IS2150/TEL2810 Introduction to Security

Homework 2

Total Points: 50

1) Exercise on Propositional/Predicate logic [25 Points]

(a) Prove that $A \oplus B \Leftrightarrow (A \wedge \neg B) \vee (\neg A \wedge B)$ (you can use the truth table) (5 points)

A	B	¬A	¬B	A∧¬B	¬A∧B	(A∧¬B) ∨ (¬A∧B)	A ⊕ B
T	T	F	F	F	F	F	F
T	F	F	T	T	F	T	T
F	T	T	F	F	T	T	T
F	F	T	T	F	F	F	F
						↑	↑

(b) Express the following sentences in propositional/ first order logic. Be sure to define all propositional components (e.g., predicate function, constants, and variables).

i) If it does not rain we will go to the Steeler's game. (3 points)

R = it rains
S = go to Steelers game
$\neg R \rightarrow S$

ii) If a subject has Secret clearance then he/she is allowed to write to Secret and Top Secret files (3 points)

SecretSubject(x) = subject x has secret clearance
TopSecretFile(f) = file f is top secret
SecretFile(f) = file f is secret
allowWrite(x, y) = subject x is allowed to write to file y

$\forall x \, \forall f \, [\text{SecretSubject}(x) \wedge ((\text{SecretFile}(f) \vee \text{TopSecretFile}(f)) \rightarrow \text{canWrite}(x,f))]$

iii) A person can approve a check or cash it but cannot do both. (3 points)

canApprove(x, c) = x can approve check c
canCash(x, c) = x can cash check c

$\forall x, \forall c \, [(\text{canApprove}(x, c) \wedge \neg \text{canCash}(x, c)) \vee (\text{canCash}(x, c) \wedge \neg \text{canApprove}(x, c))]$

iv) A directory is older than the directories and the files that it contains. (4 points)

isOlder(x, y) = x is older than y
isDirectory(x) = x is a directory
isFile(x) = x is a file
Contains(x, y) = x contains y

$\forall$d, $\forall$x [isDirectory(d) $\wedge$ (isFile(x) $\vee$ isDirectory(x)) $\wedge$ Contains(d, x) $\rightarrow$ isOlder(d, x)]

(c) Prove by induction the following statements: [20 Points]

$1^3 + 2^3 + 3^3 + ... + n^3 = [n(n+1)/2]^2$

Base Case: For n=1, the statement hodlds:

$1 = 1^3 = [1(1+1)/2]^2 = 1$

Induction Hypothesis:

Assume that $1^3 + 2^3 + 3^3 + ... + k^3 = [k(k+1)/2]^2$ holds. Show that $1^3 + 2^3 + 3^3 + ... + k^3 + (k+1)^3 = [(k+)(k+2)/2]^2$

Mathematical Induction:

Using the hypothesis: (replacing n with (k+1) in the formula)

$S(k+1) = [(k+1)(k+2)/2]^2$ ---------- (1)

Using mathematical induction:

$S(k+1) = S(k) + (k+1)^3$

Substituting from the hypothesis,

$S(k+1) = [k(k+1)/2]^2 + (k+1)^3$

Factorization yields,

$S(k+1) = [k(k+1)/2]^2 + (k+1)^3$

$= [(k^2+k)/2]^2 + (k+1)^3 = [(k^4 + k^2 + 2k^3)/4] + [4(k+1)^3/4]$

$= [(k^4 + 6k^3 + 13k^2 + 12k + 4)/4]$

$= [(k+1)(k+2)/2]^2$ ----------(2)

Comparing (1) and (2), this is proved that the mathematical induction holds for k+1.

Hence, S(n): $1^3 + 2^3 + 3^3 + ... + n^3 = [n(n+1)/2]^2$ is true for all n>=0.

2) Do the followings:

(i) Describe and differentiate between the mechanisms related to: *Setuid program in Unix* and *Impersonation in Windows*

Setuid program is used when a process executes a file in Unix. The EUID of the process [temporarily] changed to UID of the file owner if setuid bit is set on. Thus, the common users get higher privileges and can do anything that the owner is allowed to do.

Impersonation in Windows is associated to threads. By passing impersonation token to server, a client is temporarily allowed to adopt a different security context of another user. There are four impersonation levels of server: anonymous, in which token has no information about the client; identification, in which server obtains the SIDs of client and client's privileges, but server can't impersonate the client; impersonation, in which server identifies and impersonates the client; an delegation, in which server can impersonate client on local and remote systems.

[Lidan Hiang]

(ii) Note that the following two resolution rules are used in Windows. Explain how given a *security descriptor* and an *access token*, these resolution techniques are used.

(1) Positive permissions are additive

When given a security descriptor and an access token, the security descriptor is scanned (in sequence) for permissions that match up to the access token (checking whether or not rights are in the security descriptor entry). Let's say there is a user named Dave, and he belongs to two groups: administrators and readers. When it checks for Dave's rights, if the administrator group is granted write permissions and the reader group granted read permissions-Dave will be given both write and read permissions. This is an example of how positive permissions are additive.

(2) Negative permission (deny access) takes priority

[Security descriptor is scanned in same way as mentioned above] Let's say there is a user named Dave, and he belongs to two groups: administrators and readers. When it checks for Dave's rights, if the administrator group is denied write permission and the reader group is granted write permission-Dave will be denied write permission (even though he was granted it in the reader group). This is an example of how negative permissions (deny access) are given higher priority.

[Steven Madara]

(iii) Assume that when a file is created and before the umask value has been applied, the permission bits are 0626 (in class we assumed 0777). What will be the permission setting for the new files when the following umask values are applied [5 Points]

Permission bit (0626) = 110 010 110

	umask	umask bits [u]	Bitwise negation of umask [$\neg$u]	Permission bits (0626) [p]	Assigned bitwise permissions [p $\wedge \neg$u]	Assigned permission
(i)	032	000 011 010	111 100 101	110 010 110	110 000 100	0604
(ii)	031	000 011 001	111 100 110	110 010 110	110 000 110	0606
(iii)	051	000 101 001	111 010 110	110 010 110	110 010 110	0626