Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.033 Computer Systems Engineering: Spring 2012

Quiz 3

There are 13 questions and 9 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. We will round up the score for every *numbered* question to 0 if it's otherwise negative (i.e., you cannot get less than 0 on a numbered question).

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. Rudolph/Grusecki		
11:00	2. Rudolph/Grusecki	3. Abelson/Gokce	4. Katabi/Joshi
12:00		5. Abelson/Gokce	6. Katabi/Joshi
1:00	7. Shavit/Moll	8. Szolovits/Fang	
2:00	9. Shavit/Moll	10. Szolovits/Fang	

Do not write in the boxes below

1-6 (xx/32)	7 (xx/16)	8 (xx/18)	9-11 (xx/12)	12 (xx/11)	13 (xx/11)	Total (xx/100)
14 om	16 FL	18 FL	12 ug	0 ug	0 ug	60 ug

Name: Michael Plasmeier

I Reading

1. [4 points]:

In the Porcupine mail service, as described in the paper, whenever there is a membership reconfiguration occurs, the following happens:

(Circle True or False for each choice.)

A. True / False The "user map" is updated remapping users to clusters.

B. True / False The fragments in the "mailbox fragment list" are combined into a single mailbox for each user.

C. True / False Soft state is recomputed from the persistent state.

D. True / False The system manager is informed in order to reallocate resource to balance the load.

2. [6 points]: Mark true or false for each of the following statements about the Unison file synchronizer:

(Circle True or False for each choice.)

A. True / False It maintains a log of file modifications that it scans to determine the changes to be synchronized.

B. True / False Unison automatically resolves all conflicts.

C. True / False It uses the Paxos algorithm to keep the different file replicas synchronized.

3. [3 points]: Recall Abelson's lecture. According to the court rulings, the main difference between the Compuserve and Prodigy cases was:

(Circle the BEST answer)

A. Prodigy members connected via the web, while Compuserve was a bulletin board.

B. Prodigy had moderators who filtered the material in its chat rooms, while Compuserve did not.

C. The Supreme Court had ruled on the Communications Decency Act between the time of the two cases.

D. The Prodigy case was about online pornography, while the Compuserve case was about online gambling.

Initials:

4. [2 points]: Based on Butler's Lampson's paper on "Hints for Computer System Design", is the following statement true or false?

(Circle True or False for each choice.)

A. True / False Lampson argues that helping applications handle virtual memory faults leads to a simpler interface.

5. [12 points]:

Recall the buffer overflows paper from recitation, and consider the following two methods for countering buffer overrun:

- Non-executable stack: This method disallows execution from the memory region of the stack.
- Stack Canary: This method inserts a random value, which is hard for an attacker to guess or obtain, in the stack just before each function return pointer. Thus, if a buffer overflow occurs in the function, it overwrites the canary value before it overwrites the function return pointer. The canary value is checked before returning from the function to make sure it has not changed. If it did, the return pointer is not used.

Which of the following are true?

(Circle True or False for each choice.)

A. True / False Stack canaries will protect against arc injection attacks that chain multiple function calls, as described in the paper.

B. True / False A non-executable stack will protect against function-pointer clobbering.

C. True / False Stack canaries will protect against function pointer clobbering.

D. True / False Stack canaries will protect against data pointer modification.

6. [5 points]: Based on Ross Anderson's paper, indicate whether cryptographic techniques were used to protect each of the following:

(Circle True or False for each choice.)

A. True / False Storing the customer's account number on the magnetic stripe of their bank card.

B. True / False Storing bank-assigned PINs.

C. True / False Transmitting the payment authorization message from the bank's central system to the ATM.

D. True / False Storing a bank's PIN-generating key.

E. True / False Storing the identity of the ATM machine.

Initials:

II Two-phase commit

oh boy - this never ends well

The object store paper and DP2 use two-phase commit (2PC). Ben wonders if he can optimize his 2PC design for DP2 to improve commit time. He uses a simple, correct protocol that supports collaborating on only one file, and as in DP2 he assumes a fixed number collaborators with known public IP addresses. His design is as follows:

