

Homework #5

Due Monday, April 7 at 11:45PM

Lecture Contents

User Authentication
Protocol Pitfalls
Kerberos

Instructions for Preparation and Submission

- Your answer file format may be .txt (text, any platform), or PDF. Do **not** submit WORD files.
- Submit electronically using submit.ncsu.edu, with filenames hw5.txt or hw5.pdf.
- **Include your name and Student ID #** on the first page
- Do individual work, and submit individually. I do not object if you wish to ask fellow students for help / advice / tips / information, but don't copy someone else's solution.

PROBLEMS (Full credit is 100 Points)

USER AUTHENTICATION AND PASSWORDS

1. (5) Would it be possible to thwart all possible password cracking attacks by increasing the salt length to something large, like 64 bits? If not, which attacks are still likely to be effective?

2. (5) For the following types of possible attacks:

[i] Sniffing-and-replay: the attacker Eve eavesdrops to the communications between Alice and Bob. Later, Eve re-uses Alice's messages to impersonate Alice against Bob.

[ii] Trojan-horse: the attacker Eve impersonates Bob by interacting with Alice, who authenticates herself to him. Later, Eve re-uses the values Alice sent her, to impersonate Alice against Bob.

- a. Which attacks are possible with the normal Unix (crypt) login?
- b. Which attacks are possible with the S/key identification-authentication method?
- c. Which attacks are possible with the the time-based token scheme presented in lecture?

3. (5) For each of the possible and desirable biometric technology properties, which of the following is the best quality choice? Justify your answer in each case.

- (A) Retina scan
- (B) Finger print
- (C) Voice recognition
- (D) All of the above together.

4. (5) In section 12.2 Lamport's hash we mentioned the notion of using only 64 bits of the has. At each stage, 128 bits are computed, 64 bits are thrown away, and the hash of the retained 64 bits is used in the next stage. The purpose of only using 64 bits is so that a human to not need to type as long a string. Assuming the person will still only type 64 bits, does it work if hash^n does a hash of all 128 bits of $\text{hash}^{(n-1)}$, but what the person actually transmits is 64 bits of the result?

5. (5) Suppose we are using Lamport's hash, and Bob crashes before receiving Alice's reply. Suppose an intruder, Trudy, can eavesdrop and detect that Bob crashed. Then Trudy has a quantity (whatever Alice replied that Bob did not receive) which Trudy can use to impersonate Alice, if Trudy logs in before Alice attempts to log into Bob again. How can we modify Bob's behavior to prevent this threat? (Exactly when do we overwrite Bob's database, and with what?)

6. (5) In section 10.5 Eavesdropping a scheme was described in which a user has a numbered list of passwords and the system asks for some small subset on each login. Is there any advantage in asking for more than one password per login? Note that the more passwords requested, the more information an eavesdropper will get. On the other hand, when the eavesdropper attempts to impersonate the user, the more passwords requested, the less likely the eavesdropper will know all of them.

PROTOCOL PITFALLS

(The correspondence between Protocol labels in lecture, and in the textbook, is as follows:

Textbook	Lecture
11-1	B
11-2	variation on B
11-4	another variation on B
11-5	C
11-7	D
11-8	Optimized version of D
11-13	variation on D
11-14	F
11-16	I basic idea
11-17	I in practice
11-18	Needham Schroeder
11-19	Expanded Needham Schroeder
11-20	Otway-Rees)

1. (5) In section 9.6 Eavesdropping and Server Database Reading it was asserted that it is extremely difficult, without public key cryptography, to have an authentication scheme which protects against both eavesdropping and server database disclosure. Consider the following authentication protocol. Alice knows a password. Bob, a server that will authenticate Alice, stores a hash of Alice's password. Alice types her password to her workstation. The following exchange occurs:

* Alice types her name and password, sends to workstation

- * Workstation computes hash of password, sends Alice's name to server Bob
 - * Server Bob responds with nonce R, sends to workstation
 - * Workstation computes hash of (hash of password, R), sends to Bob
 - * Bob computes hash of (hash of expected password, R), compares with received value
- Is this an example of an authentication scheme that isn't based on public key cryptography and yet guards against both eavesdropping and server database disclosure? Why or why not?

2. (5) In section 11.2 Mutual Authentication, our textbook authors discuss the reflection attack and note that Protocol 11-8 is susceptible, but Protocol 11-7 is not. How about Protocol 11-11? Why or why not?

3.

a. (5) Suppose we are using a three-message mutual authentication protocol, and Alice initiates contact with Bob. Suppose we wish Bob to be a stateless server, and therefore it is inconvenient to require him to remember the challenge he sent to Alice. Let's modify the exchange so that Alice sends the challenge back to Bob, along with the encrypted challenge. So the protocol is:

Alice to Bob: "I'm Alice"

Bob to Alice: R

Alice to Bob: R, $K_{\text{Alice-Bob}}\{R\}$

Is this protocol secure? Why or why not?

b. (5) Let's modify the protocol from the previous problem so that Bob sends both the challenge and a challenge encrypted with a key that only he knows, to Alice:

Alice to Bob: "I'm Alice"

Bob to Alice: R, $K_{\text{Bob}}(R)$

Alice to Bob: $K_{\text{Bob}}(R)$, $K_{\text{Alice-Bob}}(R)$

Is this protocol secure? Why or why not? Is there a simple fix if it is not secure?

