje imaju bilo koji prioritet osim naznačenog [3, str. 408]. U jednom se selektoru može uključiti više kategorija odvajanjem pojedinih stavki zarezom, a više selektora odvajanjem točka-zarezom što može biti korisno ako više različitih kategorija poruka treba biti smješteno u istu datoteku. Bitno je naglasiti da se u slučaju povezivanja više različitih selektora točka-zarezom pravila uključuju, tj. provjeravaju s desna na lijevo. Drugim riječima, potrebno je prvo pisati općenitija pravila pa specifičnija [3, str. 410]. Sve to prikazano je u sljedećim primjerima. U njima će biti prikazano pravilo kakvo bi bilo napisano u konfiguracijskoj datoteci rsyslog alata, a ispod njega nalazi se objašnjenje pravila.

Napomena: Prioritet .none znači da se ne uključuju poruke naznačene kategorije.

Primjeri pravila s tumačenjima:

1. auth.crit /var/log/auth.critabove
    - Sve poruke sigurnosnih događaja s prioritetom crit (eng. *Critical*) i više (u ovom slučaju alert te emerg) se zapisuju u datoteku na putanji /var/log/auth.critabove
2. auth.!=crit /var/log/auth.notcrit
    - Sve poruke sigurnosnih događaja osim onih s prioritetom crit su zapisane u datoteku na putanji /var/log/auth.notcrit
3. auth.=crit /var/log/auth.crit
    - Sve poruke sigurnosnih događaja s prioritetom crit zapisuju se u datoteku na putanji /var/log/auth.crit
4. mail,ftp.=notice /var/log/mailftp.notice
    - Sve poruke vezane za mail ili ftp s prioritetom notice zapisuju se u datoteku na putanji /var/log/mailftp.notice
5. mail.notice;mail.!=emerg /var/log/mail.noticeabove.notemerg
    - Sve poruke vezane za mail s prioritetom notice ili viši, osim poruka s prioritetom emerg zapisuju se u datoteku na putanji /var/log/mail.noticeabove.notemerg
6. mail.notice;mail.!=info /var/log/mail.notice.info
    - Ovo je pravilo pogrešno napisano jer povezuje dva selektora od kojih desni nije podskup lijevog. Zapravo, prvim selektorom uključuju se sve poruke kategorije mail, a prioriteta notice ili višeg. Sljedeći selektor je prioriteta info koji ima niži prioritet u odnosu na notice. Stoga se ovaj selektor nikada neće izvršiti i nepotreban je.

Praktični primjer testiranja definiranog rsyslog pravila s prikazom rezultata

Slijedi primjer u kojemu je kreirano pravilo zapisivanja svih poruka vezanih za kontrolu pristupa i sigurnosnih događaja, što uključuje i korištenje povišenih privilegija, odnosno pokretanja procesa kao root korisnik uz prikaz rezultata njegove implementacije. Drugim riječima, nakon što je kreirano pravilo, pokrenuto je nekoliko naredbi koje je prema definiranom pravilu rsyslog trebao zapisati u definiranu datoteku. Za kraj je ispisan dio zapisa koji se odnosi na pokrenute naredbe.

Za početak, napravljena je kopija početne konfiguracije kako bi se sačuvala izvorna datoteka u slučaju pogreški. Zatim je otvorena konfiguracija u tekstualnom uređivaču nano te upisan redak na kraj konfiguracije.

```
debian@debian10:~$ sudo cp /etc/rsyslog.conf /etc/rsyslog.conf.old
debian@debian10:~$ sudo nano /etc/rsyslog.conf
# Kreirana pravila
authpriv.*      /var/log/authpriv.all.cstm
```

Nakon zatvaranja i spremanja datoteke, izvršena je sljedeća naredba:

```
debian@debian10:~$ sudo cat /var/log/authpriv.all.cstm
```

Ta je naredba zapisana u datoteku koja je definirana (/var/log/authpriv.all.cstm), a zapis te aktivnosti izgleda ovako:

```
Aug  9 21:31:59 debian10 sudo:  debian : TTY=pts/0 ; PWD=/home/debian ; USER=root ; COMMAND=/usr/bin/cat /var/log/authpriv.all.cstm
Aug  9 21:31:59 debian10 sudo: pam_unix(sudo:session): session opened for user root by (uid=0)
```

Može se vidjeti da je sesija za root korisnika otvorena. Naravno, to je posljedica činjenice da je datoteka sa zapisima konfigurirana tako da zahtijeva povišene privilegije za manipulaciju, pa tako i čitanje. Proces naredbe cat je pokrenut i u trenutku izvršavanja nije završen. To se može provjeriti dodatno naredbom tail kojom će se pročitati zadnja 3 retka u datoteci.

```
debian@debian10:~$ sudo tail -n 3 /var/log/authpriv.all.cstm
Aug  9 21:31:59 debian10 sudo: pam_unix(sudo:session): session closed for user root
Aug  9 21:42:06 debian10 sudo:  debian : TTY=pts/0 ; PWD=/home/debian ; USER=root ; COMMAND=/usr/bin/tail /var/log/authpriv.all.cstm
Aug  9 21:42:06 debian10 sudo: pam_unix(sudo:session): session opened for user root by (uid=0)
```

Podebljano je označen redak koji prikazuje da je sesija zatvorena. Također, još jednom je potvrđen taj princip time što je zadnji zapis o tome da je otvorena sesija za root korisnika nakon pokrenute naredbe sudo tail, koja je također operacija čitanja i završena je nakon otvaranja root sesije, a neposredno prije njenog zatvaranja.

Drugi način testiranja vlastitih pravila definiranih u konfiguraciji jest korištenjem *logger* alata. Da bi se provjerilo točno pravilo, moguće je korištenje zastavice –priority odnosno skraćeno -p. Ona prima argumente u obliku selektora (par kategorija.prioritet) [11].

Prema tome, ispod je prikazano korištenje alata logger te ispis iz prethodno definirane datoteke unutar pravila. Napomena: Budući da su u pravilu uključene poruke authpriv odnosno security kategorije sa svim prioritetima, u primjeru će se testirati s dva različita prioriteta authpriv poruka.

```
debian@debian10:~$ logger -p "authpriv.notice" "Testna poruka"
debian@debian10:~$ logger -p "authpriv.warning" "Testna poruka Warning"
Aug  9 22:05:05 debian10 debian: Testna poruka
Aug  9 22:10:40 debian10 debian: Testna poruka Warning
Aug  9 22:10:49 debian10 sudo:  debian : TTY=pts/0 ; PWD=/home/debian ; USER=root ; COMMAND=/usr/bin/tail /var/log/authpriv.all.cstm
Aug  9 22:10:49 debian10 sudo: pam_unix(sudo:session): session opened for user root by (uid=0)
```

Ponovno su podebljani retci koji su relevantni za prikazani primjer. U njima se vide obje poruke koje su prethodno unesene alatom logger.

4.5. Vatroštit

Vatroštit, vatrozid, ili *firewall* na razini pojedinog uređaja još je jedan sloj obrane što podupire slojeviti pristup ojačavanju sigurnosti mreža i uređaja (eng. Security in depth). Temelji se na vatroštitu za Linux sustave zvanom netfilter. Međutim, najčešće se prepoznaje kao iptables jer je iptables jedan od najvažnijih alata za upravljanje netfilterom [12, str. 67]. Njime se upravlja listama za kontrolu pristupa (eng. Access Control Lists, ACLs). Budući da su takve liste temelj vatroštita, iptables je s pravom najprepoznatljiviji kao „vatroštit za Linux sustave“. ACL liste temelje se na odozgo prema dolje pristupu (eng. *Top-down approach*) što znači da se prilikom provjere pravila za određeni paket provjeravaju pravila odozgo prema dolje. Ako paket ne zadovaljava uvjete prvog pravila, provjerava se drugo i tako dalje do zadnjeg pravila. Zbog toga se preporuča staviti detaljnija pravila na vrh listi, a općenitija na kraj [5, str. 40]. Prema tome, da bi se smanjilo korištenje resursa, na vrhu listi preporuka je

staviti najčešći promet jer to smanjuje broj pravila koje će vatroštit proći pri provjeri prometa. Npr. Ako je definirano 100 pravila i 95. pravilo kaže da se dolazni paketi preko priključka 80 propuštaju, a nijedno pravilo prije njega ne spominje takve pakete, vatroštit će za takvo pravilo proći 94 pravila prije nego nađe podudaranje. Za web poslužitelj koji poslužuje web stranicu na priključku 80, ovakva konfiguracija nije logična i bilo bi dobro takvo pravilo premjestiti što je moguće više na vrh liste. Time se smanjuje broj iteracija potreban za pronalazak pravila koje paket zadovoljava. Osim toga, na kraju liste preporuka je imati tzv. DENY ALL pravilo koje zapravo kaže da se sav mrežni promet odbacuje. To je dobro jer vatroštit sav promet koji ne zadovoljava uvjete, dolaskom do kraja liste odnosno ovog pravila odbacuje i time smanjuje neželjeni promet preko poslužitelja.

U ovom poglavlju bit će prikazan praktičan primjer postavljanja pravila pristupa korištenjem iptables alata, a zatim korištenjem alata koji se skraćeno zove ufw, a stoji za akronim prema engleskom nazivu za uncomplicated firewall. Kako i samo ime pretpostavlja, radi se o alatu za pojednostavljeno upravljanje vatroštitom koji u pozadini koristi iptables alat. Ovaj je alat namijenjen za debian distribucije. Alternativa njemu na Red Hat Enterprise Linux i sličnim distribucijama jest firewalld koji ne podržava istovremeni rad s iptables, pa je na takvim distribucijama firewalld omogućen servis, a iptables onemogućen. U novije vrijeme sve češći je alat nftables koji je za razliku od navedenih fokusiran na univerzalnost (korištenje na različitim distribucijama) [12, str. 68].

