

HoneyPot Based Password Authentication System

Name: Nischal A

Roll Number: 1801cs33

CONTENTS

1) Abstract/ Highlights of the Work.

2) System Usage and Working

3) Implementation Details

- Generation of Supporting Files

- C code description

 - Mainserver.c

 - Honeychecker.c

4) Interesting Observations

5) Conclusion.

ABSTRACT

In this assignment, I implement the Eugler's based Honey-Pots method for password authentication. The problem of securing user accounts against password attacks aimed at accessing the users' details for malicious purposes has been on rise since the password authentication system has been introduced. The importance is more so in the present day when each user has many accounts in different websites which deal with critical informations like social media, bank accounts etc. In CS392 we are taught about Eugler's honeypot based authentication system which proves as an effective means to tackle such attacks. In this assignment, I have implemented the same system and included two major tasks, i.e. user registration process and the login process. The implementation is exactly as proposed in the original paper. Each user will have an entry in the F1 file containing a set of potential indices out of which only one is correct (the sugarindex). F2 file contains the hash value of each of the indices which needs to be cracked using a password tool like John the Ripper for further processing. I have included various supporting files along with F1 and F2 and provide three major C codes mainserver.c, supportserver.c and honeychecker.c for implementing this system. I have also provided snapshots of the output from the terminal wherever necessary.

System Usage and Working

General Flow:

- a) From Attacker's Point of View
 1. Attacker obtains access to F1.txt and F2.txt files
 2. Attacker runs John the ripper on F2 to obtain the raw-text passwords
 3. Attacker tries logging in using the login portal and tries different passwords (out of the K (or <K) cracked passwords) for a target user.
- b) From Administrator's point of view
 1. The admin controls the mainserver and the honeychecker
 2. Admin is a trusted person and he uses John-The-Ripper on F2 and advises the users with weak passwords to update it.
 3. Admin also keeps checking the honeyserverlog.txt file which notifies in case of security breach i.e. in case a wrong user's password or honeypot's password is used for a given user. The IP address of the attacker will be noted. The real user will get a prompt to this contact number checking whether he is actually trying to login.
- c) From normal user's point of view:
 1. Any normal user can use this system for either: -
 1. Registering as a new user: In this case a new username and password has to be provided.
 2. Login for existing user: The user can use his/her credentials for logging in.

Detailed Explanation of Each Step with Snapshots.

Setting up the System: Makefile

provided for direct compiling.

- 1) For compiling mainserver.c please do "make main"
- 2) For compiling honeyserver.c please do "make honey"

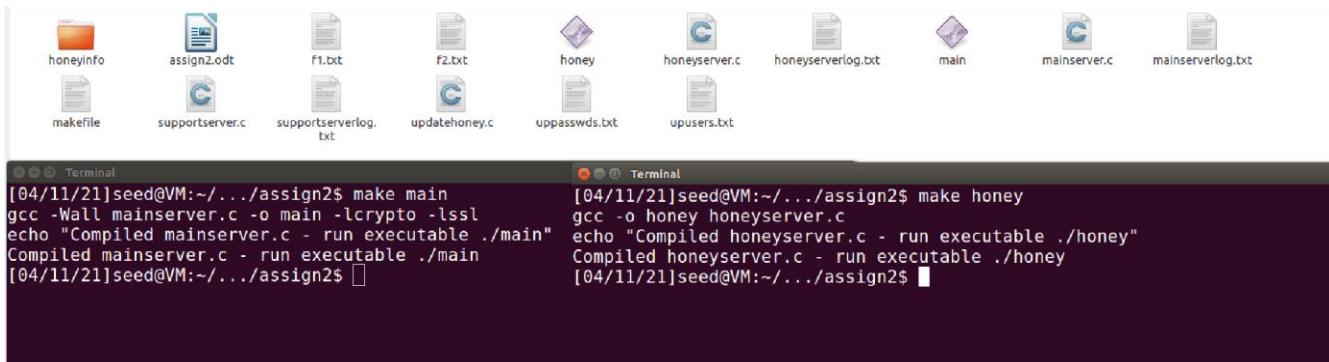


Fig1: Using makefile

Note: PLEASE RUN (mainserver and honeyserver) in TWO DIFFERENT TERMINALS.

