6.033: Computer Systems Engineering

Spring 2012

5/23

Home / News
Schedule
Submissions

General Information
Staff List
Recitations
TA Office Hours

Discussion / feedback
FAQ
Class Notes Errata
Excellent Writing Examples

2011 Home

Preparation for Quiz 3

While Quiz 3 is during finals week, it will only explicitly cover the final 1/3 of the course. Of course, the cumulative nature of this course means that terms and concepts from the first 2/3 will be used freely, but the questions will focus primarily on the material covered from recitation 18 onwards. Quiz 3 will be 90 minutes, not 50 like the first two.

The quiz will cover material from recitation 18 through recitation 26. The quiz will be open book, open laptop, no network. That means you can bring along any printed, written or electronic materials that you think might be useful but you cannot use any network connection. new

You can find old quiz questions to practice on in the Problem Sets section of the class notes. Although there has been a little rearrangement of material over the years, you should be able to answer most questions relevant to the subjects we have covered.

Old quizzes:

- 2011 - Quiz 3, Quiz 3 Solutions
- 2010 - Quiz 3, Quiz 3 Solutions
- 2009 - Quiz 3, Quiz 3 Solutions
- 2008 - Quiz 3, Quiz 3 Solutions
- 2007 - Quiz 3 Solutions
- 2006 - Quiz 3 Solutions
- 2005 - Quiz 3 Solutions
- 2004 - Quiz 3 Solutions
- 2003 - Quiz 3, Quiz 3 Solutions

Questions or comments regarding 6.033? Send e-mail to the 6.033 staff at 6.033-staff@mit.edu or to the 6.033 TAs at 6.033-tas@mit.edu.

Top // 6.033 home //

6.033

5/23

Final Quiz Review

Beyond Stack Smashing Paper

Stack Smashing – modifying return address on stack point to attacker code in stack buffer

Arc injection – return-into-libc

Control transfer to code already in memory

pointer subterfuge – attack function pointers

function pointer – pointers whose ~~stored~~ value is used as an address in a function call

heap smashing – exploitation of buffer overruns in dynamically allocated memory (not on stack)

(2)

Porcupine

Availibility

Just add another node scaling

Same or slightly worse performance

Eventual consistency

---

mail stored in mail fragments
- distributed across nodes
- list stored as softstate - on 1 node

User profile db = hard state
- partitioned + replicated across nodes

User map - soft state but everyone has a copy

Cluster map - every node knows about every other node

(3)

Mail delivered to preferable node (I belive who has other fragments)
- Lor a not busy one

proxy that sends it to a node
- retrieves from all the nodes via P2P
  - who have mail
    - discards the mail fragment list

Can replicate arbitrally
- So never goes down
- no node is perm responsible

but want to limit spread
- So performance isn't that bad

Nodes converge on consistent memory state
- Three Rand Membership Protocol (TRM)

Cord uniformly distributes users

(4)

Rebilding soft state

- (didn't read)
- Constant

Replication

- update anywhere
- eventual consistency
- total update
- no locks

log

- never blocks

Replication manager

- update → list of remaining nodes

Had Consistency, Availability performance kinda all 3

Rec!

(5)

Witty Telescope Paper

Watch unused IP blocks

tabulate packet rates + victim populations

12,000 + hosts in 75 min

- Sends it-self to 20,000 people
- Opens a random disk
- If succeeds, reseed PRNG
- Repeat

They inspected the PRNG

Saw a bunch of things

Silly Concat top 16 bits

$$X_{i}[0...15] \text{ (concat) } X_{i+1}[0...15]$$

So they can recover it by reverse engineering the PRNG

do this for uniform randomness

but they don't need

question

42 views

Version vectors vs vector timestamps

So from my understanding, a version vector contains the number of changes that have been made to the variable by each user to bring the variable to its current state, while a vector timestamp contains timestamps showing the time of the last change from each user.

also vector clocks

Why would you want to use vector timestamps over version vectors? It seems like using them provides the same ability, but has an additional requirement in that the users' clocks must be synced. #final

edit | follow 4 | like 0 | 2 days ago by Anonymous | 1 edit

the students' answer, *where students collectively construct a single answer*

Click to start off the students' answer

Should not print

the instructors' answer, *where instructors collectively construct a single answer*

Vector timestamps are helpful if you are using a system that does not increment some counter for you on each operation. For example, the standard Linux ext3 file system isn't going to increment version vectors for you, but you can still take the local mtime of a file and put it into a vector timestamp.

if building on an old system

As to the second part of your question, clocks don't need to be synchronized for vector timestamps -- that's the whole point of having a vector.

like 0 | 2 days ago by Nickolai Zeldovich | 1 edit

followup discussions, *for lingering questions and comments*

(6)

Some IPs that can't be scanned

Uniformly distributed over IPV4

(but they never say which)

The same exact # occur twice!

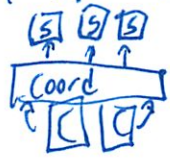
Was on initial hit list

Paxos

other ~~AAA~~ talk look at majority say when every all

Replicate state machine

- provides same input in same order to replica



- Coording rides
- but 1 point of failure

Goal fault tolerant consensus

Get machines to agree on stuff

Finish if $< \frac{n}{2}$ failures

Nodes fail-stop

question

59 views

Paxos commit point?

I'm still unclear about the commit point for paxos. Is it when the acceptors write information to disk (Na, Va) or when the OK is sent out by the acceptors to the proposer?

Thanks! #quiz3_final

edit | follow 4 | like 0

1 day ago by Anonymous | 1 edit

the students' answer, *where students collectively construct a single answer*

Click to start off the students' answer

the instructors' answer, *where instructors collectively construct a single answer*

One way to understand this would be to think about what happens if you were to crash just after some specific point. What will the system do?

Once a majority of acceptors write the same Va to disk, then even if everyone crashes, the only possible value that the system can agree on from that point forward is Va. And, conversely, if just below a majority of acceptors write the same Va to disk, it's still possible to agree on something other than Va. So, that is the commit point.

like 0

1 day ago by Nickolai Zeldovich | 2 edits

followup discussions, *for lingering questions and comments*

question

38 views

2011 Quiz 3 Question 14

#quiz3

Why is {} not considered a valid return value after A and B are written in order, leading to v1:{A}, v2:{A, B}, with v1 < v2, and then the delete goes to v1, leading to v2:{A, B}, v3:{}

edit | follow 4 | like 0 | 7 hours ago by Yuchen Feng | 1 edit

the students' answer, *where students collectively construct a single answer*

"In case of doubt the cart application reconciles by keeping any items whose status is in doubt." So when reconciling v2 and v3, it keeps both. ? Ohhhhh

~ An instructor (Nickolai Zeldovich) endorsed this students' response ~

edit | like 1 | more | 6 hours ago by Patrick Hurst | 1 edit

followup discussions, *for lingering questions and comments*

Resolved Unresolved

Yuchen Feng (5 hours ago) - i meant why is a reconciliation necessary and why is {} considered in doubt in the first place?

Patrick Hurst (1 minute ago) - Because v2 and v3 are irreconcilable version numbers. ah

Write a reply...

Resolved Unresolved

Anonymous (3 minutes ago) - Where does v3 come in? I thought the "third version" was {A} which is equal to v1. I don't see how the empty set comes in anywhere :(

Write a reply... Yeah - be interesting to see

question

24 views

spring 2011 q14

I didn't understand clearly why choice C is correct. As far as I can understand from the explanation, it seems that v1, v2 will be on different servers. But won't they both reconcile to the correct set eventually and thus everyone will have {B}? i.e. won't the set {A,B} be propagated to all nodes, leading to {B} eventually.. #final

edit | follow 4 | like 0 | 3 hours ago by Anonymous | 1 edit

the students' answer, *where students collectively construct a single answer*

see the answer below.

edit | like 0 | more | 1 hour ago by Anonymous | 1 edit

followup discussions, *for lingering questions and comments*

Resolved Unresolved

Anonymous (1 hour ago) - which answer?

Nickolai Zeldovich (Instructor) (1 hour ago) - There's already been a fair amount of discussion of this question on Piazza.

Anonymous (53 minutes ago) - I have seen the older posts. However, it's still not clear to me why C will be correct. I think I am getting confused between correctness and eventual consistency. If we are assuming eventual consistency, why aren't we guaranteed correctness, given that we are assuming none of the nodes crash permanently, as was clarified in one of the previous questions. May be an example might help.

Nickolai Zeldovich (Instructor) (47 minutes ago) - What do you mean by "correctness"?

The answer explains how you might get {A,B}. First, the user adds A, which goes to server 1 and creates v1={A}. Then, v1 is replicated to server 2. Then, the user adds B, which goes to server 2, creating v2={A,B} whose vector clock is greater than v1. Then, the user deletes A, which goes to server 1, and creates v3={} whose vector clock is concurrent with v2 (neither less than nor greater than). Then, the servers reconcile with each other, and decide they have two conflicting versions v2 and v3. Finally, when the user reads the shopping cart, the application gets both v2 and v3 back, and reconciles them by taking the union.

Anonymous (33 minutes ago) - Sorry for not being clear. By correctness I meant loss of updates. So with eventual consistency it is possible to lose some updates..right?
Thanks for explaining it!

Anonymous (31 minutes ago) - Or rather eventual consistency doesn't guarantee the end result to be same as if all the updates were done on a single machine..

Nickolai Zeldovich (Instructor) (24 minutes ago) - You are correct that eventual consistency

differs from execution on one machine.

Write a reply...

(7)

Proposers + Acceptors for each node

Proposer have idea what want consens on

Acceptors are voters who can decide who to win

Some have round #s

Example where works

1. Pick random proposal #
2. Send prepare message to each acceptor?
   if all return (+)
3. Then accept to all
   if all return (+)
4. Then decide

majority for read + write

~~Client~~ app contacts a primary who talks to the others (not app directly)

Piazza: Understand what Paxos solves - not exactly details

(8)

Unison

File synchronizer

- fast
- file name semantics diff
- range of failure scenarios

How to combine replicas

Could single-replica (ie no changes)

Or could sync

#1 goal: safety

if concurrent mod → do nothing

trace vs state based
(easier to hook)

minimize transfer
so update detection on host

9

On each ~~file~~ system maintain an archive
- State of file since last sync

So *you* know what you have changed

Recitation puts this nicely

Maintain list of machines you sync w/ + archive file w/ mod times + hashes from last sync

1st time Send it list of fingerprints + files

nth time Send list of changes + new files
n ≥ 1 ↳ task list

---

absent = special type

Conflicting
1. updated in 1 replica
2. if a any descendents updated in other ↳ like directory
3. Contents of replica are not identical

(10)

Would be robust if OSes did atomic file writes
(rechecks if user changed)

- mod times — less safe + fast
  (+ inode)
- hash — safe + slow

Trusting Trust

(This was confusing)
Recread → still don't get
Basically recompile — can hide bugs

Crypto systems fail

Mostly implementation flaws
Often not public
Banks liable for fraud in US
But not in Britain

(1)

Banks make mistakes

Emp steal
(looser pass)

Managers are ~~stalling~~ stupid

Print full acct #'s on slips

ATMs' talk w/ home not encrypted

Programming errors

Mail insecure

Test transaction
lol !!!

False terminals

Pin software
– very few words

Same pin
Since mailers secure → there sw

(12)

Calc pin from acct # w/ DES

- but secret key must be secret
  but must be used
  Secrity module (possibly)

Trap doors to test

Or manager doesn't want to enter keys

Feature creep

People leave

Or consultants don't know what they are doing
(This is prob still common)

Hints

(Don't think they will ask on this)

(13)

Object Store

(is covered!)

OO - DB System

- Storage
- transaction management
- distributed access
- assaciate queries
- Unified programming interface for perm and temp
  - no translation code, manipulate data in C
  - For manipulating complicated structures
    - LGIS, CAD
  - just lable "persistence"
  - temporal locality
    - like virtual memory
      - caches
      - faults
      - shared or exclusive
  - versions
    - check at
    - change
    - check back in

Pointers relocated (updated) when moved in memory

Segments transfered 1 at a time or en masse

Collections + cursors

- 2 types
- picked automatically
- can run methods on

→ auto locks

→ stores in same format on disk as RAM

5/23 practice

Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.033 Computer Systems Engineering: Spring 2011

Quiz 3

There are 15 questions and 10 pages in this quiz booklet. Answer each question according to the instructions given. You have **90 minutes** to answer the questions.

Some questions are harder than others and some questions earn more points than others—you may want to skim all questions before starting.

For true/false questions, you will receive 0 points for no answer, and negative points for an incorrect answer. Do not guess; if you are unsure about your answer, consult your notes. The total score for each numbered question (1 through 15) will be at least 0 (i.e., those scores cannot go negative).

If you find a question ambiguous, be sure to write down any assumptions you make. **Be neat and legible.** If we can't understand your answer, we can't give you credit!

Write your name in the space below. Write your initials at the bottom of each page.

THIS IS AN OPEN BOOK, OPEN NOTES, OPEN LAPTOP QUIZ, BUT DON'T USE YOUR LAPTOP FOR COMMUNICATION WITH OTHERS.

CIRCLE your recitation section number:

10:00	1. Lampson/Pesterev	
11:00	2. Lampson/Pesterev	3. Ports/Mutiso
12:00		4. Ports/Mutiso
1:00	5. Katabi/Raza	6. Strauss/Narula
2:00	7. Katabi/Raza	8. Strauss/Narula

Do not write in the boxes below

1-4 (xx/24)	5-7 (xx/19)	8-10 (xx/18)	11-12 (xx/12)	13-15 (xx/27)	Total (xx/100)

Name:

Red = ans key

I Reading Questions

out of scope

1. [8 points]: Based on the paper "The Recovery Manager of the System R Database Manager", which of the following statements are correct?

(Circle True or False for each choice.)

A. True / False As part of recovery from a crash, System R restores the current version of all shadowed files to their shadow copy.

B. True / False The shadow copy of a file reflects only the effects of committed transactions.

C. True / False System R will undo the effects of certain transactions during recovery.

D. True / False System R will redo the effects of certain transactions during recovery.

2. [6 points]: Based on the paper "Beyond Stack Smashing: Recent Advances in Exploiting Buffer Overruns", which of the following statements are correct?

(Circle True or False for each choice.)

A. True / False The basic form of attack described in the paper (stack smashing) allows an attacker to execute arbitrary code by overwriting far enough beyond the end of an application buffer to also overwrite the stack pointer.

overwrites return address - not sp

B. True / False Even if a system guards against buffer overruns by making the stack non-executable, and uses stack cookies to guard against arc injection / return-into-libc attacks, the system could still be vulnerable to attacks which modify function pointers.

C. True / False A heap buffer overrun writes beyond the end of a dynamically-allocated object (e.g., created with `new` in C++ or `malloc()` in C), instead of a procedure's local variables.

generally
if user allocated
memory

pointers that whose value is
used as an address
in function call

pay attention to detail!

Initials:

~~This is only~~ They won't ask the exact qu again... but reminds me of the paper

3. [6 points]: Which of the following statements is true about the Porcupine distributed email system described by Saito, Bershad and Levy?

(Circle True or False for each choice.) goal here is? Seems failure resistance mostly

A. True / False In Porcupine, soft state such as the user map is replicated on every node for fault-tolerance. Don't think so - split out - one node

B. True / False In order to balance the load across the cluster, when a new message arrives Porcupine always picks a lightly loaded node at random and adds the message to the user's mailbox fragment on that node. L hidden in paper: affinity based

4. [4 points]: Porcupine's scheme for maintaining cluster membership chooses a new coordinator whenever anyone detects an event: a node going up or down. The coordinator broadcasts a ping and gets a response from every node that is up. The total amount of work done in a cluster of n nodes, each time some node goes up or down, is proportional to which of the following?