U nastavku su prikazani primjeri postavljanja pravila pristupa alatom iptables na debian poslužitelju. U primjeru će biti prikazano ssh povezivanje drugog klijenta na debian poslužitelj te slanje ping zahtjeva u različitim konfiguracijama vatroštita. Početna konfiguracija listi za kontrolu pristupa je ispisana ispod. Na debian distribucijama je ona prazna i korisnik ih treba sam konfigurirati. Druge distribucije uključuju prethodno konfigurirana osnovna pravila pristupa. Budući da je lista prazna, sav promet je dopušten, bez obzira na polazište, odredište ili bilo što drugo.

```
root@debian10:/home/debian# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Tri su lanca (eng. *Chain*) prema kojim se pravila dijele unutar listi. To su input, forward i output. Input se odnosi na promet koji dolazi iz mreže na poslužitelj, odnosno odredišna adresa paketa tog prometa jednaka je adresi poslužitelja. Output se odnosi na promet koji šalje sam poslužitelj prema mreži, odnosno izvorišna adresa paketa tog prometa jednaka je adresi poslužitelja. Forward se odnosi na preusmjeravanje, odnosno prosljeđivanje. Točnije, to je promet koji „prolazi kroz“ poslužitelj, odnosno poslužitelj se ponaša kao usmjernik (eng. *Router*) ili je paket namijenjen drugoj mrežnoj kartici na poslužitelju [13].

Napomena za primjere: Bitno je naglasiti da je privatna adresa debian poslužitelja **192.168.0.20.** IP adresa klijenta s kojeg su poslane ping poruke na poslužitelj je **192.168.0.20**. U nekim primjerima će osim ponašanja klijenta biti prikazan i mrežni promet snimljen alatom tcpdump koji snima mrežni promet kako bi se prikazao tijek poruka između klijenta i poslužitelja.

Primjer: Pravilo DENY ALL

Ovo je vrlo jednostavno pravilo koje definira da se sav dolazni promet odbacuje.

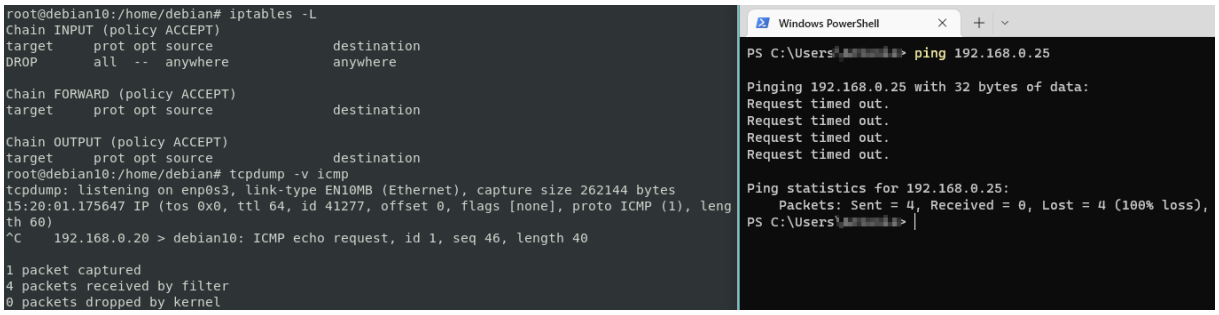
```
root@debian10:/home/debian# iptables -A INPUT -j DROP
root@debian10:/home/debian# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  anywhere             anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Zastavicom -A odabran je Input lanac, a zastavicom -j (eng. *jump*) se „skače“ na cilj pravila [14]**.** Međutim u praksi to označava akciju koja će se izvršiti s paketom koji zadovoljava ovaj uvjet. Sav dolazni promet je blokiran. Međutim, ovakva je konfiguracija dobra samo u slučaju jednosmjernih komunikacija poslužitelja prema mreži (kao što je UDP koji ne traži provjeru pristiglih paketa, nego ih samo šalje) jer u slučaju povratne veze na zahtjev poslužitelja, neće se dogoditi ništa. Jednostavan je primjer toga instaliranje alata. Pokretanjem naredbe *apt-get install* <*alat*> alat se neće moći preuzeti. Zahtjev će otići prema mreži,

međutim alat se neće moći preuzeti. U slici ispod prikazana je komunikacija ICMP protokolom između klijenta i poslužitelja na kojemu je konfigurirano ovo pravilo.

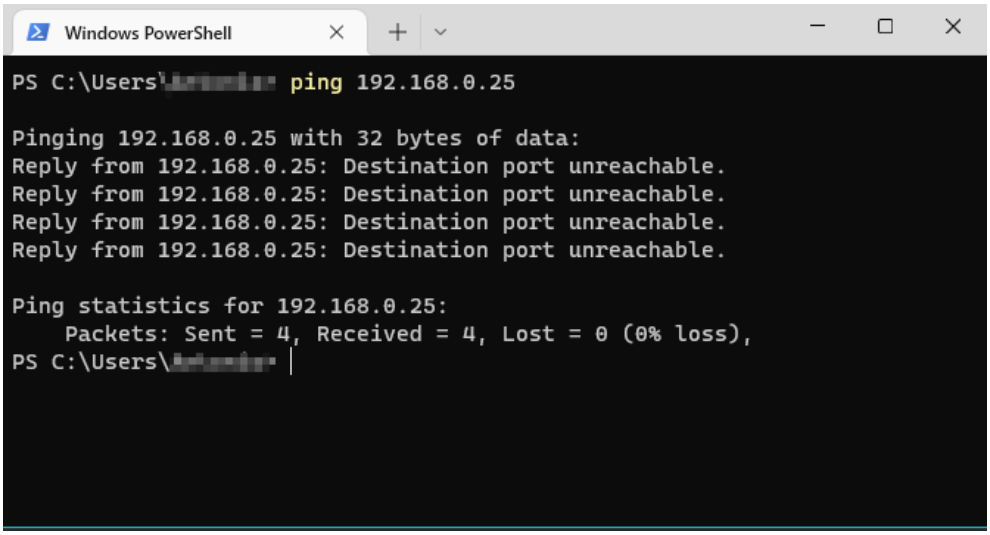


Slika 2 - DENY ALL INPUT pravilo vatroštita [autorski rad]

Primjer: REJECT pravilo s definiranim protokolom

U ovom primjeru prikazano je postavljanje REJECT pravila za ulazni promet ICMP protokolom. Ovo znači da će poslužitelj odbiti takav promet, ali s obavještenjem o tom odbijanju. U prethodnom primjeru korišteno je DROP pravilo koje također odbija promet, ali o tome ne šalje povratnu informaciju. Razlika se vidi uspoređujući sliku iznad sa slikom ispod. Dakle, u prethodnom primjeru klijent je dobio poruke *„Request timed out.“* što ne daje detaljnije informacije o poslužitelju. U ovom primjeru klijent je dobio poruke *„Reply from 192.168.0.25: Destination port unreachable.“*. Takva poruka klijentu daje do znanja da je poslužitelj konfiguriran da odbija takav promet na željeni priključak. Zbog toga je u nekim slučajevima korisno koristiti pravilo DROP umjesto pravila REJECT jer se potencijalnom napadaču ne šalju dodatne informacije o konfiguraciji poslužitelja koje bi se mogle zloupotrijebiti. Primjer za postavljanje ovakvog pravila i rezultat njegovog postavljanja slijedi ispod.

```
root@debian10:/home/debian# iptables -I INPUT -p icmp -j REJECT
```

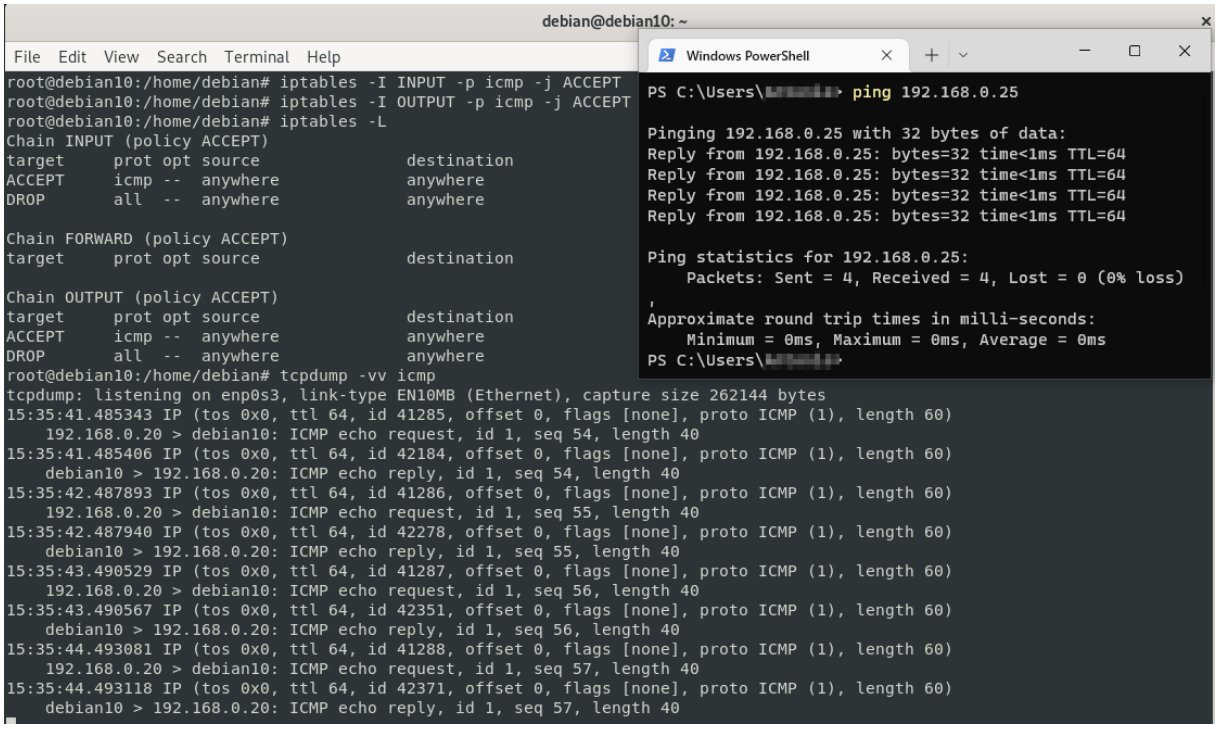


Slika 3 - REJECT ICMP INPUT pravilo vatroštita [autorski rad]

Primjer: ALLOW icmp INPUT OUTPUT

Prema prethodna dva primjera, ovaj se također zasniva na upravljanju prometa ICMP protokola. Prikazan je primjer dozvoljavanja prometa za ICMP protokol za ulazni i izlazni promet.

```
root@debian10:/home/debian# iptables -I INPUT -p icmp -j ACCEPT
root@debian10:/home/debian# iptables -I OUTPUT -p icmp -j ACCEPT
```



Slika 4 - ALLOW ICMP INPUT OUTPUT pravilo vatroštita [autorski rad]

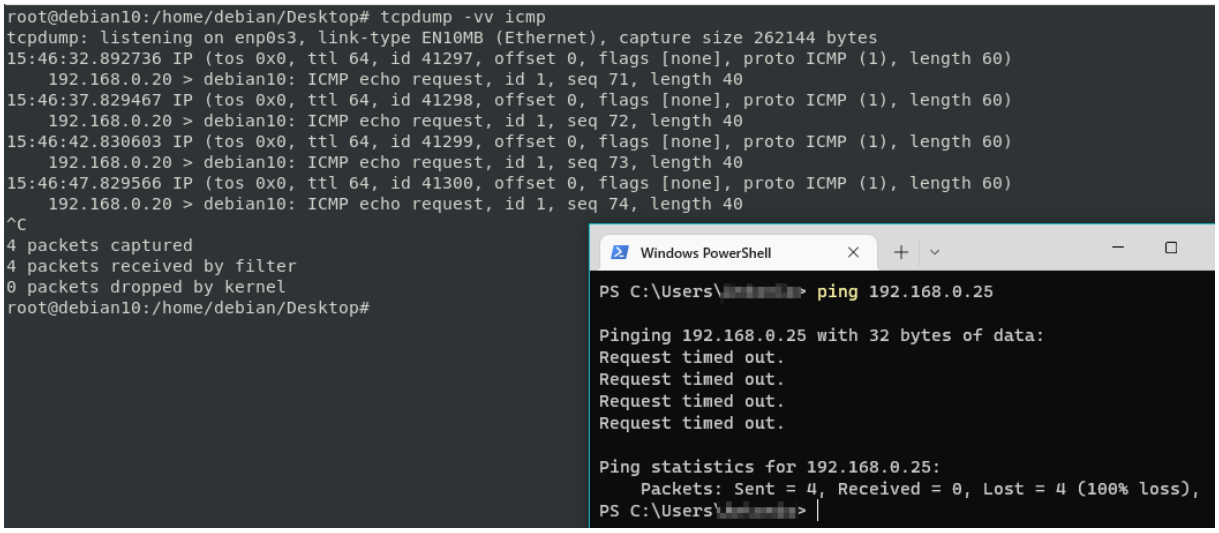
Primjer: Nepravilna konfiguracija liste za kontrolu pristupa