I have used TCP (Transfer Control Protocol) for establishing connection between the two. (More details on this in the next section.)

Compiling will give executables “main” for mainserver.c and “honey” for honeyserver.c

For mainserver.c - ./main

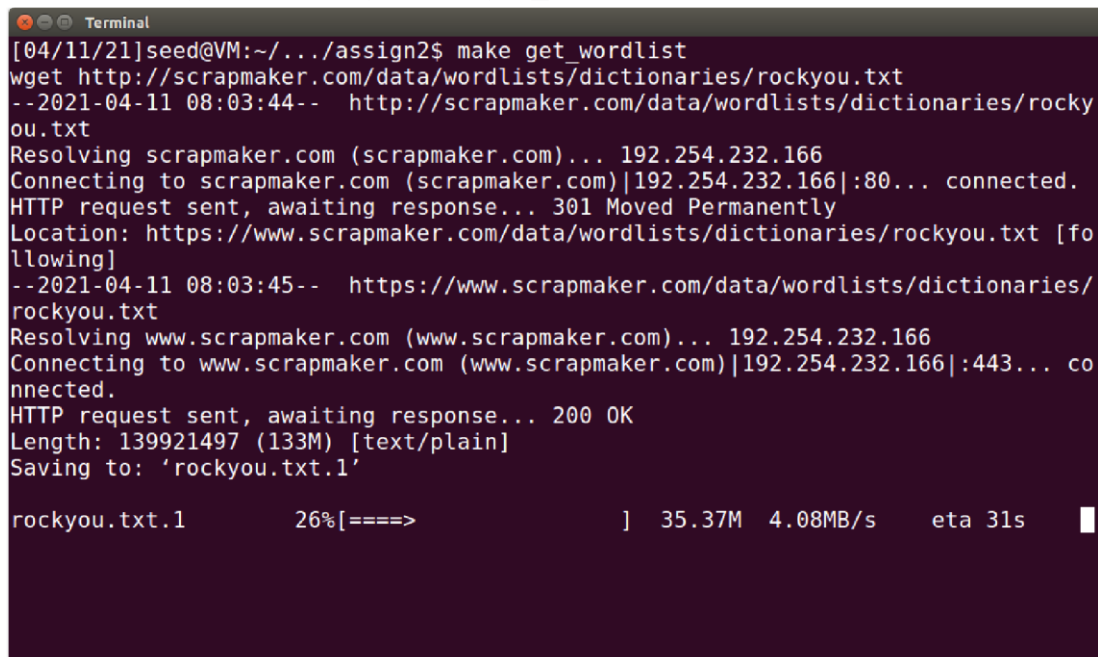
For honeyserver.c - ./honey

JOHN THE RIPPER FOR THE ADMIN

For running John the Ripper (Commands provided in makefile)

Please note due to size constraints rockyou.txt isn't provided. Please follow the instructions to download it.

1. Download rockyou.txt using **make get_wordlist**



```
Terminal
[04/11/21]seed@VM:~/.../assign2$ make get_wordlist
wget http://scrapmaker.com/data/wordlists/dictionaries/rockyou.txt
--2021-04-11 08:03:44-- http://scrapmaker.com/data/wordlists/dictionaries/rockyou.txt
Resolving scrapmaker.com (scrapmaker.com)... 192.254.232.166
Connecting to scrapmaker.com (scrapmaker.com)|192.254.232.166|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://www.scrapmaker.com/data/wordlists/dictionaries/rockyou.txt [following]
--2021-04-11 08:03:45-- https://www.scrapmaker.com/data/wordlists/dictionaries/rockyou.txt
Resolving www.scrapmaker.com (www.scrapmaker.com)... 192.254.232.166
Connecting to www.scrapmaker.com (www.scrapmaker.com)|192.254.232.166|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 139921497 (133M) [text/plain]
Saving to: 'rockyou.txt.1'

rockyou.txt.1      26%[====>                ] 35.37M  4.08MB/s   eta 31s
```

2. **make john_run** (For running the password cracking tool on F2.txt for administrator to monitor weak passwords.)

