Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.824 Spring 2005

Midterm Exam

Please write your name on this cover sheet and on any exam pages you detach from this sheet.

Some questions may be much harder than others. Read them all through first and attack them in the order that allows you to make the most progress. If you find a question ambiguous, be sure to write down any assumptions you make. Be neat. If we can't understand your answer, we can't give you credit.

There's a feedback form at the end of the exam which we'd like you to fill out if you have time.

You have 80 minutes to complete this exam.

THIS IS AN OPEN BOOK, OPEN NOTES EXAM.

Please do not write in the boxes below.

1 (xx/20)	2 (xx/40)	3 (xx/20)	4 (xx/40)	5 (xx/30)	Total (xx/150)
~~10~~ 20	37	20	38	24	~~129~~ 139 AM

Name: DAN PORTS

I Part One

10

1. [10 points]: Suppose you're running Mach 3.0 on a MIPS R3000. You're getting tired of how slow your 15-year-old CPU is. You hear that Anderson et al., authors of the Interaction paper, have started a company that sells a MIPS R3000-compatible CPU called the Mathom2000. The Mathom2000 runs operating system primitives about 100 times as fast as the R3000, and application code at the same speed as the R3000. If you upgraded to the Mathom2000, approximately how much faster overall would your programs run? Explain your answer based on evidence presented in the Interaction paper.

Under Mach 3.0, applications spend 15-20% of their time executing OS primitives (Table 7). Let's be conservative and assume 15%. With OS primitives 100 times faster, a program that runs in time t originally spends $.15t$ in the OS and $.85t$ executing app code; after the change it spends $.0015t$ in OS and $.85t$ in the app, for a total of $.8515t$. This is just under 15% faster.

10

2. [10 points]: Look at Figure 3 of Backtracking Intrusions, by King and Chen. The paper says that GraphGen ignores the event at time 7. Why is that correct? Why couldn't the contents of file 2 have been part of the attack?

The detection point is file X at time 10. The last time this file was written was by process C at time 6. C does not read file 2 until time 7, so it cannot have used its contents at time 7.

II Part Two: Porcupine

-10 **3. [10 points]:** Look at Figure 6 of the Porcupine paper by Saito et al. When skew is zero, most of the policies process about 700 to 800 messages per second. Suppose the CPUs were replaced with CPUs ten times as fast. Approximately how many more messages per second would the system be able to process? Why?

Not substantially more — the primary bottleneck is disk access time, not CPU speed.

10 **4. [10 points]:** Look at the S1 and S2 policies in Figure 6 when skew is 1.0. As you can see, the data points are hard to read. Based on the explanations in the rest of the paper, how many messages/second would you expect S1 to process with skew of 1.0? And how much faster than S1 would you expect S2 to be? Explain your answers.

With skew 1.0, every user hashes to the same value, and so with S_1, the entire workload is handled by one server. Figure 4 shows that a single node with one disk can process about 23 messages per second, and with S_1 this will be the throughput of the system. With S_2, the (least-loaded) of 2 servers is used, so throughput should be about twice as high.

10 **5. [10 points]:** Suppose you run a Porcupine experiment like D1 in Figure 6 with skew of 1.0. However, when the system is first started all the users' mailbox fragments are placed on the same server, node N0. Towards the beginning of the experiment, approximately how many messages/second will the Porcupine cluster be able to process? Towards the end? If there is a change in performance, explain when this change occurs, and what specific steps Porcupine takes as it is running that lead to the change.

Initially all the load falls on a single server, giving performance about that of S1 (23 msgs/s). As each user empties his mailbox, a new least-loaded node is chosen to be the new holder of the user's mailbox fragments. So by the end, the performance will essentially be the same as skew 0. (about 700-800 msgs/s).

7 **6. [10 points]:** Figure 6 suggests that the dynamic spread policy is only useful when skew is high. The authors generate the high-skew workloads by using user names that all hash to the same value. How valuable would dynamic spread be in real life? Are there any specific situations, that are likely to occur, in which the dynamic spread policy would significantly increase performance?

It's not likely that a disproportionate number of usernames will hash to the same value (assuming a decent hash function), so the value of dynamic spread is limited.

III Part Three: OK Auctions

You've been hired as a consultant by OK Auctions to help make their web site secure. OK Auctions runs an on-line auction system. At any given time there are thousands of auctions in progress. Users can log into the system, get a list of current auctions, and place bids on one or more auctions. Each auction lasts for a few days; when it ends, the highest bidder wins, and must pay the amount of his or her bid.

OK Auctions' most pressing security concern is to ensure that users do not learn each others' bids while an auction is in progress. OK Auctions wants to prevent users from placing bids that are only slightly higher than the existing high bid, since that would decrease winning bid values and thus profits. They are worried that someone might exploit a bug in their software and steal high bids.

OK Auctions current web site consists of a single process running on a UNIX server. The process accepts HTTP connections, checks usernames and passwords, maintains the list of all auctions, and maintains the state of each auction (including the current high bid). The process keeps all this information in memory (it doesn't use a separate database).

After reading Krohn's OKWS paper, OK Auctions decides to use OKWS and to split their software into multiple service programs. The LOGIN service will display a web page asking for a username and password, and check that the results are valid. The LIST service will display a list of all auctions. There will be one AUCTION service process per auction. Each AUCTION service will keep track of the current high bid for the auction, will display a page asking for a new bid to any authenticated user, and will update its internal current high bid if the new bid is higher. The service processes keep the state they need in memory (they don't use a separate database).

10 **7. [10 points]:** Will the use of OKWS with the LOGIN/LIST/AUCTION services make OK Auctions' web site more secure? If yes, explain how this arrangement would make it harder for a user to steal the high bid of an auction; if no, explain why attacks that work with the single-process design are still likely to work in the LOGIN/LIST/AUCTION arrangement.