Svrha ovog primjera je pokazati kako postavljanjem općenitih pravila na vrh popisa za kontrolu pristupa dovode do pojave anomalija. Dakle, u primjeru se na početku postavlja pravilo za odbacivanje ulaznog prometa, a nakon toga je pravilo koje glasi da se dozvoljava ulazni promet ICMP protokola. Nakon toga, slična je konfiguracija za odlazni promet gdje se u prvom pravilo ne dozvoljava nikakav odlazni promet, a u sljedećem se dozvoljava odlazni promet ICMP protokola. Nakon naredbi za postavljanje pravila kontrole pristupa prikazana je i hijerarhija pravila kako izgledaju u popisu kontrole pristupa. Rezultat toga može se vidjeti na slici ispod naredbi, a on je sljedeći: Budući da se pravila u popisima kontrole pristupa pregledaju odozgo prema dolje, primjenjuje se prvo pravilo koje je u potpunosti zadovoljeno. Dolaznim ping zahtjevom na poslužitelj vatroštit pregleda INPUT lanac. Njegovo prvo pravilo je da se sav dolazni promet odbacuje bez povratne poruke o tome. Budući da je toliko općenito, dolazni ping zahtjev ga zadovoljava i prema tome, zahtjev je odbačen. Isto vrijedi i za odlazni promet. Iako je promet ICMP protokola dozvoljen drugim pravilom, svaka odlazna poruka poslužitelja zadovoljit će prvo pravilo i drugo se nikad neće uzeti u obzir. Zbog ovoga poslužitelj ostaje izoliran i prilično neupotrebljiv za komunikaciju s uređajima u ili izvan mreže.

```
root@debian10:/home/debian/Desktop# iptables -A INPUT -j DROP
root@debian10:/home/debian/Desktop# iptables -A INPUT -p icmp -j
ACCEPT
root@debian10:/home/debian/Desktop# iptables -A OUTPUT -j DROP
root@debian10:/home/debian/Desktop# iptables -A OUTPUT -p icmp -j
ACCEPT
root@debian10:/home/debian/Desktop# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  anywhere             anywhere
ACCEPT     icmp --  anywhere             anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  anywhere             anywhere
ACCEPT     icmp --  anywhere             anywhere
```

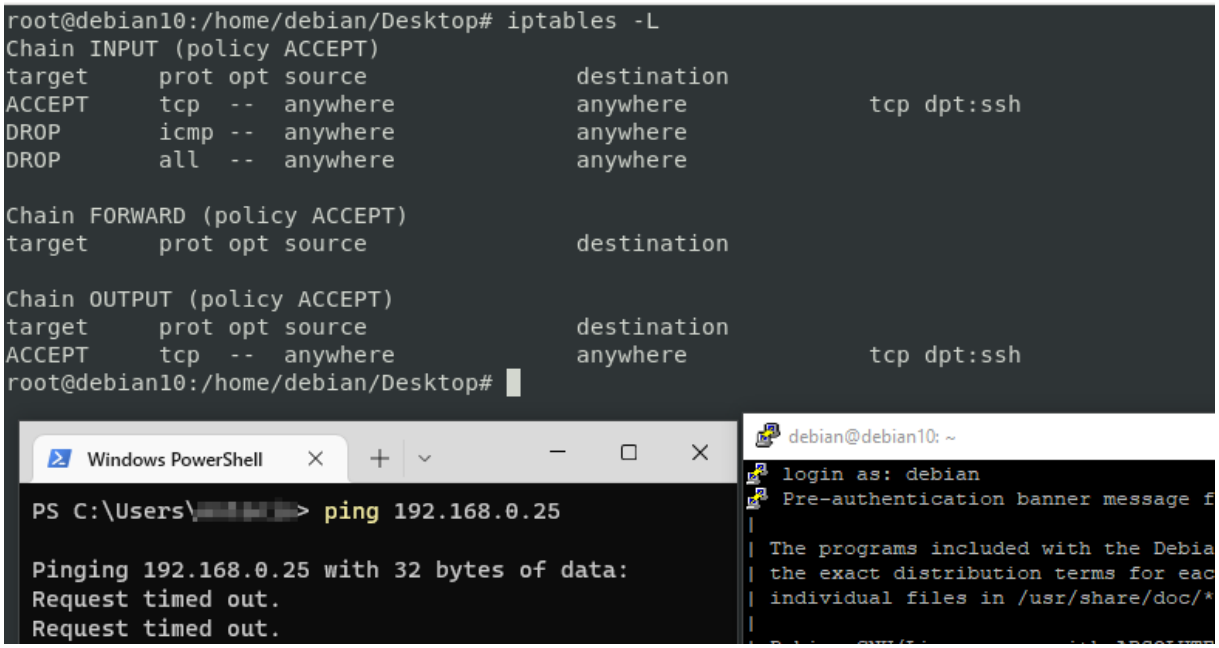


Slika 5 - Pogrešna konfiguracija pravila pristupa vatroštita [autorski rad]

Primjer: Blokiranje ICMP, dopuštanje SSH protokola

Budući da se pokretanjem ping zahtjeva često pokušava testirati aktivnost poslužitelja, često konfiguracija vatroštita sadrži pravilo za odbacivanje takvog dolaznog prometa. Naravno, još jedna česta pojava je udaljeno upravljanje poslužiteljom pa se takav promet treba dozvoliti. Prema preporuci općenito pravilo DENY ALL postavljeno je na kraj popisa pravila kako bi se smanjila količina neželjenog dolaznog prometa. Primjer jednostavne konfiguracije toga je prikazan ispod.

```
root@debian10:/home/debian/Desktop# iptables -A INPUT -p tcp --dport ssh -j ACCEPT
root@debian10:/home/debian/Desktop# iptables -A INPUT -p icmp -j DROP
root@debian10:/home/debian/Desktop# iptables -A INPUT -j DROP
root@debian10:/home/debian/Desktop# iptables -A OUTPUT -p tcp --dport ssh -j ACCEPT
```



Slika 6 - Blokiranje ICMP, dopuštanje SSH protokola [autorski rad]

Na slici iznad kao i u prethodnim primjerima prikazan je rezultat ovakve konfiguracije pravila pristupa vatroštita na poslužitelju kao i sama konfiguracija pravila. U donjem lijevom kutu slike prikazano je da ping zahtjev ne prolazi do poslužitelja, a u donjem desnom kutu prikazan je prozor u kojemu je uspješno izvršeno spajanje i autentikacija na poslužitelj SSH protokolom što je i bila svrha konfiguracije pravila u ovom primjeru.

5. Udaljeno upravljanje poslužiteljem

Udaljeno upravljanje (eng. *remote administration*) poslužiteljem bi trebala biti dozvoljena samo nakon pažljivog razmatranja rizika. Rizik omogućivanja ovakvog upravljanja uvelike varira u ovisnosti o lokaciji poslužitelja na mreži. Za upravljanje poslužiteljem iza vatroštita, udaljeno upravljanje može biti implementirano relativno sigurno iz interne mreže, ali ne bez ikakvog dodanog rizika. Udaljeno upravljanje općenito ne bi trebalo biti dozvoljeno s klijentskog računala koje nije unutar mreže organizacije osim ako se provodi s računala kojim upravlja organizacija i koje je pod njenom kontrolom (eng. *organization-controlled computer*) kroz rješenje za udaljeni pristup kojeg organizacija koristi, a to može biti VPN [4].

Ako se organizacija odluči za ovakvo upravljanje poslužiteljem, potrebno je dodatno sigurnosno očvršćiti takve sesije očvršćivanjem mreže, sigurnosnih pravila vatroštita, ali i alata koji se koristi za takvo što. Jedan od takvih alata je openssh na kojemu su izvođeni prikazani primjeri, a koristi ssh protokol za komunikaciju s poslužiteljem. Ranije su organizacije koristile protokole bez ikakvog šifriranja podataka poput telnet protokola. Korištenjem takvih protokola sva komunikacija je čitljiva eventualnom napadaču. Korištenjem SSH protokola takva se komunikacija enkriptira i postaje nečitljiva napadaču. U nastavku su neke od metoda sigurnosnog očvršćivanja sesija udaljenog upravljanja poslužiteljem.

5.1. Korištenje RSA parova ključeva za autentikaciju

Prijavljivanje korisničkim imenom i lozinkom uobičajeno je, međutim nije najsigurnije, pogotovo ako su lozinke slabe. Ako poslužitelj koristi ovakvo udaljeno spajanje, potrebno je implementirati druge mjere kao što je zaključavanje računa nakon nekoliko neuspješnih prijava itd. To će značajno usporiti i često obeshrabriti napadača koji pokušava izvesti brute-force napad jer se npr. Nakon svaka 3 pokušaja, račun zaključava na 2 sata. U 24 sata bilo bi moguće isprobati samo oko 36 kombinacija što je jako mali broj kombinacija za spomenuti napad. Pored toga moguće je implementirati i udaljeno spajanje samo za neke korisnike čime se smanjuje površina napada jer manje korisnika ima pristup i lakše ih je kontrolirati te nadzirati. Ipak, preporuka je koristiti RSA parove ključeva (privatni, javni) za autentikaciju i autorizaciju prema poslužitelju.

RSA je asimetrični kriptografski algoritam koji se sastoji od para ključeva, privatni i javni ključ [15].

Ovakav par ključeva ima veliku ulogu u sigurnosti, a jedna od njegovih upotreba je upravo u autentikaciji i autorizaciji prilikom udaljenog spajanja. Budući da su ključevi

matematički povezani, odgovaraju jedan drugom. Na taj način poslužitelj na osnovu javnog ključa zna je li zahtjev poslao baš onaj korisnik kojim se on predstavlja jer samo njegov privatni ključ odgovara za spremljeni javni ključ kojeg nitko drugi ne bi smio znati.

Za kreiranje para ključeva potreban je odgovarajući alat. Jednostavnom naredbom ssh-keygen kreira se par ključeva i dobije se ispis lokacije na kojoj je ključ spremljen. Nakon toga, javni ključ se sigurnim komunikacijskim kanalom šalje na poslužitelj i smješta pod /home/korisničko_ime/.ssh/id_rsa dokument. U konfiguraciji (/etc/ssh/sshd_config) potrebno je postaviti redak PubkeyAuthentication yes. Nakon toga, korisnik se može spojiti na poslužitelj putem ssh korištenjem svojeg privatnog ključa [12, str. 42].

