OLLSCOIL NA hÉIREANN
THE NATIONAL UNIVERSITY OF IRELAND, CORK

COLÁISTE NA hOLLSCOILE, CORCAIGH
UNIVERSITY COLLEGE, CORK

Summer Examination 2012

CS4615: Computer Systems Security

Professor Ian Gent,
Professor J. Bowen,
Dr. S.N. Foley

Answer *all* questions

1.5 Hours

1. a) Discuss effectiveness of code-signing in preventing malicious software. (*6 marks*)

A *General discussion. Student should recognize that while it does tie the code to a key, there is the problem of how to determine whether the key is trustworthy and whether it has ever signed malicious code in the past, even by accident.* **A**

b) Explain how a SYN-flood can result in a denial of service attack. (*6 marks*)

A *Answer from notes. Looking for explanation of 3-way handshake, the attack and understanding of the resource limitations for half-open connections.* **A**

c) Which of the following C programs are vulnerable to a stack smashing attack? Outline how the selected program enables the attack to be carried out. (*6 marks*)

```
  void main1(int argc, char* argv[]){
    char buffer[6];
    strcpy(buffer,argv[0]);
}/*main1*/
```

```
  void main2(int argc, char* argv[]){
    char buffer[6];
    strcpy(buffer,"long text");
}/*main2*/
```

A *Looking here for careful description of how one can smash the stack with a buffer overflow and get the program to execute your own code. Need to consider the case when the smashed program is running as a privileged operation.*

`main1` *can be smashed.* `main2` *while has an overflow, cannot be smashed.* **A**

d) Give an example of how the Access Matrix Model can be interpreted in terms of *access control lists* and in terms of *capabilities.* (*6 marks*)

A *Model is a matrix of permissions with subjects as rows and objects as columns. capability is a row: a subject with a list of permissions that it has for objects; ACL is a column: an object with a list of subjects who may access it according to given permissions.* **A**

e) Describe the operation of the Setuid (suid) permission in Unix. (*6 marks*)

A *Straightforward answer based on lecture notes* **A**

(*Total 30 marks*)

---

2. a) The java class `CreditCard` is used to manage credit card details stored in a user's workstation file `~/.mycreditcard`. The `CreditCard` operation

```
public String details(){ ... }
```

checks (via pop-up dialog box) with the user whether credit card details (in `.mycreditcard`) should be returned; if not, a null string is returned. When the user shops at `www.buy.com`, she uses the `www.buy.com/Checkout.jar` to pay for items selected; this applet invokes `CreditCard.details()` to obtain customer credit card details.

Outline how the Java security manager is used to ensure that the downloaded `Checkout` applet cannot access the user's credit card details without the permission of the user. Your answer should include: a suitable Java security policy and an explanation of how a new Java permission is declared and used by `details`, and whether `details` should be treated as a privileged operation (*15 marks*)

A **A** I don't expect students to generate complete syntax/working code with all the details. But what I am looking for is an explanation of the components and how they fit together.

Declaration of a new permission `CreditCardDetailsPermission()`. Making sure that before the security critical action `CreditCard.details` consults the security manager:

```
SecurityManager sm = System.getSecurityManager();
if (sm!=null) {
  sm.checkPermission(new CreditCardDetailsPermission());
}
```

and then executes privileged block to manipulate the `.mycreditcard` file

```
AccessController.doPrivileged(new ProviligedExceptionAction()
  public Object run() throws IOException {
      access the file, etc.
      ....}
)
```

In the policy file, we do not give `buy.com` access to the credit card file:

```
grant signedby "buy-com" {
  permission com. .. .CreditCardDetailsPermission,
  signed by "Simon"
}

grant signedby "Simon" {
  permission java.io.FilePermission
    "$(user.home)${/}.mycreditcard", "read, write";
     permission com. .. .CreditCardDetailsPermission,
     signedby "Simon"
}
```

b) Suppose that `buy.com` also offers a `checkout` application that a customer can manually download and execute. This application invokes a local application `creditCardDetails`. While these applications are implemented in C, and run natively on the customer's SELinux workstation, their behavior is similar to their Java counterparts in Question 2a. Outline how the Domain and Type Enforcement policy for this workstation might be configured to protect the user's credit card details. Your answer should include sample domains, types and domain definition table. (*10 marks*)

