Proiect IC

Honeywords: Making Password-Cracking Detectable

Apostoae Andrei-Alexandru – 341C4

Introducere

Articolul a fost realizat de Ari Juels de la laboratoarele RSA si Ronald Rivest de la MIT (R-ul din CLRS). Acesta propune o metoda de a imbunatati securitatea parolelor hashuite prin stocarea mai multor hashuri de parole false pentru fiecare utilizator. Astfel, un atacator care obtine hashurile parolelor utilizatorilor si care, eventual, reuseste sa inverseze hashurile, nu va sti care din ele este adevarata parola a utilizatorului respectiv. O incercare de a se loga cu una din aceste parole capcana ("honeywords") poate fi astfel captata de un server auxiliar care sa ia o decizie legata de acest incident.

Desi ideea este simpla, aceste parole suplimentare trebuie sa fie greu de distins de parola originala, asa ca articolul propune mai multe metode si face o comparatie intre acestea.

Descriere tehnica

Pentru fiecare user u_i, exista o parola p_i, dar sistemul nu retine aceasta parola in text, ci este retinut hashul acesteia. Articolul nu se concentreaza pe detaliile computationale a obtinerii textului din hashuri, ignorand, de exemplu, salturile. Pentru ca sistemul sa fie cat mai sigur, pe langa serverul care se ocupa cu autentificarea, mai exista un server securizat, numit si "honeychecker" sau "login monitor".

Considerand lungimea honeypotului pentru fiecare user k, in honeychecker vor fi stocate numere intregi intre 1 si k pentru fiecare user, corespunzator indexului parolei corecte din honeypot.

Astfel, pentru un user i care are parola '*apo123*' si honeypotul generat este ['*apo345*', '*apo234*', '*apo123*', '*apo989*'], in honeychecker se va stoca valoarea 3. Sistemul de gestiune a loginului comunica cu honeycheckerul prin doua comenzi:

- Set: i, j – seteaza c(i) = j
- Check: i, j – verifica daca c(i) == j

Set are loc la inregistrare sau modificarea parolei, cand o noua lista de honeywords este construita, iar check in momentul loginului.

Din punct de vedere a terminologiei, remarcam:

- sugarword = parola corecta
- honeywords, chaff, decoy = celelalte k-1 cuvinte
- sweetword ("tough nut") = o parola foarte puternica a carui hash sa nu poate fi inversat de adversar

Se ajunge, practic, ca baza de date in care sunt stocate hashurile sa fie de aproximativ k ori mai mare. Autorii propun k = 20 si mentioneaza faptul ca, in general, stocarea parolelor nu trebuie sa fie o problema pentru ca deja multe servicii online stocheaza parolele vechi.

Din punct de vedere al securitatii, exista 3 posibilitati pentru un adversar care reuseste sa inverseze hashurile:

1. castiga jocul, ghicind c(i)
2. ghiceste gresit si este prins, iar administratorii pot lua o decizie
3. refuza sa joace

Autorii introduc un concept nou, numit “flatness”, care descriu capacitatea de a genera honeyworduri cu probabilitate cat mai constanta (1/k) a unui algoritm. Desi o metoda de generare flat este ideala, autorii sustin ca si o metoda imperfecta poate fi foarte eficienta in combaterea adversarului.

Metode de generare a honeywordurilor

Autorii propun cateva metode de generare a honeywordurilor cat mai flat. Ei le impart in doua categorii:

- cu metode in care UI-ul de login ramane neschimbat (de dorit)
  - chaffing-by-tail-tweaking
  - chaffing-by-tweaking-digits
  - chaffing-with-a-password-model
- cu metode care modifica UI-ul de login
  - take-a-tail
- cu metode hibrid

Chaffing-by-tweaking

Sunt alese t pozitii ce vor fi schimbate din parola si valorile sunt randomizate.

Exemplu tail-tweaking: [*BG+7q03*, *BG+7m55*, *BG+7y45*, *BG+7o92*].
Exemplu tweaking-digits: [*42*flavors*, *57*flavors*, *18*flavors*]

Caracterele noi trebuie sa fie de acelasi tip ca cele inlocuite, astfel incat atacatorul sa nu isi poate da seama cu usurinta. Metoda este flat daca numerele alese de utilizator sunt total random, dar de obicei sunt alese numere cu semnificatie pentru user, sau sunt preferate numere de o cifra.

Chaffing-with-a-password-model

Este folosit un set de date ca generator de viitoare honeyworduri, astfel incat parola initiala a userului nu este importanta in acest proces. Pentru confuzia adversarului, se introduc si tough-nuts. Un exemplu de honeyworduri generate este: [*kebrton1*, *02123dia*, *1erapc*, *9,50PEe]KV.0?RIOtc&L-:IJ"b+Wol*, *’Sb123*].

Este propusa si o imbunatatire ce presupune in respectarea structurii parolei initiale, de exemplu pentru parola *mice3blind* (cuvant de 4 litere, urmat de o cifra, urmat de cuvant de 5 litere) se vor genera parole de aceeasi forma (ex. *gold5rings*). Acest lucru presupune exista unor dictionare, iar utilizatorul sa isi fi ales o parola ce contine cuvinte.