Prilikom generiranja para, moguće je postaviti *passphrase* odnosno lozinku kojom se otključava pristup do privatnog ključa. Time je kasnije prilikom spajanja na poslužitelj nakon korisničkog imena potrebno upisati i *passphrase* do javnog ključa. To je još jedan dodatni sloj sigurnosti koji se često implementira i podupire princip „Security in depth“. Ako se korisnik želi prijaviti samo korištenjem korisničkog imena (i u pozadini privatnog ključa), onda ne treba postaviti passphrase, ali se time žrtvuje sloj sigurnosti i izlaže većem riziku od neovlaštenog preuzimanja privatnog ključa.

Za spajanje koristeći kreirani ključ, koristi se -i zastavica kao što je prikazano.

```
Ssh -i /putanja/do/ključa korisničko_ime@ipadresa
```

5.2. Onemogućivanje ssh spajanja kao root korisnik

SSH usluga je na većini Linux distribucija podrazumijevano konfigurirana tako da osluškuje na priključku 22 i na nekima dopušta prijavu root korisničkim računom. Podrazumijevano je onemogućena prijava lozinkom za root korisnika, on se može prijaviti samo putem javnog ključa (engl. public key) [5].

Iako je ovo jedan od načina ojačavanja sigurnosti, ipak je preporuka potpuno onemogućiti spajanje kao root korisnik. Administratori sustava trebali bi koristiti vlastite račune sa sudo privilegijama koje su im potrebne kako bi upravljali sustavom. To dodatno smanjuje površinu napada jer je onda za napadača teže kompromitirati korisnički račun i dodatno eskalirati privilegije kako bi dobio potpuni root pristup.

Vrlo je lako implementirati ovu sigurnosnu mjeru. U konfiguracijskoj datoteci na putanji /etc/ssh/sshd_config potrebno je pod odjeljkom Authentication dodati redak ili izmijeniti postojeći [5]. On na kraju treba izgledati ovako:

```
#Authentication:
PermitRootLogin no
```

Dodatno je mogućnost omogućavanja spajanja kao root korisnik prema IP adresi pa bi samo korisnik s tom IP adresom mogao biti udaljeno spojen kao root korisnik. Međutim, IP adrese se lako lažiraju (eng. *IP spoofing*) i onda takva mjera nema previše smisla ako napadač zna da je pristup zasnovan na IP adresi.

5.3. Banner poruka

Ovo je jedna od metoda tzv. Zastrašivanja (eng. *Deterrent control type*). Ona ne može zaustaviti ilegalne aktivnosti, ali svojim prisustvom može zastrašiti napadača. Primjer takvih metoda su i nadzorne kamere te znakovi upozorenja poput znakova s tekstom „Samo za zaposlene“, „Objekt je pod video nadzorom“ itd.

Banner je poruka koja se prikazuje korisnicima pri otvaranju udaljene sesije s poslužiteljem i sadrži različite poruke. Česte su potrebe za postavljanjem takvih poruka u svrhu obavještavanja korisnika o sigurnosti. Takvo nešto je često i zakonska obveza.Prema U konfiguraciji ssh servisa je podrazumijevana lokacija ove poruke na putanji /etc/motd. Ovaj akronim je još jedan akronim engleskog pojma Message of the Day. To je datoteka koju treba izmijeniti i unutra upisati tekst koji treba biti prikazan korisnicima koji koriste ssh da bi se spojili na poslužitelj [16, str. 131].

6. Sigurnost poslužitelja baza podataka

O važnosti sigurnosti poslužitelja baza podataka bilo je riječi ranije. U ovome poglavlju prikazat će se neke od metoda sigurnosnog očvršćivanja dva poslužitelja baza podataka, a to su postgresql te MySQL. Konfiguracijske datoteke za oba poslužitelja navedene su ispod.

Konfiguracijska datoteka za psql: pg_hba.conf i potsgresql.conf

Konfiguracijska datoteka za mysql: my.cnf

Naravno, jedan od osnovnih koraka je redovno **ažuriranje** poslužitelja za što se koriste upravitelji paketa ili druge metode ovisne o vrsti instalacije poslužitelja. Budući da se o ažuriranju softvera na linux sustavima već govorilo, u ostatku poglavlja prikazane su ostale metode očvršćivanja, svaka u svome poglavlju.

6.1. Odvajanje u zone radi kontrole pristupa

Jedna od metoda koja se ne tiče izravno poslužitelja baze podataka nego konfiguracije mreže jest odvajanje poslužitelja u zone kako bi se smanjila površina napada i doseg potencijalnog napada. Takvo izoliranje, osim fizički, može se izvršiti i koristeći pravila vatroštita.

Prilikom konfiguracije pravila vatroštita imajući na umu sigurnost pristupa bazi najviše se pažnje posvećuje protokolu, lokalnom priključku te izvorišnoj adresi [17].

Podrazumijevani (eng. *Default*) priključak za postgresql je 5432 [17],a MySQL 3306 [18].

6.2. Upravljanje računima

Budući da se upravljanje računima odvija koristeći SQL naredbe, one su zajedničke za sve SQL baze s minimalnim razlikama u sintaksi i prema tome su obrađene prije specifičnih radnji za postgresql i mysql. Neki od principa koji se prate u procesu upravljanja računima su principi najmanjih privilegija, nultog povjerenja i sl.

SQL računi rade na sljedeći način. Postoje uloge (eng. *Roles*) koje mogu biti dio drugih uloga, a jedan korisnik, može imati više dodijeljenih uloga. Takve uloge definiraju koja prava i nad kojim elementima baze ima koji korisnik [17]. Uloge se mogu definirati i njima pridruživati različiti atributi koji mogu biti predefinirani. Na ovaj je način olakšano upravljanje pristupom bazi podataka i veća vidljivost aktivnosti.

Najjednostavnija sintaksa za dodjeljivanje uloga korisniku izgleda ovako:

```
GRANT <naziv_uloge> TO <korisničko_ime>
```

Jedna od predefiniranih uloga u postgresql jest pg_monitor koja ima ovlasti pročitati konfiguracijske varijable, čak i one koje su vidljive samo korisnicima sa superuser ulogama (najviša uloga ovlasti, kao root korisnik u linux sustavu), pregledavati različite statističke podatke te pokretati nadziranje aktivnosti [17].

Da bi se takva uloga dodijelila korisniku s korisničkim imenom korisnik123, koristi se sljedeća naredba:

```
GRANT pg_monitor TO korisnik123;
```

Na sličan način mogu se dodjeljivati i različite ovlasti nad naredbama. Npr. korisniku *korisnik123* je potreban pristup za upisivanje podataka i pregledavanje istih nad tablicama u bazi naziva *tvrtka*. To bi se izvršilo sljedećom naredbom:

```
GRANT INSERT, SELECT ON tvrtka.* TO korisnik123;
```

Ako tome korisniku u budućnosti treba maknuti pravo upisivanja podataka iz bilo kojeg razloga, to se može drugom naredbom, ključnom riječi REVOKE.

```
REVOKE INSERT ON tvrtka.* FROM korisnik123;
```

Na isti način bi se takva dodjela mogla izvršiti i ulogama, ne samo pojedinim korisnicima.

Da bi korisnik aktivirao, tj. koristio neku od dodijeljenih uloga, može koristiti sljedeću naredbu:

```
SET ROLE <naziv_uloge>;
```

Ako želi isključiti sve uloge, za naziv uloge postavlja NONE, a za sve dodijeljene uloge postavlja ALL, a uloge se brišu sljedećom naredbom [19]:

```
DROP ROLE <naziv_uloge>
```

Iz ovih primjera jasno je da je SQL jezik vrlo intuitivan i ima jednostavnu sintaksu naredbi, međutim, SQL jezik kao takav nije tema ovog rada te neće biti detaljnije obrađen.

6.2.1. Upravljanje lozinkama

Kao što je već više puta spomenuto, potrebno je koristiti složene i duge lozinke. Nažalost, većinom nije dovoljno samo učiti korisnike o tome, nego je potrebno kreirati pravila za složenost lozinki koja kao takva sprječavaju bilo kakvu lozinku koja ne zadovoljava te minimalne uvjete. Primjer toga bio je obrađen i pod poglavljem očvršćivanja linux poslužitelja

s pogleda servisa i operacijskog sustava. Budući da je takvo nešto postalo uobičajena praksa, veći dio softvera dolazi s predefiniranim pravilima koja su podrazumijevano uključena.

To je slučaj i s ova dva sustava, PostgreSQL te MySQL. Podrazumijevana pravila za oba sustava navedena su ispod prema bazi znanja sveučilišta u Indiani [20]. Za PostgreSQL predefinirana pravila su:

- Nova lozinka se treba razlikovati od prethodne bar za 4 znaka
- Treba sadržavati bar:
    - 9 znakova
    - 2 velika slova
    - 2 mala slova
    - 2 broja
    - 2 posebna znaka

  Za MySQL to su:

- Treba sadržavati bar:
    - 9 znakova
    - 2 velika slova
    - 2 mala slova
    - 2 broja
    - 2 posebna znaka

  Skup posebnih znakova je isti za oba sustava, a to su:

  ` ~ ! @ # $ % ^ & * ( ) _ - + = { } [ ] / < > , . ; ? ' : | (razmak)

6.2.2. Enkripcija podataka u prijenosu

Još jedna važan element je enkripcija svih podataka u prijenosu, ali i onih koji su pohranjeni u bazama, tablicama i sl. Za ovakvo što se uključuje korištenje TLS protokola (eng. *Transport Layer Security*) koji enkriptira podatke u prijenosu i štiti podatke od direktnog čitanja u slučaju napada čovjeka u sredini (eng. *Man in the middle attack*) u kojem napadač prisluškuje promet između klijenta i poslužitelja i presreće podatke. Za korištenje TLS-a potrebni su digitalni certifikati. Digitalni certifikati općenito igraju veliku ulogu u informacijskoj sigurnosti. To su dokumenti ili elektroničke lozinke koje dokazuju autentičnost uređaja, poslužitelja ili korisnika korištenjem kriptografije i infrastrukture javnog ključa [21].

Oni se mogu kupiti od ovlaštenih izdavača certifikata (eng. *Certificate Authorities*) koji onda garantiraju za autentičnost tih certifikata. Osim toga, mogu se i kreirati za neke jednostavnije potrebe. Jednom kada su certifikati pribavljeni, mogu se početi koristiti za uspostavljanje TLS poveznica. Za MySQL se u konfiguracijskoj datoteci postavljaju sljedeći parametri:

```
[mysqld]
ssl-ca=/path/to/ca.crt
ssl-cert=/path/to/server.crt
ssl-key=/path/to/server.key
```

Nakon toga, koristeći GRANT naredbe s dodatnom frazom REQUIRE SSL uspostavlja se pravilo za odabrane uloge ili korisnike. Tim se pravilom zahtijeva korištenje TLS prilikom komunikacije sa SQL poslužiteljem [22].

