University of Ljubljana

Univerza v Ljubljani

Fakulteta za Računalništvo in Informatiko

Tržaška 25, Ljubljana

Seminarska naloga pri predmetu "Kriptografija in računalniška varnost"

»RITMIČNA GESLA«

Izola, januar 2005

Primož Kralj

Kazalo

1. Uvod

V seminarski nalogi bom predstavil koncept prijavljanja v sistem, katerega cilj je zagotovitev večje varnosti od običajnega prijavljanja s kombinacijo uporabniško ime, geslo. To je seveda moč doseči na različne načine, naprimer s pomočjo pametnih kartic ali pa biometričnih metod, vendar vse take metode zahtevajo uporabo dodatne strojne opreme. Naša želja je, da večjo varnost zagotovimo z običajno strojno opremo, kar bomo skušali doseči na način, da je vnos gesla občutljiv na različne ritmične kombinacije posameznih znakov.

Seminarska naloga bo razdeljena na več delov. V prvem si bomo pogledali na kakšen način so v operacijskem sistemu Windows hranjena gesla. Poleg tega bomo predstavili na kakšen način in s pomočjo katerih kritpirnih algoritmov poteka običajna prijava v sistem ter katere informacije se pri tem prenašajo preko kanala med odjemalcem in strežnikom.

V drugem delu bomo našteli ter opisali napade, s pomočjo katerih skuša napadalec pridobiti geslo in na ta način vdreti v sistem. Ti napadi za isti cilj uporabljajo zelo različne tehnike oziroma trike, praviloma pa so taki, da napadalec s pomočjo aplikacij preizkuje gesla, dokler ne ugotovi pravega. S pomočjo znanj o različnih vrstah napadov ter splošno znanih priporočil bomo nato razložili kakšna gesla veljajo za dobra gesla in v čem je razlog, da le majhen delež uporabnikov izbira dobra oziroma varna gesla. To dejstvo pa bo služilo za iztočnico tretjega, osrednjega dela seminarske naloge.

V tretjem delu si bomo pogledali na kakšnih principih deluje aplikacija »Ritmična gesla« ter razložili, zakaj je v primeru napadov predlagani sistem varnejši od običajnega. Seveda bomo omenili še preostale prednosti ter pomanjkljivosti v primerjavi z običajno prijavo v sistem.

V zadnjem delu naloge bomo aplikacijo preizkusili na dveh skupinah uporabnikov. Prvo skupino bodo zastopali nekoliko starejši uporabniki, ki računalnik uporabljajo pretežno na delovnem mestu. Druga skupina pa bo zajemala študente, od katerih pričakujemo večjo spretnost pri uporabi aplikacije ter posledično boljše rezultate. Na podlagi rezultatov preizkusa bomo skušali ugotoviti dejansko uporabno vrednost aplikacije.

2. Običajna prijava s kombinacijo uporabniško ime, geslo

V poglavju si bomo natančneje pogledali koncepte, ki so v uporabi pri običajni prijavi v operacijske sisteme Windows.

2.1. Hramba gesel

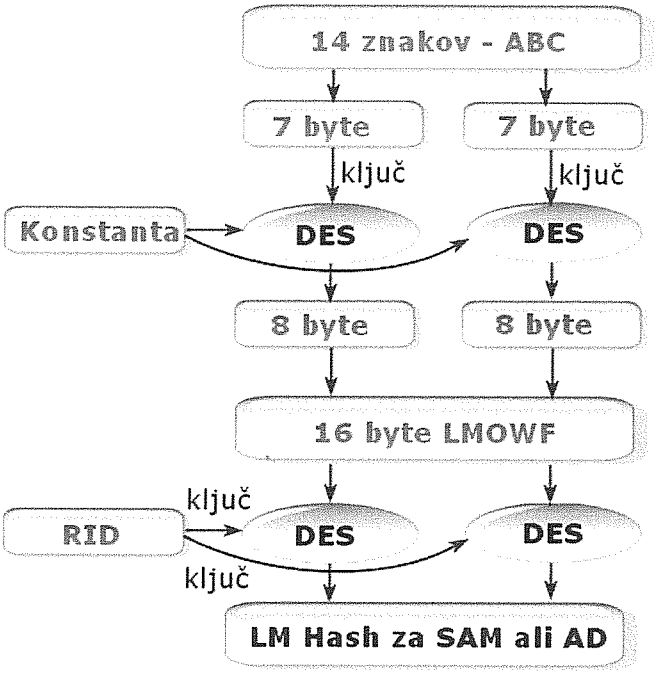
Gesla so v operacijskih sistemih Windows hranjena v podatkovni bazi SAM (Security Accounts Manager) ali pa v aktivnem direktoriju (Active directory). Gesla seveda niso zapisana v čistopisni obliki ampak v t.i. hash obliki. Hash funkcije (zgoščevalne funkcije) h=H(v) imajo v splošnem naslednje lastnosti:

- vhod v je lahko različne dolžine
- izhod funkcije je fiksne dolžine
- H(v) je enostavno izračunati za vsak v
- Iz H(v) mora biti zaradi praktičnih omejitev (omejitve časa in / ali računske moči) nemogoče izračunati h
- V doglednem času (enako kot je veljalo za prejšnjo točko) mora biti nemogoče najti vhoda v in u, za katera velja H(v)=H(u). V nasprotnem primeru smo našli trk (collision).

Bistvo hash funkcije je torej v tem, da iz nje ni mogoče izračunati originalnega sporočila, kar v našem primeru pomeni, da iz zapisa, v katerem je shranjeno geslo ni moč pridobiti gesla v čistopisni obliki.

Operacijski sistemi Windows 2000, XP zaradi združljivosti s prejšnjimi vezijami (Windows 9x) hranijo geslo v dveh oblikah. Starejša oblika se imenuje LM hash, novejša pa NT hash. V naslednjih poglavjih si bomo pogledali, kako iz čistopisa izračunamo vsako izmed njiju.

