CS526 Fall 2010 Midterm solutions, 20 October 2010
Prof. Chris Clifton

Going into this, my expectation is that an A student should get at least 26 of the available 32 points, a B student at least 19, and a C student at least 13.)

1 Access Control Matrix (10 minutes, 8 points)

Given the following command for the Harrison-Ruzzo-Ullman Access Control Matrix model:

```
command justcreate($s_0, s_1, o$)
    if $f$ in $a[s_0, s_1]$ and $m$ in $a[s_1, s_0]$
    then
        create object $o$
end
```

1.1 Operation Execution (3 minutes, 2 points)

Given the following matrix, show the result after executing the operation *justcreate*(A, B, O_5).

	O_1	O_2	O_3	O_4	…	A	B	C	…
A	r	rw	ra	e	…		f		
B	w	rw	ra	aw	…	m			
C	r				…	f			
…		…							

	O_1	O_2	O_3	O_4	…	O_5	A	B	C	…
A	r	rw	ra	e	…			f		
B	w	rw	ra	aw	…		m			
C	r				…		f			
…			…							

Scoring: 1 for added O_5, 1 for no changes to rights.

1.2 Safety of the Command (2 minutes, 3 points)

Is a system consisting of only the above command safe with respect to the right f?

- Yes
- No
- Undecideable

Briefly justify your answer.

Yes, it is safe. Since the command cannot add an f right, even given repeated executions no new f will appear in the matrix, so f is not leaked.

Scoring: 1 for "yes", 1 for "no rights leaked", one for "command doesn't add any rights".

Note that while this is a biconditional commands, and it has been shown that there exist systems of biconditional commands for which safety is not decideable, this is also a monooperational commands, a class which is decideable.

1.3 Operation Limits (5 minutes, 3 points)

Would it be possible to modify the access control matrix (not the program) such that *justcreate* would create a new object given only one subject (i.e., to run *justcreate*(C, C, O_6))? If no, explain why not. If yes, explain how.

A subject containing both m and f rights can create an object. For example, if $f \in a[C, C]$ and $m \in a[C, C]$, then *justcreate*(C, C, O_6) would create a new object O_6.

Scoring: 1 for "yes", 1 for getting matrix right, 1 for explanation.

2 Take-Grant (3 minutes, 4 points)

Answer ONE of the following three questions. (If you answer more than one, only your best answer will be scored.)

2.1 Terminal Span (3 minutes, 4 points)

Subject x and y have rights R_x, R_y over some other subjects/objects. x terminally spans to y (series of $\overrightarrow{t}$ edges from x to y). After a finite number of takes and grants, what can you say about the final set of rights of $x : R'_x$ and of $y : R'_y$ in terms of R_x and R_y? Why?

$R'_x = R_x \cup R_y$, $R'_y = R_y$. **x terminally spans to y means that x can take the rights that y has.**

2.2 Initial Span (3 minutes, 4 points)

Subject x and y have rights R_x, R_y over some other subjects/objects. x initially spans to y (series of $\overrightarrow{t}$ or $\overrightarrow{g}$ edges from x to y). After a finite number of takes and grants, what can you say about the final set of rights of $x : R'_x$ and of $y : R'_y$ in terms of R_x and R_y? Why?

$R'_y = R_y \cup R_x$, $R'_x = R_x$. **x initially spans to y means that x can grant rights to y.**

2.3 Islands (3 minutes, 4 points)

Subject x and y have rights R_x, R_y. x and y are part of a maximal tg-connected island (subject-only subgraph connected by t or g edges.) After a finite number of takes and grants, what can you say about the final set of rights of $x : R'_x$ and of $y : R'_y$ in terms of R_x and R_y? Why?

$R'_y = R_y \cup R_x$, $R'_x = R_y \cup R_x$. **x and y are in an island means that x and y can share all the rights they have.**

Please note which question you answered, 2.1, 2.2, or 2.3: Scoring: 1 for correct maximal rights, 1 for original rights, 1 for subset, 1 for rough explanation, +1 for solid explanation.

3 Protection Models (15 minutes, 6 points)

We want to model a system where *security administrator*s (SAs) have control over objects, and determine who (i.e., userid) is allowed to access what object. However, the security administrators are not allowed to access the objects themselves.

3.1 Take-Grant (7 minutes, 3 points)

Can we model such a system using take-grant? Give a sketch of how this would be done, or briefly explain why it can't be modeled using take-grant.

Such a system cannot be modeled using take-grant. In take-grant, a subject has access rights over an object or subject. In this system, if SAs do not have any access rights on the objects, then they cannot grant any such rights to others. If SAs do not have any access rights, but "take" rights from some other dummy user-id (which has all access rights to all objects), then an SA takes an access right on an object and would give it to a user-id. The SA can in fact access an object after it takes the right.

Scoring: 1 for "no", 1 for "can get any right you can give", 1 for proof sketch of this. (Note that "can't grant a right you don't have" is not quite the same as "can get any right you can give" - for example, I could grant a take right to someone enabling them to take the right they need, even though I don't have that right.)

3.2 Schematic Protection Model (8 minutes, 3 points)

Can we model such a system using the Schematic Protection Model? Give a sketch of how this would be done, or briefly explain why it can't be modeled using SPM.
Scoring: 1 for "yes", 1 for capturing types behaving differently, 1 for other good explanation.

4 Take-Grant in HRU (8 minutes, 5 points)

We would like to have "take" and "grant" commands within the HRU (access control matrix) model. Let the access control matrix be *a*. **Answer ONE of the following questions** (note which you answer - if you answer both, you will receive the score for the best one.)

4.1 Take rule (8 minutes, 5 points)

Sketch (or for full credit, give code) how you would implement the command *take(taker, takee, object, rights)*: subject *taker* takes rights *rights* on *object* from *takee.*

```
command take (taker, takee, object, rights)
if taker in S and takee in S and object in O
and rights in a[takee, object] and take in a[taker, takee]
then
enter rights in a[taker, object]
end
```

4.2 Grant rule (8 minutes, 5 points)

Sketch (or for full credit, give code) how you would implement the command *grant(grantor, grantee, object, rights)*: subject *grantor* gives *rights* on *object* to *grantee.*

```
command grant (grantor, grantee, object, rights)
if grantor in S and grantee in S and object in O
and rights in a[grantor, object] and grant in a[grantor, grantee]
```

then
enter rights in a[grantee, object]
end

Please note which question you are answering, 4.1 or 4.2: Scoring: 1 for take/grant right check, 1 for proper right ownership check, 1 for getting all subjects/objects, 1 for rough code, 1 for good code.

5 Bell-LaPadula Model (17 minutes, 9 points)

We want to extend Multics with a means to "declassify" documents. The way this will work is that an appropriately cleared subject will read an object, then write a redacted version of the document (with higher level information removed) to a lower-level object. We write a formal *declassify* rule 12 taking the subject, high and low objects, and discretionary acess control matrix, $r = (\textit{declassify}, s, o_h, o_l, \underline{d}) \in R^{(12)}$, as:

```
if $(r \notin \Delta(\rho_{12}))$ then $\rho_{12}(r, v) = (\underline{i}, v)$;
else if$(f_s(s)\ dom\ f_o(o_h)$ and $[s \in S_T$ or$f_c(s)\ dom\ f_o(o_h)]$ and $f_o(o_l)\ dom\ f_c(s)$ and
        $\underline{d} \in m[s, o_h]$ and $\underline{r} \in m[s, o_h]$ and $\underline{w} \in m[s, o_l])$
    then $\rho_{12}(r, v) = \{\underline{y}, (b \cup \{(s, o_h, \underline{r}), (s, o_l, \underline{w})\}, m, f, h))$;
else $\rho_{12}(r, v) = (\underline{n}, v)$;
```

5.1 Correctness (4 minutes, 3 points)

Does this give the desired functionality (an appropriately cleared s can read o_1, and write a declassified version to o_2 at a lower level of classification? Explain.

It would work; however, the code does not check for the declassification itself: it should check if $f_o(o_1)\ dom\ f_o(o_2)$.

Scoring: 1 for "yes", 1 for getting levels in your proof/explanation, 1 for getting the idea of trust in your proof/explanation.

5.2 Confidentiality (5 minutes, 3 points)

Does this satify the Bell-Lapadula definition of a secure system? Explain.

Yes. ssc-property remains to be true, if the system before execution of this rule also had ssc-property true. It is because $f_c(s)\ dom\ f_o(o_1)$ held earlier. *-property also holds: because $f_o(o_2)\ dom\ f_c(s)$. ds-property also holds, as there is no change.

Scoring: 1 for "yes", 1 for good explanation, 1 for noting trust, 1 for proof.

5.3 Information Flow (5 minutes, 3 points)

Describe briefly how we might use information flow concepts to measure the potential amount of information revealed using this declassification function.

Scoring: 1 for idea of entropy or other good idea, 1 for getting use correct, 1 for decent explanation of meaning.