Ostvarivanje lokalna posluživa̧ca za siguran daljinski rad preko protokola SSH

Lacković, Dominik

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Dominik Lacković

Ostvarivanje lokalna poslužitelja za siguran daljinski rad preko protokola SSH

ZAVRŠNI RAD

Varaždin, 2024.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Dominik Lacković

Matični broj: 48104/20–R

Studij: Primjena informacijske tehnologije u poslovanju

Ostvarivanje lokalna poslužiteljskih računala za siguran daljinski rad preko protokola SSH

ZAVRŠNI RAD

Mentor:

Dr. sc. Luka Milić

Varaždin, rujan 2024.

Dominik Lacković

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojega rada i da se u njegovoj izradbi nisam služio drugim izvorima osim onima koji su u njem navedeni. Za izradbu su rada rabljene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredaba u sustavu FOI-radovi

Sažetak

U radu će se istražiti načini ostvarivanja lokalnih poslužitelja u mrežama za siguran daljinski rad preko protokola SSH i opisat će se sami načini daljinskoga rada korisnika na tim poslužiteljima. Rad se usredotočuje na stvaranje prividnoga stroja u programu VirtualBox s operacijskim sustavom Ubuntu i na konfiguraciju sustava tako da omogućuje daljinski rad svih korisnika preko protokola SSH rabeći sigurnu metodu autentifikacije za administratorske korisnike koja se temelji na parovima javnih i privatnih ključeva za asimetričnu enkripciju. Isto tako, istražuje se kako osigurati vlastita pristupna prava za svakoga korisnika i skupinu korisnika. Rad ispituje i uspoređuje djelovanje poslužitelja preko SSH-klijenata kao što su PuTTY, Termius, Tabby Terminal i MobaXTerm. Cilj je rada razumjeti postupak instaliranja i konfiguriranja poslužitelja za siguran daljinski rad preko protokola SSH.

Ključne riječi: poslužitelj; SSH; SSH-klijenti; daljinski rad, prosljeđivanje vrata; Dynamic DNS; lokalne mreže; prividni stroj; pristupna prava

Kazalo

1. Uvod

U današnja digitalna doba, gdje se obavijesti i podatci neprestano razmjenjuju preko Interneta, pitanje sigurnosti i daljinskoga pristupa računalima i posluživačima postaje ključno za brojne organizacije i pojedince. U ovom se radu istražuje bitnost i ostvarivanje lokalnoga posluživača za siguran daljinski rad, usredotočujući se pri tom na protokol SSH (Secure Shell).

S obzirom na svenazočnu potrebu za pristupom daljinskim sredstvima, daljinsko je upravljanje koje bilo vrste računala i računalnih sustava postalo neizbježiv dio svakodnevnoga poslovanja malenih, ali i velikih organizacija. Ali, s povećavanjem ovisnosti o daljinskom radu, sigurnost postaje ključna briga. U tom smislu, SSH se nameće kao norma za siguran pristup, upravljanje i prijenos podataka na daljinskim sustavima. Odabir ove teme bio je od iznimne važnosti kako bi se znanje proširilo novim praktičnim vještinama koje se u poslovnom svijetu mogu vrlo lako primijeniti u skoro pa kojoj bilo vrsti posla.

Cilj ovoga rada je pružiti dublje razumijevanje SSH-protokola kroz ostvarivanje lokalna posluživača na koji će se korisnici izvan lokalne mreže moći spojiti preko protokola SSH. Kroz istraživanje različnih vidova sigurnosti i praktičnih primjena rad će ponuditi smjernice za uspostavu sigurna daljinskoga pristupa, pregled nekoliko znamenitih SSH-klijenata kao što su PuTTY, Termius, Tabby Terminal i MobaXTerm ter pregled i konfiguraciju SSH-posluživača i klijenata.

Za postizanje će se navedenoga cilja rabiti prividni stroj s operacijskim sustavom Ubuntu Server inačice 22.04 LTS (Long Term Support) kako je za zakup fizičkoga posluživača potreban određen iznos novca. Pristup u ovom radu bit će sveobuhvatan, uključujući tehničke vidove konfiguracije, sigurnosne protokole ter će se po tanko opisati mogući izazovi pri ostvarivanju lokalnoga posluživača i rada preko SSH-klijenata. Predstavit će se i alternativne rješidbe jer često ne postoji samo jedna rješidba koje će odgovarati svim slučajevima.

Rad je građen u nekoliko glavnih dijelova. Nakon uvoda slijedi teorijski pregled protokola SSH, istraživanje sigurnosnih vidova za rad preko SSH-protokola i raščlamba postojećih rješidaba. Za tim slijedi praktični dio, gdje će se korak po korak voditi kroz ostvarivanje lokalnoga posluživača. Na kraju će zaključak sažeti rad i nove spoznaje ter će se istaknuti prinosi rada.

2. Uobičajene prakse

Ostvariti poslužitelj u lokalnim mrežama moguće je na različne načine, a sam odabir metode ostvaraja uvijek se svađa na potrebe korisnika i laku mogućnost naknadne administracije koja prije svega omogućuje sigurnost podataka i aktivnih veza za sve korisnike. Na Fakultetu organizacije i informatike ostvaren je fizički poslužitelj koji rabe i profesori i slušači. Na istom poslužitelju postoji nekoliko usluga od kojih su neke javno raspoložive, a druge ne. FTP-poslužitelj raspoloživ je isključivo na lokalnoj mreži fakulteta, a pristup je omogućen jedino profesorima kako bi dijelili nastavne materijale. Za slušače tako postoji SSH-poslužitelj na koji se spajaju preko naslova *barka.foi.hr* svojim korisničkim računom koji im je dodijeljen kod upisa na fakultet. SSH-poslužitelj rabi se u nastavničke svrhe kako bi slušači mogli rješavati zadatke i preglede znanja iz različnih informatičkih kolegija ili pak prevesti vlastiti kod i steći dublje znanje rada operacijskih sustava. Trenutačni je nedostatak ovoga poslužitelja taj da korisnici s najmanjim pravima – slušači – mogu vidjeti što imade u kazalima drugih slušača pa tako i prepisati rješidbe za određene preglede znanja ter će se ta teškoća kod ovoga ostvaraja riješiti pravilnim dodjeljivanjem pristupnih prava za određene korisničke skupine. Iako je sam način ostvaraja SSH-poslužitelja *barka.foi.hr* neznan, ovako velik sustav s velikim brojem slušača ostvario bi se s pomoću prilagodljivih PAM-modula koji bi se povezali s već postojećim LDAP-poslužiteljem čim se samo naglašuje kako je SSH iznimno prilagodljiv i moguće ga je povezati s različnim sustavima autentifikacije ako je to potrebno.

2.1. Ostvarivanje poslužitelja u lokalnim mrežama

Poslužitelji u lokalnim mrežama obično se ostvaruju s pomoću virtualizacijskih oruđa kao što su VirtualBox ili VMware jer oni smanjuju operacijske troškove i troškove samoga sklopovlja omogućivanjem puno prividnih strojeva za različne službe unutar samoga fizičkoga računala dotično poslužitelja. Takov pristup omogućuje opruživost u upravljanju različnim okružjima i izolaciju poslužitelja od ostatka sustava. Kad je riječ o postavljanju kojega bilo poslužitelja, jedan je od najbitnijih koraka pravilna konfiguracija mrežnih postavaka unutar prividnoga stroja kako bi se omogućio pristup s drugih uređaja unutar lokalne mreže. U tom smislu poraba premoštenoga mrežnoga prilagodnika u VirtualBoxu omogućuje da prividni stroj dobije IP-adresu unutar mreže kao da je fizički uređaj i tako se olakšava daljinski pristup i omogućuje iznimna opruživost oko samih konfiguracija. Osim virtualizacije se poslužitelji isto tako mogu ostvariti na fizičkim računalima u lokalnoj mreži. Prigodom instalacije takova sustava bitno je osigurati da je računalo pravilno uklopljeno u mrežnu infrastrukturu. To uključuje postavljanje statične IP-adrese i odgovarajuće mrežne maske. Na taj način

poslužitelj postaje lako raspoloživ svim korisnicima unutar mreže preko SSH-protokola ako je instaliran i konfiguriran SSH-poslužitelj. Da bi se osigurala veća sigurnost i bolji nadzor pristupa, često se rabi VLAN-segmentacija koja omogućuje odvajanje mrežnoga prometa poslužitelja od ostalih dijelova mreže [1, 2].

U današnja doba iznimno su znameniti i poslužitelji u oblaku koji pružaju neviđenu opružitelj i skalabilnost u vrlo kratku razdoblju (AWS, Azure) ili pak različne kombinacije poslužitelja u oblaku i lokalnih poslužitelja ako je potreba da se kritični podatci i usluge čuvaju lokalno zbog rigoroznih sigurnosnih certifikata.

Isto su tako znamenita i računala malenih protega kao što su Raspberry Pi koja su se pokazala kao izvrsna ekonomična i energijski učinkovita rješidba za manje mreže i projekte. Ako je skaliranje i upravljanje sredstvima od iznimne bitnosti, isto su tako raspoloživi i izbori za *containere* kao što su Docker i Kubernetes za iznimno lako stvaranje i upravljanje aplikacijama u različnim *containerima*.

2.2. SSH-protokol

SSH-protokol nastao je kao nadogradnja Telnet-protokola koji je počeo s razvojem još davne 1969. godine. Za razliku od Telneta, SSH-protokol siguran je protokol za daljinsku komunikaciju jer rabi enkripciju svih podataka koji se šalju ili primaju, a ne obličje „*plain text*“. Radi na načelu klijent-poslužitelj što znamenuje da se klijent s pomoću kojega od ostvaraja SSH-protokola spaja na daljinsko računalo na kojem je pokrenut SSH-poslužitelj [3].

Za prijenosni se sloj po OSI-modelu rabi TCP kako bi se osigurala cjelovitost podataka. Uobičajena su vrata na kojima se SSH rabi 22 iako to mogu biti koja bilo druga vrata koja ne rabi ni jedna druga usluga [4].

SSH-protokol određuje autentifikaciju, enkripciju i cjelovitost prenesenih podataka preko mreže [4]:

- autentifikacija – pouzdano se utvrđuje nečiji identitet s pomoću različnih metoda kao što su kombinacija korisničkoga imena i zaporke, poraba asimetričnih kriptografskih parova ključeva, poraba certifikata;
- enkripcija – kriptiranje podataka tako da nisu prepoznatljivi nikomu osim namijenjenomu primatelju pače i ako dođe do presretanja internetskoga prometa;
- cjelovitost – podatci koji putuju preko Interneta dohode nepromijenjeni.

Iako je metoda autentifikacije preko kombinacije korisničkoga imena i zaporke najčešća jer je najjednostavnija, postoje i druge puno sigurnije metode autentifikacije kao što je

autentifikacija javnim i privatnim ključem. Klijent i poslužitelj razmjenjuju javne ključeve tako da klijent šalje svoj javni ključ poslužitelju, a poslužitelj rabi taj javni ključ za kriptiranje podataka koji se šalju klijentu. Tim se za pravo privatni ključevi nikad ne razmjenjuju jer klijent dekriptira te podatke s pomoću svojega privatnoga ključa [4].

Prigodom prvoga spajanja na SSH-poslužitelj koji rabi ovakovu metodu autentifikacije kod klijenta se lokalno pohranjuje javni ključ poslužitelja ter pripadajuća IP-adresa i ime poslužitelja (*hostname*). Ako bi došlo do promjene jednoga od navedenih podataka, klijent će biti upozoren na moguće sigurnosne probleme jer je narušena cjelovitost od prijašnjega spajanja [3].

Iako se SSH najčešće rabi za daljinsku administraciju dotično sigurnu prijavu na daljinsko računalo, isto se tako može rabiti i za siguran prijenos datoteka i prosljeđivanje vrata TCP/IP [4].

Za siguran se prijenos datoteka može umjesto zastarjela i nesigurna RCP-a (Remote Copy Protocol) rabiti SCP (Secure Copy Protocol) ili pak oruđe *rsync*. Stari protokol poput Telneta i nesigurna oruđa koja glavninom počinju slovom *r* (*rlogin, rsh, rcp*) u svojoj su početnoj inačici preko mreže izložili sve podatke poput korisničkoga imena i zaporke tako da je bilo tko na mreži mogao snimati promet u mreži i presresti te pakete te zlorabiti podatke za neovlašten pristup daljinskomu računalu. Zbog svih navedenih nedostataka navedena se stara oruđa danas rabe glavninom samo za ispitivanje jesu li određena vrata otvorena i za ostale zadatke gdje se ne izlažu privatni podatci [5].

2.2.1. SSH2

Postoje dvije inačice SSH-a, a to su SSH1 i SSH2. Najbitnije razlike inačice 2 u odnošaju na 1 su: unaprijeđen postupak kriptiranja i razmjene ključeva zbog porabe metode Diffie-Hellman, potpora za nove algoritme kriptiranja kao što su AES-128 i AES-256, potpora za PKI-certifikate i poboljšana cjelovitost podataka [3].

Zadnja se inačica SSH-a 2 sastoji za pravo od 3 protokola [6]:

- SSH-TRANS – protokol prijenosnoga sloja;
- SSH-AUTH – autentifikacijski protokol;
- SSH-CONN – vezni protokol.

SSH-TRANS omogućuje kriptiranu komunikaciju među klijentom i poslužiteljem preko TCP-veze. Kad korisnik rabi koju SSH-aplikaciju za prijavu na daljinski stroj, prvi je korak uspostava sigurnoga kanala SSH-TRANS među tima dvama uređajima. Klijent autentificira poslužitelj porabom RSA-algoritma, nakon čega se dogovaraju o ključu sjednice za šifriranje

podataka. Bitno je napomenuti da klijent dobiva javni ključ posluživača tijekom prve veze, a dalje ga pamti radi autentifikacije u budućnosti [6].

Nakon uspostavljanja kanala SSH-TRANS korisnik se autentificira na posluživaču rabeći jedan od triju mehanizama: slanje zaporke, porabu šifriranja javnim ključem ili autentifikaciju na temelju domaćina. Kroz ovu proceduru SSH omogućuje sigurnu komunikaciju među uređajima preko kriptiranoga kanala neovisno o mreži [6].

2.2.2. OpenSSH

OpenSSH (Open Secure Shell) jedan je od najznamenitijih i najraširenijih ostvaraja SSH-protokola koji se rabi. To je skup različnih sigurnosnih oruđa zasnovanih na SSH-protokolu, koji omogućuje sve značajke SSH-protokola pa tako i sigurno povezivanje i upravljanje daljinskim sustavima preko mreže. Projekt je dio projekta OpenBSD, a prvi je put predstavljen ne tako davne 1999. godine. Trenutačna inačica paketa OpenSSH je 9.8 s nadnevkom izdavanja 1. srpnja 2024. godine [7].

OpenSSH ključna je sastavnica u svijetu upravljanja daljinskim sustavima i mrežne sigurnosti. Njegova sposobnost pružanja sigurnoga prijenosa podataka, opruživost u konfiguraciji i snažna potpora zajednice čine ga nezamjenjivim oruđem za sustavske inženjere, mrežne inženjere i programere. Poraba OpenSSH-a znatno povećava sigurnost i učinkovitost upravljanja svakakovim IT-infrastrukturama što ga čini nezaobilaznim za svaku suvremenu organizaciju.

2.2.2.1. Značajke OpenSSH-a

S obzirom na to da je OpenSSH paket odnosno skup različnih programa i oruđa, spomenut će se samo oni najbitniji [8]:

- *ssh* – osnovni klijentski program koji zamjenjuje *rlogin* i programe nalik *rshu*;
- *sshd* – *ssh daemon/server*. Odatle i slovo *d* koje u Unixovskim operacijskim sustavima obilježava *daemone* – programe koji rade u podlozi bez potrebe za međudjelovanjem korisnika i često se pokreću prigodom pokretanja sustava. U ovom slučaju *ssh daemon* čeka na zahtjeve za vezu i obrađuje različne zadatke od korisničke autentifikacije, enkripcije, otvaranja i zatvaranja veze, prijenosa datoteka i tuneliranja;
- *ssh_config* – konfiguracijska datoteka za SSH-klijent;
- *sshd_config* – konfiguracijska datoteka za SSH-posluživač. Zbog sličnosti naziva treba paziti ne zamijeniti datoteke *ssh_config* i *sshd_config*;
- *ssh-agent* – agent za provjeru autentičnosti koji može pohraniti privatne ključeve;
- *ssh-add* – oruđe koje dodaje ključeve gore navedenomu agentu;

- *sftp* – program sličan FTP-u koji radi preko protokola SSH1 i SSH2;
- *scp* – program za preslikavanje datoteka koji se ponaša kao *rcp*;
- *ssh-keygen* – program za izrađanje privatnih i javnih ključeva;
- *sftp-server* – podsustav SFTP-posluživača (*sshd* sam od sebe pokreće);
- *ssh-keyscan* – uslužni program za prikupljanje javnih ključeva računala s više računala;
- *ssh-keysign* – pomoćni program za autentifikaciju temeljenu na domaćinu.

2.2.2.2. Ključne datoteke za rad s OpenSSH-om

Za klijent OpenSSH koji dolazi predinstaliran na glavnini Unixovskih operacijskih sustava, a od nedavno i na operacijskim sustavima Windows, za bolje je razumijevanje teme bitno spomenuti nekoliko različnih staza i datoteka gdje se koji ključevi pohranjuju u kazalo korisnika:

- *~/.ssh/known_hosts* – ovaj zapis ima u sebi javne ključeve svih domaćina (*host*) koje je korisnik prihvatio prigodom prvoga povezivanja. Prigodom daljih povezivanja pregledava je li se javni ključ za taj domaćin promijenio i ako jest dobiva se upozorba prije spajanja jer to može znamenovati da je daljinski poslužitelj kompromitiran ili je samo konfiguracija posluživača promijenjena. Na operacijskom sustavu Windows ova se datoteka nahodi u mapi korisnika: *%userprofile%**.ssh**known_hosts*. Varijabla *%userprofile%* predstavlja stazu do korisničke mape koja se uobičajeno nahodi na lokaciji *C:**Users**korisničko_ime*\ [9].
- *~/.ssh/authorized_keys* – ovaj zapis ima u sebi javne ključeve koji su potrebni za autentifikaciju korisnika dok se prijavljuje na poslužitelj. Ova datoteka se nahodi na daljinskom posluživaču i prigodom povezivanja korisnika na poslužitelj pregledava se nahodi li se poslani javni ključ u datoteci *authorized_keys* i ako se nahodi dotično ako je kombinacija para privatno-javnoga ključa ispravna, korisniku je dopušten pristup [10].
- *~/.ssh/id_rsa* – ovaj zapis ima u sebi privatne ključeve koji su potrebni za autentifikaciju korisnika na daljinskom posluživaču, a rabe se na klijentskoj strani prigodom uspostave SSH-veze [11];
- */etc/ssh/sshd_config* – sustavska konfiguracijska datoteka za SSH-poslužitelj;
- */etc/ssh/ssh_config* – sustavska konfiguracijska datoteka za SSH-klijent. Na stazi se *~/.ssh/config* nahodi korisnička konfiguracijska datoteka. Kod operacijskih se sustava Windows korisnička konfiguracijska datoteka nahodi na stazi *%userprofile%**.ssh**config* dok se sustavska datoteka koja vrijedi za sve korisnike nahodi na stazi *%programdata%**ssh**ssh_config* [12].

2.2.3. Sigurnost SSH-protokola

Iako je SSH-protokol sam po sebi vrlo siguran, od iznimne je bitnosti ispravno osigurati SSH-veze najvišim mogućim normama kako ne bi došlo do nepoželjnih promjena na posluživaču ili pak krađe podataka ili kojih drugih osjetljivih stvari.

2.2.3.1. Preporuke za sigurnu porabu SSH-a

Glavne preporuke za sigurnu porabu SSH-a jesu [13]:

- zahtijevanje snažnih zaporaka – snažne su zaporke koje imaju makar 12 znakova ter kombiniraju malena i velika slova ter brojke i posebne znakove eksponencijalno teže hakerima za napadaj grubom silom;
- 2-faktorska autentifikacija – pruža drugi način potvrde identiteta osim zaporke;
- zaključavanje računa – nakon nekoliko se neuspjelih pokušaja prijave račun za koji se pokušava prijaviti zaprječuje na određeno vrijeme;
- regeneracija SSH-ključeva – periodična je regeneracija SSH-ključeva dobra zamisao kako bi se smanjila prijetnja od neovlaštena pristupa ključu koji bi inače mogao proći neopaženo;
- promjena standardnih SSH-vrata – po zadanom, SSH rabi vrata 22 što je napadačima i različnim internetskim botovima prvi pokušaj kod napadaja tako da ih ovaj korak može makar malo usporiti ili makar smanjiti „šum“ u zapisima prijave;
- onemogućivanje prijave kao korijenski korisnik – korijenski korisnik imade potpun nadzor nad sustavom tako da ako napadač dobije korijenski pristup posljedice mogu biti katastrofalne;
- onemogućivanje autentifikacije zaporkom – ako je moguće, uvijek je poželjnije i sigurnije rabiti SSH-ključeve kako bi se smanjila prijetnja od napadaja grubom silom na zaporke
- Fail2Ban – ostvaraj programske rješidbe Fail2Ban može se rabiti za zaprječivanje IP-adresa koje pokušavaju izvršiti napadaje grubom silom samo od sebe

2.2.3.2. Zapis SSHFP DNS

SSHFP-zapisi omogućuju pohranjivanje otiska ključeva SSH-posluživača u DNS-u. Umjesto ručnoga pregleda ključeva domaćina SSH-klijent može potražiti otiske ključa posluživača na koji se želi spojiti u DNS-u i usporediti ih s onima koje prikazuje posluživač [14].