A *We have Domains* $= \{Shell, BuyApp, Sensitive\}$ *and Types* $= \{os, buy, creditCard\}$. *The user's shell (type os) enters domain Shell, the checkout application (type buy) enters domain BuyApp, the credit card application (type creditCard) enters domain Sensitive and the* `creditCard` *file has type creditCard. The domain definition table is*

	os	*buy*	*creditCard*
Shell	*RWX*	*X*	
BuyApp		*RW*	*X*
Sensitive	*R*	*W*	*RW*

A

(*Total 25 marks*)

---

3. a) A multilevel secure system offers a document indexing system with the operations:

- assign(n,p,s): give the document (file) located at path p the name n.
- view(n,s): view the contents of the document with name n.

where s is the subject requesting the operation. Note that document names are unique across the system and are in addition to the name (path) given to the file containing the document. A table is used to store the name-path relationship, for example:

Name	Path
ExamPaper	/home/store/a
Attendance	/home/store/b
LectureNotes	/home/store/c

i. Sketch suitable algorithms that describe the behavior of the above operations taking care to ensure that multilevel security is preserved. (*10 marks*)

A *Using the Bell-LaPadula axioms, let* $\leq$ *denote the sensitivity ordering between security levels and assume that every file and subject has a corresponding security level. Need to recognize that creating a document-path relationship is in itself a piece of information that is at the level of the subject s carrying out the insertion and we must therefore extend the table to include a security-level to avoid a covert channel.*

```
assign(n,p,s){
   if n exists then return null;
   if security-level(p) ≤ security-level(s) then
   add (n,p,s) to table
}
view(n,s){
   retrieve path with name n from table;
   if no entry found then return null;
   if security-level(p) ≤ security-level(
}
```

A

ii. Given that document names are unique in a table, explain how a Trojan-Horse running at top-secret could establish a covert-channel and signal one bit of information to a subject operating at unclassified. (*5 marks*)

A *A covert channel exists because a low-level user can test for the existence of a high level name-path table entry by attempting to assign that name to another document. For example, suppose a Trojan-Horse operating at top-secret wishes to signal a bit of (top-secret) information to a secret subject. The Trojan-Horse inserts a new document with agreed name* 1111 *meaning* 1*, or does no insertion meaning* 0*. At a later (agreed) point in time the unclassified collaborator then attempts to assign a document with name* 1111*. If the insertion fails the unclassified user deduces* 1*, or* 0 *otherwise.*

The covert channel can be removed by allowing duplicate names for different security levels. In the example above, the unclassified user would be allowed insert a secret tuple with key 1111*.* **A**

b) A conventional (non-MLS) linux server currently hosts an email-service plus a service that is similar to that in Question 3a but is used only for top-secret documents.

The owner is concerned about Trojan Horse attack and estimates that, in terms of lost of reputation, among other things, unauthorized leakage of top-secret document information would cost $500,000, while unauthorized access to email would cost $10,000. The

probability of such an attack on a conventional linux system that hosts both services is estimated to be 0.1. If the services were to be hosted on an MLS system then the probability of attack would reduce to 0.0001. Assume that an MLS system costs $5,000, while a standard linux server costs $500 and that their operational costs are zero.

Use this information to carry out a *Risk Assessment* and advise the company on how best to configure the system(s) and mitigate the risk of Trojan Horse attack. (*10 marks*)

A *Looking for ALE-style calculations along with some insight on the best deployment. The current risk can be calculated as* $Risk_{linux-doc} = 0.01 \times 500,000$ *and* $Risk_{linux-email} = 0.01 \times 10,000$ *and thus* $Risk_{linux} = Risk_{linux-doc} + Risk_{linux-email} = 5,100$. *Moving to a MLS system would give* $Risk_{mls-doc} = 0.0001 \times 500,000$ *and* $Risk_{mls-email} = 0.0001 \times 10,000$ *and thus* $Risk_{mls} = Risk_{mls-doc} + Risk_{mls-email} = 51$. *In the first year, even accounting for the upfront cost of the MLS server, the savings would be* $Risk_{linux} - (Risk_{mls} + 5000) = 49$, *with a clear risk saving in subsequent years. In this case its assumed that the partial order is modified to include a new security level* email *that is disjoint from the (*top-secret*) security level for documents.*

A shrewd student could suggest that the owner could user the existing linux server for email and migrate the (single-level) document server to a second and much cheaper linux server that is not Internet facing. It can be argued the partitioning will reduce the probability of compromise, and lead to an even cheaper solution. **A**

(*Total 25 marks*)