Na postgresql se nakon kreiranja ili dobivanja certifikata oni smještaju na putanje /etc/ssl/certs te /etc/ssl/private, postavljaju preventivna prava pristupa (400) te se postavlja korisnik koji pokreće postgresql kao vlasnik tih ključeva naredbom chown. U postgresql ljusci se upisuju sljedeće dvije naredbe kojima se omogućuje TLS korištenje te ponovno učitava konfiguracija kako bi se promjene primijenile:

```
postgres=# alter system set ssl=on;
ALTER SYSTEM
postgres=# select pg_reload_info();
pg_reload_conf
--------------
t
(1 row)
```

U konfiguracijsku se datoteku pg_hba.conf koja se ponaša kao svojevrsna lista s kontrolom pristupa, unosi sljedeći redak:

```
#TYPE DATABASEUSER    ADDRESS METHOD
Hostssl all   all     0.0.0.0/0       md5
```

Ponovno se učitava konfiguracija i TLS postaje nužan za uspostavljanje veze između klijenta i poslužitelja [23].

6.3. Zaštita rezidentnih podataka

Ovaj se pojam u ovom kontekstu uglavnom odnosi na sakrivanje osjetljivih podataka koji se čuvaju u tablicama baza podataka. Većinom se s ovim pojmom susreću programeri aplikacija koje rade s pozadinskom bazom podataka te programeri baza podataka. Srž je očuvanje integriteta te povjerljivosti podataka. Više je načina za učiniti to, međutim tri najčešća su maskiranje, tokenizacija i sažimanje podataka nekom od funkcija za sažimanje s dovoljno velikom sigurnosti ključa. Maskiranje podataka zapravo predstavlja skrivanje dijela podataka krajnjim korisnicima. Često je vidljiv primjer toga s brojevima mobilnih telefona ili brojevima kartica u kojima se uvijek prezentira nekoliko zadnjih brojeva, dok su ostali označeni znakom X ili *.

Pod pojmom maskiranje neki izvori postavljaju i pojam enkripcije koji je više puta objašnjen kroz rad. Dakle radi se o maskiranju podataka promjenom njegovog oblika (enkripcijom). Do originalnog oblika može se doći dekripcijom samo kada je prisutan odgovarajući ključ. Tokenizacija predstavlja proces u kojem se podaci smještaju na udaljenu lokaciju i podacima se dodjeljuju ključevi/tokeni koji onda predstavljaju te podatke. Umjesto korištenja izvornih podataka, koriste se tokeni koji ih predstavljaju. Zbog toga se taj proces i zove tokenizacija. Sažimanje (eng. *Hashing*) predstavlja promjenu oblika podataka tako da se ne može doći do izvornog oblika informacije. Za sažimanje se koriste funkcije ili algoritmi sažimanja koji su jednosmjerni i ne mogu vratiti izvorne podatke iz sažetih [24].

Jedna od najčešćih uporaba sažimanja je u spremanju podataka poput lozinki. Upadom u sustav i dohvaćanjem osjetljivih podataka, napadač dobije sažetke lozinki i ne može ih dešifrirati. To je jedan sloj obrane jer postoje različiti napadi poput Pass The Hash u kojemu napadač sustavu umjesto lozinke šalje sažetak i na osnovu tog podudaranja, sustav autentificira napadača kao legitimnog korisnika iako on to nije. Drugi napad povezan s ovim je poznat kao rođendanski napad (eng. *Birthday attack*), simbolično povezan s temom rođendana. Međutim, temelji se na tome da napadač u napadu ne treba pogoditi korisnikovu lozinku nego treba pogoditi kombinaciju slova, brojeva i znakova koji daju isti sažetak kao njegova lozinka. Razlog tome je pojava tzv. Podudaranja sažetaka (eng. *Hash collision*). On označava pojam kada dva različita podatka daju isti sažetak. To je moguće izbjeći, odnosno umanjiti pojavu takvih slučajeva korištenjem funkcija za sažimanje koje daju jako dug sažetak. Npr. Funkcija sažimanja md5 daje sažetak od 128b, a sha256 daje sažetak od 256b. Samim time, budući da je kombinacija slova koja daje sažetak ograničena na puno manju duljinu, veća je mogućnost pojavljivanja istih sažetaka za različite podatke. Zbog toga se preporučuje koristiti funkcije s duljim/većim sažetkom. Također postoji i napad tzv. Rainbow tablicama koje

sadrže izvorni tekst i njegov sažetak. Ako je sažetak u toj tablici, napadač zna koji tekst treba unijeti u polje kako bi ga sustav prepoznao kao autentičnog.

Međutim, postoje podaci koje je besmisleno sažimati. U Sjedinjenim Američkim Državama je to recimo njihov Social Security Number koji je set nasumičnih brojeva. Već postoje rainbow tablice koje su zlonamjerni napadači kreirali. Naime, budući da je to set nasumičnih brojeva određene duljine, moguće je kreirati sve moguće kombinacije toga. Naravno, to zahtijeva vrijeme, ali je izvedivo. Za sve te kombinacije su napadači kreirali i sažetke. Zbog toga nema smisla sažimati takve podatke jer se u slučaju napada i neovlaštenog pristupa podacima ovim napadom jako brzo „dešifriraju“ originalni podaci [24]. Sličan primjer na našim područjima bili bi OIB brojevi u Republici Hrvatskoj.

7. Sigurnost WEB poslužitelja

Danas su ovakvi poslužitelji česta meta napada jer je upotreba web usluga postala masovna. Osim toga, aplikacije koje se poslužuju često zahtijevaju veliku količinu resursa zbog čega je potrebno osigurati njihovu stabilnost da bi bile dostupne za korištenje. Samim time predmet su sigurnosnog očvršćivanja. WEB poslužitelji na kojima će ovdje biti prikazani primjeri su Apache i Nginx, jedni od najkorištenijih danas. Instalacija je vrlo jednostavna. Korištenjem upravitelja paketa, odnosno na Debian poslužitelju naredbama sudo apt install apache2, sudo apt install nginx su oba poslužitelja spremna za korištenje i konfiguraciju. Nakon toga se prema sljedećoj preporuci može manipulirati vatroštitom radi osiguranja dodatnog stupnja izoliranosti kao što je to bio slučaj i s poslužiteljima baza podataka u prethodnom poglavlju.

Preporuka za rad na ojačavanju sigurnosti WEB poslužitelja je da se privremeno blokira pristup priključcima 443 za HTTPs, 80 za HTTP i ostalima koje web poslužitelj koristi kako bi se onemogućio neovlašten pristup za vrijeme konfiguriranja postavki. Nakon testiranja i potvrde o pravilnom radu, potrebno je otvoriti priključke i povezati poslužitelj s „vanjskim svijetom“, odnosno internetom [3, str. 317].

Iako je mogućnost za takvim upadom mala jer je vremenski okvir potreban za konfiguraciju mali, a u mnoštvu je poslužitelja na internetu smanjena mogućnost odabira baš poslužitelja na kojemu se trenutno radi, nikad nije nepostojeća i treba ju uzeti u obzir.

Metode koje su u nastavku prikazane su naravno usmjerene prema održavanju principa informacijske sigurnosti: povjerljivosti, integriteta i dostupnost. Prema tome, njihovi ciljevi su smanjenje površine napada, slojevita sigurnost, korištenje principa najnižih privilegija (eng. *Least Privilege*) i dr. Osnovni princip je održavanje poslužitelja i redovno ažuriranje stabilnim verzijama softvera što je već obrađeno u prethodnim poglavljima rada. Glavna se konfiguracijska datoteka nalazi na putanji /etc/apache2/apache2.conf kod Debian/Ubuntu sustava te na putanji /etc/httpd/conf/httpd.conf na RHEL/CentOS/Fedora sustavima [25]. Slično, na putanji /etc/nginx/nginx.conf je glavna konfiguracijska datoteka za nginx poslužitelj.

U nastavku su nabrojane neke od metoda uz praktični primjer njihove izvedbe.

7.1. Sakrivanje povratne informacije o verziji poslužitelja

Da bi se smanjio broj nepotrebno izloženih podataka provodi se sakrivanje povratne informacije o verziji poslužitelja. Naime, poslužitelji su izvorno konfigurirani tako da prilikom određenih odgovora klijentu šalju informacije o verziji poslužitelja koja se koristi. U slučaju da poslužitelj nije ažuriran i sustav koristi verziju koja ima poznatu ranjivost, napadač bi odmah dobio informaciju da je poslužitelj ranjiv i imao bi detaljnije informacije o zloupotrebi te ranjivosti. Zbog toga je preporuka takve informacije sakriti u produkcijskom okruženju.

Isključivanje povratne informacije na Apache poslužitelju

Bez mijenjanja početne konfiguracije, ovo je odgovor poslužitelja koji se dobije zahtjevom nepostojeće stranice.

Slika 7 - Uključene informacije o verziji poslužitelja - Apache [autorski rad]

Prvi je korak otvaranje konfiguracijske datoteke s povišenim privilegijama.

```
debian@debian10:/$ sudo nano /etc/apache2/conf-enabled/security.conf
```

U datoteci bi ovi retci trebali biti upisani, a retci s drugim atributima za ove ključne riječi zakomentirani. Ostali retci u ovom slučaju nisu relevantni.

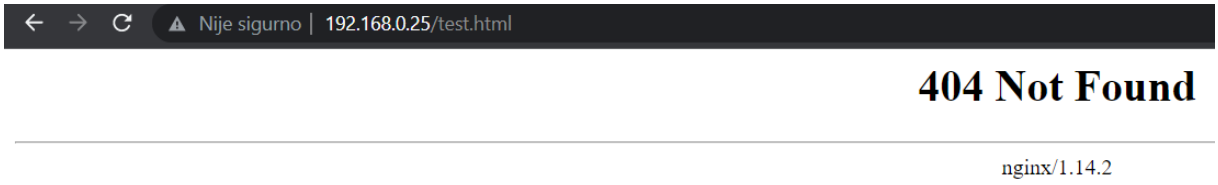
```
ServerSignature Off
ServerTokens Prod
```

Nakon resetiranja servisa apache2, odgovor koji se dobije u poruci pogreške od poslužitelja prikazan je na sljedećoj slici.

Slika 8 - Isključene informacije o verziji poslužitelja Apache [autorski rad]

Isključivanje povratne informacije na Apache poslužitelju

Bez mijenjanja početne konfiguracije, ovo je odgovor poslužitelja koji se dobije zahtjevom nepostojeće stranice.

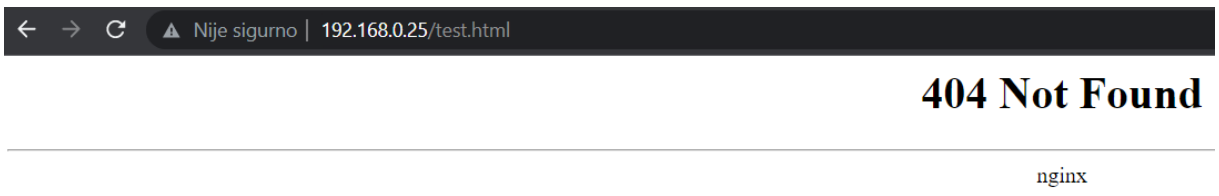


Slika 9 - Uključene informacije o verziji poslužitelja - Nginx [autorski rad]

Prvi je korak otvaranje konfiguracijske datoteke s povišenim privilegijama, a nakon toga atribut *server_tokens* treba isključiti postavljanjem ključne riječi *off* kao što je prikazano ispod.

```
debian@debian10:/$ sudo nano /etc/nginx/nginx.conf
server_tokens off
```

Nakon resetiranja servisa nginx, ovo je odgovor koji se dobije u vraćenoj HTTP poruci pogreške:



Slika 10 - Isključene informacije o verziji poslužitelja - Nginx [autorski rad]

7.2. Uklanjanje nepotrebnih modula

Uklanjanjem nepotrebnih modula koji su instalirani ranije ili su instalirani zajedno s osnovnom instalacijom poslužitelja dodatno smanjuje površinu napada. Naime, svaki vanjski modul koji se koristi i instaliran je, ima određen pristup sustavu. Svaki je softver podložan greškama, kao i sve drugo. Stoga je potrebno ukloniti nepotrebne module koji se ne koriste.

