CARNET

Osnovno sigurnosno ojačavanje *Linux* poslužitelja

CERT.hr-PUBDOC-2021-12-405

CERT.hr

Sadržaj

1 Uvod

Security hardening je pojam koji označava „ojačavanje sigurnosti“ više no što je sustav uobičajeno (u početnom stanju nakon instalacije softvera) siguran. Ojačavanje sigurnosti, kao što sam naziv kaže, dodatno je povećanje sigurnosti sustava korištenjem dodatnih alata, sigurnim konfiguriranjem postavki poslužitelja i instaliranog softvera te definiranjem i praćenjem sigurnosnih politika.

Iako se proizvođači poslužiteljskog softvera nastoje brinuti da njihov softver bude što je moguće sigurniji, uvijek postoje nepredviđene pogreške i zato je uvijek potrebno poduzeti dodatne korake za osiguravanje sustava – tako eventualne ranjivosti nekog softvera neće biti dovoljne da se kompromitira cijeli poslužitelj. Ovakav pristup ojačavanja sigurnosti naziva se „*defense in depth*“ i pruža, kao što mu i sam naziv kaže, dubinsku, tj. višeslojnu obranu.

Ojačavanje sigurnosti poslužitelja važan je, potreban i ključan korak prije priključivanja poslužitelja na mrežu i omogućavanja pristupanju korisnicima, pogotovo kad je riječ o produkcijskom poslužitelju.

Sigurnosno ojačan poslužitelj znatno će teže biti uspjeti napasti nego poslužitelj u čije ojačavanje nije uložen nikakav dodatni trud osim podrazumijevane (engl. *default*) konfiguracije.

Svaki uspješni napad na poslužitelj stvara velik problem za organizaciju koju opslužuje i njeno poslovanje, a posljedice napada mogu biti uznemirujuće, ozbiljne pa i teške. Napadač može:

- Podatke na računalu šifrirati ucjenjivačkim softverom (engl. *ransomware*) i za njihovo dešifriranje, tražiti otkupninu (engl. *ransom*);
- ukrasti povjerljive podatke i time naštetiti tvrtki ili znatno ugroziti njenu reputaciju;
- napadati posjetitelje aplikacije koja je pohranjena na poslužitelju;
- spriječiti pristup poslužitelju (DoS ili DDoS napadima);
- preuzeti kontrolu nad poslužiteljem.

Sve te posljedice iziskuju mnogo novca, truda i stresa kako bi ih se ispravilo, i zato je bolje spriječiti nego liječiti – sigurnosno ojačavanje trebalo bi biti uobičajena praksa svakog sistemskog administratora ili programera koji postavlja i održava poslužitelj.

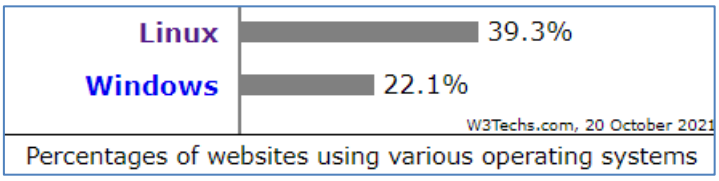
U ovom dokumentu navest ćemo i prokomentirati neke od najčešćih pogrešaka u konfiguraciji Linux poslužitelja na koje ukazuju penetracijska testiranja i sigurnosne provjere ranjivosti. Također, prokomentirat ćemo na koje je načine moguće zloupotrijebiti sustav ako se ne primjenjuje sigurnosno ojačavanje i kako treba izgledati ispravna konfiguracija.

U daljnjem kontekstu dokumenta podijelit ćemo Linux distribucije na dvije skupine koje se razlikuju po načinu upravljanja programskim paketima i sukladno tome se neka pravila i alati moraju drukčije konfigurirati:

- *DEB-based* skupina
  Pod *DEB-based* distribucije spadaju npr. Debian, Ubuntu, Linux Mint…

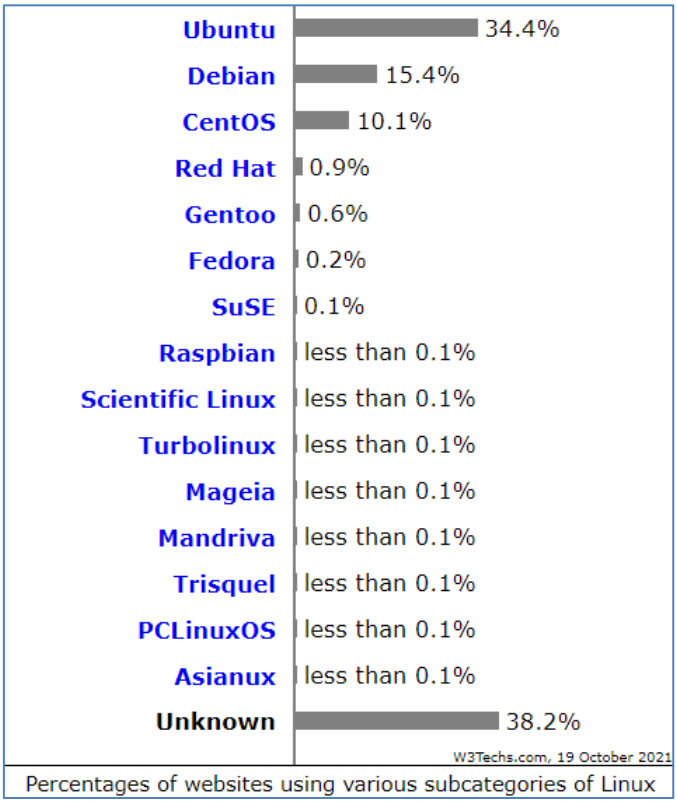
- *RPM-based* skupina
  Pod *RPM-based* spadaju npr. CentOS, Red Hat Enterprise, openSUSE, Fedora…

Statistika koju provodi *W3Techs* pokazala je da se u 2021. godini na *web* poslužiteljskim računalima više koristio Linux nego Windows.



Slika 1 ***Na poslužiteljskim računalima prevladava Linux*** **[1]**

Također, prema prikupljenim podacima, najkorištenija distribucija operacijskog sustava Linux na *web* poslužiteljima je *Ubuntu* kojoj ćemo upravo iz tog razloga i pridati najviše pažnje u nastavku dokumenta:



Slika 2 ***Najkorištenija distribucija Linuxa na web poslužiteljskim računalima je Ubuntu*** **[2]**

Najznačajniji rizici na Linux poslužiteljima postoje zbog:

- neažuriranog i nepotrebnog softvera,
- nesigurnog upravljanja dozvolama i korisničkim računima,
- nesigurnih lozinki korisnika s visokim privilegijama,
- nepotrebno otvorenih priključaka (engl. *ports*) i pokrenutih usluga (engl. *services*),
- nesigurne konfiguracije operacijskog sustava i poslužiteljskog softvera,
- nesigurno postavljenog vatrozida (engl. *firewall*) i
- pohranjivanja osjetljivih podataka na poslužitelju.

U nastavku dokumenta prokomentirat ćemo svaku od navedenih loših praksi i iznijeti preporuku kako ju ispraviti.

2 Sigurnosno ojačavanje Linux poslužitelja

Smjernica za sigurnosno ojačavanje ima puno, ali u nastavku ovog dokumenta fokusirat ćemo se na najčešće pogreške uočene tijekom penetracijskog testiranja i sigurnosnih provjera aplikacija. Navedene pogreške je najčešće jednostavno ispraviti, a značajno će doprinijeti sigurnosti poslužitelja.

2.1 Ispravna instalacija i ažuriranje softvera na poslužitelju

Za sigurnost sustava iznimno je bitno ispravno odabrati, instalirati i zatim redovito ažurirati sav softver koji je instaliran na poslužiteljsko računalo.

2.1.1 Što instalirati?

Na poslužiteljskom računalu treba biti instaliran minimum softvera koji je potreban za njegov rad. Nakon što se instalira operacijski sustav, instaliraju se programi i usluge neizostavne za rad poput poslužiteljskog softvera, baze podataka, itd.

Na Linux poslužitelje iz sigurnosnih, ali i funkcionalnih razloga (manja potrošnja resursa), ne instaliramo nepotrebni softver, što uključuje i grafičko sučelje (engl. *Desktop Environment*), osim ako nam je uistinu potrebno za funkcionalnost aplikacije koju poslužitelj izvodi. Sukladno tome, na poslužitelj se ne instalira ni softver poput npr. *LibreOfficea*, klijenta za čitanje e-pošte, i sl. jer poslužiteljsko računalo nije namijenjeno radnjama za koje se inače koristi osobno računalo. Iznimka su *web* preglednici koji se mogu instalirati i koristiti isključivo za otklanjanje pogrešaka (engl. *debug*), a nikako za npr. čitanje novinskih portala, posjećivanje društvenih mreža, itd. Također, treba paziti da se instalirani *web* preglednik redovito ažurira jer korištenje neažuriranog *web* preglednika predstavlja veliki sigurnosni rizik, o čemu se više pisalo u dokumentima Nacionalnog CERT-a Exploit kitovi i Sigurnosni rizici JavaScript kôda prilikom pregledavanja weba.

Radnje kao što su pretraživanje *weba*, čitanje e-pošte i otvaranje PDF datoteka na poslužitelju predstavljaju značajan i nepotreban sigurnosni rizik, a instalacija usluga i programa koji se ne koriste samo je potencijalni dodatni sigurnosni rizik kojeg je bolje eliminirati već u početku.

2.1.2 Otkud instalirati

Prilikom instalacije softvera preporučljivo je koristiti službeni središnji repozitorij korištene Linux distribucije. Softver za upravljanje paketima (engl. *Packet Manager*) poput YUM-a, *apt*/*apt-get* i sl. spaja se na službeni repozitorij i s njega preuzima softver, njegova ažuriranja i sigurnosne zakrpe. Drugim riječima, ako u naredbenu konzolu upišemo npr.:

```
$ sudo apt-get install apache2
```

instalirat ćemo poslužiteljski softver *Apache* upravo iz službenog repozitorija korištene Linux distribucije (u ovom slučaju *Debian* 11) jer se softver za upravljanje paketima *apt-get* spojio na njega i preuzeo navedeni softver.

Instalacija iz službenog repozitorija preporučljiva je iz više razloga:

- za softver koji se nalazi u središnjem repozitoriju smo najsigurniji da je riječ o ispravnom legitimnom softveru;

- središnji se repozitorij redovito održava i pazi se da ne dođe do nekompatibilnosti različitog softvera namijenjenog istoj distribuciji i

- softver za upravljanje paketima može lako pronaći i primijeniti ažuriranja.

No čak i kod instalacije softvera iz službenog repozitorija, treba se upoznati s politikom podrške softverskih paketa koju pruža određena Linux distribucija. Ni jedna Linux distribucija (ili drugi operacijski sustav) se ne obvezuje da će u nedogled pružati podršku za određenu inačicu operacijskog sustava, tako da softverski paketi, i čak cijeli operacijski sustav, imaju navedene datume (engl. *end-of-life*) kada više neće dobivati nikakav oblik podrške, uključujući sigurnosne zakrpe (engl. *security patches*). Uobičajeno je da određena inačica (engl. *version*) Linux distribucije (npr. Debian 11, Ubuntu 20.04, Red Hat Enterprise Linux 8 ...) ima podršku nekoliko godina, nakon čega korisnici moraju ažurirati cijeli operacijski sustav na novi inačicu ako žele nastaviti primati podršku.

Također, postoje i razlike u politici podrške na razini individualnih paketa. Primjerice, iako tvrtka *Canonical* (proizvođač distribucije Ubuntu) načelno pruža pet godina službene podrške za Ubuntu inačicu 20.04, to ne znači da svaki individualni softverski paket dobiva jednaku razinu podrške. Primjerice, *Canonical* će za neke pakete iz Ubuntu repozitorija pružiti službenu podršku, uključujući sigurnosne zakrpe i ispravke grešaka (engl. *bug fix*) svih pet godina podrške operacijskog sustava, dok neki drugi paketi iz Ubuntu repozitorija uopće neće imati nikakvu garanciju službene podrške.

Drugim riječima, moguće je naredbom `apt` iz službenih repozitorija distribucije Ubuntu instalirati softverske pakete koji neće primati sigurnosne zakrpe ili će dostupnost sigurnosnih zakrpi ovisiti o zajednici *Ubuntu* korisnika/volontera. Korištenje naredbe `apt` na distribuciji Ubuntu zapravo obuhvaća četiri repozitorija: *main*, *universe*, *restricted*, *multiverse*. Uzmimo za primjer repozitorije *main* i *universe*:

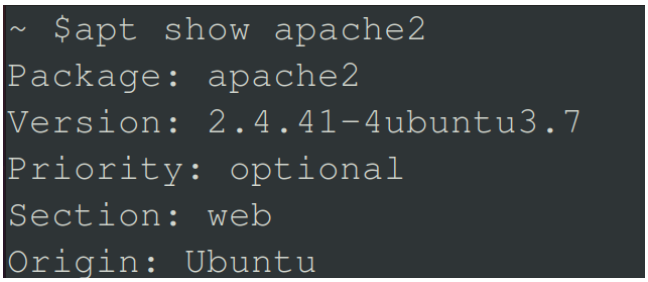
- main – temeljni paketi operacijskog sustava Ubuntu koji primaju punu podršku tvrtke *Canonical*, uključujući sigurnosne zakrpe i ispravke grešaka. Primjerice, Ubuntu 20.04 prima podršku do travnja 2025. godine i to uključuje sve pakete iz repozitorija *main*.

- universe – sastoji se od paketa koji nisu ključni za korištenje Ubuntua, ali su popularni i moguće je da će ih korisnik htjeti instalirati, npr. *Wireshark*. Iako nam je taj softver dostupan za preuzimanje, ne možemo očekivati da će se u repozitorij dodavati sigurnosne zakrpe i ažurirane inačice, jer ta podrška ovisi o zajednici korisnika/volontera, a ne o tvrtki *Canonical*. Drugim riječima, čak i da naredbom `apt update` ne otkrijemo nikakva dostupna ažuriranja, to ne znači da je instalirani softver siguran i da ima sve sigurnosne zakrpe, već samo da nema dostupnih ažuriranja u službenom repozitoriju (jer njihova podrška ovisi o zajednici volontera). Softver iz ovog repozitorija koristimo na vlastitu odgovornost.

Uzmimo za primjer poslužiteljski softver *Apache* i alat za uređivanje slika *Gimp*. Sljedećom naredbom možemo provjeriti kojem repozitoriju pripada softver *Apache*.

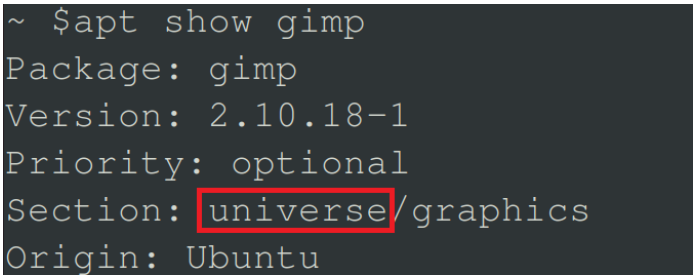
```
$ apt show apache2
```

Kao što vidimo na slici **Error! Reference source not found.** pod „*Section*“, *Apache* pripada repozitoriju *main* (jer nije naveden zasebni naziv repozitorija) i pripada kategoriji *web*.