(Circle the BEST answer)

- n
- $n \log n$
- n^2

#events = lin w/ # of node

~~#~~ each node each event n) n^2

Availability

Just add another node scaling

Same or slightly worse peforance

Eventual consistency

Oh whole system

Initials:

II Lectures

5. [4 points]: According to Prof. Abelson's lecture, the "Good Samaritan Provision" of the Communications Decency Act provided protection to which of the following parties?

(Circle True or False for each choice.)

A. True / False CMU so that it can study the use of pornographic information on the Internet without being liable.

B. True / False AOL in the case against Ken Zeran, because AOL merely distributed information but was not responsible for the content.

C. True / False Prodigy in the case versus Stratton-Oakmont because Prodigy filtered offensive content.

but eventually overturned predated CDA

D. True / False The nastiest place on earth because it merely distributed content that others generated.

predated as well

6. [8 points]: Consider the mechanisms for storing and checking user's passwords, as described in lecture.

(Circle True or False for each choice.)

A. True / False Rather than storing user's login passwords directly, Unix stores the result of applying a hash function to the user's password. One reason for this approach is so that users' plaintext passwords are not revealed if an attacker manages to read the password database.

B. True / False Unix combines *salts* with a user's password before sending them through the hash function to enact a policy that requires users to periodically revise their passwords.

not why salt

C. True / False Let $H()$ be a hash function. If given a value p, it is easy to find a value P such that $H(P)$ equals p, then $H()$ is a poor choice of hash function to use as part of a password checking system.

If that was the case - crappy system

D. True / False The study of password use described in lecture found that different people almost never choose the same passwords.

I'm in Scope

7. [7 points]: Consider the Same Origin Policy (SOP) used for browser security.

(Circle True or False for each choice.)

port protocol host

A. True / False Javascript code from the page at `http://web.mit.edu/6.033` can access the DOM of another page at `http://web.mit.edu/6.041` loaded in the same browser. Assume the second page runs no Javascript code.

B. True / False Javascript code from the page at `http://web.mit.edu/6.033` can access the DOM of another page at `http://courses.csail.mit.edu/6.033` loaded in the same browser. Assume the second page runs no Javascript code.

not in diff windows

Initials:

C. True / False Javascript code from the page at `http://web.mit.edu/6.033` can access the DOM of another page at `http://www.google.com` loaded in the same browser. Assume the second page runs no Javascript code.

D. True / False Suppose the 6.033 staff copies the Javascript code for an advertisement from `attacker.com` into the page at `http://web.mit.edu/6.033` (for example, by including a tag like <SCRIPT SRC="`http://attacker.com/ad.js`">). The Same-Origin Policy will allow that Javascript code from attacker.com to access the DOM of the 6.033 page (`http://web.mit.edu/6.033`).

E. True / False The Same-Origin Policy allows Javascript code from `site1.com` to access the cookie of `site2.com`, as long as the two sites are concurrently loaded in two different tabs in the same window.

Initials:

III Worm Trouble

It is May 2011 and there is a new worm, *nitty*, released on the Internet. Security experts quickly analyze the code corresponding to Nitty to be as follows:

```
1  int X = 0; /* X is a global variable */
2
3  int rand() {
4    /* Note that 32-bit integers obviate the need for a modulus operation here. */
5    /* A = 214013, B = 2531011. */
6    X = X * A + B;
7    return X;
8  }
9
10 srand(seed) {
11   X = seed;
12 }
13
14 main() {
15   srand(get_tick_count());
16   while (1) {        /* infinite loop */
17     for (int i = 0; i < 20,000; ++i) {
18       dest_ip = rand();
19       dest_port = rand() [0:15];      /* low 16 bits */
20       packetsize = 1000;              /* 1000 bytes */
21       packetcontents = top_of_stack;
22       sendto();
23     }
24     int rand_temp = rand();
25     if (open(physicaldisk, rand_temp[13:15]))
26       overwrite_block(rand_temp[0:14] || 0x4e20);
27   }
28 }
```

You learned about network telescopes in 6.033 and decided to see if you can apply the concepts to infer information about this new worm. You set up a router at MIT and configure it to listen on two unused portions of the MIT IP address space corresponding to 18.32.0.0/11 and 18.96.0.0/11. This worm appears to differ from the Witty worm in lines 16 (always loop back to the same place), 18, 20 (always choose the same packet size), and 24–26.

8. [4 points]: What fraction of the total 32 bit IP address space does your network telescope listen on?

Initials:

9. [8 points]: Say your network telescope receives two packets from an infected node with the following parameters: (dest_ip_1, dest_port_1) and (dest_ip_2, dest_port_2) respectively. Given the two received packets, how would you identify whether they are likely to be from a single for loop execution?

If on same PRNG

(roughly $X_{Port[1..15]} = X_{IP[1-32]} || Y \cdot a \bmod m$

but can't see other eqn - so don't know

repd call rand see if repeats

↑ more actionable mine more elegant

Oh if on same loop

$X_{Port2} = ((X_{IP2} || Y \cdot a \bmod m) * a \bmod m)$

(could not actually find - not that good at # theory).

If there is a Y that works in both

10. [6 points]: Finally, you would like to measure the bandwidth of the infected node. To do this you look at two packets, from a single for loop execution, you received at your network telescope at times T_1 and T_2. The two packets had the following parameters: (dest_ip_1, dest_port_1) and (dest_ip_2, dest_port_2). Assume the packet header size is 100 bytes, how would you measure the bandwidth of the infected node?

Know how many packets sent b/w them

And diff in recieved time

And size = 1100 bytes

Enough to calc ✓

Initials:

IV Two-Phase Commit

Consider the two-phase commit protocol for executing atomic transactions across multiple sites. This protocol is described in Sections 9.6.2 and 9.6.3 of the text, as well as in lecture, and is summarized below

One node is designated as the *coordinator*, and the other nodes are designated as *workers*. Under normal circumstances, the protocol proceeds as follows:

- The coordinator sends a `PREPARE` message to each worker.
- The worker responds by writing a `PREPARED` record to its log, then sending a `VOTE COMMIT` message to the coordinator.
- After receiving `VOTE COMMIT` responses from all workers, the coordinator writes a `COMMITTED` record to disk and sends a `COMMIT` message to each worker.
- When a worker receives this message, it writes a `COMMITTED` record to its log.

11. [8 points]: Which of the following statements about this protocol are true? Assume that none of the components (network, coordinator, and workers) crash permanently.

(Circle True or False for each choice.)

A. True / False If any worker commits a transaction (by writing `COMMITTED` to its log), all other workers will also eventually commit that transaction. Since coord recvd it

B. True / False It guarantees that all workers will commit the transaction within a fixed time. ∞ time

C. True / False If a worker is using two-phase locking to guarantee before-or-after atomicity, it cannot release the locks held by a transaction until after it receives the `COMMIT` message from the coordinator.

D. True / False If the coordinator fails to receive a `VOTE COMMIT` message from some worker in a timely manner, it can decide to abort the transaction.

12. [4 points]: What is the earliest point after which the transaction is guaranteed to commit? Again, assume that none of the components (network, coordinator, and workers) crash permanently.

(Circle the BEST answer)

- After the coordinator sends `PREPARE` messages
- After the last worker sends its `VOTE` message
- After the coordinator writes the `COMMITTED` record to its log
- After all workers have written the `COMMITTED` record to their logs

Initials:

V Consistency/Replication

The software that runs the retail web site for amazon.com uses an eventually consistent key-value storage system called Dynamo. The basic operations in such a system are to read and write the value for a particular key. For example, a customer's shopping cart is a single object, with the customer's user ID as its key and the set of items in the cart as the value. Versions are labeled by vector clocks, and an update overwrites any earlier versions with smaller clock labels. For simplicity, assume that a set operation contacts only one server, and that server then propagates the new value to the other replicas asynchronously. Each set operation can choose a different server (e.g., if it believes the last server it talked to may have failed or is otherwise unresponsive).

If nodes fail and recover, or respond slowly, it's possible to have two or more versions with clock labels that are not totally ordered; in this case it's the application's job to reconcile them when it reads the cart from Dynamo. In case of doubt the cart application reconciles by keeping any items whose status is in doubt. Yes

(So must contact >1 server

13. [8 points]:. Suppose the cart is empty and the user adds item A to the cart, then adds item B, then deletes A. What are the possible contents of the cart that a subsequent read could see?

(Circle True or False for each choice.)

on a diff server? assuming yes

A. True / False {A}

B. True / False {B}

C. True / False {A, B}

D. True / False {}

14. [12 points]: With the same sequence of operations, what values could a read see after the system reaches eventual consistency, and no more updates need to be propagated, if there are no other writes to the cart?

(Circle True or False for each choice.)

A. True / False {A}

B. True / False {B} ← oh that's the goal state!

C. True / False {A, B} ← this if V_1 {A} then deletes V_1 so V_2 {A,B} V_2 {A,B} V_3 { }

D. True / False {}

15. [7 points]: It's possible to avoid these anomalies completely and have a strongly consistent (single-copy) storage system by running a replicated state machine, using a consensus protocol such as Paxos, to do the read and write operations. The disadvantages of this method, which presumably caused Amazon to reject it in favor of eventual consistency, include which of the following?

I don't know why this?

(Circle True or False for each choice.)

Not new version Vector?

Initials:

but here mostly black box

Oh if delete goes to sep server?

Confusing ← then consistency process runs on wrong one ← delete goes to wrong ver

prob since heavy duty ?

A. True / False You can't do writes without access to a majority of the replicas.

B. True / False You can't do reads without access to a majority of the replicas. do

C. True / False It's more complicated for the application to deal with. single copy simpler

D. True / False It requires the application to contact several servers directly. use contacts primary which talks to other

E. True / False The latency for both read and write operations is greater.

yes but how is this false

End of Quiz

Please double check that you wrote your name on the front of the quiz, and circled your recitation section number.

don't know in that much detail in lecture

Initials:

Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

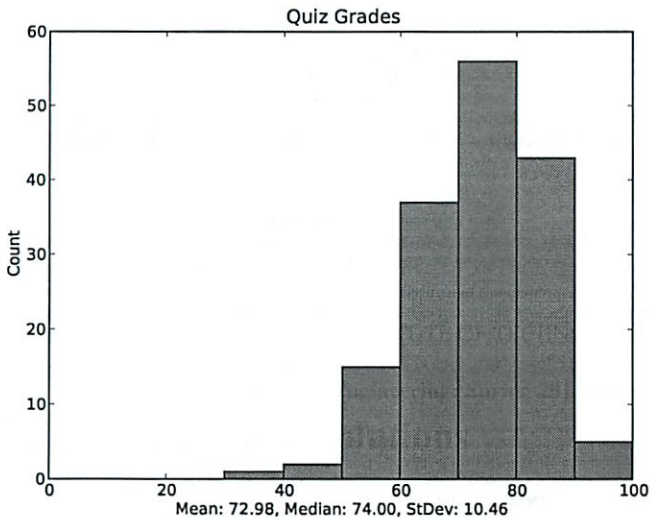
6.033 Computer Systems Engineering: Spring 2011

Quiz 3 Solutions

There are 15 questions and 11 pages in this quiz booklet. Answer each question according to the instructions given. You have **90 minutes** to answer the questions.

THIS IS AN OPEN BOOK, OPEN NOTES, OPEN LAPTOP QUIZ, BUT DON'T USE YOUR LAPTOP FOR COMMUNICATION WITH OTHERS.

Grade distribution histogram:



I Reading Questions

1. [8 points]: Based on the paper "The Recovery Manager of the System R Database Manager", which of the following statements are correct?
(Circle True or False for each choice.)

A. **True / False** As part of recovery from a crash, System R restores the current version of all shadowed files to their shadow copy.

Answer: True. System R discards all current edits, and restores all files to the shadowed copy which was created at the time of the last checkpoint.

B. **True / False** The shadow copy of a file reflects only the effects of committed transactions.

Answer: False: Uncomitted transactions in progress at the time of the last checkpoint may be present in the shadow copy

C. **True / False** System R will undo the effects of certain transactions during recovery.

Answer: True: it will undo transactions started before the last checkpoint but not committed.

D. **True / False** System R will redo the effects of certain transactions during recovery.

Answer: True: it will redo comitted transactions completed after the last checkpoint.

2. [6 points]: Based on the paper "Beyond Stack Smashing: Recent Advances in Exploiting Buffer Overruns", which of the following statements are correct?
(Circle True or False for each choice.)

A. **True / False** The basic form of attack described in the paper (stack smashing) allows an attacker to execute arbitrary code by overwriting far enough beyond the end of an application buffer to also overwrite the stack pointer.

Answer: False: it overwrites the return address, not the stack pointer.

B. **True / False** Even if a system guards against buffer overruns by making the stack non-executable, and uses stack cookies to guard against arc injection / return-into-libc attacks, the system could still be vulnerable to attacks which modify function pointers.

Answer: True: function pointers could exist in local variables without needing to overwrite a cookie value.

C. **True / False** A heap buffer overrun writes beyond the end of a dynamically-allocated object (e.g., created with `new` in C++ or `malloc()` in C), instead of a procedure's local variables.

Answer: True

Initials:

3. [6 points]: Which of the following statements is true about the Porcupine distributed email system described by Saito, Bershad and Levy?

(Circle True or False for each choice.)

A. True / False In Porcupine, soft state such as the user map is replicated on every node for fault-tolerance.

Answer: False. By definition, soft state can be computed from something else, so it doesn't need to be replicated for fault-tolerance. Porcupine replicates the user map for performance.

B. True / False In order to balance the load across the cluster, when a new message arrives Porcupine always picks a lightly loaded node at random and adds the message to the user's mailbox fragment on that node.

Answer: False. Porcupine stores a new message on a lightly loaded node that already has a mailbox fragment for the user, unless the number of such nodes is less that the spread. This is done to limit the number of nodes that need to be contacted when the user reads their mail.

4. [4 points]: Porcupine's scheme for maintaining cluster membership chooses a new coordinator whenever anyone detects an event: a node going up or down. The coordinator broadcasts a ping and gets a response from every node that is up. The total amount of work done in a cluster of n nodes, each time some node goes up or down, is proportional to which of the following?

(Circle the BEST answer)

- n
- $n \log n$
- n^2

Answer: The number of events per second scales linearly with the number of nodes, and each node has to respond to each event, so the total amount of work scales like n^2.

Initials:

II Lectures

5. [4 points]: According to Prof. Abelson's lecture, the "Good Samaritan Provision" of the Communications Decency Act provided protection to which of the following parties?

(Circle True or False for each choice.)

A. True / False CMU so that it can study the use of pornographic information on the Internet without being liable.

Answer: False. The CMU report wasn't the topic of a case under the CDA.

B. True / False AOL in the case against Ken Zeran, because AOL merely distributed information but was not responsible for the content.

Answer: True. AOL successfully defended itself using the Good Samaritan Provision.

C. True / False Prodigy in the case versus Stratton-Oakmont because Prodigy filtered offensive content.

Answer: False. This case predated the CDA, but was a precedent.

D. True / False The nastiest place on earth because it merely distributed content that others generated.

Answer: False. This case predated the CDA and has no immediate relation to it.

6. [8 points]: Consider the mechanisms for storing and checking user's passwords, as described in lecture.

(Circle True or False for each choice.)

A. True / False Rather than storing user's login passwords directly, Unix stores the result of applying a hash function to the user's password. One reason for this approach is so that users' plaintext passwords are not revealed if an attacker manages to read the password database.