Instalirani i omogućeni moduli mogu se vidjeti u direktoriju /etc/apache2/mods-enabled, a dostupni koji nisu omogućeni, ali su prisutni na poslužitelju, mogu se vidjeti u direktoriju /etc/apache2/mods-available [26]. Svi moduli nalaze se na putanji /usr/lib/apache2/modules/, a njihove datoteke unutar /etc/apache2/ direktorija su zapravo konfiguracijske datoteke koje sadrže upute za njihovo učitavanje. Pomoću njih apache2 pri pokretanju pokreće i potrebne module.

Jednostavnim prebacivanjem datoteke iz direktorija mods-available u mods-enabled i resetiranjem servisa se omogućuje određeni modul. Dodatno, ako se želi instalirati dodatni modul, njegova se datoteka preuzima i stavlja u određeni direktorij na poslužitelju. Kreira se konfiguracijska datoteka s putanjom do modula i koristeći LoadModule naredbu daje se uputa za učitavanje tog modula. Ispod je prikazan primjer omogućivanja trenutno onemogućenog modula lua.conf. Najprije se provjeravaju omogućeni moduli koristeći apache2ctl alat, a zatim se modul omogućava i provjerava njegova prisutnost. *(Tri točkice označavaju da je irelevantan sadržaj ispisa skraćen.)*

```
debian@debian10:/$ sudo apache2ctl -M
Loaded Modules:
 …
 deflate_module (shared)
 dir_module (shared)
 env_module (shared)
 filter_module (shared)
 mime_module (shared)
 mpm_event_module (shared)
 …
debian@debian10:/$ sudo mv /etc/apache2/mods-available/lua.load /etc/apache2/mods-enabled/
debian@debian10:/$ sudo systemctl restart apache2
debian@debian10:/$ sudo apache2ctl -M
```

```
Loaded Modules:
 …
 deflate_module (shared)
 dir_module (shared)
 env_module (shared)
 filter_module (shared)
 lua_module (shared)
 mime_module (shared)
 mpm_event_module (shared)
 …
```

Dodatno, ovo je sadržaj lua.load datoteke:

```
debian@debian10:/$ cat /etc/apache2/mods-enabled/lua.load
LoadModule lua_module /usr/lib/apache2/modules/mod_lua.so
```

Specifično za apache2 poslužitelj, postoje alati a2enmod za omogućavanje modula te a2dismod za onemogućavanje modula koji se mogu koristiti nakon što se modul instalira za njegovo upravljanje umjesto pisanja LoadModule naredbi u konfiguracijske datoteke. Potpuno brisanje modula može se uraditi brisanjem modula iz predefinirane lokacije za apache2 module, a to je /usr/lib/apache2/modules/.

Za instaliranje i brisanje apache2 modula također je moguće koristiti apt upravitelj paketa kao što je prikazano ispod:

```
$ sudo apt search libapache2-mod- # Ubuntu and Debian
[sudo] password for user:
Full Text Search... Done
libapache-mod-jk-doc/jammy/en 1:1.2.46-1 all
  Documentation of libapache2-mod-jk package

libapache2-mod-apparmor/jammy/en 2.13.3-5ubuntu1 amd64
  changehat AppArmor library as an Apache module

--
sudo apt install --assume-yes libapache2-mod-security2
```

Budući da se isti princip primjenjuje na nginx poslužitelj, u primjeru ispod nije prikazan detaljan postupak omogućavanja nego samo sadržaj direktorije modules-enabled. Razlika je to što se u datotekama modula kod apache2 koristi LoadModule. Za nginx je naziv drukčiji, ali primjena i svrha su identične, kao što se može vidjeti iz primjera ispod.

```
debian@debian10:/$ ls /etc/nginx/modules-enabled/
50-mod-http-auth-pam.conf  50-mod-http-geoip.conf          50-mod-http-upstream-fair.conf  50-mod-stream.conf
50-mod-http-dav-ext.conf    50-mod-http-image-filter.conf   50-mod-http-xslt-filter.conf
50-mod-http-echo.conf       50-mod-http-subs-filter.conf    50-mod-mail.conf
debian@debian10:/$   cat   /etc/nginx/modules-enabled/50-mod-http-auth-pam.conf
load_module modules/ngx_http_auth_pam_module.so;
```

7.3. Praćenje zapisa (eng. *Monitoring logs*)

Nakon instaliranja poslužitelja, predefinirano je osnovno zapisivanje aktivnosti. Budući da je u prethodnim poglavljima ovog rada izražen značaj praćenja zapisa zbog uočavanja anomalija u ponašanju, neautoriziranih pristupa i sl., bitno je napomenuti da i ovi poslužitelji imaju takvu funkcionalnost. Zapisi se nalaze na putanji /var/log/nginx za nginx poslužitelj te /var/log/apache2 za apache poslužitelj. Uzevši u obzir sadržaj poglavlja u kojem je obrađen rsyslog, vidljivo je da i ova dva poslužitelja svoje zapise spremaju na predefinirano mjesto za takve dokumente (/var/log). Ispod je radi primjera prikazan sadržaj zapisa koji čuvaju podatke o pristupima za oba poslužitelja.

```
debian@debian10:~$ sudo tail -n 3 /var/log/nginx/access.log
192.168.0.20 - - [15/Aug/2022:21:48:12 +0200] "GET /test.html HTTP/1.1" 404 199 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/104.0.0.0 Safari/537.36"
192.168.0.20 - - [15/Aug/2022:21:48:13 +0200] "GET /test.html HTTP/1.1" 404 199 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/104.0.0.0 Safari/537.36"
192.168.0.20 - - [15/Aug/2022:21:48:21 +0200] "GET /test.html HTTP/1.1" 404 193 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/104.0.0.0 Safari/537.36"
debian@debian10:~$ sudo tail -n 3 /var/log/apache2/access.log
192.168.0.20 - - [15/Aug/2022:22:15:06 +0200] "GET / HTTP/1.1" 403 400 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/104.0.0.0 Safari/537.36"
192.168.0.20 - - [15/Aug/2022:22:15:55 +0200] "GET / HTTP/1.1" 200 3364 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/104.0.0.0 Safari/537.36"
```

```
192.168.0.20 - - [15/Aug/2022:22:15:55 +0200] "GET /icons/openlogo-75.png HTTP/1.1" 200 6024 "http://192.168.0.25/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/104.0.0.0 Safari/537.36"
```

Dodatno je moguće uključiti druge module koji bi zapisivali detaljnije, odnosno specifičnije podatke o aktivnostima, ako je to potrebno. Osim toga, moguće je i ove zapise preusmjeravati na centralizirani poslužitelj o čemu je bilo riječi ranije.

7.4. Implementacija TLS za HTTPS protokol

Potrebno je kreirati certifikate koji će omogućiti korištenje TLS. Moguće je kreirati vlastito potpisane certifikate (eng. *Self-signed certificates*). Oni se mogu koristiti u svrhu testiranja. U produkciji takvi certifikati predstavljaju rizik jer iza njih ne postoji službeni izdavač certifikata (eng. *Certificate Authority*) koji garantira njihovu vrijednost. Zbog toga će web preglednik prije otvaranja stranice koja koristi vlastito potpisane certifikate izbaciti poruku da je stranica nesigurna i neće ju otvoriti dok korisnik to izričito ne uradi. Najpopularniji alat za ovu vrstu certifikata je openssl. U sljedećem primjeru kreirani su certifikati koji će se koristiti za web poslužitelj. Prikazan je proces kreiranja da bi se u kasnijim snimkama zaslona moglo vidjeti da su na poslužitelju upravo certifikati s atributima definiranim u ovom procesu.

```
debian@debian10:~$ sudo mkdir /root/certs
debian@debian10:~$ sudo su
root@debian10:/home/debian# cd /root/certs
root@debian10:~/certs# openssl req -new -newkey rsa:4096 -x509 -sha256 -days 365 -nodes -out MojCertifikat.crt -keyout MojKljuc.key
Generating a RSA private key
.................................++++
..........................................................................................................................................................................................................................................++++
writing new private key to 'MojKljuc.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
-----
Country Name (2 letter code) [AU]:HR
State or Province Name (full name) [Some-State]:Varazdinska
```

```
Locality Name (eg, city) []:Varazdin
Organization Name (eg, company) [Internet Widgits Pty Ltd]:FOI_student
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:amatic@student.foi.hr
root@debian10:~/certs# ls
MojCertifikat.crt  MojKljuc.key
root@debian10:~/certs# chmod 400 ./MojKljuc.key
```

Ovime su kreirani certifikati koje će web poslužitelj koristiti. Ispod su objašnjeni dijelovi korištene naredbe openssl alata:

- -newkey rsa:4096 kreira 4096 bitni RSA ključ koji će se koristiti za certifikat.
- -x509 kreira vlastito potpisani certifikat
- -sha256 generira zahtjev za certifikatom koristeći sha-256 algoritam sažimanja
- -days označava koliko dana će certifikat vrijediti
- -nodes kreira certifikat za koji ne treba passphrase, odnosno svojevrsna lozinka koja se treba unijeti pri rukovanju s certifikatom. Ako se ova opcija izostavi, bit će potrebno unijeti passphrase svaki put kada se aplikacija ponovno pokrene.

Na kraju, potrebno je napraviti pohranu ključa i certifikata na vanjsku pohranu.

