SYSTEM SECURITY (CS F321)
SEM I 2024-2025
ASSIGNMENT 2
MAX MARKS: 20 (WEIGHTAGE: 10%)
DEADLINE: 19th NOVEMBER 2024, 2:00 AM
(HARD DEADLINE, Late submission will incur penalty)

NOTE: PLEASE READ THE ENTIRE DOCUMENT AND NOT JUST THE PROBLEM STATEMENT. THIS DOCUMENT CONTAINS IMPORTANT INFORMATION REGARDING SUBMISSION GUIDELINES, LATE SUBMISSION POLICY, PLAGIARISM POLICY AND DEMO GUIDELINES IN ADDITION TO THE PROBLEM STATEMENT.

ALLOWED PROGRAMMING LANGUAGE: C

PROBLEM STATEMENT:

In this assignment, you will have to implement a role mining algorithm proposed in the research paper titled "***Meeting Cardinality Constraints in Role Mining***" (file name Meeting_Cardinality_Constraints_in_Role_Mining.pdf is attached along with the assignment document) published in IEEE Transactions on Dependable and Secure Computing (TDSC). Hence, it is highly recommended that you read the research paper first and then go through the assignment statement. In addition to the research paper mentioned above, you may also need to go through the following research papers for better understanding.

- Title: *Fast Exact and Heuristic Methods for Role Minimization Problems* (file name mbc-sacmat.pdf is attached along with the assignment document) published in the 13th ACM Symposium on Access control Models and Technologies (SACMAT).
- Title: *Towards Role Mining with Restricted User-Role Assignment* (file name Towards_role_mining_with_restricted_user-role_assignment.pdf is attached along with the assignment document) published in the 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE).

You need to implement (in C language only) the concurrent processing framework proposed in the paper titled "Meeting Cardinality Constraints in Role Mining" (described in Section 4 of the paper). This framework implements a constrained role mining algorithm. The cardinality constraints enforced are Role-usage cardinality constraint (maximum number of roles that can be assigned to a user) and the Permission-distribution cardinality constraint (maximum number of roles to which a permission can belong). The concurrent processing framework enforces the two cardinality constraints while mining the minimal number of roles and not as a post-processing step. You have to implement Algorithms 4 and 5 proposed in the paper. Note that Algorithm 4 uses a dual of Algorithm 5. This dual algorithm has not been specifically mentioned in the paper. You need to figure out the dual algorithm as well as implement it. Though you are required to implement only one framework proposed in the paper, it is highly recommended that you read the entire paper to gain a holistic understanding. Note that the algorithm used here to mine the minimal number of roles is a heuristic one and may not always produce the optimal solution in presence of the two cardinality constraints.

The input to your program should be the following. Note that these inputs should not be hardcoded in the program.

- The UPA matrix is supplied in the form of a text file (.txt file). The format of the text file is as follows.
    - The first line contains the number of users of the UPA.
    - The second line contains the number of permissions of the UPA.

- Each subsequent line contains the row number and the column number of the cell of the UPA matrix that contains a 1. Note that the row and column numbering start from 1. Hence, for the UPA matrix shown below, the contents of the text file are given in UPA.txt (UPA.txt is attached along with the assignment document).

	p1	**p2**	**p3**	**p4**	**p5**	**p6**	**p7**	**p8**
u1	1	0	1	0	1	0	1	0
u2	1	1	1	1	1	0	1	0
u3	1	1	1	1	1	1	0	1
u4	0	1	0	1	1	0	0	1

- Your program should take the name of the file as a user input using a prompt as follows.

```
Enter the name of the UPA matrix file:
```

Complete file along with the extension should be entered.

- The value of the role-usage cardinality constraint using a prompt as follows.

```
Enter the value of the role-usage cardinality constraint:
```

- The value of the permission-distribution cardinality constraint using a prompt as follows.

```
Enter the value of the permission-distribution cardinality constraint:
```

You are given six real-world datasets which have been widely used by researchers for evaluating the performance of role mining algorithms in the form of UPA matrices. These datasets are given in the form of text files in a zipped file named Datasets. The format of these text files is the same as the one mentioned above for the UPA matrix file. You are not allowed to change the format and the name of any of the datasets. During the demo, you may be supplied with fresh copies of the same datasets and the ones attached with the assignment may not be used. The details of the datasets are shown in Table 1. These details are also present in Table 1 of the paper. The last column indicates the number of roles generated by an unconstrained role mining algorithm, i.e., an algorithm that does not consider any cardinality constraint.