Answer: True. The hash function is difficult to reverse, so it is hard to recover the plaintext passwords.

B. True / False Unix combines *salts* with a user's password before sending them through the hash function to enact a policy that requires users to periodically revise their passwords.

Answer: False. A salt is added such that the same password does not always map to the same hashed value

C. True / False Let $H()$ be a hash function. If given a value p, it is easy to find a value P such that $H(P)$ equals p, then $H()$ is a poor choice of hash function to use as part of a password checking system.

Answer: True. If the hash function were reversible, an attacker could find a working password easily without necessarily guessing the correct one

D. True / False The study of password use described in lecture found that different people almost never choose the same passwords.

Answer: False. It found several very commonly repeated passwords.

Initials:

7. [7 points]: Consider the Same Origin Policy (SOP) used for browser security.
(Circle True or False for each choice.)

A. True / False Javascript code from the page at `http://web.mit.edu/6.033` can access the DOM of another page at `http://web.mit.edu/6.041` loaded in the same browser. Assume the second page runs no Javascript code.

Answer: True. The port, protocol, and host are all exactly the same, so this is permitted.

B. True / False Javascript code from the page at `http://web.mit.edu/6.033` can access the DOM of another page at `http://courses.csail.mit.edu/6.033` loaded in the same browser. Assume the second page runs no Javascript code.

Answer: False. The hosts are different, so this is not permitted. Even though the hosts are part of the same subdomain (mit.edu), the same origin policy treats them as different unless each page declares a common host, but that cannot be done without Javascript code.

C. True / False Javascript code from the page at `http://web.mit.edu/6.033` can access the DOM of another page at `http://www.google.com` loaded in the same browser. Assume the second page runs no Javascript code.

Answer: False. The hosts are different, so this is not permitted.

D. True / False Suppose the 6.033 staff copies the Javascript code for an advertisement from `attacker.com` into the page at `http://web.mit.edu/6.033` (for example, by including a tag like <SCRIPT SRC="`http://attacker.com/ad.js`">). The Same-Origin Policy will allow that Javascript code from attacker.com to access the DOM of the 6.033 page (`http://web.mit.edu/6.033`).

Answer: True. This is permitted because the 6.033 page explicitely requested the attacker's code.

E. True / False The Same-Origin Policy allows Javascript code from `site1.com` to access the cookie of `site2.com`, as long as the two sites are concurrently loaded in two different tabs in the same window.

Answer: False. The hosts are different, so this is not permitted

Initials:

III Worm Trouble

It is May 2011 and there is a new worm, *nitty*, released on the Internet. Security experts quickly analyze the code corresponding to Nitty to be as follows:

```
1  int X = 0; /* X is a global variable */
2
3  int rand() {
4    /* Note that 32-bit integers obviate the need for a modulus operation here. */
5    /* A = 214013, B = 2531011. */
6    X = X * A + B;
7    return X;
8  }
9
10 srand(seed) {
11   X = seed;
12 }
13
14 main() {
15   srand(get_tick_count());
16   while (1) {        /* infinite loop */
17     for (int i = 0; i < 20,000; ++i) {
18       dest_ip = rand();
19       dest_port = rand() [0:15];      /* low 16 bits */
20       packetsize = 1000;              /* 1000 bytes */
21       packetcontents = top_of_stack;
22       sendto();
23     }
24     int rand_temp = rand();
25     if (open(physicaldisk, rand_temp[13:15]))
26       overwrite_block(rand_temp[0:14] || 0x4e20);
27   }
28 }
```

You learned about network telescopes in 6.033 and decided to see if you can apply the concepts to infer information about this new worm. You set up a router at MIT and configure it to listen on two unused portions of the MIT IP address space corresponding to 18.32.0.0/11 and 18.96.0.0/11. This worm appears to differ from the Witty worm in lines 16 (always loop back to the same place), 18, 20 (always choose the same packet size), and 24–26.

8. [4 points]: What fraction of the total 32 bit IP address space does your network telescope listen on?

Answer: Each unused /11 portion corresponds to $1/2^{11}$ of the 32 bit IP address space. So the combined fraction from the two ranges is $1/2^{10}$ or 2^{-10}.

Initials:

9. [8 points]: Say your network telescope receives two packets from an infected node with the following parameters: (dest_ip_1, dest_port_1) and (dest_ip_2, dest_port_2) respectively. Given the two received packets, how would you identify whether they are likely to be from a single for loop execution?

Answer: There are two calls to rand() in each for loop iteration, and we know that either dest_ip_1 or dest_ip_2 is equal to X when the first packet is sent. We can thus use dest_ip_1 as a seed and repeatedly call rand() to reach dest_ip_2, or try dest_ip_2 as a seed and repeatedly call rand() to reach dest_ip_1. In either case, if the number of applications of rand() is even and less than 40000, then the two packets are from the same loop execution.

10. [6 points]: Finally, you would like to measure the bandwidth of the infected node. To do this you look at two packets, from a single for loop execution, you received at your network telescope at times T_1 and T_2. The two packets had the following parameters: (dest_ip_1, dest_port_1) and (dest_ip_2, dest_port_2). Assume the packet header size is 100 bytes, how would you measure the bandwidth of the infected node?

Answer: As before, start with dest_ip_1 as seed and apply rand() until you get dest_ip_2, or start with dest_ip_2 as seed and apply rand() until you get dest_ip_1. In either case, say the number of rand() invocations needed is C. Then, the number of packets sent between the two packets is $C/2$. The size of each packet is 100 bytes (packet header) + 1000 bytes (packet size from source code) = 1100 bytes. The time in which this happens is $(T_2 - T_1)$.

Thus, the bandwidth of the infected node is $\frac{550C}{T_2-T_1}$ bytes/second.

Initials:

IV Two-Phase Commit

Consider the two-phase commit protocol for executing atomic transactions across multiple sites. This protocol is described in Sections 9.6.2 and 9.6.3 of the text, as well as in lecture, and is summarized below

One node is designated as the *coordinator*, and the other nodes are designated as *workers*. Under normal circumstances, the protocol proceeds as follows:

- The coordinator sends a `PREPARE` message to each worker.
- The worker responds by writing a `PREPARED` record to its log, then sending a `VOTE COMMIT` message to the coordinator.
- After receiving `VOTE COMMIT` responses from all workers, the coordinator writes a `COMMITTED` record to disk and sends a `COMMIT` message to each worker.
- When a worker receives this message, it writes a `COMMITTED` record to its log.

11. [8 points]: Which of the following statements about this protocol are true? Assume that none of the components (network, coordinator, and workers) crash permanently.

(Circle True or False for each choice.)

A. True / False If any worker commits a transaction (by writing `COMMITTED` to its log), all other workers will also eventually commit that transaction.

Answer: True. This is the goal of two-phase commit. If a worker committed the transaction, that means the coordinator sent it a `COMMIT` message and recorded the transaction commit in the log. Even if there are failures, a worker that queries the coordinator will find that the transaction committed, and commit its part of the transaction.

B. True / False It guarantees that all workers will commit the transaction within a fixed time.

Answer: False. The two-generals problem says that this is impossible. If there are failures, it might take arbitrarily long to complete the protocol.

C. True / False If a worker is using two-phase locking to guarantee before-or-after atomicity, it cannot release the locks held by a transaction until after it receives the `COMMIT` message from the coordinator.

Answer: True. If the worker releases its locks, that makes the effects of the transaction visible to other concurrent transactions. It can't do this until receiving the `COMMIT` message, because until then the transaction might still abort.

D. True / False If the coordinator fails to receive a `VOTE COMMIT` message from some worker in a timely manner, it can decide to abort the transaction.

Answer: True. The coordinator can unilaterally abort the transaction until it writes the `COMMITTED` record to its log. This is fine, because all of the workers are in the tentative commit state and can still be aborted.

Initials:

12. [4 points]: What is the earliest point after which the transaction is guaranteed to commit? Again, assume that none of the components (network, coordinator, and workers) crash permanently.

(Circle the BEST answer)

- After the coordinator sends PREPARE messages
- After the last worker sends its VOTE message
- **After the coordinator writes the COMMITTED record to its log.**
- After all workers have written the COMMITTED record to their logs

Answer: The third choice is correct. After the coordinator writes COMMITTED record to its log, the decision to commit will survive even if the coordinator crashes. If any of the workers crash, they will resend their VOTE COMMIT message when they recover, and learn from the coordinator that the transaction committed.

The second choice is incorrect because the coordinator can still abort the transaction. For example, it might do so if one of the VOTE messages is dropped by the network.

Initials:

V Consistency/Replication

The software that runs the retail web site for amazon.com uses an eventually consistent key-value storage system called Dynamo. The basic operations in such a system are to read and write the value for a particular key. For example, a customer's shopping cart is a single object, with the customer's user ID as its key and the set of items in the cart as the value. Versions are labeled by vector clocks, and an update overwrites any earlier versions with smaller clock labels. For simplicity, assume that a set operation contacts only one server, and that server then propagates the new value to the other replicas asynchronously. Each set operation can choose a different server (e.g., if it believes the last server it talked to may have failed or is otherwise unresponsive).

If nodes fail and recover, or respond slowly, it's possible to have two or more versions with clock labels that are not totally ordered; in this case it's the application's job to reconcile them when it reads the cart from Dynamo. In case of doubt the cart application reconciles by keeping any items whose status is in doubt.

13. [8 points]:. Suppose the cart is empty and the user adds item A to the cart, then adds item B, then deletes A. What are the possible contents of the cart that a subsequent read could see?

(Circle True or False for each choice.)

A. True / False {A}

B. True / False {B}

C. True / False {A, B}

D. True / False {}

Answer: All are true. In normal operation the sequence of values will be v1:{A}, v2:{A, B}, v3:{B}, with $v1 < v2 < v3$, and the result will be v3: {B}. If failure makes all these values inaccessible, the result will be {}. If v2 and v3 are written to different servers and v3's server becomes inaccessible, the result will be {A, B}. Or A and B could be written to incomparable versions, leading to versions v1:{A} and v2: {B}. A read that sees both will return {A, B}.

14. [12 points]: With the same sequence of operations, what values could a read see after the system reaches eventual consistency, and no more updates need to be propagated, if there are no other writes to the cart?

(Circle True or False for each choice.)

A. True / False {A}

B. True / False {B}

C. True / False {A, B}

D. True / False {}

Answer: B and C are true. The latter can happen if A and B are written in order, leading to v1:{A}, v2:{A, B}, with $v1 < v2$, and then the delete goes to v1, leading to v2:{A, B}, v3:{} with $v2 \neq v3$. Now after reconciling we end up with {A, B}.

Initials:

15. [7 points]: It's possible to avoid these anomalies completely and have a strongly consistent (single-copy) storage system by running a replicated state machine, using a consensus protocol such as Paxos, to do the read and write operations. The disadvantages of this method, which presumably caused Amazon to reject it in favor of eventual consistency, include which of the following?

(Circle True or False for each choice.)

A. **True / False** You can't do writes without access to a majority of the replicas.

B. **True / False** You can't do reads without access to a majority of the replicas.

C. **True / False** It's more complicated for the application to deal with.

D. **True / False** It requires the application to contact several servers directly.

E. **True / False** The latency for both read and write operations is greater.

Answer: A, B, and E are true. B is true because otherwise you might read an older value than was written by the most recent write (the primary can take out a lease, a lock with a timeout, to ensure that it remains the primary for some length of time, and then it can handle reads unilaterally). E is true because the need to contact a majority makes the latencies greater. Not C; single-copy is simpler for the application. Not D; the application can contact one server, usually called the primary, which must talk to a majority to carry out the operation.

End of Quiz

Please double check that you wrote your name on the front of the quiz,
and circled your recitation section number.

Initials:

Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.033 Computer Systems Engineering: Spring 2010

Quiz III

There are 13 questions and 11 pages in this quiz booklet. Answer each question according to the instructions given. You have **90 minutes** to answer the questions.

Some questions are harder than others and some questions earn more points than others—you may want to skim all questions before starting.

If you find a question ambiguous, be sure to write down any assumptions you make. **Be neat and legible.** If we can't understand your answer, we can't give you credit!

Write your name in the space below. Write your initials at the bottom of each page.

THIS IS AN OPEN BOOK, OPEN NOTES QUIZ.
You may use a computer to look at PDFs and notes but not for any other purpose.

CIRCLE your recitation section number:

10:00	1. Lampson/Kushman	
11:00	2. Jones/Rieb	3. Rudolph/Kushman
12:00		4. Rudolph/Rieb
1:00	5. Gifford/Post	6. Jones/Spicer
2:00	7. Gifford/Spicer	8. Lampson/Post

5/2 #3
Pratice

Do not write in the boxes below

1-4 (xx/24)	5-9 (xx/30)	10-12 (xx/20)	13-15 (xx/26)	Total (xx/100)

Name:

I Multiple-Choice Questions

out of scope

1. [6 points]: With respect to the paper "The Recovery Manager of the System R Database Manager" by Gray *et al.* mark each of the following statements true or false.

(Circle True or False for each choice.)

A. True / False Before modifying a "shadowed" file, the entire file is copied, and only the "current" version is modified: the shadowed version is not changed.

B. True / False Before any modified pages can be written to disk, the COMMIT log record for the transaction must be forced to disk.

C. True / False After a checkpoint is written to disk, System R discards all log records that precede the checkpoint record.

D. True / False After saving a shadowed file (making the current version the new shadow version), the old shadow versions of the modified pages can be safely marked as free and reused.

out of scope

2. [6 points]: In class we learned that DNS is an example of an eventually consistent system. Which of the following statements about DNS are true?

(Circle True or False for each choice.)

A. True / False If DNS is initially configured to resolve name N to IP address A, and is later reconfigured to resolve N to IP address B, clients looking up N after this reconfiguration may continue to receive A as an answer for lookups of N.

B. True / False When there are no network partitions, DNS lookups see changes to DNS records immediately.

C. True / False When consistency has been achieved, a given DNS name resolves to exactly one IP address.

3. [6 points]: Indicate which of the following statements about Ross Anderson's paper "Why Cryptosystems Fail" are true.

(Circle True or False for each choice.)

A. True / False Anderson argues that discussing security failures openly improves security.

B. True / False Most of the security failures described are a result of compromised cryptographic protocols. Human

C. True / False Anderson suggests that system designers must consider the operation of the equipment as part of the security of the system.

D. True / False The "dual control" concept, where two individuals must collaborate to perform a function, is inconvenient and is sometimes be bypassed by people wanting to save time.

E. True / False Anderson suggests that if we had a set of "perfectly secure" components, a system composed of these components would also be perfectly secure.

No way - 6.033 big idea

4. [6 points]: Indicate which of the following statements about the ObjectStore system described in the paper by Lamb et al (reading 18) are true.

(Circle True or False for each choice.)

A. True / False ObjectStore stores persistent objects in essentially the same format as the ordinary transient C++ objects. make avilable in - it talked about access

B. True / False To get transactions that are atomic with respect to other transactions that execute concurrently, ObjectStore requires the programmer to lock an object before using it, by explicitly invoking acquire, unlike an ordinary relational database system. ; auto locks

Store serve I guess

C. True / False If transaction T1 touches objects A and B and no others, and transaction T2 touches objects C and D and no others, then ObjectStore's locking system ensures that T1 and T2 can run concurrently. if on same pg should be good

D. True / False In ObjectStore an object of type T can be either persistent or transient, and the choice is made when the object is created.

I don't think so (the whole pt) not in paper anymore

Ok R13 - included

Same - VM systems change

(cons does not seem to live up)

Initials:

5. [6 points]: Indicate which of the following statements about the paper "Hints for Computer System Design" by Lampson (reading 26) are true.