- When a collaborator wants to commit to a version for submission, the collaborator initiates 2PC by becoming the coordinator and writing a BEGIN record to its 2PC log, recording a unique version name for the file, the coordinator's IP address, and a unique ID.
- When the coordinator and one of the collaborators ("cohort") are both online, they exchange 2PC logs. The cohort copies the coordinator's records into its local log. If the cohort sees a new 2PC BEGIN record, the cohort adds a COMMIT record (containing its public IP address and the unique ID of the BEGIN record), if the cohort has the current version and if the cohort has no modifications to that version. The cohort adds an ABORT record if the cohort has a newer version of the file. As the final step in syncing the cohort sends a copy of those records to the coordinator, which logs them in the coordinator's log.
- If the coordinator syncs with a coordinator for another concurrent instance, then the 2PC instance with the highest ID wins. The coordinator for the losing instance writes a END-ABORT record, and becomes a cohort for the winning instance. So if 2 commits at same time
- When the coordinator has obtained commit or abort log entries from all cohorts, it writes a END-COMMIT record in the log if all cohorts agree to the commit. Otherwise, it writes an END-ABORT record. Both actions include the unique ID of the BEGIN record and end that instance of 2PC. Cohorts learn about the outcome on the next sync with the coordinator, and copy the final record into their local logs.

The log is stored on disk, and each write to the log results in flushing the log to disk.

more like 1 phase

uh no - same but described differently

They said correct earlier

(See next page.)

Initials:

7. [16 points]: Which of the following optimizations are correct? (i.e., do not break the DP2 correctness)?

(Circle True or False for each choice.)

16

A. True / False To avoid having the coordinator to sync with each cohort explicitly, Ben modifies the protocol to allow cohorts to sync and exchange 2PC log records. When the coordinator syncs with a cohort, the cohort gives the coordinator copies of all its records, including ones that it learned from other cohorts.

Ohh, cohorts coordinate - That could work - just proxies/middlemen who forward everything on

B. True / False To avoid relying on a disk, Ben modifies the protocol to keep the log in main memory. The contents of memory are lost after a power failure.

no good recovery

C. True / False Ben modifies the writing to the log by batching writes in main memory, replying over the network immediately, and writing all batched entries after a 30 seconds delay, to increase disk throughput.

Write before replying

D. True / False When a cohort doesn't hear from the coordinator within 1 day, and the 2PC instance hasn't committed or aborted, it aborts the 2PC instance by writing an END-ABORT record on behalf of the coordinator into its log, and starts a new 2PC instance.

~~Unless coord d~~

Could have missed END-Commit (lost in the mail)

Coord must do the same thing

Initials:

III Version vectors

Ben implements his DP2 design with version vectors. He tests it out on a deployment where he shares a file *F* among 3 users (including Ben). Sometime during testing, Ben's laptop has the following version vector for its local copy of *F*: [1, 2, 3]. The first entry in the version vector is for Ben's laptop and represents the fact that Ben's laptop has made one change to its local copy of *F*. The other two entries represent that Ben's local copy of *F* reflects 2 changes from user 2 and 3 changes from user 3.

8. [18 points]: When Ben synchronizes with some other user, the software compares the received version vector with Ben's local version vector, and updates the file *F* and the local version vector accordingly. Which of the following statements are correct, if Ben wants his system to work properly?

(Circle True or False for each choice.)

A. True / False If the received version vector is [1, 2, 4], the software should set the local copy of *F* to the received copy of *F* and set the local version vector to the received one.

B. True / False If the received version vector is [1, 2, 2], the software should ignore the received version vector and file *F*.

C. True / False If the received version vector is [2, 2, 3], the software should terminate and indicate that the software has a bug.

D. True / False If the received version vector is [1, 3, 4], the software should set the local copy of *F* to the received copy of *F* and set the local version vector to the received one.

E. True / False If the received version vector is [1, 3, 2], and the local *F* and remote *F* are not identical, the software should signal a conflict and ask Ben to merge the local *F* and remote *F*.

F. True / False If Ben resolves a conflict, the Ben's local version vector should be set to the one received from the other user.

IV Passwords

Alyssa P. Hacker is designing a system to store passwords for 1 million users (10^6). She is worried about what happens when an adversary might get a copy of her entire password database, and decides to store hashes of user passwords, instead of the actual passwords. Alyssa's hash function is relatively expensive to compute: assume it takes about 1 second on one CPU.

good

How long would it take an adversary to guess every user's password (in CPU-seconds), if the adversary were to obtain Alyssa's entire password database, assuming that each user's password consists of 5 random digits (0 through 9), for the following scenarios?

weak

4

9. [4 points]: Alyssa's database stores the hash of each user's password.

Build a rainbow table 10^5 seconds

(or download one from the internet – $O(1)$)

4

Unique salt per user

10. [4 points]: Alyssa's database stores the hash of each user's password with salting, where each salt consists of 4 random digits, and the salt is stored along with the password hash. The attacker obtains both the password hashes and the corresponding salts.

