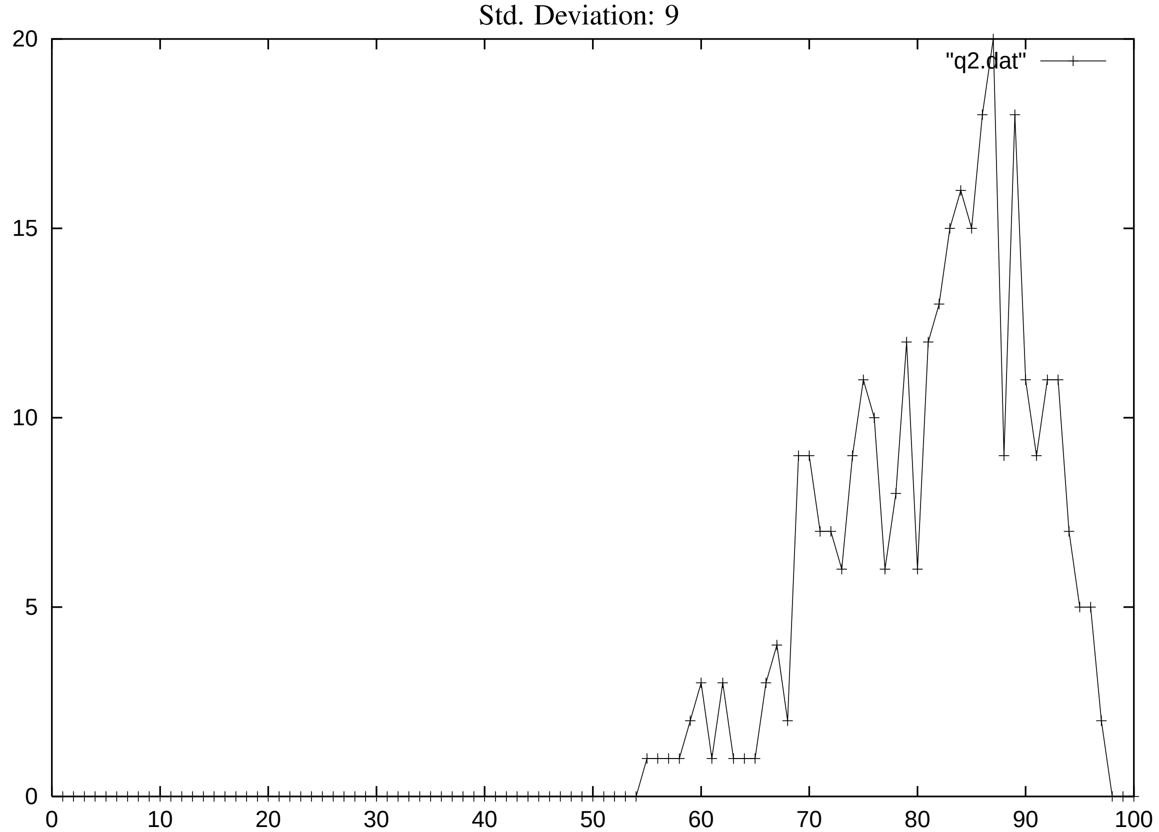
Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.033 Computer Systems Engineering: Spring 2001

Handout 20 - Quiz 2 Solutions

Average: 81
Median: 83
Std. Deviation: 9



I Reading questions

1. [6 points]: What problems do Network Address Translators (NATs) cause?
(Circle ALL that apply)

A. NATs are not transparent to all end-to-end applications.

B. NATs require changes to the link layer.

C. NATs break the Internet's universal addressing model.

D. NATs increase the consumption of IP addresses.

2. [6 points]: The following features of Grapevine (as described in Reading #15) make it scale well
(Circle ALL that apply)

A. Decentralizing administrative functions

B. Partitioning the name space into multiple, independent registries

C. Exchanging complete databases nightly to resolve inconsistencies

D. Allowing databases to be inconsistent temporarily

Exchanging the complete database nightly will take longer and longer as the system grows, until it takes more than a night.