3. **make john_show** (For showing the cracked passwords).

Now, we will simulate it as follows: -

```
root@VM: /home/seed/Desktop/ssd/assgn2/assign2
root@VM: /home/seed/Desktop/ssd/assgn2/assign2# make john_show
$JOHN --show --format=raw-md5 f2.txt
1:alice123
2:iloveyou999
3:billgates
5:ihateyou
6:rockandroll
7:basketball
8:princess
11:sachin123
12:myspace1
13:superman
14:chocolate
```

Very weak passwords will be reported to users by the Administrator.

CASE 1: Normal User trying to Login

Subcase A: A good user already knows his username and password. (Currently I have 77 users registered in my system.)

```
Terminal
[04/11/21]seed@VM:~/.../assign2$ ./main
Login Module Setup
Please enter choice (1/2)
1. Register as new user
2. Login as existing user
2
Please enter username and Password
Username: jill
Password: burgerstake
Correct password. Login Successfull
[04/11/21]seed@VM:~/.../assign2$

Terminal
[04/11/21]seed@VM:~/.../assign2$ ./honey
[+]Server socket created successfully.
[+]Binding successfull.
[+]Listening...
Output only visible to the ADMINISTRATOR
Mainserver initiates request
Honeyserver accepts request and start
Recieved Value of usernum 65 and indexnum 26
Sending computed result back
[04/11/21]seed@VM:~/.../assign2$
```

Please run ./main and ./honey in separate terminals.

In main – Chose option 2 (Login as existing user)

Here, username is jill and Password: burgerstake.

The password in this case is correct and hence login successful!

```
Terminal
[04/11/21]seed@VM:~/.../assign2$ ./main
Login Module Setup
Please enter choice (1/2)
1. Register as new user
2. Login as exisiting user
2
Please enter username and Password
Username: trump
Password: idontknow
Entered user not registered
[04/11/21]seed@VM:~/.../assign2$ █

Terminal
[04/11/21]seed@VM:~/.../assign2$ ./honey
[+]Server socket created successfully.
[+]Binding successfull.
[+]Listening...
Output only visible to the ADMINISTRATOR
Mainserver initiates request
Honeyserver accepts request and start
Recieved Value of usernum -1 and indexnum -1
Main program handled the situation.
[04/11/21]seed@VM:~/.../assign2$ █
```

Subcase B: Entered username is wrong. trump is not registered in this

system. Hence error message is detected.

Subcase C: Entered Password is wrong (not a honeypot just some random wrong password).

```
Terminal
[04/11/21]seed@VM:~/.../assign2$ ./main
Login Module Setup
Please enter choice (1/2)
1. Register as new user
2. Login as exisiting user
2
Please enter username and Password
Username: varun
Password: idontknow
Entered password is wrong
[04/11/21]seed@VM:~/.../assign2$ █

Terminal
[04/11/21]seed@VM:~/.../assign2$ ./honey
[+]Server socket created successfully.
[+]Binding successfull.
[+]Listening...
Output only visible to the ADMINISTRATOR
Mainserver initiates request
Honeyserver accepts request and start
Recieved Value of usernum -1 and indexnum -1
Main program handled the situation.
[04/11/21]seed@VM:~/.../assign2$ █
```

Appropriate error message is displayed.

CASE 2: Attacker Trying to Login (Hack)

Step 1: Obtaining the F1 and F2 files. We assume that the attacker has his hands on F1 and F2 and now is trying to use the information from there for logging in.