(Circle True or False for each choice.)

A. True / False According to the paper (and the doctrine of 6.033), simplicity of design and interface are always the highest priority.

B. True / False DNS' hierarchical lookups are a good example of following the hint to use brute force.

C. True / False "Keep secrets of the implementation" and "Dont hide power behind an interface" are hints that are often at odds with each other.

D. True / False "Keep basic interfaces stable" is a hint that severely obstructs progress.

6. [6 points]: With respect to the paper "Manageability, availability and performance in Porcupine: a highly scalable, cluster-based mail service" by Saito et. al, which of the following statements is true?

(Circle True or False for each choice.)

A. True / False A given user's mailbox fragments are all stored on the same node.

B. True / False When a failed node recovers after being down for a day the mailbox fragments it stores are brought back up to date from logs on the other nodes.

C. True / False The mailbox fragment list is hard state that keeps track of the nodes that contain a user's mailbox.

D. True / False It is possible that membership services will break a cluster into two disconnected groups of nodes in the presence of certain unusual network failures.

Initials:

7. [6 points]: Given the context of the paper "Exploiting Underlying Structure for Detailed Reconstruction of an Internet-scale Event" by Kumar et. al which of the following statements are true?

(Circle True or False for each choice.)

Source was true

A. True / False Witty generated packets with random forged source IP addresses, and thus network telescopes were able to detect replies from victim hosts to the IP addresses fabricated by Witty.

B. True / False The Witty worm exploited a buffer overflow in the ICQ client in Microsoft Windows.

Raters from IBMISS

C. True / False It is possible to operate an Internet wide network telescope by telling your computer to listen for the packets addressed to the address range you wish to monitor.

only blank - or if big enter

D. True / False The origin of "Patient Zero" was determined by carefully observing which host was first observed at the network telescope.

re from diff pattern

8. [6 points]: Indicate the truth or falsehood of each of the following with respect to Ken Thompson's paper "Reflections on Trusting Trust."

(Circle True or False for each choice.)

specifically

cheap shot...

A. True / False Thompson believes that self-reproducing programs shouldn't be trusted.

he doesn't say anything about

B. True / False A Trojan horse like the one Thompson describes could not have been hidden in a compiler for a more modern language like Java.

even more so

C. True / False The Trojan horse Thompson embedded in the login program could have been found by looking at the machine instructions being executed by the CPU.

ARM code

D. True / False A programmer can prevent the type of attack Thompson describes by writing all of his or her programs in assembly code.

Not compiled

Assembler is also compiled could be similar

Initials:

Buffer overrun, as explained in the paper "Beyond stack smashing: recent advances in exploiting buffer overruns" by Pincus and Baker, can overwrite the procedure return address that is stored on the stack. To avoid this problem, Betty writes a compiler that generates code that maintains two different stacks. One is the usual kind but without the procedure return address, and a second stack, stored in a different part of memory, is used only for the procedure return address.

9. [6 points]: Indicate whether each statement is true or false with respect to Betty's compiler.
(Circle True or False for each choice.)

A. True / False Betty's compiler avoids the classic stack smashing exploit as explained in papers previous to Pincus and Baker, by such hackers as AlephOne and DilDog.

B. True / False Betty's compiler avoids the Arc injection exploit outlined in the paper.

C. True / False Betty's compiler avoids the Function Pointer clobbering subterfuge exploit outlined in the paper.

D. True / False Betty's compiler avoids the Exception-handler hijacking exploit outlined in the paper.

E. True / False Betty's compiler avoids the Heap smashing exploit outlined in the paper.

Were they mentioned?

Can't still return there - since return address controlled in other pg

write your own return code

no - still reads from sp

Arc

Initials:

II Two-Phase Commit

Suppose you are running a transactional 3 node log-based storage system that is using two-phase commit as described in class.

Node 1 is the coordinator, and node 2 and 3 are just workers.

After awhile, node 1 crashes, and the log on its disk looks as follows (here ... indicates some number of UPDATE operations; you can assume in the following three questions that transactions do not UPDATE any of the same records, and that every transaction updates at least one data item):

```
BEGIN T1
BEGIN T2
...
ABORT T1
BEGIN T3
...
COMMIT T3
```

10. [6 points]:

For each of the following transactions, indicate whether the coordinator will end up considering the transaction to have committed or aborted, or whether the information in the log is not sufficient to decide.

(Circle committed, aborted, or can't tell for each transaction.)

A. T1	Committed	Aborted	Can't tell
B. T2	Committed	Aborted	Can't tell
C. T3	Committed	Aborted	Can't tell

Node 1 recovers, and resumes processing transactions. After awhile, node 2 crashes, and the log on its disk looks as follows (assume you know nothing about the coordinator's state other than what is implied by the following):

```
BEGIN T4
...
PREPARE T4
BEGIN T5
BEGIN T6
...
PREPARE T6
...
COMMIT T4
```

Initials:

11. [6 points]:

For each of the following transactions indicate whether node 2 will end up considering the transaction to have committed or aborted, or whether the information in the above log is not sufficient to decide.

(Circle committed, aborted, or can't tell for each transaction.)

A. T4	Committed	Aborted	Can't tell	
B. T5	Committed	Aborted	Can't tell	1 auto aborts the others
C. T6	Committed	Aborted	Can't tell	when restarts? not clear in notes

Ben Bitdiddle notices that two-phase commit workers have to write two log records for every transaction (a PREPARE record and a COMMIT or ABORT record.) Ben proposes a protocol called *Ben's 2 Phase Commit (B2PC)*. In B2PC, the messages and operation of the protocol are identical to the two-phase commit protocol we learned in class. The only difference is that, when a transaction commits, the workers do not write a COMMIT record to the log (they do, however, still write ABORT records to the log.) When scanning the log during recovery, workers assume that a transaction they prepared actually committed unless an ABORT record for the transaction appears in the log. The coordinator still logs COMMIT records as in the original protocol.

silly

if abort message never valid

12. [8 points]: Which of the following statements about B2PC protocol are true? Assume that "correct transactional behavior" means the outcome (i.e., COMMIT or ABORT) of the transaction would be the same as in the unmodified 2PC protocol.

(Circle True or False for each choice.)

A. True / False In the absence of crashes or other faults on any of the nodes, B2PC provides correct transactional behavior.

log never used

For the following choices, assume that the workers or coordinator may crash, and that there are no other faults in the system.

B. True / False In this case, B2PC provides correct transactional behavior.

C. True / False Assume the coordinator remembers the outcome of all transactions forever. Suppose Ben modifies the protocol described above so that, when recovering from a crash, workers contact the coordinator for the outcome of any prepared transaction that does not have an ABORT record in their log. In this case, B2PC would provide correct transactional behavior.

D. True / False Suppose Ben modifies the protocol described above so that workers do write COMMIT records, but the coordinator omits them. In this case, B2PC would provide correct transactional behavior.

no explanation ? never

Does the coord respond? If could fail

Coord restarting

splitting hairs
trivia
prob could argue

Initials:

III Trusting Ted's Terrific Telegraphic Text Teamware

Theodore is developing a collaborative editor called Ted's Terrific Telegraphic Text Teamware, or TTTTT. Theodore plans to have lots of enthusiastic customers, and he knows that he will have to add new features and issue new releases constantly. He is mulling over the problem of how his customers can verify the authenticity of each new release. A release consists of a single executable file, called ttttt-V.exe (where V is the version number), so the problem boils down to each customer being able to verify that their ttttt-V.exe file has the contents that Theodore intended. Your job is to give Theodore advice about three different authentication schemes he is contemplating.

Theodore's first scheme is to calculate a hash of the content of each release, and to post the hash along with the release on his web site. He uses a cryptographic hash function as described in section 11.2.3 in the course textbook. For each release, Theodore calculates $h_V = H(\text{"TTTTT"} + V + R_V)$, where R_V is the content of the release file ttttt-V.exe and $+$ indicates string concatenation. Theodore does all his development, compiling, and computing of hash values on his laptop, which only he has access to. He posts the release and h_V on his web site at these URLs:

```
http://ted.com/ttttt-V.exe
http://ted.com/V-hash.dat
```

He tells all his customers to fetch both files, and to check for authenticity by comparing the fetched hash value with $H(\text{"TTTTT"} + V + R'_V)$, where R'_V is the contents of the ttttt-V.exe file they fetched. If the two are equal, the customer should accept the new release. If they are not equal, the customer should reject the release.

13. [8 points]: Indicate the truth or falsehood of each of the following statements about Theodore's first scheme.

(Circle True or False for each choice.)

A. True / False An attacker capable of modifying the packets of the customer's transfer of the ttttt-V.exe file could do so in a way way that would cause the customer's ttttt-V.exe file to be different from what Theodore intended, but would cause the customer to nevertheless accept the release.

B. True / False It would be easy for an attacker who can read and modify any files on Theodore's server to cause customers to accept a release that has different contents from what Theodore intended.

C. True / False If a customer sees a "mirror site" of TTTTT that is not affiliated with Theodore, consisting of (for example) http://mirror.com/ttttt-V.exe and http://mirror.com/V-hash.dat, it would be just as safe to fetch those files and compare hashes as it would be to fetch the files from Theodore's web site.

Initials:

Theodore's second scheme is to generate authentication tags for each software version using a shared-secret message authentication code (MAC), as described in the textbook in sections 11.3.3, 11.3.4, and 11.3.5. Theodore generates a separate key K_i for each of his customers, and contacts each customer on the telephone to give them their K_i. For each new release V, Theodore calculates an authentication tag $T_{V,i}$ for each of his customers by calling SIGN("TTTTT" + V + R_V,K_i), where R_V is the content of ttttt-V.exe. Theodore does all his development, compiling, and computing of MAC values on his laptop, and he stores the K_i keys only on his laptop; only Theodore has access to this laptop. Theodore posts the release file and all the tag files on his web site, at these URLs:

here keyed

```
http://ted.com/ttttt-V.exe
http://ted.com/V-1.tag
http://ted.com/V-2.tag
...
```

same key?

He tells his customers to each fetch the new release and their tag from the web site and to call VERIFY("TTTTT"+ V + R'_V,T,K_i), where R'_V is the content of the ttttt-V.exe file they fetched and T is the content of the V-i.tag file they fetched. If the call returns ACCEPT, the customer should accept the new release; otherwise the customer should reject the new release.

14. [9 points]: Indicate the truth or falsehood of each of the following statements about Theodore's second scheme.

(Circle True or False for each choice.)

A. True / False An attacker capable of modifying the packets of the customer's transfer of the ttttt-V.exe file could do so in a way way that would cause the customer's ttttt-V.exe file to be different from what Theodore intended, but would cause the customer to nevertheless accept the release.

if stupid → see why crypto-sys fail

B. True / False It would be easy for an attacker who can read and modify any files on Theodore's server to cause customers to accept a release that has different contents from what Theodore intended.

C. True / False If a customer sees see a "mirror site" of TTTTT that is not affiliated with Theodore, consisting of (for example) http://mirror.com/ttttt-V.exe and http://mirror.com/V-i.tag files, it would just as safe to fetch those files and verify with VERIFY as it would be to fetch the files from Theodore's web site.

no call

mirror op can't change tag

Didn't really read close enough...

Initials:

Theodore's third scheme is to configure his web server to use SSL with a certificate from a well-known certificate authority. Theodore's server holds a particular private key, and the certificate says that whoever knows that private key is the rightful owner of the DNS domain ted.com. Theodore tells his customers to fetch new releases from

```
https://ted.com/ttttt-V.exe
```

Theodore tells his customers that they do not have to take any special steps to check the authenticity of the software, since, if you tell a web browser to connect to https://*servername*, it will use SSL to check that the server can prove ownership of a certificate for *servername* from a well-known authority. Theodore tells his customers to check that the right URL appears in their browsers' URL box. SSL (also known as TLS) was described in lecture, and you can also read about it in section 11.10 of the textbook.

15. [9 points]: Indicate the truth or falsehood of each of the following statements about Theodore's third scheme.

(Circle True or False for each choice.)

A. True / False An attacker capable of modifying the packets of the customer's transfer of the ttttt-V.exe file could do so in a way way that would cause the customer's ttttt-V.exe file to be different from what Theodore intended, but would cause the customer to nevertheless accept the release.

B. True / False It would be easy for an attacker who can read and modify any files on Theodore's server to cause customers to accept a release that has different contents from what Theodore intended.

C. True / False If a customer sees a "mirror site" of TTTTT that is not affiliated with Theodore, consisting of (for example) https://mirror.com/ttttt-V.exe, it would be just as safe to fetch those files via SSL as it would be to fetch the files from Theodore's web site.

End of Quiz III

Please double check that you wrote your name on the front of the quiz,
and circled your recitation section number.

Initials:

Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.033 Computer Systems Engineering: Spring 2010

Quiz III Solutions

I Multiple-Choice Questions

1. [6 points]: With respect to the paper "The Recovery Manager of the System R Database Manager" by Gray *et al.* mark each of the following statements true or false.

(Circle True or False for each choice.)

A. **True / False** Before modifying a "shadowed" file, the entire file is copied, and only the "current" version is modified: the shadowed version is not changed.

Answer: False. Only the modified pages are copied.

B. **True / False** Before any modified pages can be written to disk, the COMMIT log record for the transaction must be forced to disk.

Answer: False. Dirty records can be written to disk, since the log contains sufficient information to UNDO those operations.

C. **True / False** After a checkpoint is written to disk, System R discards all log records that precede the checkpoint record.

Answer: False. The log must contain all records for any active transactions. If a transaction that was started before the checkpoint is still running, the log can only be truncated up to that point.

D. **True / False** After saving a shadowed file (making the current version the new shadow version), the old shadow versions of the modified pages can be safely marked as free and reused.

Answer: True. These pages are no longer referenced by the new shadow, and if the shadow file is saved correctly, they will never be used by recovery.

2. [6 points]: In class we learned that DNS is an example of an eventually consistent system. Which of the following statements about DNS are true?

(Circle True or False for each choice.)

A. **True / False** If DNS is initially configured to resolve name N to IP address A, and is later reconfigured to resolve N to IP address B, clients looking up N after this reconfiguration may continue to receive A as an answer for lookups of N.

Answer: True. DNS may continue serving the old answer for a while from caches or from secondary nameservers that have not updated from their primaries.

B. **True / False** When there are no network partitions, DNS lookups see changes to DNS records immediately.

Answer: False. Same reasons as the previous answer.

C. **True / False** When consistency has been achieved, a given DNS name resolves to exactly one IP address.

Answer: False. A given name can have multiple DNS IP address records.

3. [6 points]: Indicate which of the following statements about Ross Anderson's paper "Why Cryptosystems Fail" are true.

(Circle True or False for each choice.)

A. True / False Anderson argues that discussing security failures openly improves security.

Answer: True. His position is that system designers must be able to learn from previous failures, in order to fix them.

B. True / False Most of the security failures described are a result of compromised cryptographic protocols.

Answer: False. Most of the failures are caused by people bypassing the security technology.

C. True / False Anderson suggests that system designers must consider the operation of the equipment as part of the security of the system.

Answer: True. Since many failures in this paper are caused by people using a "secure" system incorrectly, operating the system is a critical part.

D. True / False The "dual control" concept, where two individuals must collaborate to perform a function, is inconvenient and is sometimes bypassed by people wanting to save time.

Answer: True. While requiring two individuals to collaborate can provide good security, since both must collude to "break" the system, it can also be inconvenient. Anderson discusses two examples: a bank removing dual control to save time and money (although it seemed to lead to 10X fraud), and bank managers happily giving ATM technicians the crypto keys, so they do not have to wait while a machine is being serviced.