Vrlo se je vjerojatno prigodom prve prijave na neki SSH-posluživač vidjela sljedeća poruka:

```
ssh domba@ssh.dominik-lackovic.from.hr -p 2222

The authenticity of host '[ssh.dominik-lackovic.from.hr]:2222
([37.244.248.88]:2222)' can't be established.

ECDSA key fingerprint is
SHA256:70s5wQYnMClvqJe+vG0F/xellfdcYb8ort+Ns55uikc.

Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

Ova poruka prikazuje otisak javnoga ključa SSH-posluživača i traži da se pregleda je li posluživač na koji se povezuje ispravan sustav. SSHFP automatizira ovaj postupak s DNS-om, a jedini je preduvjet da domena SSH-posluživača bude konfigurirana s uključenim DNSSEC-izborom i da ovaj lokalni DNS-posluživač može posluživati DNSSEC-zahtjeve. Prvo na posluživaču treba izroditi zapise SSHFP DNS:

```
ssh-keygen -r ssh.dominik-lackovic.from.hr
```

I rečene dodati u DNS-posluživač. Primjer zapisa DNS SSHFP:

```
ssh.dominik-lackovic.from.hr. 300 IN    SSHFP   3 1
D7D7746568C3C818F0C72F5981CD8A7A5FC838DB

ssh.dominik-lackovic.from.hr. 300 IN    SSHFP   1 2
1F6F21B95EFA8A10789415EF4D141BFB11C93C4817F217175BC14112 899E6AF6
```

Nakon toga preostaje još samo u konfiguraciju SSH-klijenta dodati mogućnost

```
VerifyHostKeyDNS yes
```

Tim se za pravo govori klijentu da pregleda postoje li SSHFP-zapisi za domenu na koju se pokušava spojiti jer po zadanom to nije uključeno. Sad, ako je sve ispravno postavljeno prigodom prijave na SSH-posluživač ne bi se trebala vidjeti poruka da se ne može pregledati autentičnost posluživača iako otisak javnoga ključa nije pohranjen u datoteku „*known_hosts*“. Ako se i dalje dobiva poruka kao prije uz dodatak: „*Matching host key fingerprint found in DNS.*“ to znamenuje da je klijent OpenSSH pronašao SSHFP-zapise, ali ih nije mogao potvrditi rabeći DNSSEC pa treba pregledati je li DNSSEC ispravno postavljen kod domene i rabi li ovo klijentsko računalo DNS-posluživač koji podupire DNSSEC. U ovom slučaju DNS-posluživač nije bio postavljen da rabi DNSSEC pa se je promijenila konfiguracija DNS-posluživača:

```
sudo nano /etc/systemd/resolved.conf
```

I odredili sljedeći DNS-posluživači:

```
[Resolve]
DNS=1.1.1.1 1.0.0.1
FallbackDNS=8.8.8.8 8.8.4.4
DNSSEC=yes
```

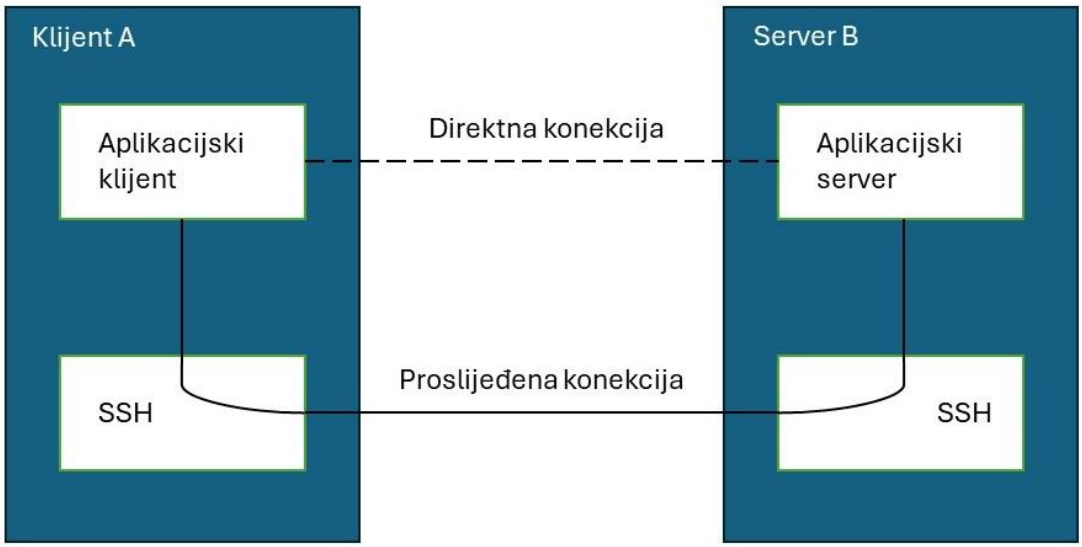
Nakon pohrane se je konfiguracije ponovno pokrenula DNS-usluga:

```
sudo systemctl restart systemd-resolved
```

I prigodom spajanja na SSH-posluživač više nije dočekala prijašnja poruka što znamenuje da SSHFP ispravno radi i da je sad mogućnost za napadaje „MITM“ (*man-in-the-middle*) svedena na najmanju.

2.2.4. SSH-prosljeđivanje vrata

Takozvani „SSH-tuneli“ omogućuju preko protokola SSH-CONN protokola prosljeđivanje prometa koji dohodi na SSH-vratima na koja druga vrata na daljinskom računalu. Dakle, SSH prvo dekriptira što imade u paketima koji dohode od korisnika A pa to za tim prosljeđuje na „prava“ vrata na posluživaču B gdje drugi posluživač sluša i prima namijenjeni promet. Same mogućnosti SSH-a su bezbrojne – od zaobilaženja zapriječenih vrata pa do stvaranja privatnih prividnih mreža [15].



Slika 1. SSH-prosljeđivanje vrata

Postoje tri glavne vrste prosljeđivanja vrata: lokalno, daljinsko i dinamično. Svaka od njih ima svoje značajne primjene i rabi se u različnim slučajima.

2.2.4.1. Lokalno prosljeđivanje vrata

Lokalno prosljeđivanje vrata omogućuje da se promet put lokalnoga računala i vrata prosljeđuje na daljinski posluživač i vrata preko SSH-tunela. Ovo se obično rabi kad se želi

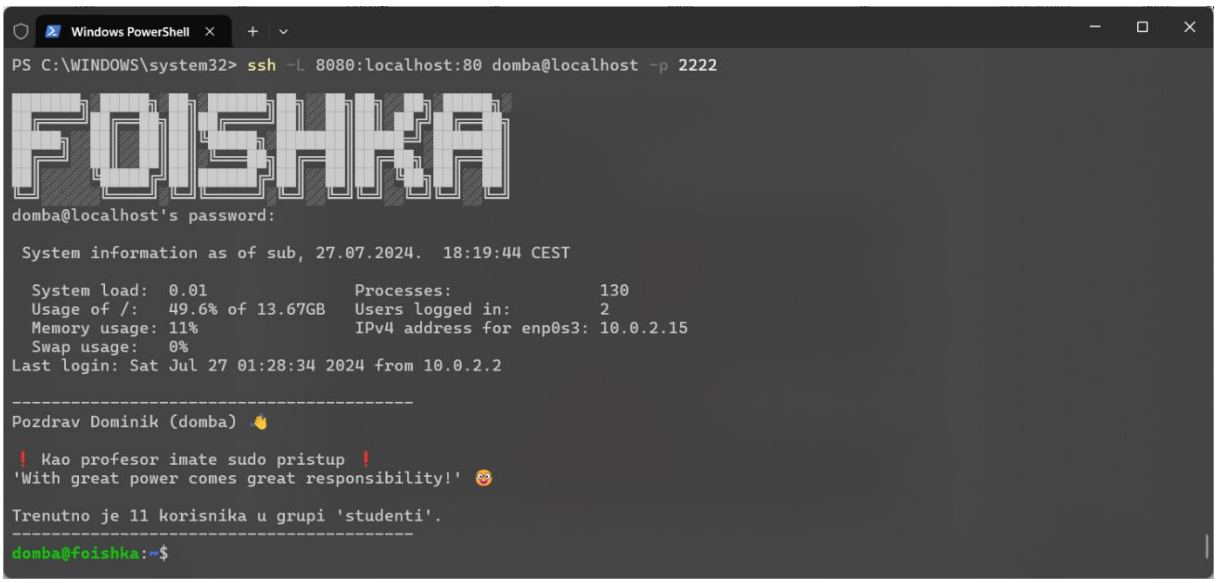
pristupiti sredstvu na daljinskom posluživaču koje nije izravno raspoloživo zbog vatrozida ili kojih drugih ograničenosti [16, 17].

Npr. ako se želi pristupiti daljinskom *web*-posluživaču koji nije izravno raspoloživ ili u lokalnoj mreži, a može se spojiti na nj s pomoću SSH-veze. Po zadanom prividni stroj ima svoj mrežni prilagodnik i nije moguć pristup uslugama na prividnom stroju s računala-domaćina.

Na domaćinu se pokreće SSH-veza dotično lokalno prosljeđivanje vrata sljedećom naredbom:

```
ssh -L 8080:localhost:80 domba@localhost -p 2222
```

- Argument -L obilježava da se radi o lokalnom prosljeđivanju vrata dotično da se vrata 8080 na lokalnom računalu prosljeđuju na *localhost* na vratima 80. Umjesto se *localhosta* možemo rabiti i koja druga adresa po potrebi ako je ta adresa raspoloživa i dohvatljiva preko daljinskoga računala (zaobilaženje vatrozida ili zapriječenih vrata).
- Argument -p obilježava vrata preko kojih se ostvaruje SSH-veza. Nakon uspješne će autentifikacije promet biti proslijeđen kroz SSH-tunel do daljinskoga posluživača na vrata 80 pa se sad na računalu domaćinu može pristupiti web-posluživaču na vratima 80 daljinskoga računala.



Slika 2. Lokalno prosljeđivanje vrata

Slika 3. Lokalno prosljeđivanje vrata – *web*-poslužitelj NGINX

2.2.4.2. Daljinsko prosljeđivanje vrata

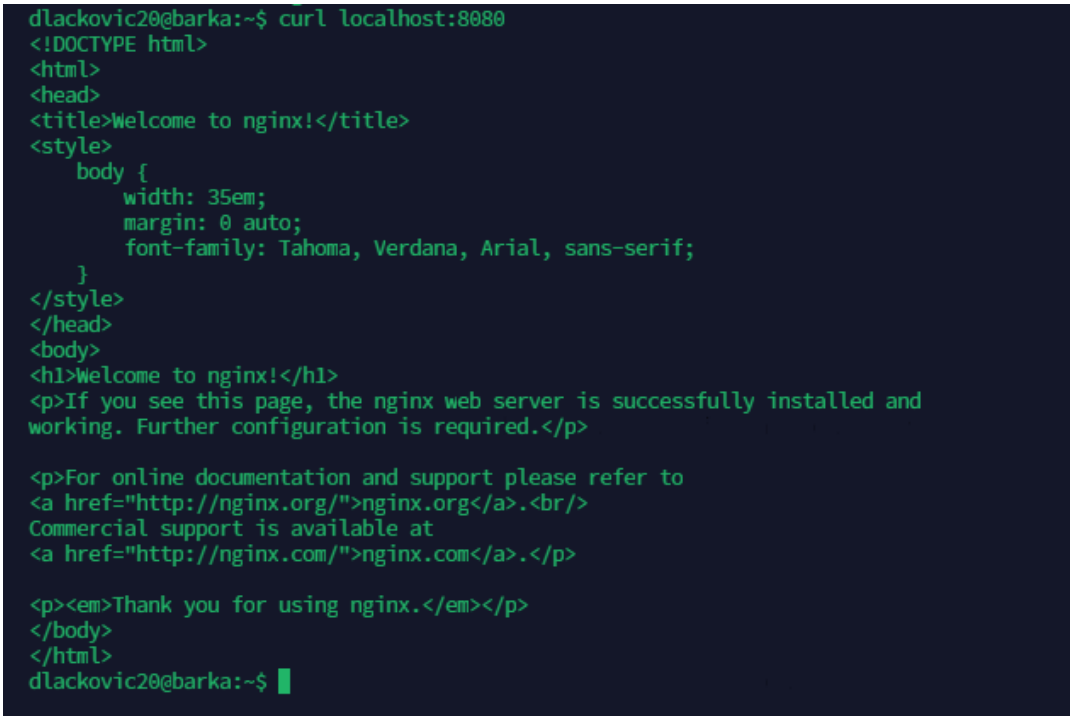
Daljinsko prosljeđivanje vrata omogućuje suprotno od lokalnoga prosljeđivanja vrata dotično omogućuje da se promet na daljinskom računalu, a put lokalnoga računala, prosljeđuje upravo na lokalni poslužitelj i vrata preko SSH-tunela. Ovo se rabi kad se želi omogućiti pristup sredstvu na ovom lokalnom računalu s daljinskoga poslužitelja [16, 17].

Npr. ako se daljinskomu računalu *barka.foi.hr* želi omogućiti pristup ovomu *web*-poslužitelju na vratima 80, na ovom bi se poslužitelju-domaćinu izvršila slična naredba kao i prije kod lokalnoga prosljeđivanja vrata:

```
ssh -R 8080:localhost:80 dlackovic20@barka.foi.hr
```

Dakle, kad se uspostavi SSH-veza s poslužiteljem barka.foi.hr, stvorit će se tunel koji će omogućiti korisnicima na poslužitelju *barka.foi.hr* pristup lokalnomu poslužitelju na njihovim daljinskom vratima 8080 koji se lokalno izvađa na vratima 80. Da bi se pregledalo, može se izvršiti naredba:

```
curl localhost:8080
```



Slika 4. Dohvat *web*-stranice preko daljinskoga prosljeđivanja vrata

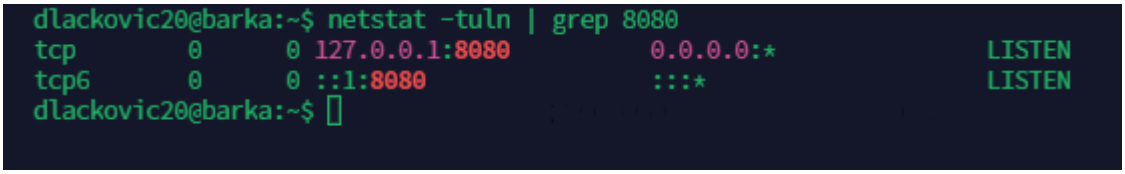
Vidi se da se je dobio HTML-zapis prijašnje ispitne stranice koja se je mogla vidjeti i kod lokalnoga prosljeđivanja vrata. Bitno za napomenu kod daljinskoga prosljeđivanja vrata jest da daljinski poslužitelj mora imati omogućen izbor za prosljeđivanje vrata. U konfiguracijskoj datoteci SSH-poslužitelja (*/etc/ssh/sshd_config*) na daljinskom poslužitelju sljedeće postavke trebaju biti omogućene:

- `AllowTcpForwarding yes`
- `GatewayPorts yes`

Ako se i dalje nalijeće na teškoće kod prosljeđivanja vrata naredbom se *netstat* može pregledati koja su vrata i veze trenutačno otvorena ter jesu li u stvari otvoreni jer možebiti vatrozid zaprječuje određena vrata:

```
netstat -tuln | grep 8080
```

Argument je *-t* i *-u* u svezi s prikazom veza *tcp* i *udp*, *-l* s vratima koja „slušaju“ (*listening*) jer su to vrata koja čekaju na dolazne veze, a *-n* prikazuje brojevne zapise adresa bez DNS-razrješivanja ili prevađanja u nazive usluga. U ovom slučaju vrata 8080 bi se bez te mogućnosti razriješila u „*localhost:http-alt*“ pa je teže prepoznati o kojim se precizno vratima radi. Daljim će se preusmjeravanjem izlaza naredbe s pomoću naredbe *grep 8080* dobiti kao ispis samo redovi koji imaju u sebi 8080 dotično ova željena vrata.



Slika 5. Pregled otvorenih vrata

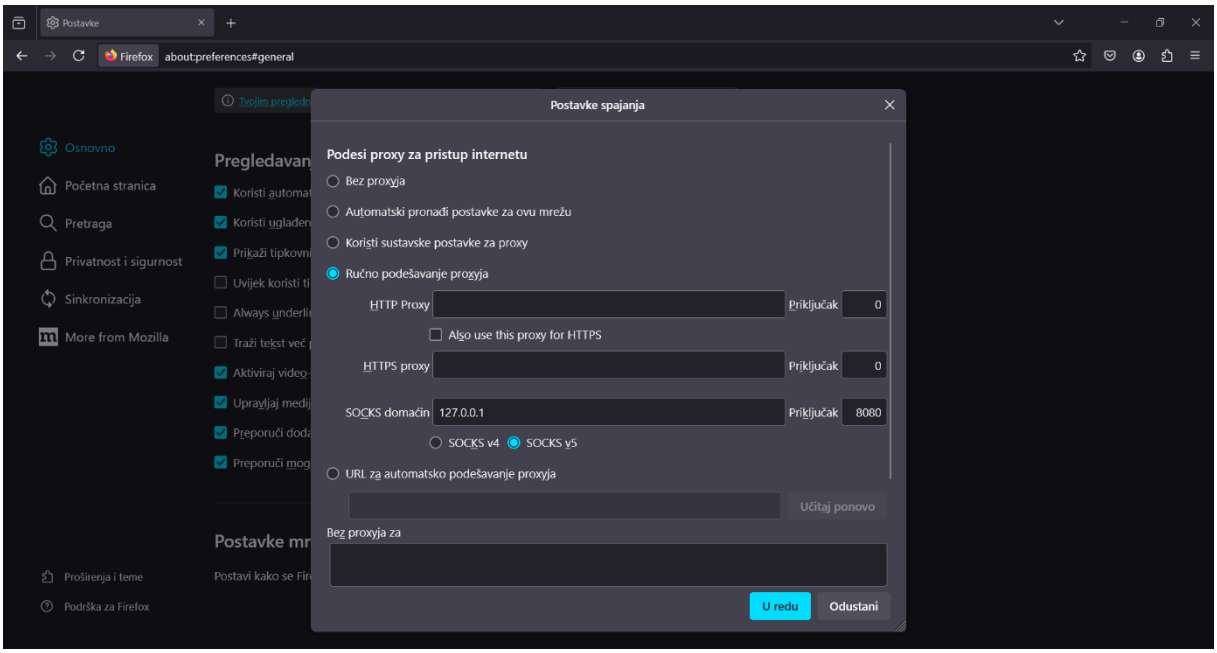
2.2.4.3. Dinamično prosljeđivanje vrata

Dinamično prosljeđivanje vrata omogućuje da se promet s lokalnoga računala prosljeđuje na različne odredišne poslužiteljske računalne programe preko SSH-tunela rabeći *SOCKS proxy*. Ovo se rabi kad se želi uspostaviti tunel koji može prosljeđivati promet do bilo kojega poslužiteljskoga računalnoga programa s daljinskoga poslužiteljskoga računala, slično VPN-u [16, 17].

Npr. pokretanjem naredbe:

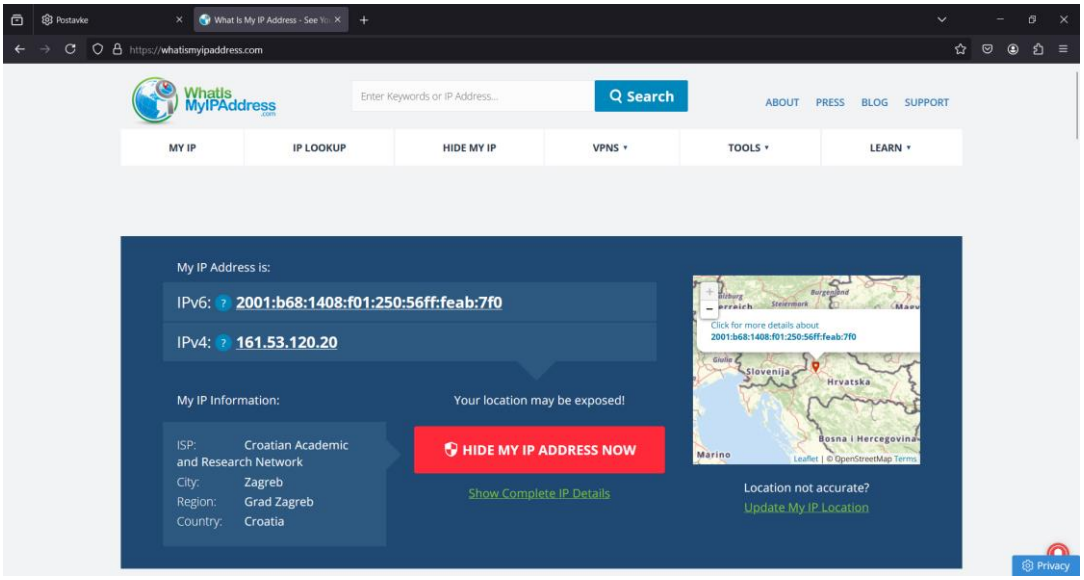
```
ssh -D 8080 dlackovic20@barka.foi.hr
```

pokreće se poslužiteljski računalni program *SOCKS proxy* na lokalnim vratima 8080 i preostaje samo postaviti *web*-preglednik ili operacijski sustav da rabi *SOCKS proxy* na naslovu *localhost:8080*.



Slika 6. Konfiguracija poslužiteljskoga računalnoga programa *SOCKS proxy* u *web*-pregledniku Mozilla Firefox

Nakon konfiguriranja preglednika da rabi *SOCKS proxy* sav promet iz preglednika ili aplikacije ide kroz SSH-tunel preko daljinskoga posluživača s pomoću SSH-tunela i tim se dobiva dodatna sigurnost i privatnost na Internetu. Da bi se pregledalo djeluje li to u istinu, može se pohoditi *web*-stranica usluge *whatismyipaddress.com* i na istoj se vidi da IP-adresa ne odgovara ovoj IP-adresi koja je dodijeljena od ISP-a tako da *SOCKS proxy* djeluje ispravno.



Slika 7. *Web*-usluga *whatismyipaddress.com*

3. Stvaranje prividnoga stroja za SSH-poslužitelj

Ostvaraj SSH-poslužitelja može se izvršiti na različne načine, a ponajviše ovisi o operacijskom sustavu i samim potrebama budućih korisnika poslužitelja, kako je OpenSSH poduprt na glavnini operacijskih sustava – Windows, Linux/Unix, macOS. Ključni preduvjeti uključuju pristup korisniku s administratorskim povlasticama ter instalaciju i konfiguraciju SSH-programa za poslužitelj i klijente. Najznamenitiji je način ostvaraja koje bilo vrste poslužitelja zakupom prividnoga stroja u oblaku rabeći usluge u oblaku poput usluge Amazon AWS (EC2, Lightsail), Microsoft Azurea (Virtual Machines), Google Clouda, Oracle Clouda, DigitalOceana, Vultra, Linodea, Hetznera, OVHclouda i brojnih drugih. Glavna prednost ostvaraja u oblaku jest ta da se dobije statična IP-adresa već za nekoliko eura mjesečno ter prividni stroj s određenom hitrošću i brojem procesora, kapacitetima radne memorije i diskovnoga prostora kao i internetskom vezom s određenom propusnošću u smislu hitrosti uzlazne i silazne veze ter ograničenosti u broju gigabajta ili terabajta prijenosa podataka s poslužitelja.

Bilo da se ostvaruje na fizičkim strojevima, prividnim strojevima ili u *containerima*, ispravno konfiguriran SSH-poslužitelj pružit će siguran način za daljinsko upravljanje i prijenos podataka među svim korisnicima, a pri tom će i biti zaštićen od neželjenih upada.

S obzirom na to da je cilj rada prikazati kako bilo tko može ostvariti vlastiti poslužitelj, kao metoda se je ostvaraja SSH-poslužitelja odabrao prividni stroj na operacijskom sustavu Windows koja će u daljim poglavljima biti po tanje opisana.

3.1. Preduvjeti

Osim već spomenutoga preduvjeta za pristup korisnika s administratorskim povlasticama kako bi se mogao instalirati OpenSSH naravno da je potreban i pristup Internetu (po mogućnosti stalan kako se radi o poslužitelju), a samim tim i IP-adresa poslužitelja kako bi mu se moglo pristupiti izvan lokalne mreže. IP-adresa poslužitelja treba biti javno raspoloživa, a vrata na kojima je SSH-poslužitelj trebaju biti otvorena. SSH-poslužitelj po zadanom sluša na vratima broj 22, ali naravno da ih je moguće promijeniti na koja bilo druga željena vrata.

Kako će se dalji ostvaraj činiti na lokalnom računalu s fiksnim Internetom od lokalnoga ISP-a, kao dodatan preduvjet trebat će se pregledati kod ISP-a nahodi li se za CGNAT-om.

Naravno da treba zadovoljiti i uvjete za pokretanje operacijskoga sustava na prividnom stroju. Kako bi računalo-domaćin moglo normalno djelovati, a uza to cijelo vrijeme izvršavati i

prividni stroj, potreban je procesor s više jezgara i puno RAM-a. Za operacijski sustav Ubuntu Server 22.04.4 LTS glavni preduvjeti jesu: 1024 MB RAM-a, 5 GB mjesta na disku i 64-bitni procesor ako se rabi procesor Intel ili AMD [18]. Najmanji su preduvjeti najmanji pa je uvijek dobra praksa ako to sredstva računala dopuštaju da se u prividnom stroju dodijeli ipak nešto više sredstava da bi u slučaju iznenadnih povećavanja obujma rada na posluživaču rečeni mogao sve uredno izvršiti ne narušujući cjelovitost računala domaćina.

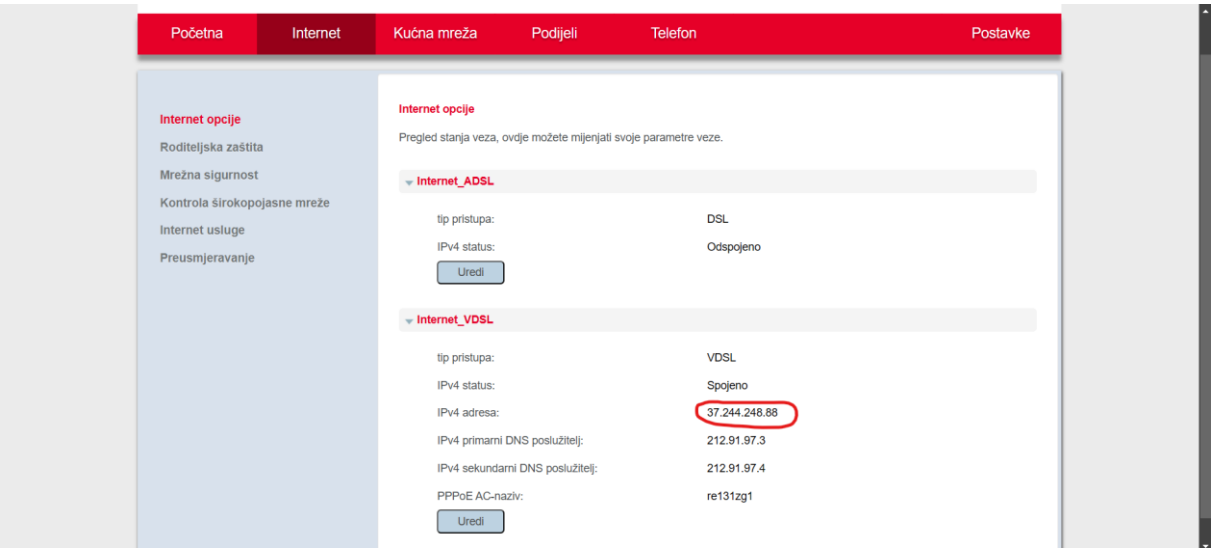
3.1.1. NAT i CG-NAT

Posluživač treba imati javno raspoloživu IP-adresu, a ako će se rabiti fiksni Internet kućnoga ISP-a to često ne će biti slučaj. Razlog se krije u NAT-u i CG-NAT-u.

Network Address Translation (NAT) mrežna je tehnika koja omogućuje promjenu IP-adresa u zaglavljima paketa koji prohode kroz usmjernik ili vatrozid. NAT omogućuje uređajima na lokalnoj mreži (LAN) s privatnim IP-adresama pristup Internetu rabeći jednu ili više javnih IP-adresa. Tako se osigurava skrivenost mreže dotično privatnih IP-adresa jer one ostaju skrivene za javnom IP-adresom usmjernika. Isto tako, štedi se na IP-adresama jer se jedna javna IP-adresa rabi za više privatnih uređaja ter se tim podupiru i istodobne veze rabeći različna vrata [19, 20].

S druge je strane Carrier-Grade NAT (CG-NAT) isto tako znan kao Large Scale NAT (LSN) inačica NAT-a koju rabe pružatelji internetskih usluga (ISP – Internet Service Provider) kako bi omogućili pristup Internetu za više korisnika preko ograničena broja javnih IP-adresa. Tim ISP-i štede na adresama IPv4 koje su svjetski ograničene ter ih ponestaje i mogu upravljati velikim brojem korisnika bez hitne i masovne migracije na protokol IPv6. CG-NAT djeluje tako da se postavlja sloj NAT-a unutar mreže ISP-a. Privatne se IP-adrese korisnika prvo preusmjeruju na lokalne IP-adrese unutar ISP-ove mreže, a za tim se te lokalne IP-adrese preusmjeruju na javne IP-adrese koje ISP rabi. Hodeći Internetom, katkad se zna dogoditi da se naiđe na web-stranicu koja zaprječuje pristup ili je pak prepoznana sumnjiva djelatnost s ove IP-adrese koja je dijeljena s ostalima pa je potrebno riješiti izazov reCAPTCHA. To je znamen da je ta prepoznana djelatnost stvarno došla odavle ili pak vrlo vjerojatno od drugoga kompromitiranoga računala koje dijeli istu javnu IP-adresu [21, 22].

Da bi se pregledalo nahodi li se za CG-NAT-om potrebno je prijaviti se na administratorsko sučelje usmjernika. Kod glavnine usmjernika ono se nahodi na lokalnoj adresi 192.168.1.1. Nakon prijave s administratorskim korisničkim imenom i zaporkom treba pronaći odjeljak s stanjem i podatcima internetske veze ter pregledati koja je IP-adresa zabilježena tj. dodijeljena kod ovoga usmjernika. Ovo su općenite smjernice jer svaki usmjernik ima drugačije sučelje i postavke se nahode na različnim mjestima.



Slika 8. Stanje internetske VDSL-veze i IP-adrese u nadzornoj ploči za *web* usmjernika

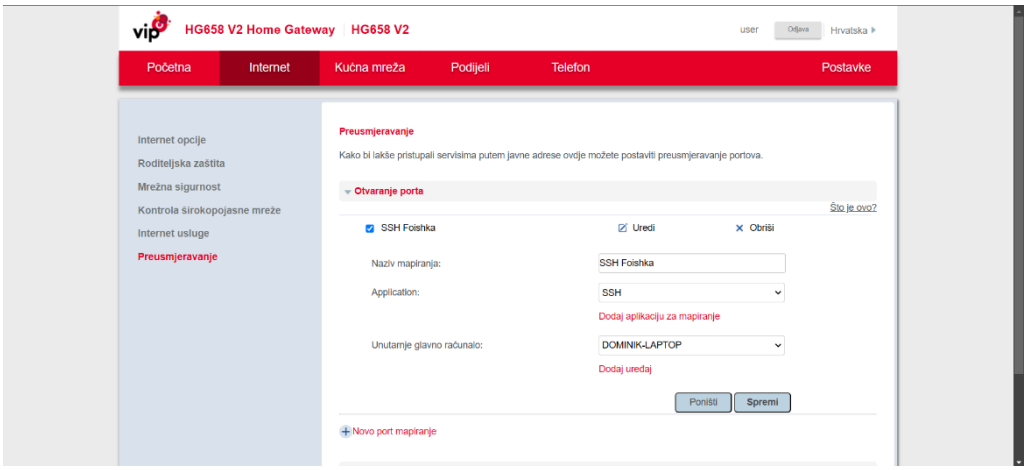
Dalje, ako se spomene već prije spomenute usluge iz poglavlja 2.2.4.3 – https://whatismyipaddress.com - i ako se ondje pokaže da je ova javna IP-adresa jednaka ovoj IP-adresi koja je dodijeljena na usmjerniku, to znamenuje da se ne nahodi za CG-NAT-om i da ISP tu IP-adresu nije dodijelio nikomu drugomu tako da se mogu ove usluge javno dijeliti. Ako pak se utvrdi da je javna IP-adresa drugačija, potrebno je čuti se s ISP-om dotično službom za korisnike preko poziva ili *maila* i zatražiti da se makne s CG-NAT-a, a vjerojatno će biti potrebno i navesti valjan razlog.

3.1.2. Otvaranje vrata

Sad se kad je ova IP-adresa javno raspoloživa trebaju i otvoriti željena vrata tako da se dopusti promet na tim vratima dotično da usmjernik znade kad primi promet na tim vratima na koje računalo tj. lokalnu IP-adresu proslijediti promet. Tako da se prvo treba osigurati da računalo-domaćin ima statičnu lokalnu IP-adresu. Da bi se osiguralo da DHCP-posluživač na usmjerniku uvijek dodjeljuje statičnu IP-adresu ovomu računalu, potrebno je znati MAC-adresu računala. MAC-adresa na operacijskom sustavu Windows 11 može se doznati preko slikovnoga sučelja tako da se otvore „Postavke“ – „Napredne mrežne postavke“ – pod „Mrežni prilagodnik“ odabere se onaj preko kojega je spojeno na Internet (Wi-Fi ili Ethernet) – „Prikaži dodatna svojstva“ i iščita se stupac „Fizička adresa (MAC)“. Za one koji više vole manje lutanja i pritiskanja po izbornicima naravno da je rečeno moguće doznati i preko konzole dotično Command Prompta ili Powershella upisivanjem naredbe *ipconfig /all* koja će izlistati sve podatke o raspoloživim mrežnim prilagodnicima.

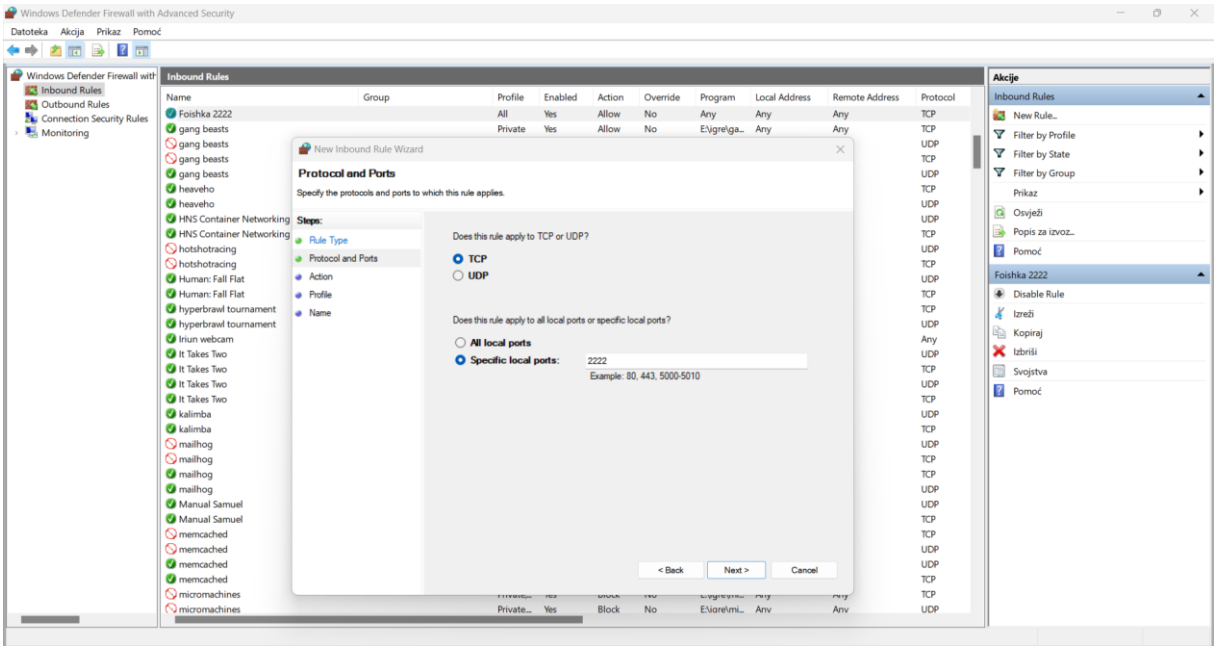
Pošto se znade MAC-adresa mrežnoga prilagodnika, doba su da se u usmjerniku dodijeli statična IP-adresa ovomu računalu dotično da se učini DHCP-predbilježba za MAC-adresu koja se je malo prije doznala. Ako se je dodijelila IP-adresa koja je različna od trenutačno dodijeljene, treba promijeniti konfiguraciju mrežnoga prilagodnika tako da se ponovno ručno unese statična IP-adresa ili se isključi s Wi-Fi-ja i ponovno poveže kako bi se povukla nova DHCP-konfiguracija. Isto vrijedi i za povezivanje preko LAN-kabela.

Zadnji korak kod otvaranja vrata preostaje da se konačno otvore vrata. U postavkama usmjernika je potrebno odrediti koja se vrata žele otvoriti i put kojega uređaja/IP-adrese će se rečena prosljeđivati. Ako se rabe nestandardna vrata, možebiti će potrebno biti prvo dodati novu aplikaciju za preslikavanje vrata, ali to ponovno ovisi o samoj izvedbi usmjernika. U ovom slučaju kako će se rabiti vrata 2222 trebala se je odrediti nova aplikacija, dodati joj naziv, vanjski priključak, nutarnji priključak i protokol koji će se prosljeđivati. Vanjski i nutarnji priključak dotično vrata odredila su se kao raspon od 2222 do 2222, a protokol je TCP jer SSH rabi TCP.



Slika 9. Otvaranje vrata u nadzornoj ploči za *web* usmjernika

Kad su vrata otvorena na usmjerniku, preostaje otvoriti i vatrozid na računalu dotično izraditi pravilo da se propušta dolazni promet na vratima 2222. Na Windowsima je potrebno otvoriti „Windows Defender Firewall with Advanced Security“ – „Inbound Rules“ – „New Rule“ – „Port“ – „TCP“, „Specific local ports: 2222“ – „Allow the connection“ – odaber se na kojim se sve mrežama želi da se pravilo izvršuje (privatne mreže, javne mreže, domena) – dodade se naziv pravila i pohrani se.



Slika 10. Windows Defender Firewall – propuštanje SSH-prometa na vratima 2222

Na operacijskim bi se sustavima Linux rabila sljedeća naredba za otvaranje vrata 2222 u vatrozidu:

```
sudo iptables -A INPUT -p tcp --dport 2222 -j ACCEPT
```

- Argument *-A INPUT* znamenuje da se dodaje (*add*) pravilo u lanac *"INPUT"* dotično da se pravilo primjenjuje na dolazni promet.
- Argument *-p tcp* obilježava da se radi o TCP-prometu.
- Argument *--dport 2222* – obilježava odredišna vrata dotično vrata koja se žele otvoriti na dolaznom prometu.
- Argument *-j ACCEPT* obilježava da se paketi koji odgovaraju gore navedenomu pravilu prihvate.

Nakon izvršivanja naredbe treba trajno pohraniti pravila jer se ona inače nakon ponovnoga pokretanja sustava gube:

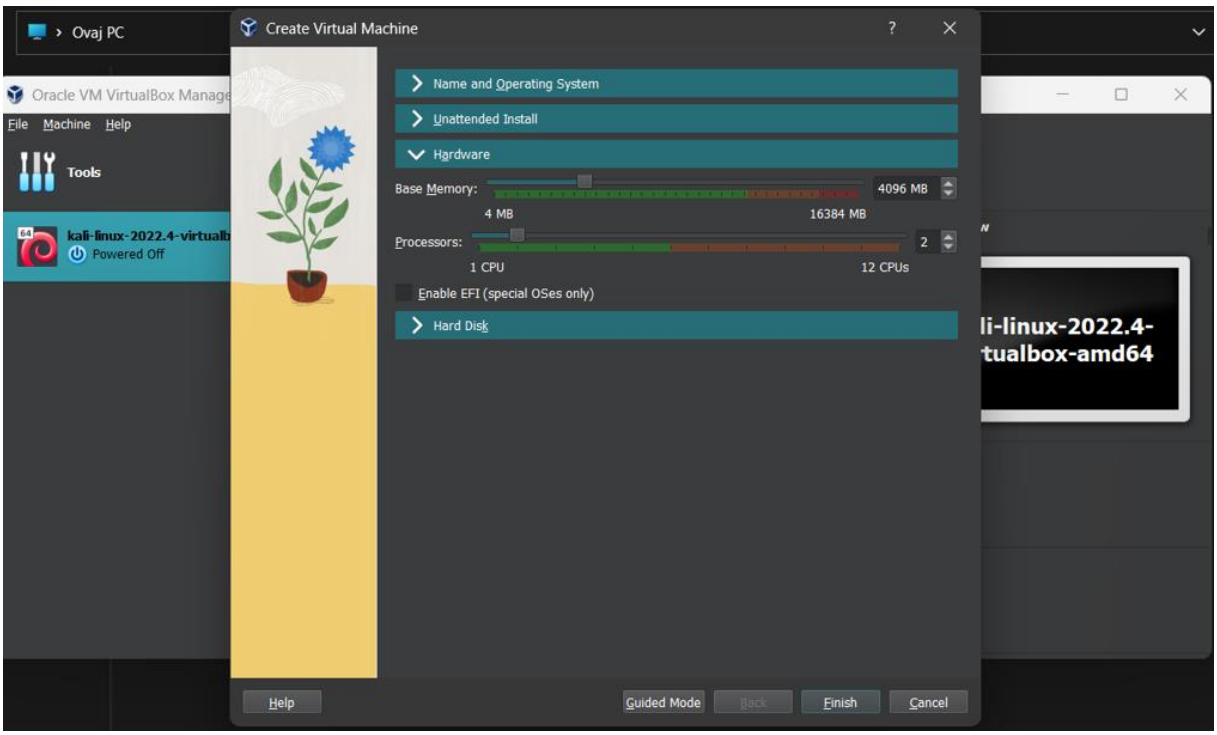
```
sudo netfilter-persistent save
```

Skladnja vatrozida *iptables* često znade biti zbunjujuća pa za manje spretne korisnike preporučuje se *ufw* (*uncomplicated firewall*) čija je skladnja vrlo jednostavna i čitljivija:

```
sudo ufw allow 2222/tcp
```

3.2. Instalacija Ubuntu Servera

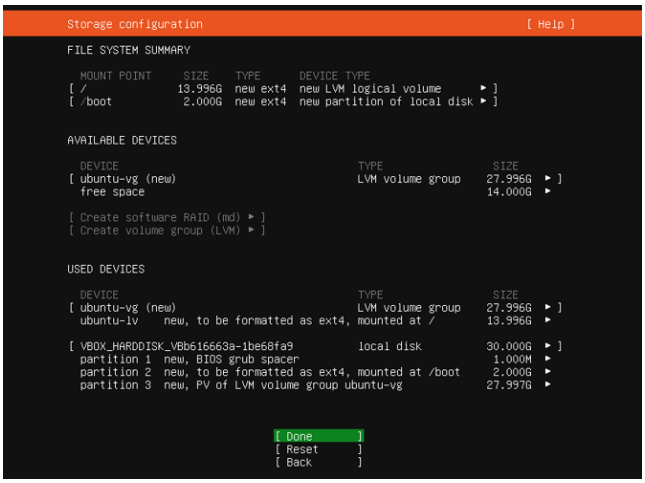
Za instalaciju Ubuntu Servera inačice 22.04.4 LTS koji će služiti kao ovaj SSH-poslužitelj rabit će se oruđe VirtualBox s pomoću kojega će se stvoriti prividni stroj na operacijskom sustavu Windows. Odabrana je inačica LTS (Long Term Support) jer je to inačica koja ima potporu za sigurnosna osvježavanja kroz dulje vrijeme, a u slučaju inačice 22.04 to će biti do travnja 2027. godine. Nakon preuzimanja instalacijske datoteke .iso s službenih stranica https://ubuntu.com/download/server, samo stvaranje prividnoga stroja vrlo je lako i intuitivno ter se samo slijedi čarobnjak za stvaranje novoga prividnoga stroja, a za sklopovske se je potrebe prividnoga stroja dodijelilo 4096 MB radne memorije i 2 jezgre procesora ter 30 GB prostora na tvrdom disku dotično SSD-u što je za potrebe ovoga SSH-poslužitelja i više nego dovoljno.



Slika 11. Stvaranje prividnoga stroja u VirtualBoxu

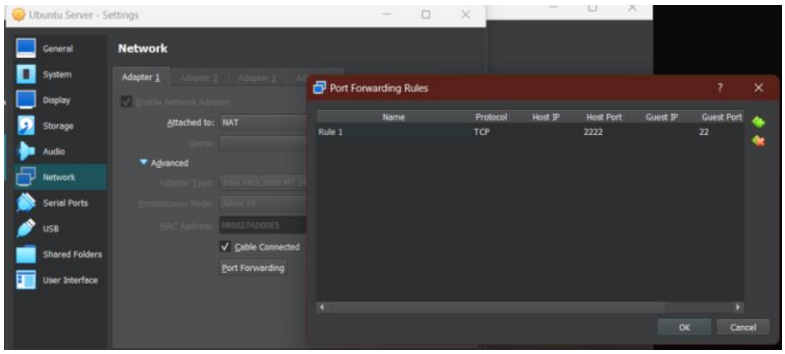
Nakon pokretanja prividnoga stroja slijedi instalacijski čarobnjak koji je isto tako raspoloživ na hrvatskom jeziku, a ako je raspoloživa nadogradnja instalacijskoga programa, korisnik će biti upitan želi li da se sama od sebe skine zadnja inačicu. Za tim se postavlja raspored i inačica tipkovnice koja se rabi ter odabire koja se inačica Ubuntu Servera želi instalirati – standardna ili najmanja. Najmanja ima manji memorijski otisak, ali nije preporučljiva

osim ako nema precizne sigurnosti da to treba jer dohodi bez nekih osnovnih programa kao što je uređivač teksta i sl. Odabire se mrežno sučelje, konfigurira se disk i dobiva se sažetak svih promjena koje će instalacijski čarobnjak učiniti. Nakon potvrde se stvara korisnički profil i zaporka, bira se naziv posluživača (*hostname*) i odabire se da se želi instalirati i posluživač OpenSSH. Nakon toga instalacija počinje i nakon 10-ak se minuta dobiva pristup konzoli gdje se prijavljuje kao korisnik koji se je stvorio u instalacijskom čarobnjaku.



Slika 12. Instalacijski čarobnjak za Ubuntu Server 22.04 – sažetak instalacije

Sad, kad se konačno imade ispravan SSH-posluživač, želi mu se pristupiti izvan prividnoga stroja jer tomu i služi. Potrebno je postaviti prosljeđivanje vrata domaćina na vrata prividnoga stroja. U postavkama prividnoga stroja u VirtualBoxu pod odjeljkom se „Network“ odabire mrežni prilagodnik, otvaraju se napredne postavke i dohodi se do izbora „Port Forwarding“ gdje se određuje koja se vrata žele prosljeđivati. Ako se na domaćinu želi pristupiti preko vrata 2222 na vrata 22 u prividnom stroju tako se i postavi:



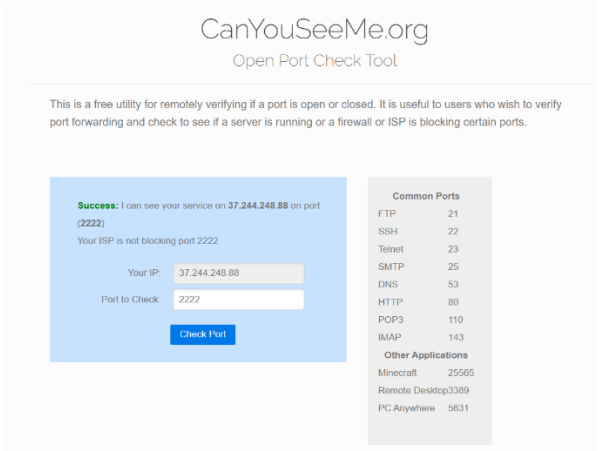
Slika 13. Prosljeđivanje vrata u VirtualBoxu

Na domaćinu se sad može s pomoću SSH-klijenta spojiti na Ubuntu Server rabeći naredbu:

```
ssh domba@127.0.0.1 -p 2222
```
ili
```
ssh domba@localhost -p 2222
```

Argument *-p* određuje SSH-vrata, a u ovom se slučaju vrata 2222 prosljeđuju na vrata 22 u prividnom stroju. Nakon uspješne je autentifikacije uspješno prijavljeno kao korisnik *domba* preko SSH-a na prividnom stroju.

Trenutačno je prijavljeno samo na lokalnom računalu, ali željelo bi se daljinski upravljati dotično se spajati na SSH-poslužitelj s drugih mreža. Prvo će se pregledati jesu li ova vrata otvorena i je li SSH raspoloživ vanjskim korisnicima. Za to se može rabiti usluga canyouseeme.com gdje se upisuje broj vrata za koja se želi pregledati jesu li raspoloživa, a IP-adresa sama se od sebe popuni. Ako se je sve dobro konfiguriralo i stvorila pravila za vatrozid koja propuštaju vrata 22, trebala bi se vidjeti poruka uspjeha.



Slika 14. Internetska usluga CanYouSeeMe.org za pregled otvorenosti vrata

Da bi se zaista uvjerilo u samu ispravnost konfiguracije, najbolje je da se pokuša spojiti na SSH-poslužitelj preko druge mreže, npr. preko mobitela dotično pokretnih podataka. Upisivanjem bi se naredbe na mobitelu

```
ssh domba@KUĆNA_IP_ADRESA -p 2222
```

trebalo biti upitan za zaporku korisnika što znamenuje da sve radi. Ako se veza ne uspostavi u roku od nekoliko sekunda, vrlo vjerojatno nešto ipak nije dobro postavljeno i treba pregledati sve zapise za lakše otklanjanje mogućih pogrješaka.

Slika 15. SSH-pristup s mobitela i vanjske IP-adrese

Kod prvoga spajanja na SSH-poslužitelj dobiva se poruka da se autentičnost domaćina *127.0.0.1:2222* ne može utvrditi. Ispiše se otisak SHA256 javnoga ključa poslužitelja i upitano je želi li se nastaviti veza s obzirom na to da ovaj ključ nije znan ni pod kojim drugim imenom. Kad se potvrdi, javni se ključ poslužitelja pohranjuje u datoteku „*known_hosts*“ zajedno s nazivom domaćina. Kod svake sljedeće prijave otisak javnoga ključa poslužitelja uspoređuje se s otiscima ključeva i nazivima poslužitelja u datoteci „*known_hosts*“ i ako se što od toga dvojega ne poklapa ili ne postoji u datoteci, znamenuje da je došlo do promjene konfiguracije na samom poslužitelju ili se je žrtva napadaja „*man-in-the-middle*“ (MITM) dotično da je netko presreo ovu vezu i za pravo se spaja na drugi, zaraženi poslužitelj.

Najbolja bi praksa iz sigurnosnoga vida bila da se nakon instaliranja SSH-poslužitelja samostalno izračunaju otisci javnih ključeva poslužitelja i pohrane se na tajno mjesto kako bi se poslije moglo pregledati spaja li se u istinu na željeni poslužitelj. Da bi se to učinilo, potrebno je rabiti oruđe *ssh-keygen*:

```
sudo ssh-keygen -lf /etc/ssh/ssh_host_ed25519_key.pub -E sha256
```

Argument *-l* upozorava da se želi izračunati otisak ključa, *-f* je staza do samoga javnoga ključa, a *-E* upozorava na algoritam koji se želi rabiti za izračunavanje otiska javnoga ključa. Primjer nalaza izvršivanja prijašnje naredbe:

```
SHA256:C2FqZJIJBVe3KFS/YaFPOKEI root@foishka (ED25519)
```

Kad se usporedi otisak ključa s onim otiskom koji se je dobio kod prve prijave na SSH-posluživač, vidi se da se radi o istom ključu što znamenuje da se spaja na ovaj posluživač. Osim ključa koji rabi algoritam ED25519, postoje i različni drugi algoritmi koji se mogu rabiti za izrađanje ključeva, a najčešći su još RSA, DSA i ECDSA. ED25519 pruža najbolji omjer među sigurnošću i hitrošću i zbog toga je odabran kao zadan odabir kod brojnih distribucija operacijskih sustava.

Naravno da kad se spaja na javne SSH-posluživače tj. one na kojima se nema korijenski pristup i nije se sudjelovalo u instalaciji i konfiguraciji, ne može se na prijašnji način pregledati autentičnost SSH-posluživača dotično javnoga ključa posluživača nego je potrebno čuti se s administratorom toga SSH-posluživača da dopravi otisak javnoga ključa. Pošto se dobije otisak ključa, prije SSH-veze može se dohvatiti javni ključ s pomoću oruđa *ssh-keyscan*:

```
ssh-keyscan -p 2222 127.0.0.1 > temp_key.pub
```

Javni se ključ s SSH-posluživača na vratima 2222 pohranjuje u datoteku na računalu i tako se može izračunati otisak ključa rabeći prije opisano oruđe *ssh-keygen* i usporediti ga s dobivenim otiskom od administratora.

4. Instalacija programa i konfiguracija SSH-poslužitelja

Ako se kod instalacije Ubuntu Servera nije odabrala instalacija poslužitelja OpenSSH, uvijek se može ručno instalirati SSH-poslužitelj sljedećom naredbom:

```
sudo apt install openssh-server
```

apt je oruđe koje služi za instaliranje novih programskih paketa, nadogradnju postojećih programskih paketa pa pače i nadogradnju cijeloga Ubuntu-sustava. Ako će postojati potreba za izlaznim SSH-vezama s poslužitelja, potrebno je instalirati i paket *openssh-client*. Poželjno je i osvježiti svu programsku potporu izvršivanjem naredbe

```
sudo apt full-upgrade
```

Ako SSH-poslužitelj nije pokrenut, može se ga pokrenuti naredbom:

```
sudo systemctl restart sshd.service
```

ili

```
sudo service ssh start
```

Da bi se SSH-poslužitelj kod ponovnoga pokretanja sustava sam od sebe pokrenuo potrebno je izvršiti naredbu:

```
sudo systemctl enable sshd.service
```

4.1. Konfiguracija SSH-poslužitelja

Za što sigurniji će se daljinski rad na SSH-poslužitelju promijeniti zadana SSH-konfiguracija poslužitelja tako da bude nešto kruća. Konfiguracijska se datoteka nahodi na stazi */etc/ssh/sshd_config* i u rečenoj će se učiniti nekoliko promjena po već navedenim preporukama u poglavlju 2.2.3:

- *LoginGraceTime 30s* – ako se korisnik ne uspije autentificirati unutar 30 sekunda poslužitelj će sam od sebe zatvoriti vezu. Ova postavka može pomoći u smanjivanju opterećenja poslužitelja i sprječavanju napadaja grubom silom;
- *MaxAuthTries 6* – nakon što se korisnik ne uspije autentificirati šest puta za redom, poslužitelj će zatvoriti vezu. Isto tako pomaže zaštititi poslužitelj od napadaja grubom silom na zaporke. Valja napomenuti da se ovdje broji svaka raspoloživa metoda

autentifikacije pa ako se je moguće autentificirati zaporkom ili javnim ključem, nakon 3 neuspjelih pokušaja veza će se zatvoriti;

- *MaxStartups 10:50:30* – vrlo bitna postavka u sprječavanju napadaja grubom silom, a trenutačna konfiguracija predstavlja da nakon 10 neautentificiranih veza postoji izgled od 50% da će sljedeća veza biti odbačena, a pošto bude 30 takovih veza, sve će se odbacivati;
- *MaxSessions 2* – SSH podupire otvaranje više sjednica unutar jedne SSH-veze. Npr. korisnik može imati otvoren terminal i istodobno rabiti prijenos datoteka (SCP/SFTP) ili prosljeđivati vrata preko iste SSH-veze. Ova postavka ograničuje broj tih istodobnih sjednica na najviše dvije po već ostvarenoj SSH-vezi. Ovo može pomoći u sprječavanju prekomjerne porabe sredstava posluživača od jednoga korisnika;
- *PermitEmptyPasswords no* – kao što i sam naziv govori, korisnici koji nemaju postavljenu zaporku ne će se moći prijaviti jer se ne dopuštaju prazne zaporke;
- *LogLevel VERBOSE* – ako se žele bilježiti neuspjeli pokušaji autentifikacije moraju se postaviti postavke zapisivanja na opsežno. Zapisi se nahode u datoteci na stazi */var/log/auth.log*;
- *X11Forwarding no* – kako se na posluživaču ne rabe aplikacije s slikovnim sučeljem nego posluživač prvenstveno služi za izvršivanje tekstovnih naredaba, nema potrebe da se omogućuje prosljeđivanje slikovnoga posluživača X11;
- *PermitRootLogin prohibit-password* – upozorava se da se korijenski korisnik može prijaviti preko SSH-sučelja, ali ne zaporkom nego će morati rabiti koju sigurniju metodu poput prijave s pomoću para javnih i privatnih SSH-ključeva;
- *Banner /etc/ssh/banner.txt* – iako nije izravno povezano s sigurnošću, moguće je dodati kakovu bilo poruku koja će se pojaviti prigodom prijave na posluživač prije same autentifikacije. Sljedeći je takozvani natpis „*ASCII art*“ stvoren je s pomoću internetske usluge fsymbols raspoložive na *web*-stranici https://fsymbols.com/generators/carty/.

Slika 16. *Banner* "Foishka" stvoren preko *web*-usluge fsymbols

```
Match Group studenti
    DisableForwarding yes
    AllowTcpForwarding no
    PermitTunnel no
```

Prijašnji se dio dodaje na dno konfiguracijske datoteke, a samo je djelovanje isto tako prilično razumljivo. Za sve se korisnike koji su u skupini „*studenti*“ isključuju sve vrste prosljeđivanja i tuneliranja.

```
Match User root
    AuthenticationMethods publickey
```

Za korijenskoga će korisnika jedina metoda prijave biti s pomoću javnih ključeva dok će sve ostale metode biti onemogućene. Ova je metoda preporučljiva za sve korisnike, ali je nestvarno očekivati da će svi korisnici poslužitelja samostalno izroditi parove SSH-ključeva i prenijeti ih na poslužitelj jer to ipak zahtijeva određeno tehničko znanje, a i postoji velika mogućnost za pogrješku tako da korisnici manje prijetnje i najmanjih ovlasti (slušači) mogu nastaviti s prijavom preko zaporke. U kratko, za korisnika za kojega se želi omogućiti metoda prijave s pomoću javnih ključeva prvo treba izroditi par privatnih i javnih ključeva na lokalnom računalu s pomoću naredbe *ssh-keygen,* pohraniti ih na zadanu lokaciju „*~/.ssh*“ u datoteke *id_rsa* ili *id_rsa.pub* ili na koju drugu lokaciju s nazivima po želji. Bitan je dalji korak, a to je da se javni ključ prepiše u datoteku „*~/.ssh/authorized_keys*“, a kod SSH-prijave upozorava se da se rabi prijava preko javnih ključeva i u SSH-prijavu uključi se ovaj privatni ključ.

Iako nepovezano s konfiguracijskom datotekom SSH-poslužitelja, moguće je i ograničiti hitrost kojom jedna IP-adresa može uspostavljati nove SSH-veze konfiguriranjem „nezamršenoga vatrozida“ (*ufw – uncomplicated firewall*). Ako se koja IP-adresa pokuša povezati više od 10 puta u 30 sekunda, ni jedan sljedeći pokušaj ne će uspjeti jer će se veze prekinuti. Pravilo se dodaje vatrozidu pokretanjem jedne naredbe:

```
sudo ufw limit ssh
```

Isto tako, za korisnike koji imaju korijenski pristup ili ovlasti sudo, postavit će se da se sama od sebe gasi SSH-veza nakon određenoga vremena nedjelatnosti. U sustavsku će se datoteku */etc/profile*, koja se ukrcava za sve korisnike, dodati malena *bash*-skripta kako je *bash* ljuska terminala:

```
if id -nG "$USER" | grep -qw "profesori\|root"; then
    export TMOUT=60
else
    export TMOUT=600
fi
```

Naredba *id* vraća skupine korisnika za trenutačnog korisnika jer je u lokalnoj varijabli okružja pohranjena vrijednost trenutačnoga korisnika. Argumenti *-nG* obilježavaju da se brojčane vrijednosti skupina pretvaraju u nazive skupina koje se za tim preusmjere na naredbu *grep* koja pregledava dobivene skupine jesu li to cijeli nizovi „*profesori*“ ili „*root*“ s pomoću argumenta *-w*, a *-q* mijenja rad *grep*-naredbe u „tihi“ (*quiet*) koji kao nalaz šalje kodove stanja. U Unixovskim operacijskim sustavima kod stanja *0* predstavlja uspjeh tako da se, kad se pronađe željena skupina korisnika, postavlja varijabla okružja na 60 sekunda, a inače na 600 sekunda što znamenuje da će SSH-veza nastavnika i korijenskoga korisnika biti sama od sebe zatvorena nakon 60 sekunda nedjelatnosti u *bash*-ljuski.

4.2. Daljinski jednostavni pristup SSH-posluživaču

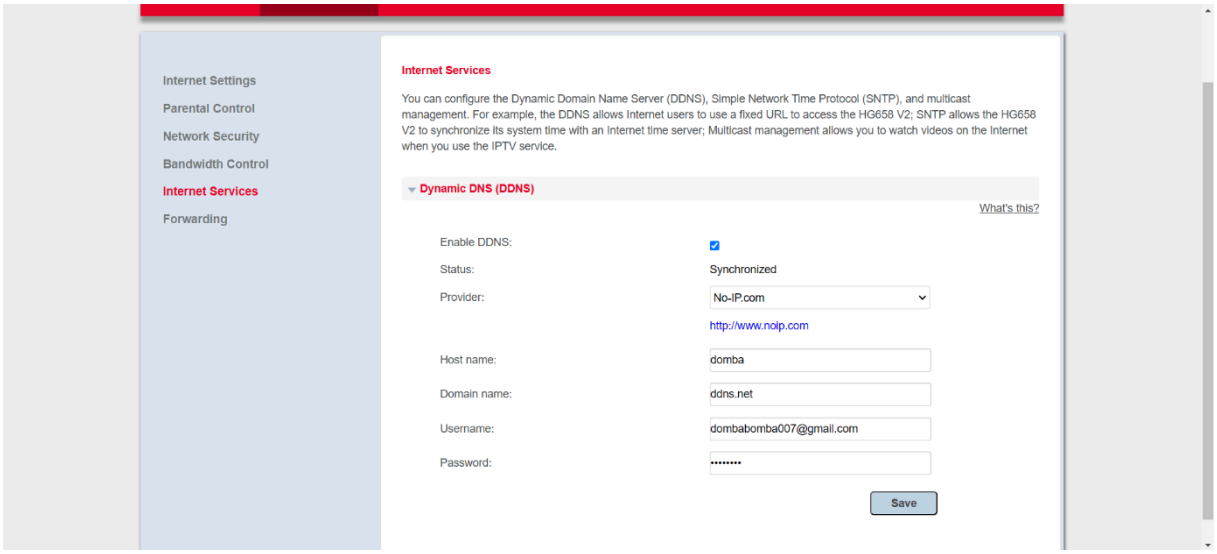
Kako je trenutačno pristup SSH-posluživaču moguć samo uz poznavanje precizne IP-adrese posluživača ili iz lokalne mreže, treba se omogućiti da korisnici ne trebaju znati preciznu IP-adresu kako se ona kod fiksnoga/kućnoga Interneta mijenja često, a glavninom svaki dan u polnoći ili po kojim drugim postavkama ISP-a. To se može učiniti na 2 načina – s pomoću plaćenih usluga Dynamic DNS ili će se pak stvoriti način za samostalno osvježavanje zapisa DNS A za određenu domenu ili poddomenu.

4.2.1. Dynamic DNS (DDNS)

Dynamic DNS (Dynamic Domain Name System) usluga je koja sama od sebe osvježava IP-adresu s povezanom domenom u DNS-sustavu. Rabi se za održavanje povezanosti među dinamičnim IP-adresama i statičnim domenama. Dynamic DNS omogućuje metode osvježavanja DNS-zapisa u stvarnom vremenu, a prvenstveno se rabi za povezivanje domene s mrežom koja nema statičnu IP-adresu. Tako se osigurava da željeni DNS-zapis uvijek pokazuje na ispravnu IP-adresu, pače i kad se ta IP-adresa promijeni. Neke od najznamenitijih usluga Dynamic DNS jesu No-IP, DynDNS, DuckDNS [23].

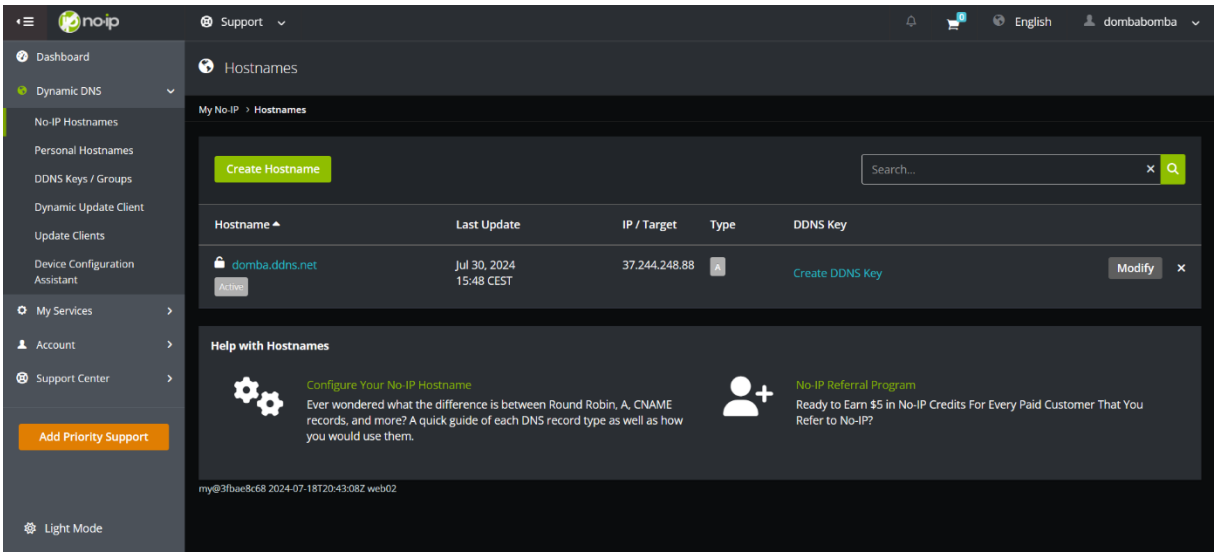
4.2.1.1. No-IP Dynamic DNS

No-IP pruža besplatne usluge Dynamic DNS za svoje korisnike ter i besplatnu domenu lako pamtljiva naziva u obličju *naziv-domene.ddns.net* ili koju drugu kombinaciju od nekolicine drugih ponuđenih domena kao što su *hopto.org*, *zapto.org* ili *sytes.net*. Nakon registracije računa na *web*-stranici *no-ip.com* i registracije željene slobodne domene, preostaje ili preuzeti njihovu aplikaciju koja mora raditi u podlozi računala ili posluživača ili pak u postavkama usmjernika (ako on to omogućuje) omogućiti sinkronizaciju s uslugom Dynamic DNS kao što je No-IP [24].



Slika 17. Konfiguracija usluge Dynamic DNS u administratorskom *web*-sučelju usmjernika

Ovaj usmjernik podupire No-IP Dynamic DNS pa su se postavile postavke tako da se ne treba preuzimati nikakov dodatan program za rad u podlozi i potrošnju sredstava. Za napomenuti je da je ako se imade uključena dvočimbenična autentifikacija za račun No-IP potrebno rabiti DDNS-ključ ili isključiti dvočimbenična autentifikacija ako se želi rabiti željena metoda osvježavanja Dynamic DNS-a preko usmjernika. DDNS-ključ koji izrodi usluga No-IP za pravo je novo korisničko ime i zaporka koja omogućuje preinaku samo toga određenoga zapisa tako da glavni račun ostaje zaštićen dvočimbeničnom autentifikacijom. U tom slučaju ne upisuje se ova glavna domena, nego *all.ddnskey.com*.



Slika 18. Usluga Dynamic DNS No-IP.com

U administratorskom sučelju usluge No-IP vidi se da ova poddomena *domba.ddns.net* sad imade vrijednost IP-adrese ovoga usmjernika dotično ove mreže ter će se kod svake promjene IP-adrese rečena osvježiti i za zapis *domba.ddns.net*. Da bi se pregledalo razrješuje li se DNS-zapis za *domba.ddns.net* u istinu na ovu IP-adresu može se rabiti oruđe *dig* za pretragu DNS-zapisa:

```
dig domba.ddns.net
      …
;; ANSWER SECTION:
domba.ddns.net.          60      IN      A       37.244.248.88
```

ili *nslookup* kao jednakovrijednica na Windowsima:

```
nslookup domba.ddns.net
```

4.2.1.2. Stvaranje vlastite skripte Dynamic DNS-a

Kako je glavnina usluga Dynamic DNS plaćena, a besplatne inačice rečenih imaju različne ograničenosti u smislu odabira same domene, broja DNS-zapisa koje mogu osvježiti ili pak potvrđivanja domene svakih 30 dana i sl., za stvaranje je samostalnoga dinamičnoga DNS-a potrebna domena i ugošćivanje DNS-a za tu domenu. Svaka fizička osoba s prebivalištem ili državljanstvom u RH ima pravo na besplatnu domenu u obličju *ime-prezime.from.hr*, a rečenu je moguće zatražiti na *web*-stranici *domene.hr*. Nakon registracije domene rečenu domenu treba smjestiti na ugošćivanje DNS-a, a jedno je od najznamenitijih ter značajkama i cjelokupnom uslugom najbogatije Cloudflare. Cloudflare osim ugošćivanja DNS-a pruža i usluge svjetskoga CDN-a, vatrozida u oblaku ter različne druge usluge za sigurnost i poboljšavanje hitrosti svih vrsta aplikacija – zaštitu od DDoS-napadaja, WAF, proizvode za sigurnost temeljene na načelima „*zero trust*“ i brojne druge [25, 26].

Kad se imade domena i kad se je rečena smjestila na Cloudflareove DNS-poslužiteljske, potrebno je stvoriti i ključ za Cloudflare API jer će se preko API-poziva dinamično osvježavati ovi zapisi DNS A za domenu čim će se takoreći postignuti iste mogućnosti koje pružaju i usluge za dinamični DNS. Za tim će se stvoriti skripta za dinamični DNS:

```
#!/bin/bash
LC_TIME=hr_HR.UTF-8
ip_current="$(dig +short txt ch whoami.cloudflare @1.1.1.1 | tr -d '\"')"
ip_saved="$(cat /home/domba/skripte/ip_current.txt)"
echo -e "Trenutna IP adresa: $ip_current\n"
echo -e "Zadnje ažurirana IP adresa: $ip_saved\n"
```

```
API_TOKEN="XXX"
ZONE_ID="XXX"
RECORD_NAME="ssh.dominik-lackovic.from.hr"
if [ "$ip_current" != "$ip_saved" ]
then
    echo -e "Nova IP adresa detektirana. Ažuriram $RECORD_NAME A zapis na novu IP adresu - $ip_current\n"
    RECORD_ID=$(curl -s -X GET "https://api.cloudflare.com/client/v4/zones/$ZONE_ID/dns_records" \
        -H "Authorization: Bearer $API_TOKEN" | jq -r '.result[] | select(.name == "'$RECORD_NAME'") | .id')
    current_date_time=$(date "+%A, %d. %B %Y. %H:%M:%S")
    response=$(curl -s -X PUT "https://api.cloudflare.com/client/v4/zones/$ZONE_ID/dns_records/$RECORD_ID" \
        -H "Authorization: Bearer $API_TOKEN" \
        -H "Content-Type: application/json" \
        --data "{\"type\":\"A\",\"name\":\"$RECORD_NAME\",\"content\":\"$ip_current\",\"comment\":\"Updated on: $current_date_time\"}")
    success=$(echo "$response" | jq -r '.success')
    errors=$(echo "$response" | jq -r '.errors[]')
    if [ "$success" = "true" ]; then
       echo "✅ Uspješno ažuriran DNS zapis."
       echo "$ip_current" > /home/domba/skripte/ip_current.txt
    else
       echo -e "❌ DNS zapis nije ažuriran. Dogodile su se pogreške:\n"
       echo "$errors"
fi
else
echo "Stara IP adresa. Ne radim ništa..."
fi
```

Na početku se skripte određuju lokalne varijable okružja na hrvatsko obličje jer će se poslije rabiti doba osvježavanja skripte pa da obličje vremena bude lokalizirano. U varijablu se *ip_current* pohranjuje nalaz izvršivanja skupa naredaba gdje se s pomoću oruđa *dig* čini upit na Cloudflareov DNS-posluživač (1.1.1.1) jer će on vratiti trenutačnu IP-adresu posluživača za TXT-zapis u razredu *ch* (CHAOS) za upit *whoami.cloudflare*. Rabi se kratak zapis kako bi se moglo manipulirati tim podatkom, a kako se IP-adresa vraća u navodnicima ("*37.244.248.88*") rečenu je potrebno odvojiti od navodnika s pomoću oruđa *tr* gdje se upozorava da je razdjelnik upravo znak navodnik. Nakon toga se ono što imade u ispisu datoteke *current_ip.txt* pohranjuje u varijablu *ip_saved* kako bi se poslije mogle usporediti te dvije varijable jer svako sučelje API-

poziva ima određene ograničenosti i nije dobra praksa pozivati ih svaki put pače i dok nema nikakovih promjena. Za tim se postavlja *API token* koji je zbog sigurnosnih razloga redigiran, ID zone koja se rabi u Cloudflareu i naziv samoga zapisa koji se želi pregledati i po potrebi osvježiti. Ako se IP-adresa razlikuje od pohranjene IP-adrese dotično od one IP-adrese na koju je zadnje DNS-zapis bio osvježen, onda se ispisuje poruka da je nova IP-adresa otkrivena i da se DNS-zapis osvježava onim što imade u novoj IP-adresi. Da bi se osvježio traženi DNS-zapis, prvo se mora pronaći ID samoga zapisa, a to će se dobiti API-pozivom gdje se kao nalaz dobivaju svi DNS-zapisi koji postoje u ovoj DNS-zoni. Kod poziva se API-ja dotično *curl*a rabi argument *-s* (*silent*) da se ne ispisuju nikakove poruke ili pogrješke da se može dalje manipulirati stanjem i podatcima koje vraća API. Argumentom se *-x* postavlja HTTP-metoda na GET, a argumentom se *-H* postavlja u sam zahtjev API-poziva zaglavlje s autorizacijskim *API tokenom*. Kako Cloudflareov API vraća JSON, izlaz će se naredbe API-poziva preusmjeriti na oruđe *jq* jer je to oruđe za rad s JSON-podatcima koje omogućuje pročišćivanje, pretraživanje i manipulaciju JSON-podataka na jednostavan i učinkovit način. *jqu* argument *-r* obilježava da je izlaz naredbe u sirovu obličju dotično bez navodnika što je korisno za dalju manipulaciju i porabu u skripti ljuske. Za tim se iterira kroz niz „*result*“ dotično kroz sve DNS-zapise i pročišćuju se samo zapisi niza koji imaju u sebi ime jednako traženomu zapisu i konačno se vuče polje „*id*“ iz pročišćenoga objekta i pohranjuje kao nalaz naredbe u varijablu „*RECORD_ID*“. Za tim se postavlja varijabla „*current_date_time*“ na trenutačni nadnevak i vrijeme u obličju "*utorak, 30. srpnja 2024. 22:21:32*", a to će se rabiti u sljedećem API-pozivu. API-pozivi za osvježavanje obično rabe HTTP-metodu „PUT“ i ovaj se put dodaje novo zaglavlje s JSON-podacima u obličju kakvo je potrebno API-ju. Ako je vrijeme pohranjeno u pogrješnom obličju, potrebno je izvršiti naredbu za promjenu vremenske zone:

```
timedatectl set-timezone Europe/Zagreb.
```

Dakle, vrijednost se željenoga DNS-zapisa postavlja na trenutačnu IP-adresu, a kao komentar se dodaju doba osvježavanja. Za tim se na kraju parsira JSON-odgovor i u varijablu „*success*“ pohranjuje vrijednost niza *success* koja treba biti „*true*“, a u *errors* se pohranjuju pogrješke ako se rečene pojave. Ako je poziv bio uspješan, ispisuje se poruka da je DNS-zapis uspješno osvježen i pohranjuje se IP-adresa lokalno u datoteku kako se ne bi moralo svaki put pregledavati i činiti nov API-poziv ili DNS-zahtjev. Datoteci je skripte potrebno dodati dopusnicu za izvršivanje:

```
chmod +x update_cf_dns.sh
```

chmod je naredba za mijenjanje dopusnica datoteka, a argument *+x* predstavlja da se želi samo dodati dopusnica za izvršivanje. Skripta se može pokrenuti ručnim pokretanjem

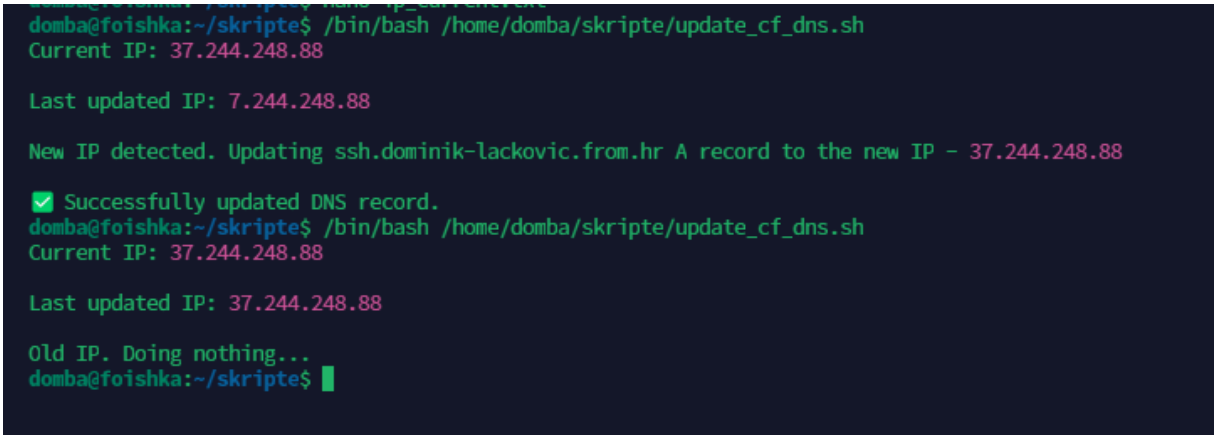
```
./update_cf_dns.sh
```

Da bi se skripta izvršivala svake minute dotično da bi se svake minute pregledavalo je li se IP-adresa pregledala i osvježio DNS-zapis, potrebno je urediti *cron* (raspoređivač zadataka) i dodati sljedeći zapis:

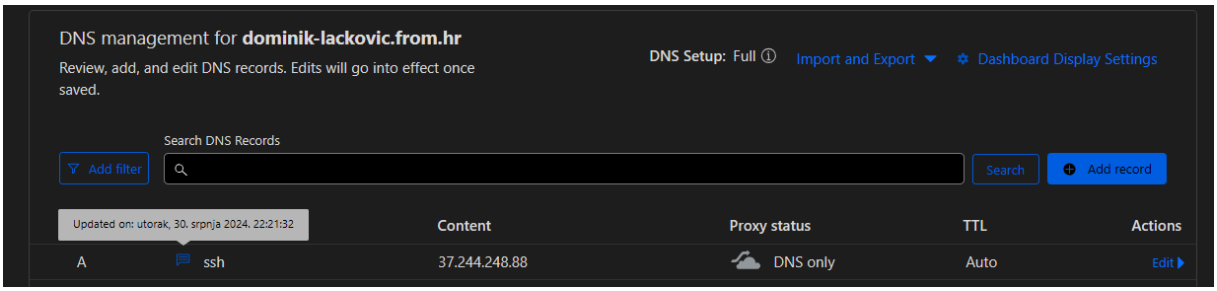
`crontab -e` (uređivanje *crona*)

```
* * * * * /bin/bash /home/domba/skripte/update_cf_dns.sh >
/home/domba/skripte/update_cf_dns.log
```

cron je vrlo osjetljiv na nekim sustavima pa je najbolje uvijek upozoriti na preciznu stazu programa koji treba pokrenuti koju naredbu ili skriptu. Dakle, rabeći *bash* svake minute izvršit će se ova skripta, a zapis će se izvršivanja pohraniti u zapisničku datoteku. Ako bi bilo potrebno još hitrije osvježavanje dotično pregled je li došlo do promjene IP-adrese, to bi trebalo učiniti u *while*-petlji u samoj skripti ljuske.



Slika 19. Izvršavanje skripte za ažuriranje zapisa DNS A



Slika 20. Cloudflareovo korisničko *web*-sučelje – upravljanje DNS-zapisima – osvježeni zapis i komentar zapisa dotično nadnevak i vrijeme stvaranja

4.3. Konfiguracija poslužitelja i korisnikove dopusnice

Osnovna bi svrha ovoga poslužitelja bila rad nastavnika („profesori“) i slušača („studenti“). Nastavnici bi trebali imati uvid u sve datoteke u osobnim mapama slušača, a slušači bi imali pristup samo svojoj korisničkoj mapi iz razloga da ne mogu prepisati tuđe rješidbe ako bi se rješavao kakov zadatak.

Prvo će se stvoriti skupine *profesori* i *studenti*:

```
sudo groupadd profesori
sudo groupadd studenti
```

Kako bi se ubrzao postupak dodavanja nastavnika i slušača, pripravit će se 2 CSV-datoteke koje ćemo se rabiti za uvoz korisnika. CSV-datoteka treba biti u jednostavnu obličju: *<ime>,<prezime>,<korisničko ime>,<zaporka>*.

Sad se može stvoriti skripta za uvoz slušača naziva *import_studenti.sh*.