3. [6 points]: IP tunneling as described by Johnson in the mobile IP paper (Reading #12) is a technique
(Circle ALL that apply)

A. To reduce the cost of the Big Dig by replacing physical tunnels with virtual tunnels.

B. To allow stationary hosts to communicate through mobile hosts that don't speak Mobile IP

C. To allow a packet to be routed using IP to a mobile host's current location, if that location is different than its home network location

D. To reduce space in Mobile-IP packets

4. [6 points]: Why are queues in Click (as described in Reading #14) explicit?
(Circle ALL that apply)

A. Simplifies most elements, since they don't have to worry about packet storage

B. Because queues are the only elements that have only push ports

C. Allows the construction of sophisticated scheduling configurations based on queues

D. Reduces the number of elements in a router

5. [6 points]: Ross Anderson in "Why cryptosystems fail" (Reading #18) argues
(Circle ALL that apply)

A. Cryptosystems are unnecessary

B. Achieving security in a system is a difficult engineering problem rather than just a problem of cryptography

C. Design and implementation techniques for safety-critical systems also apply to systems that need to be secure

D. Cryptography needs to be improved to enable secure systems

II WebTrust.com, OutOfMoney.com part II

After their disastrous experience with OutOfMoney.com, the 16-year-old kids regroup. They rethink their business plan and switch from being a service provider to a technology provider. Reading many war stories about security has convinced the kid wizards that there should be a market for a secure client authentication product for Web servers. The kids re-incorporate as WebTrust.com.

The kids study up on how the Web works. They discover that HTTP 1.0 is a simple protocol with essentially two remote procedure calls:

```
GET (URL) returns document
POST (URL, form) returns document
```

The `GET` procedure gets a document from a Web server. The `POST` procedure sends back to the server the entries that the user filled out on a form that was in a previously retrieved document. The `POST` procedure also gets a document. The browser invokes the `POST` procedure when the user hits the submit button on the form.

These remote procedure calls are sent over a reliable transport protocol, TCP. A web browser opens a TCP connection, calls a procedure (`GET` or `POST`), and waits for a result from the server. The Web server waits for a TCP connection request, accepts the connection, and waits for the `GET` or `POST` call. Once the call arrives, it executes it, sends the result over the connection, and closes the connection. The browser displays the response and closes the connection on its side. (Thus, a connection is setup for each request.)

Simple URLs are of the form:

```
http://www.WebTrust.com/index.html
```

6. [6 points]: `www.WebTrust.com` in the above URL is

(Circle best answer)

A. a DNS name

B. a protocol name

C. a pathname for a file

D. an IP address

Name:

The objective of WebTrust.com's product is to authenticate users of on-line services. The intended use is for a user to login once per session and to allow only logged-in users access to the rest of the site. The product consists of a collection of Web pages and some server software. The company employs their own product to authenticate customers of the company's Web site.

To allow Internet users to create an account, WebTrust.com has a Web form in which a user types in her username and two copies of her password. When the user types in her password, the browser doesn't echoed it, but instead displays a "*" for each typed character. When the user hits the submit button, the user browser calls the `POST` procedure to send the form to the server.

When the server receives an create-account request, it makes sure that the two passwords are the same and makes sure that the account doesn't exist. If these conditions are true, then it creates an entry in its local password file. If either of the conditions is false, the the server returns an error.

The form to create an account is stored in the document "create.html" on WebTrust's Web site. Another document on the server contains:

```
<a href="create.html">Create an account</a>
```

7. [6 points]: What is the source of the context reference that identifies the context in which the name "create.html" will be resolved?

(Circle best answer)

A. The browser derives a default context reference from the URL of the document that contains the relative URL.

B. It is configured in the Web browser when the browser is installed.

C. The server derives it from information it remembers about previous documents it sent to this client.

D. The user types it into the browser.

8. [6 points]: Why does the form for creating an account ask a user to type in her password twice?
(Circle ALL that apply)

A. To allow a password not to be echoed on the screen while enabling users to catch typos.

B. To detect transmission errors between the keyboard and the browser.

C. To reduce the probability that a packet with a password has to be retransmitted if the network deletes the packet.

D. To make it harder for users to create fake accounts.

Full credit was awarded to students who did not circle B, but noted that detecting transmission errors was more likely to have been a side-effect than a goal of the design.