4. (5) Design a two-message authentication protocol, assuming that Alice and Bob know each other's public keys, which accomplishes both mutual authentication and establishment of a session key, and that does not require clock synchronization.

5. (5) A "rule of thumb" for security is that someone masquerading as a server Bob should not be able to trick a client Alice into signing or decrypting an arbitrary value. Which of protocols 11-1 through 11-6 provide this type of security?

6. (5) For the following protocol:

A to B: R_a

B to A: $B, A, R_a, R_b, h_K(B, A, R_a, R_b)$

A to B: $A, R_b, h_{K'}(A, R_b)$

Assume A and B share two long-term symmetric keys K and K' and use a keyed hash function h_K and a keyed one-way function $h_{K'}$. The shared key is $k = h_{K'}(R_b)$. Show this provides mutual authentication and a shared key negotiation. Analyze it to see if it has any weaknesses, according to the "checklist" of our chapter 11.

7. (5) Modify the Needham-Schroeder key exchange protocol so that both parties A and B can contribute input to the generation of the session key.

KERBEROS

1. (5) Design a variant of Kerberos in which the workstation generates a TGT. The TGT will be encrypted with the user's master key rather than the KDC's master key. How does this compare with standard Kerberos in terms of efficiency, security, etc.? What happens in each scheme if the user changes her password during a login session?
2. (5) Why is the authenticator field not of security benefit when asking the KDC for a ticket for V, but useful when logging into V?
3. (5) Consider the following variant of Kerberos. Instead of having postdated or renewable ticks, a server which notes that the start-time is older than some limit present the ticket to the TGS and asks if it should believe the ticket. What are the trade-offs of this approach relative to the Kerberos V5 approach?
4. (5) Design a different method of the server V authenticating Alice when V doesn't remember its own master key, which places the work on V instead of Alice. Alice will act as if V does remember its own master key, and V interacts appropriately with the KDC so that Alice will be unaware that V didn't know its own master key.
5. (5) In the mutual authentication in the Needham-Schroeder protocol upon which Kerberos is based, the authenticator contained only an encrypted timestamp. The protocol is that Alice sends Bob the authenticator, and then Bob must decrypt the authenticator add one to the value inside, re-encrypt it, and send it back to Alice.
 - a. Why was it necessary for Bob to increment the value before re-encrypting it and sending it to Alice?
 - b. Why isn't it necessary in Kerberos V5, in the AP_REP message?
 - c. In Kerberos V4, it is the checksum field (which isn't really a checksum) that is extracted and incremented. Would it have been just as secure in V4 for Bob to send back the contents of the checksum field encrypted and not incremented?
6. (5) What feature(s) if any of Kerberos V4 prevent(s) replay attacks from succeeding? What features if any prevent(s) reflection attacks from succeeding? What features if any prevent(s) man-in-the-middle attacks from succeeding?

EXTRA CREDIT

1. (6) Suppose Trudy hijacks a conversation between Alice and Bob. This means that after the initial handshake, Trudy sends message with source address equal to Alice's source address. Suppose the network allows Trudy to insert a fake source address, but does not deliver packets destined for Alice's address to Trudy. What are the problems involved in having Trudy transmit a file to Bob as if she were Alice? Consider potential

problems with flow control and file transfer protocols when Trudy cannot see return traffic from Bob.

2. (6) A protocol T has been developed for broadcast authentication. In its simplified form, T randomly generates a key K_n and computes $K_i = H(K_{i+1})$ for $i = n - 1, n - 2, \dots, 0$, where H is a hash function. In addition, T partitions a period of operation time into n time intervals, denoted as I_1, I_2, \dots, I_n . Each key K_i is associated with the time interval I_i , and used to generate MACs for all the messages the sender broadcasts during I_i . However, the sender doesn't disclose K_i until T time units after I_i . Let's denote the beginning time of I_i is T_i . Then the sender doesn't disclose K_i until $T_{i+1} + T$. Each receiver buffers the messages received during I_i . It can authenticate the messages broadcasted during I_i after it receives K_i disclosed by the sender. Assume all the receivers know the sender's public key PKs.

(a) Develop a way so that each receiver can authenticate each K_i disclosed by the sender.

(b) When a receiver receives a broadcast message authenticated with K_i at time t, how can it determine if the message wasn't forged by an attacker that just learned the K_i disclosed by the sender? In other words, develop a security condition, by checking which a receiver can determine if the message was sent before K_i is disclosed. Assume the maximum clock discrepancy between the sender and the receiver is 1 minute, and the time required for message transmission is negligible.