Slika 3 ***Poslužiteljski softver Apache nalazi se u main repozitoriju***

Ako pokrenemo istu naredbu za uređivač slika *Gimp*, vidimo da on pripada repozitoriju *universe* (i kategoriji *graphics*):



Slika 4 ***Uređivač slika Gimp nalazi se u universe repozitoriju***

Kako bismo provjerili ažuriranja *Gimpa* koja su dodana u repozitorij koristimo sljedeću naredbu:

```
$ apt-get changelog gimp
```

Nakon upisivanja naredbe, iako se radi o *Ubuntu* inačici 20.04 koja je trenutno i dalje službeno podržana, vidimo [Slika 5] da je zadnje dodano ažuriranje iz ožujka 2020. godine, tj. ovaj softverski paket nije dobio nikakvo ažuriranje otprilike godinu i pol.

```
gimp (2.10.18-1) unstable; urgency=medium

  * New upstream release (Closes: #954143)
  * Bump minimum libbabl-dev to 0.1.74 & libgegl-dev to 0.4.22
  * Bump minimum libmypaint-dev to 1.14
  * Disable automatic update checking since the distro handles that
  * debian/libgimp2.0.symbols: Add new symbols
  * Drop the 3 backported patches applied in the new release

 -- Jeremy Bicha <jbicha@debian.org>  Sun, 29 Mar 2020 01:27:45 -0400

gimp (2.10.14-3) unstable; urgency=medium

  * Team upload
  * d/p/Don-t-crash-when-a-plugin-defines-an-invalid-property-nam.patch:
    Apply patch from upstream 2.10.16 release. If a plugin defines an
    invalid property name, don't use that plugin instead of crashing.
    (Closes: #953880, #953854, #953794, LP: #1857254)
  * d/p/Fix-invalid-property-name-in-pagecurl-plugin.patch:
    Apply patch from upstream 2.10.16 release to fix an invalid property
    name in the pagecurl plugin. GLib 2.64 validates property names
    in line with rules that were previously only in the documentation.

 -- Simon McVittie <smcv@debian.org>  Tue, 17 Mar 2020 12:01:32 +0000
```

Slika 5 ***Posljednja ažurirana inačica Gimpa dodana je u universe repozitorij 2020. godine***

Dakle, ako želimo biti sigurni da imamo podršku za neki softver na distribuciji *Ubuntu*, obavezno ga treba instalirati iz repozitorija *main*. Ako nam treba neki softver koji nije dostupan u repozitoriju *main*, morat ćemo riskirati s instalacijom iz drugog nepodržanog repozitorija, ili se, preporučeno, snaći na drugi način (npr. koristiti službeni repozitorij proizvođača softvera, *docker image* s potrebnim softverom, *snap/flatpak* paket, itd.).

Kako bismo saznali stanje podrške softverskih paketa na distribuciji *Ubuntu*, možemo koristiti naredbu:

```
$ ubuntu-security-status
```

```
~ $ubuntu-security-status
1661 packages installed, of which:
1619 receive package updates with LTS until 4/2025
  42 could receive security updates with ESM Apps until 4/2030

Enable Extended Security Maintenance (ESM Apps) to get 0 security
updates (so far) and enable coverage of 42 packages.
```

Slika 6 ***Na sustavu je instalirano 1619 paketa koji primaju podršku do travnja 2025.***

Naredba `ubuntu-security-status` ima dodatne opcije:

- `--thirdparty` – vraća informacije o paketima koji nisu instalirani iz središnjeg repozitorija, već od treće strane (engl. *third party*)
- `--unavailable` – vraća informacije o paketima koji više nisu dostupni

Sljedećom naredbom možemo vidjeti prethodno navedene dodatne opcije:

```
$ ubuntu-security-status --help
```

```
~ $ubuntu-security-status --help
usage: ubuntu-security-status [-h] [--thirdparty] [--unavailable]

Return information about security support for packages

optional arguments:
  -h, --help     show this help message and exit
  --thirdparty
  --unavailable
~ $
```

Slika 7 ***Dodatne opcije naredbe*** **`ubuntu-security-status`**

Zaključno, moramo biti svjesni otkud instaliramo programe i kako ćemo ih u budućnosti moći ažurirati.

Ponekad se javlja potreba za instalacijom softvera koji se ne nalazi u središnjem repozitoriju. U tom slučaju softver se mora preuzeti s neke druge lokacije. Tada se treba držati sljedećih smjernica:

- Iza preuzetog softvera trebala bi stajati tvrtka ili zajednica koja brine o sigurnosti i održavanju svog proizvoda. Rizično je preuzeti i koristiti program kojeg smo pronašli na *GitHub* repozitoriju pojedinca koji je razvio program kojeg ne održava ili ga ne održava redovito, već kako mu slobodno vrijeme dopusti.

- Ako se ipak koristi softver ili program od pojedinca koji je napustio njegovo održavanje ili ga ne održava redovito, potrebno je preuzeti i obvezu održavanja programa i ispravljanja pogrešaka koje predstavljaju ranjivost za poslužitelj na koji je softver instaliran.

2.1.3 Zašto redovito ažurirati?

Proizvođači softvera za poslužiteljska računala nastoje ga detaljno testirati i pažljivo programirati kako bi bio što sigurniji. No, unatoč tome, nijedan softver nije savršen, već sadrži i neke pogreške u kôdu ili konfiguraciji kojih proizvođač postane svjestan tek nakon što objavi softver, a korisnici su ga već preuzeli, instalirali i koriste ga.

Takve pogreške proizvođač može:

- otkriti sam analizom postojećeg kôda,

- može mu ih prijaviti korisnik (npr. sistem administrator koji je zaposlenik tvrtke koja koristi proizvođačev softver na svojim poslužiteljima i koji je uočio potencijalnu ranjivost),
- mogu mu ih prijaviti sigurnosni stručnjaci koji se bave traženjem ranjivosti,
- u najgorem slučaju, može ih postati svjestan nakon što napadač preko pogrešaka u kôdu ili konfiguraciji softvera neovlašteno preuzme kontrolu nad poslužiteljskim računalom, uzrokuje nedostupnost poslužitelja uskraćivanjem usluge (engl. *Denial of Service*, DoS), pristupi osjetljivim podacima, šifrira poslužiteljsko računalo itd.

Pogreške koje se mogu zloupotrijebiti za napad na računalo predstavljaju ranjivosti i ako ih napadač otkrije prije no što ih proizvođač softvera ispravi ili korisnici ne instaliraju sigurnosnu zakrpu, postoji mogućnost da preko njih napadne poslužitelj.

Velik broj ranjivosti je javno poznat i, ako već ne postoji zakrpa, proizvođači rade na njihovom ispravljanju. Javno poznate ranjivosti često su dokumentirane i informacije o njima mogu se pronaći na CVE stranici. Tamo svaka ranjivost ima svoj identifikator, opis, posljedicu te naziv i inačice softvera na kojima je ranjivost prisutna.

Sigurnosni stručnjaci i/ili hakeri pokušavaju razviti programski kôd koji će uspješno iskoristiti ranjivost, zbog demonstracije (engl. *Proof of Concept*) ili s ciljem zlouporabe. Takav programski kôd naziva se *exploit* i jednom kad se razvije, često se objavi u *GitHub* repozitoriju, na *ExploitDB* stranici itd. odakle ga svatko zainteresiran može preuzeti.

U kontekstu iskorištavanja ranjivosti klijentskog softvera (*web* preglednika, PDF čitača i sl.), više javno dostupnih *exploita* može se skupiti u kolekciju koja se naziva *Exploit kit*. *Exploit kit* ima popis raznih softvera i njihovih inačica, ranjivosti koje postoje na tim inačicama i spreman *exploit* za iskorištavanje ranjivosti ako pronađu ranjivu inačicu softvera na računalu korisnika. *Exploit kitovi* detaljno su opisani u dokumentu Nacionalnog CERT-a Exploit kitovi.

Postoje slični gotovi alati koji automatizirano obavljaju cijeli proces skeniranja (računalni crvi, *vulnerability scanneri*...) i napada na računalo i zato za pronalazak i iskorištavanje ranjivosti nije potrebno nikakvo stručno znanje.

Čim se neko poslužiteljsko računalo spoji na mrežu i postane dostupno preko interneta, vrlo brzo će početi skeniranje i pokušaji spajanja na njega. Razni *botovi* neprestano skeniraju cijelu mrežu i pokušavaju pronaći neki javni poslužitelj koji nije dobro sigurnosno konfiguriran i kojeg mogu napasti. Ako tijekom skeniranja otkriju da poslužitelj koristi neažuran softver, alat može iskoristiti spreman *exploit* koji se odnosi na određenu ranjivost softvera i napasti ga. Čak i ako trenutno ne postoji razvijen *exploit* za određenu javno dostupnu ranjivost, sama činjenica da se javno zna o kojoj je ranjivosti riječ dovoljan je sigurnosni rizik jer je pitanje vremena kad će biti razvijen *exploit* za nju.

Često se sustav uspješno napadne upravo iskorištavajući ranjivost neažurnog softvera za koji je postojala sigurnosna zakrpa, ali nije instalirana na vrijeme. Jedan takav scenarij dogodio se 2019. godine [3]:

- 2019. godine je pronađena ranjivost *Apache* HTTPD poslužiteljskog softvera koja se mogla iskoristiti za eskalaciju privilegija s nisko privilegiranog korisnika na administratorskog (*root*) korisnika. Ta je ranjivost zabilježena pod oznakom CVE-2019-0211.

- Detaljan opis navedene ranjivosti može se pronaći na poveznici, ali ukratko je riječ o pogrešci u kôdu koja omogućava eskalaciju privilegija na HTTP poslužitelju instaliranom na *Unix* sustavima. Korisnici s niskim privilegijama napadom na navedenu ranjivost stjecali su pravo pisanja i izvršavanja s administratorskim (*root)* ovlastima.

- Programeri koji razvijaju i održavaju *Apache* brzo su reagirali, ispravili pogrešku i izdali novu inačicu softvera sa sigurnosnom zakrpom – *Apache* HTTPD 2.4.39.

- Administratori koji su redovito ažurirali poslužiteljski softver *Apache* brzo su instalirali ispravljenu inačicu i tako spriječili eventualne napade na svoje sustave.

- Ako administratori nisu instalirali inačicu sa sigurnosnom zakrpom, njihovi su sustavi i dalje bili izloženi napadima (i neki od njih su uspješno napadnuti) sve do trenutka dok nisu ažurirali *Apache*.

Kako bi se minimizirao rizik od napada i spriječili ovakvi scenariji, neizostavno je redovito pratiti objavljivanje ažuriranja i sigurnosnih zakrpa (engl. *security patches*) i čim prije ih instalirati. Kako bi pravovremeno bio informiran o svim dostupnim ažuriranjima i sigurnosnim zakrpama, korisnik se može pretplatiti na takozvanu *mailing* listu distribucije koju koristi i primiti poruku e-poštom svaki put kad se objavi ažuriranje ili sigurnosna zakrpa za navedenu Linux distribuciju. Neki primjeri *mailing* lista na koje se korisnici mogu predbilježiti (engl. *subscribe*) i biti u toku s novim objavama i sigurnosnim ažuriranjima za različite distribucije su:

- za distribuciju Debian postoji *Debian Security Announcements*

- za distribuciju Ubuntu postoji *Ubuntu Security Announcements*

- za distribuciju RedHat postoji *Red Hat Security announcements*

2.1.4 Kako redovito ažurirati?

Sav softver koji je instaliran iz središnjeg repozitorija moguće je ažurirati softverom za upravljanje paketima (npr. YUM, *apt*/*apt-get*…), ovisno o distribuciji koju koristimo. Za *DEB-based* distribucije možemo koristiti sljedeće naredbe:

1) `$ sudo apt update` – ažurira listu dostupnih ažuriranja za instalirani softver

2) `$ sudo apt upgrade` – instalira dostupna ažuriranja za instalirani softver

Ako je softver preuzet od treće strane, treba kod te treće strane redovito provjeravati jesu li objavljene sigurnosne zakrpe i primjenjivati ih.

Preporučeno je izbjeći da čovjek ručno provjerava stranice s kojih je preuzeo softver, već prebaciti odgovornost na softver za upravljanje paketima. Kako bi se to omogućilo,

repozitorij iz kojeg smo preuzeli softver može se dodati kao još jedan izvor (kao glavni izvor podrazumijeva se izvor središnji repozitorij) za preuzimanje i ažuriranje softverskih paketa. Npr. repozitorij u kojem se nalazi softver *Docker* na *DEB-based* distribucijama možemo dodati kao dodatni izvor za preuzimanje i ažuriranje sljedećom naredbom:

```
$ sudo add-apt-repository \
  "deb [arch=amd64] https://download.docker.com/linux/debian \
  $(lsb_release -cs) \
  stable"
```

Prilikom idućeg unosa naredbe:

```
$ sudo apt update
```

pretražit će se dostupna ažuriranja i za softver *Docker*, iako se on ne nalazi u službenom repozitoriju distribucije, već u repozitoriju tvrtke *Docker*.

Prethodno navedeni načini spadaju u ručno ažuriranje softvera. No, kao i izdavanje sigurnosnih zakrpa, i ažuriranje softvera je proces koji se neprestano ponavlja, a čovjek može zaboraviti provjeriti postoje li dostupne sigurnosne zakrpe i pravovremeno ih primijeniti. Zato je poželjno konfigurirati sustav tako da obavijesti korisnika o dostupnim sigurnosnim zakrpama ili čak automatski obavlja ažuriranje.

Postoji više načina za automatsko ažuriranje softverskih paketa ovisno o vrsti Linux distribucije, a u nastavku će se demonstrirati konfiguracija alata *Unattended Upgrades* namijenjenog *DEB-based* distribucijama.

Neki od paketa koje možemo koristiti za *RPM-based* distribucije su *yum-cron* ili *dnf-automatic* paket koji se konfiguriraju na sličan način. Svakako, trebali bismo, za distribuciju koju koristimo, provjeriti koji se softver za automatsko ažuriranje može koristiti.

Kao što i sam naziv „ažuriranje bez nadzora“ kaže, *Unattended Upgrades* omogućava automatsko preuzimanje i instalaciju ažuriranja i sigurnosnih zakrpa softvera instaliranog na (poslužiteljsko) računalo bez intervencije administratora računala.

Unattended Upgrades instalira se naredbom:

```
$ sudo apt install unattended-upgrades
```

Prilikom korištenja sustava *Unattended Upgrades*, korisno je nadzirati promjene u središnjem repozitoriju u odnosu na inačice softvera koje su već instalirane na računalo.

Iako se neke promjene mogu iščitavati iz datoteka `/var/log/dpkg.log` i dnevničkih zapisa (eng. *logs*) u mapi `/var/log/unattended-upgrades/`, jednostavan i automatiziran način za pratiti detalje promjena je pomoću alata *apt-listchanges* kojeg se preporuča instalirati zajedno s *Unattended Upgradesom*:

```
$ sudo apt install apt-listchanges
```

Apt-listchanges uspoređuje i korisniku prikazuje promjene koje su se dogodile u novoj inačici paketa u odnosu na instalirane inačice paketa. *Apt-listchanges* može se konfigurirati i na način da korisniku pošalje poruku e-pošte svaki put kad naiđe na promjene, tj. novije inačice softvera. Više o konfiguraciji *apt-listchanges* moguće je pronaći na poveznici.

Kako bismo mogli konfigurirati *Unattended Upgrades*, prvo moramo provjeriti je li omogućeno automatsko ažuriranje naredbom:

```
$ cat /etc/apt/apt.conf.d/20auto-upgrades
```

```
~ $cat /etc/apt/apt.conf.d/20auto-upgrades
APT::Periodic::Update-Package-Lists "1";
APT::Periodic::Unattended-Upgrade "1";
```

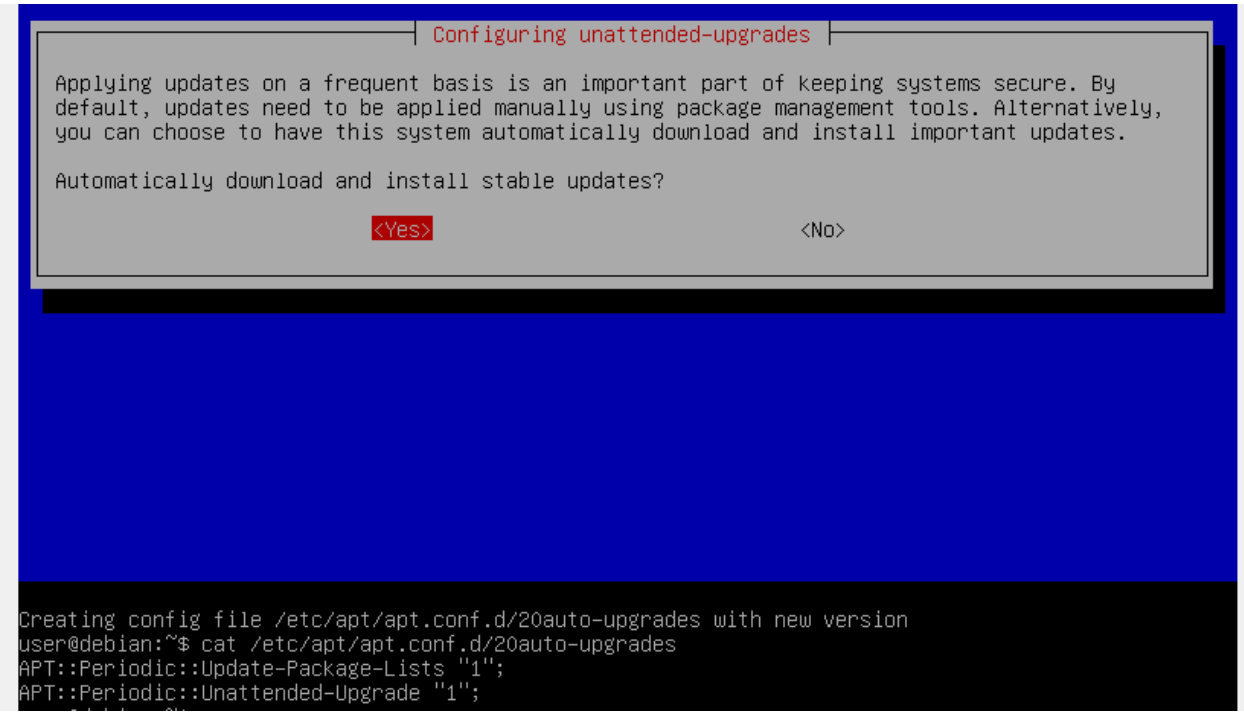
Slika 8 ***Sadržaj datoteke /etc/apt/apt.conf.d/20auto-upgrades***

Ako je pored polja `APT::Periodic::Update-Package-Lists` i `APT::Periodic::Unattended-Upgrade` vrijednost „1“, to znači da je omogućeno automatsko preuzimanje i ažuriranje softverskih paketa.

