HRVATSKA AKADEMSKA I ISTRAŽIVAČKA MREŽA
CROATIAN ACADEMIC AND RESEARCH NETWORK

Sigurnost UNIX zaporki

CCERT-PUBDOC-2006-11-175

CARNet CERT u suradnji s LS&S

Sigurnosni problemi u računalnim programima i operativnim sustavima područje je na kojem CARNet CERT kontinuirano radi.

Rezultat toga rada ovaj je dokument, koji je nastao suradnjom CARNet CERT-a i LS&S-a, a za koji se nadamo se da će Vam koristiti u poboljšanju sigurnosti Vašeg sustava.

CARNet CERT, www.cert.hr - nacionalno središte za **sigurnost računalnih mreža** i sustava.

LS&S, www.lss.hr - laboratorij za sustave i signale pri Zavodu za elektroničke sustave i obradbu informacija Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu.

Sadržaj

1. Uvod

U današnje vrijeme kad računala imaju vrlo široku primjenu, potrebno je osigurati pouzdanu autentikaciju korisnika kako bi se ograničio pristup podacima i resursima. Zaporke su još uvijek najčešće korišteni način autentikacije korisnika te je stoga potrebno posvetiti posebnu pažnju njihovoj sigurnosti i provjeri njihove ranjivosti kako bi se osigurala sigurnost sustava i spriječio neovlašteni pristup podacima.

U ovom dokumentu opisuje se način upravljanja zaporkama na Unix i ostalim *nix sustavima, načini provjere njihove ranjivosti te se daju kratke smjernice za povećanje njihove sigurnosti. Također, dan je kratak pregled PAM modula koji posreduju u procesu autentikacije između programa kojima se korisnik autenticira i samog sustava za pohranu zaporki. Na kraju dokumenta raspoloživ je i opis dva alata za provjeru ranjivosti Unix zaporki (*John the Ripper* i *Crack*).

2. Općenito o mogućnostima skrivanja zaporki i upravljanje sustavom zaporki

Potreba za korištenjem zaporki na računalima pojavila se sredinom 80-ih godina prošlog stoljeća kad su se računala počela široko primjenjivati u uredskom poslovanju. U tvrtkama su vrlo brzo shvatili da je potrebno ograničiti pristup podacima. Najlakši način identificiranja korisnika bio je dodjeljivanje korisničkog imena svakom korisniku. Međutim, samo korisničko ime nije pružalo dovoljnu sigurnost jer je bilo vrlo lako pogoditi tuđe korisničko ime. Zbog toga se javila potreba za unosom dodatne tajne zaporke kojom će se korisnici identificirati.

Kod primjene zaporki vrlo je važna politika njihovog korištenja. Pod politikom zaporki podrazumijeva se niz pravila koje zaporke moraju zadovoljavati da bi mogle biti korištene, ali i niz mjera kojima se osigurava provođenje tih pravila. Važan dio politike zaporki je i edukacija korisnika o važnosti uporabe jakih zaporki te, općenito, o važnosti računalne sigurnosti. Također, bitno je i osigurati ravnomjerno provođenje politika prema svim korisnicima i bez iznimki.

Jakom zaporkom smatra se ona koju je teško odgonetnuti kako čovjeku, tako i računalu. Sama definicija jake zaporke mijenja se ovisno o okruženju, ali također i s vremenom. Zaporke koje su smatrane jakima prije pet ili deset godina danas su slabe. Glavni razlog za to je brzina računala. Zaporke koje su se nekad razbijale godinama danas mogu biti razbijene u roku od nekoliko sati. Stoga je povećanje procesorske snage dostupnih računala potrebno pratiti i "pojačanjem" primjenjivanih zaporki.

Neke od osnovnih smjernica za kreiranje jakih zaporki su:

- ograničenje valjanosti na rok od 45 dana,
- minimalna duljina od 10 znakova,
- zaporka mora sadržavati po barem jedno slovo (malo i veliko), brojku i posebni znak,
- slova, brojke i posebni znakovi moraju biti izmiješani, a ne samo dodani na kraj (npr. "kkzvjvpz&88" je loša zaporka dok je "sd&jz2s9#mo" dobra),
- ne smiju se koristiti riječi iz rječnika, čak ni ako pripadaju stranom jeziku,
- ne smije se koristiti 5 prethodnih zaporki,
- zaporka se ne smije promijeniti u toku 10 dana od njenog postavljanja (iako se čini da je ovo pravilo u suprotnosti s preporukom o čestom mijenjaju zaporki, ono je bitno kako bi se korisnike onemogućilo u pokušaju njegovog zaobilaženja jer bez zadanog minimalnog vremena valjanosti zaporke, korisnici je mogu redefinirati pet puta za redom, čime se ponovno mogu vratiti na njen izvoran oblik),
- nakon nekoliko neuspjelih pokušaja prijave korisnički se račun blokira na određeno vrijeme, itd.

