

CS 461 / ECE 422 Information Assurance

HW #3

Due submitted to Compass by 3:30 p.m. Feb. 28

1. (2 points) Of the two ways of slicing an Access Control Matrix (by row, or by column), explain which of these (if either) would likely be faster at checking an access by a given user to a given object. Explain which of these (if either) would likely require less overall storage space than another.

**(1 pt) In most systems, the number of users is much less the number of shared objects. Thus, slicing ACM by column (object) is likely to be faster at checking an access by a given user to a given object.**

**(1 pt) By the same reason, the required overall storage space is also usually less than that of slicing ACM by row (subject).**

2. (2 points, 0.5 point for each option) Consider a Unix system, and a file /home/prof/ece422/exam/key.doc . prof is a user, with home directory /home/prof. 'staff' is a group that contains 'prof' and user 'TA'. Suppose that the permissions on these directories (and file) are

Name	user	group	permissions
home	root	root	drwxr-x-r-x
prof	prof	staff	drwxr-x—x
ece422	prof	class	drwxr-x—x
exam	prof	prof	drwxrwx---
key.doc	prof	staff	-rw-r-xr-x

Which of the following accesses are permitted (circle)

**User prof can delete file key.doc**

- User TA can add a new file corrected-key.doc to subdirectory exam.
- User student can read file key.doc (the directory “exam” holding “key.doc” cannot be accessed by student, so student cannot read “key.doc” either)**

**User TA can execute the ‘ls’ command on path /home/prof/ece422 and see all the names of files and directories it contains.**

If you assume “class group” includes TA, the fourth option could be circled. Otherwise, it should not be circled.

3. (2 points) Describe the difference between segmented memory, and paged memory, with respect to the security mechanisms each can employ and efficiency of memory use.

**(1point) Segmented memory is sliced in terms of logical unit. Each segment stores a piece of data with semantic meanings, such as code, data, etc. Operating system can place any segment at any location or move any segment to any location. Every address reference passes through the operating system, so there**

is an opportunity to check each on for protection. Because segments are logical units, segmentation offers a number of security benefits: we can associate different segments with individual protection rights, such as read-only (non-variable) or execute-only (for program code). Many different classes of data items can be assigned different levels of protection. Two or more users can share access to a segment, with potentially different access rights. The main drawback of segmentation is segment size. Since a segment has fixed size, for each access, the *offset* must be checked to prevent reference from exceeding the bound. It also cannot handle dynamic data structure which could demand more space than the segment can provide initially. Encoding segment name also introduces the inefficiency of instruction decode and data sharing.

(1 point) In paged memory, the program is divided into equal-sized pieces. Paging can provide protection between users or processes in coarse granularity. The problems in segmentation due to size are fixed inherently. However, each page has no semantic meanings, i.e. there is no necessary unity to the items on a page, so there is no way to establish that all values on a page should be protected at the same level.

A solution is to combine paging with segmentation, taking advantage of both implementation efficiencies in paging and logical protection characteristics in segmentation.

4. (1 point) Could one implement a “tagged architecture” in software only, or is it absolutely necessary for there to be hardware support? Discuss the issues and any tradeoffs that may exist.

Yes, it is possible. One basic method is to use one or two more bytes memory for each memory unit to indicate the unit’s protection level. When a process wants to access the unit, the extra byte (just like a tag) will be checked first. It also can be implemented by segmentation if the size of each segment is small enough to meet the security requirements and corresponding security tag/bits are added to segment translation table. Obviously, such mechanisms will introduce much more overhead with respect to both time and space. For the segmentation solution, the problems due to the size of segments will become more serious.

5. (3 points) Suppose that a password may be comprised of any strings from the set {a-z, A-Z, 0-9, -, \_, ~, !, @, #, \$, %, ^, &, \*, +}, with the provision that the first character must be alphanumeric (e.g., from a-Z, A-Z, or 0-9). An “attack” is considered to be the event that an attacker obtains a list of 100 password hashes, where the attacker knows that a hash output comes from is one of 4096 different possible hash functions that the attacker can compute. The attacker can generate, hash (once) and test against the list  $10^6$  passwords per second (the hash+test is the dominate cost, you can assume the password itself takes 0 time to generate). How long should the system administrator insist passwords be to ensure that the probability of any one of the passwords being discovered in 1 year of sustained attack is less than 0.1? For simplicity, assume
  - a. That all passwords are exactly of the minimum length

- b. The attacker knows this password length
- c. The attacker generates passwords at random (allowing for duplicates)

**With one million password tests per second, the attack can test the password**

**$60 \cdot 60 \cdot 24 \cdot 365 \cdot 10^6 = 3.15 \cdot 10^{13}$  times. The probability of one match is:**

**$\frac{(\text{\# of password hashes known to the attacker})}{(\text{\# possible passwords} \times \text{\# hash functions})} = \frac{100}{62 \cdot (74)^{n-1} \cdot 4096}$ . Since the attacker**

**generates passwords at random, the probability that the attacker doesn't get any**

**one of the passwords in a year is  $p = \left(1 - \frac{100}{62 \cdot (74)^{n-1} \cdot 4096}\right)^{3.1510^{13}}$ . By**

**solving  $p \geq 0.9$ , we get the length of password  $n$  should be at least 7.**