E. True / False Anderson suggests that if we had a set of "perfectly secure" components, a system composed of these components would also be perfectly secure.

Answer: False. Many security problems in this paper are caused by the interactions between components.

4. [6 points]: Indicate which of the following statements about the ObjectStore system described in the paper by Lamb et al (reading 18) are true.

(Circle True or False for each choice.)

A. True / False ObjectStore stores persistent objects in essentially the same format as the ordinary transient C++ objects.

Answer: True. It uses the same instruction sequences to access fields of both kinds of object, and the only difference between the disk format and the RAM format is possible adjustments to the values of the pointers (swizzling).

B. True / False To get transactions that are atomic with respect to other transactions that execute concurrently, ObjectStore requires the programmer to lock an object before using it, by explicitly invoking acquire, unlike an ordinary relational database system.

Answer: False. ObjectStore uses the VM system to detect reads and writes to objects and acquires the necessary locks automatically.

C. True / False If transaction T1 touches objects A and B and no others, and transaction T2 touches objects C and D and no others, then ObjectStore's locking system ensures that T1 and T2 can run concurrently.

Answer: False. The unit of locking is a VM page, so if A or B happens to be on the same VM page as C or D, T1 and T2 will have conflicting locks.

D. True / False In ObjectStore an object of type T can be either persistent or transient, and the choice is made when the object is created.

Answer: True. The formats are the same and the VM system rather than the compiled instructions takes care of bringing persistent objects into memory.

5. [6 points]: Indicate which of the following statements about the paper "Hints for Computer System Design" by Lampson (reading 26) are true.

(Circle True or False for each choice.)

A. True / False According to the paper (and the doctrine of 6.033), simplicity of design and interface are always the highest priority.

Answer: False. Simplicity is worth a lot, but if you really need performance or fault-tolerance then you have to design for them, even though it will make things more complicated. Simplicity is not a substitute for getting it right.

B. True / False DNS' hierarchical lookups are a good example of following the hint to use brute force.

Answer: False. Brute force would be a single server.

C. True / False "Keep secrets of the implementation" and "Dont hide power behind an interface" are hints that are often at odds with each other.

Answer: True. The secret that you keep may sometimes be the only way to release power. But a good interface like the file system needn't hide power, especially if its augmented with hints about prefetching and future needs for space.

D. True / False "Keep basic interfaces stable" is a hint that severely obstructs progress.

Answer: False. Almost always it's possible to move forward while providing compatibility with old interfaces. In the extreme, virtual machines make it possible to run an entire old OS, and protocols like X-windows that remote the user interface make it easy to mix access to old and new worlds.

6. [6 points]: With respect to the paper "Manageability, availability and performance in Porcupine: a highly scalable, cluster-based mail service" by Saito et. al, which of the following statements is true?
(Circle True or False for each choice.)

A. **True / False** A given user's mailbox fragments are all stored on the same node.

Answer: False. Each mailbox fragment may be located on a different node, to permit very large mailboxes to be distributed across multiple machines.

B. **True / False** When a failed node recovers after being down for a day the mailbox fragments it stores are brought back up to date from logs on the other nodes.

Answer: True. A recovering node contacts the cluster and asks other nodes which updates it missed.

C. **True / False** The mailbox fragment list is hard state that keeps track of the nodes that contain a user's mailbox.

Answer: False. It is soft state, i.e. it can be reconstructed from other information.

D. **True / False** It is possible that membership services will break a cluster into two disconnected groups of nodes in the presence of certain unusual network failures.

Answer: True. The paper notes that this may lead to inconsistent state, and Porcupine is designed to accommodate that.

7. [6 points]: Given the context of the paper "Exploiting Underlying Structure for Detailed Reconstruction of an Internet-scale Event" by Kumar et. al which of the following statements are true?
(Circle True or False for each choice.)

A. **True / False** Witty generated packets with random forged source IP addresses, and thus network telescopes were able to detect replies from victim hosts to the IP addresses fabricated by Witty.

Answer: False. The paper only makes sense assuming that the telescopes are able to see correct source addresses in the packets that Witty generates.

B. **True / False** The Witty worm exploited a buffer overflow in the ICQ client in Microsoft Windows.

Answer: False. It infected ICQ analyzers.

C. **True / False** It is possible to operate an Internet wide network telescope by telling your computer to listen for the packets addressed to the address range you wish to monitor.

Answer: False. You can't ordinarily ask to see packets addressed to destinations other than your IP address.

D. **True / False** The origin of "Patient Zero" was determined by carefully observing which host was first observed at the network telescope.

Answer: False. It wasn't the first host observed. It was different in that it was clearly running slightly different software (e.g. generating different patterns of random target IP addresses).

8. [6 points]: Indicate the truth or falsehood of each of the following with respect to Ken Thompson's paper "Reflections on Trusting Trust."
(Circle True or False for each choice.)

A. **True / False** Thompson believes that self-reproducing programs shouldn't be trusted.

Answer: False. While his "trusting trust" attack hides in programs that produce programs, he doesn't say anything about this making them more or less trustworthy.

B. **True / False** A Trojan horse like the one Thompson describes could not have been hidden in a compiler for a more modern language like Java.

Answer: False. There is nothing language specific about this attack.

C. **True / False** The Trojan horse Thompson embedded in the login program could have been found by looking at the machine instructions being executed by the CPU.

Answer: True. It might be difficult, but the back door does appear in the binary executable code.

D. **True / False** A programmer can prevent the type of attack Thompson describes by writing all of his or her programs in assembly code.

Answer: False. It would be just as easy to create a similar bug in the assembler used to translate the assembly code to machine code.

Buffer overrun, as explained in the paper "Beyond stack smashing: recent advances in exploiting buffer overruns" by Pincus and Baker, can overwrite the procedure return address that is stored on the stack. To avoid this problem, Betty writes a compiler that generates code that maintains two different stacks. One is the usual kind but without the procedure return address, and a second stack, stored in a different part of memory, is used only for the procedure return address.

9. [6 points]: Indicate whether each statement is true or false with respect to Betty's compiler.
(Circle True or False for each choice.)

A. **True / False** Betty's compiler avoids the classic stack smashing exploit as explained in papers previous to Pincus and Baker, by such hackers as AlephOne and DilDog.

Answer: True.

B. **True / False** Betty's compiler avoids the Arc injection exploit outlined in the paper.

Answer: True.

C. **True / False** Betty's compiler avoids the Function Pointer clobbering subterfuge exploit outlined in the paper.

Answer: False.

D. **True / False** Betty's compiler avoids the Exception-handler hijacking exploit outlined in the paper.

Answer: False.

E. **True / False** Betty's compiler avoids the Heap smashing exploit outlined in the paper.

Answer: False.

II Two-Phase Commit

Suppose you are running a transactional 3 node log-based storage system that is using two-phase commit as described in class.

Node 1 is the coordinator, and node 2 and 3 are just workers.

After awhile, node 1 crashes, and the log on its disk looks as follows (here ... indicates some number of UPDATE operations; you can assume in the following three questions that transactions do not UPDATE any of the same records, and that every transaction updates at least one data item):

```
BEGIN T1
BEGIN T2
...
ABORT T1
BEGIN T3
...
COMMIT T3
```

10. [6 points]:

For each of the following transactions, indicate whether the coordinator will end up considering the transaction to have committed or aborted, or whether the information in the log is not sufficient to decide.

(Circle committed, aborted, or can't tell for each transaction.)

A.	T1	Committed	Aborted	Can't tell **Answer: Aborted**
B.	T2	Committed	Aborted	Can't tell **Answer: Aborted**
C.	T3	Committed	Aborted	Can't tell **Answer: Committed**

Node 1 recovers, and resumes processing transactions. After awhile, node 2 crashes, and the log on its disk looks as follows (assume you know nothing about the coordinator's state other than what is implied by the following):

```
BEGIN T4
...
PREPARE T4
BEGIN T5
BEGIN T6
...
PREPARE T6
...
COMMIT T4
```


11. [6 points]:

For each of the following transactions indicate whether node 2 will end up considering the transaction to have committed or aborted, or whether the information in the above log is not sufficient to decide.

(Circle committed, aborted, or can't tell for each transaction.)

A.	T4	Committed	Aborted	Can't tell **Answer: Committed**
B.	T5	Committed	Aborted	Can't tell **Answer: Aborted**
C.	T6	Committed	Aborted	Can't tell **Answer: Can't tell**

Ben Bitdiddle notices that two-phase commit workers have to write two log records for every transaction (a PREPARE record and a COMMIT or ABORT record.) Ben proposes a protocol called *Ben's 2 Phase Commit (B2PC)*. In B2PC, the messages and operation of the protocol are identical to the two-phase commit protocol we learned in class. The only difference is that, when a transaction commits, the workers do not write a COMMIT record to the log (they do, however, still write ABORT records to the log.) When scanning the log during recovery, workers assume that a transaction they prepared actually committed unless an ABORT record for the transaction appears in the log. The coordinator still logs COMMIT records as in the original protocol.

12. [8 points]: Which of the following statements about B2PC protocol are true? Assume that "correct transactional behavior" means the outcome (i.e., COMMIT or ABORT) of the transaction would be the same as in the unmodified 2PC protocol.

(Circle True or False for each choice.)

A. True / False In the absence of crashes or other faults on any of the nodes, B2PC provides correct transactional behavior.

Answer: True. If there are no faults, B2PC is the same as 2PC.

For the following choices, assume that the workers or coordinator may crash, and that there are no other faults in the system.

B. True / False In this case, B2PC provides correct transactional behavior.

Answer: False. If a worker crashes after responding to a prepare message, and restarts, it will assume the transaction committed; but some other worker could have responded "no" to the prepare, in which case the transaction would have actually aborted.

C. True / False Assume the coordinator remembers the outcome of all transactions forever. Suppose Ben modifies the protocol described above so that, when recovering from a crash, workers contact the coordinator for the outcome of any prepared transaction that does not have an ABORT record in their log. In this case, B2PC would provide correct transactional behavior.

Answer: True.

D. True / False Suppose Ben modifies the protocol described above so that workers do write COMMIT records, but the coordinator omits them. In this case, B2PC would provide correct transactional behavior.

Answer: False. If the coordinator crashes just after sending out COMMIT messages, it will assume the transaction aborted after it restarts.

III Trusting Ted's Terrific Telegraphic Text Teamware

Theodore is developing a collaborative editor called Ted's Terrific Telegraphic Text Teamware, or TTTTT. Theodore plans to have lots of enthusiastic customers, and he knows that he will have to add new features and issue new releases constantly. He is mulling over the problem of how his customers can verify the authenticity of each new release. A release consists of a single executable file, called ttttt-V.exe (where V is the version number), so the problem boils down to each customer being able to verify that their ttttt-V.exe file has the contents that Theodore intended. Your job is to give Theodore advice about three different authentication schemes he is contemplating.

Theodore's first scheme is to calculate a hash of the content of each release, and to post the hash along with the release on his web site. He uses a cryptographic hash function as described in section 11.2.3 in the course textbook. For each release, Theodore calculates $h_V = H(\text{"TTTTT"} + V + R_V)$, where R_V is the content of the release file ttttt-V.exe and $+$ indicates string concatenation. Theodore does all his development, compiling, and computing of hash values on his laptop, which only he has access to. He posts the release and h_V on his web site at these URLs:

```
http://ted.com/ttttt-V.exe
http://ted.com/V-hash.dat
```

He tells all his customers to fetch both files, and to check for authenticity by comparing the fetched hash value with $H(\text{"TTTTT"} + V + R'_V)$, where R'_V is the contents of the ttttt-V.exe file they fetched. If the two are equal, the customer should accept the new release. If they are not equal, the customer should reject the release.

13. [8 points]: Indicate the truth or falsehood of each of the following statements about Theodore's first scheme.

(Circle True or False for each choice.)

A. True / False An attacker capable of modifying the packets of the customer's transfer of the ttttt-V.exe file could do so in a way that would cause the customer's ttttt-V.exe file to be different from what Theodore intended, but would cause the customer to nevertheless accept the release.

Answer: False. If the attacker can only modify the ttttt-V.exe file, then he will not be able to produce a different file that matches the hash.

B. True / False It would be easy for an attacker who can read and modify any files on Theodore's server to cause customers to accept a release that has different contents from what Theodore intended.

Answer: True. The attacker can install any ttttt-V.exe file on the server, and install a matching hash file.

C. True / False If a customer sees a "mirror site" of TTTTT that is not affiliated with Theodore, consisting of (for example) http://mirror.com/ttttt-V.exe and http://mirror.com/V-hash.dat, it would be just as safe to fetch those files and compare hashes as it would be to fetch the files from Theodore's web site.

Answer: False. Using a mirror opens the user up to an additional set of attacks in which whoever operates the mirror can serve incorrect (but matching) ttttt-V.exe and hash files.

Theodore's second scheme is to generate authentication tags for each software version using a shared-secret message authentication code (MAC), as described in the textbook in sections 11.3.3, 11.3.4, and 11.3.5. Theodore generates a separate key K_i for each of his customers, and contacts each customer on the telephone to give them their K_i. For each new release V, Theodore calculates an authentication tag $T_{V,i}$ for each of his customers by calling SIGN("TTTTT" + V + R_V,K_i), where R_V is the content of ttttt-V.exe. Theodore does all his development, compiling, and computing of MAC values on his laptop, and he stores the K_i keys only on his laptop; only Theodore has access to this laptop. Theodore posts the release file and all the tag files on his web site, at these URLs:

```
http://ted.com/ttttt-V.exe
http://ted.com/V-1.tag
http://ted.com/V-2.tag
...
```

He tells his customers to each fetch the new release and their tag from the web site and to call VERIFY("TTTTT"+ V + R'_V,T,K_i), where R'_V is the content of the ttttt-V.exe file they fetched and T is the content of the V-i.tag file they fetched. If the call returns ACCEPT, the customer should accept the new release; otherwise the customer should reject the new release.

14. [9 points]: Indicate the truth or falsehood of each of the following statements about Theodore's second scheme.

(Circle True or False for each choice.)

A. True / False An attacker capable of modifying the packets of the customer's transfer of the ttttt-V.exe file could do so in a way that would cause the customer's ttttt-V.exe file to be different from what Theodore intended, but would cause the customer to nevertheless accept the release.

Answer: False. VERIFY would return REJECT.

B. True / False It would be easy for an attacker who can read and modify any files on Theodore's server to cause customers to accept a release that has different contents from what Theodore intended.

Answer: False. The attacker would not be able to generate V-i.tag files that caused VERIFY to ACCEPT.

C. True / False If a customer sees see a "mirror site" of TTTTT that is not affiliated with Theodore, consisting of (for example) http://mirror.com/ttttt-V.exe and http://mirror.com/V-i.tag files, it would be just as safe to fetch those files and verify with VERIFY as it would be to fetch the files from Theodore's web site.

Answer: True. Even a malicious mirror operator would not be able to generate V-i.tag files that caused VERIFY to ACCEPT.

Theodore's third scheme is to configure his web server to use SSL with a certificate from a well-known certificate authority. Theodore's server holds a particular private key, and the certificate says that whoever knows that private key is the rightful owner of the DNS domain ted.com. Theodore tells his customers to fetch new releases from

```
https://ted.com/ttttt-V.exe
```

Theodore tells his customers that they do not have to take any special steps to check the authenticity of the software, since, if you tell a web browser to connect to https://*servername*, it will use SSL to check that the server can prove ownership of a certificate for *servername* from a well-known authority. Theodore tells his customers to check that the right URL appears in their browsers' URL box. SSL (also known as TLS) was described in lecture, and you can also read about it in section 11.10 of the textbook.