Step 2: Obtaining the passwords from F2 file. For obtaining the passwords attacker uses John The Ripper on F2 file.

```
root@VM: /home/seed/Desktop/ssd/assgn2/honeysimulate
[04/10/21]seed@VM:~/.../honeysimulate$ su
Password:
root@VM:/home/seed/Desktop/ssd/assgn2/honeysimulate# export JOHN="/home/seed/source/john/run/john"
root@VM:/home/seed/Desktop/ssd/assgn2/honeysimulate# time $JOHN --wordlist=/home/seed/Desktop/ssd/assgn2/test/rockyou.txt --format=raw-md5 f2.txt
Using default input encoding: UTF-8
Loaded 76 password hashes with no different salts (Raw-MD5 [MD5 256/256 AVX2 8x3])
Remaining 61 password hashes with no different salts
Press 'q' or Ctrl-C to abort, almost any other key for status
jessica          (52)
michelle         (51)
jennifer         (30)
112233           (49)
555555           (56)
pepper          (53)
zxcvbnm          (31)
freedom          (59)
131313           (58)
zxcvbn           (55)
11111111         (54)
trustno1         (28)
77777777         (60)
```

Attacker can run the appropriate command (here format is set to raw-md5 and wordlist to rockyou.txt).

Now, attacker can see these in a more organized manner using the show command.

```
root@VM: /home/seed/Desktop/ssd/assgn2/honeysimulate
root@VM:/home/seed/Desktop/ssd/assgn2/honeysimulate# $JOHN --show --format=raw-md5 f2.txt
1:alice123
2:iloveyou999
3:billgates
5:ihateyou
6:rockandroll
7:basketball
8:princess
11:sachin123
12:myspace1
13:superman
14:chocolate
15:alexander
16:samantha
17:godblessamerica
20:cooper123
27:killer123
28:trustno1
29:jordan123
30:jennifer
31:zxcvbnm
32:asdfgh123
33:hunter123
```

Step 3: Attacker targets a user account and tries the respective passwords of the K indices.

Out of the 77 entries let us say Attacker choses account of “nobita”

1	Richard:72,5,19,54,4,59,65,64
2	Christopher:6,17,21,10,28,14
3	nobita:16,1,24,18,7,13,11,6
4	Joshua:6,16,21,28,33,29
5	Brittany:70,5,19,67,4,43,69,15
6	Amanda:6,13,21,33,34,14
7	Crystal:76,5,19,60,4,64,42,46
8	Jeffrey:74,5,19,60,4,11,13,15

Now using the show command from the cracked passwords of F2 file, he tries to retrieve the text passwords of all indices – 16, 1, 24, 18, 7, 13, 11 and 6 (Note: Here K is 8 – I have made K value dynamic to vary from 6 to 9 randomly for each user registration.) 16:samantha

1:alice123

24:<not able to crack>

18:<not able to crack>

7:basketball

13:superman

11:sachin123

6:rockandroll

Now, since, 24 and 18 are not crackable by John, attacker chose any of the other passwords.

Let's say attacker tries number 13:superman

```
Terminal
[04/11/21]seed@VM:~/.../assign2$ ./main
Login Module Setup
Please enter choice (1/2)
1. Register as new user
2. Login as exisiting user
2
Please enter username and Password
Username: nobita
Password: superman
Login Attempt Unsuccessful!
This incident will be reported
[04/11/21]seed@VM:~/.../assign2$
```

```
Terminal
[04/11/21]seed@VM:~/.../assign2$ ./honey
[+]Server socket created successfully.
[+]Binding successfull.
[+]Listening....
Output only visible to the ADMINISTRATOR
Mainserver initiates request
Honeyserver accepts request and start
Recieved Value of usernum 3 and indexnum 13
Sending computed result back
[04/11/21]seed@VM:~/.../assign2$
```

Login Attempt is unsuccessful!

