Name: _____

Let's try visiting a web site that is password-protected and examine the sequence of HTTP message exchanged for such a site. The URL http://www.technicalprogramming.ca/CITX2182 is password protected. The username is "CITX", and the password is "citx". So let's access this "secure" password-protected site. Do the following:

- Make sure your browser's cache is cleared, as discussed above, and close down your browser. Then, start up your browser
- Start up the Ethereal packet sniffer
- Enter the following URL into your browser:

  http://www.technicalprogramming.ca/CITX2182

Type the requested user name and password into the pop up box:

- Stop Ethereal packet capture, and enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window.

Now let's examine the Ethereal output. You might want to first read up on HTTP authentication by reviewing the easy-to-read material on "HTTP Access Authentication Framework" at:

http://frontier.userland.com/stories/storyReader$2159

The username (CITX2182) and password (bcit) that you entered are encoded in the string of characters (Q0lUWDIxODI6YmNpdA==) following the "Authorization: Basic" header in the client's HTTP GET message. While it may appear that your username and password are encrypted, they are simply encoded in a format known as Base64 format.

The username and password are *not* encrypted! To see this, go to:

http://www.securitystats.com/tools/base64.php

Enter the base64-encoded string and press decode. You have translated from Base64 encoding to ASCII encoding, and thus should see both your username and password! Since anyone can download a tool like Ethereal and sniff packets passing by their network adaptor, and anyone can translate from Base64 to ASCII, it should be clear to you that simple passwords on WWW sites are not secure unless additional measures are taken.

Answer the following questions:

1. What is the server's response (status code and phrase) in response to the initial HTTP GET message from your browser?

2. When your browser's sends the HTTP GET message for the second time, what new field is included in the HTTP GET message?

In public key encryption a server can publish a public key KS, any browser can then send a message M containing private information to it that has been encrypted using KS:$[M]_{KS}$. This is essentially a simplified version of the service performed by Secure Sockets Layer (SSL). SSL can perform encryption and authentication in both directions. As a result http requests and responses can be protected from eavesdropping and manipulation.

Simple Version of SSL as implemented on a secure web site.

1. Client sends 'client hello' message to the server. This contains its name C, transaction serial number C#, and a random nounce N_C.
2. Server responds with a 'server hello' message. This contains its name S, transaction serial number S#, a random nounce N_S, and a certificate CS containing its public key KS. The client verifies the certificate CS using a root certificate issued by a company such as Thawt and stored in the browser.
3. The client sends a 'key exchange' message containind a 'pre-master-secret' key K_0, enciphered the server public key KS. It also sends a finished message with a message authentication code (MAC) computed on all messages to date. The key for this MAC is the master-secret K_1. This key is computes by hashing the pre-master-secret key with the nounces sent by the client and server: $K_1 = h (K_{CS}, N_C, N_S)$. From this point on the traffic is encrypted $\{\ldots\}_{KCS}$ in the client to server direction and $\{\ldots\}_{KSC}$ in the server to client direction. The keys are generated in turn by hashing the nounces with K_1.
4. The server also sends a finished message with a MAC computed on all the messages to date. It then finally starts sending the data.

$C \rightarrow S$: C, C#, N_C
$S \rightarrow C$: S, S#, N_S, CS
$C \rightarrow S$: $\{K_0\}_{KS}$
$C \rightarrow S$: $\{\text{finished, MAC } (K_1, \text{everything\_to\_date})\}_{KCS}$
$C \rightarrow S$: $\{\text{finished, MAC } (K_1, \text{everything\_to\_date})\}_{KSC}$

Now, let's try visiting a web site that is password-protected and uses SSL to protect privacy. We will examine the sequence of HTTP messages exchanged. The URL https://www.technicalprogramming.ca/ is encrypted and password protected. The username is "CITX", and the password is "citx". So let's access this "secure" password-protected site. Do the following:

- Make sure your browser's cache is cleared, as discussed above, and close down your browser. Then, start up your browser
- Start up the Ethereal packet sniffer
- Enter the following URL into your browser:

  https://www.technicalprogramming.ca/

Accept the certificate and type the requested user name and password into the pop up box:

- Stop Ethereal packet capture, and enter "tcp" in the display-filter-specification window, so that only captured HTTPS messages will be displayed later in the packet-listing window.

Carefully examine the sequence of events that sets up the SSL connection.

3. Is the sequence exactly as described in the SSL explanation listed above? If not how is it different?

4. Can you locate the client to server packet containing the user name and password? If so explain why and how, and if not explain why and how.

5. Can you think of additional steps that you would take to make this protocol more secure?

6. The certificate on this site was not acquired from a CA but was compiled by the site administrator. With regard to this do you consider this site is safe to use for credit card transactions? Explain.

Mark /10