15. [9 points]: Indicate the truth or falsehood of each of the following statements about Theodore's third scheme.

(Circle True or False for each choice.)

A. True / False An attacker capable of modifying the packets of the customer's transfer of the ttttt-V.exe file could do so in a way that would cause the customer's ttttt-V.exe file to be different from what Theodore intended, but would cause the customer to nevertheless accept the release.

Answer: False. SSL protects the transfer against modification.

B. True / False It would be easy for an attacker who can read and modify any files on Theodore's server to cause customers to accept a release that has different contents from what Theodore intended.

Answer: True.

C. True / False If a customer sees a "mirror site" of TTTTT that is not affiliated with Theodore, consisting of (for example) https://mirror.com/ttttt-V.exe, it would be just as safe to fetch those files via SSL as it would be to fetch the files from Theodore's web site.

Answer: False. The browser would raise no error (after all it might really be connecting to the real mirror.com), but the mirror server could serve any data it liked, including incorrect data.

End of Quiz III

Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.033 Computer Systems Engineering: Spring 2009

Quiz III

There are 15 questions and 17 pages in this quiz booklet. Answer each question according to the instructions given. You have **90 minutes** to answer the questions.

Most questions are multiple-choice questions. Next to each choice, circle the word **True** or **False**, as appropriate. A correct choice will earn positive points, and a wrong choice or no choice will score 0. Some questions are harder than others and some questions earn more points than others—you may want to skim all questions before starting.

If you find a question ambiguous, be sure to write down any assumptions you make. **Be neat and legible.** If we can't understand your answer, we can't give you credit!

Write your name in the space below AND at the bottom of each page of this booklet.

THIS IS AN OPEN BOOK, OPEN NOTES QUIZ.
NO PHONES, NO COMPUTERS, NO LAPTOPS, NO PDAS, ETC.

CIRCLE your recitation section number:

10:00	1. Girod/Badirkhanli	
11:00	2. Girod/Badirkhanli	3. Zeldovich/Reid
12:00		4. Zeldovich/Reid
1:00	5. Jackson/Benjamin	6. Newton/Dowgun
2:00	7. Jackson/Benjamin	7. Newton/Dowgun

Do not write in the boxes below

1-3 (xx/16)	4-7 (xx/32)	8-11 (xx/28)	12-15 (xx/24)	Total (xx/100)

Name:

I Reading Questions

1. [4 points]: Based on the description of the Witty worm in "Exploiting Underlying Structure for Detailed Reconstruction of an Internet-Scale Event", by Kumar, Paxson and Weaver (reading #18), which of the following are true?

(Circle True or False for each choice.)

A. True / False Bugs in the worm's design made Witty's behavior harder to analyze. easy

B. True / False To remain effective at detecting worms, it is important for network telescopes to keep their IP address ranges secret.

among perfect worms but some things harder

go w/ most likely

2. [4 points]: Which of the following hints appear in Butler Lampson's "Hints for Computer System Design" paper (reading #20), possibly in different words? Mark each True if it appears in the paper, and False if it does not.

(Circle True or False for each choice.)

A. True / False Keep secrets in an implementation, hiding from clients aspects that might change.

B. True / False Implementations are more important than interfaces, because implementations determine performance.

C. True / False On coding: don't get it right, get it written; you can always fix it later.

D. True / False Keep caches small: when in doubt, flush it out.

adaptive

3. [8 points]: Based on the paper "Why Cryptosystems Fail", by Ross Anderson (reading #17), which of the following are true?

(Circle True or False for each choice.)

A. True / False The paper argues that the traditional threat model for cryptosystems is wrong.

B. True / False The paper argues that secure systems cannot be designed the same way safety critical systems are.

C. True / False ATM security breaches require that the thief determine both your account number and PIN.

D. True / False For one bank's ATM network to provide access for a user from another bank, both banks must know the PIN key corresponding to the user.

Complicated key system

not about goal of paper - but middles this

Name: Wilda Males me like the precise useful difference

4. [8 points]: Based on the description of System R in the paper "The Recovery Manager of the System R Database Manager" by Gray, McJones, et al. (reading #21), which of the following are true? **(Circle True or False for each choice.)**

A. **True / False** RAM buffering of disk I/O helps ensure atomicity.

B. **True / False** Shadow copies, without a log, are sufficient to ensure atomicity in the presence of concurrent transactions that both update the same file.

C. **True / False** A transaction is guaranteed to survive a crash once its log entry is written to memory.

D. **True / False** Uncommitted transactions may have issued writes *before* the last checkpoint. Therefore checkpoints may include incomplete transactions.

5. [8 points]: Based on the paper "Beyond Stack Smashing: Recent Advances in Exploiting Buffer Overruns", by Pincus and Baker (reading #16), which of the following are true? **(Circle True or False for each choice.)**

A. **True / False** By not allowing writes outside of the bounds of objects, Java eliminates all risk of attacks based on stack smashing, assuming that the VM and any native libraries are bug free.

B. **True / False** Setting the permissions on the stack memory to prevent execution of code would foil attacks based on "return into libc".

C. **True / False** Making the stack begin at a memory location chosen randomly at runtime would foil the original stack smashing exploit.

D. **True / False** Using function pointers presents additional opportunities for arc injection.

6. [8 points]: Based on the description of ObjectStore in the paper "The ObjectStore Database System" by Lamb, Landis, et al. (reading #23), state whether each of the following is true or false. **(Circle True or False for each choice.)**

A. **True / False** If an existing program, with its own implementation of lists and sets, wants to use ObjectStore to make its data persistent, it must switch to ObjectStore's list and set collections.

B. **True / False** ObjectStore needs to know the location of all pointers in all persistent data structures.

C. **True / False** The caching protocol assumes the programmer will obtain a lock before modifying a persistent object.

D. **True / False** The locking protocol always allows applications to execute concurrently, as long as they are not accessing the same object.

Name:

7. [8 points]: Based on the description of Porcupine in the paper "Manageability, Availability and Performance in Porcupine: A Highly Scalable Internet Mail Service" by Saito, Bershad and Levy, state whether each of the following is true or false.

(Circle True or False for each choice.)

A. True / False If all of the servers storing mailbox fragments for some user are down, the system will not be able to accept new mail for that user.

B. True / False Assume you have a large-scale Porcupine deployment, there are more concurrent users than servers, and all users have similar usage patterns. Storing more mailbox fragments for each user would reduce throughput.

C. True / False A user that fetches but does not delete their mail from a Porcupine server twice in a row can see different messages, even if no new messages are received.

D. True / False If one user's mailbox fragment list is causing too much load on one server, Porcupine can move just that user's mailbox fragment list to another server.

II BLOP

Ben Bitdiddle is building a distributed gambling system called Ben's Land Of Poker (BLOP). In BLOP, users play hands of poker (cards) against each other. Each user is given two *private* cards that the other users can't see. Three additional *public* cards (which can be seen by all users) are revealed one-by-one. Users place bets in four rounds of betting, one after users receive their two cards, and one after each public card is revealed. Bets are in dollars, are > 0, and are not more than a user's remaining balance. At the end of the fourth round of betting, the user with the best hand (according to the rules of poker) wins.

Each user in BLOP has an account with a balance that is stored on one of BLOP's servers. Different users playing in a hand may have their accounts hosted on different servers. Between hands, users can add money to an account with a credit card. BLOP credits a user's account when that user wins a hand. BLOP withdraws from a user's account whenever the user places a bet. During a given hand, one server is appointed a *leader* that is responsible for running the hand: it draws the cards, transfers money from users' accounts to a central *pot* that contains the money bet so far, and sends data to the clients.

During a hand, clients talk only to the leader. The leader sends information about public and private cards to the clients, who connect from their own desktop machines, and also updates the balances of accounts stored on the disk of the non-leader servers (the *subordinates*) and the balance of the pot stored locally on the leader's disk.

The pseudocode used by the leader is as follows:

```
run_hand:
  // (1) beginning of hand
  curPot = 0
  write(pot,0)  // store value of pot on disk

  for each client c:
    // handMsg tells clients about their cards
    sendRPC handMsg(privateCards[c]) to c

  for (round in [0..3])
    // collectBets does one round of betting with clients,
    // returning each of their bets
    bets = collectBets(clients)

    for each client c:
      // servers is an array that stores the subordinate
      //    server for each client's account data
      sendRPC deductMsg(c, bets[c]) to servers[c]
      curPot = curPot + bets[c]
      write (pot, curPot)  // update value of pot on disk
      if (round != 3) // last round is just for betting
         sendRPC handMsg(publicCards[round]) to c

  winner = computeWinner(hands)
  sendRPC deductMsg(winner, -curPot) to servers[winner]
  // (2) end of hand
```

Name:

The code to process `deductMsg` on each of the servers looks as follows:

```
deductMsg(account, amt):
  prevBal = read(account)
  if (prevBal > amt):
     write(account, prevBal - amt)
  else
     write(account, 0)
```

Assume that the network uses a reliable (exactly once) RPC protocol to between the leader and the subordinate servers, such that the leader waits to receive an acknowledgment to each `sendRPC` request before proceeding. Also assume that `write` operations are atomic—that is, they either complete or do not complete, and after they complete, balances are on disk.

Initially, Ben's servers run a hand without using any transactions, logging, or special fault tolerance. If the leader does not receive an acknowledgment to an RPC within two minutes, it tells the clients the hand is aborted but takes no other recovery action. If the leader crashes, the clients eventually detect this and notify the users that the hand has aborted. Initially, the leader performs no special action to recover after a crash.

During a hand, each client is given up to two minutes to place a bet. If they do not respond within two minutes (either because they left the hand, or their machine crashed), they forfeit the hand and lose any money they may have bet (play continues for the other clients in the hand.)

8. [6 points]: Which of the following could go wrong if one of the subordinate servers crashes in the middle of a hand, assuming only one hand runs at a time:

(Circle True or False for each choice.)

A. True / False After the leader aborts the hand, and the failed subordinate restarts, it is possible for the sum of all of the on-disk balances of the users in the hand to be greater than when the hand started.

B. True / False After the leader aborts the hand, and the failed subordinate restarts, it is possible for the sum of all of the on-disk balances of the users in the hand to be less than when the hand started.

Name:

Alyssa P. Hacker tells Ben that he should use transactions and two-phase commit in his implementation of BLOP. He modifies BLOP so that reads and writes of the pot and of user accounts on the subordinates are done as a part of a transaction coordinated with two-phase commit and logs. All log writes go directly to an on-disk log. Ben's scheme operates as follows:

- Prior to beginning a hand (before the comment labeled (1)), the leader writes a start of transaction (SOT) log entry and sends each subordinate a BEGIN message. Each subordinate logs an SOT log entry.
- Prior to any update to the pot, the leader writes an UPDATE log entry. Prior to any update to a user account balance, subordinates write an UPDATE log entry.
- At the end of a hand (at the comment labeled (2)), the leader sends each subordinate a PREPARE message for the transaction. If the subordinate is participating in the transaction, it logs a PREPARED log entry and sends a YES vote to the leader. If the subordinate is not participating in the transaction (because, for example, it crashed and aborted the transaction before preparing), it sends a NO vote.
- If all subordinates vote YES, the leader logs a COMMIT log entry and sends a COMMIT message to each of the subordinates. Subordinates log a COMMIT record and send an ACK message.
- Otherwise, the leader logs an ABORT log entry and sends a ABORT message to each of the subordinates. Subordinates log an ABORT record, roll back the transaction, and send an ACK message.

Assume Alyssa's additions to Ben's code correctly implement two-phase commit, and that the system uses the standard two-phase commit and log-based recovery protocols for handling and detecting both leader and subordinate failures and recovery. Both two-phase commit and log-based recovery were discussed in lecture. Two-phase commit is described in Section 9.6.3 of the course notes, and log-based recovery is described in Section 9.3.3 and 9.3.4 of the course notes.

Ben also modifies his implementation so that if one of the subordinates doesn't respond to a `deductMsg` RPC, the leader initiates transaction abort.

Name:

9. [9 points]:

Which of the following statements about the fault tolerance properties of Ben's BLOP system with two-phase commit are true?

(Circle True or False for each choice.)

A. True / False If a subordinate crashes after the leader has logged a COMMIT, and then the subordinate completes recovery, and the leader notifies all subordinates of the outcome of the transaction, it is possible for the sum of all of the balances of the users in the hand to be less than when the hand started.

B. True / False If the leader crashes before it has logged a COMMIT and then completes recovery and notifies all subordinates of the outcome of the transaction, the sum of all of the balances of the users in the hand is guaranteed to be equal to the sum of their balances when the hand started.

C. True / False If the leader crashes after one the subordinates has logged a PREPARE, it is OK for that non-leader to commit the transaction, since the transaction must have completed on the subordinate.

10. [4 points]: Ben runs his system with 2 subordinates and 1 separate leader. Suppose that the mean time to failure of a subordinate in Ben's system is 1000 minutes, and the time for a subordinate to recover is 1 minute, and that failures of nodes are independent. Assuming that each hand uses both subordinates, and that the leader doesn't fail, the availability of Ben's system is approximately:

(Circle the BEST answer)

A. 499/500

B. 999/1000

C. 999/2000

D. 1/1000

Name:

To increase the fault-tolerance of the system, Ben decides to add replication, where there are are two replicas of each subordinate.

Ben's friend Dana Bass suggests an implementation where one replica of each subordinate is appointed the *master*. The leader sends messages only to masters, and each master sends the balance of any accounts it hosts that were updated in a transaction to the other *worker* replica, after it receives the COMMIT message for that transaction. Masters do not wait for an acknowledgment from their worker before beginning to process the next transaction.

When a master fails, its worker can take over for it, becoming the master. When the failed replica recovers, it simply copies the balance of all bank accounts from the new master and becomes the worker. Dana's implementation does nothing special to deal with the case where a COMMIT completes on a master and the master fails before sending the transaction to the worker, which can result in the worker taking over without learning about the most recent committed transaction.

11. [9 points]: Which of the following statements about this approach are true, assuming that failures of masters and workers are independent, and that the leader node never fails:

(Circle True or False for each choice.)

A. True / False Dana's approach improves the availability (that is, the probability that some subordinate responds to `deductMsg` for a given client's account) versus a non-replicated system, as long as the worker node can take over for a failed master in less than the time it takes for the master to restart.

B. True / False Dana's implementation ensures single-copy serializability, since a user can never see results of hands that reveal that the system is replicated.

C. True / False Suppose Dana modifies her approach to have three replicas for each subordinate (two workers and a master.) Compared to the the approach with two replicas per subordinate, this three node approach has lower availability since the probability that one of the three replicas crashes is higher than the probability that one of two replicas crashes in the original version.

Name:

III BitPot

In order to back up your laptop's files, you sign up with BitPot. BitPot is an Internet-based storage service. They offer an RPC interface through which you can read and write named files. BitPot gives each of their customers an identification number (cid, an integer). The RPC interface looks like:

```
putfile(cid, filename, content)
getfile(cid, filename) -> content
```

`putfile()` and `getfile()` send their arguments over a network connection to the BitPot server, and wait for a reply.

BitPot provides a separate file namespace for each `cid`; for example, `getfile(1, "x")` and `getfile(2, "x")` will retrieve different data. Neither BitPot nor `getfile()` / `putfile()` do anything special to provide security. Here is what the BitPot server's RPC handlers do:

```
putfile_handler(cid, filename, content):
  name1 = "/customers/" + cid + "/" + filename
  write content to file name1 on the BitPot server's disk
  return a success indication

getfile_handler(cid, filename):
  name1 = "/customers/" + cid + "/" + filename
  if file name1 exists on the BitPot server's disk:
    content = read file name1
    return content
  else:
    return a failure indication
```

You are worried about the security of your files: that other people (perhaps even malicious BitPot employees) might be able to read or modify your backup files without your permission.