Konfiguracija HTTPS protokola na Nginx poslužitelju

U konfiguracijskoj datoteci potrebno je dodati novi server blok koji će koristiti TLS. U njega se upisuju upute za lokaciju certifikata i ključa te za korištenje određenih TLS verzija protokola [27]:

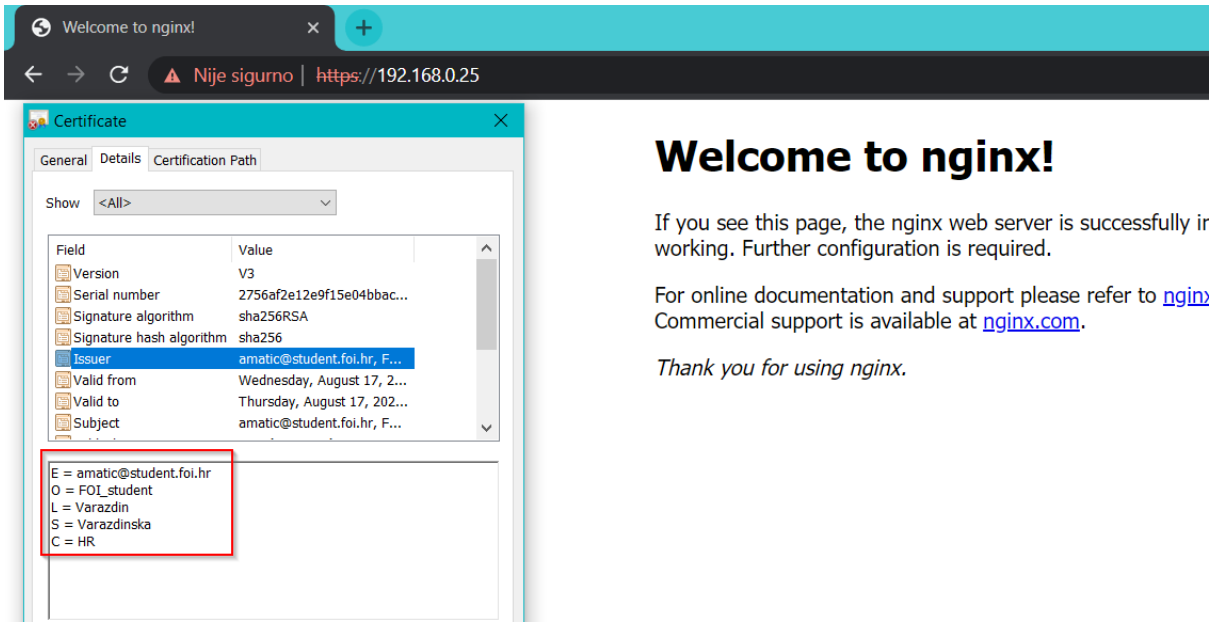
```
server {
                listen 192.168.0.25:443 ssl;
                ssl_certificate /root/certs/MojCertifikat.crt;
                ssl_certificate_key /root/certs/MojKljuc.key;
                ssl_protocols   TLSv1 TLSv1.1 TLSv1.2;
        }
```

U drugi server blok koji je kreiran ranije i koristi se za „običan“, tj. nešifrirani promet na priključku 80, dodaje se redak u kojemu se promet prema http poslužitelju preusmjerava na poslužitelj koji osluškuje na priključku 443 kako bi se osigurala enkripcija prometa [27].

Taj blok izgleda ovako:

```
server {
                if ($blockedagent){
                        return 403;
                }
                listen 192.168.0.25:80;
                return 301 https://$host$request_uri;
        }
```

Upisujući IP adresu u web preglednik, poslužitelj usmjerava promet na 443 priključak i koristi TLS. Budući da je certifikat vlastito potpisan, preglednik javlja da stranica nije sigurna jer certifikat nije važeći, međutim stranica se može posjetiti kao i vidjeti detalji o certifikatu koji su ranije uneseni.



Slika 11 - Certifikati Nginx poslužitelja [autorski rad]

Konfiguracija HTTPS protokola na Apache poslužitelju

U ovim primjerima će biti korišteni ranije kreirani certifikati.

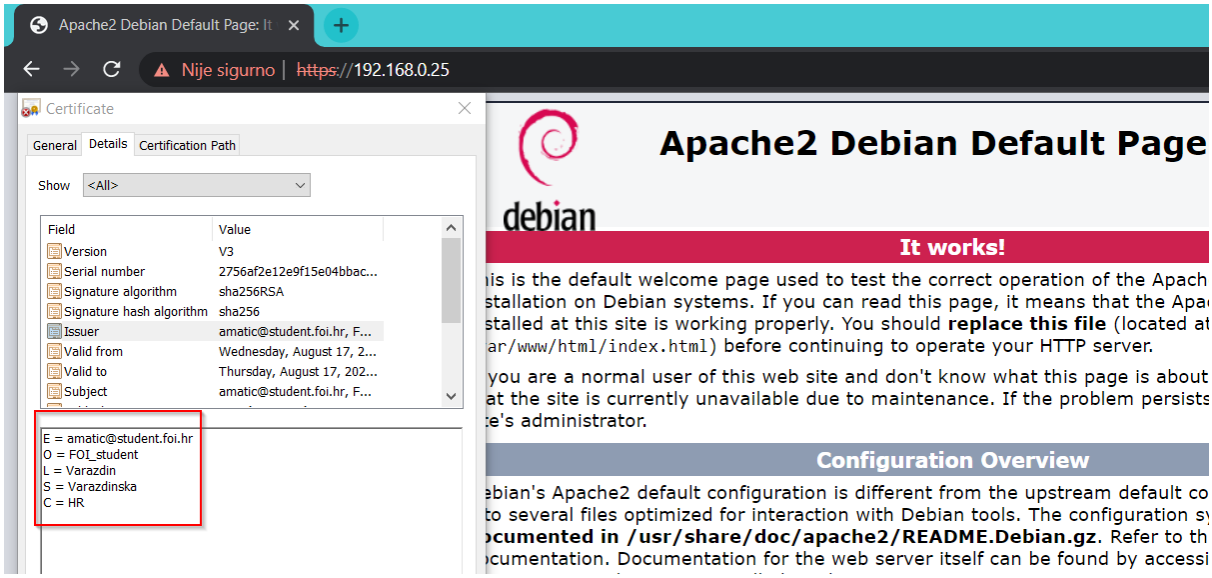
Potrebno je omogućiti ssl modul korištenjem naredbe:

```
Sudo a2enmod ssl
```

Nakon toga u konfiguracijskoj datoteci kreira se novi blok za virtualni poslužitelj (VirtualHost block) slično kao server blok kod nginx poslužitelja. Taj blok izgleda ovako:

```
<VirtualHost *:443>
        SSLEngine on
        SSLCertificateFile /root/certs/MojCertifikat.crt
        SSLCertificateKeyFile /root/certs/MojKljuc.key
        ErrorLog ${APACHE_LOG_DIR}/sslerror.log
        CustomLog ${APACHE_LOG_DIR}/sslaccess.log combined
</VirtualHost>
```

Može se primijetiti da su unutar ovog bloka definirane i lokacije datoteka za zapisivanje aktivnosti. Nakon ovoga, ponovno se pokreće servis apache2 i pri posjetu stranici dobije se ista poruka kao i ranije kod nginx poslužitelja, zajedno s istim certifikatom što se može vidjeti na slici ispod.



Slika 12 - Certifikati Apache poslužitelja [autorski rad]

7.5. Blokiranje korisničkih agenata (eng. *User Agents*)

User agent je pojam koji označava softver koji se koristi prilikom uspostavljanja veze s web poslužiteljem. Njega koriste i automatizirana računala (eng. *Bots*) koji mogu biti maliciozni i stvarati bespotrebni promet na poslužitelju kojim ga opterećuju [27]. Često se na osnovu agenta može reći radi li se o legitimnom klijentu ili nekom automatiziranom malicioznom softveru. Zbog toga se web poslužitelj može dodatno osigurati filtriranjem tih agenata, odnosno ne dozvoljavajući promet paketima koji koriste korisničkog agenta koji nije

standardiziran i uobičajen kao što je npr. Mozilla/5.001, korisnički agent kojeg koristi web preglednik Mozilla (Firefox).

Napomena: primjer blokiranja korisničkih agenata za Nginx poslužitelj kreiran je uz pomoć uputa autora G. Canepa [27].

Da bi se kod poslužitelja nginx lakše kreiralo takvo pravilo, moguće je napraviti datoteku s popisom formata naziva korisničkih agenata kojim nginx ne treba dozvoliti promet. Npr. To može biti datoteka /etc/nginx/blockuseragents.rules, sljedećeg sadržaja:

```
map $http_user_agent $blockedagent {
        default         0;
        ~*malicious     1;
        ~*bot           1;
        ~*backdoor      1;
        ~*crawler       1;
        ~*bandit        1;
}
```

Takvu datoteku treba uključiti u konfiguraciju poslužitelja postavljajući sljedeći redak unutar konfiguracijske datoteke prije bloka parametara za server.

```
Include /etc/nginx/blockuseragents.rules;
```

Također je moguće definirati ponašanje kada nginx otkrije da se promet upućuje koristeći neki od nedozvoljenih agenata. To se omogućuje postavljanjem jednostavne if selekcije unutar server bloka. If selekcija izgleda ovako, a daje uputu da se vrati HTTP kod 403 Forbidden kada se uoči nedozvoljeni agent:

```
If ($blockedagent){ return 403; }
```

Sada se ponašanje može testirati korištenjem *wget* alata. Korištenjem agenta koji odgovara kriterijima zabrane, dobije se 403 odgovor, a u protivnom 200 OK kao što bi i trebalo biti prema napravljenoj konfiguraciji.

```
debian@debian10:~$ wget http://192.168.0.25/index.html --user-agent
"bot"
--2022-08-16 22:18:39--  http://192.168.0.25/index.html
Connecting to 192.168.0.25:80... connected.
HTTP request sent, awaiting response... 403 Forbidden
2022-08-16 22:18:39 ERROR 403: Forbidden.

debian@debian10:~$ wget http://192.168.0.25/index.html
--2022-08-16 22:18:49--  http://192.168.0.25/index.html
Connecting to 192.168.0.25:80... connected.
```

```
HTTP request sent, awaiting response... 200 OK
Length: 612 [text/html]
Saving to: ‘index.html.3’

index.html.3
100%[=======================================>]     612  --.-KB/s    in
0s

2022-08-16 22:18:49 (57.4 MB/s) - ‘index.html.3’ saved [612/612]
```

Sljedeći primjer za apache poslužitelj služi istoj svrsi, a napravljen je uz pomoć uputa na Techexpert.tips web stranici [28]:

Dakle, isto se može napraviti i na apache poslužitelju. U konfiguracijsku datoteku na putanji /etc/apache2/sites-enabled/000-default.conf upisuju se sljedeći retci unutar VirtualHost bloka.

```
RewriteEngine On
RewriteCond %{HTTP_USER_AGENT} (yandex|bingbot) [NC]
RewriteRule .* - [F,L]
```

Agenti se mogu mijenjati prema potrebama, ovdje su dva nedozvoljena agenta yandex i bingbot. Nakon resetiranja servisa, promjene se primjenjuju i može se provesti testiranje, kao što je prikazano ispod:

```
debian@debian10:~$ wget http://192.168.0.25/index.html --user-agent
"bingbot"
--2022-08-16 22:25:28--  http://192.168.0.25/index.html
Connecting to 192.168.0.25:80... connected.
HTTP request sent, awaiting response... 403 Forbidden
2022-08-16 22:25:28 ERROR 403: Forbidden.

debian@debian10:~$ wget http://192.168.0.25/index.html
--2022-08-16 22:25:36--  http://192.168.0.25/index.html
Connecting to 192.168.0.25:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10701 (10K) [text/html]
Saving to: ‘index.html.4’

index.html.4
100%[=======================================>]  10.45K  --.-KB/s    in
0s

2022-08-16 22:25:36 (122 MB/s) - ‘index.html.4’ saved [10701/10701]
```

7.6. Gašenje izlistavanja direktorija (Apache poslužitelj)

Pod pojmom izlistavanja direktorija (eng. *directory listing*) podrazumijeva se na pristup i prikaz strukture nekog direktorija pod putanjom prikazana kroz sučelje web preglednika. Primjer izlistavanja direktorija može se vidjeti na slici ispod.