Ako datoteka `/etc/apt/apt.conf.d/20auto-upgrades` ne postoji, može se stvoriti ponovnom konfiguracijom alata *Unattended Upgrades* naredbom:

```
$ sudo dpkg-reconfigure --priority=low unattended-upgrades
```

nakon koje će nam se na ekranu pokazati sadržaj sa slike 9.



Slika 9 ***Stvaranje i sadržaj datoteke 20auto-upgrades***

`/etc/apt/apt.conf.d/50unattended-upgrades` je konfiguracijska datoteka *Unattended Upgradesa*. Ako pogledamo sadržaj te datoteke, vidimo da se sastoji od:

a. Izvora s kojih je dozvoljeno automatski ažurirati.
   Ako se brinemo da se nešto na sustavu ne pokvari, možemo odabrati i da se automatski instaliraju i ažuriraju samo sigurnosne zakrpe.

```
// Automatically upgrade packages from these (origin:archive) pairs
//
// Note that in Ubuntu security updates may pull in new dependencies
// from non-security sources (e.g. chromium). By allowing the release
// pocket these get automatically pulled in.
Unattended-Upgrade::Allowed-Origins {
        "${distro_id}:${distro_codename}";
        "${distro_id}:${distro_codename}-security";
        // Extended Security Maintenance; doesn't necessarily exist for
        // every release and this system may not have it installed, but if
        // available, the policy for updates is such that unattended-upgrades
        // should also install from here by default.
        "${distro_id}ESMApps:${distro_codename}-apps-security";
        "${distro_id}ESM:${distro_codename}-infra-security";
//      "${distro_id}:${distro_codename}-updates";
//      "${distro_id}:${distro_codename}-proposed";
//      "${distro_id}:${distro_codename}-backports";
};
```

Slika 10 ***Izvori s kojih je dozvoljeno automatski preuzimati ažuriranja i sigurnosne zakrpe***

b. Paketi koje je zabranjeno automatski ažurirati.
   U ovom slučaju je dopušteno automatski ažurirati sve pakete jer su primjeri zakomentirani oznakom „//“. No, u komentarima je objašnjeno kako treba navesti paket kojeg ne želimo automatski ažurirati koristeći *Python* regularne izraze: ako naziv ili dio naziva paketa stavimo između navodnih znakova („“), to će se odnositi na sve pakete čiji naziv počinje s navedenim izrazom. No, ako nakon naziva ili dijela naziva paketa stavimo oznaku „$“, to će se odnositi isključivo na paket s točno navedenim nazivom.

```
// Python regular expressions, matching packages to exclude from upgrading
Unattended-Upgrade::Package-Blacklist {
    // The following matches all packages starting with linux-
//  "linux-";

    // Use $ to explicitely define the end of a package name. Without
    // the $, "libc6" would match all of them.
//  "libc6$";
//  "libc6-dev$";
//  "libc6-i686$";

    // Special characters need escaping
//  "libstdc\+\+6$";

    // The following matches packages like xen-system-amd64, xen-utils-4.1,
    // xenstore-utils and libxenstore3.0
//  "(lib)?xen(store)?";
```

Slika 11 ***Paketi koji su na crnoj listi, tj. ne želimo ih automatski ažurirati***

c. Dodatne opcije
Dodatnim opcijama automatsko ažuriranje *Unattended Upgradesom* može se konfigurirati na više načina:

- ažuriranja ili sigurnosne zakrpe se niti preuzimaju niti instaliraju, ali se korisniku šalje poruka e-pošte o dostupnim ažuriranjima i sigurnosnim zakrpama;
- nakon ažuriranja operacijski se sustav ponovno pokreće, odmah ili u određeno vrijeme (npr. u 2:00 ujutro);
- automatsko ažuriranje dogodit će se samo ako je računalo spojeno na struju (relevantno za prijenosna računala);
- nakon ažuriranja će se automatski ukloniti sve (sad nepotrebne) ovisnosti (engl. *dependencies*).

Prilikom instalacije *Unattended Upgradesa* korisniku se sugerira preuzimanje i alata *needrestart* koji će detektirati i ponovno pokrenuti procese koji su zahvaćeni ažuriranjem.

```
~ $sudo apt install unattended-upgrades
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  bsd-mailx default-mta | mail-transport-agent needrestart
The following NEW packages will be installed:
  unattended-upgrades
0 upgraded, 1 newly installed, 0 to remove and 1 not upgraded.
Need to get 48,7 kB of archives.
After this operation, 451 kB of additional disk space will be u
Get:1 http://hr.archive.ubuntu.com/ubuntu focal-updates/main ar
des all 2.3ubuntu0.1 [48,7 kB]
```

Slika 12 ***Prilikom instalacije Unattended Upgradesa predlaže se instalacija needrestarta***

Ponovno pokretanje (engl. *restart*) vrlo je bitno kako bi se uistinu počela koristiti nova inačica softvera, a ne stara koja se nalazi u radnoj memoriji. Npr. nakon što ažuriramo poslužiteljski softver *Apache*, moramo ga ponovno pokrenuti (engl. *restart*) kako bi se uistinu primijenila nova inačica, inače će se i dalje koristiti stara koja je već učitana u radnu memoriju.

Isto vrijedi i za ažuriranje jezgre (engl. *kernel*) operacijskog sustava, nakon ažuriranja mora se obavezno ponovno pokrenuti (engl. *reboot*).

Problem je što nije uvijek jednostavno odrediti koje procese treba ponovno pokrenuti – ako ažuriramo *libc* knjižnicu (engl. *library*), postoji mnogo procesa koji je koriste i sve ih treba ponovno pokrenuti. Alat *needrestart* olakšava taj proces na način da nakon svakog ažuriranja izvještava administratora o tome koje je procese potrebno ponovno pokrenuti, ili po potrebi čak i automatski ponovno pokreće sve što je potrebno. Ako želimo izbjeći

ponovno pokretanje kako ne bi došlo do problema na sustavu, možemo konfigurirati *needrestart* da nam na adresu e-pošte pošalje sve što treba ponovno pokrenuti, i zatim ručno provjeriti o čemu je riječ.

No ponekad, primjerice kada je ažurirana jezgra ili neki temeljni servis operacijskog sustava, potrebno je ponovno pokrenuti i cijelo računalo, što se može konfigurirati u *Unattended Upgradesu*.

Upute kako postaviti *Unattended Upgrades,* korak po korak, dostupne su na poveznici.

Ako želimo izbjeći da sustav bilo što sam automatski radi, možemo se predbilježiti na takozvane „*mailing* liste" Linux distribucije koje koristimo, kao što je prethodno navedeno. Jednom kad se predbilježimo, na adresu e-pošte stizat će nam poruke sa svim informacijama o sigurnosnim zakrpama za odabranu distribuciju, pa je na temelju njih moguće reagirati i ručno ažurirati pakete.

2.2 Administracija korisnika

Linux poslužitelj istovremeno koristi više vrsta korisnika s različitim ulogama i razinama ovlasti. Uobičajeno su to:

- korisnici s administratorskim privilegijama koji imaju najveću razinu ovlasti i potpunu kontrolu nad poslužiteljem,
- korisnici s visokom razinom privilegija, npr. programeri, administratori baze podataka itd.,
- korisnici s niskom razinom privilegija, npr. korisnik aplikacije.

Administracija korisnika okvirno se može podijeliti u dvije kategorije:

- upravljanje dozvolama na način da svaki korisnik ima upravo onoliko dozvola koliko mu treba;
- osiguravanje sigurnosti korisničkih računa, pogotovo administratorskih.

2.2.1 Upravljanje dozvolama

Linux operacijske sustave može istovremeno koristiti više korisnika. Kod višekorisničke okoline vrlo je bitno osigurati da si ti korisnici međusobno ne stvaraju probleme, tj. da pojedini korisnik ne može pristupiti podacima koje drugi korisnik ne želi da su svima vidljivi ili izmjenjivati datoteke ili direktorije koji pripadaju drugom korisniku. Također, nijedan korisnik ne bi smio moći ugroziti sustav, već je za njega odgovoran i na njega može utjecati samo korisnik s administratorskim privilegijama.

Upravljanje dozvolama upravo je mehanizam zaštite kontrole pristupa datotekama i direktorijima. Svaki korisnik treba imati upravo onoliko dozvola koliko mu je stvarno potrebno za rad, pri čemu posebice treba voditi računa o sljedećim smjernicama:

- ne smiju svi moći pokretati ili zaustavljati poslužiteljski softver;

- ne smiju svi moći mijenjati konfiguracijske datoteke ili programski kod pohranjen na poslužitelju;
- ne smiju svi moći brisati postojeće datoteke ili stvarati nove;
- ne smiju svi moći pregledavati programski kod ili neke osjetljive datoteke;
- ako se uspije kompromitirati jedan servis, to ne bi smjelo utjecati na ostatak operacijskog sustava;
- ako se uspije kompromitirati račun nisko privilegiranog korisnika, to ne bi smjelo utjecati na ostale korisničke račune ili poslužitelj.

Kako bi se gore navedene smjernice zadovoljile, potrebno je ispravno postaviti dozvole (engl. *permissions*).

2.2.1.1 Korisnici i grupe

Uobičajeno je da korisnički računi postoje za ljude koji koriste poslužitelj. Također, korisnički računi se stvaraju i za neke servise koji su pokrenuti na poslužitelju. Operacijski sustav prepoznaje svaki korisnički račun po njegovom jedinstvenom korisničkom identifikacijskom broju (engl. *User ID = UID*).

Informacije o korisnicima pohranjuju se u datoteci `/etc/passwd`. Svaka linija te datoteke sadrži informaciju o jednom korisničkom imenu, jedinstvenom korisničkom identifikacijskom broju, identifikacijskom broju grupe kojoj korisnik pripada, lokaciji korisnikovog *home* direktorija, itd.

```
systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:106::/nonexistent:/usr/sbin/nologin
syslog:x:104:110::/home/syslog:/usr/sbin/nologin
_apt:x:105:65534::/nonexistent:/usr/sbin/nologin
tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
uuidd:x:107:114::/run/uuidd:/usr/sbin/nologin
tcpdump:x:108:115::/nonexistent:/usr/sbin/nologin
avahi-autoipd:x:109:116:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin
usbmux:x:110:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
rtkit:x:111:117:RealtimeKit,,,:/proc:/usr/sbin/nologin
dnsmasq:x:112:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
cups-pk-helper:x:113:120:user for cups-pk-helper service,,,:/home/cups-pk-helper:/usr/sbin/nologin
speech-dispatcher:x:114:29:Speech Dispatcher,,,:/run/speech-dispatcher:/bin/false
avahi:x:115:121:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/usr/sbin/nologin
kernoops:x:116:65534:Kernel Oops Tracking Daemon,,,:/:/usr/sbin/nologin
saned:x:117:123::/var/lib/saned:/usr/sbin/nologin
nm-openvpn:x:118:124:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
hplip:x:119:7:HPLIP system user,,,:/run/hplip:/bin/false
whoopsie:x:120:125::/nonexistent:/bin/false
colord:x:121:126:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
geoclue:x:122:127::/var/lib/geoclue:/usr/sbin/nologin
pulse:x:123:128:PulseAudio daemon,,,:/var/run/pulse:/usr/sbin/nologin
gnome-initial-setup:x:124:65534::/run/gnome-initial-setup/:/bin/false
gdm:x:125:130:Gnome Display Manager:/var/lib/gdm3:/bin/false
sssd:x:126:131:SSSD system user,,,:/var/lib/sss:/usr/sbin/nologin
user:x:1000:1000:user,,,:/home/user:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
vboxadd:x:998:1::/var/run/vboxadd:/bin/false
user2:x:1001:1001::/home/user2:/bin/sh
user3:x:1002:1002::/home/user3:/bin/sh
john:x:1003:1003::/home/john:/bin/sh
ftp:x:127:134:ftp daemon,,,:/srv/ftp:/usr/sbin/nologin
Debian-exim:x:128:135::/var/spool/exim4:/usr/sbin/nologin
```

Slika 13 ***Sadržaj datoteke*** `/etc/passwd`

Korisnike organiziramo u skupine – grupe. Npr. svi korisnici sustava koji pripadaju odjelu računovodstva mogli bi biti članovi grupe naziva „racunovodstvo“. Takvo grupiranje korisnika olakšava upravljanje dozvolama jer se definirana pravila mogu primijeniti na sve korisnike grupe. Npr. ako postoji datoteka s privatnim informacijama zaposlenika potrebnih za isplatu plaće, nju bi smjeli vidjeti samo korisnici sustava koji rade u računovodstvu (i pripadaju grupi korisnika „racunovodstvo“). Ako dodijelimo pravo na čitanje toj grupi, svi njeni korisnici naslijedit će pravo i moći čitati datoteku, dok ostali korisnici operacijskog sustava neće (i ne bi ni smjeli). Ako se u odjelu računovodstva zaposli novi zaposlenik i operacijski sustav dobije novog korisnika, njega jednostavno dodamo u grupu „racunovodstvo“ i automatski nasljeđuje sva prava koja su definirana na razini grupe.

Takav pristup organiziranja korisnika u grupe i dodjeljivanja prava na razini grupe brži je i jednostavniji no što bi bio slučaj kod dodjeljivanja dozvola pojedinačnim korisnicima. Još je važnije da je tako bolji pregled nad ovlastima i njihovo održavanje.

Informacije o svim postojećim grupama pohranjene su u datoteci `/etc/group`. Svaki redak ove datoteke pohranjuje informacije o nazivu jedne grupe, identifikacijskom broju grupe (GID), te korisničkim imenima svih korisnika koji pripadaju grupi.

Ako želimo provjeriti kojim sve grupama pripadamo, unosimo naredbu:

```
$ groups
```

Za provjeru kojim sve grupama pripada neki drugi korisnik (onaj koji nije trenutno prijavljen), naredbi pridružujemo i korisničko ime, npr.:

```
$ groups user2
```

Slika 14 ***Provjera kojoj grupi pripadaju korisnici user (trenutno prijavljen) i user2***

Kao što je prikazano na slici 14, korisnik user2 trenutno pripada samo istoimenoj grupi user2 koja mu je podrazumijevano dodijeljena u trenutku stvaranja korisnika.