Take-a-tail

Utilizatorul este fortat la inregistrare sa propuna o parola, dupa care sa adauge la sfarsit un numar random care va fi noua parola. Astfel, este foarte usor sa se genereze honeywords pe baza acelei cozi random, dar intervine o incomoditate pentru utilizator in a retine numarul original.

Hybrid

1. se utilizeaza chaffing-with-a-password-model pentru a genera a sweetworduri
2. se utilizeaza chaffing-by-tweaking-digits pentru a generat b tweakuri, rezultand in a*b sweetwords
3. se permuta setul de sweetwords

Ca factori in evaluarea acestor metode sunt considerati: flatness, rezistenta la denial of service, costul stocarii si protectie pe sisteme multiple.

Honeyword method	**Flatness** (§3)	**DoS resistance** (§7.5)	**Storage cost (# of hashes)** (§5.4)	**Legacy -UI?** (§4)	**Multiple-system protection** (§7.6)
Tweaking (§4.1.1)	$(1/k)$ if U constant over $T(p)$	weak	1	yes	no
Password-model (§4.1.2)	$(1/k)$ if $U \approx G$	strong	k	yes	no
*Tough nuts** (§4.1.3)	N/A	strong	k	yes	no
Take-a-tail (§4.2)	$(1/k)$ unconditionally	weak	k	no	yes
Hybrid (§5.5)	$(1/k)$ if $U \approx G$ and U constant over $T(p)$	strong	$\sqrt{k}$	yes	no

Implementare

Am ales sa implementez doua metode: take-a-tail si chaffing-with-a-password-model. Pentru simplitatea demonstratiei, am hostat un web server cu posibilitate de login si inregistrare pe aceeasi masina cu honeypotul.

Structura proiect:

- webserver.py – primeste ca argument in linia de comanda ‘take-a-tail’ sau ‘chaffing’ si porneste un server web care suport login, register si afiseaza mesaje relevante
- attacker.py – primeste ca argument numele unui utilizator si o metoda si afiseaza posibilele parole dupa inversarea hashurilor si ‘?’ pentru tough-nuts
- generate_honeywords.py – modul auxiliar pentru generarea de honeyworduri prin ambele metode
- users_tail.db – baza de date serializata folosita de take-a-tail
- users_chaffing.db – baza de date serializata folosinda de password-chaffing
- rockyou_10000.txt – contine top 10000 de parole folosite de utilizatorii RockYou

Take-a-tail

La inregistare, se genereaza aleator o coada de 3 cifre pentru parola aleasa de utilizator pe care acesta trebuie sa o rescrie si memoreze. Se genereaza apoi inca 4 parole cu coada diferita, iar ca imbunatatire adusa de mine, parolele noi vor diferi prin cel putin 2 caractere, astfel incat sa nu se ajunga la typouri, situatie in care un utilizator ar putea fi catalogat gresit ca atacator.

Adversarul are astfel de ales intre 5 variante, neavand cum sa le deosebeasca intre ele, deci are practic 20% sanse de a ghici.

Chaffing-with-a-password-model

Sunt generate parole precum este descris in Appendix. Tough-nuts se genereaza cu o probabilitate de 8%. Se alege un cuvant random din datasetul de parole si se retine lungimea acestuia. Se salveaza primul caracter intr-o noua parola. Restul se imparte in:

- 10% sansa ca ultimul cuvant sa fie inlocuit cu altul din dataset de aceeasi lungime si sa se salveze in noua parola caracterul de la pozitia curenta
- 40% sansa ca ultimul cuvant sa fie inlocuit cu altul din dataset cu aceeasi lungime si cu acelasi caracter pe pozitia precedenta ca parola noua pe care o construim
- 50% sansa sa nu schimbam cuvantul si sa copiem caracterul de pe pozitia la care am ajuns

Analiza personala si concluzii

Metodele prezentate in acest articol, impreuna cu analiza acestora ofera un sprijin extraordinar serviciilor care vor un strat aditional de securitate. Costurile unor astfel de implementari sunt mici, iar posibilitatea de a opri adversari este foarte mare.

Dificultatile apar in implementarea unui generator flat de parole. Un mare dezavantaj pe care l-am observat la password-chaffing este ca daca folosesc o parola in limba romana, parolele generate o sa arate destul de asemanator cu limba engleza, ceea ce ar putea scoate in evidenta parola adevarata si sistemul nu ar fi util. Daca parola este, in schimb, putin mai abstracta si in engleza, este foarte greu de distins.

Take-a-tail, desi foarte simplu si robust, presupune ca utilizatorul sa memoreze un numar aleator in plus, ceea ce poate fi un factor incomod avand in vedere ca, de foarte multe ori, utilizatorii folosesc aceeasi parola pentru mai multe siteuri, si astfel ar uita parola si fi nevoiti sa schimbe mult mai des decat in mod normal. Bineinteles, acest lucru este benefic din punct de vedere al securitatii, dar incomoditatea adusa nu justifica aceasta metoda.