Name:

For all of the following questions, attackers have limited powers, including only the following:

- Observe any packet traveling through the network;
- Modify any packet traveling through the network;
- Send a packet with any content, including copies (perhaps modified) of packets observed on the network;
- Perform limited amounts of computation (but not enough to break cryptographic primitives);
- Read and write the contents of the BitPot server's disk (for attackers that are BitPot employees);
- Observe or modify the behavior of the BitPot server's software (for attackers that are BitPot employees);

Attackers have no powers not listed above. For example, an attacker cannot guess a cryptographic key; cannot guess the content of the files on your laptop; cannot observe or modify computations on your laptop; and cannot exploit buffer overruns or other bugs on your laptop or BitPot's servers (such as manipulating path names used to read and write files).

You should assume that there are no failures (except to the extent that the attacker's powers allow the attacker to do things that might be construed as failures).

Scheme One

You decide to encrypt each file you send to BitPot with a key that only you know, using a shared-secret cipher (see section 11.4.2 "Properties of ENCRYPT and DECRYPT" in the course notes). When you want to back up a file to BitPot, you call `backup1()`:

```
backup1(filename):
  plaintext = read contents of filename from your laptop's disk
  ciphertext = ENCRYPT(plaintext, K)
  putfile(cid, filename, ciphertext)
```

and when you need to retrieve a file from BitPot, you call `retrieve1()`:

```
retrieve1(filename):
  ciphertext = getfile(cid, filename)
  plaintext = DECRYPT(ciphertext, K)
  print plaintext
```

`cid` is your BitPot customer ID and `K` is your cipher key.

Only you and your laptop know `K`. `ENCRYPT` and `DECRYPT` withstand all the attacks mentioned in 11.4.2.

12. [8 points]: Which of the following are true about Scheme One?
(Circle True or False for each choice.)

A. True / False `retrieve1(f)` will return exactly the same data that your most recent completed call to `backup1(f)` for the same `f` read from your laptop's disk, despite anything an attacker might do.

B. True / False Eavesdroppers watching packets on the network may see ciphertext but are very unlikely to be able to figure out the plaintext content of your files.

C. True / False BitPot's employees may see ciphertext but are very unlikely to be able to figure out the plaintext content of your files.

D. True / False If someone modifies one of the files BitPot stores for you, `retrieve1()` is guaranteed to print random data (or to signal an error).

Name:

Scheme Two

Your friend Belyssa says you need to use authentication, using `SIGN` and `VERIFY` as described in section 11.3.4 of the course notes. She's not sure quite how best to do this, and suggests the following plan.

Belyssa's plan operates at the RPC layer, beneath `putfile()` / `getfile()`. The client (your laptop) and the server (BitPot) each have a shared-secret signing key (`Kc` and `Ks`, respectively). Each `SIGN`s each RPC message it sends, and `VERIFY`s each RPC message it receives. Each of them ignores any received message that doesn't verify. For simplicity, assume there is only one client and only one server, so that the server doesn't have to manage a table of per-client keys. The client and server both know both `Kc` and `Ks`.

```
// client putfile() and getfile() send requests like this:
send_request(msg):
  T = SIGN(msg, Kc)
  send {msg, T} to BitPot

// the server calls this with each incoming network message:
receive_request(msg, T):
  if VERIFY(msg, T, Kc) == ACCEPT:
    result = call putfile_handler() or getfile_handler()
    send_reply(result)
  else:
    // ignore the request

send_reply(msg):
  T = SIGN(msg, Ks)
  send {msg, T} to client

// the client calls this with each incoming network message:
receive_reply(msg, T):
  if VERIFY(msg, T, Ks) == ACCEPT:
    process msg (i.e. tell getfile() or putfile() about the reply)
  else:
    // ignore the reply
```

Name:

Only your laptop and BitPot's server know `Kc` and `Ks`. `SIGN` and `VERIFY` withstand all the attacks mentioned in 11.3.4.

You execute the following procedure on your laptop using Scheme Two:

```
test():
  write "aaaa" to file xx
  backup1(xx)
  write "bbbb" to file yy
  backup1(yy)
  write "cccc" to file yy
  backup1(yy)
  retrieve1(yy)
```

That is, you back up file `xx`, then you back up two different versions of `yy`, then you retrieve `yy`. `test()` will print a value (from the call to `retrieve1()`).

13. [7 points]: Which of the following values is it possible for `test` to print?
(Circle ALL that apply)

A. `aaaa`

B. `bbbb`

C. `cccc`

D. `(^.^) insert witty message here (^.^)`

Name:

Scheme Three

Your other friend, Allen, is uneasy about the properties of Belyssa's scheme. He proposes eliminating Belyssa's changes, and instead `SIGN`ing and `VERIFY`ing only in the backup and retrieve procedures.

Allen's new backup (called `backup2()`) includes `SIGN`'s authentication tag in the "content" it sends to BitPot, and Allen's new retrieve extracts and `VERIFY`s the tag from the data sent by BitPot.

```
backup2(filename):
  plaintext = read contents of filename from your laptop's disk
  ciphertext = ENCRYPT(plaintext, K)
  T = SIGN(ciphertext, Kx)
  what = {ciphertext, T}
  putfile(cid, filename, what)

retrieve2(filename):
  what = getfile(cid, filename)
  {ciphertext, T} = what
  if VERIFY(ciphertext, T, Kx) == ACCEPT:
    plaintext = DECRYPT(ciphertext, K)
    print plaintext
  else:
    // ignore the getfile reply
```

`K` is the shared-secret cipher key from Scheme One, which only you and your laptop know. `Kx` is a shared-secret signing key known only to you and your laptop.

You execute the following procedure on your laptop using Scheme Three. (This is the same procedure as for the previous question, but uses `backup2()` and `retrieve2()`).

```
test2():
  write "aaaa" to file xx
  backup2(xx)
  write "bbbb" to file yy
  backup2(yy)
  write "cccc" to file yy
  backup2(yy)
  retrieve2(yy)
```

Name:

14. [7 points]: Which of the following values is it possible for `test2` to print?
(Circle ALL that apply)

A. `aaaa`

B. `bbbb`

C. `cccc`

D. `(^.^) insert witty message here (^.^)`

Name:

IV Systems Design Experience

There are known knowns.
There are things we know
that we know.

There are known unknowns.
That is to say
There are things that we now know
we don't know.

But there are also unknown unknowns,
There are things we do not know
we don't know.

15. [2 points]: The aforementioned quote is highly applicable to the design of large computer systems. According to the guest lecture on May 11, who first uttered this sage advice?
(Circle the BEST answer)

A. Albert Einstein

B. Butler Lampson

C. Mahatma Gandhi

D. Donald Rumsfeld

End of Quiz III

Please ensure that you wrote your name on the front of the quiz,
and circled your recitation section number.

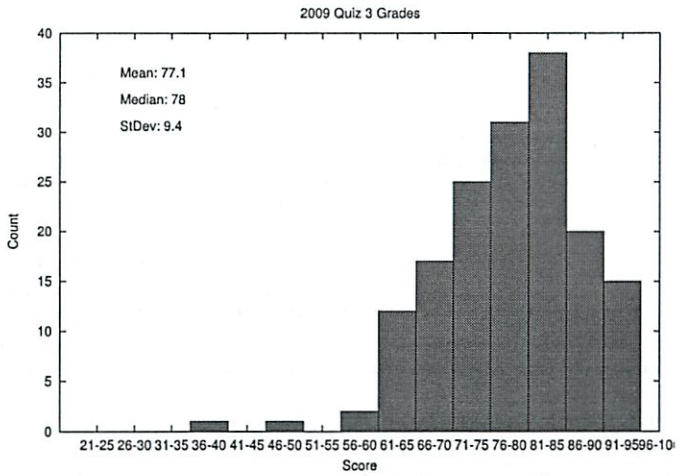
Name:

Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.033 Computer Systems Engineering: Spring 2009

Quiz III Solutions



I Reading Questions

1. [4 points]: Based on the description of the Witty worm in "Exploiting Underlying Structure for Detailed Reconstruction of an Internet-Scale Event", by Kumar, Paxson and Weaver (reading #18), which of the following are true?

(Circle True or False for each choice.)

A. **True / False** Bugs in the worm's design made Witty's behavior harder to analyze.

Answer: False. Bugs made it easier to analyze.

B. **True / False** To remain effective at detecting worms, it is important for network telescopes to keep their IP address ranges secret.

Answer: True. Otherwise worms that scan the IP address space could leave out known telescope ranges.

2. [4 points]: Which of the following hints appear in Butler Lampson's "Hints for Computer System Design" paper (reading #20), possibly in different words? Mark each True if it appears in the paper, and False if it does not.

(Circle True or False for each choice.)

A. **True / False** Keep secrets in an implementation, hiding from clients aspects that might change.

Answer: True. Section 2.4 of the paper.

B. **True / False** Implementations are more important than interfaces, because implementations determine performance.

Answer: False.

C. **True / False** On coding: don't get it right, get it written; you can always fix it later.

Answer: False.

D. **True / False** Keep caches small: when in doubt, flush it out.

Answer: False.

3. [8 points]: Based on the paper "Why Cryptosystems Fail", by Ross Anderson (reading #17), which of the following are true?

(Circle True or False for each choice.)

A. True / False The paper argues that the traditional threat model for cryptosystems is wrong.

Answer: True.

B. True / False The paper argues that secure systems cannot be designed the same way safety critical systems are.

Answer: False. The paper argues that these can be designed the same way, making specific reference to models for designing planes and trains.

C. True / False ATM security breaches require that the thief determine both your account number and PIN.

Answer: False. Many of the examples in the paper involve other kinds of attacks at various stages of the ATM's processing.

D. True / False For one bank's ATM network to provide access for a user from another bank, both banks must know the PIN key corresponding to the user.

Answer: False. The PIN key is kept local to the bank.

4. [8 points]: Based on the description of System R in the paper "The Recovery Manager of the System R Database Manager" by Gray, McJones, et al. (reading #21), which of the following are true?

(Circle True or False for each choice.)

A. True / False RAM buffering of disk I/O helps ensure atomicity.

Answer: False. Buffering makes atomicity more complex.

B. True / False Shadow copies, without a log, are sufficient to ensure atomicity in the presence of concurrent transactions that both update the same file.

Answer: False. System R requires the incremental log to ensure transaction consistency after a crash.

C. True / False A transaction is guaranteed to survive a crash once its log entry is written to memory.

Answer: False. The log entry is not stable until written to disk.

D. True / False Uncommitted transactions may have issued writes *before* the last checkpoint. Therefore checkpoints may include incomplete transactions.

Answer: True.

5. [8 points]: Based on the paper "Beyond Stack Smashing: Recent Advances in Exploiting Buffer Overruns", by Pincus and Baker (reading #16), which of the following are true?

(Circle True or False for each choice.)

A. True / False By not allowing writes outside of the bounds of objects, Java eliminates all risk of attacks based on stack smashing, assuming that the VM and any native libraries are bug free.

Answer: True.

B. True / False Setting the permissions on the stack memory to prevent execution of code would foil attacks based on "return into libc".

Answer: False. The "return into libc" attack executes preexisting functions (i.e., `system()`), which do not reside on the stack.

C. True / False Making the stack begin at a memory location chosen randomly at runtime would foil the original stack smashing exploit.

Answer: True.

D. True / False Using function pointers presents additional opportunities for arc injection.

Answer: True.

6. [8 points]: Based on the description of ObjectStore in the paper "The ObjectStore Database System" by Lamb, Landis, et al. (reading #23), state whether each of the following is true or false.

(Circle True or False for each choice.)

A. True / False If an existing program, with its own implementation of lists and sets, wants to use ObjectStore to make its data persistent, it must switch to ObjectStore's list and set collections.

Answer: False.

B. True / False ObjectStore needs to know the location of all pointers in all persistent data structures.

Answer: True.

C. True / False The caching protocol assumes the programmer will obtain a lock before modifying a persistent object.

Answer: False.

D. True / False The locking protocol always allows applications to execute concurrently, as long as they are not accessing the same object.

Answer: False. Locking is performed on a page granularity in ObjectStore, hence no two locked objects in the same page may be accessed concurrently.

7. [8 points]: Based on the description of Porcupine in the paper "Manageability, Availability and Performance in Porcupine: A Highly Scalable Internet Mail Service" by Saito, Bershad and Levy, state whether each of the following is true or false.

(Circle True or False for each choice.)

A. True / False If all of the servers storing mailbox fragments for some user are down, the system will not be able to accept new mail for that user.

Answer: False. The system may write incoming mail to new fragments on alternate servers.

B. True / False Assume you have a large-scale Porcupine deployment, there are more concurrent users than servers, and all users have similar usage patterns. Storing more mailbox fragments for each user would reduce throughput.

Answer: True. Reading and writing to additional servers will require more disk accesses, and more queries when fetching mail for a user.

C. True / False A user that fetches but does not delete their mail from a Porcupine server twice in a row can see different messages, even if no new messages are received.

Answer: True. Messages are propagated "lazily" in Porcupine, and may take some time to be communicated between servers.

D. True / False If one user's mailbox fragment list is causing too much load on one server, Porcupine can move just that user's mailbox fragment list to another server.

Answer: False. Fragment lists are mapped to servers at a coarser granularity than individual users, and hence users cannot be moved individually.

II BLOP

Ben Bitdiddle is building a distributed gambling system called Ben's Land Of Poker (BLOP). In BLOP, users play hands of poker (cards) against each other. Each user is given two *private* cards that the other users can't see. Three additional *public* cards (which can be seen by all users) are revealed one-by-one. Users place bets in four rounds of betting, one after users receive their two cards, and one after each public card is revealed. Bets are in dollars, are > 0, and are not more than a user's remaining balance. At the end of the fourth round of betting, the user with the best hand (according to the rules of poker) wins.

Each user in BLOP has an account with a balance that is stored on one of BLOP's servers. Different users playing in a hand may have their accounts hosted on different servers. Between hands, users can add money to an account with a credit card. BLOP credits a user's account when that user wins a hand. BLOP withdraws from a user's account whenever the user places a bet. During a given hand, one server is appointed a *leader* that is responsible for running the hand: it draws the cards, transfers money from users' accounts to a central *pot* that contains the money bet so far, and sends data to the clients.

During a hand, clients talk only to the leader. The leader sends information about public and private cards to the clients, who connect from their own desktop machines, and also updates the balances of accounts stored on the disk of the non-leader servers (the *subordinates*) and the balance of the pot stored locally on the leader's disk.

The pseudocode used by the leader is as follows:

```
run_hand:
  // (1) beginning of hand
  curPot = 0
  write(pot,0)  // store value of pot on disk

  for each client c:
    // handMsg tells clients about their cards
    sendRPC handMsg(privateCards[c]) to c

  for (round in [0..3])
    // collectBets does one round of betting with clients,
    // returning each of their bets
    bets = collectBets(clients)

    for each client c:
      // servers is an array that stores the subordinate
      //    server for each client's account data
      sendRPC deductMsg(c, bets[c]) to servers[c]
      curPot = curPot + bets[c]
      write (pot, curPot)  // update value of pot on disk
      if (round != 3) // last round is just for betting
         sendRPC handMsg(publicCards[round]) to c

  winner = computeWinner(hands)
  sendRPC deductMsg(winner, -curPot) to servers[winner]
  // (2) end of hand
```

The code to process `deductMsg` on each of the servers looks as follows:

```
deductMsg(account, amt):
  prevBal = read(account)
  if (prevBal > amt):
     write(account, prevBal - amt)
  else
     write(account, 0)
```

Assume that the network uses a reliable (exactly once) RPC protocol to between the leader and the subordinate servers, such that the leader waits to receive an acknowledgment to each `sendRPC` request before proceeding. Also assume that `write` operations are atomic—that is, they either complete or do not complete, and after they complete, balances are on disk.