```
#!/bin/bash
GRUPA="studenti"
CSV_FILE="studenti.csv"
while IFS=, read -r ime prezime username lozinka; do
    sudo useradd -m -p $(openssl passwd -1 $lozinka) -c "$ime $prezime" -s /bin/bash $username
    sudo usermod -aG $GRUPA $username
    sudo chown $username:profesori /home/$username
    sudo chmod 770 /home/$username
      sudo chmod g+s /home/$username
      echo "umask 007" | sudo tee -a /home/$username/.profile >/dev/null
      echo -e "✅ Korisnik $ime $prezime ($username) uspješno kreiran i dodan u grupu $GRUPA!"
done < $CSV_FILE
```

Ova skripta u *while*-petlji postavlja razdjelnik dotično IFS (Internal Field Separator) na znak zareza (,), čita svaki redak u CSV-datoteci i vrijednosti pohranjuje u varijable *ime*, *prezime*, *username* i *lozinka*. Za tim stvara nova korisnika gdje argument *-m* obilježava da će se stvoriti i kazalo „*home*“ za novoga korisnika. *-p* postavlja zaporku (*password*), a rečena će se kriptirati rabeći algoritam *-1* što u ovom slučaju predstavlja algoritam MD5. U komentar će se korisnika pohraniti ime i prezime ter će se odrediti ljuska korisnika koja će biti *bash* i na kraju se isto tako određuje i korisničko ime. Za tim se s pomoću naredbe *usermod* korisnik dodaje u skupinu „*studenti*“, mijenja se vlasništvo njegova kazala „*home*“ da vlasnik bude on, a skupina u *profesori* zato što se želi da profesori imaju potpun pristup svim kazalima slušača.

Iako su *profesori* dio *sudo*-skupine, ovim se pojednostavljuje postupak jer ne trebaju upisivati zaporku dok budu radili s kazalima slušača. Isto se tako mijenjaju i dopusnice korisničkoga kazala na *770* što znamenuje da vlasnik i korisnici skupine imaju potpun pristup (čitanje, pisanje, izvršivanje), a svi ostali korisnici nemaju baš nikakov pristup pa pače ni pristup čitanju tako da će im pristup biti zabranjen. Naredba *chmod g+s* postavlja *setgid*-bit na korisničko kazalo tako da sve nove datoteke i kazala unutar njega nasljeđuju skupinu "*profesori*". Na kraj se datoteke *.profile* dodaje i naredba *umask 007* koja osigurava da nove datoteke koje korisnik stvori ne će imati dopusnice za druge korisnike (samo vlasnik i skupina imaju dopusnice) i na samom se kraju ispisuje poruka o uspješnu dodavanju novoga korisnika.

Skripta je za dodavanje nastavnika istovjetna prijašnjoj, ali ima dodanu jedino naredbu za dodavanje nastavnika u *sudo*-skupinu:

```
sudo usermod -aG sudo $username
```

Te su se dvije skripte mogle spojiti u jednu, ali bi se onda i CSV-datoteke trebale spojiti ter dodati nov redak s skupinom u CSV-datoteku kako bi se moglo pregledati koji se korisnik treba smjestiti u koju skupinu.

Ako se ne želi rabiti CSV-uvoz ili se pak želi naknadno dodati samo jedan korisnik, može se rabiti sljedeća skriptu koja uzimlje ulazne argumente kod poziva skripte:

```
#!/bin/bash
if [ "$#" -ne 5 ]; then
    echo -e "❌ Kriva upotreba. Potrebno je točno 5 argumenata.\nIspravan primjer: $0 <ime> <prezime> <username> <lozinka> <grupa>"
    exit 1
fi
IME="$1"
PREZIME="$2"
USERNAME="$3"
LOZINKA="$4"
GRUPA="$5"
if [ "$GRUPA" != "studenti" ] && [ "$GRUPA" != "profesori" ]; then
    echo "Greška: Grupa mora biti 'studenti' ili 'profesori'"
    exit 1
fi
sudo useradd -m -p $(openssl passwd -1 $LOZINKA) -c "$IME $PREZIME" -s /bin/bash $USERNAME
sudo usermod -aG $GRUPA $USERNAME
if [ "$GRUPA" == "profesori" ]; then
    sudo usermod -aG sudo $USERNAME
```

```
fi
sudo chown $USERNAME:profesori /home/$USERNAME
sudo chmod 770 /home/$USERNAME
sudo chmod g+s /home/$USERNAME
echo "umask 007" | sudo tee -a /home/$USERNAME/.profile >/dev/null
echo -e "✅ Korisnik $IME $PREZIME ($USERNAME) uspješno dodan!"
```

Skripta je vrlo slična prijašnjoj, a glavna je promjena način porabe, pregled je li upisano svih 5 potrebnih argumenata i je li upisana ispravna skupina:

```
./dodaj_korisnika.sh „Ime“ „Prezime“ „korisnicko_ime“ „lozinka“ „studenti“
```

4.4. Pozdravna poruka kod prijave

Prigodom se prijave na SSH-posluživač želi da se ispisuje pozdravna poruka, ali različna za svaku skupinu. Stvorit će se nova skripta na stazi */etc/profile.d/login_message.sh* i dodijeliti joj dopusnica za izvršivanje *chmod +x login_message.sh*.

```
#!/bin/bash
user_firstname=$(getent passwd $USER | cut -d: -f5 | cut -d" " -f1)
num_students=$(getent group studenti | awk -F: '{print $4}' | tr ',' '\n' | wc -l)
num_profs=$(getent group profesori | awk -F: '{print $4}' | tr ',' '\n' | wc -l)
if groups $USER | grep -q '\broot\b'; then
    echo -e "\n----------------------------------------"
    echo -e "Pozdrav SUPERADMIN $user_firstname ($USER) 👋 \nPazi kaj delaš!"
    echo -e "'With great power comes great responsibility!' 😳\n"
    echo "Trenutno je $num_students korisnika u grupi 'studenti'."
    echo "----------------------------------------"
    echo "Trenutno je $num_profs korisnika u grupi 'profesori'."
    echo "----------------------------------------"
elif groups $USER | grep -q '\bprofesori\b' && ! groups $USER | grep -q '\broot\b'; then
    echo -e "\n----------------------------------------"
    echo -e "Pozdrav $user_firstname ($USER) 👋 \n"
    echo -e "❗ Kao profesor imate sudo pristup ❗"
    echo -e "'With great power comes great responsibility!' 😳\n"
    echo "Trenutno je $num_students korisnika u grupi 'studenti'."
    echo "----------------------------------------"
```

```
else
    echo -e "\n----------------------------------------"
    echo -e "Pozdrav $user_firstname ($USER) 👋🎓 \n"
    echo "----------------------------------------"
fi
```

Prvo se vuče korisnikovo ime rabeći naredbu *getent passwd $USER*. Ovdje se iz datoteke */etc/passwd* pronađe redak s ovim korisničkim imenom jer je *$USER* varijabla okružja koja ima u sebi trenutačno korisničko ime korisnika. Da bi se izvadilo samo ime, mora se prvo odvojiti redak razdjelnikom „:“ i uzeti 5. član nakon razdvajanja i dodatno razdvojiti po razmaku da se dobije ime. Slična stvar je i s brojem slušača i nastavnika. Iz popisa se skupine ispisuju svi korisnici i potanki podatci o skupini kao što su ID skupine, ali se treba samo četvrti dio koji je odvojen razdjelnikom navodnika i za tim se transliteriraju zarezi znakom za nov red i na kraju se prebroji broj redaka.

Ako se korisnik nahodi u skupini „*root*“ onda se ispisuje poruka s brojem korisnika koji su u skupini *studenti* i *profesori*. Naredba *grep* argumentom *-q* daje izlazne kodove uspjeha (*0*) ili neuspjeha (*1* ili više) dok „*\b*“ obilježava da se radi o pretraživanju cijeloga niza, a ne djelomična podudaranja. Ako se korisnik ne nahodi u *root*-skupini, ali se nahodi u skupini *profesori*, ispisat će se poruka upozorbe da se kao profesor imade *sudo*-pristup i ukupni broj korisnika u skupini slušača, dok će studenti dobiti samo običnu pozdravnu poruku.

5. Ispitivanje posluživača i alternativni načini pristupa SSH-posluživaču

Za pristup je ovomu ili kojemu bilo drugomu SSH-posluživaču potreban SSH-klijent. Kako bi se pristupilo ovomu posluživaču potrebno je prvo odlučiti se preko koje će se metode na rečeni povezati – rabeći izravan pristup ili koji od alternativnih načina koji će se poslije opisati, a rješavaju teškoću zatvorenih vrata na mreži. Glavnina SSH-klijenata omogućuje pohranu različnih profila tako da je moguće konfigurirati profile za sve vrste veze koji će se rabiti u različnim zgodama. Jedan od mogućih načina pristupa ovomu SSH-posluživaču jest sljedeći:

```
ssh domba@ssh.dominik-lackovic.from.hr -p 2222
```

Kad je ostvarena SSH-veza, korisnici mogu izvršivati različne naredbe i zadatke s obzirom na razinu ovlasti koje imaju, a administratori mogu izvršivati i skripte za dodavanje novih korisnika ili nastavnika. Ispitivanje se je SSH-posluživača činilo na nekolicini SSH-klijenata i sažele su se glavne značajke svakoga klijenta kako bi se dobio pregled za lakši odabir.

5.1. SSH-klijenti

Postoje brojne rješidbe ostvaraja SSH-protokola na klijentskoj strani tako da je od iznimne bitnosti odabrati SSH-klijent koji odgovara potrebama. Dobar SSH-klijent imade sljedeće značajke:

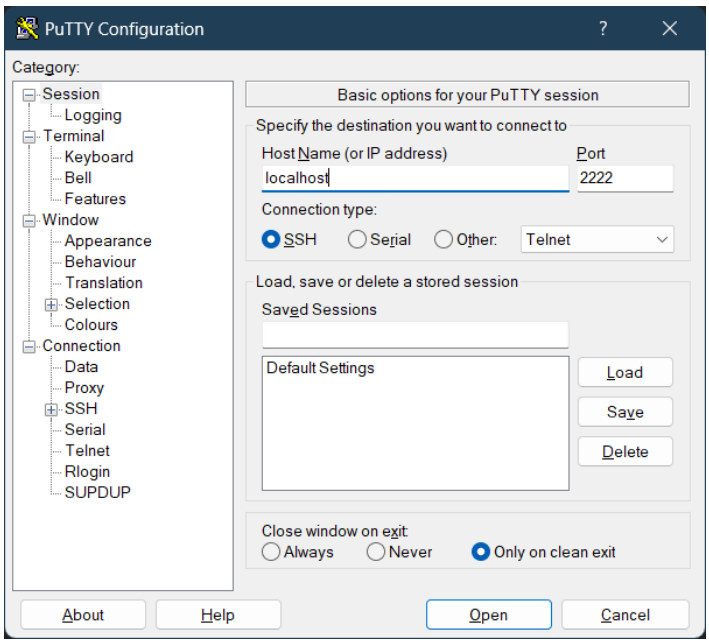
- autentifikacija - autentifikacija preko različnih metoda kao što su autentifikacija preko kombinacije korisničkoga imena i zaporke, autentifikacija preko para javnih ključeva i autentifikacija preko certifikata;
- enkripcija – poraba simetrične i asimetrične kriptografije;
- potpora za stariji protokol – zbog suradljivosti je potreban i protokol SSH1;
- tuneliranje – stvaranje SSH-tunela i prosljeđivanje vrata ili uspostava veze na nestandardna vrata;
- slikovno sučelje/tekstovno sučelje – slikovno je sučelje možebiti prikladnije za korisnike koji nisu izrazito tehnički potkovani dok je tekstovno sučelje siguran i hitriji odabir za sve prave sustavske administratore. Potpora je i jednomu i drugomu sučelju bitna kako bi se zadovoljile sve skupine korisnika;

- potpora za operacijske sustave – suradljivost zahtijeva da klijent radi dosljedno na svim operacijskim sustavima;
- prijenos datoteka – mogućnost prijenosa datoteka preko SCP-a (Secure Copy) ili SFTP-a (SSH File Transfer Protocol).

5.1.1. PuTTY

PuTTY je jedan od najznamenitijih SSH-klijenata na operacijskim sustavima Windows s *xterm*-emulatorom konzole. PuTTY je isto tako *telnet*-klijent i dohodi u paketu s mnoštvom dodatnih programa koji proširuju značajke. Izvorno je razvijen od Simona Tathama za Windows-platformu, a prva je inačica izdana 1999. godine. PuTTY je program otvorena koda što je za sigurno prinijelo njegovoj znamenitosti, a i dan-danas razvija se od dragovoljaca diljem svijeta. PuTTY dohodi kao paket različnih oruđa, svaki s posebnom mogućnošću [27]:

- PuTTY – glavni emulator terminala za SSH, Telnet i druge protokole;
- PuTTYgen – oruđe za izrađanje SSH-ključeva i upravljanje njima;
- PSCP – PuTTY Secure Copy, oruđe za sigurno preslikavanje datoteka preko SCP-a;
- PSFTP – PuTTY SFTP Client, oruđe za siguran prijenos datoteka preko SFTP-a;
- Plink – PuTTY Link, oruđe za automatizaciju SSH-naredaba preko naredbenoga retka;
- Pageant – SSH-agent za upravljanje privatnim ključevima u sjednicama;
- PuTTYtel – Telnet-inačica PuTTY-ja;
- pterm – samostalni emulator terminala koji je dio samo Unix-inačice PuTTY-ja.

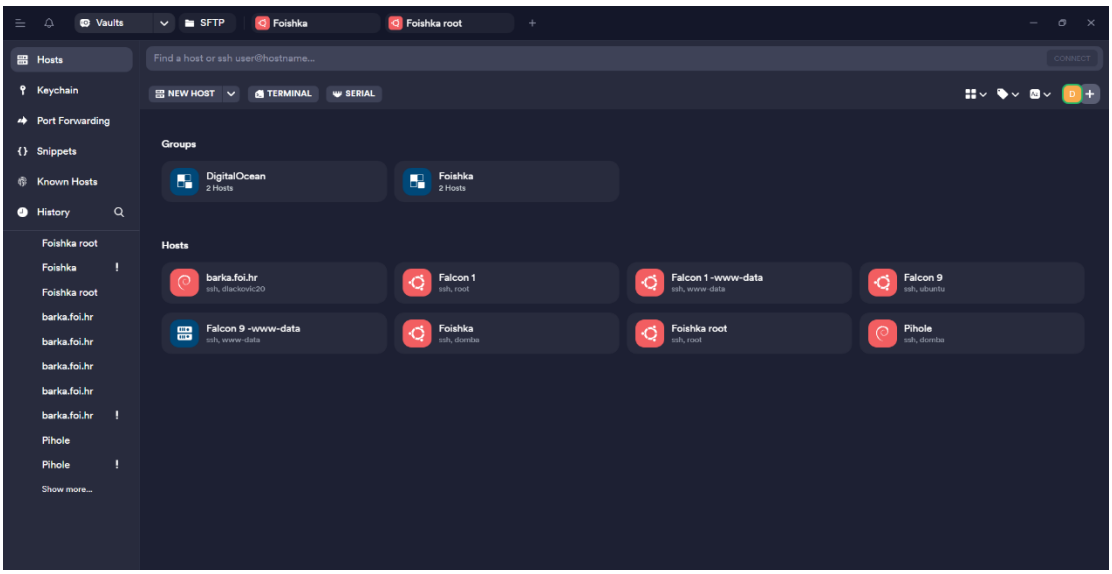


Slika 21. SSH-klijent PuTTY

Iako slikovno sučelje nije najsuvremenije, bogato je mogućnostima i podupire različne prilagodbe – od *fontova* i boja pokazivača pa do bilježenja sjednica u datoteke za naknadne raščlambe ili jednostavno kao pismohranu što se je činio i precizno kad na kojem SSH-posluživaču. Podupire različne enkripcijske algoritme poput AES-a, 3DES-a, Blowfisha, ChaChaa 20. Oruđe PuTTYgen koje dohodi u paketu služi za izrađanje privatnih i javnih ključeva RSA-a, DSA-a, ECDSA-a, Eda 25519 i drugih. Isto tako, omogućuje pohranu sjednice s različnim konfiguracijama tako da se može vrlo lako ukrcati sjednica s postavkama za određeni posluživač da se ne gubi vrijeme svaki put na postavljanje ili pisanje naredaba.

5.1.2. Termius

Termius je pravi suvremeni ostvaraj klijentskoga protokola SSH-a koji osim lijepa i oku ugodna sučelja ima i pregršt mogućnosti.



Slika 22. SSH-klijent Termius

SSH-posluživači mogu se logički posložiti u različne skupine, može se izvršiti sam od sebe uvoz konfiguracija za SSH-povezivanje ako se rabe usluge u oblaku kao što su AWS, DigitalOcean ili Microsoft Azure što znatno ubrzava rad jer se više ne treba gubiti vrijeme na svakakove SSH-konfiguracije nego je sve pohranjeno i pripravno za vezu na SSH-posluživač. Ostale glavne značajke Termiusa su: sinkronizacija osobnih i ekipnih „riznica“ na svim uređajima pa tako ako što iskrsne, može se i u pokretu se spojiti na SSH-posluživač s pomoću mobitela Android ili iOS. Podupire i biometrijske ključeve poput Windows Helloa ili pak

standard FIDO 2 za prijavu bez zaporaka s pomoću sigurnosnih ključeva kao što su YubiKey i sl.

Glavne su značajke Termiusa koje podižu produktivnost kod SSH-sjednica isječci koda odnosno naredaba koje je moguće pohraniti i pozivati po potrebi i tako više nije potrebno pamtiti određene zamršene dijelove naredaba nego je dovoljan jedan pritisak da se naredba izvrši ili po potrebi preinači prije izvršivanja. Te se iste naredbe iz isječaka koda mogu same od sebe pokretati pri prijavi na SSH-poslužitelj što može biti korisno kod ponavljajućih zadataka. Isto tako, Termius podupire pametno samopopunjanje i dovršivanje naredaba prigodom tipkanja. Npr. kad se želi prebaciti u koje kazalo, čim se počne pisati naziv on nudi sve mogućnosti u padajućem izborniku baš kao što to čini tipka tabulatora, ali ovdje nije potrebno ništa dodatno činiti ter se predlažu i naredbe iz povijesti. Postoji i AI-generator naredaba gdje se jednostavnim jezikom opiše koja se naredba želi i treba pa će AI-model pomoći u tom zadatku. Naravno da treba biti oprezan s naredbama koje se ne razumiju, a po gotovo ako se izvršuju kao korijenski korisnik.

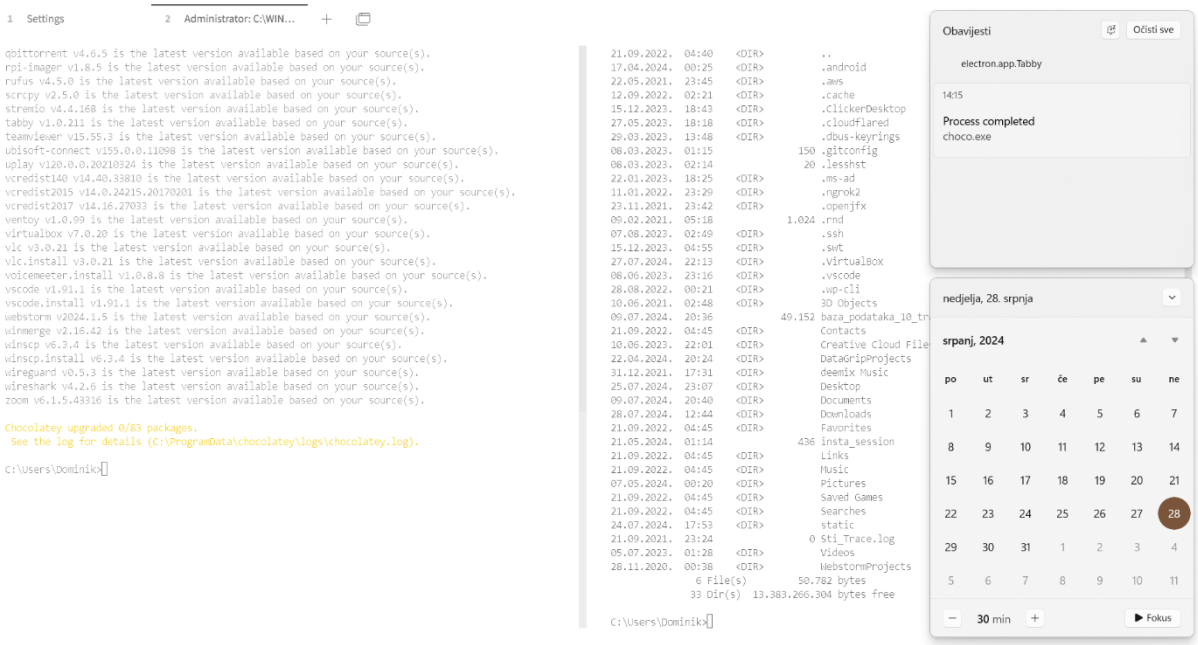
Termius podupire i različne tunele kao što su prosljeđivanje vrata (lokalno, daljinsko, dinamično), *HTTP proxy*, *SOCKS proxy*, suradnju u stvarnom vremenu, dijeljenje terminala preko poveznice tako da se ovaj terminal može podijeliti s drugim osobama i ulančavanje SSH-poslužitelja tako da se može spojiti izravno na određeni SSH-poslužitelj preko posrednika. Slično kao i argument *-J* ili izbor *ProxyJump* u inačici OpenSSH-a. Bitno napomenuti jest da je Termius potpuno besplatan za sve slušače koji su se prijavili za „GitHub Education Pack“ [28, 29].

5.1.3. Tabby

Tabby je visoko prilagodljiva višeplatformska aplikacija terminala otvorena koda za sve vrste veza SSH, Telnet i serijskih. Ono je što ju čini drugačijom od drugih činjenica da je otvorena koda i raspoloživi su različni dodatci za rečenu, a po potrebi svatko može izraditi i svoj dodatak kako bi bolje odgovarao određenomu slučaju porabe. Zbog toga, a i različnih mogućnosti, znamenita je kod suvremenih programera i sustavskih administratora. Po broju je funkcija najsličnija Termiusu pa će se nabrojiti i istaknuti samo ono što Termius ne nudi, a Tabby nudi [30]:

- pametne kartice – kartice s vezama SSH ili drugima koje otkrivaju naprjedak i šalju obavijesti na operacijski sustav kad je postupak gotov ili kad se pojavi nova djelatnost. Može biti korisno kad se čeka da se neki postupci koji obično dulje traju poput prevađanja ili instaliranja aplikacija završe tako da se u prednjici može činiti što drugo i, kad se zadatak završi, bude se obaviješten iste sekunde;

- povijest terminala i kartica – Tabby pamti otvorene kartice, a kad se slučajno zatvore, vraća potpuno stanje terminala kakovo je bilo kod zadnje veze;
- podijeljene kartice – radni se zaslon može slobodno preurediti na predodređena okna (gore, dolje, lijevo ili desno ili koja od tih kombinacija) tako da se na jednom dijelu zaslona recimo može pratiti stanje sustava, a na drugom mogu izvršivati ostale naredbe. Isto tako, takova se konfiguracija podijeljenih kartica može pohraniti kao račun kako bi se poslije moglo nastaviti raditi s željenim prikazom kartica;
- uklop u surječni izbornik Windows Explorera;
- *web*-aplikacija – terminal Tabby, ali kao *web*-aplikacija s samostalnim ugošćivanjem napisana u JavaScriptu. Isto tako pruža uslugu sinkronizacije konfiguracije među stolnim Tabby-klijentima.

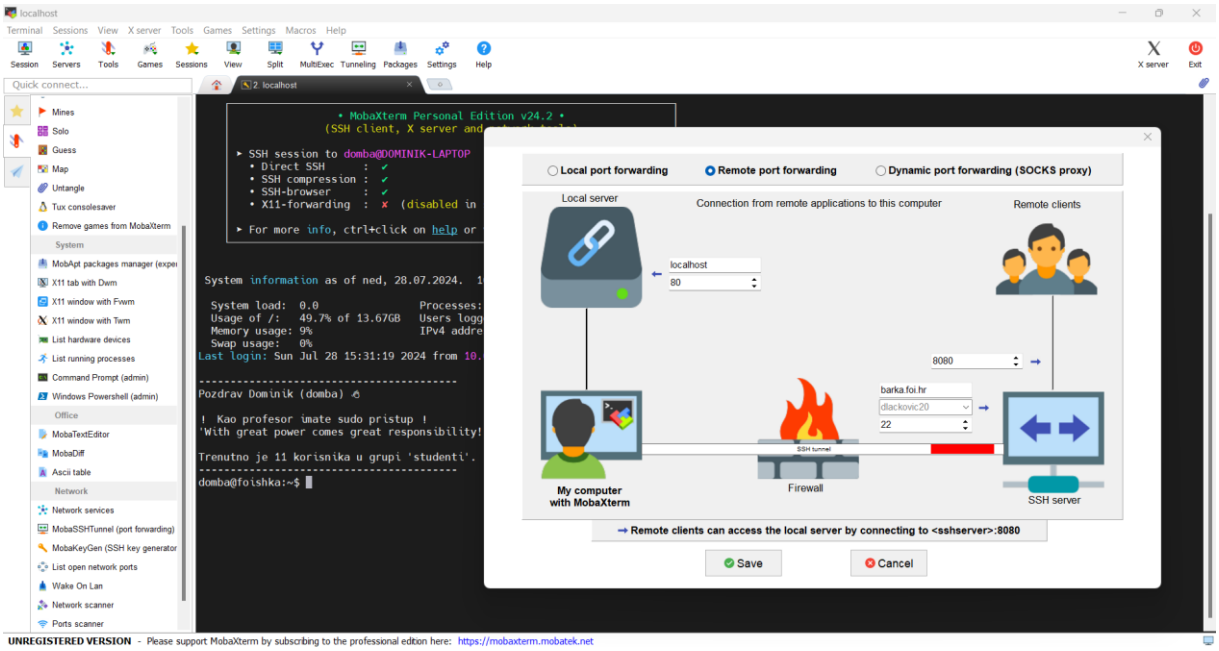


Slika 23. SSH-klijent Tabby

5.1.4. MobaXterm

MobaXterm isto je tako jedno od sveobuhvatnih oruđa za daljinski rad, osmišljeno samo za operacijski sustav Windows. Uklapa više mrežnih klijenta i oruđa za daljinski pristup kao što su SSH, Telnet, RDP, VNC, X11, SFTP pa pače i za uslugu AWS S3 skladištenja objekata u oblaku. MobaXterm razvila je francuska tvrtka Mobatek, a tijekom se je godina oruđe razvilo kako bi uključilo brojne nove mogućnosti i potporu za različne mrežne protokole, postajući jedan od najsveobuhvatnijih oruđa za daljinski rad i mrežno upravljanje [31].

MobaXterm tako imade i ugrađene poslužitelje koji omogućuju pokretanje mrežnih *daemona* za daljinski pristup. Nikakovo dodatno oruđe nije potrebno za pokretanje vlastitih poslužitelja TFTP, FTP, HTTP, SSH/SFTP, Telnet, NFS, VNC ili Iperf. Jedinstvena je mogućnost i višestruko izvršivanje naredaba. Naredba se upiše jedan put i odabere na kojim se poslužiteljima želi izvršiti čim se omogućuje izvršivanje istih naredaba na više različnih poslužitelja istodobno, a nalaz se izvršivanja može pratiti izravno jer se mogu podijeliti zasloni dotično veze na različne dijelove zaslona. Kako je MobaXterm specijaliziran za operacijske sustave Windows, ne čudi se činjenici da podupire i RDP-protokol (Remote Desktop Protocol) tako da se na hitar i jednostavan način može preuzeti nadzor i upravljanje nad daljinskim računalima/poslužiteljima Windowsa rabeći RDP-protokol. Isto tako, podupire i različne dodatke koji se mogu instalirati izravno s slikovnoga sučelja aplikacije, makronaredbe i različne druge funkcije. Program nudi toliko funkcija da bi se pače nazvao i prekrcanim jer su dodali i igre poput Minesweepera, Solitairea i Sudokua što nije razumljivo jer tomu nije mjesto u nekom SSH-klijentu. Svakako bi se pohvalilo slikovno i animirano stvaranje prosljeđivanja vrata jer se je kod stvaranja rečenih koji put zabunilo s konfiguracijom tako da je ovdje skoro pa nemoguće pogriješiti s konfiguracijom dok je sve lijepo slikovno rastumačeno.



Slika 24. SSH-klijent MobaXterm – prosljeđivanje vrata

5.2. Alternativni načini pristupa

Ako se iz kojega razloga ne mogu otvoriti vrata na ovoj mreži npr. kad se bude za NAT-om (CG-NAT-om), kad se bude ograničen s neke strane vatrozida ili jednostavno računalo nije izravno dohvatljivo na Internetu, mogu se rabiti različne usluge kako bi se ostvarila veza s izlaznom vezom poslužitelja gdje se rabimo drugi poslužitelj kao posrednik koji za tim prosljeđuje vezu natrag na ovaj poslužitelj preko SSH-prosljeđivanja vrata. Samo neke od usluga s pomoću kojih je isto tako moguće ostvariti željenu mogućnost jesu serveo.net, Cloudflare Tunnel, ngrok i Zero Tier.

5.2.1. serveo.net

serveo.net za pravo je samo preinačen SSH-poslužitelj koji služi kao posrednik kako bi se mogli izložiti lokalni poslužitelji k Internetu. Djeluje na način daljinskoga prosljeđivanja vrata, a njim je vrlo lako javno podijeliti lokalni poslužitelj. Npr. *serveo.net* može se rabiti kako bi se izložio lokalni *web*-poslužitelj što može biti korisno kad se želi ispitna *web*-stranica podijeliti s drugima dok još nije smještena na domeni jer *serveo.net* dodijeli poddomenu s koje se prosljeđuje promet s lokalnoga poslužitelja. Isto tako, može se i zatražiti određena poddomena ako je ona slobodna. Ako su vrata 22 zapriječena na ovoj mreži, *serveo* sluša i na vratima 443 koja vrlo vjerojatno nisu zapriječena kako se radi o HTTPS-vratima [32].

Za ove će se potrebe rabiti *serveo.net* kako bi se prosljeđivao SSH-promet. Potrebno je izravno na poslužitelju pokrenuti SSH-prosljeđivanje vrata, a ne preko vrata računala-domaćina:

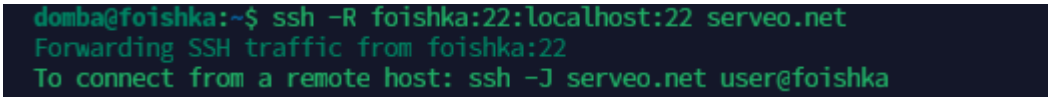
```
ssh -R foishka:22:localhost:22 serveo.net
```

Dakle, uspostavlja se SSH-veza s poslužiteljem *serveo.net*, a argument *-R* otvara i daljinsko prosljeđivanje vrata gdje će se na daljinskoj strani stvoriti takozvani *alias* dotično na naslovu će se *foishka* na vratima 22 prosljeđivati promet s ovoga lokalnoga SSH-poslužitelja.

Da bi se s drugoga računala uspostavila SSH-veza, potrebno je izvršiti naredbu:

```
ssh -J serveo.net domba@foishka
```

Mora se rabiti značajka „*JumpHost*“ (*-J*) poslužitelja OpenSSH koja konfigurira da se prvo učini SSH-veza put poslužitelja *serveo.net* s kojega će se za tim uspostaviti SSH s željenim korisnikom na željenoj adresi, a kako adresa *foishka* prosljeđuje promet na ovaj poslužitelj s pomoću SSH-prosljeđivanja vrata, uspješno se ostvaruje-SSH veza.



Slika 25. Pristup SSH-posluživaču preko „*JumpHosta*“ *serveo.net*

Ako dođe do promjene internetske veze, a doći će jer se rabi fiksni internet i ne plaća se skupa statična IP-adresa, veza će se navedenoga SSH-tunela prekinuti. Kako bi se osiguralo da se tunel dotično prosljeđivanje vrata samo od sebe ponovno pokrene, može se rabiti oruđe *autossh*. Kako ovo oruđe nije raspoloživo ni u jednom repozitoriju, potrebno je preuzeti izvorni kod i samostalno ga prevesti [33].

Sama je poraba oruđa *autossh* vrlo jednostavna:

```
autossh -M 0 -R foishka:22:localhost:22 serveo.net
```

Argument *-M 0* obilježava da se ne želi pregled veze oruđa *autossh* nego se želi da *autossh* uvijek radi i da se uvijek ponovno uspostavlja veza ako se dogodi kakov bilo prekid veze.

Kako se želi da se ta naredba izvršuje uvijek, dotično da se pokrene po pokretanju posluživača, potrebno je stvoriti novu uslugu na posluživaču. Stvori se nova datoteka dotično usluga na stazi */etc/systemd/system/autossh.service* s ovim:

```
[Unit]
Description=AutoSSH Service
After=network.target

[Service]
User=domba
ExecStart=/usr/local/bin/autossh -M 0 -R foishka:22:localhost:22 serveo.net
Restart=always

[Install]
WantedBy=multi-user.target
```

U odjeljku se „*Unit*“ određuje naziv usluge i redoslijed pokretanja usluga dotično da se SSH-veza može početi uspostavljati tek pošto je pokrenuta usluga za mrežu.

U odjeljku se „*Service*“ određuje pod kojim će se korisnikom pokrenuti ova nova usluga, a *ExecStart* za pravo je ova naredba koja će se pokretati gdje je potrebno napisati punu stazu do izvršne datoteke *autossh*. Napomenuti je da naredba ne će biti izvršena ako u datoteci *known_hosts* ne postoji zapis za *serveo.net* tako da se prvo može pokušati se spojiti na SSH-

posluživač *serveo.net* kako bi se stvorio zapis u datoteci *known_hosts* (*ssh serveo.net*) ili se pak može rabiti argument *-o StrictHostKeyChecking=no* koji isključuje krut pregled nahodi li se javni ključ u datoteci *known_hosts* kako se bi moglo uspješno pokrenuti SSH-prosljeđivanje vrata u podlozi. Pošto se je stvorila ova usluga potrebno je osvježiti *daemon* za usluge:

```
sudo systemctl daemon-reload
```

i pokrenuti uslugu:

```
sudo systemctl start autossh
```

Pošto se pregleda stanje usluge:

```
sudo systemctl status autossh
```

ter utvrdi da nema pogrješaka i da usluga radi ispravno, može se omogućiti pokretanje usluge kod pokretanja posluživača:

```
sudo systemctl enable autossh
```

5.2.2. Cloudflare Tunnel

Cloudflare Tunnel izvrsna je mogućnost ako se već rabe Cloudflareove usluge ili pak imade domena smještena na Cloudflareu. Cloudflareovi tuneli pružaju siguran način povezivanja ovih sredstava s Cloudflareovom mrežom bez objavljivanja ove lokalne IP-adrese čim je napadačima puno teže doznati ovu IP-adresu i izvršiti kakov bilo napadaj. Cloudflareovi tuneli djeluju tako da *daemon* stvara samo izlazne veze s svjetskom mrežom Cloudflarea i tako se može sigurno povezati koja bilo vrsta posluživača ili protokoli *web*-posluživača, SSH-posluživača, daljinskih radnih površina i brojni drugi. Tuneli su trajni predmeti koji usmjeruju promet na DNS-zapise Cloudflareova svjetskoga CDN-a koji promet preusmjeruje u najbliže Cloudflareovo podatkovno središte, a u koji se upućuje veza iz ovoga tunela [34].

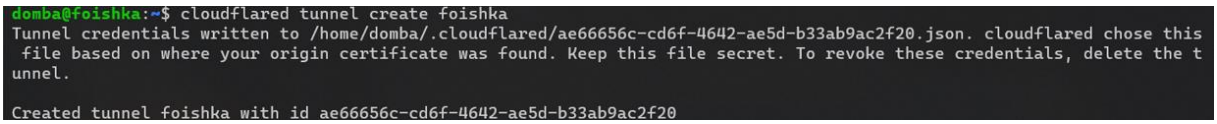
Prvo je potrebno instalirati *daemon cloudflared* koji omogućuje rad s Cloudflareovim tunelima i ostalim Cloudflareovim uslugama. Može se ili ručno skinuti izvršna datoteka ili dodati iz Cloudflareova repozitorija čim će se omogućiti lakša osvježavanja *daemona cloudflared* sama od sebe. Potrebno je izvršiti sljedeće korake: dodati Cloudflareov ključ za potpisivanje paketa, dodati Cloudflareov *apt*-repozitorij u ovaj *apt*-repozitorij ter na kraju osvježiti repozitorije i instalirati *cloudflared* (*sudo apt update* i *sudo apt install cloudflared*). Pošto se bude imao ispravan *daemon cloudflared*, treba se prijaviti ovim računom:

```
cloudflared tunnel login.
```

Izvršivanje ove naredbe otvorit će *web*-preglednik u kojem će se prijaviti u ovaj Cloudflareov račun i odabrati za koju se domenu želi ovaj *daemon* vezati ter će stvoriti i

certifikat računa i pohraniti ga u zadano kazalo *cloudflared* (*~/.cloudflared/cert.pem*). Po tom se može stvoriti tunel:

```
cloudflared tunnel create foishka
```



Slika 26. Stvaranje Cloudflareova tunela s pomoću *daemona cloudflared*

Potrebno je kopirati ID tunela za konfiguracijsku datoteku koja se stvara u kazalu *.cloudflared*:

```
nano config.yaml
```

```
tunnel: ae66656c-cd6f-4642-ae5d-b33ab9ac2f20
credentials-file: /home/domba/.cloudflared/ae66656c-cd6f-4642-ae5d-b33ab9ac2f20.json
ingress:
  - hostname: foishka.dominik-lackovic.from.hr
    service: ssh://localhost:22
  - service: http_status:404
```

Upozorava se na poddomenu na kojoj se želi da se prosljeđuje ova usluga dotično protokol *ssh* koji se izvršuje na lokalnom poslužiteljskom računalu na standardnim vratima *22* dok će svi ostali zahtjevi koji nisu zahtjevi za SSH-veze uzrokovati pogrješku s stanjem *404* za odabranu poddomenu.

Preostaje još dodati zapis CNAME DNS koji će voditi na upravo stvoreni tunel, a zapis će se dodati za domenu *foishka.dominik-lackovic.from.hr*. Tim se osigurava da će kad je god tunel pokrenut naslov *foishka.dominik-lackovic.from.hr* voditi na ovaj tunel jer je tunel povezan s varijablom *tunnelID*. *Daemon cloudflared* će stvaranje DNS-zapisa učiniti sam naredbom:

```
cloudflared tunnel route dns foishka foishka.dominik-lackovic.from.hr
```

Pokreće se Cloudflareov tunel:

```
cloudflared tunnel run foishka
```

i ako nema nikakovih pogrješaka znamenuje da je sve u redu i da se može stvoriti nova usluga kako bi tunel uvijek radio:

```
cloudflared service install
```

S obzirom na to da je za stvaranje usluge potrebna *sudo*-ovlast ova naredba ne će raditi jer se konfiguracija za Cloudflareov tunel trenutačno nahodi u kazalu korisnika *domba* pa je potrebno kao argument naredbi proslijediti i stazu konfiguracije tunela:

```
sudo cloudflared --config /home/domba/.cloudflared/config.yaml service
                                    install
```

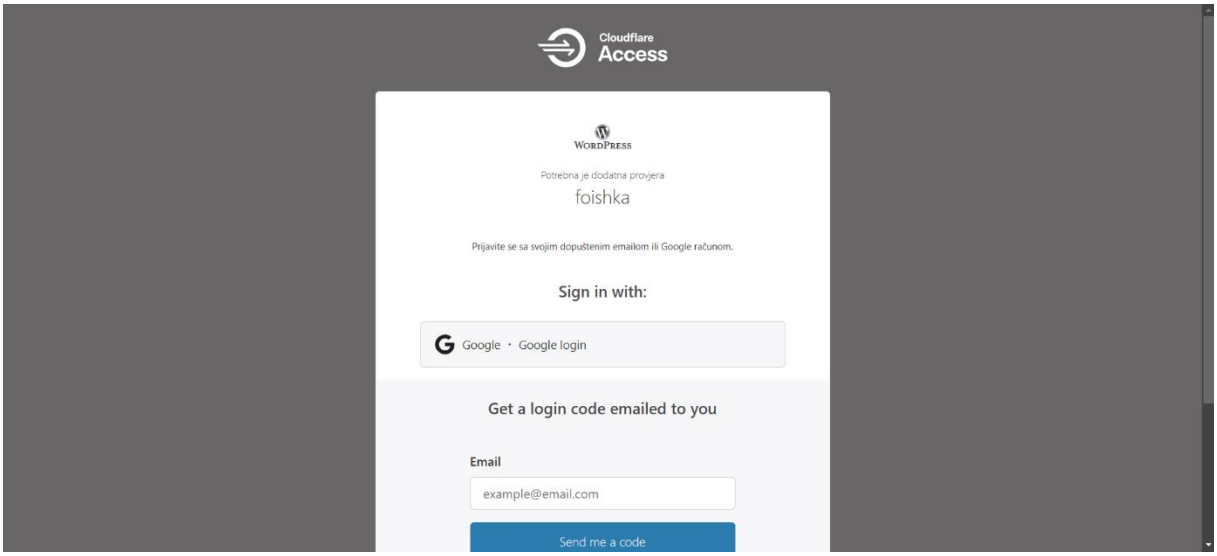
Da bi se na kraju konačno moglo spojiti na SSH-poslužitelj preko lako pamtljive domene *foishka.dominik-lackovic.from.hr*, potrebno je u konfiguraciju SSH-klijenta (*~/.ssh/config*) dodati još *proxy*-naredbu:

```
Host foishka.dominik-lackovic.from.hr
      ProxyCommand cloudflared access ssh --hostname %h
```

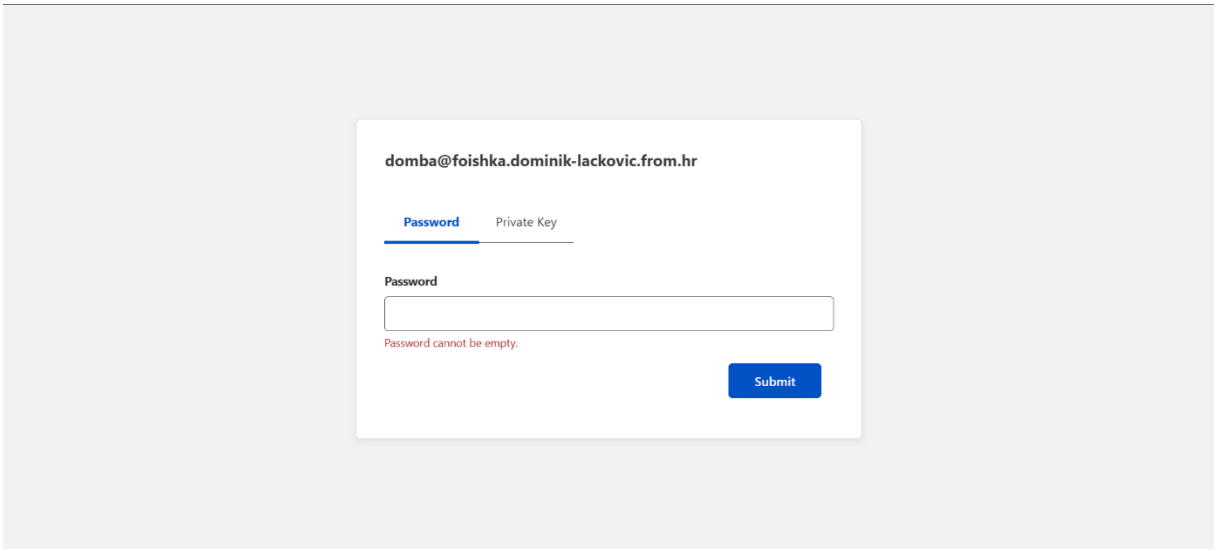
Sad kad se pokuša prijaviti na SSH-poslužitelj naredbom *ssh domba@foishka.dominik-lackovic.from.hr*, prvo će se izvršiti *proxy*-naredba koja će stvoriti SSH-vezu s Cloudflareovim poslužiteljima, a koji će za tim proslijediti vezu na ovaj poslužitelj i samim tim omogućiti SSH-vezu.

Cloudflareovi tuneli omogućuju i vezu za SSH i VNC-veze s prikazivanjem u pregledniku bez potrebe za klijentskim programom ili promjenama konfiguracije krajnjega korisnika. Da bi se to konfiguriralo, potrebno je u nadzornoj ploči Cloudflare Zero Trust stvoriti novu aplikaciju na poddomeni *foishka.dominik-lackovic.from.hr* i u naprjednim postavkama omogućiti „*Browser rendering*“ – SSH. Isto tako, korisna stvar može biti da se pristup SSH-sučelju preko preglednika može ograničiti samo na određene korisnike. Npr. samo na korisnike čija je *mail*-domena *foi.hr* ili *student.foi.hr*. Autentifikacija se korisnika obavlja preko prethodno postavljenih pružatelja identiteta, a neki od znamenitijih su Googleova prijava, Facebook, Azure Active Directory, Okta, SAML, OpenID Connect, GitHub i ručno ovjerovljivanje slanjem jednokratnoga koda na *mail* korisnika.

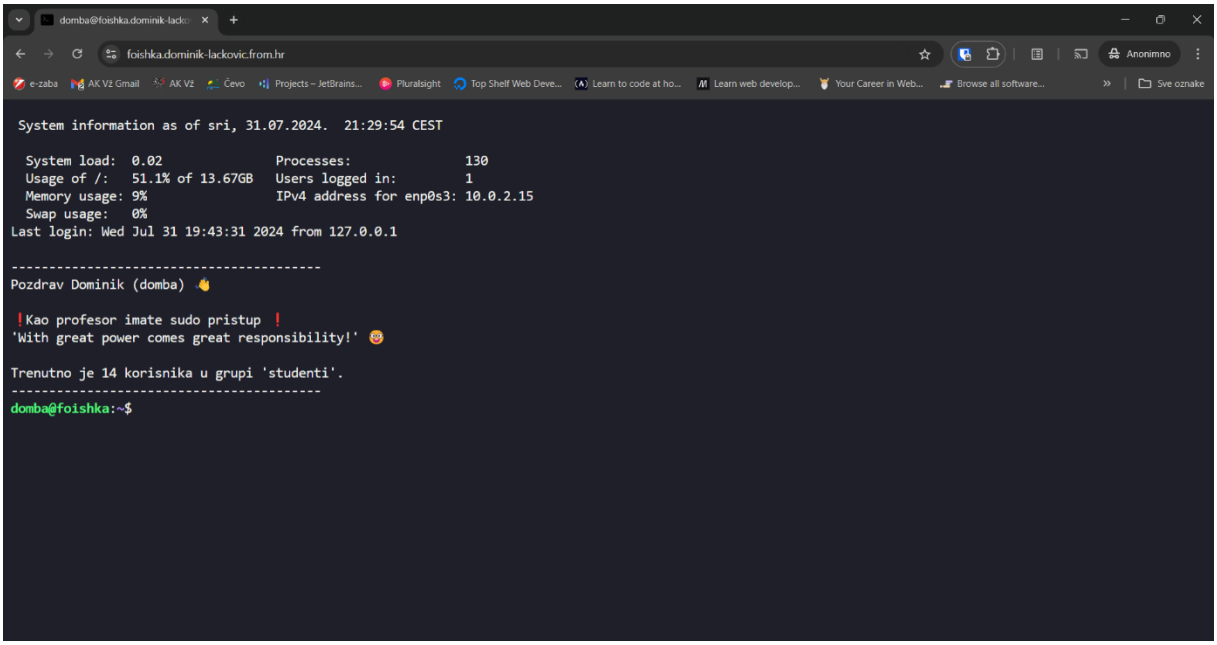
Svaka autentifikacija, uspješna ili neuspješna, bilježi se i administratori sustava imaju pregled nad svim prijavama pa ako uoče da je koji račun kompromitiran mogu poduzeti potrebne korake. Sad se kad se u *web*-pregledniku pohodi naslov foishka.dominik-lackovic.from.hr susreće zaslon za prijavu korisnika gdje se može prijaviti ili Googleovim računom ili se unaša *mail* na koji će stignuti jednokratni kod za potvrdu. Nakon što se prijavi kako bilo i ako je program Cloudflare Access konfiguriran da dopušta pristup na temelju odabrane metode ili identiteta, bit će omogućen dalji pristup ovomu SSH-poslužitelju.



Slika 27. Prijava Cloudflare Access u aplikaciju „*Foishka*“



Slika 28. Pristup Cloudflareovu tunelu i SSH-vezi – upisivanje zaporke za korisnika *domba*



Slika 29. SSH-veza preko *web*-preglednika dotično preko Cloudflareova tunela

Trenutačni nedostatak ovakove metode koja rabi prikaz SSH-veze u pregledniku jest da se korisnik mora 2 puta autentificirati. Prvi put kod prijave u samu uslugu Cloudflare Access i drugi put u SSH gdje upisuje željeno korisničko ime na posluživaču i zaporku ili se zalijepi privatni ključ ako se rabi autentifikacija javnim i privatnim ključevima.

Cloudflare ima i mogućnost kako elegantno skratiti ova 2 koraka autentifikacije u samo 1 s pomoću kratkotrajnih certifikata. Cloudflare Access i tako može zamijeniti običajne modele SSH-ključeva kratkotrajnim certifikatima koji se izdaju korisnicima na temelju *tokena* izrođena njihovom prijavom na Access. Tako se uklanja breme izrađanja ključa s krajnjega korisnika, a istodobno poboljšava sigurnost pristupa infrastrukturi kratkotrajnim certifikatima [35].

Prvo se u nadzornoj ploči Cloudflare Zero Trust izrodi kratkotrajan certifikat za aplikaciju koja se je prije stvorila: Access – Service Auth – SSH i prepiše dobiveni javni ključ. Na SSH-posluživaču u mapi se */etc/ssh* stvara nova datoteka *ca.pub* s onim što imade u prepisanom javnom ključu. Za tim se u konfiguraciji SSH-posluživača treba dodati izbor da se rabi autentifikacija s pomoću javnih ključeva:

```
sudo nano /etc/ssh/sshd_config
PubkeyAuthentication yes
TrustedUserCAKeys /etc/ssh/ca.pub
```

Za tim se ponovno pokreće SSH-usluga:

```
sudo service ssh restart
```

ter ako se sad pohodi foishka.dominik-lackovic.from.hr i prijavi se *mailom* koji se završava na *@foi.hr* ili *@student.hr* bit će omogućen pristup SSH-posluživaču i bit će se sam od sebe prijavljen kao korisnik s kojim se je prijavilo u Cloudflare Access. Npr. ako se je prijavilo *mailom* dlackovic20@student.foi.hr, bit će se prijavljen na SSH-posluživač kao korisnik *dlackovic20*. Naravno da taj korisnik već prije treba biti izrođen na posluživaču. Ako se prijavi kao korisnik koji ne zadovoljava konfiguriranu sigurnosnu policu dotično ako *mail* nije iz domene *foi.hr*, pristup SSH-posluživaču ne će biti omogućen.

Ako se na klijentskoj strani isto tako žele rabiti kratkotrajni certifikate u SSH-klijentu po izboru, potrebno je instalirati *daemon cloudflared* i na klijentskom računalu ter izvršiti naredbu:

```
cloudflared access ssh-config --hostname foishka.dominik-lackovic.from.hr --short-lived-cert
```

koja će izroditi konfiguraciju za SSH-klijent koju je potrebno prenijeti u *~/.ssh/config*.

U ovom se slučaju želi omogućiti da stari način prijave i dalje djeluje tako da se recimo korisnik čija se adresa završava *@foi.hr* ili *@student.foi.hr* može prijaviti i kao drugi korisnik. U tom se slučaju konfiguracija koju je izrodio *cloudflared* mora malo preinačiti:

```
Host foishka.dominik-lackovic.from.hr
      ProxyCommand cloudflared access ssh --hostname %h
Match host foishka-sso.dominik-lackovic.from.hr exec "cloudflared access ssh-gen --hostname foishka.dominik-lackovic.from.hr"
  ProxyCommand cloudflared access ssh --hostname foishka.dominik-lackovic.from.hr
  IdentityFile ~/.cloudflared/foishka.dominik-lackovic.from.hr-cf_key
  CertificateFile ~/.cloudflared/foishka.dominik-lackovic.from.hr-cf_key-cert.pub
```

Dakle, sad kad se pokuša *ssh domba@foishka.dominik-lackovic.from.hr*, otvorit će se prijava na Cloudflare Access ter se i dalje mora potvrditi identitet adresom *@foi.hr* i onda će se upitati za zaporku korisnika „*domba*“.

Ako će se rabiti *alias*-naredba *ssh dlackovic20@foishka-sso.dominik-lackovic.from.hr*, u pregledniku će se otvoriti prijava na Cloudflare Access i nakon uspješne autentifikacije samo adresom *dlackovic20@foi.hr* bit će se sam od sebe prijavljen i u SSH-posluživač kao korisnik *dlackovic20*. Ako bi se tko pokušao prijaviti kao drugi korisnik, a u Cloudflare Access se prijavi svojim *mailom*, SSH-posluživač to ne će dopustiti dotično bit će se upitan za zaporku toga korisnika jer je način autentifikacije preko javnih ključeva bio neuspješan.

6. Zaključak

Ovaj se je rad bavio temom sigurna daljinskoga pristupa računalnim sustavima porabom protokola SSH (Secure Shell) s posebnim usredotočivanjem na ostvarivanje lokalna posluživača. Kroz rad je istražena bitnost sigurnosti u današnja digitalna dobu i pružene su smjernice za siguran daljinski rad, a predstavljeno je i nekoliko kakvoćnih alternativnih rješidaba za ostvaraj SSH-pristupa. U radu se je osim pregleda SSH-protokola bavilo i tehničkim vidovima konfiguracije, povećavanjem sigurnosti SSH-posluživača, a svaki izazov kod ostvaraja koji se je pojavio jest potanko opisan kroz samu rješidbu izazova. Raščlanilo se je i usporedilo nekoliko znamenitih SSH-klijenata kao što su PuTTY, Termius, Tabby Terminal i MobaXterm. Proveo se je ostvaraj lokalna posluživača na prividnom stroju s operacijskim sustavom Ubuntu Server 22.04 LTS. Razmotrile su se i uvele dodatne sigurnosne mjere i konfiguracije koje osiguravaju daljinski pristup SSH-posluživaču od neželjenih napadača. Isto tako, predstavljeno je i nekoliko alternativnih rješidaba koje omogućuju još sigurniji pristup SSH-posluživaču nego već dobro znani i osvjedočeni klasični pristup jer ne odgovaraju svima jednako određeni pristupi rješavanju neke teškoće.

SSH je vrlo učinkovit i siguran način za daljinski pristup računalnim sustavima za pristup naredbenomu retku, ali i pristup slikovnim aplikacijama ter prosljeđivanju bilo kojih vrata. Ostvarivanje se lokalnoga posluživača može uspješno provesti na prividnom stroju uz pravilnu konfiguraciju mreže i sigurnosnih protokola. Postoje brojni SSH-klijenti koji omogućuju jednostavan i siguran pristup daljinskim sredstvima, a korisnikov je odabir uvijek po njegovim osobnim preferencijama i prednostima.

Neke su se od početnih pretpostavaka o sveopćosti određenih rješidaba pokazale neispravnima, budući da različni operacijski sustavi djeluju drugačije. Ali, glavnina mogućnosti paketa OpenSSH radi dosljedno na svim platformama.

Naposljetku, smatra se kako je ovaj rad uspješno obuhvatio i ostvario sve ciljeve postavljene u uvodu. Nada se da će ovaj rad poslužiti kao koristan vodič za sve koji se bave ili pak se kane baviti SSH-protokolom i daljinskim pristupom ter im pružiti vrijedne smjernice i upute za ostvaraj sigurnih daljinskih sustava ter da će se nadahnuti i uspjeti riješiti teškoće koje imaju s sigurnim daljinskim pristupom i ostvariti sve zahtjeve svojih korisnika.

Popis literature

[1] Brian Ward, The Book of VMware: The Complete Guide to VMware Workstation, No Starch Press, 2002.

[2] Brendan Burns, Designing Distributed Systems, O'Reilly Media, 2018.

[3] Hrvoje Horvat, Operativni sustavi i računalne mreže - Linux u primjeni. Hrvoje Horvat, 2023. doi: 10.5281/zenodo.10084684.

[4] Daniel J. Barrett i Richard E. Silverman, SSH, the Secure Shell: The Definitive Guide. O'Reily, 2001.

[5] Christopher NEGUS, *Linux Bible*, Indianapolis, 2020.

[6] Michael D. BAUER, *Building secure servers with LINUX*, Beijing etc., 2002.

[7] „OpenSSH Project History“, OpenSSH, https://www.openssh.com/history.html, pristupljeno 22.05.2024.

[8] „OpenSSH Manual Pages“, OpenSSH, https://www.openssh.com/manual.html, pristupljeno 22.05.2024.

[9] „What are SSH Host Keys?“, SSH Communications Security, https://www.ssh.com/academy/ssh/host-key, pristupljeno 27.05.2024.

[10] „Authorized Keys File in SSH“, SSH Communications Security, https://www.ssh.com/academy/ssh/authorized-keys-file, pristupljeno 27.05.2024.

[11] „How to Use ssh-keygen to Generate a New SSH Key“, SSH Communications Security, https://www.ssh.com/academy/ssh/keygen, pristupljeno 27.05.2024.

[12] „OpenSSH Server configuration for Windows Server and Windows“, Microsoft, https://learn.microsoft.com/en-us/windows-server/administration/openssh/openssh_server_configuration, pristupljeno 27.05.2024.

[13] Daniel Olaogun, „SSH Security Best Practices: Protecting Your Remote Access Infrastructure“, Tailscale, https://tailscale.com/learn/ssh-security-best-practices-protecting-your-remote-access-infrastructure/, pristupljeno 22.06.2023.

[14] Daniel Holt, „Use SSHFP Records to Verify SSH Host Keys“, Vultr, https://docs.vultr.com/use-sshfp-records-to-verify-ssh-host-keys, pristupljeno 30.06.2024.

[15] Larry PETERSON, *Computer networks. A systems approach*, Cambridge, 2022.

[16] Dmitry Babinsky, „Port Forwarding“, Termius, https://support.termius.com/hc/en-us/articles/4402386576793-Port-Forwarding, pristupljeno 30.06.2024.

[17] „Port Forwarding“, Ubuntu https://help.ubuntu.com/community/SSH/OpenSSH/PortForwarding, pristupljeno 30.06.2024.

[18] „System requirements“, Ubuntu, https://ubuntu.com/server/docs/system-requirements, pristupljeno 05.07.2024.

[19] „What Is Network Address Translation (NAT)?“, Cisco, https://www.cisco.com/c/en/us/products/routers/network-address-translation.html, pristupljeno 05.07.2024.

[20] „What is the Difference Between NAT and CGNAT?“, NFWare Inc, https://nfware.com/blog/what-is-the-difference-between-nat-and-cgnat, pristupljeno 05.07.2024.

[21] „What is Carrier-grade NAT (CGN/CGNAT)?“, A10 Networks Inc, https://www.a10networks.com/glossary/what-is-carrier-grade-nat-cgn-cgnat/, pristupljeno 05.07.2024.

[22] „How does virtual CGNAT (Carrier-Grade NAT) work?“, NFWare Inc, https://nfware.com/blog/how-does-virtual-cgnat-work, pristupljeno 05.07.2024.

[23] „What is dynamic DNS (DDNS)?“, Cloudflare, Inc, https://www.cloudflare.com/learning/dns/glossary/dynamic-dns/, pristupljeno 29.07.2024.

[24] „Remote Access with Dynamic DNS, Vitalwerks Internet Solutions“, LLC dba No-IP, https://www.noip.com/remote-access, pristupljeno 29.07.2024.

[25] „Registracija domena“, CARNET, https://domene.hr/portal/register/info-free-fromhr, pristupljeno 30.07.2024.

[26] „Connect, protect and build everywhere“, Cloudflare, Inc, https://www.cloudflare.com/, pristupljeno 30.07.2024.

[27] „PuTTY User Manual“, PuTTY, https://the.earth.li/~sgtatham/putty/0.81/htmldoc/, pristupljeno 30.07.2024.

[28] Dmitry Babinsky, „Host Chaining“, Termius, https://support.termius.com/hc/en-us/articles/4407561837209-Host-Chaining, pristupljeno 01.08.2024.

[29] „Termius Education | Github“, Termius, https://termius.com/education, pristupljeno 01.08.2024.

[30] „Tabby - a terminal for a more modern age“, Tabby, https://tabby.sh/, pristupljeno 01.08.2024.

[31] „MobaXterm free Xserver and tabbed SSH client for Windows“, Mobatek, https://mobaxterm.mobatek.net/, pristupljeno 01.08.2024.

[32] Trevor Dixon, „Serveo: expose local servers to the internet using SSH“, serveo.net https://serveo.net/, pristupljeno 01.08.2024.

[33] Carson Harding, „autossh - Automatically restart SSH sessions and tunnels“, https://www.harding.motd.ca/autossh/, pristupljeno 01.08.2024.

[34] „Cloudflare Tunnel“, Cloudflare, Inc, https://developers.cloudflare.com/cloudflare-one/connections/connect-networks/, pristupljeno 01.08.2024.

[35] „Configure short-lived certificates“, Cloudflare, Inc, https://developers.cloudflare.com/cloudflare-one/identity/users/short-lived-certificates, pristupljeno 01.08.2024.

Popis slika