

CS4605 Final Exam Questions

Summer 2004

George W. Dinolt

September 15, 2004

Introduction

You should print out a hard copy of this exam, if you have not already done so. Please answer all the questions on these pages. Make sure your name is on the title page and each separate sheet, if the pages are not stapled together. You may also create an electronic copy which you may fill in and send to John Clark via email. Be sure that your name is on the pages that you send him. If you send him an email copy, your exam has not been “accepted” until you receive a receipt by return mail.

The exam is due to John Clark, SP 531A, or via email at 0900 hrs on Wednesday, Sept. 22, 2004.

The exam is made up of three parts, several PVS theorems to be proved, a pvs specification to be fixed and a series of questions to be answered. You may use your class notes and slides and papers and books you have read to answer any of the questions. If you use a paper as a source, please give a reference to the paper.

You can attach the evidence of your “proofs” as a separate sheet to the exam.

Several of the questions are based on the paper *THE CHINESE WALL SECURITY POLICY*, by David Brewer and Michael Nash published in the proceedings of the **1989 IEEE Symposium on Security and Privacy**. You can find the paper on the CD that I handed out in class. You should read this paper and try to make some sense of the model.

The Questions

1. Briefly describe each of the security policies. It would be helpful if you used the same terms, preferably the ones we talked about in class, for each of the policies.

(a) Mandatory Access Control Policy

(b) Discretionary Access Control Policy

(c) High Water-mark

(d) Chinese Wall

(e) Biba Integrity Policy

(f) Role Based Access Control Policy

(g) Clark/Wilson Policy

(h) Non-Interference (Goguen/Meseguer)

2. Which of the above policies could be used together on the same system?

3. In the class, we have described the process of the application of “formal methods” to the design and implementation of secure systems. What is this process?

4. What reasons are there for **not** using formal methods on a project?

5. What reasons are there for using formal methods on a project?

6. In the Landwehr paper *Formal Models for Computer Security* that we discussed in class, where does the Chinese Wall Policy fit in the table “Comparisons of Properties of Models” on page “273”? You should add a column to the table that includes the Chinese Wall Security Policy. Explain the meaning of the entries to the table that you add and why you put them there.

The Biba Security Model

As you may remember, the *Biba Integrity Model* is similar to the *Bell & LaPadual Model* in the sense that there are subjects, objects and labels. In the *Biba* case, they are called **Integrity Labels**. The basic idea is that a subject should only read information of higher integrity and write information at lower integrity. Information at high integrity should not be compromised by information at low integrity.

The goal of this exercise is to show that PVS may be useful in finding flaws in specifications. I have constructed a specification of the *Biba Integrity Model*. Unfortunately, the specification has a problem. One can prove that the `empty` state is secure but one cannot prove that the `transform` function is secure.

Your job is to find out where the proof fails, and as a result find the simple (single) place in the specification where a change can be made that will allow you to complete the proof. The change you make should maintain the spirit of the Biba Policy.

In your exercise below, you should not use `grind` or similar commands. The work should be accomplished using the various commands we have illustrated in the other exercises.

What you should Hand In

1. You should provide enough detail about the proof of the `transformSecure` lemma that will show exactly where the problem arises, i.e. the place where the hypotheses and the conclusions contradict each other,
2. You should provide a corrected version of the specification, highlighting the change(s) that you have made and why you made the changes, and finally
3. You should provide a proof of the `transformSecure` lemma using your new specification.

Make sure that your name and the label “Biba Model” is on every page you hand in, to ensure we know which proof to credit.

The Specification

You can download the “broken” specification from <http://www.nps.navy.mil/cs/dinolt/Courses/AY2004/Summer/CS4605/Final/Biba.pvs> or on proof in the file `/disk1/cisr/pvs-examples/Final/Biba.pvs`. It is also presented below.

```

biba: THEORY
BEGIN

Subjects: TYPE+

Objects: TYPE+

AccessModes: TYPE =
{read, write}

Labels: TYPE+ =
{i: nat | i ≤ 4}

Accesses: TYPE+ =
[# sub: Subjects,
 ob: Objects,
 am: AccessModes #]

State: TYPE+ = setof[Accesses]

OLB: [Objects → Labels]

SLB: [Subjects → Labels]

transform(a: Accesses, s: State): State =
  IF (a'am = read ∧ SLB(a'sub) ≥ OLB(a'ob)) ∨
    (a'am = write ∧ SLB(a'sub) ≥ OLB(a'ob))
  THEN (s ∪ {a})
  ELSE s
  ENDIF

secureAccess?(a: Accesses): bool =
  IF a'am = read
  THEN SLB(a'sub) ≤ OLB(a'ob)
  ELSE SLB(a'sub) ≥ OLB(a'ob)
  ENDIF

secureState?(s: State): bool =
  ∀ (a: Accesses):
    s(a) ⇒ secureAccess?(a)

secureEmpty: LEMMA

```

secureState?(emptyset)

transformSecure: LEMMA

$\forall (a: \text{Accesses}, s: \text{State}):$
secureState?(s) \Rightarrow
secureState?(transform
 (a, s))

END biba

The Chinese Wall Security Model Specification

Below is a specification of the Chinese Wall security model. It does not look quite the same as the specification in the paper, but one can prove the basic theorems from the paper with the specification provided.