2.1.1. LM Hash



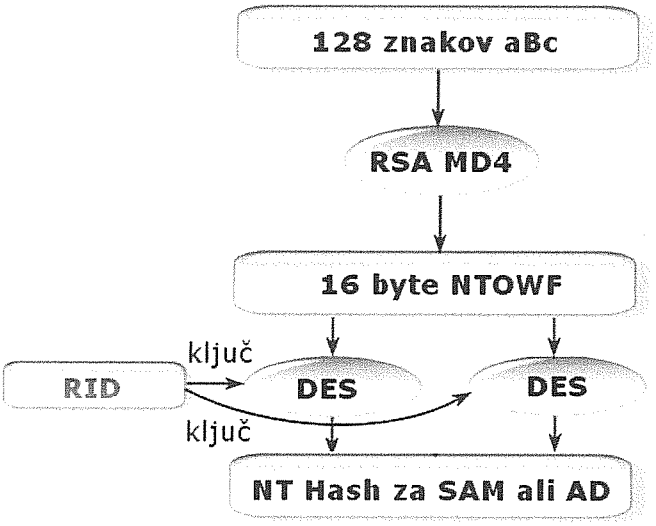
Slika 1: Prikaz postopka za izračun LM Hash

Postopek izračuna LM Hash-a je naslednji:

1. Vnešeno geslo, ki je maksimalne dolžine 14 znakov, se pretvori v velike tiskane črke. Krajšim geslom se do dolžine 14 znakov dodajo ničle.
2. Geslo se razdeli na levo in desno polovico, ki služita kot ključ za DES
3. Z algorimom DES se kriptira javno znana konstanta.
4. Izhod algoritma predstavlja 16 bytni LMOWF (one way function). Tega zaradi potrebe po različnih hash-ih za uporabnika z enakim geslom, še enkrat kriptiramo z DES algoritmom. Tokrat za ključ uporabimo RID (user identifier), ki predstavlja enolično številko uporabnika.
5. Izhod algoritma predstavlja LM Hash, ki ga operacijski sistem hrani v SAM ali AD.

Kot lahko hitro opazimo, ima LM Hash veliko varnostnih pomankljivosti. Pravzaprav je uporaba imena hash v tem primeru napačna, saj gre pri algoritmu DES za simetrično kriptiranje, iz česar sledi, da zahteve za hash funkcije, ki smo jih našteli v točki 2.1. niso zagotovljene. Podrobneje si bomo pomankljivosti LM Hash-a pogledali v poglavju o napadih.

2.1.2. NT Hash



Slika 2: Prikaz postopka za izračun NT Hash

Postopek izračuna NT hash:

1. Vnešeno geslo za izračun NT Hash-a je maksimalne dolžine 128 znakov in je občutljivo na velikost črk. Za izračun LM Hash-a se geslo nad 14 znakov odreže.
2. Za izračun NTOWF uporabimo RSA MD4 zgoščevalno funkcijo. MD4 je bila razvita leta 1990. Dokazano je, da je z običajnim osebnim računalnikom moč najti trke v manj kot minuti. MD4 danes velja za razbito funkcijo.
3. NTOWF enako kot pri izračunu LM Hash-a kriptiramo z algoritmom DES.
4. Dobljeni NT Hash shranimo v SAM ali AD.

3. Napadi na gesla

Napade na gesla delimo na dve večji skupini in sicer na t.i. online ter offline napade. Razlika je v tem, da pri online napadu za pridobitev gesla ne izvajamo dodatnih algoritmov na lokalnem nivoju, ampak je celoten postopek pridobitve gesla vezan na mrežno povezavo.

Pri offline napadu potrebne informacije pridobimo preko omrežja ali pa fizično na računalniku, kjer so shranjena gesla. Na podlagi teh informacij pa se nato na lokalnem nivoju skušamo dokopati do gesel v čistopisni obliki.

3.1. Online napadi

3.1.1. Online poskušanje

Najenostavnejši in hkrati najprimitvnejši napad na geslo je online poskušanje oz. uganjevanje gesla. Tak napad je le redko uspešen. Pogoji, ki nekoliko dvigujejo verjetnost uspeha pa so naslednji:

- Število poskusov vnosa gesla mora biti neomejeno.
- Osebo, katere geslo želimo ugotoviti moramo dobro poznati, saj lahko na ta način testiramo običajna gesla, ki jih izbirajo nevešči uporabniki.

Taka gesla, imenovali jih bomo »slaba gesla«, so največkrat:

- uporabniško ime
- lastno ime, ime družinskega člana, ime hišnega ljubljenčka
- letnica rojstva, datum rojstva, letnica rojstva družinskega člana, letnica pomembnega dogodka za določeno osebo
- enostavne kombinacije sosednjih črk ali številk na tipkovnici: 12345,qwert,55555
- vsako geslo krajše od 4 znakov

Ker je število poskusov avtentikacije na strežnik omejeno, je tak napad v tem primeru obsojen na propad. Možno pa je online poskušanje uporabiti npr. pri spletnih aplikacijah, ftp, pop3, imap ali telnet strežnikih, kjer je število poskusov praviloma neomejeno. V tem primeru si lahko pomagamo z online brute force aplikacijami, ki za nas opravijo težaško delo testiranja različnih kombinacij gesel. Problem je seveda ta, da je online brute force napad v primerjavi z offline napadom izredno počasen, saj testira okoli 500 gesel na sekundo. Iz tega razloga je potreben pogoj za uspeh poznavanje uporabnika, izgradnja lastnega oz. namenskega slovarja za napad (dictionary attack) in vzporeden napad preko več proxy strežnikov. Kot zanimivost navedimo nekatere popularnjše online brute force aplikacije, kot so Brutus, AccessDiver, MooRer's, WWWhack.

Proti online poskušanju se je relativno lahko ubraniti. Potrebno je le to, da ne izberemo predhodno opisanega »slabega gesla«.

3.1.2. Keylog

Najelegantnejši način za pridobitev gesla uporabnika je gotovo keylog. Pri tem načinu gre za uporabo aplikacije, ki jo namestimo na odjemalec »žrtve«, ta pa nam posreduje vse uporabnikove pritiske tipkovnice in s tem tudi samo geslo.