Honeyserver logs this event in its file honeyserverlog.c

```
honeyserverlog.txt (~/Desktop/ssd/assgn2/assign2) - gedit
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
BREACH (HONEYPOT ACCOUNT TYPE DETECTED). NOTING DOWN IP.
Plain Text Tab Width: 8 Ln 161, Col 1 INS
```

The index 13:superman happened to be that of a honeypot. Hence, incident is reported!

Please note, here the correct index for nobita is 24 and the actual password is thesunshallrise! - the password is purposely chosen hard so that John cannot crack it! Next I will demonstrate a case when the password is weak and attack happens. Now, we chose username "alice":2,8,6,17,1,12


```

61  alice:2,8,6,17,1,12
62  Jonathan:48,31,33,26,2,13,14,9
63  Kevin:60,59,31,26,2,55,10,9
64  rajesh:7,4,9,18,17,15
65  jill:16,11,1,24,7,26
66  Matthew:6,15,21,13,30,17
67  raja:15,16,6,3,4,8
68  Anthony:53,14,46,26,2,33,52,9
69  chang:2,1,6,12,9,14
70  penny:20,21,13,19,6,14
71  aravind:77,1,61,3,42,60,22,20
72  milky:6,4,7,14,12,13
73  aaron:8,3,1,10,7,11
74  Robert:6,10,21,33,38,32
75  parker:14,12,13,6,8,7
76  gates:8,14,6,3,5,12
77  bay:9,2,17,3,16,11

```

2:iloveyou999

8:princess

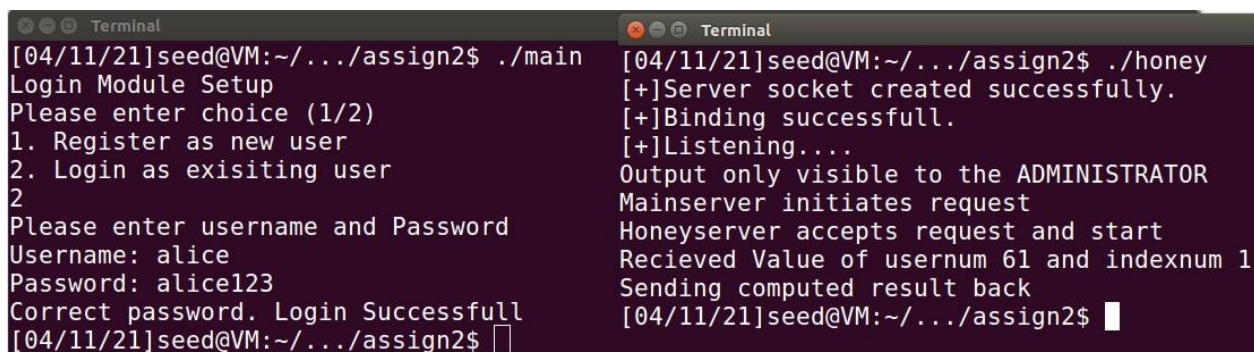
6:rockandroll

17:godblessamerica

1:alice123

12:myspace1

Out of all the cracked passwords for alice – alice123 looks like the most likely one to the attacker.



```

[04/11/21]seed@VM:~/.../assign2$ ./main
Login Module Setup
Please enter choice (1/2)
1. Register as new user
2. Login as exisiting user
2
Please enter username and Password
Username: alice
Password: alice123
Correct password. Login Successfull
[04/11/21]seed@VM:~/.../assign2$

[04/11/21]seed@VM:~/.../assign2$ ./honey
[+]Server socket created successfully.
[+]Binding successfull.
[+]Listening....
Output only visible to the ADMINISTRATOR
Mainserver initiates request
Honeyserver accepts request and start
Recieved Value of usernum 61 and indexnum 1
Sending computed result back
[04/11/21]seed@VM:~/.../assign2$

```

Unfortunately, hacker has gained access to Alice's account. This is because Alice's password is very weak and it contains her name in it. **Hence, the administrator should have earlier warned her, to change her password. Such type of problems must be handled by the administrator before an attacker can exploit it.**

CASE 3: Normal User trying to register

Note: Initially there are 77 accounts (normal users + honeypots included).