Potrebno je napomenuti kako su ovo samo smjernice i da ih je moguće i potrebno prilagođavati sredini u kojoj se zaporke koriste.

Dobar način za kreiranje jake zaporke je uzimanje prvih slova riječi neke lako pamtljive rečenice. U tako dobivenom nizu slova moguće je zatim zamijeniti određena slova znamenkama koja im nalikuju. Primjerice 3 umjesto E, 4 umjesto A i sl. Nakon toga potrebno je još samo ubaciti posebne znakove u tako dobiveni niz da bi se dobila jaka zaporka koja će biti lako pamtljiva.

Od samog početka korištenja zaporki do danas traje nadmudrivanje korisnika i administratora prilikom osmišljavanja zaporki. Sa stajališta administratora potrebno je zaporke učiniti što jačima, odnosno napadaču onemogućiti njihovo lako pogađanje. Uvođenjem određenih pravila za kreiranje zaporki (minimalna duljina zaporke, sadržavanje brojeva i posebnih znakova u zaporci, itd.) same zaporke postaju teže pamtljive pa korisnici pribjegavaju njihovom zapisivanju, što predstavlja sigurnosni rizik jednako kao i korištenje jednostavnih zaporki koje su lako pamtljive. Kao moguća rješenja ovog nadmudrivanja nameću se jednokratne zaporke i biometrijsko identificiranje korisnika (otisak prsta, dlana i sl.).

Sve zaporke koje se čuvaju na sustavu moraju biti zaštićene od neovlaštenog razotkrivanja, mijenjanja i uklanjanja. To je u prošlosti bio problem kod različitih operacijskih sustava. Da bi se zaporke zaštitilo one ne mogu biti pohranjene u čistom tekstualnom obliku. Ukoliko bi postojala datoteka sa svim zaporkama u takvom zapisu, svaki korisnik bi ju mogao pročitati i otkriti zaporke svih ostalih korisnika.

Rješenje opisanog problema pronađeno je u enkripciji. U svom najjednostavnijem obliku enkripcija je pretvaranje običnog teksta u šifrirani niz znakova. Postoje tri osnovne vrste enkripcije:

- simetrična ili enkripcija jednim ključem,
- asimetrična ili enkripcija s dva ključa i
- sažeci (eng. *hash*) ili enkripcija bez ključa.

Kod simetrične enkripcije koristi se isti ključ i za šifriranje i dešifriranje podataka. Ovakva enkripcija je brza, ali je potreban siguran način razmjene ključa prije početka komunikacije.
Asimetrična enkripcija koristi javni i privatni ključ. Javni ključ je dostupan svima, a sve podatke šifrirane javnim ključem moguće je dešifrirati samo uporabom privatnog ključa. Na taj je način riješen problem razmjene ključa prije komunikacije kod simetrične enkripcije. Nedostatak ovakve enkripcije je sporost.
Funkcije za izračunavanje sažetaka su jednosmjerne funkcije jer provode jednosmjernu transformaciju informacije koja nije povratna. Funkcija za izračunavanje sažetka od dobivenog ulaznog niza znakova radi jedinstveni niz znakova fiksne duljine iz kojeg nije moguće ponovo dobiti početni niz.
Korištenje funkcija za izračunavanje sažetaka nameće se kao najbolji izbor za enkripciju zaporki jer nema potrebe za korištenjem ključeva. Također, zbog nepovratnosti nije moguće dobiti izvornu zaporku iz njezinog sažetka. Kod svake prijave korisnika na sustav računa se sažetak unesene zaporke i uspoređuje ga se sa spremljenim sažetkom. Ukoliko su sažetci isti znači da je i unesena zaporka točna.
Budući da funkcija za izračunavanje sažetka daje jedinstven sažetak za jednaku zaporku, u slučajevima kada postoji više korisnika s jednakim zaporkama, svi će imati i jednak sažetak. Ovo svojstvo se smatra nedostatkom jer se razbijanjem jedne zaporke automatski razbija i druga. Kako bi se to izbjeglo najčešće se prije izračuna sažetka zaporci pridružuje "sol" (eng. *salt*). Sol je slučajan broj koji se dodaje na zaporku prije izračunavanja sažetka te se sprema za svakog korisnika zajedno sa sažetkom. Njenim korištenjem dva korisnika s istim zaporkama neće imati jednake sažetke.