Ker imamo le redko dostop do odjemalca skušamo pogosto žrtev prelisičiti tako, da si aplikacijo namesti kar sama. To storimo s pomočjo t.i. trojanskih konjev. Trojanski konj je vsaka aplikacija, ki za uporabnika izgleda nedolžna (benigna), dejansko pa vsebuje škodljivo kodo. Najenostavnejša in hkrati najprimitvnejša strategija za izdelavo trojanskega konja je

poimenovanje keylog ali podobne aplikacije z imenom, ki uporabniku izgleda nedolžno ali pa zanimivo. Drugi, nekoliko kompleksnješi način je t.i. wrapping, kjer s pomočjo wrapping aplikacije keylog združimo s poljubno nedolžno aplikacijo v eno izvršilno datoteko. Na ta način uporabnik ob zagonu ne bo posumil, da je karkoli narobe, saj se bo ob kliku na izvršitveno datoteko poleg keylog aplikacije dejansko zagnala tudi želena aplikacija.

Trojanskih konjev se lahko branimo na več načinov. V primeru, ko iz interneta downloadamo znane aplikacije lahko njihovo »čistost« preverimo s pomočjo znanega rezultata (checksum) MD5 zgoščevalne funkcije. Zelo priporočljiva je seveda uporaba antivirusnih programov, ki so učinkoviti za razširjene oz. znane trojanske konje. Če anitivirusni program keylog aplikacije ne prepozna, pa je zadnja možnost za preprečitev kraje gesla zaznava trojanskega konja s pomočjo požarnega zidu, ki bo ob pravilni konfiguraciji javil, da želi določen proces komunicirati z zunanjim IP naslovom. Za veščega uporabnika bo to alarm, da je nekaj narobe in bo takemu procesu preprečil dostop, v nasprotnem primeru pa bo geslo, ne glede na njegovo kvaliteto, ukradeno.

3.2. Offline napadi

3.2.1. Pridobitev SAM datoteke

Kot smo že povedali so gesla v hash obliki shranjena v datoteki SAM, katere pa brez pripomočkov ne moremo prebrati, saj je datoteka zaklenjena za branje s strani uporabnika. V primeru, da smo v sistem prijavljeni kot administrator, lahko do SAM datoteke dostopamo s pomočjo aplikacije SAMInside ali SAMdump, ki preko sistemski ukazov prebereta želene podatke.

Le redko se zgodi, da ima napadalec na voljo administratorsko geslo, a tudi brez gesla je moč priti do SAM datoteke. Da to lahko stori, potrebuje napadlec fizični dostop do strežnika, kjer z zagonsko disketo naloži svoj operacijski sistem npr. NTFSDOS PRO, nato pa dostopa do particije, kjer se nahaja datoteka SAM in jo brez težav prekopira. V primeru, da je BIOS strežnika zaklenjen z geslom in je zagon sistema iz prenosnih medijev onemogočen napadalcu preostane le še fizičen poseg v strežnik ter odstranitev baterije iz osnovne plošče s čimer se BIOS geslo izniči in omogoči zagon operacijskega sistema preko prenosnega medija.
Kako se torej zaščitimo pred krajo SAM datoteke? V prvi vrsti je potrebno zelo skrbno varovati administratorsko geslo. Onemogočiti je potrebno fizični dostop do strežnika in zakleniti BIOS z geslom. Poleg tega je zelo pomembno, da se na strežniku nahaja le en operacijski sistem.

Predpostavimo sedaj, da je napadalec našel luknjo v našem varnostnem sistemu ter se dokopal do SAM datoteke. Ker je matematično iz hash oblike nemogoče direktno izračunati čistopis, lahko gesla pridobimo le v obratni smeri, torej s poskušanjem oz. uporabo brute force aplikacij.

Ker imamo gesla v hash obliki shranjena na lokalnem računalniku, je testiranje gesel seveda neprimerno hitrejše kot pri online poskušanju. Orodja, kot so naprimer SAMInside, MDCrack ali LophtCrack, na povprečnem računalniku z 1,8 GHz AMD Athlon procesorjem testirajo okoli 5,5 milijonov gesel na sekundo.

Kot že omenjeno, obstaja več načinov za preiskovanje gesel:
- »čisti« bruteforce napad, kjer zaporedno preiskujemo vse kombinacije gesel
- napad s slovarjem, kjer testiramo gesla shranjena v slovarjih, ki vsebujejo ogromno število potencijalnih gesel
- mask napad, ki ga lahko uporabimo v primeru, ko imamo podatek o delu gesla

Koliko je tak način učinkovit v praksi? V primeru, da v sistemu nismo izklopili shranjevanje gesla v obliki LM Hash, se lahko ob kraji SAM datoteke od gesel vnaprej poslovimo. Ker je LM hash neobčutljiv na velikost črk in se izračuna ločeno za prvih sedem ter drugih sedem znakov (kar je

razvidno iz slike 1), je število vseh možnih kombinacij (skupaj s številkami 0-9), ki jih je potrebno preveriti:

$35^7 * 2 =$ 128.678 milijonov

Zgornja meja razbitja gesla v SAM datoteki je z metodo čitega bruteforce napada pri omenjeni hitrosti (5,5 M gesel / s) okoli 6 ur.

Če želimo torej napadalcu vsaj nekoliko zagreniti življenje je na strežniku nujno potrebno izklopiti LM Hash, za kar lahko najdemo navodila na Microsoftovih spletnih straneh.

Z izklopljenim LM Hashom ter posledično napadom na NT Hash se situacija bistveno spremeni, saj sta sedaj odločilnega pomena kvaliteta samega gesla ter iznajdljivost napadalca.

dolžina gesla	število kombinacij	čas, potreben za razbitje
4 znaki	60^4	2 sekundi
5 znakov	60^5	141 sekund
6 znakov	60^6	141 minut
7 znakov	60^7	141 ur
8 znakov	60^8	352 dni

Tabela 1: Čas potreben za razbitje NT Hash upoštevajoč 60 različnih znakov

Napadalec bo NT Hash najverjetneje napadel z različnimi slovarji, kjer se v praksi izkaže, da bo 90% gesel iz SAM datoteke pridobil v nekaj minutah. Ostala gesla lahko nato skuša pridobiti s čistim bruteforce na omejeni množici znakov (npr. samo velike črke, samo male črke, samo pogoste črke ipd.), s čimer bo razbil enostavnejša daljša gesla. V primeru da ima napadlec dodatne informacije o uporabniku, bo lahko poskušal z več mask napadi.

Kot vidimo iz tabele, smo v primeru uporabe malih in velikih črk, posebnih znakov in številk v geslu pri dolžini gesla 8 znakov ali več skorajda povsem varni pred napadalcem. Problem je seveda v tem, da je v realnem svetu le malo gesel take vrste, saj si tako geslo povprečen uporabnik kar težko zapomni.