Now let us see what happens if a new user registers.

Terminal Output

```
[04/11/21]seed@VM:~/.../assign2$ ./main
Login Module Setup
Please enter choice (1/2)
1. Register as new user
2. Login as existing user
1
Please enter username and Password
Username: norman
Password: sweethome
[04/11/21]seed@VM:~/.../assign2$
```

```
[04/11/21]seed@VM:~/.../assign2$ ./honey
[+]Server socket created successfully.
[+]Binding successfull.
[+]Listening...
Output only visible to the ADMINISTRATOR
Mainserver initiates request
Honeyserver accepts request and start
Recieved Value of usernum -1 and indexnum -1
Main program handled the situation.
[04/11/21]seed@VM:~/.../assign2$
```

F2 file will get a new entry corresponding to 78

```
49 49:d0970714757783e6cf17b26fb8e2298f
50 50:c630878c9717e8fab0177ff81024b8a
51 51:2345f10bb948c5665ef91f6773b3e455
52 52:aae039d6aa239cfc121357a825210fa3
53 53:b3f952d5d9adea6f63bee9d4c6fceeaa
54 54:1bbd886460827015e5d605ed44252251
55 55:b427ebd39c845eb5417b7f7aaf1f9724
56 56:5b1b68a9abf4d2cd155c81a9225fd158
57 57:1bbd886460827015e5d605ed44252251
58 58:e04755387e5b5968ec213e41f70c1d46
59 59:d5aa1729c8c253e5d917a5264855eab8
60 60:30e535568de1f9231e7d9df0f4a5a44d
61 61:97226236817061354807ba38c29559a0
62 62:77a39b8abc3537a3fdd89f45b60d4dc8
63 63:f34a3908f59a0b2f45e14dff8491ea5c
64 64:ba3fa31be0d37cfb6757af4e7aa15970
65 65:c177bb62142364a3a74fab69164e7ea4
66 66:3b5f93284ccd7328b62b2f33659a783d
67 67:ff42997d070390d82c5f5a526068ca75
68 68:5395b0afdcf8ad4c06e00db665d69786
69 69:0a7f78ddb5ca97c3ef6a7ce91c971e5a
70 70:ce174a820a207c24cf0823d90dfdf4fd
71 71:9aed2ef650794132fddb209791fea693
72 72:02ef8320d8752baede9ae5d7a54e7c2f
73 73:7410ccda4e35c38a5961726a55b2b9f1
74 74:e78495f9cbcb7bee2794e4a703f89773
75 75:1a9cdee77605eccf557029f80867e182
76 76:8242fb44bc89d635dd527ea537ec92a4
77 77:b1471adc34c852d9ca3f03f5f47ff496
78 78:7334aff097dbb250e91adb9a97b757fc
79
```

F1 file will get a new entry corresponding to Norman, however, **ALL THE ENTRIES are RESHUFFLED** in F1. This is done so that the attacker doesn't realize which is the sugar index among the given indices. (Note: Here norman is in row 18 and has K (number of indices) value 8 – Administrator randomly chose this value for him)