3. passwd i shadow datoteke za pohranu korisničkih imena i zaporki

Kod ranih inačica UNIX operacijskog sustava u tekstualnoj datoteci `/etc/passwd` bila su spremljena korisnička imena i šifrirane lozinke svih korisnika sustava. Tu su datoteku mogli čitati svi korisnici. Osim korisničkog imena i enkriptirane lozinke u `passwd` datoteci su pohranjeni identifikacijski brojevi korisnika i početne grupe u koju korisnik pripada, puno ime korisnika, početni direktorij i ljuska korisnika.
Pojavom programa za razbijanje zaporki i sve većom brigom za sigurnost sustava ovakva konfiguracija je postala veliki sigurnosni rizik. Bilo koji korisnik mogao je iz nje pročitati sažetak zaporke administratora (eng. *root*) te primjenom nekog alata za razbijanje zaporki *brute force* metodom razotkriti njen izvorni oblik.
Rješenje problema pronađeno je u podjeli `passwd` datoteke na dva dijela. Prvi dio i dalje je pohranjen u `passwd` datoteci i sadrži sve podatke osim enkriptiranih zaporki. One su pohranjene u drugoj datoteci s imenom `shadow`. Pristup `shadow` datoteci ima samo administrator računala (eng. *root*). Ukoliko se na sustavu koristi `shadow` datoteka, u `passwd` datoteci na mjestu enkriptirane zaporke nalazit će se samo znak `x`. U `shadow` datoteci osim enkriptirane zaporke i korisničkog imena sadržani su i podaci o minimalnom i maksimalnom trajanju zaporke, broju dana nakon kojih zaporka ističe, broj dana od kad je korisnički račun onesposobljen, broj dana od zadnje promjene zaporke te vremenski period prije isteka zaporke u kojem će se korisnika obavještavati o potrebi njene promijene. Svi zapisi koji na mjestu zaporke imaju znak * označavaju korisnike kojima je onemogućena prijava na sustav.
Slijedi primjer zapisa u uobičajenoj `passwd` datoteci uz korištenje `shadow` datoteke za pohranu sažetaka:

```
marko:x:1017:1013:Marko,,,:/home/marko:/bin/bash
josip:x:1018:1013:Josip,,,:/home/josip:/bin/bash
ivana:x:1019:1013:Ivana,,,:/home/ivana:/bin/zsh
direktor:x:1022:1013:Magda,,,:/home/direktor:/bin/tcsh
```

Odgovarajući zapisi `shadow` datoteke dostupni su u slijedećem pregledu:

```
marko:$1$zTwFrKJJ$UA.Ega8.fg7xyx7CpsiZH1:13474:0:99999:7:::
josip:$1$2Bq6JisY$U0I112VT2AxV8t/w4Gi4p.:13474:0:99999:7:::
ivana:$1$T318/ctw$l//7LTZ/tdoW9rZFTZPiq.:13474:0:99999:7:::
direktor:$1$nst7kQq1$WtB2oX/NMiY3QDpBX4byd0:13474:0:99999:7:::
```

Najčešće korišteni algoritmi za računanje sažetaka na današnjim UNIX i Linux računalima su MD5 i *Blowfish*.

4. PAM mehanizam

PAM (eng. *Pluggable Authentication Module*) je mehanizam koji posreduje između korisničkih programa i jezgre sustava kod autentikacije korisnika i neovisan je o načinu provođenja same autentikacije. Njime se izbjegava potreba za izmjenom programa koji zahtijevaju autentikaciju korisnika zajedno s izmjenom načina autentikacije. Neki od programa koji koriste PAM za autorizaciju korisnika su: ftp, samba, pop3, ssh, http (Apache) i imap poslužitelji, i mnogi drugi.
PAM je 1986. razvila tvrtka Sun Microsystems i trenutno je podržan na slijedećim operacijskim sustavima:

- AIX,
- HP-UX,
- Solaris,
- Linux,
- FreeBSD,
- NetBSD i
- Mac OS X.

Osnovni elementi povezani s radom PAM modula su:

- programi(servisi) koji koriste PAM za autorizaciju korisnika,
- moduli koji provode specifične zadatke prilikom autentikacije,
- konfiguracijske datoteke koje određuju proces autentikacije za svaki podržani servis (svaki servis ima vlastitu konfiguracijsku datoteku koja se nalazi u `/etc/pam.d/` direktoriju, a neki servisi imaju i dodatne konfiguracijske datoteke u direktoriju `/etc/security/`).

Slijedi primjer konfiguracijske datoteke za naredbu `su` (eng. *super user*) koja dodjeljuje administratorske ovlasti različitim korisnicima:

```
auth sufficient /lib/security/pam_rootok.so
auth required /lib/security/pam_wheel.so
auth required /lib/security/pam_pwdb.so
shadow nullok account required /lib/security/pam_pwdb.so
password required /lib/security/pam_pwdb.so
session required /lib/security/pam_pwdb.so
```

U konfiguracijskim datotekama moduli mogu biti pozvani za četiri aktivnosti:

- *auth* – određuje postupak autentikacije,
- *account* – postavlja atribute korisničkog računa,
- *password* – provodi izmjenu zaporke unutar određenog servisa i
- *session* – postavlja i uništava radnu okolinu te logira aktivnost putem *syslog* sustava.

Moduli unutar jedne aktivnosti pozivaju se redom kojim su navedeni te čine stog. Svaki modul u stogu vraća obavijest o uspjehu ili neuspjehu koja u kombinaciji s kontrolnom zastavicom (*requisite*, *required*, *sufficient*, *optional*) utječe na završni rezultat autentikacije na slijedeći način:

- *requisite* – u slučaju uspjeha stog se nastavlja izvršavati, dok u slučaju neuspjeha autentikacija završava neuspješno,
- *required* – stog se nastavlja izvršavati bez obzira na uspjeh ili neuspjeh ovog modula, ali u slučaju neuspjeha autentikacija završava neuspješno,

- *sufficient* – stog se nastavlja izvršavati, a u slučaju uspješnog izvršenja modula autentikacija završava uspješno osim ako je neki od prethodnih *required* modula završio neuspjehom, te
- *optional* – nastavlja se izvršavanje stoga i autentikacija završava neuspješno jedino ako je neki od prethodnih modula završio neuspješno.

Općenito vrijedi da će autentikacija završiti uspješno ukoliko nijedan modul ne zabrani autentikaciju i barem jedan ju dozvoli.

Najčešće korišteni PAM moduli su:

- *pam_unix.so* – provodi autentikaciju korisnika na temelju zapisa u `/etc/passwd` i `/etc/shadow` datotekama,
- *pam_unix2.so* – inačica *pam_unix.so* s dodatnim mogućnostima,
- *pam_securetty.so* – ne dozvoljava prijavu *root* korisnika osim ako je PAM_TTY varijabla radne okoline postavljena na neku od vrijednosti navedenih u `/etc/securetty` datoteci,
- *pam_env.so* – postavlja varijable okoline iz datoteke `/etc/security/pam_env.conf`,
- *pam_mail.so* – provjerava elektronsku poštu u direktoriju navedenom u *dir=* direktorij opciji i izvještava korisnika o rezultatu provjere,
- *pam_lastlog.so* – prikazuje informaciju o zadnjoj prijavi korisnika,
- *pam_deny.so* – uvijek vraća neuspješnu autentikaciju (ovaj modul je koristan u konfiguracijama kod kojih prethodni moduli mogu vratiti dvosmislene rezultate),
- *pam_limits.so* – postavlja ograničenja na korisnikovo zauzimanje procesora, memorije i ostalih resursa prema sadržaju `/etc/security/limits.conf` datoteke,
- *pam_mkhomedir.so* – kreira korisnikov home direktorij ukoliko on ne postoji,
- *pam_pwcheck.so* – provjerava korisnikovu zaporku prema pravilima zadanim u datoteci `/etc/login.defs`, kako bi se poboljšala sigurnost prilikom promjene zaporke,
- *pam_cracklib.so* – provjerava nalazi li se nova zaporka u rječniku i
- *pam_stack.so* – poziva *system-auth* stog i vraća njegov rezultat.

U konfiguracijskim datotekama moguće je postavljati opcije za pojedine module. Neke od najučestalijih su:

- *debug* – zapisuje u log datoteke dodatne informacije za otkrivanje grešaka u radu,
- *try_first_pass* – modul će koristiti zaporku koja je bila predana prethodnom modulu u stogu i ukoliko autentikacija ne uspije s tom zaporkom modul će zatražiti ponovni unos zaporke (ovu opciju moguće je predati većini modula u grupi *auth*),
- *use_first_pass* – ova opcija je slična prethodnoj s razlikom da u slučaju neuspješne autentikacije prethodnom zaporkom modul neće tražiti ponovni unos nego će javiti neuspjelu autentikaciju,
- *nullok* – dozvoljava korištenje prazne zaporke (zaporke koja ne sadrži nijedan znak),
- *shadow* – moduli koji prihvaćaju ovu opciju koriste određene informacije iz `shadow` datoteke (primjerice, rok valjanosti zaporke).

Zahvaljujući velikom broju dostupnih modula, PAM omogućuje veliku fleksibilnost i uvelike olakšava rad sa sustavom upravljanja zaporki na UNIX/Linux sustavima. Također, zbog svoje transparentnosti prema servisima omogućuje izmjene sustava upravljanja zaporki bez potrebe za bilo kakvom intervencijom u konfiguraciji servisa koji ga koriste. Međutim, zbog osjetljivosti samog procesa autentikacije prije svakog mijenjanja konfiguracije PAM-a preporuča se izrada odgovarajućih sigurnosnih kopija te zadržavanje otvorene ljuske *root* korisnika. U slučaju neispravne konfiguracije, ove će mjere osigurati mogućnost ponovnog pokušaja.

5. Načini provjere ranjivosti zaporki

Provjeru ranjivosti zaporki moguće je provoditi prilikom postavljanja zaporke ili nakon što je zaporka postavljena. Korištenjem određenih programa moguće je prilikom postavljanja zaporke ili njenog

mijenjanja provjeriti zadovoljava li određene uvjete (duljina, nije riječ iz rječnika i sl.) te ne dozvoliti njeno postavljanje tako dugo dok ne zadovolji sve tražene uvjete.
Također je važna provjera "jakosti" zaporki nakon što su one postavljene. Takva provjera svodi se na pokušaj njihovog ručnog ili automatiziranog razbijanja. Ručno razbijanje zaporki svodi se na otkrivanje korisničkog imena, sastavljanja liste mogućih zaporki, rangiranja zaporki po vjerojatnosti te provjeravanja svake zaporke. Opisana tehnika je vrlo neučinkovita, pogotovo na sustavima s velikim brojem korisnika.
Kod automatiziranog razbijanja zaporki, postupak uključuje slijedeće korake:

1. otkrivanje važećeg korisničkog imena,
2. otkrivanje korištenog algoritma za enkripciju,
3. pribavljanje enkriptirane zaporke,
4. kreiranje liste mogućih zaporki,
5. enkriptiranje svake moguće zaporke,
6. usporedba sa stvarnim enkriptiranim zaporkama i
7. ponavljanje koraka 1 do 6.

Razbijanje zaporki važno je zbog nekoliko razloga:

- zbog provjere njihove sigurnosti,
- zbog oporavka zaboravljenih zaporki i
- zbog migracije korisnika na drugi sustav.

Razbijanjem zaporki također je moguće pratiti problematične korisnike te ih dodatno educirati o važnosti korištenja sigurnih i jakih zaporki. Nedostaci tehnike razbijanja zaporki su postojanje datoteke s nekriptiranim zaporkama i postojanje osobe koja zna zaporke svih korisnika.
Uz malo kreativnosti moguće je izvršiti provjeru ranjivosti zaporki bez da se razbiju sve zaporke. Primjerice, ako je politikom zaporki utvrđena minimalna duljina zaporki od 8 znakova, moguće je provjeriti pridržavaju li se svi korisnici takve politike pokušajem razbijanja zaporki kraćih od 8 znakova. Ukoliko su se korisnici pridržavali pravila neće biti razbijena ni jedna zaporka.
Postoji nekoliko tehnika razbijanja zaporki:

- pomoću rječnika,
- primjenom sirove sile (*eng. brute force*) i
- mješovita tehnika.