3.2.2. Network sniff

Pri metodi network sniff skuša napadalec prestreči podatke, ki si jih izmenjujeta odjemalec in strežnik v postopku prijave v sistem, kar smo prikazali v poglavju 2.2. V primeru, da napadalec prestreže challange strežnika in response odjemalca ima dovolj podatkov, da izvede brute force napad nad posameznim geslom.

Za prestrezanje (vohljanje) v omrežju lahko uporabimo eno izmed številnih aplikacij, ki jih je moč najti na internetu, nato pa uporabimo eno izmed že omenjenih orodij za brute force napad.

Proti vohljanju v omrežju se le stežka zaščitimo. Skrajno priporočljivo je, da v lokalnem omrežju uporabljamo le stikala (switch) in ne hub-ov, proti napadu, ki izvira izven lokalnega omrežja, pa se obranimo le z dobro konfiguriranim požarnim zidom.

3.3. Težave pri izbiri dobrega gesla

Kot smo ugotovili v prejšnjih poglavjih, je poleg splošnih administratorskih priporočil za varnost bistvenega pomena kvaliteta samih gesel. Kakšne lastnosti naj torej ima »dobro geslo«?

- število znakov v geslu naj bo 6 ali več (priporočljivo 8 ali več)
- v geslu naj bodo uporabljeni različni znaki, velike in majhne črke ter številke

4. Ritmična gesla

4.1. Principi delovanja

Osnovna ideja ritmičnega gesla je v tem, da sistem poleg znakovne testira tudi ritmično pravilnost vnosa gesla. Z drugimi besedami, uporabnik mora pravilno geslo vnesti v pravilnih časovnih zamikih med posameznimi znaki gesla. Na ta način je geslo obogateno z dodatno, to je informacijsko vrednostjo ritma vnosa, kar se bi moralo pri smotrni realizaciji že v osnovi kazati v višji stopnji varnosti.

Od sistema zahtevamo naslednje lastnosti:

1. Ker od uporabnika v nasprotju z znakovno ne moremo pričakovati ritmične eksaktnosti, mora sistem omogočati določena odstopanja oz. tolerance pri preverjanju ritmične komponente gesla.
2. Absolutna hitrost vnosa (hitrost takta) ne sme vplivati na pravilnost gesla. To zahtevo bi sicer lahko odpravili, če bi ob vnosu sistem zvočno predvajal ali grafično prikazovali metronom. Prvo možnost lahko takoj izključimo, saj bi na ta način celotna okolica vedela, da uporabnik vnaša geslo, kar močno zniža varnost. Na drugi strani se grafični metronom morda zdi dobra rešitev, vendar pa se običajni uporabnik brez zvočnega impulza nanj zelo težko orientira. Iz teh razlogov od sistema zahtevamo, da mora geslo sprejeti ne glede na to, ali ga uporabnik vnese hitro ali počasi.
3. Sistem mora biti mogoče enostavno aplicirati na obstoječe module za preverjanje gesel s čimer zagotovimo široke možnosti uporabe.

Zahteve zadovoljimo z dokaj enostavnim algoritmom, katerega naloga je, da ritmično geslo pretvori v običajno geslo. To stori tako, da diskretizira časovne periode med posameznimi pritiski tipkovnice v 6 razredov, pri čemer vsak razred predstavlja določen znak. Na ta način zadovoljimo tretjo zahtevo, saj lahko vsi algoritmi in protokoli kateregakoli sistema za prijavo ostanejo nespremenjeni.

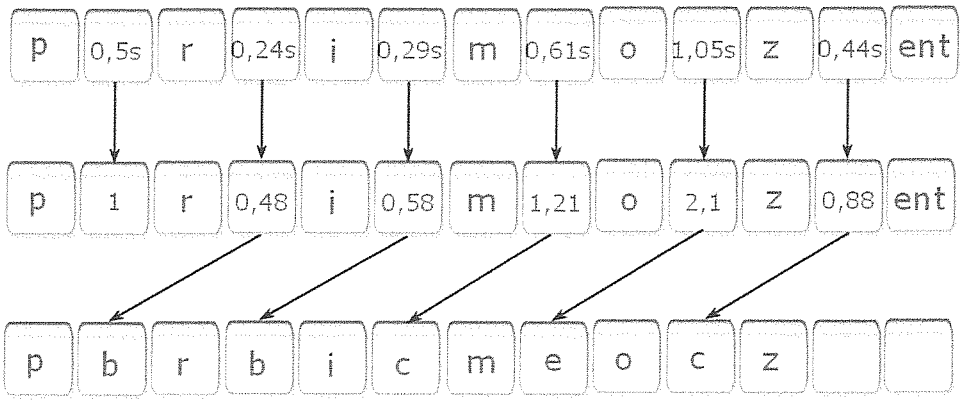
Ker druga zahteva veli, da pravilnost gesla ni odvisna od hitrosti takta, algoritem ne sme diskretizirati absolutnih časovnih period, ampak relativne periode med posameznimi znaki. Na tej točki se nam zastavi dilema, kaj vzeti kot osnovo za primerjavo. Ponuja se nam več možnih rešitev kot so prva perioda, najkrajša perioda ali pa najdaljša perioda. V našem primeru, smo se odločili za prvo periodo, ker predpostavljamo, da ima večina ritmičnih vzorcev v prvi periodi osnovno dolžino (osnovno notno vrednost), to je celinko, polovinko, četrtinko oz. osminko. Za dobro delovanje algoritma je namreč to nujno potrebno, saj bi se v primeru neosnovne periode za primerjavo diskretizacijski razredi postavili na neosnovne notne vrednosti. To pa bi pomenilo, da bi bil sistem mnogo bolj občutljiv na eventuelne ritmične napake (nenatančnosti) uporabnika.

razred	relativna dolžina periode	
	od	do
A		0,25
B	0,25	0,75
C	0,75	1,25
D	1,25	1,75
E	1,75	2,25
F	2,25	

Tabela 2: Razredi diskretizacije po principu prve periode

Ker smo, kot že rečeno za osnovo izbrali prvo periodo, ta vedno pade v isti razred in torej ne prispeva nobene informacijske vrednosti. Iz tega razloga jo lahko izvzamemo iz postopka pretvorbe v določen znak.