1	Richard:72,5,19,54,4,59,65,64
2	Christopher:6,17,21,10,28,14
3	nobita:16,1,24,18,7,13,11,6
4	Joshua:6,16,21,28,33,29
5	Brittany:70,5,19,67,4,43,69,15
6	Amanda:6,13,21,33,34,14
7	Crystal:76,5,19,60,4,64,42,46
8	Jeffrey:74,5,19,60,4,11,13,15
9	melissa:5,7,1,10,14,17
10	varun:7,17,5,11,9,2
11	Brian:50,35,43,26,2,17,37,9
12	Brandon:6,29,21,37,43,14
13	Heather:54,50,41,26,2,39,52,9
14	Thomas:62,11,14,26,2,44,38,9
15	Joseph:6,32,21,16,40,29
16	dgrason:1,3,7,14,13,12
17	Nicole:51,16,34,26,2,47,30,9
18	norman:21,58,23,78,19,44,5,13
19	king:2,7,9,10,13,15
20	Amber:69,5,19,16,4,60,62,64
21	Amy:75,5,19,69,4,30,28,17
22	Megan:58,31,13,26,2,50,51,9
23	Jessica:6,27,21,28,29,14
24	sheldon:13,1,7,9,20,17
25	Michael:6,12,21,11,27,14
26	bengio:21,11,6,13,8,22
27	Ryan:6,27,21,41,42,36
28	Daniel:6,29,21,12,35,31
29	David:6,28,21,31,36,17
30	Adam:57,31,13,26,2,51,34,9
31	Elizabeth:56,40,47,26,2,31,44,9
32	John:6,33,21,37,39,14
33	bob:13,2,12,4,7,6
34	obama:9,15,18,19,1,12
35	William:47,29,34,26,2,39,38,9
36	Justin:45,31,13,26,2,37,38,9
37	Jason:6,28,21,42,44,43
38	Laura:67,5,19,61,4,13,60,66
39	Lauren:68,5,19,62,4,14,42,47
40	Andrew:6,35,21,33,41,37
41	katie:3,9,8,11,14,17
42	Danielle:71,5,19,29,4,54,48,46
43	Jennifer:6,11,21,16,32,28
44	Markel:59,16,17,26,2,39,13,9

CASE 4: Administrator using this tool for reporting weak passwords

```
make get_wordlist
make john_run make
john_show
```

Step 1: Run John on F2 and record the time.

```
root@VM: /home/seed/Desktop/ssd/assgn2/honeysimulate
dallas123      (73)
asdfgh123     (32)
ranger12      (45)
starwars21    (47)
robert999     (42)
princess987   (66)
summer456     (70)
ramesh123     (77)
tigger999     (39)
love456789    (71)
amanda456     (69)
ashley4567    (72)
andrew999     (38)
41g 0:00:00:02 DONE (2021-04-10 09:56) 15.64g/s 5474Kp/s 5474Kc/s 120438KC/s fi
limani..[04][03];Vamos![03]
Warning: passwords printed above might not be all those cracked
Use the "--show --format=Raw-MD5" options to display all of the cracked password
s reliably
Session completed.

real    0m3.359s
user    0m1.292s
sys     0m0.168s
root@VM:/home/seed/Desktop/ssd/assgn2/honeysimulate#
```

Here, we can see that the time for cracking about 61 passwords is 3-4seconds. These passwords are easily obtainable by John. However, we also observe that some difficult passords like “koothrapali123” for username rajesh, cannot be obtained despite any attempts of cracking.

Administrator can alert weak passwords like alice123 for alice and make sure, the password is changed for her.

Implementation Details

A. mainserver.c and supportserver.c

These programs are used for the main server processing which includes:

1. Implementing the login module and the new registration module.
2. During login process: The server checks if the entered password matches with that of any index for that user. If yes, then honeychecker server is contacted. If not, the situation is handled locally.

- During registration process: The new user is assigned an index and construct a set of sweet indices randomly from the honeypot accounts and other users accounts. The entry is added to F1 file and all rows are shuffled randomly to avoid detection. F2 file is updated with the index number and the hash value. The correct index along with username is communicated with honeychecker for keeping a note.

Here, f2 and f1 are used for reading each entry for F2 and F1 file respectively. This array of structures is used for storing the contents.

- md5hash is computed for each password. (Function defined in code)

- TCP server is used for communicating with honeyserver.

- supportserver.c is

imported to mainserver.c for additional implementations and for making code

understandability complex/ distributed (for preventing software reverse engineering based attacks)

B. honeyserver.c

This program is the implementation of the honeyserver

- Honeyserver receives username and matched index number from the mainserver. Honeyserver checks with correctindex.txt file if the matched index is indeed the sugarindex.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <string.h>
5  #include <assert.h>
6  #include <time.h>
7  #include <sys/socket.h>
8  #include <netinet/in.h>
9  #include <arpa/inet.h>
10 #include <openssl/md5.h>
11 #include "supportserver.c"
12 #define PORT 8080
13 #define MAX 1000
14 #define SIZE 1024
15 #define honeymin 6
16 #define honeymax 8
17
18 FILE *flogmain;
19
20 int n;
21 int sockfd;
22 struct f2
23 {
24     int index;
25     char hash[33];
26 };
27 struct f1
28 {
29     int k;
30     char name[MAX];
31     int list[10];
32     char hashlist[10][33];
33 };
34
35 struct f2 f2file[MAX];
36 struct f1 f1file[MAX];
37
```

- 2) If matched index is not sugarindex, then it is either of a honeypot account or of a normal user account. For this, the index is checked with the files users.txt and honeypots.txt that contain the information of the indices of the users and honeypots of the system respectively.
- 3) If breach is detected, then this is noted down in the log file and reported to the admin.
- 4) Appropriate message is sent back to the mainserver notifying it of the login status

(successful or unsuccessful)

structures will store the correct index for each user. This will read from a protected file, which contains the details.

TCP Transfer Control Protocol is used for establishing the connections between the servers.

C. updatehoney.c

- 1) Reads from files upusers.txt and uppasswd.txt to automatically update about 50 honeypot accounts and their respective passwords in this authentication system.

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <string.h>
5  #include <assert.h>
6  #include <time.h>
7  #include <sys/socket.h>
8  #include <netinet/in.h>
9  #include <arpa/inet.h>
10 #define MAX 1000
11 #define SIZE 1024
12
13 int n;
14 int u;
15 int h;
16 int new_sock;
17
18 FILE *flog;
19
20 struct correctkey
21 {
22     int usernum;
23     int sugarindex;
24 };
25
26 struct correctkey keys[MAX];
27
28 char crrectkeydetails[MAX][20];
29
30 int users[MAX];
31 int honeypots[MAX];

```

- 2) Please note: I have already run this and hence now I have 77 accounts in the system. One can just change the username and password in these files and run this code. It will automoaatically update the honeypots.

3) Please note: It is not required to run this file now, since, I have already run it.

D. All supporting files

F1.txt – Contains username and corresponding sweet indices.

F2.txt – Contains index and corresponding hash value

honeyinfo/correctindex.txt – Contains details of correct index for each user

(line matching done with F1.txt – not direct mapping) honeyinfo/users.txt –

Contains those indices who are normal users honeyinfo/honeypots.txt –

Contains info of honeypot accounts

E. **All Log files** mainserverlog.txt – Log file generated from mainserver.c

supportserverlog.txt - Log file generated from supportserver.c honeyserverlog.txt -

Log file generated from honeyserver.c

Interesting Observations

1. In the software point of view, I found it interesting that John the ripper 1.9.0 version works better than 1.8.0 (which gets install using apt-get on this seedubuntu machine). I put extra effort in downloading and compiling John the ripper latest version from Github and installed in from source. (Hence I use \$JOHN as the environment variable pointing to the executable john program in the local system for running it). In makefile I have provided just “john” hence it is compatible in most systems.
2. Correlations between username and passwords like alice and alice123 are easy prey for attackers. Hence, administrator should use John tool and make sure such incidents don't happen (before attacker exploits this).
3. This idea shows that there are few storage concerns with this approach (since k indices for each user). We can use methods like “honey circular list” or “paired distance protocol” proposed in the paper “[Towards Improving Storage Cost and Security Features of Honeyword Based Approaches.](#)”

Thank you!

Name: Nischal A