CS6600 Homework 7

Matthew Whitesides

I. Chapter 8 Problems

1) Sanitized objects must be in their COI class because sanitized data contains information that is accessible by subjects without creating a COI within the companies data (i.e., if it's public data), so to keep track of that without restricting the data they must be in their class with only sanitized objects.
2) An algorithm to generate an access control matrix *A* from a given history *H* could look something like this.
   - For each subject *s* in *H* append a new row and column to *A*.
   - For each object *s* in *H* append a new row and column to *A*.
   - This gives us ACM *A[(#S+#O),(#S+#O)]* set all initial rights to **allowed**.
   - For each *h[s,o]* in *H* set all cells in *For each o' in O': A[s, o']* rights to **not allowed** if o is not sanatized.
     - Where *O'* is a set of all values where $CD(O') \Rightarrow CD(O)$.
3) While the BLP model cannot show history over time, the CWM can at any iteration support the BLP model. Take a given CWM construction and make the following modifications to support BLP.
   - Sanatized and unsanatized sets represent a security level S for sanatized and U for unsanatized where S dom U.
   - Each CD and COI object represent categoires a subject has initial access to (i.e. $s1 = (S, \{a, c, b, s\})$).
   - Each time a subject access an object in a COI class the objects in all other COI classes where $CD(O') \Rightarrow CD(O)$ are removed from the subjects set.

   While this generally gives most subjects a top-level security clearance and each category is its object, it does fit the rules for BLP. Ideally, you may have another set of subcategories to capture the CD and COI sets if you want to convert it back to a BLP model.

II. Chapter 9 Problems

3) Consider the two-bit machine.
   a) The values of ach will be:
      - $proj(Holly, cs, oo)$: 0101010
      - $proj(Lucy, cs, oo)$: 1110
      - $\pi Lucy(Cs)$: (Holly, xor0)
      - $\pi Holly(Cs)$: (Lucy, xor0), (Lucy, xor1)
      - $\pi Lucy, xor0(Cs)$: (Holly, xor0), (Lucy, xor1)
      - $\pi Holly, xor0(Cs)$: (Lucy, xor0), (Lucy, xor1)
      - $\pi Lucy, xor1(Cs)$: (Holly, xor0), (Lucy, xor0)
      - $\pi Holly, xor1(Cs)$: (Holly, xor0), (Lucy, xor0), (Lucy, xor1)
      - $\pi xor1(Cs)$: (Holly, xor0), (Lucy, xor0)
   b) The values of each will be:
      - $proj(Holly, cs, oo)$: 01010100
      - $proj(Lucy, cs, oo)$: 1110
      - $\pi Lucy(Cs)$: (Holly, xor0)
      - $\pi Holly(Cs)$: (Lucy, xor0), (Lucy, xor1)
      - $\pi Lucy, xor0(Cs)$: (Holly, xor0), (Lucy, xor1)
      - $\pi Holly, xor0(Cs)$: (Lucy, xor0), (Lucy, xor1)
      - $\pi Lucy, xor1(Cs)$: (Holly, xor0), (Lucy, xor0)
      - $\pi Holly, xor1(Cs)$: (Holly, xor0), (Lucy, xor0), (Lucy, xor1)
      - $\pi xor1(Cs)$: (Holly, xor0), (Lucy, xor0)
4) a) If blocking was not used the system would not be noninterference-secure, for example if louie sends messages to BLH nad BLDH and signals to dewey that their complete at the same time hughie attempts to read from BH his high output will appear in BL which dewey may attempt to read from thus making dewey potentially read from a HIGH source.
   b) If the buffers were unbounded this would lead to a similar issue with the non-blocking, data from the high source potentailly could be added to the buffer and a block is released, however only part of the data may be read leaving data in the buffer for the next messages, thus potentailly leaving open a cycle where high information is in the buffer with low information being used by louie or dewey.

5) No in this new rule set if only the High or Low state bit was output based upon the operation performed that would give lucy both a different number of output bits (violating 9-4) and since the bits not effected are no longer output this would effect the outcome of the low bits thus giving lucy information about the operations performed.
6) The following input sequences could achive a low ouput of 011011 thus the total output could be 10?1?1?0?1?1: $0_H010_H100, 01_H0101_H100, 1_H010_H100, 1_H011_H100$.

ACKNOWLEDGMENT

The author would like to thank Professor Bruce McMillin with the Department of Computer Science, Missouri University of Science and Technology.

Matthew Whitesides Master's Student at Missouri University of Science and Technology.