Na sliki 4 si lahko ogledamo primer pretvorbe ritmičnega gesla v običajno geslo. Kot vidimo iz slike, je v ritimčnem geslu upoštevana tudi perioda med zadnjim znakom gesla in tipko za potrditev (enter). Rezultat ritimičnega vnosa gesla 'primoz' je običajno geslo 'pbrbicmeocz'.



Slika 4: Primer pretvorbe ritmičnega v običajno geslo po principu prve periode

4.2. Področja uporabe

Če se v prvi fazi omejimo le na računalniško programsko opremo se možnosti uporabe kažejo pravzaprav v vseh sistemih v katere vstopamo s kombinacijo uporabniško ime, geslo. Kot primere naštejmo spletne aplikacije, telnet, ftp, ERP sisteme, oddaljen dostop, prijava na strežnik, prijava v operacijski sistem ipd.

Ker lahko z gotovostjo trdimo, da ritmična gesla niso primerna za celotno populacijo uporabnikov ampak le določen 'ritmično nadarjen' delež, potrebujemo zasnovo, ki bo omogočala, da lahko določeni uporabniki ritmična gesla uporabljajo neodvisno od ostalih (običajnih) uporabnikov istega sistema.

Pri zasnovi aplikacije 'ritmična gesla', ki bo v celoti zadostila omenjeni zahtevi, imamo dve možnosti.

Prva možnost je izdelava običajne Windows aplikacije, ki bi delovala kot neopazen proces v ozadju. Ko bi se uporabnik postavil v katerokoli polje za vnos gesla, bi aplikacija to zaznala, se avtomatično aktivirala in pretvorila vnešeno ritmično geslo v običajno geslo. Seveda pa bi morala aplikacija omogočati tudi ročen zagon ali zaustavitev delovanja. Ker ne želimo, da kdorkoli ugotovi uporabo ritmičnega gesla, je lastnost neopaznosti bistvenega pomena.

V primeru, da bi uporabnik ritmična gesla uporabljal le za prijave preko internetnega brskalnika bi bilo mogoče aplikacijo zasnovati kot priključek oziroma plugin. Prednost takega pristopa je v tem, da ne potrebujemo procesa, ki teče v ozadju operacijskega sistema, žal pa se nam nekoliko skrčijo možnosti uporabe.

Druga možnost, ki je od prve nekoliko manj elegantna je zamenjava modula, ki od uporabnika prebere geslo in ga posreduje nadaljnjim modulom v preveritev. Tega pristopa se moramo poslužiti v primeru, če želimo sistem implementirati za potrebe prijave v operacijski sistem, saj na tej točki še ne moremo zaganjati uporabniških procesov, kot pri prvem pristopu. Konkretno bi bilo za vstop v okolje Windows XP z ritimčnimi gesli potrebno zamenjati modul GINA (Graphical identification and Authentiaction), ki se nahaja v knjižnici »msgina.dll«. Naloga te knjižnice je ta, da uporabniku prikaže grafični vmesnik, pridobi vse potrebne informacije, ter jih posreduje modulu Winlogon, ki poskrbi za preveritev gesla. Ker želimo, da lahko uporabniki izberejo, ali želijo uporabiti ritmično geslo, bi morala naš modul imeti možnost za vnos ritmičnega ali pa običajnega gesla.

Omenimo še, da na Microsofotovih straneh najdemo članek »Essentials of Replacing MSGINA.dll«, ki v podrobnosti opisuje lastnosti in funkcionalnosti modula GINA ter podaja priporočila o zamenjavi standarnega Microsoft modula s prirejenim.

Poleg uporabe v računalniških aplikacijah bi bilo ritmična gesla moč uporabiti še na bančnih avtomatih, sefih, alarmih ipd. Tudi pri taki uporabi pa je nujna možnost aktivacije ritmičnega gesla le na željo posameznega uporabnika.

4.3. Napadi na ritmična gesla

V poglavju o napadih si bomo pogledali varnost ritmičnih gesel v primerjavi z varnostjo običanjih gesel opisanih v poglavju 3.

4.3.1. Online napadi

4.3.1.1. Online poskušanje

Za napad z metodo online poskušanja so bila že običajna gesla dovolj varna, seveda s pripombo, da uporabnik ni smel izbrati t.i. slabega gesla. Ritmično geslo tako bojazen povsem izključi, saj med znake gesla vrine dodatne znake, ki naredijo geslo odporno tako za poskušanje na podlagi informacij o uporabniku, kot tudi za online brute force napade.

Ker pri sistemu ritmičnih gesel ne želimo skriti algoritma delovanja aplikacije, saj se tako skrivanje praviloma izkaže za neučinkovito (security by obscurity), moramo premisliti tudi primer, ko napadalec pridobi informacijo, da uporabnik uporablja ritmična gesla. V tem primeru lahko napadalec sam napiše brute force napad, kjer na sodih mestih testira le 6 različnih znakov. Kot rečeno pri online brute force napadu s hitrostjo 500 gesel na sekundo pride v poštev le napad s slovarjem.

Predpostavimo, da je naše geslo t.i. 'slabo geslo', ki se nahaja v slovarju napadalca, ter da slovar sestoji iz 30.000 gesel. V tem primeru bi napadalec običajno geslo odkril v najslabšem primeru v 60 sekundah. Pri povprečni dolžini gesla v slovarju 6 znakov, pa bi bilo ob posebej zato napisanem algoritmu za napad potrebno testirati: $30.000 * (6\text{-}1)^6$ gesel, kar znese 468.750.000 kombinacij. Za tako razbitje bi napadalec potreboval cca. 11 dni.

Opis napada	Običajno geslo	Ritmično geslo
Ob izbiri 'dobrega gesla'	Varno	Varno
Ob izbiri 'slabega gesla', kjer napadlec nima informacije, da uporabnik uporablja ritmična gesla	Nevarno	Varno
Ob izbiri 'slabega gesla', kjer napadalec ima informacijo, da uporabnik uporablja ritmična gesla	Nevarno	Deloma varno

Tabela 3: Primerjava varnosti pri napadu 'online poskušanje'

Povdariti velja, da je online poskušanje mogoče le nad sistemi, ki dovoljujejo neomejeno število poskusov prijave.