Initially, Ben's servers run a hand without using any transactions, logging, or special fault tolerance. If the leader does not receive an acknowledgment to an RPC within two minutes, it tells the clients the hand is aborted but takes no other recovery action. If the leader crashes, the clients eventually detect this and notify the users that the hand has aborted. Initially, the leader performs no special action to recover after a crash.

During a hand, each client is given up to two minutes to place a bet. If they do not respond within two minutes (either because they left the hand, or their machine crashed), they forfeit the hand and lose any money they may have bet (play continues for the other clients in the hand.)

8. [6 points]: Which of the following could go wrong if one of the subordinate servers crashes in the middle of a hand, assuming only one hand runs at a time:

(Circle True or False for each choice.)

A. True / False After the leader aborts the hand, and the failed subordinate restarts, it is possible for the sum of all of the on-disk balances of the users in the hand to be greater than when the hand started.

Answer: False. The leader only deducts from user accounts during the hands.

B. True / False After the leader aborts the hand, and the failed subordinate restarts, it is possible for the sum of all of the on-disk balances of the users in the hand to be less than when the hand started.

Answer: True.

Alyssa P. Hacker tells Ben that he should use transactions and two-phase commit in his implementation of BLOP. He modifies BLOP so that reads and writes of the pot and of user accounts on the subordinates are done as a part of a transaction coordinated with two-phase commit and logs. All log writes go directly to an on-disk log. Ben's scheme operates as follows:

- Prior to beginning a hand (before the comment labeled (1)), the leader writes a start of transaction (SOT) log entry and sends each subordinate a BEGIN message. Each subordinate logs an SOT log entry.
- Prior to any update to the pot, the leader writes an UPDATE log entry. Prior to any update to a user account balance, subordinates write an UPDATE log entry.
- At the end of a hand (at the comment labeled (2)), the leader sends each subordinate a PREPARE message for the transaction. If the subordinate is participating in the transaction, it logs a PREPARED log entry and sends a YES vote to the leader. If the subordinate is not participating in the transaction (because, for example, it crashed and aborted the transaction before preparing), it sends a NO vote.
- If all subordinates vote YES, the leader logs a COMMIT log entry and sends a COMMIT message to each of the subordinates. Subordinates log a COMMIT record and send an ACK message.
- Otherwise, the leader logs an ABORT log entry and sends a ABORT message to each of the subordinates. Subordinates log an ABORT record, roll back the transaction, and send an ACK message.

Assume Alyssa's additions to Ben's code correctly implement two-phase commit, and that the system uses the standard two-phase commit and log-based recovery protocols for handling and detecting both leader and subordinate failures and recovery. Both two-phase commit and log-based recovery were discussed in lecture. Two-phase commit is described in Section 9.6.3 of the course notes, and log-based recovery is described in Section 9.3.3 and 9.3.4 of the course notes.

Ben also modifies his implementation so that if one of the subordinates doesn't respond to a `deductMsg` RPC, the leader initiates transaction abort.

9. [9 points]:

Which of the following statements about the fault tolerance properties of Ben's BLOP system with two-phase commit are true?

(Circle True or False for each choice.)

A. True / False If a subordinate crashes after the leader has logged a COMMIT, and then the subordinate completes recovery, and the leader notifies all subordinates of the outcome of the transaction, it is possible for the sum of all of the balances of the users in the hand to be less than when the hand started.

Answer: False. All the subordinates will COMMIT, they will all complete their updates as directed by `run_hand`, and `run_hand` ensures that the sum of the balances ends up unchanged.

B. True / False If the leader crashes before it has logged a COMMIT and then completes recovery and notifies all subordinates of the outcome of the transaction, the sum of all of the balances of the users in the hand is guaranteed to be equal to the sum of their balances when the hand started.

Answer: True. In both the COMMIT and ABORT cases, the sum remains the same.

C. True / False If the leader crashes after one the subordinates has logged a PREPARE, it is OK for that non-leader to commit the transaction, since the transaction must have completed on the subordinate.

Answer: False. It is only safe for a subordinate to commit if *all* subordinates vote YES.

10. [4 points]: Ben runs his system with 2 subordinates and 1 separate leader. Suppose that the mean time to failure of a subordinate in Ben's system is 1000 minutes, and the time for a subordinate to recover is 1 minute, and that failures of nodes are independent. Assuming that each hand uses both subordinates, and that the leader doesn't fail, the availability of Ben's system is approximately:

(Circle the BEST answer)

A. 499/500

B. 999/1000

C. 999/2000

D. 1/1000

Answer: 499/500. To a first approximation one or the other subordinate will be unavailable for 2 minutes out of 1000.

To increase the fault-tolerance of the system, Ben decides to add replication, where there are are two replicas of each subordinate.

Ben's friend Dana Bass suggests an implementation where one replica of each subordinate is appointed the *master*. The leader sends messages only to masters, and each master sends the balance of any accounts it hosts that were updated in a transaction to the other *worker* replica, after it receives the COMMIT message for that transaction. Masters do not wait for an acknowledgment from their worker before beginning to process the next transaction.

When a master fails, its worker can take over for it, becoming the master. When the failed replica recovers, it simply copies the balance of all bank accounts from the new master and becomes the worker. Dana's implementation does nothing special to deal with the case where a COMMIT completes on a master and the master fails before sending the transaction to the worker, which can result in the worker taking over without learning about the most recent committed transaction.

11. [9 points]: Which of the following statements about this approach are true, assuming that failures of masters and workers are independent, and that the leader node never fails:

(Circle True or False for each choice.)

A. True / False Dana's approach improves the availability (that is, the probability that some subordinate responds to `deductMsg` for a given client's account) versus a non-replicated system, as long as the worker node can take over for a failed master in less than the time it takes for the master to restart.

Answer: True.

B. True / False Dana's implementation ensures single-copy serializability, since a user can never see results of hands that reveal that the system is replicated.

Answer: False. For example, suppose a master fails after replying to a COMMIT for a balance deduction but before sending the updated balance to the worker. The worker will then take over as master with a balance that is too high. This could not have happened in the non-replicated system.

C. True / False Suppose Dana modifies her approach to have three replicas for each subordinate (two workers and a master.) Compared to the the approach with two replicas per subordinate, this three node approach has lower availability since the probability that one of the three replicas crashes is higher than the probability that one of two replicas crashes in the original version.

Answer: False. The system has higher availability, since it is available if any one of the three replicas is alive.

III BitPot

In order to back up your laptop's files, you sign up with BitPot. BitPot is an Internet-based storage service. They offer an RPC interface through which you can read and write named files. BitPot gives each of their customers an identification number (cid, an integer). The RPC interface looks like:

```
putfile(cid, filename, content)
getfile(cid, filename) -> content
```

`putfile()` and `getfile()` send their arguments over a network connection to the BitPot server, and wait for a reply.

BitPot provides a separate file namespace for each `cid`; for example, `getfile(1, "x")` and `getfile(2, "x")` will retrieve different data. Neither BitPot nor `getfile()` / `putfile()` do anything special to provide security. Here is what the BitPot server's RPC handlers do:

```
putfile_handler(cid, filename, content):
  name1 = "/customers/" + cid + "/" + filename
  write content to file name1 on the BitPot server's disk
  return a success indication

getfile_handler(cid, filename):
  name1 = "/customers/" + cid + "/" + filename
  if file name1 exists on the BitPot server's disk:
    content = read file name1
    return content
  else:
    return a failure indication
```

You are worried about the security of your files: that other people (perhaps even malicious BitPot employees) might be able to read or modify your backup files without your permission.

For all of the following questions, attackers have limited powers, including only the following:

- Observe any packet traveling through the network;
- Modify any packet traveling through the network;
- Send a packet with any content, including copies (perhaps modified) of packets observed on the network;
- Perform limited amounts of computation (but not enough to break cryptographic primitives);
- Read and write the contents of the BitPot server's disk (for attackers that are BitPot employees);
- Observe or modify the behavior of the BitPot server's software (for attackers that are BitPot employees);

Attackers have no powers not listed above. For example, an attacker cannot guess a cryptographic key; cannot guess the content of the files on your laptop; cannot observe or modify computations on your laptop; and cannot exploit buffer overruns or other bugs on your laptop or BitPot's servers (such as manipulating path names used to read and write files).

You should assume that there are no failures (except to the extent that the attacker's powers allow the attacker to do things that might be construed as failures).

Scheme One

You decide to encrypt each file you send to BitPot with a key that only you know, using a shared-secret cipher (see section 11.4.2 "Properties of ENCRYPT and DECRYPT" in the course notes). When you want to back up a file to BitPot, you call `backup1()`:

```
backup1(filename):
  plaintext = read contents of filename from your laptop's disk
  ciphertext = ENCRYPT(plaintext, K)
  putfile(cid, filename, ciphertext)
```

and when you need to retrieve a file from BitPot, you call `retrieve1()`:

```
retrieve1(filename):
  ciphertext = getfile(cid, filename)
  plaintext = DECRYPT(ciphertext, K)
  print plaintext
```

`cid` is your BitPot customer ID and `K` is your cipher key.

Only you and your laptop know `K`. `ENCRYPT` and `DECRYPT` withstand all the attacks mentioned in 11.4.2.

12. [8 points]: Which of the following are true about Scheme One?
(Circle True or False for each choice.)

A. True / False `retrieve1(f)` will return exactly the same data that your most recent completed call to `backup1(f)` for the same `f` read from your laptop's disk, despite anything an attacker might do.

Answer: False. For example, if an attacker modifies the contents your data on BitPot's disks, `retrieve1()` will produce something other than the original content of your file.

B. True / False Eavesdroppers watching packets on the network may see ciphertext but are very unlikely to be able to figure out the plaintext content of your files.

Answer: True.

C. True / False BitPot's employees may see ciphertext but are very unlikely to be able to figure out the plaintext content of your files.

Answer: True.

D. True / False If someone modifies one of the files BitPot stores for you, `retrieve1()` is guaranteed to print random data (or to signal an error).

Answer: False. For example, suppose you back up two files to BitPot, f1 and f2. If a malicious BitPot employee copies /customers/cid/f1 to /customers/cid/f2, then `retrieve1(f2)` will yield the backed-up content of f1.

Scheme Two

Your friend Belyssa says you need to use authentication, using `SIGN` and `VERIFY` as described in section 11.3.4 of the course notes. She's not sure quite how best to do this, and suggests the following plan.

Belyssa's plan operates at the RPC layer, beneath `putfile()` / `getfile()`. The client (your laptop) and the server (BitPot) each have a shared-secret signing key (`Kc` and `Ks`, respectively). Each `SIGN`s each RPC message it sends, and `VERIFY`s each RPC message it receives. Each of them ignores any received message that doesn't verify. For simplicity, assume there is only one client and only one server, so that the server doesn't have to manage a table of per-client keys. The client and server both know both `Kc` and `Ks`.

```
// client putfile() and getfile() send requests like this:
send_request(msg):
  T = SIGN(msg, Kc)
  send {msg, T} to BitPot

// the server calls this with each incoming network message:
receive_request(msg, T):
  if VERIFY(msg, T, Kc) == ACCEPT:
    result = call putfile_handler() or getfile_handler()
    send_reply(result)
  else:
    // ignore the request

send_reply(msg):
  T = SIGN(msg, Ks)
  send {msg, T} to client

// the client calls this with each incoming network message:
receive_reply(msg, T):
  if VERIFY(msg, T, Ks) == ACCEPT:
    process msg (i.e. tell getfile() or putfile() about the reply)
  else:
    // ignore the reply
```

Only your laptop and BitPot's server know Kc and Ks. SIGN and VERIFY withstand all the attacks mentioned in 11.3.4.

You execute the following procedure on your laptop using Scheme Two:

```
test():
  write "aaaa" to file xx
  backup1(xx)
  write "bbbb" to file yy
  backup1(yy)
  write "cccc" to file yy
  backup1(yy)
  retrieve1(yy)
```

That is, you back up file xx, then you back up two different versions of yy, then you retrieve yy. test() will print a value (from the call to retrieve1()).

13. [7 points]: Which of the following values is it possible for test to print?
(Circle ALL that apply)

A. aaaa

Answer: aaaa is possible. A BitPot employee could copy the backed up xx to the backed up yy.

B. bbbb

Answer: bbbb is possible. A BitPot employee could save the old backed up yy and copy it to the backed up yy after the final backup1(yy).

C. cccc

Answer: cccc is possible.

D. (^.^) insert witty message here (^.^)

Answer: The properties of SIGN/VERIFY and ENCRYPT/DECRYPT allow this possibility for attackers who are BitPot employees. However, with practical implementations of ENCRYPT/DECRYPT, it might be difficult for an attacker to produce this result.

Scheme Three

Your other friend, Allen, is uneasy about the properties of Belyssa's scheme. He proposes eliminating Belyssa's changes, and instead SIGNing and VERIFYing only in the backup and retrieve procedures.

Allen's new backup (called backup2()) includes SIGN's authentication tag in the "content" it sends to BitPot, and Allen's new retrieve extracts and VERIFYs the tag from the data sent by BitPot.

```
backup2(filename):
  plaintext = read contents of filename from your laptop's disk
  ciphertext = ENCRYPT(plaintext, K)
  T = SIGN(ciphertext, Kx)
  what = {ciphertext, T}
  putfile(cid, filename, what)

retrieve2(filename):
  what = getfile(cid, filename)
  {ciphertext, T} = what
  if VERIFY(ciphertext, T, Kx) == ACCEPT:
    plaintext = DECRYPT(ciphertext, K)
    print plaintext
  else:
    // ignore the getfile reply
```

K is the shared-secret cipher key from Scheme One, which only you and your laptop know. Kx is a shared-secret signing key known only to you and your laptop.

You execute the following procedure on your laptop using Scheme Three. (This is the same procedure as for the previous question, but uses backup2() and retrieve2()).

```
test2():
  write "aaaa" to file xx
  backup2(xx)
  write "bbbb" to file yy
  backup2(yy)
  write "cccc" to file yy
  backup2(yy)
  retrieve2(yy)
```

14. [7 points]: Which of the following values is it possible for `test2` to print?
(Circle ALL that apply)

A. `aaaa`

Answer: `aaaa` is possible; the example from the previous question works.

B. `bbbb`

Answer: `bbbb` is possible; the example from the previous question works.

C. `cccc`

Answer: `cccc` is possible.

D. `(^.^) insert witty message here (^.^)`

Answer: not possible. The VERIFY in `retrieve2()` will reject any content not previously signed by the laptop.

IV Systems Design Experience

There are known knowns.
There are things we know
that we know.

There are known unknowns.
That is to say
There are things that we now know
we don't know.

But there are also unknown unknowns,
There are things we do not know
we don't know.

15. [2 points]: The aforementioned quote is highly applicable to the design of large computer systems. According to the guest lecture on May 11, who first uttered this sage advice?
(Circle the BEST answer)

A. Albert Einstein

B. Butler Lampson

C. Mahatma Gandhi

D. Donald Rumsfeld

Answer: Donald Rumsfeld.

End of Quiz III