CS 161 Computer Security
Fall 2005 Joseph/Tygar/Vazirani/Wagner MT 1 Soln

Problem 1. [Short answer] (30 points)

Give brief answers (one or two sentences) to each of the following.

(a) What is the principle of least privilege? Why is it important?

It states that programs and users should receive only the minimal amount of privilege needed to function correctly, and nothing more. This limits the amount of damage done by the program if it is buggy or hacked.

(b) Is a TCP connection secure against eavesdropping? Why or why not?

No. TCP sends data in cleartext (unencrypted), so an eavesdropper can see all data transmitted.

(c) You have a copy of Anthony Joseph's certificate chain: his certificate is signed by the EECS department; the EECS department's certificate is signed by UC Berkeley; UC Berkeley's certificate is signed by Verisign. Whose public keys do you need to know in advance in order to obtain the correct public key for Anthony?

Verisign's key suffices. In fact, any one of the public keys mentioned is enough.

Problem 2. [Packet filters] (20 points)

(a) We have an internal webserver, used only for testing purposes, at IP address `5.6.7.8` on our internal corporate network. The packet filter is situated at a chokepoint between our internal network and the rest of the Internet. Can such a packet filter block all attempts by outside hosts to initiate a direct TCP connection to this internal webserver? If yes, show a packet filtering ruleset that provides this functionality; if no, explain why a (stateless) packet filter cannot do it.

Yes. An ruleset such as the following will do the trick:

```
drop tcp *:* -> 5.6.7.8:*
```

The following might be a little better, because it does not restrict outbound connections initiated by our internal server:

```
drop tcp *:* -> 5.6.7.8:* (if SYN flag set)
```

(b) Can a packet filter block all incoming email containing the phrase "Make money fast"? If yes, show a packet filtering ruleset that provides this functionality; if no, explain why a (stateless) packet filter cannot do it.

No. The phrase "Make money fast" might be spread across multiple packets (e.g., "Make money " in the first packet, "fast" in the second). A stateless packet filter cannot remember any state from prior packets, so cannot usefully block such email.

Problem 3. [Source Code Analysis] (20 points)

```
/* Escapes all newlines in the input string, replacing them with "\n". */
/* Requires: p != NULL; p is a valid '\0'-terminated string */
void escape(char *p) {
    while (*p != '\0')
        switch (*p) {
            case '\n':
                memcpy(p+2, p+1, strlen(p));
                *p++ = '\\'; *p++ = 'n';
                break;
            default:
                p++;
        }
}
```

You may assume that `escape()`'s argument is always non-null and points to a `'\0'`-terminated string.

What's wrong with this code (from a security point of view)?

Buffer overrun. If the input string contains a newline character, then this will write past the end of the input buffer. In the worst case, the size of the string might double. For instance, if the caller allocates a buffer on the stack that is just large enough to hold the string, and passes it to `escape()`*, then a stack-smashing attack would be possible.*

[Incidentally, another problem is that `memcpy()` *invokes undefined behavior when invoked on overlapping memory regions. You didn't need to notice this.]*

Problem 4. [Crypto] (30 points)

Alice wants to send a cellphone text message to Bob securely, over an insecure communication network. Alice's cellphone has a RSA public key K_A and matching private key v_A; likewise, Bob's cellphone has K_B and v_B. Let's design a cryptographic protocol for doing this, assuming both know each other's public keys.

Here is what Alice's cellphone will do to send the text message m:

1. Alice's phone randomly picks a new AES session key k and computes $c = \text{RSA-Encrypt}(K_B, k)$, $c' = \text{AES-CBC-Encrypt}(k, m)$, and $t = \text{RSA-Sign}(v_A, (c, c'))$.
2. Alice's phone sends (c, c', t) to Bob's phone.

And here is what Bob's cellphone will do, upon receiving (c, c', t):

1. Bob's phone checks that t is a valid RSA signature on (c, c') under public key K_A. If not, abort.
2. Bob's phone computes $k' = \text{RSA-Decrypt}(v_B, c)$ and $m' = \text{AES-CBC-Decrypt}(k', c')$.
3. Bob's phone informs Bob that Alice sent message m'.

(a) Does this protocol ensure the confidentiality of Alice's messages? Why or why not?

Yes. Since AES-CBC-Encrypt is secure, no one can recover m from c' without knowledge of k. Also, since RSA-Encrypt is secure, only someone who knows K_B—namely, Bob—can recover k.

(b) Does this protocol ensure authentication and data integrity for every text message Bob receives? Why or why not?

Yes. Since RSA-Sign is secure, if (c, c') passes step 1, then only someone who knew v_A—namely, Alice—could have sent (c, c'). Now (c, c') uniquely determines m, the message that Alice wanted to send. Conclusion: If Bob accepts m in step 3, then Alice sent m.

(c) Suppose that Bob is Alice's stockbroker. Bob hooks up the output of this protocol to an automatic stock-trading service, so if Alice sends a text message "Sell 100 shares MSFT" using the above protocol, then this trade will be immediately and automatically executed from Alice's account. Suggest one reason why this might be a bad idea from a security point of view.

No protection against replays. An active attacker could replay a valid ciphertext from Alice 10 times, causing 1000 shares to be sold—even though Alice only wanted 100 sold.

Denial-of-service. An active attacker could prevent Alice's ciphertext from reaching Bob. Since Alice doesn't receive any acknowledgement, she will think her trade was executed, when it actually wasn't.

If Alice's cellphone is lost or stolen, then its new owner can cause trades to be executed from Alice's account without Alice's authorization.

[It suffices for you to mention any one of these problems.]