Velik broj korisnika, zbog lakšeg pamćenja, zaporke sastavlja uporabom riječi iz rječnika. Kod razbijanja zaporki pomoću rječnika koristi se popis s velikim brojem riječi koje se koriste za pogađanje korisničkih zaporki. Ova tehnika je daleko brža od iskušavanja svih mogućih kombinacija znakova. Još jedna njena prednost je mogućnost prilagođavanja korištenog rječnika ovisno o okolini u kojoj se obavlja razbijanje zaporki. Primjerice, ako je među korisnicima velik broj ljubitelja znanstvene fantastike, moguće je u rječnik dodati riječi iz tog područja. Velik broj rječnika iz različitih područja i različitih jezika dostupan je na Internetu.
Značajan broj korisnika povezuje izbor dovoljno duge zaporke s nemogućnošću njenog razbijanja. Međutim, danas, neovisno o vrsti zaštite, ne postoji zaporka koju nije moguće razbiti. Kvalitetu zaporki određuje samo količina vremena potrebnog za njihovo kompromitiranje. Uz dovoljno brzo računalo i dovoljno vremena za iskušavanje svih mogućih kombinacija znakova, svaka će zaporka prije ili kasnije biti probijena. Opisana tehnika iskušavanja svih kombinacija znakova poznata je i kao primjena sirove sile (eng. *brute force*).
Kod nje se sve svodi na bitku između brzine računala i vremena potrebnog za kompromitiranje zaporke. Kako s vremenom računala postaju sve jača i brža, tako opada i vrijeme potrebno za razbijanje zaporki koje su nekad bile jake.
Još jedna mogućnost kod primjene sirove sile je korištenje distribuiranog razbijanja tj. korištenja više računala za razbijanje iste zaporke ili skupine zaporki. Na taj se način vrijeme potrebno za razbijanje zaporke bitno smanjuje.
Zaštita od pokušaja kompromitiranja zaporke korištenjem sirove sile uključuje skraćivanje vremena valjanosti zaporke na period kraći od vremena potrebnog za njeno razbijanje.
Prilikom razbijanja zaporki mješovitom metodom kombiniraju se obje dosad opisane metode, odnosno pretpostavljene zaporke se generiraju korištenjem kombinacija riječi iz rječnika i dodatnih znakova. Na taj je način moguće relativno brzo razbiti zaporke koje su kombinacija riječi iz rječnika.

5.1. Unix alati za provjeru ranjivosti (razbijanje) zaporki

Provjeru ranjivosti zaporki na Linux sustavima moguće je provoditi u dva trenutka: prilikom njihovog postavljanja i u trenutku kada su već postavljene.
Za promjenu, odnosno postavljanje lozinke na Linux sustavima koristi se naredba *passwd* koja pomoću PAM (eng. *Pluggable Authentication Module*) modula provjerava da li zaporka zadovoljava postavljene kriterije (minimalna duljina zaporke, nova zaporka nije nastala promjenom malih i velikih slova u staroj zaporci, nije slična staroj, nije nastala rotacijom stare zaporke itd.). Korištenje modula za provjeru ranjivosti zaporke prilikom njenog postavljanja ili promjene je prvi korak u osiguravanju sustava. Unatoč tome, potrebno je obratiti pažnju prilikom konfiguriranja kako se proces promjene zaporki ne bi učinio pretjerano složenim. To je potrebno jer se primjenom pretjeranih pravila provjere korisniku bitno otežava izbor nove zaporke.
Sama provjera prilikom postavljanja zaporki nije dovoljna pa je neophodno sustavno kontrolirati pridržavaju li se korisnici postavljenih pravila. Najbolji način za ovakvu provjeru je pokušaj razbijanja zaporki. Dva danas najčešće korištena alata za tu namjenu su *John the Ripper* i *Crack*.

5.1.1. *John the Ripper*

John the Ripper (ili skraćeno *John*) je alat za razbijanje zaporki koji radi na UNIX i Windows operacijskim sustavima.
Samo korištenje alata je vrlo jednostavno, a u slijedećem ispisu dan je primjer korištenja:

```
john –single /etc/shadow
```

Neke od najčešće korištenih opcija su:

- *single* – razbija lozinke iz samo jedne datoteke (ovo je najjednostavniji način korištenja alata),
- *wordlist:*datoteka – omogućuje korištenje rječnika,
- *rules* – omogućuje upotrebu pravila za izmjenu riječi iz rječnika,
- *restore:*datoteka – nastavlja prekinutu sjednicu,
- *session:*datoteka – omogućuje izbor datoteke u koju će se zapisati podaci o sjednici,
- *show* – pokazuje razbijene zaporke iz zadnje sjednice,
- *test* – provjerava ispravan rad sustava,
- *users:*[-]LOGIN|UID[,...] – razbija zaporke određene skupine korisnika. Ova mogućnost je korisna ukoliko postoji potreba za češćom provjerom zaporki kritičnih korisničkih računa,
- *groups:* [-]GID[,...] – omogućuje provjeru zaporki određenih grupa.

Alat također ima mogućnost obavještavanja vlasnika razbijenih zaporki putem elektronske pošte.
U datoteci `john.conf` koja se u pravilu nalazi u `/etc/john/` direktoriju moguće je definirati dodatna pravila na temelju kojih će se modificirati riječi navedene u korištenom rječniku. Kombinacijom kvalitetnog rječnika i dobrih pravila moguće je vrlo efikasno razbiti velik broj zaporki.

5.1.2. *Crack*

Crack je alat za razbijanje zaporki koga je, također, vrlo jednostavno koristiti. Primjer njegovog pokretanja vidljiv je u slijedećem ispisu:

```
crack /etc/passwd
```

Osim imena datoteke u kojoj se nalaze podaci o korisnicima na sustavu, *Crack* također prima i razne parametre. Najčešće korišteni su:

- *debug* – prikazuje detaljnije informacije o aktivnosti *Crack* skripte (u početku je preporučljivo pokretati *Crack* s ovom opcijom),
- *recover* – omogućuje nastavljanje prekinute sjednice,

- *fgnd* – pokreće program u prednjem planu, pri čemu su standardni ulaz (*stdin*), izlaz (*stdout*) i izlaz za greške (*stderr*) preusmjereni na zaslon kako bi korisnik bolje mogao pratiti rad alata,
- *fmt* – specifikacija formata ulazne datoteke
- *keep* – sprečava brisanje privremene datoteke u kojoj se čuva korisnikov unos,
- *mail* – šalje obavijest elektronskom poštom korisnicima čije zaporke su razbijene,
- *network* – pokreće program u mrežnom modu,
- *nice* – mijenja prioritet izvršavanja programa.

Crack se od alata *John the Ripper* razlikuje po tome što se izvršava u pozadini. Nakon pokretanja programa rezultate njegovog rada moguće je pogledati pomoću programa *Crack-Reporter*.
Najveća prednost *Crack* alata je mogućnost mrežnog rada, odnosno distribuiranje rada na više računala u mreži, čime se znatno skraćuje vrijeme potrebno za razbijanje zaporke.

6. Zaključak

Budući da su UNIX/Linux sustavi od svog nastanka bili višekorisnički te da se vrlo rano javila potreba za identifikacijom korisnika, postupak identifikacije je vrlo razvijen te se uz primjenu odgovarajućih pravila može postići visok stupanj sigurnosti.

Bez obzira na sve alate koje pružaju takvi sustavi, korisnici su i dalje najslabija karika u njihovoj zaštiti. Zbog toga najveću pažnju treba posvetiti upravo edukaciji korisnika, kako bi i oni postali svjesni važnosti sigurnosti i potrebe svog doprinosa njenom ostvarenju.

7. Reference

[1] Eric Cole: Hackers Beware, New Riders Publishing, 2002.
[2] John the Ripper web stranice, http://www.openwall.com/john/, studeni 2006.
[3] Crack download ftp stranica, ftp://ftp.cert.dfn.de/pub/tools/password/Crack/, studeni 2006.
[4] Linux Magazine, Configuring PAM Part One, http://www.linux-mag.com/content/view/2146/0/1/0/, studeni 2006.
[5] Linux Magazine, Configuring PAM Part Two, http://www.linux-mag.com/content/view/2195/0/1/0/, studeni 2006.
[6] Linux Magazine, Securing Linux with PAM, http://www.linux-mag.com/content/view/515/2017/1/0/, studeni 2006.
[7] Linux PAM page, http://www.kernel.org/pub/linux/libs/pam/, studeni 2006.