I have provided 4 LEMMA's and one THEOREM. These mimic the equivalent statements in the Brewer/Nash paper. They all follow directly from the definitions **SECURE_ELEMENT** and **SECURESTATE**. You may notice that I have used **XOR** instead of the operator **OR** used in the paper. It turns out that **TH1** isn't true if one uses **OR**.

The **XOR** operator needs some special handling in PVS. It needs to be expanded. Suppose one has

$$(x = y) \text{ XOR } (u = v)$$

as part of either a hypothesis or conclusion. When one *expands* **XOR**, one obtains

$$(x = y) \neq (u = v)$$

where the symbol \neq is the “not-equal” sign in PVS. Either of the statements means that exactly one of $(x = y)$ and $(u = v)$ is true and the other is false, that is they both can't be true and they both can't be false. To simplify handling of the \neq one can use the command *bddsimp* (binary decision diagram and simplification). The command takes no arguments. It separates the inequality up into cases.

Other than these hints, be prepared to look at substitutions using the *replace* command and carefully manage your *instantiations*.

What you should prove

*You should provide proofs only of **AX1**, **emptySecure** and **TH1**.* You should hand in sufficient evidence to show that you have completed the proofs. You should not use *grind* or similar commands.

The LEMMAS **AX2** and **AddToEmpty** are provided purely for illustration purposes. Each takes about 50 steps and doesn't succumb to the *grind* command. If you are interested, you might want to try them.

You can find the specification below and on the Web on my home pages.

```
chinesewall: THEORY
BEGIN
```

```
Subjects: TYPE+
```

```
Objects: TYPE+
```

```
ConflictDomains: TYPE+
```

Companies: TYPE+

Label: TYPE+ =
[# cd: ConflictDomains,
cp: Companies #]

LB: [Objects \rightarrow Label]

NType: TYPE+ =
[# sb: Subjects, ob: Objects #]

$X(\text{ob: Objects}): \text{ConflictDomains} =$
LB(ob)'cd

$Y(\text{ob: Objects}): \text{Companies} =$
LB(ob)'cp

NState: TYPE = setof [NType]

SECURE_ELEMENT($N: \text{NState}, \text{nx}: \text{NType}$): bool =
 $\forall (\text{na}: \text{NType}):$
 $N(\text{na}) \wedge \text{sb}(\text{na}) = \text{sb}(\text{nx}) \Rightarrow$
 $(\neg (X(\text{ob}(\text{na})) = X(\text{ob}(\text{nx})))) \text{ XOR}$
 $(Y(\text{ob}(\text{na})) =$
 $Y(\text{ob}(\text{nx}))))$

SECURESTATE($N: \text{NState}$): bool =
 $\forall (\text{nx}: \text{NType}):$
 $N(\text{nx}) \Rightarrow$
SECURE_ELEMENT(N, nx)

EMPTYSubject($N: \text{NState}, s: \text{Subjects}$): bool =
 $\forall (\text{na}: \text{NType}):$
 $\text{sb}(\text{na}) = s \Rightarrow$
 $\neg (\text{na} \in N)$

AX1: LEMMA

$\forall (N: \text{NState}, \text{na}, \text{nb}: \text{NType}):$
SECURESTATE(N) \wedge
 $\text{sb}(\text{na}) = \text{sb}(\text{nb}) \wedge$
 $N(\text{na}) \wedge N(\text{nb}) \wedge Y(\text{ob}(\text{na})) = Y(\text{ob}(\text{nb}))$
 $\Rightarrow X(\text{ob}(\text{na})) = X(\text{ob}(\text{nb}))$

emptySecure: LEMMA
 $\forall (N: NState):$
 $N = \text{emptyset} \Rightarrow$
 $\text{SECURESTATE}(N)$

TH1: THEOREM
 $\forall (N: NState, nsx, nsy: NType):$
 $\text{SECURESTATE}(N) \wedge N(nsx) \wedge N(nsy) \wedge sb(nsx) = sb(nsy)$
 \Rightarrow
 $Y(\text{ob}(nsx)) = Y(\text{ob}(nsy)) \vee$
 $\neg (X(\text{ob}(nsx)) =$
 $\quad X(\text{ob}(nsy)))$

AddToEMPTYSubjectSecure: LEMMA
 $\forall (N: NState, ns: NType):$
 $\text{SECURESTATE}(N) \wedge \text{EMPTYSubject}(N, sb(ns)) \Rightarrow$
 $\text{SECURESTATE}((N \cup \{ns\}))$

AX2: LEMMA
 $\forall (N: NState, nx: NType):$
 $\text{SECURESTATE}(N) \wedge$
 $(\forall (nb: NType):$
 $\quad N(nb) \wedge sb(nb) = sb(nx) \Rightarrow$
 $\quad (\neg (X(\text{ob}(nb)) = X(\text{ob}(nx))) \text{ XOR}$
 $\quad \quad Y(\text{ob}(nb)) = Y(\text{ob}(nx))))$
 \Rightarrow
 $\text{SECURESTATE}((N \cup \{nx\}))$

END chineseWall