No single rainbow table possible

~~Must try each password each time $10^5 \cdot 1$ million $= 10^{11}$ sec~~

└ has the given salt

Though only 10^4 possible salts – so actually $10^5 \cdot 10^4$ to build rainbow table for each (w/ lots of memory) 10^9

4

11. [4 points]: Alyssa's database stores the hash of each user's password with salting, where each salt consists of 10 random digits, and the salt is stored along with the password hash. The attacker obtains both the password hashes and the corresponding salts.

Here must try each $10^5 \cdot 10^6 = 10^{11}$

Initials:

V Worms

Consider the following pseudocode for the Bitty worm, which is based on the Witty worm but has a different `rand()` function and constructs `dest_ip` in a different way:

```
rand() {
  # Note that 32-bit integers obviate the need for
  # a modulus operation here.
  X = X * 2 + 11;
  return X;
}

srand(seed) {
  X = seed;
}

main() {
  srand(get tick count());
  for (i=0; i < 20000; ++i) {
    dest_ip[0..7]  <- rand()[24..31];
    dest_ip[8..31] <- rand()[0..23];
    dest_port      <- rand()[8..23];
    sendto();
  }
}
```

Assume that the least-significant bit of a 32-bit integer is bit 31, as used by the square-brackets operator. When an IP address written as A.B.C.D, A is the most significant 8 bits.

12. [11 points]: Ben B and his friends have received authorization from MIT to use the MIT /8 IP domain (18.0.0.0) as an Internet Telescope for a few days. Will they receive packets sent by the Bitty worm shown above?

(Circle the BEST answer)

A. Yes, 2 times per infected server.

B. Yes, 11 times per infected server.

C. We cannot know how many times without additional information.

D. No, it will never detect it.

Initials:

VI Trusting Trust

Just look at the byte code...

13. [11 points]:

Ben Bitdiddle is worried that his Unix system's C compiler remains trojaned by Ken Thompson, as described in the "Reflections on Trusting Trust" paper. His friend Alyssa P. Hacker has a C compiler binary (and corresponding source code) that can run on Ben's Unix system, and her compiler binary and source code are both known to contain no trojans. Help Ben detect whether his compiler is trojaned or not, by writing down two compilation chains whose results (i.e., generated binaries) Ben can compare to detect Ken Thompson's compiler trojan. You can assume all compilers are deterministic (i.e., compiling the same source code using the same compiler always generates the same output), and that the source code for Ben's compiler does not contain a trojan. but not the binary

In your answer, use $X \rightarrow Y$ to denote the result of using X to compile Y. You can chain this operator: for example, $X \rightarrow Y \rightarrow Z$ denotes the result of first using X to compile Y, and then using the resulting executable to compile Z.

Use B and S to denote the binary and source for Ben's compiler (where B may contain a trojan, but S does not), and use A to denote the binary for Alyssa's compiler (which is known to be trojan-free).

To get you familiar with this notation, Ken Thompson's compiler inserts a trojan when compiling the source code of the Unix login program, L (i.e., $B \rightarrow L$ is a trojaned login executable). The compiler also inserts a trojan when compiling its own source code S, so that $B \rightarrow S$ produces a trojaned compiler, and $B \rightarrow S \rightarrow L$ produces a trojaned binary using the re-compiled compiler. ; safe is that or something else;

Contain trojan: may, does not — not, not

Check: | B → S | $\stackrel{?}{=}$ | A → S |

(does the hacked binary equal the clean binary

Hes woried about the integrity of his compiler

End of Quiz

very confused

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

Initials:

End of Quiz

70 4th day
65 3rd
60 min 2nd - just last qu
30 min 1st

Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

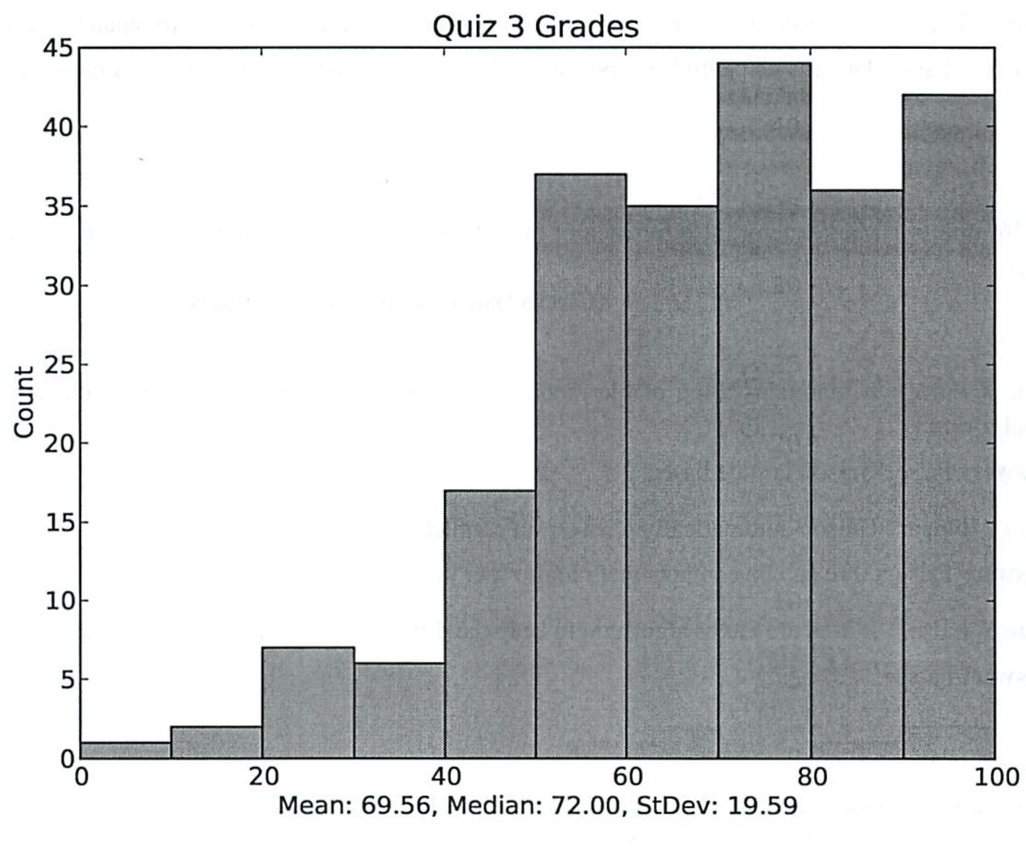
6.033 Computer Systems Engineering: Spring 2012

Quiz 3 Solutions

There are 13 questions and 10 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

1. [4 points]:

In the Porcupine mail service, as described in the paper, whenever there is a membership reconfiguration occurs, the following happens:

(Circle True or False for each choice.)

A. True / False The "user map" is updated remapping users to clusters.

Answer: True. (We meant nodes instead of clusters.)

B. True / False The fragments in the "mailbox fragment list" are combined into a single mailbox for each user.

Answer: False.

C. True / False Soft state is recomputed from the persistent state.

Answer: True.

D. True / False The system manager is informed in order to reallocate resource to balance the load.

Answer: False. There is no central component in Porcupine that needs to be informed for load balancing.

2. [6 points]: Mark true or false for each of the following statements about the Unison file synchronizer:

(Circle True or False for each choice.)

A. True / False It maintains a log of file modifications that it scans to determine the changes to be synchronized.

Answer: False. Unison is state based.

B. True / False Unison automatically resolves all conflicts.

Answer: False; there can be conflicts that require user input.

C. True / False It uses the Paxos algorithm to keep the different file replicas synchronized.

Answer: False

3. [3 points]: Recall Abelson's lecture. According to the court rulings, the main difference between the Compuserve and Prodigy cases was:

(Circle the BEST answer)

A. Prodigy members connected via the web, while Compuserve was a bulletin board.

B. Prodigy had moderators who filtered the material in its chat rooms, while Compuserve did not.

C. The Supreme Court had ruled on the Communications Decency Act between the time of the two cases.

D. The Prodigy case was about online pornography, while the Compuserve case was about online gambling.

Answer: B.

4. [2 points]: Based on Butler's Lampson's paper on "Hints for Computer System Design", is the following statement true or false?

(Circle True or False for each choice.)

A. True / False Lampson argues that helping applications handle virtual memory faults leads to a simpler interface.

Answer: False. Lampson argues that such features lead to complexity, which may lead to security problems, such as in the Tenex system.

5. [12 points]:

Recall the buffer overflows paper from recitation, and consider the following two methods for countering buffer overrun:

- Non-executable stack: This method disallows execution from the memory region of the stack.
- Stack Canary: This method inserts a random value, which is hard for an attacker to guess or obtain, in the stack just before each function return pointer. Thus, if a buffer overflow occurs in the function, it overwrites the canary value before it overwrites the function return pointer. The canary value is checked before returning from the function to make sure it has not changed. If it did, the return pointer is not used.

Which of the following are true?

(Circle True or False for each choice.)

A. True / False Stack canaries will protect against arc injection attacks that chain multiple function calls, as described in the paper.

Answer: True. Multiple function calls require return instructions which will catch possible stack buffer overflows.