9. [6 points]: In this system, to what attacks is creating an account vulnerable? (Assume an active attacker.)
(Circle ALL that apply)

A. An attacker can learn the password for a user by eavesdropping

B. An attacker can modify the password

C. An attacker can overwhelm server by creating many accounts

D. An attacker can run a server that pretends to be "www.WebTrust.com"

*Some students noted that A is a passive attack; but active attack*ers *have more tools than passive attackers, not fewer. Some students noted that all network services are vulnerable to denial of service attacks; but account creation is also vulnerable to attacks that overwhelm the account database. Some students thought that DNS servers would thwart impersonating a server, but an active attacker can also get around or subvert DNS.*

To login, the user visits the web page "login.html", which asks the user for her account name and password. When the user hits the submit button, the browser invokes the `POST` procedure, which sends the user name and password to the server. The server checks the stored password against the password in the login request. If they match, the user is allowed to access the server; otherwise, the server returns an error.

10. [6 points]: To what attacks is the login procedure vulnerable? (Assume an active attacker.)
(Circle ALL that apply)

A. An attacker can login by replaying a recorded `POST` from a legitimate login

B. An attacker can login as any user by replaying a single recorded `POST` for login

C. An attacker can impersonate WebTrust.com to any registered user

D. An attacker can impersonate WebTrust.com to an unregistered user

***A** is correct: Since the POST does not contain anything that assures freshness, the browser has no way of detecting and rejecting a later copy replayed by an attacker. **B** is not correct: The POST contains a user's name and password, so simply replaying it will log in that user; it can't log in any other user. **C** is correct. It's a difficult attack, but it can be accomplished. The thing that makes it hard is that the registered user knows how the real webtrust.com responds, so the attacker must behave in exactly the same way, even to the extent of rejecting wrong passwords. But it can do just that by standing in the middle, passing some or all of the user's requests to the real webtrust.com, and passing its responses back. **D** is correct; all the attacker needs to do is reject the login the same way that webtrust.com does.*