Table 1: Details of Real-world Datasets

Dataset	**No. of Users**	**No. of Permissions**	**No. of Roles (by unconstrained role mining algo.)**
apj	2044	1164	456
domino	79	231	20
healthcare	46	46	15
firewall1	365	709	69
firewall2	325	590	10
americas_small	3477	1587	211

After implementing the algorithm for the concurrent processing framework, you need to recreate the following experimental results (shown in Table 2 below) reported in the paper (the detailed set of results is reported in the supplemental material of the paper). More specifically, for the given values of the two constraints, your implementation should generate the mentioned number of roles while ensuring that the solution is 0-consistent. Marks will be deducted if constraint(s) is/are not satisfied or you generate more number of roles than those reported in the paper for the given combination of constraint values. Note that you are not required to generate the full set of roles reported in the paper. The results that you are required to generate correspond to the heuristic named as NU in the paper.

Table 2: Experimental Results

Dataset	Role-usage cardinality constraint (MRC_{user})	Permission-distribution cardinality constraint (MRC_{perm})	No. of Roles
apj	8	67	458
	6	67	461
domino	11	7	20
	6	6	20
healthcare	6	8	15
	6	6	15
firewall1	21	25	70
	16	25	70
firewall2	9	3	10
	8	2	10
americas_small	16	70	215
	12	60	222

There is no need to report the execution time of your program. However, the execution time of your program should not be too much higher than what is reported in the paper (so that all the outputs are generated within your demo slot). Your program should generate the following outputs for each of the six datasets.

- The number of roles generated should be printed on the console using a display message as follows.
  `Number of roles = x` (`x` will be an integer)

Marks will be deducted if you do not display the number of roles on the console.

- A UA matrix should be generated in the form of a text file. The contents of the UA matrix should not be printed on the console. The format of the UA matrix will be as follows.
  - The first line contains the number of users.
  - The second line contains the number of roles generated.
  - Each of the subsequent lines corresponds to one row of the UA matrix. Each element of a row is separated by a whitespace character. The UA created for UPA.txt for MRC_{user} = 3 and MRC_{perm} = 2 is attached as UA.txt along with the assignment document. Note that for large datasets, a single row of the UA may span multiple lines.
  - Name the text file for UA as dataset_UA.txt (like apj_UA.txt, firewall1_UA.txt, americas_small_UA.txt, etc.). You need to follow the naming convention exactly.
- A PA matrix should be generated in the form of a text file. The contents of the PA matrix should not be printed on the console. The format of the PA matrix will be as follows.
  - The first line contains the number of roles generated.
  - The second line contains the number of permissions.
  - Each of the subsequent lines corresponds to one row of the PA matrix. Each element of a row is separated by a whitespace character. The PA created for UPA.txt for MRC_{user} = 3 and MRC_{perm} = 2 is attached as PA.txt along with the assignment document. Note that for large datasets, a single row of the PA may span multiple lines.
  - Name the text file for PA as dataset_PA.txt (like apj_PA.txt, firewall1_PA.txt, americas_small_PA.txt, etc.). You need to follow the naming convention exactly.

If your program simply prints the number of roles generated, no marks will be awarded at all. It is mandatory for you to generate the UA and PA matrices files as well.

You will be evaluated in terms of the following parameters.

- Correct implementation of the concurrent processing framework algorithm.
- Generating the same number of roles for the given constraint values as reported in Table 2 of the assignment document. **Marks will be deducted if you generate more number of roles than what is reported in Table 2.**
- The solution generated should be 0-consistent, i.e., UA $\otimes$ PA = UPA and should satisfy both the cardinality constraints simultaneously. These will be checked using a checker program whose output file (named as checker.out) has been shared along with the assignment document. Note that this checker program will be used during the demos. The checker program will take the names of the UPA, UA and PA files (like UPA.txt, UA.txt and PA.txt respectively) as well as the values of the role-usage and permission-distribution cardinality constraints as inputs and will inform you whether the solution is 0-consistent and whether it satisfies both the cardinality constraints simultaneously. In case of any constraint violation(s), the relevant message(s) will be displayed on the screen. This consistency checker program will expect the formats of the UPA, UA and PA files to be exactly the same as described earlier. So, make sure that you follow the exact file formats for UA and PA, and you should not alter the formats of the given UPA files. No concessions will be made regarding this during the demos.
- **No marks will be awarded if your solution is not 0-consistent** irrespective of the number of roles generated.
- **Marks will be deducted if your solution violates any one or both constraints.**

SUBMISSION GUIDELINES:

- **All submitted program(s) should be C programs.**
- **All codes should run on Ubuntu 22.04 or 24.04 systems. You can be asked to demo the application on either system variant.**
- Submissions are to be done using the following Google Form: **https://docs.google.com/forms/d/e/1FAIpQLScOeLNYkhIyDYYFbIE3Kp4SIsAdXkylwhvJhtIOlzl8OJkF5Q/viewform**
- There should be only submission per group. If multiple submissions are made, only one will be selected randomly for evaluation.
- Each group should submit a zipped file containing all the relevant C programs and a text file containing the correct names and IDs of all the group members. Names and IDs should be written in uppercase letters only. The zipped file should be named as GroupX_A2 (X stands for the group number). You have been notified of your group number in a few days. **DON'T MAKE ANY MISTAKE WHILE MENTIONING GROUP NUMBER**.
- **DO NOT SUBMIT ANY FILE OTHER THAN THE ONES MENTIONED ABOVE WHILE SUBMITTING THE ASSIGNMENT. NO UPA, UA OR PA MATRIX FILES SHOULD BE SUBMITTED. DO NOT SUBMIT THE checker.out FILE EITHER.**
- Your group composition has frozen now. You cannot add or drop any group members now.

LATE SUBMISSION POLICY:

- The Google form will accept submissions beyond 2 AM, 19th November 2024.
- ***SUBMITTING AFTER THE DEADLINE WILL INCUR A LATE PENALTY.***
- **For every 15 minutes of delay, there will be a late penalty of 0.25 marks. However, no distinction will be made within that 15-minute window. For eg., If a group submits between 2:01 AM and 2:15 AM (inclusive), then that group will lose 0.25 marks. If the submission is made at 2:10 AM, then also the penalty will be of 0.25 marks.**

- It is your discretion what you want to do – take a penalty and correct some last-minute error or submit a partially correct implementation (which will lead to marks deduction).
- For calculating the late penalty, the date and timestamp of the submission via the Google form will be considered. No arguments will be entertained in this regard.

PLAGIARISM POLICY:

- All submissions will be checked for plagiarism.
- Lifting code/code snippets from the Internet is plagiarism. Taking and submitting the code of another group(s) is also plagiarism. However, plagiarism does not imply discussions and exchange of thoughts and ideas (not code).
- All cases of plagiarism will result in being awarded a hefty penalty. Groups found guilty of plagiarism may be summarily awarded zero also.
- **Be very careful about sharing codes with students belonging to other groups via email or chats.**
- All groups found involved in plagiarism, directly or indirectly, will be penalized.
- The entire group will be penalized irrespective of the number of group members involved in code exchange and consequently plagiarism. So, each member should ensure proper group coordination.
- The course team will not be responsible for any kind of intellectual property theft. So, if anyone is lifting your code from your laptop, that is completely your responsibility. Please remember that it is not the duty of the course team to investigate cases of plagiarism and figure out who is guilty and who is innocent.
- **PLEASE BE CAREFUL ABOUT SHARING CODE AMONG YOUR GROUP MEMBERS VIA ANY ONLINE CODE REPOSITORIES. BE CAREFUL ABOUT THE PERMISSION LEVELS (LIKE PUBLIC OR PRIVATE). INTELLECTUAL PROPERTY THEFT MAY ALSO HAPPEN VIA PUBLICLY SHARED REPOSITORIES.**

DEMO GUIDELINES:

- The assignment also consists of a demo component to evaluate each student's effort and level of understanding of the implementation.
- You will be supplied with the UPA files and the checker.out file during the demo.
- The evaluators will evaluate the output using the checker program. They will have their own checker program.
- **The demos will be conducted in either the I-block labs or D-block labs. Therefore, the codes will be tested on the lab machines.** No group will be allowed to give the demo on their own laptop.
- **The codes should run on Ubuntu 22.04 or 24.04.**
- All group members should be present during the demo. Any absent group member will be awarded zero.
- The demos will be conducted in person. The demos will not be rescheduled.
- Though this is a group assignment, each group member should have full knowledge of the complete implementation. During the demo, questions may be asked from any aspect of the assignment.
- **Demos will be conducted on 23rd and 24th November 2024. Demos can be conducted on any one or both days. So, you need to be available on campus for both days. You need to book your demo slots as per the availability of your entire group.**
- The code submitted via the Google form will be used for the demo. No newer version of the code will be considered for the demo.
- Each group member will be evaluated based on overall understanding and effort. A group assignment does not imply that each and every member of a group will be awarded the same marks.
- Any form of heckling and/or bargaining for marks with the evaluators will not be tolerated during the demo.

****