B. True / False A non-executable stack will protect against function-pointer clobbering.

Answer: False. An adversary can still clobber a function pointer that's not on the stack (e.g., a global struct).

C. True / False Stack canaries will protect against function pointer clobbering.

Answer: False. Same as above.

D. True / False Stack canaries will protect against data pointer modification.

Answer: False. Same as above.

6. [5 points]: Based on Ross Anderson's paper, indicate whether cryptographic techniques were used to protect each of the following:

(Circle True or False for each choice.)

A. True / False Storing the customer's account number on the magnetic stripe of their bank card.

Answer: False.

B. True / False Storing bank-assigned PINs.

Answer: True.

C. True / False Transmitting the payment authorization message from the bank's central system to the ATM.

Answer: False.

D. True / False Storing a bank's PIN-generating key.

Answer: True.

E. True / False Storing the identity of the ATM machine.

Answer: False.

Initials:

II Two-phase commit

The object store paper and DP2 use two-phase commit (2PC). Ben wonders if he can optimize his 2PC design for DP2 to improve commit time. He uses a simple, correct protocol that supports collaborating on only one file, and as in DP2 he assumes a fixed number collaborators with known public IP addresses. His design is as follows:

- When a collaborator wants to commit to a version for submission, the collaborator initiates 2PC by becoming the coordinator and writing a BEGIN record to its 2PC log, recording a unique version name for the file, the coordinator's IP address, and a unique ID.
- When the coordinator and one of the collaborators ("cohort") are both online, they exchange 2PC logs. The cohort copies the coordinator's records into its local log. If the cohort sees a new 2PC BEGIN record, the cohort adds a COMMIT record (containing its public IP address and the unique ID of the BEGIN record), if the cohort has the current version and if the cohort has no modifications to that version. The cohort adds an ABORT record if the cohort has a newer version of the file. As the final step in syncing the cohort sends a copy of those records to the coordinator, which logs them in the coordinator's log.
- If the coordinator syncs with a coordinator for another concurrent instance, then the 2PC instance with the highest ID wins. The coordinator for the losing instance writes a END-ABORT record, and becomes a cohort for the winning instance.
- When the coordinator has obtained commit or abort log entries from all cohorts, it writes a END-COMMIT record in the log if all cohorts agree to the commit. Otherwise, it writes an END-ABORT record. Both actions include the unique ID of the BEGIN record and end that instance of 2PC. Cohorts learn about the outcome on the next sync with the coordinator, and copy the final record into their local logs.

The log is stored on disk, and each write to the log results in flushing the log to disk.

(See next page.)

Initials:

7. [16 points]: Which of the following optimizations are correct? (i.e., do not break the DP2 correctness)?

(Circle True or False for each choice.)

A. True / False To avoid having the coordinator to sync with each cohort explicitly, Ben modifies the protocol to allow cohorts to sync and exchange 2PC log records. When the coordinator syncs with a cohort, the cohort gives the coordinator copies of all its records, including ones that it learned from other cohorts.

Answer: True. It doesn't matter how the coordinator learns about all of the cohorts committing, as long as it does so correctly.

B. True / False To avoid relying on a disk, Ben modifies the protocol to keep the log in main memory. The contents of memory are lost after a power failure.

Answer: False. The commit records can be lost, and the coordinator might commit even though one of the cohorts doesn't realize it.

C. True / False Ben modifies the writing to the log by batching writes in main memory, replying over the network immediately, and writing all batched entries after a 30 seconds delay, to increase disk throughput.

Answer: False. Commit records can be lost again.

D. True / False When a cohort doesn't hear from the coordinator within 1 day, and the 2PC instance hasn't committed or aborted, it aborts the 2PC instance by writing an END-ABORT record on behalf of the coordinator into its log, and starts a new 2PC instance.

Answer: False. The failure may have been in the network between the coordinator (which committed) and the cohort. As a result, the cohort may abort even though the coordinator committed.

III Version vectors

Ben implements his DP2 design with version vectors. He tests it out on a deployment where he shares a file F among 3 users (including Ben). Sometime during testing, Ben's laptop has the following version vector for its local copy of F: [1, 2, 3]. The first entry in the version vector is for Ben's laptop and represents the fact that Ben's laptop has made one change to its local copy of F. The other two entries represent that Ben's local copy of F reflects 2 changes from user 2 and 3 changes from user 3.

8. [18 points]: When Ben synchronizes with some other user, the software compares the received version vector with Ben's local version vector, and updates the file F and the local version vector accordingly. Which of the following statements are correct, if Ben wants his system to work properly?

(Circle True or False for each choice.)

A. True / False If the received version vector is [1, 2, 4], the software should set the local copy of F to the received copy of F and set the local version vector to the received one.

Answer: True. The remote machine has one more change (from user 3), and Ben should accept it.

B. True / False If the received version vector is [1, 2, 2], the software should ignore the received version vector and file F.

Answer: True. The remote machine has one fewer changes (from user 3), and Ben need not do anything to his own state.

C. True / False If the received version vector is [2, 2, 3], the software should terminate and indicate that the software has a bug.

Answer: True. The remote machine appears to know about one more change from Ben's machine than Ben's machine itself does.

D. True / False If the received version vector is [1, 3, 4], the software should set the local copy of F to the received copy of F and set the local version vector to the received one.

Answer: True. The remote machine knows about one more change from user 2, and one more change from user 3, than Ben's machine does.

E. True / False If the received version vector is [1, 3, 2], and the local F and remote F are not identical, the software should signal a conflict and ask Ben to merge the local F and remote F.

Answer: We accepted either True or False. The remote machine has seen one more change from user 2, but one fewer change from user 3, so the changes may be in conflict. However, DP2 required automatic merging whenever possible, so it might be possible to avoid signaling a conflict.

F. True / False If Ben resolves a conflict, the Ben's local version vector should be set to the one received from the other user.

Answer: False. This would discard any changes that Ben's version vector reflects which aren't in the remote version vector.

IV Passwords

Alyssa P. Hacker is designing a system to store passwords for 1 million users (10^6). She is worried about what happens when an adversary might get a copy of her entire password database, and decides to store hashes of user passwords, instead of the actual passwords. Alyssa's hash function is relatively expensive to compute: assume it takes about 1 second on one CPU.

How long would it take an adversary to guess every user's password (in CPU-seconds), if the adversary were to obtain Alyssa's entire password database, assuming that each user's password consists of 5 random digits (0 through 9), for the following scenarios?

9. [4 points]: Alyssa's database stores the hash of each user's password.

Answer: 10^5 seconds. There are 10^5 possible passwords, so an adversary can compute such a table, and then efficiently look up everyone's password. (We assumed that comparisons were free.)

10. [4 points]: Alyssa's database stores the hash of each user's password with salting, where each salt consists of 4 random digits, and the salt is stored along with the password hash. The attacker obtains both the password hashes and the corresponding salts.

Answer: 10^9 seconds. There are 10^5 possible passwords, and 10^4 possible salt values, leading to 10^9 possible hash inputs. An adversary can build up a table as above.

11. [4 points]: Alyssa's database stores the hash of each user's password with salting, where each salt consists of 10 random digits, and the salt is stored along with the password hash. The attacker obtains both the password hashes and the corresponding salts.

Answer: 10^{11} seconds. There are still 10^5 possible passwords, but only 10^6 possible salts (the ones actually present in the password database). An adversary can compute a table in 10^{11} seconds as above.

Initials:

V Worms

Consider the following pseudocode for the Bitty worm, which is based on the Witty worm but has a different rand() function and constructs dest_ip in a different way:

```
rand() {
  # Note that 32-bit integers obviate the need for
  # a modulus operation here.
  X = X * 2 + 11;
  return X;
}

srand(seed) {
  X = seed;
}

main() {
  srand(get tick count());
  for (i=0; i < 20000; ++i) {
    dest_ip[0..7]  <- rand()[24..31];
    dest_ip[8..31] <- rand()[0..23];
    dest_port      <- rand()[8..23];
    sendto();
  }
}
```

Assume that the least-significant bit of a 32-bit integer is bit 31, as used by the square-brackets operator. When an IP address written as A.B.C.D, A is the most significant 8 bits.

12. [11 points]: Ben B and his friends have received authorization from MIT to use the MIT /8 IP domain (18.0.0.0) as an Internet Telescope for a few days. Will they receive packets sent by the Bitty worm shown above?

(Circle the BEST answer)

A. Yes, 2 times per infected server.

B. Yes, 11 times per infected server.

C. We cannot know how many times without additional information.

D. No, it will never detect it.

Answer: D, because the low bit of rand()'s return value is always 1, and the low bit of MIT's "18" is 0.

Initials:

VI Trusting Trust

13. [11 points]:

Ben Bitdiddle is worried that his Unix system's C compiler remains trojaned by Ken Thompson, as described in the "Reflections on Trusting Trust" paper. His friend Alyssa P. Hacker has a C compiler binary (and corresponding source code) that can run on Ben's Unix system, and her compiler binary and source code are both known to contain no trojans. Help Ben detect whether his compiler is trojaned or not, by writing down two compilation chains whose results (i.e., generated binaries) Ben can compare to detect Ken Thompson's compiler trojan. You can assume all compilers are deterministic (i.e., compiling the same source code using the same compiler always generates the same output), and that the source code for Ben's compiler does not contain a trojan.

In your answer, use $X \to Y$ to denote the result of using X to compile Y. You can chain this operator: for example, $X \to Y \to Z$ denotes the result of first using X to compile Y, and then using the resulting executable to compile Z.

Use B and S to denote the binary and source for Ben's compiler (where B may contain a trojan, but S does not), and use A to denote the binary for Alyssa's compiler (which is known to be trojan-free).

To get you familiar with this notation, Ken Thompson's compiler inserts a trojan when compiling the source code of the Unix login program, L (i.e., $B \to L$ is a trojaned login executable). The compiler also inserts a trojan when compiling its own source code S, so that $B \to S$ produces a trojaned compiler, and $B \to S \to L$ produces a trojaned binary using the re-compiled compiler.

Check: [ ] $\stackrel{?}{=}$ [ ]

Answers:

- $B \to S \to S = A \to S \to S$
- $B \to S = A \to S \to S$
- $B \to S \to L = A \to S \to L$
- $B \to L = A \to S \to L$

Note that $B \to S$ and $A \to S$ can be different, because Ben's and Alyssa's compilers can generate different code for the same input S. However, once you compile S, you get a compiler binary that will produce the same code if used to build S again. Thus, using the two compilers $B \to S$ and $A \to S$, we can compile S again, and check if the resulting binary is the same. The binary for $A \to S \to S$ contains no trojans, but if B is trojaned, $B \to S \to S$ would contain a trojan.

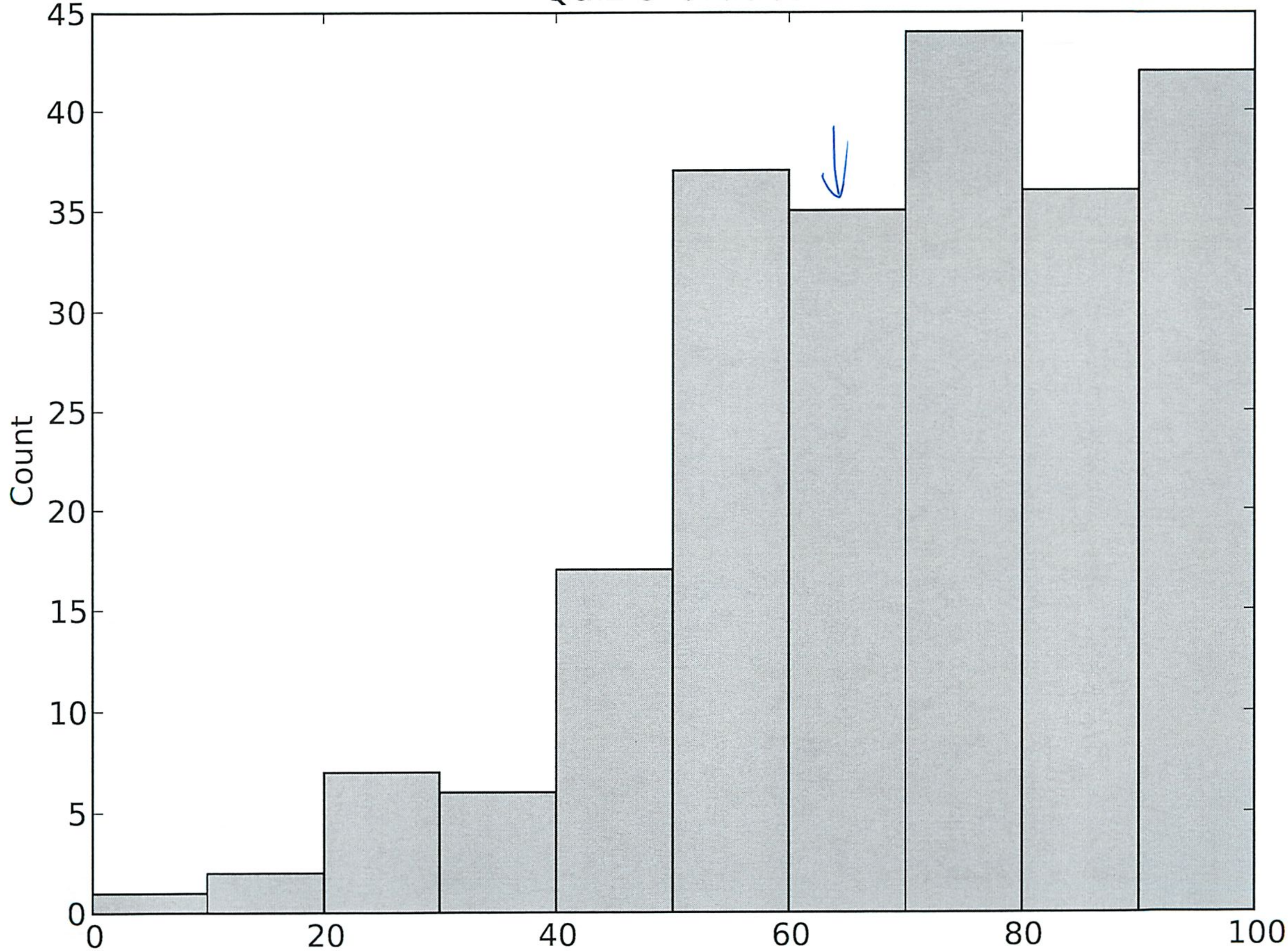
End of Quiz

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

Initials:

6.033

Quiz 3 Grades



Mean: 69.56, Median: 72.00, StDev: 19.59

6.033 Assignments for Plasmeier, Michael E

Therac Memo

- Submit Browse...
- Mon Feb 13 23:10:57 2012: submission: submit-0.pdf
- Thu Feb 16 18:05:20 2012: response from Grusecki, Travis: response-travisrg-0.pdf
- Grade: A

Therac Memo Revised

- Submit Browse...
- Sun Feb 19 14:15:46 2012: submission: submit-0.pdf
- Mon Feb 27 19:16:25 2012: response from Custer, Dave: response-custer-1.pdf
- Mon Feb 27 19:16:25 2012: response from Custer, Dave: response-custer-0.pdf

Hands-on 1 - File System

- Submit Browse...
- Sun Feb 19 22:38:25 2012: submission: submit-0.pdf
- Grade: Check +

Hands-on 2 - Unix

- Submit Browse...
- Sun Feb 26 21:06:48 2012: submission: submit-0.pdf
- Grade: Check +

Design Project 1 Proposal

- Submit Browse...
- Tue Feb 28 02:40:55 2012: submission: submit-0.pdf
- Fri Mar 16 11:00:22 2012: response from Grusecki, Travis: response-travisrg-0.pdf
- Tue Mar 13 21:15:57 2012: response from Custer, Dave: response-custer-1.pdf
- Tue Mar 13 21:14:30 2012: response from Custer, Dave: response-custer-0.pdf
- Grade (writing): A

Quiz 1

- Submit Browse...
- Grade: 73

Hands-on 3 - Traceroute

- Submit Browse...
- Mon Mar 12 01:22:30 2012: submission: submit-0.pdf
- Grade: Check +

Design Project 1

- Submit Browse...
- Thu Mar 22 15:47:50 2012: submission: submit-0.pdf
- Thu Apr 5 15:31:37 2012: response from Custer, Dave: response-custer-0.pdf
- Grade (technical): A
- Grade (writing): A

Hands-on 4 - DNS

- Submit Browse...
- Thu Mar 29 00:43:37 2012: submission: submit-0.pdf
- Grade: Check+

Hands-on 5 - Logging

- Submit Browse...
- Sat Apr 7 19:31:07 2012: submission: submit-0.pdf
- Grade: Check+

Hands-on 6 - Databases

- Submit Browse...
- Sun Apr 8 18:48:51 2012: submission: submit-0.pdf
- Sun Mar 18 14:29:41 2012: response from Zeldovich, Nickolai: response-kolya-0.txt
- Grade: Check+

Quiz 2

- Submit Browse...
- Grade: 52

Design Project 2 Proposal

- Submit Browse...
- Thu Apr 26 14:12:11 2012: submission: submit-1.pdf

- Thu Apr 26 02:28:12 2012: submission: submit-0.pdf

Hands-on 7 - Crypto

- Submit Browse...
- Mon May 7 22:54:40 2012: submission: submit-0.pdf
- Grade: Check+

Design Project 2

- Submit Browse...
- Thu May 10 14:56:32 2012: submission: submit-1.pdf
- Thu May 10 10:13:15 2012: submission: submit-0.pdf
- Grade: A

Preliminary Recitation Participation Grade

- Submit Browse...
- Grade: A

Design Project 1 Writing Revision

- Submit Browse...

Writing Section Participation Grade

- Submit Browse...
- Grade: A

Final Recitation Participation Grade

- Submit Browse...
- Grade: A

Quiz 3

- Submit Browse...
- Grade: 60