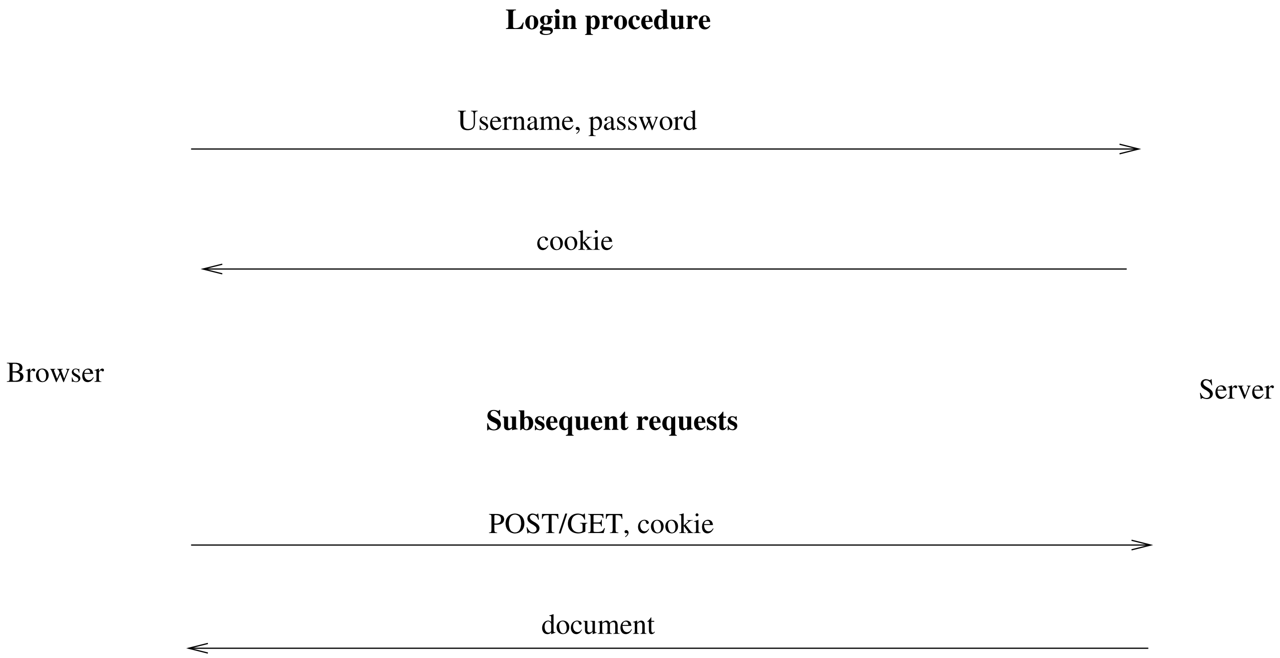


Figure 1: The protocol

To authenticate subsequent Web requests from a user after they have logged in, WebTrust.com exploits something called cookies in Web terminology. A server can install some state (called a "cookie") in the Web browser. The server installs this state by including in the response a "set-cookie" directive containing data to be stored in the cookie.

WebTrust.com's use of cookies is summarized in Figure 1. The document containing the response to a login request includes the directive[1]:

```
set-cookie (cookie)
```

The browser stores the cookie in memory. (In practice, cookies are named, but for this quiz, you can assume that a cookie always refers to WebTrust.com's cookie.)

On subsequent calls (i.e., `GET` or `POST`) to the server that installed the cookie, the browser will send the installed cookie along with the call. Thus, once WebTrust.com has set a cookie in a browser, it will see that cookie on all subsequent requests from that browser.

The server requires that the browser send a cookie along with all calls, except a create or login call. If the cookie is missing (e.g., the browser has failed and lost the cookie stored in memory, or an attacker is leaving the cookie out on purpose), the server will return an error to the browser and ask the user to login (again).

[1] To be precise, the HTTP reply header includes the directive, but that doesn't matter for the subsequent questions

An important issue is to determine what should go in `cookie`. WebTrust.com offers a number of alternatives.

The first option is to compute the cookie as follows (the operator "+" is concatenation):

```
cookie = expiration-time + MAC(expiration-time, key)
```

`MAC` computes a message authentication code (a string of bytes); it takes as the first argument the string to be MACed and as the second argument a key.

The key `key` is a value known to only the server, which remembers it for only 24 hours. All cookies in that period use the same key. All cookies expire at 5am EST, at which time the server changes the key too.

When the server receives the cookie, it verifies it using:

```
boolean verify (cookie c) {
  if (MAC (c.expiration-time, key) == c.MAC) {
        if (c.expiration-time >= current-time ()) {
            return true;
        }
  }
  return false;
}
```

(The notation `c.field` refers to the corresponding field in the cookie.)

The procedure `verify` recomputes the MAC from the plaintext `expiration-time` stored in the cookie. If the MAC is valid, then the server checks whether the cookie is still fresh (i.e., if the expiration time is larger than the current time). If it is, then `verify` returns `true`; the server can now execute the request. In all other cases, `verify` returns `false` and the server returns an error to the browser.

11. [6 points]: What is the role of the MAC in this protocol?
(Circle ALL that apply)

A. To help detect transmission errors

B. To privately communicate `key` from the server to the browser

C. To privately communicate `expiration-time` from the server to the browser.

D. To help detect a forged `cookie`.

B and C are false; in this protocol nothing is sealed, so nothing could be communicated privately.

12. [6 points]: What attacks does this protocol prevent?
(Circle ALL that apply)

A. Replayed cookies

B. Forged expiration times

C. Forged cookies

D. Dictionary attacks on passwords

***A** is false: an attacker can replay any cookie to gain illicit access, until the cookie expires. **B** and **C** are two ways of saying the same thing, since the cookie contains only the expiration time.*

Another supported option is to compute `cookie` as follows (the operator "+" is concatenation):

```
cookie = expiration-time + username  + MAC(expiration-time + username, key)
```

The server uses for `username` the name of the user in the login request. The usage of this cookie is similar to before.