No. Exploiting any bug in the AUCTION service would give an attacker the ability to tamper with the in-memory database, changing the high bid for an auction, obtaining the current high bid, etc.

10 **8. [10 points]:** Suggest a partition of functionality among service processes that would be more secure, and explain why it's more secure. Outline an attack that might have successfully stolen a current high bid in the LOGIN/LIST/AUCTION arrangement that would fail with your arrangement.

Create a database server that holds all auctions and their current high bids. Allow access to the DB only through a database proxy that supports only two RPCs: one that obtains a list of all auctions, and one that takes a new bid for an auction and replaces the current high bid with that bid if it's higher. Then create LIST and AUCTION services that simply validate their input, perform the appropriate RPC, and generate the result page. (For simplicity, I'm ignoring LOGIN; it could work in much the same way.)

Now a buffer overflow attack to AUCTION could only compromise the AUCTION service, which does not have access to the database. Before, the attacker could read or change the high bid; now the attacker can only send narrowly defined RPCs to the database proxy, which doesn't gain him much.

IV Part Four: Lab Four

Larry Locker is working on adding locks to his file server for Lab 4. Remember that Lab 4 had locking but no caching of blocks or locks (no REVOKE, no flush()). The point of the locking is to serialize concurrent NFS RPCs that affect the same file or directory, so that the final results are as if the RPCs executed one at a time. Thus, for example, concurrent CREATEs of the same file name would not result in two separate files with the same name.

Larry is working on SETATTR. He knows that at the end of the SETATTR he must release the lock on the file handle, reply to the RPC, and put() the attributes to the block server. But he's not sure what the right order is.

8

9. [10 points]: Please write "yes" after each of the following orders that would result in proper serialization of NFS RPCs, and "no" after each order that would not. You can assume that Larry would implement each order without bugs (for example by using callbacks to ensure the correct sequence).

`release, reply, put` N

`release, put, reply` N

`reply, release, put` N

`reply, put, release` N X Yes

`put, release, reply` Y

`put, reply, release` Y

10. [10 points]: Larry stores everything about a file (both attributes and contents) in a single block, whose key is the file handle. Larry observes that the READ RPC does not modify the file or its attributes, and thinks that means he doesn't have to acquire any locks in READ. Is Larry right? Explain why or why not, or what extra information you need to decide. If Larry is wrong, describe a specific situation in which lack of locking would lead to an incorrect file system state; include a description of the incorrect state and what the correct state should be. Remember that Larry is working on Lab 4, not Lab 5.

10

This is right. All the file data is contained in the same block, so assuming other RPCs always store consistent data in the block, the READ will always see a consistent view of the file.

11. [10 points]: Larry also thinks he doesn't need to acquire any locks for WRITE. After all, WRITE replaces bytes in the file: it doesn't need to know the old values of the bytes. Is Larry right? Explain why or why not, or what extra information you need to decide. If Larry is wrong, describe a specific situation in which lack of locking would lead to an incorrect file system state; include a description of the incorrect state and what the correct state should be.

10

This is wrong. Suppose a file contains "aa" and two WRITES arrive simultaneously, one changing the first character and one changing the second character.

W_1 and W_2 fetch "aa" from block server
W_1 writes "ba"
W_2 writes "ac".
correct result should be "bc", but
W_R performed its read before and its write after W_1's write.

12. [10 points]: Larry is considering a different design in which he stores the attributes in one block and the file contents in a different block. The file content block's key would be the file handle with an "x" added to the end. Does this design change whether READ needs to lock? How about WRITE? Explain your answers.

10

Yes. Now READ and WRITE both need to lock. WRITE needs locking for the same reason as before. READ needs locking to ensure that a WRITE isn't in progress while it's reading. If a WRITE changes the file size, this will need to update both the file size and file content. Since these are now in different blocks, the READ might see the change to only one of them if it didn't have locking, so it could return inconsistent results.

V Part Five: XOM

The XOM paper by Lie et al. describes store_secure and load_secure instructions that use encryption and hashing to protect data stored in off-chip RAM (see Sections 2.1 and 3).

10 **13. [10 points]:** XOM's stated goal is to copy-protect programs. Why does XOM need to protect *data* as well as instructions? Explain an example application in which copy-protected instructions are not useful without protected data.

If an attacker could look at a XOM-secured program's register and data values, he could feed various inputs into the XOM program, observe all of the intermediate states of the program's memory and deduce what operations are being performed, in order to reconstruct a (unprotected) copy of the code

6 **14. [10 points]:** Suppose a XOM program in compartment 17 executes store_secure, giving an address of 1024 and a 32-bit data value of 55. Assume that a cache block holds just one 32-bit data value. What information will XOM store in RAM as a result? Write each distinct value stored in RAM on a different line below (there may be fewer than four). For example, write "L0: 1024 L1: Hash(17, current time, 55)" if you think XOM stores the value 1024, followed by the value computed by a cryptographic hash of the concatenation of the three indicated arguments.

L0: Encryption of 55 with compartment 17's session key

L1: MAC(55, 17's session key, interrupt counter)

L2:

L3:

15. [10 points]: How does XOM decide whether data it fetches from off-chip RAM is correct? Assume that the address given to load_secure is *Addr*. and that the current compartment number is *Comp*. Express the steps in terms of the locations from your previous answer. For example, you might answer "XOM accepts the data in L3 if L0 is equal to the current time and L1 is equal to Comp."

Decrypt L0 with current session key; call this x.

Compute MAC of x, session key and current interrupt counter; call this y.

If y == L1
  return x
else
  exception.

End of Exam