4.3.1.2. Keylog

Če napadalcu na računalnik žrtve uspe namestiti trojanskega konja, napadalec pridobi informacijo o vseh pritiskih tikpovnice s strani uporabnika. V primeru, da uporabnik uporablja ritmična gesla, bo napadalec ob poskusu vnosa gesla pridobljenega preko keyloga presenečeno ugotovil, da geslo ni pravilno.

V tej točki imamo spet dva možna scenarija. Če napadalec ni seznanjen z uporabo ritmičnega gesla je uporabnik varen. V nasprotnem primeru, pa je varnost odvisna od tega, ali sistem dovoljuje neomejeno število poskusov prijave. Če je temu tako, na podlagi informacije o 'znakovni komponenti' ritmičnega gesla veščemu napadalcu ne bo prevelik izziv testirati vse možne 'ritmične' kombinacije (sodi znaki). V nasprotnem primeru pa je uporabnik varen.

Opis napada	Običajno geslo	Ritmično geslo
Napadalec nima informacije, da uporabnik uporablja ritmična gesla	Nevarno	Varno
Napadalec ima informacijo, da uporabnik uporablja ritmična gesla, a je sistem zaščiten z omejenim številom prijav	Nevarno	Varno
Napadalec ima informacijo, da uporabnik uporablja ritmična gesla in sistem ni zaščiten z omejenim številom prijav	Nevarno	Nevarno

Tabela 4: Primerjava varnosti pri napadu 'keylog'

4.3.2. Offline napadi

4.3.2.1. Pridobitev SAM datoteke

Koliko z ritmičnimi gesli pridobimo v primeru, ko napadalec pridobi SAM datoteko? V primeru, da na strežniku LM Hash ni izklopljen ritmična gesla k varnosti ne prispevajo ničesar, saj lahko napadalec kot že rečeno, ločeno obravnava prvi in drugi del gesla ter s pomočjo bruteforce napada tako zelo hitro pride do gesel v čistopisni obliki.

Za primer gesel hranjenih v NT Hash obliki pa je situacija nekoliko drugačna. Z napadom s slovarjem ali mask napadom lahko napadalec pri uporabi običajnih gesel v neznatnem času razbije vsa 'slaba gesla'. Če uporabnik izbere 'slabo' geslo ter hkrati uporablja ritmična gesla pa je varen glede na to, ali napadalec ve za uporabo ritmičnega gesla. Če je temu tako, lahko napadalec kot opisano v poglavju 4.3.1.1. testira vse 'ritmične kombinacije' nad slovarjem, kar pri hitrosti 5,5 milijonov gesel na sekundo ne predstavlja nobene težave. V primeru, da napadalec ne ve za uporabo ritmičnih gesel, je uporabnik kljub izbiri slabega gesla pred napadom s slovarjem ali mask napadom povsem varen.

Pri uporabi 'dobrega gesla' smo pred napadom s slovarjem ali mask napadom varni tako v primeru običajnega kot tudi ritmičnega gesla.

Opis napada	Običajno geslo	Ritmično geslo
Ob izbiri 'dobrega gesla'	Varno	Varno
Ob izbiri 'slabega gesla', kjer napadalec nima informacije, da uporabnik uporablja ritmična gesla	Nevarno	Varno
Ob izbiri 'slabega gesla', kjer napadalec ima informacijo, da uporabnik uporablja ritmična gesla	Nevarno	Nevarno

Tabela 3: Primerjava varnosti pri offline napadu s slovarjem

V primeru čistega brute force napada uporaba ritmičnega gesla podaljša čas, potrebnega za razbitje gesla, saj se običajno geslo dolžine d transformira v geslo dolžine 2*d-1. Pri tem velja omeniti tudi slabost ritmičnega gesla, ki je v tem, da le izredno motorično nadarjen uporabnik lahko v geslu uporabi tako male kot velike črke. Napadalec bo zato ob vedenju uporabe ritmičnega gesla najverjetneje testiral ločeno le velike oziroma male črke, ter tako precej zmanjšal čas potreben za razbitje.

Čase, potrebne za razbitje povprečno dolgega gesla (6 znakov) si poglejmo iz tabele.

geslo	št. Kombinacij	čas razbitja
LM Hash (neglede ali gre za običajno ali ritmično geslo)	35^6	5 minut
NT Hash - običajno geslo (dolžina 6 znakov)	60^6	141 minut
NT Hash - primerljivo ritmično geslo (dolžina 11 znakov) - napadalec ima informacijo, da uporabnik uporablja ritmična gesla	35^6 * 6^5 * 2	60 dni
NT Hash - primerljivo ritmično geslo (dolžina 11 znakov) - napadalec nima informacije, da uporabnik uporablja ritmična gesla	60^11	210.000 let

Tabela 6: Primerjava varnosti pri čistem brute force napadu

Pri offline brute force napadu si varnost zagotovimo z dobrim geslom. Ritmično geslo je v primeru uporabe običajnih aplikacij za brute force napade absolutno varno, v primeru napadov, pisanih posebej za razbitje ritmičnega gesla, pa tega ne moremo trditi.

4.3.2.2. Network sniff

Pri napadu Network sniff veljajo enake zakonitosti kot pri pridobitvi SAM datoteke. Razlika je le v tem, da napadalec v tem primeru skuša razbiti posamezno geslo in ne skupine gesel.

4.3.3. Ostale varnostne prednosti in pomanjkljivosti

Poleg prednosti naštetih v predhodnih poglavjih, lahko omenimo še delno zaščito pred primitivnim napadom, kjer bi napadalec bil fizično prisoten v prostoru ali pa preko skrite kamere skušal uporabniku 'gledati pod prste'. Če bi uporabnik uporabljal ritmična gesla, bi napadalec presenečno ugotovil, da geslo, ki ga je na ta način pridobil, ni pravilno.

Globalno gledano lahko trdimo, da so ritmična gesla zelo varna v primeru, ko napadalec skuša razbiti geslo z običajnimi metodami. Žal to ne velja v primeru uporabe metod, prirejenih posebej za razbitje ritmičnih gesel. Tolažimo se lahko z dejstvom, da je končen produkt, kot že rečeno, zamišljen tako, da ga lahko uporabimo nad vsakim sistemom v katerega se prijavljamo z uporabniškim imenom in geslom. Še več, napadalec na podlagi sistema v katerega bo skušal udreti, ne bo mogel vedeti ali gre pri določenem geslu za uporabo ritmičnega ali običajnega gesla, saj je uporaba ritmičnega gesla odvisna od posameznega uporabnika.

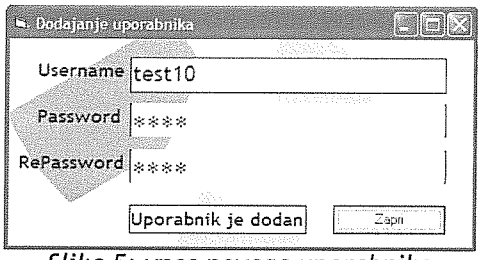
Dejstvo, da bi tako aplikacijo tudi v najboljšem primeru uporabljal le majhen delež vse populacije, ki se prijavlja v najrazličnejše sisteme, nam zagotavlja zelo visoko stopnjo varnosti. Kolikšen delež uporabnikov je pravzaprav primeren oziroma sposoben za uporabo ritmičnih gesel, pa bomo skušali ugotoviti v naslednjem poglavju.

5. Testiranje aplikacije

5.1. O aplikaciji

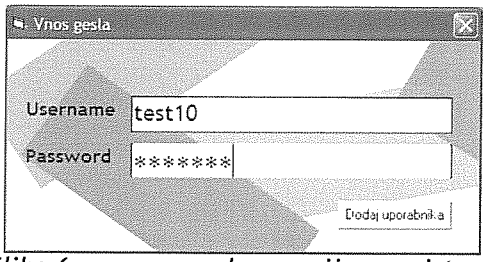
Za potrebe seminarske naloge je bila razvita testna aplikacija s funkcionalnostjo ritmičnih gesel. Aplikacija sicer ni primerna za prijavljanje v realne sisteme, saj ne deluje kot proces v ozadju oziroma 'plugin' internet brskalnika (natančnejši opis v poglavju 4.2), ampak le simulira prijavo v sistem z ritmičnimi gesli.

Prva faza uporabe aplikacije predstavlja vnos uporabniškega imena in gesla s strani uporabnika. Tako kot pri običajnih geslih, se geslo zaradi možnosti napak vnaša dvakrat. V primeru, da je vnos ritmičnega gesla v obeh primerih enak in tako rezultira enako običajno oz. transformirano geslo, aplikacija shrani vnos.

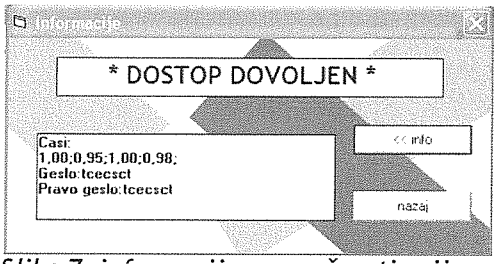


Slika 5: vnos novega uporabnika

V drugi fazi se skuša uporabnik prijaviti v namišljen sistem. V uporabniški vmesnik vpiše svoje uporabniško ime ter ritmično geslo. Aplikacija sporoči ali je bila prijava uspešna, ter po želji izpiše pravilno geslo shranjeno v podatkovni bazi, transformirano geslo pridobljeno iz prijave uporabnika ter relativne čase med posameznimi znaki gesla.



Slika 6: vnosna maska za prijavo v sistem



Slika 7: informacije o uspešnosti prijave

S pomočjo tako zasnovane aplikacije lahko v celoti preverimo sposobnost posameznikov za uporabo ritmičnih gesel.

5.2. Način testiranja

Aplikacijo bomo testirali na dveh skupinah uporabnikov. V prvo skupino bomo uvrstili uporabnike, ki so manj vešči upravljanja z računalnikom. Tu gre predvsem za nekoliko starejše uporabnike, ki računalnik uporabljajo pretežno le v službenem času. Sama znanja o računalništvu seveda na sposobnost uporabe ritmičnih gesel ne vplivajo, vendar pa zaradi manjše suverenosti pri tipkanju ter vprašljivih ritmičnih sposobnosti, od te skupine uporabnikov ne pričakujemo najboljših rezultatov.

V drugi skupini imamo nekoliko mlajše uporabnike, večinoma študentsko populacijo, ki računalnik uporablja za delo in zabavo praktično vse življenje. Od njih pričakujemo mnogo boljše motorične oz. spretnostne sposobnosti, ter posledično boljše rezultate.

Testiranje poteka na naslednji način. Vsakemu uporabniku v grobem razložimo koncept ritmičnih gesel, nato pa ga prosimo, da si izbere ritmično geslo po lastni izbiri. Pri izbiri gesla se mora držati dveh omejitev in sicer minimalne dolžine gesla, ki znaša 4 znake, ter omejitve, da mora biti geslo sestavljeno iz različnih znakov. Vsak uporabnik si lahko vzame nekaj časa za preizkušanje delovanja aplikacije ter lastnih sposobnosti pri kreiranju novih uporabniških imen. Nato se izvede dejanski test.

V prvi fazi skuša uporabnik vnesti novo uporabniško ime, tako da 2x vnese enako ritmično geslo. V primeru, da mu to prej kot v treh poskusih uspe, je v drugi fazi naprošen, da se skuša prijaviti v sistem. Testiranje prijave nato še enkrat ponovi po preteku nekaj ur, saj želimo ugotoviti, v kolikšni meri si uporabniki poleg tekstovnega zapomnejo tudi ritmični del gesla.

Za vsako prijavo ima testiranec tri poskuse, pri čemer beležimo uspešnost vsakega izmed njih.

5.3. Rezultati

zap. št. uporabnika	uspešnost vnosa gesla			uspešnost prijave (nekaj minut)			uspešnost prijave (nekaj ur)			dolžina izbranega gesla
	I.	II.	III.	I.	II.	III.	I.	II.	III.	(št. znakov)
1	X	✓		✓	X	✓	X	X	X	5
2	X	X	X							5
3	✓			✓	✓	✓	✓	X	X	6
4	X	X	✓	X	✓	✓	X	X	X	4
5	X	X	X							6
6	X	X	X							6
7	✓			✓	✓	X	✓	✓	✓	4

Tabela 7: Rezultati uporabe rimičnih gesel - prva skupina

zap. št. uporabnika	uspešnost vnosa gesla			uspešnost prijave (nekaj minut)			uspešnost prijave (nekaj ur)			dolžina izbranega gesla
	I.	II.	III.	I.	II.	III.	I.	II.	III.	(št. znakov)
1	✓			✓	✓	✓	✓	X	✓	4
2	✓			✓	X	X	X	X	X	6
3	✓			✓	✓	✓	✓	✓	✓	5
4	X	✓		X	✓	✓	X	✓	X	6
5	X	X	X							5
6	X	✓		✓	X	✓	X	✓	✓	5
7	X	X	✓	X	✓	X	X	✓	X	6

Tabela 8: Rezulati uporabe ritmičnih gesel - druga skupina

Rezultati, kot jih lahko razberemo iz tabel, so potrdili domnevo, da se bodo uporabniki iz druge skupine izkazali bolje, čeprav razlika ni zelo očitna. Ocena sposobnosti uporabe ritmičnih gesel je naslednja:

	Prva skupina	Druga skupina	skupaj
Število testirancev	7	7	14
Sposobnih za uporabo ritmičnih gesel (vsaj 2/3 prijav uspešnih ob vsakem testiranju)	1	3	4
Nesposobnih za uporabo ritmičnih gesel	6	4	9
Sposobnih %	14%	42%	28%

Opaziti je mogoče tudi kar visoko korelacijo med dolžino izbranega gesla in uspešnostjo prijave. To, sicer lahko razumljivo dejstvo, bi znalo biti problematično v primeru, če bi si uporabniki zaradi olajšanja prijave izbirali prekratka gesla. S tem bi aplikacija izgubila na pomenu, saj varnost glede na običajno prijavo ne bi bila povečana. Razmišljati bi bilo torej potrebno v smeri omejitve najkrajšega možnega gesla, kljub temu da to pomeni zmanjšanje populacije, ki je aplikacijo sposobna uporabljati.

Če pa razmišljamo v smeri povečanja populacije, ki je aplikacijo sposobna uporabljati, bi bilo moč manj veščim uporabnikom podati naslednja priporočila za izbiro ritmičnega gesla:

- uporabite lahko le sosednje tipke, ter si tako olajšate vnos
- uporabite lahko 'flat' ritem, torej ritem, pri katerem so vsi časovni razmaki enako dolgi

Taki olajšavi na varnost sistema bistveno ne vplivajo, seveda če je število znakov v geslu dovolj veliko. Pri izbiri ritmičnega gesla gremo lahko celo v skrajnost, kjer za geslo uporabimo le en znak.

Iz navedenega razmišljanja bi si upal podati oceno, da je aplikacijo sposobno uporabljati med 15% in 20% vseh uporabnikov. Za natančnejšo oceno bi bilo seveda potrebno izvesti obsežnejše testiranje z nekaj deset ali celo sto testiranci.

Posebej zanimivi bi bili rezultati na še mlajši skupini uporabnikov, ki so spretnosti pridobile s pomočjo računalniških iger. Predvidevam, da bi se testiranci izkazali nadpovprečno dobro.

Omeniti je potrebno še en pogled na aplikacijo, ki ga je bilo mogoče opaziti med testiranjem. Visokemu odstotku testirancev, še posebej pa tistim iz druge skupine, predstavlja uspešna uporaba ritmičnih gesel izziv. Opaziti je bilo, da se zgodi neke vrste efekt računalniške igre, torej to, da se uporabniki ob neuspehu znova in znova trudijo, dokler jim s pomočjo vaje ne uspe doseči dovolj visoko stopnjo spretnosti za uporabo ritmičnih gesel. Nekateri testiranci so ob uspeho prešli iz nižjega števila znakov na višje, ali pa na zapletenejše ritme, ter tako višali 'težavnostno stopnjo' oziroma izziv. Ta 'efekt' je gotovo šteti aplikaciji v prid, saj lahko na ta način pri velikem številu uporabnikov preko zabave dosežemo stopnjo spretnosti, ki omogoča zelo visoko varnost.

6. Zaključek

Ideja za ritmična gesla se mi je porodila v začetku leta 2004. Ko sem v tistem času iskal reference na internetu nisem našel nobene strani, ki bi opisovala tako ali podobno rešitev. Iz tega razloga sem bil tudi sam nekoliko skeptičen do ideje, a se mi je vseeno zdela dovolj zanimiva za izdelavo seminarske naloge, katero sem pred razredom predstavil v juniju 2004.

Med pisanjem tega teksta sem še enkrat preiskal internet za določene reference in na svoje presenečenje ugotovil, da je IBM v avgustu leta 2004 patentiral (US 2004/143767) vnos gesla glede na različne časovne razmake. O sistemu je mogoče več prebrati v 2462. izdaji New Scientist Magazina.

Razmišljanje v smeri ritmičnih gesel torej gotovo ni bilo napačno. Morda je bila ideja celo zamujena poslovna priložnost.

7. Viri

- http://support.microsoft.com/kb/299656/en-us/ - »How to prevent Windows from storing a LAN manager hash of your password in Active Directory and local SAM databases«
- http://www.x5.net/faqs/crypto/q94.html - »What is a hash function«
- http://www.insidepro.com/doc/002e.shtml - »Security in Windows 2000/XP«
- http://www.insidepro.com/eng/saminside.shtml - »SAM Inside«
- http://noether.vassar.edu/~myers/help/Passwords.html - »How to choose a bad password«
- http://www.recovery-solutions.com/brute-force.html - »Password recovery using brute force«
- http://www.informit.com/articles/article.asp?p=102181&seqNum=2&rl=1 - »Trojan horses«
- http://c3rb3r.openwall.net/mdcrack/nsindex2.html - »MD Crack«
- http://www.cs.umd.edu/faq/Passwords.shtml - »Choosing a good password«
- http://www.microsoft.com/windows2000/techinfo/administration/security/msgina.asp - »The essentials of replacing MSGina.dll«
- http://www.itsecurity.com/asktecs/jul101.htm - »How are brute force password cracking so successful»
- http://www.winnetmag.com/Article/ArticleID/9186/9186.html - »Cracking user passwords in Windows 2000«
- http://support.microsoft.com/default.aspx?scid=kb;EN-US;q102716 - »User authentication with Windows NT«
- http://www.newscientist.com/article.ns?id=mg18324623.700 - »Get rhythm in your password«