The corresponding verification procedure for this cookie option is:

```
boolean verify (cookie c) {
  if (MAC (c.expiration-time + c.username, key) == c.MAC) {
        if (c.expiration-time >= current-time ())
            return true;
        }
  }
  return false;
}
```

13. [6 points]: If the server receives a cookie with "Alice" as `username` and the server verifies it successfully (i.e., `verify` returns `true`), what does the server know? (Assume active attacks.)

(Circle ALL that apply)

A. No-one modified the cookie

B. The server accepted a login from "Alice" recently

C. The cookie was generated recently

D. The browser of the user "Alice" sent this cookie recently

Because a cookie is a really a message from the server to itself, only the server knows the MAC key; thus the server will detect an attacker's attempts to create or modify cookies. The server creates a cookie saying "Alice" only if someone successfully logs in with Alice's user name and password; thus **B** *is true. The timestamp allows the server to reject a cookie if it wasn't created recently.* ***D*** *is not true: An attacker may have intercepted the cookie on the way from the server to Alice's browser and is now sending it back to the server.*

14. [6 points]: For this question only, assume that all Alice's Web requests are sent over a single TCP connection that is sealed and authenticated, and that the setup all has been done appropriately (i.e., only the browser and server know the sealing and authentication keys). After Alice has logged in over this connection, the server has received a cookie with "Alice" as the `username` over this connection, and has verified it successfully (i.e., `verify` returns `true`), what does the server know? (Assume active attacks.)

(Circle ALL that apply)

A. No-one but the server and the browser of the user "Alice" knows the cookie

B. The server accepted a login from "Alice" recently

C. The cookie was generated recently

D. The browser of the user "Alice" sent this cookie recently

The server can't know whether Alice's browser, or some other software running on her computer, has given away the cookie. But the server can know that Alice's browser sent the cookie, since it arrived over an authenticated connection, and the question states that only the browser and server know the authentication key.

15. [6 points]: Is there an *additional* security *risk* with storing cookies persistently (i.e., the browser stores them in a file on the local operating system) instead of in the run-time memory of the browser? (Assume the operating system is a multi-user operating system such as Linux or Windows2000, including a virtual memory system.)

(Circle ALL that apply)

A. Yes, because the file with cookies might be accessible to other users.

B. Yes, because the next user to login to the client machine might have access to the file with cookies.

C. Yes, because the trusted computing base is extended to include the local operating system

D. Yes, because the trusted computing base is extended to include the hard disk

C *is not true because the trusted computing base already included the local operating system.* **D** *is not true because the trusted computing base already included the hard disk: the virtual memory of the browser probably includes the hard disk, and the binary of the operating system and the browser were almost certainly stored on the hard disk.*

16. [2 points]: For what applications is WebTrust's product (without the sealing and authenticating TCP connection) appropriate (i.e., useable without grave risk)?
(Circle ALL that apply)

A. For protecting access to bank accounts of an electronic bank

B. For restricting access to electronic news articles to clients that have subscription service

C. For protecting access to student data on MIT's on-line student services

D. For electronic shopping, say, at amazon.com

E. None of the above

Any answer (except E and another choice) is acceptable; this is a matter of judgement, not a strictly technical question. You might find it interesting to know that in practice, Web sites use considerably weaker schemes than the ones you were questioned about, and use them for applications like B, C, and D.

Mark Bittdiddle—Ben's 16-year kid brother–proposes to change the protocol slightly. Instead of computing `cookie` as:

```
cookie = expiration-time + username  + MAC(expiration-time + username, key)
```

Mark suggests that the code be simplified to:

```
cookie = expiration-time + username  + MAC(expiration-time, key)
```

He also suggests the corresponding change for the procedure `verify`. The protocol, as before, runs over an ordinary TCP connection (i.e., unsealed, unauthenticated).

17. [8 points]: Describe one attack that *this change* opens up and illustrate the attack by describing a scenario (e.g., "Lucifer can now ... by ..."); answering, for example, "a replay attack" is insufficient.
(Be brief; use no more than 50 words)

Lucifer can create an account, log into the server, change his cookie's username to "Alice" (or any other user), and send the modified cookie to the server. The server will believe that he is Alice, since the MAC doesn't cover the username. You can read more about the security of cookie authentication schemes here: `http://cookies.lcs.mit.edu/pubs.html`

End of Quiz 2

Name: