University of Maryland
CMSC414 — Computer and Network Security
Professor Jonathan Katz

Problem Set 3

Due by Nov. 15, 11:59 PM

The goals of this assignment are: (1) to get hands-on experience reverse-engineering and attacking protocols, as well as (2) to learn a bit about authentication protocols and their potential flaws. This homework is challenging, but hopefully you will enjoy it!

A FAQ will once again be posted on the course homepage.

This homework has two parts: first, you will attack a widely-used authentication protocol by playing the role of a passive eavesdropper. In part 2, you will attack a protocol developed by the TAs; here you will have the ability to act as an active adversary as well.

Part 1: You will attack the basic authentication protocol used by Yahoo! mail.[1] Yahoo! uses a variation of the following authentication protocol discussed in class: Assume a user U and a server S share a password pw_U. To begin, S sends a random challenge r to the user, who computes $t = H(pw_U, r)$ for some cryptographic hash function H. The user sends (U, t) back to S, and S accepts U if and only if $t \overset{?}{=} H(pw_U, r)$. In slightly more detail:

1. When a user loads the site http://mail.yahoo.com in their browser, Yahoo! includes a random challenge r as part of the page. The page also includes various fields into which the user can type their username and password, along with javascript code whose purpose will be described next.

2. The user types their username U and password pw into the appropriate places on the page. When the user presses the "Sign In" button, this invokes a javascript program which essentially computes (a variant of) $t = H(pw_U, r)$ and sends (U, t) (along with other information) back to the Yahoo! server, as described above.

3. The Yahoo! server compares the received value t with the value $H(pw_U, r)$ computed on their end. The user is allowed to access their email if these values match.

The above description omits some details which you will have to figure out on your own as part of this project. The description also simplifies some aspects of the protocol which are not relevant for this project.

In class we discussed why the above protocol is vulnerable to *off-line dictionary attacks* if a "short" password is used. Such an attack only requires eavesdropping on a single execution of the protocol. Unfortunately, you cannot install a packet sniffer on the cluster machines and so you cannot actually "eavesdrop" on an execution of the protocol. Instead,

[1] Note that many other sites use similar protocols, and Yahoo! was picked arbitrarily as an example. Also, the vulnerability you will be exploiting here is widely known and only works when users choose "short" passwords; Yahoo! offers a more secure authentication protocol which avoids the attack, and also recommends that users choose difficult-to-guess passwords.

we will provide each team with a set of simulated transcripts obtained by eavesdropping. Your goal is to recover the password(s) being used in each case.

In full detail, you will need to do the following:

1. Load http://mail.yahoo.com in a javascript-enabled browser. Make sure that you are on the page implementing "standard" authentication. View the page source and then copy and paste this source code into a text editor; save it in a file called attack.html.

2. *The first thing you should then do* is to make the changes in attack.html as specified on the FAQ linked from the course homepage. After making these changes, you will then be able to examine and experiment with the source code (by loading the file attack.html in your browser) to get an understanding of how it works. Javascript tutorials will be linked from the FAQ as well.

3. As mentioned earlier, each team will be given transcripts of an execution of the authentication protocol. (These will be available from the course homepage.) Your ultimate goal is to figure out the password being used in each of your assigned transcripts. *Each team will receive different transcripts, and the passwords used in the various transcripts will all be different.* Further information about the space of possible passwords will be available along with the transcripts.

4. In general, it does not matter how you figure out the password as long as you do it yourself without any outside help. The easiest way to do it (in my opinion) is by modifying the Javascript source code, but if you like you can use "pure" Java or C or anything else you like.

   Important hint: Keep in mind that *you do not need to understand every single line of the program* in order to get an overall sense of what is going on, nor do you need to understand the "low-level" details of what is going on in order to attack the protocol. Don't waste time trying to understand things that are not necessary in order to implement your attack.

5. **What to turn in.** The main thing to turn in is your guess for the passwords of your assigned transcripts. The precise method for turning this in will be described in the online FAQ. In addition, you should turn in (1) a README file containing a high-level description of how the Yahoo! authentication protocol works ("high level" means along the lines of what I show in class, where you simply indicate the flows and what information is sent in each flow) as well as a brief explanation of how you attacked the protocol and recovered the password(s); and (2) any code you wrote for this part.

Part II. You will attack a credit-card authorization protocol written by the TAs. Here, you will have the ability to passively eavesdrop on as many protocol executions as you like and will also have the ability to act as an *active* adversary who controls the messages sent between the parties.

You will download a TAR file from the course homepage which contains three programs: a *client*, a *server*, and a "router". The first two will be given to you as executables; you will be given the source code only for the router. (Information on how exactly to run the client/server/router will be included as part of the FAQ.) The client and server communicate via the router (e.g., when the client wants to send a message to the server, it sends it to the router which forwards it to the server), and the router is initially configured to simply forward messages unchanged between the two parties. Since you have "compromised" the router, you can modify its behavior and thereby carry out an active attack.

The client and server are set up to execute a protocol in which the client communicates a credit card number to the server. The company which developed this protocol has kept the details of the protocol secret in order to improve its security[2] and all you initially know about the protocol is the following:

- The goal of the protocol is to securely transmit a credit card number from the client to the server. The credit card number is 10 decimal digits long. For a given team, the client always uses the same credit card number.[3]
- The protocol consists of two phases. In the first phase, the client authenticates that it is communicating with the server. In the second phase, the client sends its credit card number to the server.
- The server has established a public-/private-key pair, and its public key is known to the client. The server's public key is widely available, and you can find it on the FAQ.
- All the components of the protocol were covered in class and/or in the textbook.

What to turn in. The main thing to turn in is your guess for the client's credit card number. The precise method for turning this in will be described in the FAQ. In addition, you should turn in (1) a README file containing a high-level description of both phases of the protocol ("high level" means along the lines of what I show in class, where you simply indicate the flows and what information is sent in each flow) as well as a brief explanation of how you attacked the protocol and recovered the credit card number; and (2) your modified code for the router (and also any other code you wrote for this part).

If you get stuck. One of the goals of this project is to learn how to reverse engineer a protocol by monitoring (and possibly modifying) the messages sent. If you find that you are unable to figure out the protocol, however, you can ask the TAs to tell you the protocol for a 10 point penalty. This is an "all or nothing" proposition: even if you have figured out half the protocol, it will cost you 10 points to learn the other half.

[2]It seems the engineers writing the protocol have not taken this course!

[3]Note, however, that the credit card number used by the client will be different for each team.