Slika 13 - Izlistavanje direktorija omogućeno [autorski rad]

Ovime je omogućeno i preuzimanje prikazanih datoteka, otvaranje prikazanih direktorija jednostavnim klikom na njihovo ime u popisu koje je zapravo poveznica. Iz dosadašnje rasprave o različitim metodama sigurnosnog očvršćavanja i različitih napada, može se zaključiti da ovo predstavlja veliki sigurnosni propust ako se ne održava po strogim pravilima. Napadač bi ovime mogao pristupiti datoteki s lozinkama korisnika ili drugim osjetljivim podacima. Zbog toga je u većini slučajeva preporuka isključiti ovu opciju. Primjer toga bit će prikazan za Apache poslužitelj. Prema autoru T. Shrivastava [25], sve što je potrebno učiniti za onemogućivanje ove opcije jest unos sljedećeg retka u konfiguracijsku datoteku:

```
Options -Indexes
```

Potrebno je otvoriti konfiguracijsku datoteku u uređivaču teksta s povišenim privilegijama:

```
debian@debian10:~$ sudo nano /etc/apache2/apache2.conf
```

Nakon toga, u konfiguracijsku datoteku za željeni direktorij u kojemu je potrebno isključiti opciju izlistavanja direktorija unosi se spomenuti redak. Postavljajući taj redak u

/var/www/html putanju, onemogućeno je izlistavanje direktorija za cijeli korijenski direktorij Apache poslužitelja. Taj blok u datoteci izgleda ovako:

```
<Directory /var/www/html>
      Options -Indexes
</Directory>
```

Nakon spremljenih promjena, potrebno je ponovno pokrenuti servis Apache poslužitelja kako bi promjene bile primijenjene.

```
debian@debian10:~$ sudo systemctl restart apache2
```

Na slici ispod prikazan je odgovor poslužitelja kada je zahtjev pristup istom direktoriju kao na prethodnoj slici, ali s onemogućenim izlistavanjem direktorija.

Slika 14 - Izlistavanje direktorija omogućeno [autorski rad]

8. Zaključak

Ovome je radu svrha bila prikazati načine sigurnosnog očvršćivanja GNU/Linux poslužitelja, a time i objasniti temeljne principe informacijske sigurnosti, zašto je ona važna u današnjici, objasniti najkorištenije metode za sigurnosno očvršćivanje sustava i obranu od napada te prikazati njihovu praktičnu implementaciju zajedno s prikazom rezultata njihove primjene. Da bi to bilo omogućeno, korištena je literatura različitih tvrtki orijentiranih na sigurnost te autora iz područja informacijske sigurnosti. Za potrebe izvedbe praktičnog dijela rada kreirano je lokalno virtualno okruženje u kojemu je postavljen virtualni Debian GNU/Linux poslužitelj. Taj je poslužitelj korišten kao osnova za sve primjere u ovome radu. Za prikaz rezultata određenih postavki koje se tiču komunikacije s poslužiteljem korišten je dodatni Windows sustav kako bi se simulirao stvarni svijet u kojemu je još uvijek većina poslužitelja na linux operacijskom sustavu, a većinu korisničkih računala pokreće Windows operacijski sustav.

Prvi dio rada obuhvaća teoriju o informacijskoj sigurnosti te njenim glavnim principima koji se nastoje ostvariti različitim metodama i tehnikama. Osim toga, obrađene su i opće upute kojima se nastoji poboljšati kvaliteta provođenja procesa sigurnosnog očvršćivanja.

Drugi dio prikazuje praktičnu implementaciju prethodno obrađenih metoda uz njihovo dodatno tumačenje te prikaz rezultata njihove primjene. Provedeno je sigurnosno očvršćivanje Linux operacijskog sustava, sesija udaljenog upravljanja sustavom te poslužitelja baza podataka i web poslužitelja. Kroz praktične primjere prikazane su metode kojima se osiguravaju povjerljivost, integritet te dostupnost informacija, sustava i mreža. Takve metode su redovno i oprezno ažuriranje softvera i servisa, upravljanje računima i njihovim lozinkama, nadzor sustava korištenjem *logova*, očvršćivanje sesija za udaljeno upravljanje sustavom, enkripciju podataka u prijenosu, rezidentnih podataka, uklanjanje nepotrebnog softvera, modula, izoliranje mreže i njenih dijelova korištenjem pravila vatroštita. Te su metode zajedničke većini poslužitelja, softvera i drugih elemenata čija se sigurnost treba očvrsnuti, a prikazane su i druge metode specifične za određene sustave, kao što je dodjeljivanje uloga u poslužiteljima baza podataka, blokiranje prometa određenih korisničkih agenata prema web poslužiteljima itd.

Svim spomenutim metodama postiže se smanjena bespotrebna izloženost podataka, smanjena površina napada, manji utjecaj eventualnog napada na sustav, a u konačnici postižu se povjerljivost, integritet te dostupnost informacija i sustava.

Popis literature

[1] Sigurnosno-obavještajna agencija RH (SOA), „Informacijska sigurnost“, n.d. https://www.soa.hr/hr/podrucja-rada/informacijska-sigurnost/ (pristupljeno 12. kolovoz 2022.).

[2] R. Brooks, „The CIA Triad and Real-World Examples“, *The CIA triad and its real world application*, n.d. https://blog.netwrix.com/2019/03/26/the-cia-triad-and-its-real-world-application/ (pristupljeno 12. kolovoz 2022.).

[3] M. D. Bauer, *Linux Server Security*. Sebastopol etc.: O'Reilly., 2005.

[4] K. A. Scarfone, W. Jansen, i M. Tracy, „Guide to general server security“, National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 800-123, 2008. doi: 10.6028/NIST.SP.800-123.

[5] CERT.hr, „Osnovno sigurnosno ojačavanje Linux poslužitelja“, *CERT.hr* -, n.d. https://www.cert.hr/sigurnosno-ojacavanje-linux/ (pristupljeno 13. kolovoz 2022.).

[6] Netwrix, „Linux Server Hardening & Security Best Practices“. Netwrix, n. d.

[7] O. Zlotnik, „System Hardening Guidelines for 2022: Critical Best Practices“, *Hysolate*, 21. rujan 2021. https://www.hysolate.com/blog/system-hardening-guidelines-best-practices/ (pristupljeno 13. kolovoz 2022.).

[8] „systemd(1) - Linux manual page“, n.d. https://www.man7.org/linux/man-pages/man1/systemd.1.html (pristupljeno 13. kolovoz 2022.).

[9] „Chapter 9. Keeping your Debian system up-to-date“, n.d. https://www.debian.org/doc/manuals/debian-faq/uptodate.en.html (pristupljeno 23. kolovoz 2022.).

[10] „rsyslog - Gentoo Wiki“, *Gentoo*, n.d. https://wiki.gentoo.org/wiki/Rsyslog (pristupljeno 27. srpanj 2022.).

[11] „logger(1) - Linux manual page“, n.d. https://man7.org/linux/man-pages/man1/logger.1.html (pristupljeno 17. srpanj 2022.).

[12] D. A. Tevault, *Mastering Linux security and hardening: secure your Linux server and protect it from intruders, malware attacks, and other external threats*. Birmingham Mumbai: Packt, 2018.

[13] R. Natarajan, „Linux Firewall Tutorial: IPTables Tables, Chains, Rules Fundamentals“, 24. siječanj 2011. https://www.thegeekstuff.com/2011/01/iptables-fundamentals/ (pristupljeno 24. kolovoz 2022.).

[14] „iptables(8) - Linux manual page“, n.d. https://www.man7.org/linux/man-pages/man8/iptables.8.html (pristupljeno 24. kolovoz 2022.).

[15] „RSA Algorithm in Cryptography - GeeksforGeeks“, 13. lipanj 2022. https://www.geeksforgeeks.org/rsa-algorithm-cryptography/ (pristupljeno 18. kolovoz 2022.).

[16] T. Bažant *i ostali*, „Security and Hardening Guide - SUSE Linux Enterprise Server 15 SP1“. n.d.

[17] D. Page, „How to Secure PostgreSQL: Security Hardening Best Practices & Tips“, *EDB*, n.d. https://www.enterprisedb.com/blog/how-to-secure-postgresql-security-hardening-best-practices-checklist-tips-encryption-authentication-vulnerabilities (pristupljeno 16. kolovoz 2022.).

[18] M. Frank, „MySQL :: MySQL Guide to Ports“, n.d. https://dev.mysql.com/blog-archive/mysql-guide-to-ports/ (pristupljeno 16. kolovoz 2022.).

[19] „The Ultimate Guide To MySQL Roles By Examples“, *MySQL Tutorial*, n.d. https://www.mysqltutorial.org/mysql-roles/ (pristupljeno 25. kolovoz 2022.).

[20] „Password strength requirements for MySQL, EDB PostgreSQL, and Oracle database accounts“, n.d. https://kb.iu.edu/d/aphg#oracle (pristupljeno 25. kolovoz 2022.).

[21] „What Are Digital Certificates?“, *Fortinet*, n.d. https://www.fortinet.com/resources/cyberglossary/digital-certificates (pristupljeno 25. kolovoz 2022.).
[22] K. Rankin, „5 Essential Steps to Hardening Your MySQL Database“, *Linode Cube*, 25. kolovoz 2017. https://medium.com/linode-cube/5-essential-steps-to-hardening-your-mysql-database-591e477bbbd7 (pristupljeno 25. kolovoz 2022.).
[23] Agustín, „Enabling and Enforcing SSL/TLS for PostgreSQL Connections“, *Percona Database Performance Blog*, 05. srpanj 2022. https://www.percona.com/blog/enabling-and-enforcing-ssl-tls-for-postgresql-connections/ (pristupljeno 25. kolovoz 2022.).
[24] MarkGreisiger, „Safeguarding Data: Encryption, Tokenization and Hashing“, *NetDiligence*, 21. siječanj 2013. https://netdiligence.com/blog/2013/01/safeguarding-data-encryption-tokenization-and-hashing/ (pristupljeno 25. kolovoz 2022.).
[25] T. Shrivastava, „13 Apache Web Server Security and Hardening Tips“, 15. listopad 2013. https://www.tecmint.com/apache-security-tips/ (pristupljeno 25. kolovoz 2022.).
[26] Rick, „Get a List of All Static and Dynamic/Shared Modules Available and Enabled in Apache HTTP Server“, *CodingShower*, 27. travanj 2020. https://codingshower.com/list-static-and-dynamic-modules-in-apache-http-server/ (pristupljeno 25. kolovoz 2022.).
[27] G. Cánepa, „The Ultimate Guide to Secure, Harden and Improve Performance of Nginx Web Server“, 21. prosinac 2015. https://www.tecmint.com/nginx-web-server-security-hardening-and-performance-tips/ (pristupljeno 25. kolovoz 2022.).
[28] V. C. LPIC2 PMP, CCNP, MCSE, „Tutorial Apache - Blocking the USER-AGENT access [Step by step]“, *TechExpert*, 03. travanj 2021. https://techexpert.tips/apache/apache-blocking-user-agent-access/ (pristupljeno 25. kolovoz 2022.).

Popis slika