Novu grupu možemo stvoriti naredbom:

```
$ sudo groupadd <new_group_name>
```

npr. `$ sudo groupadd new_group`

Tada korisnika user2 možemo pridružiti novoj grupi naredbom:

```
$ sudo usermod –a –G <group_name> user2
```

Korisnik user2 sad pripada grupama user2 (podrazumijevana grupa) i novostvorenoj grupi new_group, kao što je vidljivo na slici 15:

```
user@debian:~$ sudo usermod -a -G new_group user2
user@debian:~$ groups user2
user2 : user2 new_group
```

Slika 15 ***Grupe kojima pripada korisnik user2***

Samo korisnici s administratorskim privilegijama smiju dodavati nove grupe i korisnike. Ako bi korisnik user2 (koji nije administrator) pokušao dodati novu grupu, ne bi uspio – dobio bi obavijest prikazanu na slici 16 koja obavještava korisnika da se ne nalazi u datoteci `sudoers` (u kojoj se nalaze članovi grupe sudo, tj. članovi s administratorskim privilegijama).

```
user2@debian:/$ sudo groupadd user2_group
[sudo] password for user2:
user2 is not in the sudoers file.  This incident will be reported.
```

Slika 16 ***Korisnici bez administratorskih privilegija ne mogu dodavati nove grupe***

Posebna vrsta korisnika je *root*, tj. superkorisnik (engl. *superuser*). Ovaj korisnički račun ima najveće privilegije i potpuni, neograničeni pristup svim naredbama i datotekama.

Ako smo prijavljeni kao korisnik *root*, u naredbenom retku prije naredbe ćemo obično vidjeti „#“ umjesto „$“ (iako *root* i to po volji može mijenjati).

Najbolje je **nikada** se ne prijavljivati na računalo kao *root* korisnik. Prisjetimo se, procesi imaju istu razinu dopuštenja kao i korisnik koji ih je pokrenuo. Ne želimo da se procesi pokreću s *root* ovlastima jer na taj način mogu naštetiti operacijskom sustavu!

Korisnike kojima su potrebne administratorske privilegije dodajemo u grupu `sudo` - na taj će način, dodavanjem naredbe `sudo` ispred neke naredbe (i lozinke na upit sustava), korisnik moći koristiti administratorske privilegije. Također, na ovaj način možemo pratiti aktivnosti svakog pojedinog korisnika s administratorskim privilegijama i detektirati čiji je račun kompromitiran ili tko obavlja zlonamjerne aktivnosti jer sad svaki korisnik koristi vlastiti korisnički račun i znamo o kome je riječ. I inače, treba učiniti svaki napor da **ni u kojoj situaciji i ni za koju potrebu dva fizička korisnika koriste isti korisnički račun**. Svatko uvijek mora koristiti samo i isključivo svoj korisnički račun. Jedino je tako moguće s pouzdanošću pratiti tko što radi u sustavu i postići odgovarajuću razinu sigurnosti.

2.2.1.2 Dopuštenja

Dopuštenja, tj. dozvole (engl. *permissions*) su prava koje korisnik ima na operacije s određenom datotekom unutar datotečnog sustava. Svaki pokrenuti proces imat će ista prava kao i korisnik koji ga je pokrenuo[1].

[1] postoje i neki procesi koji neće naslijediti prava korisnika koji ga je pokrenuo, već će naslijediti prava korisnika odnosno grupe koja je vlasnik datoteke (korištenjem zastavica *setuid* i *setgid*); primjerice, naredba `sudo` može običnom korisniku dati administratorska prava upravo zbog zastavice *setuid*, no to je napredna tema

Razlikujemo tri osnovne vrste dopuštenja koje korisnik može imati nad svakom datotekom:

- čitanje (engl. *read*) – pregledavanje sadržaja datoteke
- pisanje (engl. *write*) – prepisivanje, brisanje ili dodavanje novih podataka u datoteku
- izvršavanje (engl. *execute*) – izvršavanje kôda koji je pohranjen u datoteci

Ista ta dopuštenja primjenjuju se i na mapama, tj. direktorijima (engl. *directory*), ali je značenje drukčije:

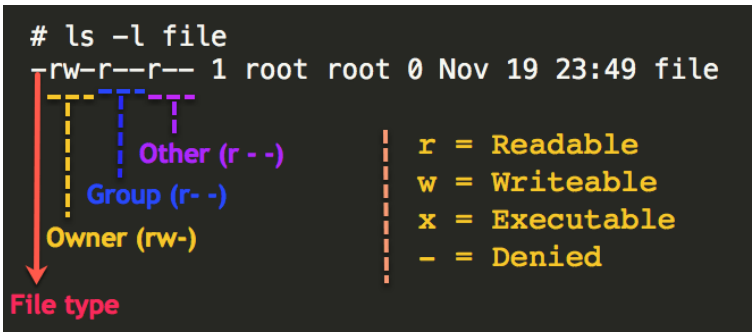
- čitanje (engl. *read*) – izlistavanje sadržaja direktorija
- pisanje (engl. *write*) – stvaranje, brisanje ili preimenovanje datoteka i (pod)direktorija
- izvršavanje (engl. *execute*) – dopušten ulazak u direktorij (npr. naredbom `cd` ili dvostrukim klikom na mapu ako je instalirano grafičko sučelje) i pristup postojećim datotekama i direktorijima unutar njega (ako dozvole te datoteke/direktorija dopuštaju odgovarajući pristup)

Za svaku datoteku i direktorij, moguće je dodijeliti tri skupa dozvola za čitanje/pisanje/izvršavanje – jedan skup dozvola za vlasnika, jedan za grupu i jedan za sve ostale.

Koja sve dopuštenja postoje za određenu datoteku ili direktorij možemo provjeriti unosom naredbe:

```
$ ls -l <ime_datoteke>
```

Prikazat će nam se ispis kao na slici 17:



Slika 17 ***Prikaz dopuštenja nad datotekom*** **[4]**

Za vježbu i razumijevanje, provjerimo dopuštenja nad nekim karakterističnim datotekama na Linux sustavu:

```
user@debian:~$ ls -l /etc/passwd
-rw-r--r-- 1 root root 1686 Apr  2 17:30 /etc/passwd
```

Slika 18 *Dopuštenja nad datotekom* `/etc/passwd`

Kao što vidimo na slici 18, dopuštenja za datoteku `/etc/passwd` s informacijama o korisnicima sustava su:

- vlasnik datoteke (*root*) smije čitati i pisati u datoteku (rw-)
- grupa korisnika kojoj je dodijeljeno pravo (istoimena grupa *root* u kojoj se nalazi samo korisnik *root*) smije čitati datoteku (r--)
- svi ostali korisnici smiju čitati datoteku (r--)

```
~ $ls -l /bin/cp
-rwxr-xr-x 1 root root 153976 ruj   5  2019 /bin/cp
```

Slika 19 Dopuštenja nad datotekom `/bin/cp`

Dopuštenja za datoteku u kojoj je pohranjen programski kod za izvršavanje naredbe `cp` koja kopira datoteke i direktorije:

- vlasnik datoteke (*root*) smije čitati, pisati i izvršavati datoteku, tj. naredbu `cp` (rwx)
- grupa korisnika kojoj je dodijeljeno pravo (*root*) smije čitati i izvršavati datoteku, tj. naredbu `cp` (r-x)
- svi ostali korisnici mogu čitati i izvršavati datoteku, tj. naredbu `cp` (r-x)

I za kraj, pogledajmo dopuštenja za neke od direktorija i datoteka koje se nalaze unutar korijenskog (*root*) direktorija:

```
user@debian:~$ ls -l /
total 64
lrwxrwxrwx  1 root root     7 Sep 26  2019 bin -> usr/bin
drwxr-xr-x  3 root root  4096 Apr  1 01:45 boot
drwxr-xr-x 17 root root  3340 Apr 15 10:18 dev
drwxr-xr-x 84 root root  4096 Apr 15 10:19 etc
drwxr-xr-x  5 root root  4096 Mar 31 15:10 home
```

Slika 20 Dio rezultata izvršavanja naredbe „`ls -l /`“

Primijetimo da na prvom mjestu retka s dopuštenjima više nemamo „-“ koji je označavao uobičajenu datoteku. Sad se susrećemo s oznakama:

- „l“ što označava simboličku poveznicu (engl. *symlink*)
- „d“ što označava direktorij/mapu (engl. *directory/folder*)

Kao što vidimo, direktorij `bin` je zapravo simbolička poveznica na direktorij `usr/bin`. Zato, unatoč tome što naizgled ima dozvole koje dopuštaju sve, za njega zapravo efektivno vrijede iste dozvole kao i za direktorij na kojega pokazuje (`usr/bin`). Za direktorije `boot`, `dev`, `etc`, i `home` koji nisu simboličke poveznice vrijedi:

- vlasnik (*root*) smije izlistavati sadržaj direktorija, pristupati datotekama i direktorijima unutar njega (pod uvjetom da dozvole na toj datoteci/direktoriju to dopuštaju) te stvarati, preimenovati i brisati datoteke i direktorije (rwx)
- grupa kojoj je dodijeljeno pravo (*root*) smije izlistavati sadržaj direktorija te pristupati datotekama i direktorijima unutar njega (r-x)
- svi ostali smiju izlistavati sadržaj direktorija te pristupati datotekama i direktorijima unutar njega (r-x)

Osnovna dopuštenja za svaku datoteku se pohranjuju pomoću 9 bitova: 3 za svaku kategoriju korisnika (vlasnik, grupa s dodijeljenim pravom, ostali korisnici).

Za izmjenu dopuštenja koristi se naredba `chmod`, što je kratica za „*change mode*“. Za demonstraciju, stvorimo i novu datoteku *demo.txt*:

```
user@debian:~$ touch demo.txt
user@debian:~$ ls -l demo.txt
-rw-r--r-- 1 user user 0 Apr 15 11:10 demo.txt
```

Slika 21 Stvaranje i pregled dopuštenja za datoteku demo.txt

U ovom slučaju, podrazumijevano (engl. *by default*) dodijeljena dopuštenja su redom:

- vlasnik (user) smije čitati i pisati u datoteku (rw-)
- grupa s dodijeljenim pravima (user) smije čitati datoteku (r--)
- svi ostali korisnici smiju čitati datoteku (r--)

Podrazumijevano (engl. *default*) dodijeljena dopuštenja određena su postavkom zvanom *umask* (*file mode creation mask*) i moguće ih je mijenjati istoimenom naredbom `umask`.

Bitovi koji pohranjuju informacije o dopuštenjima su u tom slučaju postavljeni na: 110 | 100 | 100, pri čemu 1 označava da je neka radnja dopuštena, a 0 da nije. Kada bismo ovaj svaki od ta tri binarna zapisa pretvorili u oktalni, dobili bismo 6 | 4 | 4.

Upravo takav zapis možemo koristiti za specificiranje dopuštenja naredbom `chmod`.

Ako želimo svima dati dopuštenje za sve radnje, moramo postaviti bitove na 111 | 111 | 111, tj. unijeti naredbu:

```
$ chmod 777 demo.txt
```

Ako želimo svima dati dopuštenje za čitanje i izvršavanje, a vlasniku datoteke i za pisanje, moramo postaviti bitove na način: 111 | 101 | 101, tj. unijeti naredbu

```
$ chmod 755 demo.txt
```

2.2.1.3 Access Control List

Ponekad ćemo, kad izvršimo naredbu:

```
$ ls -l
```

na kraju popisa dopuštenja vidjeti i znak „+“ kao na slici 22:

```
me@home:~$ ls -l script.sh
-rwxrwxr-x+ 1 me me 0 Oct 28 21:31 script.sh
```

Slika 22 ***Rezultat izvršavanja naredbe ls -l za datoteku script.sh*** **[4]**

Taj „+“ govori nam da je datoteci pridružen i ACL, tj. lista za kontrolu pristupa (engl. *Access Control List*). Korištenjem ACL-a vlasnik datoteke može fleksibilnije dodjeljivati dopuštenja ostalim korisnicima dvjema osnovnim naredbama [5]:

- `setfacl` – postavljanje ACL-a za određenu datoteku
- `getfacl` – pregledavanje ACL-a koja se odnosi na određenu datoteku.

Na slici 23 prikazan je sadržaj ACL-a koja se odnosi na datoteku script.sh.

```
me@home:~$ getfacl -l script.sh
# file: script.sh
# owner: me
# group: me
user::rwx
group::r-x
other::r-x
```

Slika 23 ***Sadržaj ACL-a za datoteku script.sh*** **[5]**

Kako bismo mogli stvoriti ACL, prvo moramo pregledati postavke i omogućiti korištenje ACL-a na datotečnom sustavu.

Ako bismo htjeli korisniku *john* dodijeliti prava za čitanje, pisanje i izvršavanje, unijeli bismo naredbu kao na slici 24. Pritom zastavica `-m` označava izmjenu liste, a korištenjem zastavice `-x` bismo mogli obrisati postojeća dopuštenja iz ACL-a.

```
me@home:~$ setfacl -m u:john:rwx script.sh
me@home:~$ getfacl script.sh
# file: script.sh
# owner: me
# group: me
user::rwx
user:john:rwx
group::rwx
mask::rwx
other::r-x
```

Slika 24 ***Dodjeljivanje prava korisniku john*** **[5]**

Dodijeliti prava cijeloj grupi (u ovom slučaju accounts), a ne pojedinačnom korisniku, možemo sljedećom naredbom koja specificira da je riječ o grupi, navodi njen naziv i dopuštenja koja će joj se dodijeliti:

```
$ setfacl -m g:accounts:rwx script.sh
```

```
me@home:~$ setfacl -m g:accounts:rwx script.sh
me@home:~$ getfacl script.sh
# file: script.sh
# owner: me
# group: me
user::rwx
user:john:rwx
group::rwx
group:accounts:rwx
mask::rwx
other::r-x
```

Slika 25 ***Dodjeljivanje prava grupi accounts*** **[5]**

Postojeća dopuštenja mogu se ukloniti postavljanjem zastavice „x“ i navođenjem koje se dopuštenje želi ukloniti. Npr. naredba unesena na slici 26 uklonit će sva dopuštenja za korisnika *john*:

```
me@home:~$ setfacl -x u:john script.sh
```

Slika 26 ***Oduzimanje svih prava korisniku john*** **[5]**

Kao i korisniku, dopuštenja se mogu ukinuti i grupama:

```
me@home:~$ setfacl -x g:accounts script.sh
```

Slika 27 ***Oduzimanje svih prava grupi accounts*** **[5]**

2.2.1.4 Servisi

Pregledom datoteke `/etc/passwd` (s podacima o svim korisnicima) vidljivo je da velik dio korisnika u stvari nisu ljudski korisnici, već razni servisi, poput npr. FTP-a.

```
systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:106::/nonexistent:/usr/sbin/nologin
syslog:x:104:110::/home/syslog:/usr/sbin/nologin
_apt:x:105:65534::/nonexistent:/usr/sbin/nologin
tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
uuidd:x:107:114::/run/uuidd:/usr/sbin/nologin
tcpdump:x:108:115::/nonexistent:/usr/sbin/nologin
avahi-autoipd:x:109:116:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin
usbmux:x:110:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
rtkit:x:111:117:RealtimeKit,,,:/proc:/usr/sbin/nologin
dnsmasq:x:112:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
cups-pk-helper:x:113:120:user for cups-pk-helper service,,,:/home/cups-pk-helper:/usr/sbin/nologin
speech-dispatcher:x:114:29:Speech Dispatcher,,,:/run/speech-dispatcher:/bin/false
avahi:x:115:121:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/usr/sbin/nologin
kernoops:x:116:65534:Kernel Oops Tracking Daemon,,,:/:/usr/sbin/nologin
saned:x:117:123::/var/lib/saned:/usr/sbin/nologin
nm-openvpn:x:118:124:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
hplip:x:119:7:HPLIP system user,,,:/run/hplip:/bin/false
whoopsie:x:120:125::/nonexistent:/bin/false
colord:x:121:126:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
geoclue:x:122:127::/var/lib/geoclue:/usr/sbin/nologin
pulse:x:123:128:PulseAudio daemon,,,:/var/run/pulse:/usr/sbin/nologin
gnome-initial-setup:x:124:65534::/run/gnome-initial-setup/:/bin/false
gdm:x:125:130:Gnome Display Manager:/var/lib/gdm3:/bin/false
sssd:x:126:131:SSSD system user,,,:/var/lib/sss:/usr/sbin/nologin
user:x:1000:1000:user,,,:/home/user:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
vboxadd:x:998:1::/var/run/vboxadd:/bin/false
user2:x:1001:1001::/home/user2:/bin/sh
user3:x:1002:1002::/home/user3:/bin/sh
john:x:1003:1003::/home/john:/bin/sh
ftp:x:127:134:ftp daemon,,,:/srv/ftp:/usr/sbin/nologin
Debian-exim:x:128:135::/var/spool/exim4:/usr/sbin/nologin
```

Slika 28 ***Sadržaj datoteke /etc/passwd***

Ljudske korisnike od servisa obično možemo razlikovati po UID-u: uobičajena je praksa da servisi imaju dodijeljen identifikacijski broj manji od 1000, a ljudski korisnici veći.

Ispravna sigurnosna praksa je da svaki servis treba biti pokrenut kao zasebni korisnik koji ima dodijeljen minimum nužnih prava potrebnih za rad.

Npr. ako smo pratili službene upute za instalaciju poslužiteljskog softvera *Apache*, primijetili smo da možemo stvoriti novog korisnika (npr. www-data) i dodijeliti mu samo dopuštenja za čitanje i izvršavanje datoteka *web* stranice (npr. HTML i PHP datoteke).

Takvo odjeljivanje servisa sigurnosni je mehanizam koji osigurava da, ako već netko uspije napasti i preuzeti kontrolu nad poslužiteljskim softverom, neće uspjeti preuzeti kontrolu nad cijelim računalom jer ni sam poslužiteljski softver nema *root* ovlasti ili pristup direktorijima ostalih servisa ili korisnika (osim ako su mu potrebni za ispravan rad). Kad bi *Apache* procesi bili pokrenuti s pravima korisnika *root*, šteta koju bi napadač mogao napraviti bila bi daleko veća jer ako bi korisnik uspio napasti *Apache*, mogao bi izvoditi radnje na poslužitelju s *root* ovlastima.

Bitna napomena: samo *root* smije imati UID=0 jer to označava da ima sva dopuštenja na sustavu!

Ako nakon unosa naredbe:

```
$ awk -F: '($3 == "0") {print}' /etc/passwd
```

vidimo još kojeg korisnika osim

```
root:x:0:0:root:/root:/bin/bash
```

moramo ga ili obrisati ili biti sigurni da je korisnik autoriziran za takvu razinu ovlasti.

2.2.2 Sigurnost korisničkih računa

Sigurnost korisničkih računa odnosi se na sigurnost autentifikacije. Neki primjeri nesigurne autentifikacije su:

- jednostavne lozinke koje napadač lako „probije“;
- neefikasan sustav ponovnog postavljanja lozinke poput odgovaranja na pitanja „U kojem sam gradu rođen“ itd. na koja i napadači mogu pogoditi odgovor (u slučaju ponovnog postavljanja lozinke, korisnik mora odgovoriti na neka pitanja kako bi potvrdio svoj identitet i mogao ponovno postaviti lozinku);
- neograničen broj pokušaja prijave koja omogućava napadaču da isprobava kombinacije dok ne pogodi ispravnu lozinku (engl. *brute-force* attack)

Iako je administrator sustava najčešće upoznat sa sigurnosnim praksama i pretpostavka je da će inicijalna lozinka koju dodijeli korisniku biti snažna, problem nastaje kad korisnik sam mijenja svoju lozinku. Korisnik može odabrati neku jednostavnu i predvidljivu lozinku poput „`password`“, „`123456`“, „`zagreb`“, itd.

Takve lozinke bi napadi uzastopnim pogađanjem (engl. *brute-force*) i rječnici koji se koriste za *dictionary attack* uspješno probili u svega nekoliko sekundi. Više o takvim napadima može se pročitati u dokumentu Nacionalnog CERT-a.

Ako uzmemo u obzir da se korisnici mogu udaljeno prijavljivati na poslužitelj (protokolima SSH ili RDP), jasno je zašto nesigurne lozinke korisnika (pogotovo onih s visokim privilegijama) značajno ugrožavaju sigurnost poslužitelja. Kad napadač uspije visoko privilegiranom korisniku preuzeti račun, može napraviti sve što može i taj korisnik na sustavu, što znači da je stekao neovlašten pristup i kontrolu nad poslužiteljem.

Nažalost, čak i kad su korisnici svjesni sigurnosnog rizika koji donose nesigurne lozinke, pokušavaju postaviti lozinke koje im je jednostavno zapamtiti. U nastavku poglavlja opisat ćemo postavljanje politike sigurnih lozinki (engl. *Secure Password Policy*), odnosno koje sve osnovne korake možemo poduzeti da korisnika „zaštitimo od njega samoga“, tj. spriječimo ga da postavi nesigurnu lozinku.

Neke uobičajene sigurnosne smjernice su:

1) Zahtijevati minimalnu duljinu lozinke
2) Zahtijevati kompleksnost lozinke
3) Natjerati korisnika da periodički mijenja lozinku

4) Zabraniti ponovno postavljanje već korištenih lozinki

5) Ograničiti koliko puta korisnik može unijeti pogrešnu lozinku, nakon čega će mu se zaključati račun

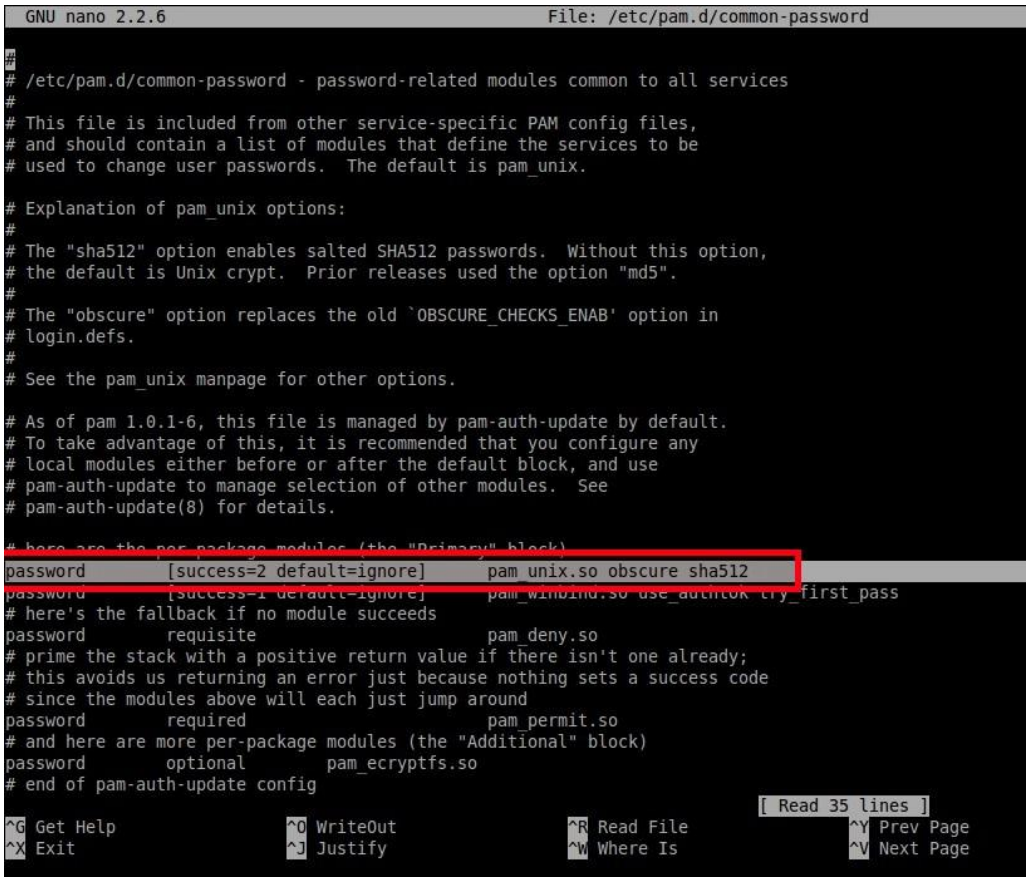
6) Provjeriti sigurnost postojećih lozinki pokušajima napada, tj. *crackanjem*, upravo onako kako bi to napravio i napadač

2.2.2.1 Postavljanje minimalne duljine lozinke

Sigurna lozinka trebala bi imati minimalno 10 znakova, i to pod uvjetom da nije riječ o jednostavnoj ili predvidljivoj lozinki.

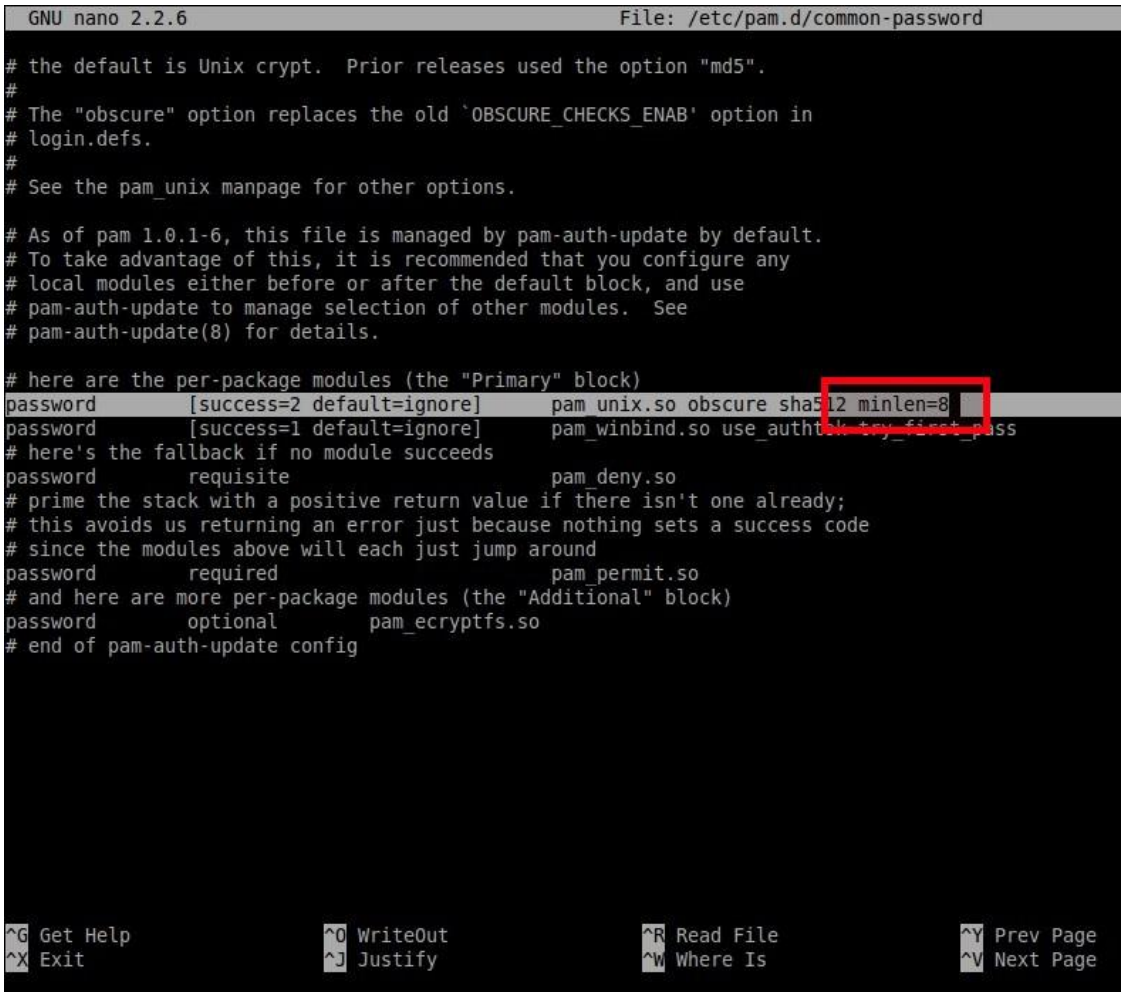
Na *DEB-based* distribucijama konfiguracijske datoteke vezane uz lozinke i autentifikaciju nalaze se unutar mape `/etc/pam.d/`.

Kako bi se postavila minimalna dopuštena duljina lozinke, potrebno je, s administratorskim privilegijama, otvoriti datoteku `/etc/pam.d/common-password` i pronaći redak koji je označen na slici 29:



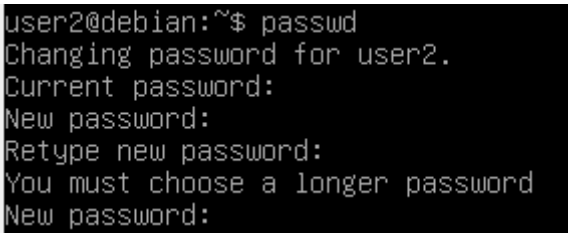
Slika 29 *Sadržaj datoteke /etc/pam.d/common-password* [6]

U označeni redak dodajemo parametar „`minlen=8`“ gdje, u ovom primjeru, broj 8 predstavlja minimalnu duljinu lozinke.



Slika 30 ***Dodavanje upute da lozinka mora imati najmanje osam znakova*** **[6]**

Ako bi sad korisnik user2 pokušao promijeniti svoju lozinku i unijeti neku koja se sastoji od samo šest znakova, ne bi uspio i sustav bi ga obavijestio da mu je lozinka prekratka, kao što je prikazano na slici 31.



Slika 31 ***Korisnik ne može postaviti lozinku kraću od osam znakova***

Na *RPM-based* distribucijama minimalnu dopuštenu duljinu lozinke može se provjeriti naredbom:

```
$ sudo grep "^minlen" /etc/security/pwquality.conf
```

a promijeniti se može naredbom:

```
$ sudo authconfig --passminlen=8 --update
```

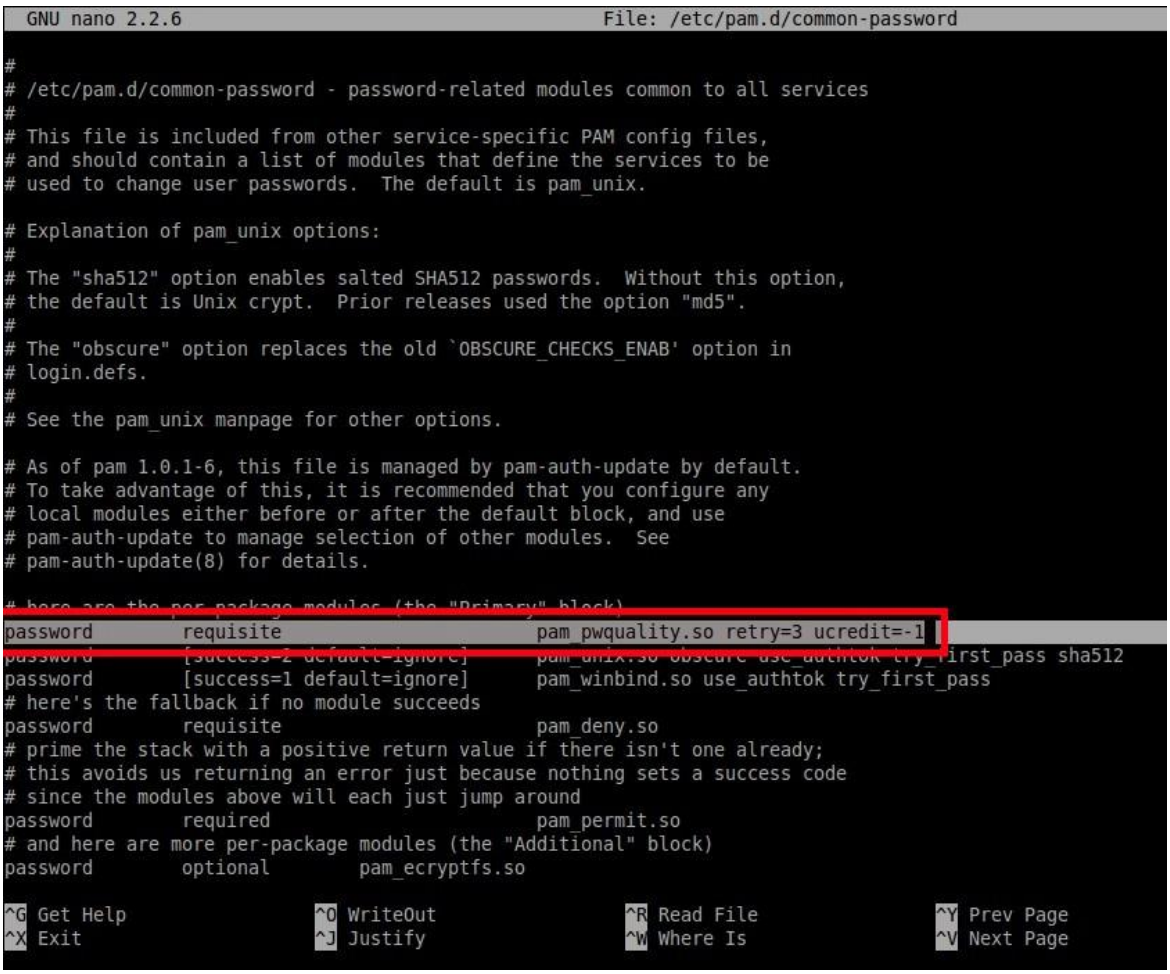
2.2.2.2 Postavljanje tražene kompleksnosti lozinke

Postaviti traženu kompleksnost lozinke znači definirati od koliko se različitih vrsta znakova, (velikih slova, malih slova, specijalnih znakova) treba sastojati lozinka.

Na *DEB-based* sustavima za to je potrebno instalirati knjižnicu `libpam-pwquality`, koja, provjerava kvalitetu lozinke:

```
$ sudo apt-get install libpam-pwquality
```

Zatim s administratorskim ovlastima otvorimo datoteku `/etc/pam.d/common-password` u uređivaču teksta.



Slika 32 *Datoteka /etcp/pam.d/common-password* [6]

Kako bi sustav znao koju kategoriju/klasu znakova tražiti od korisnika prilikom postavljanja lozinke, dajemo mu neki/e od sljedećih parametara::

- `ucredit` – velika slova (engl. *uppercase*)
- `dcredit` – mala slova (engl. *lowercase*)
- `ocredit` – specijalni znakovi (engl. *other letters*) i

- `minclass` – broj različitih klasa znakova od kojih se sastoji lozinka

Ako bismo postavili parametar `ucredit=-1` (kao što je pokazano na slici 32), dali bismo uputu da se u lozinki mora nalaziti najmanje jedno veliko slovo.

Na *RPM-based* sustavima isto ćemo postići izvršavanjem nekih od sljedećih naredbi s administratorskim privilegijama:

`# authconfig --enablereqlower –update` // obavezno malo slovo

`# authconfig --enablerequpper –update` // obavezno veliko slovo

`# authconfig --enablereqdigit –update` // obavezna znamenka

`# authconfig --enablereqother –update` // obavezan specijalan znak

Napomena: na starijim *RPM-based* sustavima (6.x) potrebno je izmijeniti datoteku `/etc/pam.d/system-auth` tako da u liniju koda

```
password requisite pam_cracklib.so try_first_pass retry=3 type=
minlen=8
```

dodamo parametre

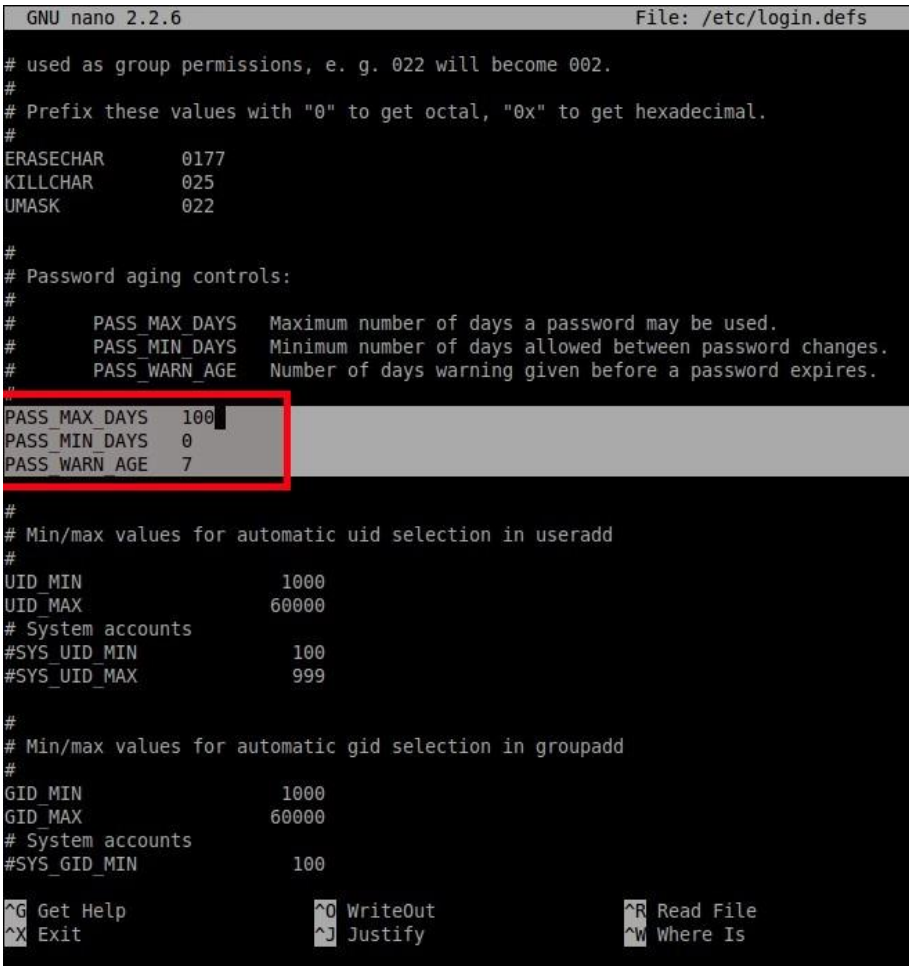
`dcredit=-1 ucredit=-1 lcredit=-1 ocredit=-1`.

2.2.2.3 Natjerati korisnika da periodički mijenja lozinku

Iako ovo nije moderna preporučena praksa (jer korisnici postavljaju gotovo identične lozinke kojima samo npr. inkrementalno mijenjaju brojčani sufiks što rezultira lozinkama poput: `Zagreb01`, `Zagreb02`...), postoje situacije kad moramo pratiti ovu regulativu.

Kako bismo natjerali korisnika da periodički mijenja lozinku, moramo mu ograničiti period valjanosti trenutne lozinke.

Taj period definira se u datoteci `/etc/login.defs` uređivanjem parametara `PASS_MAX_DAYS`, `PASS_MIN_DAYS` i `PASS_WARN_AGE` kao što je prikazano na slici 33:



Slika 33 ***Uređivanje parametara datoteke*** `/etc/login.defs`

Na primjeru pokazanom na slici 33, korisnici sustava morat će promijeniti lozinku svakih 100 dana i obavijest o promjeni lozinke će im se početi pojavljivati tjedan dana prije isteka valjanosti trenutne lozinke.

No, potrebno je napomenuti da će se ove postavke primijeniti samo na novostvorene korisnike (one koje administrator doda naredbom `adduser`).

Kako bismo ih primijenili i na već postojeće korisnike sustava, unosimo sljedeću naredbu:

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

pri čemu je `<days>` broj dana valjanosti lozinke, a `<username>` korisničko ime korisnika na kojeg želimo primijeniti ovo pravilo. Ako korisnik ne promijeni lozinku, zaključat će mu se račun.

Kad su korisnici prisiljeni redovito mijenjati lozinke, često se snađu na način da osmisle sigurnu lozinku koju će sustav prihvatiti, a zatim inkrementiraju (povećavaju za jedan) broj koji se nalazi na kraju lozinke. Možemo im zabraniti takve kombinacije ili kombinacije koje su previše slične prethodnim lozinkama koristeći *pam_cracklib* biblioteku.

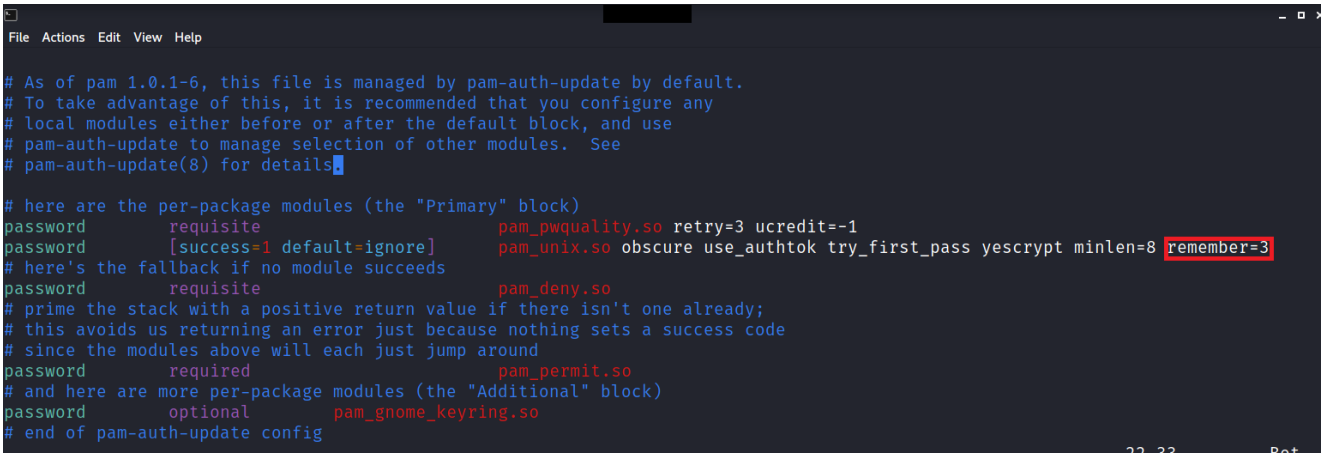
Više detalja o dodatnim mogućnostima ograničavanja lozinke koju će korisnik postaviti može se pronaći na službenoj stranici. Upute za konfiguraciju i korištenje *pam_cracklib* biblioteke na *DEB-based* i *RPM-based* distribucijama mogu se pronaći na blogu.

2.2.2.4 Zabrana ponovnog postavljanja već korištenih lozinki

Nije rijetkost da korisnici izmjenjuju dvije ili tri lozinke koje im je lako zapamtiti. Ako te lozinke koriste i na drugim mjestima (npr. za prijavu na razne *web* stranice), to predstavlja sigurnosni rizik. Siliti korisnika da redovito mijenja lozinke nema smisla ako ne spriječimo da te lozinke ne mogu biti uvijek iste.

Korisnika možemo spriječiti da ponovi prethodnu ili nekoliko prethodnih lozinki.

Ako želimo spriječiti korisnika da ponovi prethodnu ili nekoliko prethodnih lozinki, za *DEB-based* sustave, to možemo konfigurirati u datoteci `/etc/pam.d/common-password` dodavanjem parametra `remember=X`, gdje je X broj prethodnih lozinki koje se ne smiju ponoviti. Ako npr. napišemo `remember=3`, nova lozinka koju korisnik postavlja neće moći biti ista kao prethodne tri.



Slika 34 ***Dodan parametar koji određuje koliko lozinki se pamti***

Za *RPM-based* sustave možemo urediti datoteku `/etc/pam.d/system-auth` tako da dodamo `remember=3` na kraj linije:

```
password sufficient pam_unix.so sha512 shadow nullok try_first_pass use_authtok remember=3
```

2.2.2.5 Ograničiti broj unosa pogrešne lozinke

Kako bismo spriječili *brute-force* napade uzastopnim pogađanjem lozinke, nakon određenog broja neuspješnih pokušaja prijava na računalo ili putem SSH protokola, potrebno je zaključati korisnički račun na neko vrijeme.

Na taj će način potencijalni *brute-force* napad biti značajno usporen jer će napadač morati čekati definirano vrijeme da se račun otključa.

Za *DEB-based* sustave izmjenjujemo datoteku `/etc/pam.d/common-auth` na način da dodamo sljedeću liniju koda:

```
auth    required        pam_tally2.so onerr=fail deny=3 unlock_time=600 audit
```

Slika 35 ***Linija koda za zaključavanje korisničkog računa nakon tri neuspjela pokušaja prijave*** **[7]**

Objašnjenje parametara:

- `onerr`: u slučaju pogreške (primjerice ako se ne može otvoriti datoteka u koju se zapisuje broj pokušaja prijave), proglasi radnju neuspješnom
- `deny=X`: nakon X neuspješna pokušaja prijave, zaključaj korisnički račun
- `unlock_time=X`: račun će biti zaključan X sekundi. U slučaju sa slike 35, račun će biti zaključan 600 sekundi, tj. 10 minuta
- `audit`: zapiši zapise o pokušajima prijavama u datoteku dnevnika

Ako želimo da se navedena pravila primijene i na *root* korisnika, dodajemo i parametre `even_deny_root` i `root_unlock_time`:

```
pam_tally2.so onerr=fail deny=3 unlock_time=600 audit even_deny_root root_unlock_time=600
```

Slika 36 ***Dodavanje parametara za zaključavanje i root korisničkog računa*** **[7]**

U primjeru sa slike 36 će *root* račun biti zaključan nakon tri neuspješna pokušaja prijave i bit će zaključan 600 sekundi, odnosno 10 minuta.

Nakon izmjena, bitno je ponovno pokrenuti servis *ssh* kako bi se primijenila nova pravila:

```
$ sudo systemctl restart sshd
```

Za *RPM-based* distribucije dodajemo sljedeće tri linije koda točno navedenim redoslijedom u datoteke `/etc/pam.d/password-auth` i `/etc/pam.d/system-auth` i zatim ponovno pokrenemo *ssh* servis:

```
auth     required       pam_faillock.so preauth silent audit deny=3 unlock_time=600
auth     [default=die]  pam_faillock.so authfail audit deny=3 unlock_time=600
account  required       pam_faillock.so
```

Slika 37 ***Linije koda koje treba dodati u `/etc/pam.d/password-auth` & `/etc/pam.d/system-auth`*** **[7]**

Objašnjenje parametara je identično kao i za *DEB-based* sustave.

Kako bi se navedena pravila primijenila i na *root* korisnika, dodajemo i parametar `even_deny_root` kao što je prikazano na slici 38:

```
auth    required      pam_faillock.so preauth silent audit even_deny_root deny=3 unlock_time=600
auth    [default=die] pam_faillock.so authfail audit even_deny_root deny=3 unlock_time=600
```

Slika 38 ***Dodavanje parametra za zaključavanje root korisničkog računa*** **[7]**

2.2.2.6 Provjera sigurnosti lozinki

Jedan od čestih napada kojima se pokušava preoteti korisnički račun je pogađanje lozinke takozvanim *dictionary attackom*. *Dictionary attack*, tj. napad rječnikom je isprobavanje različitih lozinki popisanih u rječniku (kojeg možemo sami sastaviti ili preuzeti s *weba*) u nadi da je korisnik za svoju lozinku odabrao neku od njih.

Možemo konfigurirati *pwquality* da, kad korisnik postavlja (novu) lozinku, provjeri nalazi li se lozinka u nekom zadanom rječniku.

Kao zadani rječnik je korisno koristiti neki poput top X lozinki npr. s Probable Wordlists repozitorija.

Pwquality također ima podršku i za provjeru je li korisničko ime dio nove lozinke i zabranu takve lozinke. To je iznimno korisno jer napadači, kad sastavljaju vlastite rječnike vrlo često isprobavaju različite kombinacije s korisničkim imenima.

Kako bi napadaču bilo što teže pogoditi lozinku koristeći *dictionary attack*, korisno je slijediti sljedeća pravila:

- lozinke se moraju sastojati od 10 ili više znakova,
- lozinke moraju koristiti znakove iz više (najbolje svih) različitih skupova znakova (mala slova, velika slova, brojevi, simboli),
- lozinke se ne smiju primarno sastojati od neke riječi iz rječnika, imena ili prezimena korisnika (ili varijacije), korisničkog imena, riječi vezane uz aplikaciju/poslužitelj/servis i slično,
- lozinke trebaju biti jedinstvene (ne smije se koristiti ista lozinka na više mjesta),
- lozinka ne smije biti neka od općenito često korištenih lozinka; to je primjerice moguće provjeriti pomoću "Pwned Passwords" servisa

Primjerice, lozinka "`Zagreb456!`" nije sigurna unatoč tome što ima 10 znakova te sadrži znakove iz četiri različite skupine.

Napadaču je lako pogoditi takvu lozinku jer se primarno sastoji od jedne riječi uz neke transformacije (prvo slovo veliko, dodan broj i specijalni znak).

Jedan način za smišljanje sigurne lozinke koju je moguće lako zapamtiti je korištenje rečenica ili fraza (eng. *passphrase*).

Primjerice, moguće je smisliti rečenicu "`Jako često koristim Internet.`" te ju malo izmijeniti, npr. "`jako ČESTO99 koristim? Internet`" i tako dobiti prilično sigurnu lozinku koju ujedno nije teško zapamtiti.

Još jednom je važno napomenuti – čak i izrazito složene lozinke mogu biti nesigurne ako se koriste na više mjesta.

Kako bi si korisnik olakšao korištenje jedinstvenih i sigurnih lozinki, može koristiti tzv. upravitelje lozinkama (eng. *password managers*). Više informacija o upraviteljima lozinkama dostupno je u dokumentu Nacionalnog CERT-a KeePass.

Za dodatnu sigurnost (primjerice za posebno osjetljive servise), korisno je implementirati višefaktorsku autentifikaciju (eng. *multi-factor authentication*).

2.3 Izloženost osjetljivih podataka

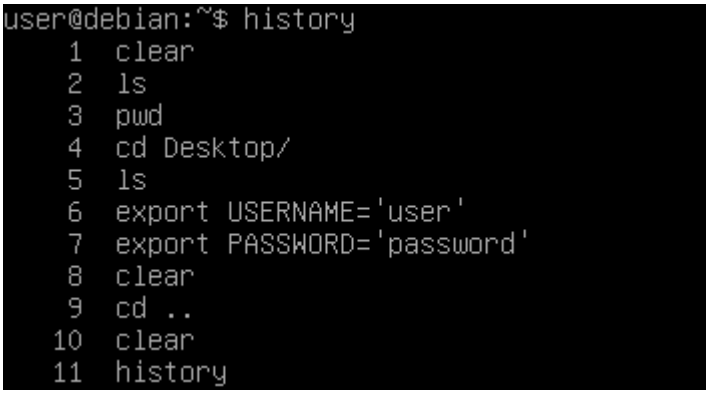
Osjetljivi se podaci ne bi trebali pohranjivati na poslužiteljima osim ako to nije nužno jer, ako uspije pristupiti ili kompromitirati poslužitelj, napadač bi uspio pristupiti i tim podacima, saznati povjerljive informacije i dodatno ih zloupotrijebiti za daljnju kompromitaciju.

Kad je riječ o povjerljivim podacima, to mogu biti privatne informacije o zaposlenicima ili klijentima, tekstualne datoteke s popisom korisničkih imena i lozinki, i slično.

No, bespotrebna izloženost osjetljivih podataka javlja se i u:

1) *Bash history*

   Sve naredbe koje smo upisali u naredbenu konzolu spremaju se u tzv. *bash history*. Ako smo nekom prilikom upisali lozinku u konzolu, ona ostaje zabilježena u *bash historyju* kao što je prikazano na slici 39 i onaj tko ima pristup *bash historyju* je može iščitati.



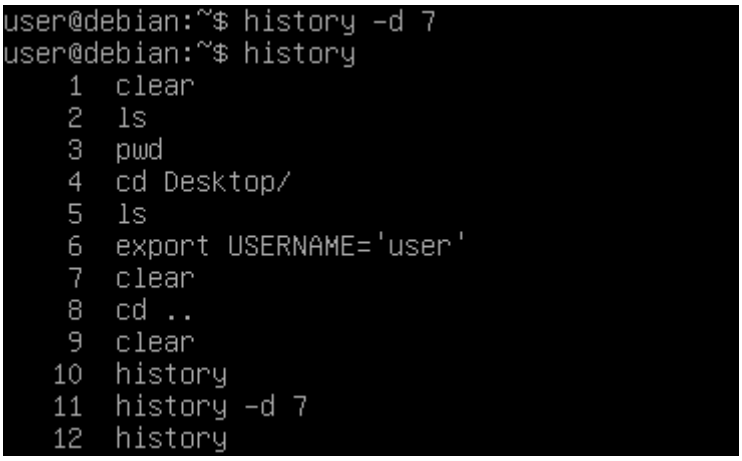
Slika 39 ***Lozinka je zabilježena u bash historyju***

   Kako bismo spriječili iščitavanje osjetljivih informacije iz *bash historyja*, možemo obrisati postojeći zapis ili spriječiti da se zapisi zabilježe.

   U slučaju brisanja postojećeg zapisa, potrebno je unijeti naredbu:

```
$ history -d <broj_retka>
```

   Na slici 40 prikazan je sadržaj *bash historyja* nakon brisanja zapisa s lozinkom koji se na prethodnoj slici nalazio u 7. retku.



Slika 40 ***Lozinka je obrisana iz bash historyja***

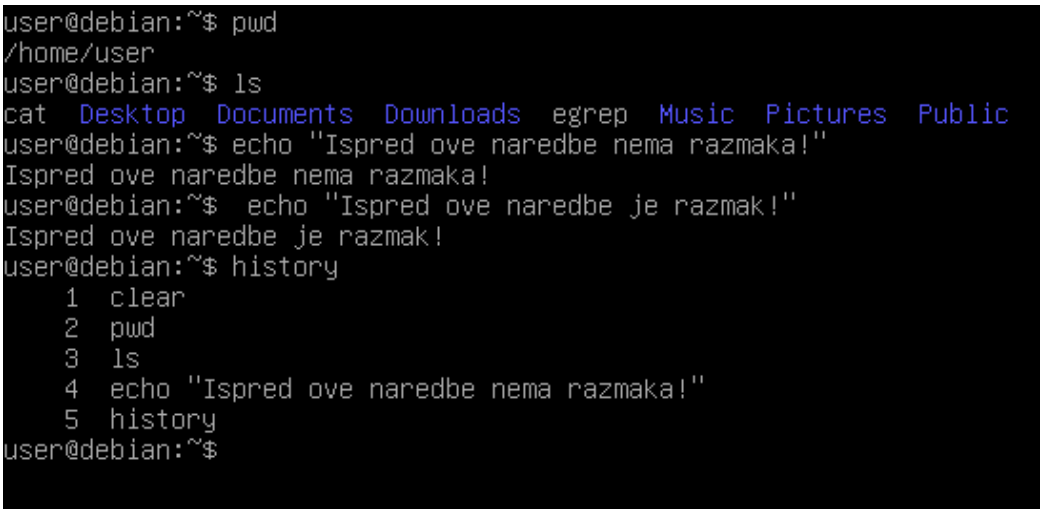
Možemo konfigurirati za koje naredbe ne želimo da se spreme u *bash history* pomoću varijable okoline HISTIGNORE. Uobičajeno su to kratke naredbe poput `ls`, `exit`, `history` i sl. i njihovo spremanje se sprječava zbog urednosti i jer ih je lako upisati pa nemamo potrebu lako ih dohvatiti *up* i *down* tipkama.

Npr. ako bismo postavili HISTIGNORE na sljedeći način:

```
$ export HISTIGNORE="history"
```

izvršavanje naredbe `history` ne bi se spremilo u *bash history*.

Na isti način možemo spriječiti spremanje naredbi s osjetljivim informacijama. Također, ako se ispred određene naredbe doda određeni znak (često je konfigurirano da je taj znak razmak, odnosno *space*), to je uputa da se naredba ne treba spremiti u *bash history*. Na slici 41 prikazano je kako se naredba kojoj ne prethodi razmak sprema u *bash history*, a naredba kojoj prethodi razmak ne sprema.



Slika 41 ***Naredba kojoj prethodi razmak nije zabilježena u bash historyju***

2) SSH privatni ključ

   Samo javni SSH ključ treba biti pohranjen na poslužitelju (kako bi se korisnik mogao prijaviti). Nema potrebe i nije preporučljivo na poslužitelj pohraniti i privatni ključ. Privatni ključ treba biti tajan, tj. poznat samo vlasniku koji je za njega odgovoran i treba ga dobro čuvati od ostalih. Unatoč tome, događa se da korisnici zabunom kopiraju i privatni ključ na poslužitelj, ili da SSH ključ generiraju na poslužitelju, pa im od tog postupka generiranja privatni ključ ostane na poslužitelju. Ako se isti SSH ključ koristi za pristup većem broju poslužitelja (što je pogrešno, ali često praksa korisnika), onda napadač koji kompromitira jedan poslužitelj (i tako dođe do privatnog ključa) može ostvariti pristup i drugim poslužiteljima.

2.4 Uklanjanje nepotrebnih usluga i zatvaranje nepotrebno otvorenih priključaka

Preporuka za sigurnu konfiguraciju poslužitelja je uvijek zatvoriti sve nepotrebno otvorene priključke i ukloniti nepotrebne usluge, tj. servise (engl. *services*) i pozadinske procese (engl. *daemons*).

Na većinu Linux distribucija već su instalirani neki mrežni servisi koji slušaju i primaju dolazne konekcije s interneta. Neke od tih mrežnih servisa naš poslužitelj možda ne treba, i zato ih moramo detektirati, ukloniti i zatvoriti priključke na kojima slušaju.

Npr. možda smo, kako bismo nešto testirali na poslužitelju, instalirati *ftp* servis, ali sad nam više ne treba. Ne postoji razlog zašto bi servis i dalje trebao biti pokrenut na poslužitelju jer ničemu ne služi, a predstavlja nam dodatni sigurnosni rizik.

Na taj način smanjit ćemo tzv. površinu napada (engl. *attack surface*) na koju nas napadač može pokušati napasti i minimizirati ćemo rizik od iskorištavanja ranjivosti koje se mogu pojaviti u tim programima.

Provjeriti sve pokrenute servise možemo naredbom:

```
$ systemctl --type=service --state=running
```

Ako na popisu vidimo neki servis kojeg ne koristimo (uzmimo za primjer *Apache*), možemo ga zaustaviti naredbom:

```
$ systemctl stop apache2
```

provjeriti da je isključen naredbom:

```
$ systemctl status apache2
```

i zatim onemogućiti (spriječiti njegovo automatsko pokretanje prilikom pokretanja računala) naredbom:

```
$ systemctl disable apache2
```

Ako je servis onemogućen prethodnom naredbom (`systemctl disable <ime_servisa>`), nakon ponovnog pokretanja računala on se neće automatski pokrenuti, već ga je potrebno ručno pokrenuti. Ako nije onemogućen, onda će se svaki put automatski pokrenuti ponovnim pokretanjem računala čak i ako je bio prethodno zaustavljen (naredbom `system stop <ime_servisa>`). Osim isključivanja servisa, možemo i potpuno obrisati instaliran softverski paket za servis. Naredba za brisanje ovisi o softveru za upravljanje paketima na distribuciji, ali npr. za Ubuntu bismo unijeli naredbu:

```
$ sudo apt purge <package_name>
```

Isto pravilo vrijedi i za priključke (engl. *ports*). Iako ne koristimo neki priključak, ako ga ostavimo otvorenog, potencijalni napadač ga može otkriti alatima za skeniranje poput *nmapa* i postoji mogućnost da preko njih napadne poslužitelj.

Zato je bitno detektirati sve otvorene priključke, razmisliti koji od njih nam nisu potrebni, i zatim ih zatvoriti.

Popisati sve otvorene priključke možemo unosom sljedeće naredbe s administratorskim privilegijama:

```
$ sudo netstat -tulp
```

Isto je moguće napraviti novom naredbom `ss` (*socket statistics*):

```
$ sudo ss -tulp
```

Ako na popisu otvorenih priključaka vidimo neki kojeg ne koristimo ili smo ga nekad koristili, ali sad nam više ne treba, možemo ga zatvoriti na dva načina:

- ugasiti servis koji koristi taj priključak npr. naredbom (obično se radi o servisu koji čeka veze/pakete na nekom priključku):
    - `$ sudo systemctl stop <ime_servisa>`
- *Firewallom* blokirati pristup tom priključku npr., ako se koristi, naredbom *ufw*:
    - `$ sudo ufw deny <port>`

U sljedećem poglavlju detaljnije ćemo opisati vatrozid (engl. *firewall)* i kako ga možemo koristiti.

2.5 Vatrozid (engl. firewall)

Vatrozidom možemo definirati tko (koja IP adresa) se smije spojiti na što (IP adresu, priključak).

U definiranju pravila vatrozida možemo biti vrlo specifični oko prometa i spajanja koje dopuštamo i koje odbijamo. Sigurnim postavljenjem vatrozida može se spriječiti neovlašten upad u sustav i skeniranje otvorenih priključaka na našem poslužitelju alatima poput *nmapa*.

Postoje dvije vrste vatrozida: nezavisni uređaj koji se spaja tipično između lokalne mreže i javnog Interneta i sav promet ide kroz njega ako on to dozvoli, te vatrozid instaliran na poslužitelju koji štiti (samo) aplikacije na tom računalu. Ovdje govorimo o vatrozidu koji je na samom Linux računalu, ne o posebnom uređaju ili softveru na usmjerivaču (engl. *router*) mreže.

Najsigurnije je dopustiti samo nužan promet, a sve ostale IP pakete odbiti.

Iako se novije Linux distribucije prebacuju na korištenje softverskog paketa *nftables* za postavljanje i upravljanje vatrozidom, trenutno je i dalje na većini poslužitelja moguće koristiti softverski paket *iptables* koji je dugo vrijeme bio glavni softverski paket za upravljanje vatrozidom na Linux distribucijama i podrazumijevano (engl. *by default*) je instaliran na njih.

Iptables kontrolira dolazni i odlazni mrežni promet. Za svaki paket provjerava sadržaj i ako paket zadovoljava definirani uvjet, propušta ga dalje, a ako ne zadovoljava, odbacuje.

Npr. ako smo definirali da dopuštamo spajanje na IP adresu poslužitelja, na priključak 80 samo računalima iz privatne mreže, nitko s javne mreže neće se moći spojiti na naš poslužitelj.

Pravila se primjenjuju po definiranom redoslijedu: od vrha prema dolje. Ako paket kojeg *iptables* pregledava zadovolji neki od prvih uvjeta, daljnji se uvjeti neće pregledati, i zato je bitno ispravno postaviti redoslijed pravila.

Ako bismo htjeli konfigurirati da naš poslužitelj prima dolazne konekcije samo na priključku 80, a sve ostale pakete odbaci, napisali bismo sljedeće pravilo:

```
iptables -A INPUT -p tcp -m tcp --dport 80 -m state --state
NEW,ESTABLISHED -j ACCEPT

iptables -A INPUT -j DROP
```

Nakon postavljanja parametara vatrozida, moramo ga obavezno ponovno pokrenuti.

Kao što vidimo, *iptables* nije trivijalan za korištenje jer moramo unositi mnogo parametara i pravila mogu biti jako duga. Iz tog razloga su se razvila dva softverska paketa koja olakšavaju korištenje *iptablesa*:

- *FirewallD* – sustav više razine za postavljanje vatrozida koji u pozadini koristi *iptables*, korišten na *RPM-based* Linux distribucijama

- *UFW* – aplikacija za jednostavnu konfiguraciju pravila vatrozida koji u pozadini koristi *iptables*, namijenjena *DEB-based* Linux distribucijama

Nećemo ulaziti u detalje sintakse za postavljanje pravila jer se sve informacije mogu pronaći pretraživanjem *weba*, ali uputit ćemo čitatelja na neka pitanja koja si mora postaviti prilikom konfiguracije vatrozida:

- Smije li poslužitelju pristupiti itko s javne mreže ili možemo pristup ograničiti samo na privatnu mrežu?

- Smiju li korisnici iz svih dijelova svijeta pristupiti našem poslužitelju ili ipak očekujemo spajanje samo iz određene države i možemo ograničiti raspon IP adresa?

- Na kojim priključcima očekujemo dolazne konekcije? Na onima na kojima ne očekujemo, ne smijemo ni dozvoliti spajanje.

2.6 Ostalo

2.6.1 SSH

SSH (engl. *Secure Shell*) protokol koristi se za udaljeno spajanje na poslužitelj i potrebno ga je posebno dodatno zaštititi kako bi se spriječilo da se i napadač uspješno udaljeno spoji. Neke smjernice su:

- **Redovito ažuriranje softvera**
  Kao što smo već nekoliko puta spomenuli, jako je bitno redovito ažurirati softver, a to uključuje i *OpenSSH* softverski paket koji se obično koristi kao implementacija servisa SSH.

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

  Prijava lozinkom se može omogućiti za *root* korisnika i kako bismo spriječili da se napadač spoji na priključak 22, unese korisničko ime „root“ i zatim napadne uzastopnim pogađanjem (engl. *brute-force*) naš poslužitelj dok ne pogodi lozinku, administrator se ne smije prijavljivati *root* korisničkim računom, već korisničkim računom sa sudo privilegijama. Na taj način napadač mora pogoditi i korisničko ime i lozinku, a ne samo lozinku superkorisnika.

  Kako bismo onemogućiti ssh prijavu *root* korisniku, u konfiguracijskoj datoteci `/etc/ssh/sshd_config` pod odjeljkom *Authentication* trebamo pronaći liniju koda (ako ne postoji, trebamo ju dodati, ili ako je dio komentara, trebamo maknuti znak „#“):

  ```
  PermitRootLogin prohibit-password
  ```

  i izmijeniti je u

  ```
  PermitRootLogin no
  ```

  Alternativno, možemo dopustiti prijavu *root* korisničkim računom samo s određenih IP adresa navođenjem IP adrese u liniju *AllowUsers* u konfiguracijskoj datoteci:

  ```
  AllowUsers root@192.168.1.110
  ```

Na ovaj će se način dopustiti spajanje s *root* korisničkim računom samo računalu na IP adresi 192.168.1.110.

- **Korisnici kojima je dopušteno spajanje na računalo SSH protokolom**
  U konfiguracijsku datoteku `/etc/ssh/sshd_config` dodajemo (ako već ne postoji) liniju:

  ```
  AllowUsers korisnik1 korisnik2 korisnik3
  ```

  koja određuje da se samo korisnici s korisničkim imenom korisnik1, korisnik2 i korisnik3 smiju spojiti SSH protokolom na računalo. Svim ostalim korisnicima spajanje će biti zabranjeno.

  Kao i za *root* korisnika, i za ostale korisnike možemo navesti s koje IP adrese im dopuštamo spajanje.

  Također, moramo onemogućiti spajanje korisničkim računima koji nemaju lozinku, tj. imaju praznu lozinku. Za to je potrebno izmijeniti datoteku konfiguracijsku datoteku `/etc/ssh/sshd_config` na način da postavimo liniju koda:

  ```
  PermitEmptyPasswords no
  ```

- **SSH ključevi**

  Podrazumijevano se korisnici SSH protokolom na računalo prijavljuju korisničkim imenom i lozinkom – što je u redu samo ako smo sigurni da imaju snažne lozinke. No često se dogodi da nisu sve lozinke snažne, pa je zato dobra sigurnosna praksa onemogućiti prijavljivanje korisničkim imenom i lozinkom i postaviti prijavljivanje ključevima.

  Prijavljivanje ključevima istovremeno osigurava dodatnu sigurnost, ali i ubrzava spajanje na poslužitelj.

  Kako bismo mogli koristiti prijavljivanje ključevima, moramo generirati privatni i javni ključ za šifriranje naredbom:

  ```
  $ ssh-keygen
  ```

  Preporučeno je postaviti i lozinku (engl. *password*) za datoteku ključa, kako se ključem ne bi mogao koristiti onaj tko ga nekako uspije saznati, već bi morao znati i lozinku.

  Kad utvrdimo da se uspješno možemo spojiti koristeći ključeve, onemogućujemo prijavu korisničkim imenom i lozinkom sljedećom linijom koda u konfiguracijskoj datoteci:

  ```
  PasswordAuthentication no
  ```

- **Ograničavanje broja pokušaja prijave**

  Kao što smo već spomenuli u poglavlju o administraciji korisnika, korisnik se ne smije beskonačno puta moći pokušati prijaviti jer to napadaču olakšava napade uzastopnim pogađanjem. U konfiguracijskoj datoteci definiramo koliko se puta korisnik smije pokušati prijaviti prije no što mu zaključamo račun i onemogućimo daljnje pokušaje prijave:

  ```
  MaxAuthTries 3
  ```

- **Promjena priključka na kojem SSH usluga osluškuje i prima konekcije**

  Iako se sigurnosni stručnjaci razilaze u mišljenju oko ove sigurnosne prakse, opisat ćemo i ovaj koncept.

  Iako je ovaj korak nepotreban ako smo isključili prijavu korisničkim imenom i lozinkom i poduzeli ostale sigurnosne mjere, često se spominje kao jedan od osnovnih savjeta za zaštitu poslužitelja na kojem se nalazi SSH usluga.

  SSH podrazumijevano prima konekcije na priključku 22. Poslužitelji se toga pridržavaju zbog konvencije i jer se klijentski programi spajaju na njega. Ako bismo zamijenili taj priključak, morali bismo to eksplicitno reći svakome tko se pokušava spojiti SSH protokolom na naš poslužitelj. Jednostavno skeniranje poslužitelja alatom poput *nmapa* odmah bi otkrilo na kojem priključku usluga SSH sluša i napadač bi mogao jednostavno preusmjeriti napade na taj priključak.

  Iako se čini beskorisno promijeniti priključak, izazvati zbunjenost svih koji se spajaju na računalo, a zauzvrat usporiti napad nekoliko sekundi ili minuta koliko je *nmapu* potrebno da otkrije novi priključak, ova tehnika ipak ima i jednu značajnu prednost – spriječit će automatizirane i neiskusne pokušaje napada kojih zapravo ima najviše na internetu.

  Razni *botovi* skeniraju cijelu mrežu i pokušavaju izvesti pripremljene automatizirane napade. Ako uspiju, odlično, a ako ne, idu dalje na sljedeći poslužitelj. Takvim *botovima* se ne isplati potrošiti vrijeme na skeniranje poslužitelja i traženja gdje sluša SSH, već preskoče taj poslužitelj i napadaju sljedeći.

  U konfiguracijskoj datoteci pronalazimo liniju koda na kojoj je definiran priključak:

  ```
  Port 22
  ```

  i izmijenimo ga u neki drugi, npr. 2222.

  Napominjemo, ako smo onemogućili prijavu *root* korisničkim računom, ograničili broj pokušaja spajanja na poslužitelj, naveli eksplicitno koji se korisnici smiju spojiti i s kojih IP adresa i onemogućili prijavu korisničkim imenom i lozinkom, poslužitelj je već dovoljno siguran i navedeni *botovi* ga također neće napasti već preskočiti.

Uspješne napade na poslužitelj možemo spriječiti pregledavanjem datoteka dnevnika, promatranjem pokušava li se netko previše puta spojiti isprobavajući razne vjerodajnice, i zatim blokirati IP adresu s koje zlonamjerni promet dolazi.

Kako to ne bismo radili ručno ili pisali skripte, dostupni su nam razni alati koji automatski pregledavaju datoteke dnevnika i sprječavaju napade na poslužitelj blokiranjem IP adresa s kojih dolaze zahtjevi. Neki od takvih alata su *SSHGuard*, *Fail2ban* ili *DenyHosts*.

2.6.2 Fail2ban

Nažalost, nije moguće unaprijed znati otkud će dolaziti zlonamjeran promet i unaprijed ga blokirati vatrozidom.

Iz tog razloga razvili su se sustavi zaštite temeljeni na automatskoj detekciji i izolaciji određenih napada na sustav.

Jedan od takvih sustava je i *Fail2ban* - softverski dodatak vatrozidu koji automatizirano čita datoteke dnevnika na poslužitelju (engl. *logs*) i pri uočavanju određenih nepravilnosti ili sumnjivih aktivnosti reagira sukladno konfiguraciji (npr. prilagodi pravila vatrozida tako da blokira određenu IP adresu) ili obavještava sistemskog administratora koji zatim poduzima daljnje korake.

Iako *Fail2Ban* nije jedini sustav zaštite temeljen na ovom principu, popularan je iz razloga što je vrlo jednostavan za korištenje.

Fail2Ban može gotovo u potpunosti eliminirati neke jednostavne napade koji se oslanjaju na uzastopno spajanje i slanje poruka mrežnim servisima. Neki primjeri takvih napada su DoS napadi na aplikacijskoj razini ili napadi uzastopnim pogađanjem (engl. *brute-force*). *Fail2ban* će primijetiti velik broj zahtjeva s određene IP adrese i zatim dodati pravilo u vatrozid kojim zabranjuje daljnje spajanje toj adresi na poslužitelj i slanje zahtjeva, pri čemu je napadač spriječen u izvršavanju napada.

Fail2Ban može otežati i vremenski usporiti i sofisticiranije napade ozbiljnijih napadača čime pomaže sigurnosnim stručnjacima i sistemskim administratorima da dobiju na vremenu i obrane se od raznih pokušaja napada.

Detalji o korištenju i konfiguraciji alata *Fail2ban* mogu se pročitati u dokumentu Nacionalnog CERT-a.

2.6.3 Host intrusion detection

Host intrusion detection sustavi nadziru poslužiteljsko računalo, sve aktivnosti koje se događaju na njemu i stanje svih resursa (procesor, memorija, I/O, mreža…) kako bi detektirali neuobičajene ili zlonamjerne aktivnosti.

Host intrusion detection sustavi mogu:

- nadzirati aktivnosti korisnika i sustava,
- procjenjivati integritet datoteka važnih za rad sustava,

- prepoznati poznate uzorke napadačkih aktivnosti tijekom rada sustava
- identificirati neuobičajene i zlonamjerne aktivnosti,
- sami ispraviti pogreške u konfiguraciji sustava ili brzo reagirati na napad dok on traje.

Pritom se koriste dvjema osnovnim tehnikama:

- Tehnike utemeljene na prepoznavanju uzoraka nepoželjnog ponašanja (engl. *Misuse Detection*)

  Sličan pristup koriste i antivirusni alati - imaju bazu s uzorcima nepoželjnih ponašanja, i kad detektiraju takvo ponašanje na računalu, smatraju ga zlonamjernim.

- Tehnike utemeljene na proučavanju nepravilnosti i odstupanja od uobičajenog rada sustava i ponašanja korisnika (engl. *Anomaly Detection*)

  Sustav „uči“ kako se korisnici uobičajeno ponašaju za računalom, i ako naiđu na odstupanja, smatraju ponašanje zlonamjernim.

Za razliku od vatrozida koji štiti računalo od „vanjskog svijeta“, HID sustavi štite računalo i od potencijalnih unutarnjih zlonamjernih korisnika, npr. napadača koji su ostvarili ograničeni pristup sustavu ili od zaposlenika.

Jedan primjer ovakvog sustava je *Tripwire*. *Tripwire* je alat za provjeru integriteta koji upozorava sistemske administratore ili druge osobe zadužene za sigurnost sustava na izmjene u sistemskim datotekama. Izmjene u sistemskim datotekama mogu biti posljedica napadačevih aktivnosti ili aktivnosti zlonamjernog softvera.

Jedan primjer naprednijeg HID sustava je OSSEC (engl. *Open Source HIDS SECurity*) – besplatan softver otvorenog koda koji analizira datoteke dnevnika, analizira integritet datoteka, nadzire aktivnosti na računalu, detektira zlonamjerni softver (engl. *malware*) i reagira na određenu prijetnju.

3 Zaključak

Sigurno konfiguriran poslužitelj nužan je preduvjet za priključivanje poslužitelja na mrežu i omogućavanja pristupanju korisnicima iz više razloga:

- zaštita poslovanja;
- zaštita korisnika koji se spajaju na poslužitelj;
- zaštita posjetitelja aplikacije.

Tehnika dodatnog osiguravanja poslužitelja korištenjem dodatnih alata, sigurnim konfiguriranjem postavki poslužitelja i instaliranog softvera te definiranjem i praćenjem sigurnosnih politika naziva se *security hardening*, što u prijevodu znači sigurnosno ojačavanje.

Sigurnosno ojačavanje zahtijeva malo truda, a uklonit će rizik od većine automatiziranih napada koji se na *webu* neprestano pokušavaju izvesti. Sigurnosno ojačan poslužitelj znatno je teže uspješno napasti nego poslužitelj s podrazumijevanom (engl. *by default*) zaštitom i konfiguracijom.

U ovom dokumentu prikazane su najčešće osnovne pogreške s kojima smo se imali iskustva susresti prilikom višegodišnjeg penetracijskog testiranja i sigurnosnih provjera Linux poslužitelja:

- instalacija nepotrebnog softvera, neažuriranje ili zakašnjelo ažuriranje softvera,
- dodjeljivanje previše dopuštenja koja rezultiraju time da korisnik/servis može kompromitirati ostale korisnike ili servise, a čak i operacijski sustav računala,
- dopuštanje postavljanja nesigurnih lozinki koje napadač može *crackati* i prijaviti se na poslužitelj s ukradenim korisničkim vjerodajnicama,
- pokrenute nepotrebne i nekorištene usluge i otvoreni nekorišteni priključci,
- nesigurno konfiguriran vatrozid,
- izloženost osjetljivih podataka,
- …

Za svaki od navedenih propusta objašnjene su ispravne sigurnosne prakse koje bi se trebale primijeniti.

Na raspolaganju su nam i različiti alati koji nam mogu olakšati zaštitu poslužitelja i alarmirati nas kad se događa nešto sumnjivo te ponekada i zaustaviti napad – *Fail2ban*, *Host Intrusion Detection* sustavi poput *Tripwirea* i sl., a bitno je za spomenuti i napredne *Mandatory Access Control* sustave poput *SELinuxa* i *AppArmora*. Takvi sustavi strogo provjeravaju tko smije čemu pristupiti i nitko osim administratora ne može mijenjati pravila. Na taj način, uz ispravno postavljena pravila, kompromitacija jednog servisa ili korisnika neće rezultirati kompromitacijom ostalih procesa na poslužitelju.

Naravno, bitan element sigurnosne politike je i stalna spremnost da će poslužitelj biti uspješno napadnut – u tom slučaju treba moći čim prije detektirati napad, oporaviti se od napada i ponovno podignuti sustav, pri čemu su neizostavne pričuvne kopije (engl. *backups*).

4 Literatura

1. **W3Techs.** Comparison of the usage statistics of Linux vs. Windows for websites. *W3Techs.* [Mrežno] 2021. [Citirano: 20. listopada 2021.] https://w3techs.com/technologies/comparison/os-linux,os-windows.

2. —. Usage statistics of Linux for websites. *W3Techs.* [Mrežno] 2021. [Citirano: 20. listopada 2021.] https://w3techs.com/technologies/details/os-linux.

3. **Gatlan, Sergiu.** Apache Bug Lets Normal Users Gain Root Access Via Scripts. *Bleeping Computer.* [Mrežno] 2. travnja 2019. [Citirano: 1. travnja 2020.] https://www.bleepingcomputer.com/news/security/apache-bug-lets-normal-users-gain-root-access-via-scripts/.

4. **The Geek Diary.** Understanding Basic File Permissions and ownership in Linux. *The Geek Diary.* [Mrežno] [Citirano: 21. travnja 2020.] https://www.thegeekdiary.com/understanding-basic-file-permissions-and-ownership-in-linux/.

5. **fasihxkhatib.** Understanding Linux Permissions. *Linux Academy.* [Mrežno] [Citirano: 28. studenog 2016.] https://linuxacademy.com/guide/12593-understanding-linux-permissions/.

6. **SK.** How To Set Password Policies In Linux. *OS TechNix.* [Mrežno] 14. ožujka 2020. [Citirano: 22. travnja 2020.] https://www.ostechnix.com/how-to-set-password-policies-in-linux/.

7. **Kumar, Pradeep.** Lock User Account After n Failed Login attempts in Linux. *Linuxtechi.* [Mrežno] 3. siječnja 2020. [Citirano: 4. svibnja 2020.] https://www.linuxtechi.com/lock-user-account-incorrect-login-attempts-linux/.

8. **W3Techs.** Usage statistics of web servers. *W3Techs.* [Mrežno] 2020. [Citirano: 4. svibnja 2020.] https://w3techs.com/technologies/overview/web_server.

9. **-.** sshd_config manual page. [Mrežno] 9. listopada 2021. [Citirano: 25. listopada 2021.] https://man.openbsd.org/sshd_config.

10. —. Release Notes - OpenSSH. [Mrežno] 11. kolovoza 2015. [Citirano: 25. listopada 2021.] https://www.openssh.com/txt/release-7.0.