ECE 478 / CS 419: **Network Security**

Homework 4: due Friday May 9, 10am

Instructions:

- Submit typed solutions, following the rules in the syllabus.
- Include the "disclaimer" given in the syllabus!

This homework refers to the following website:

http://web.engr.oregonstate.edu/cgi-bin/cgiwrap/rosulekm/hash-hw.cgi

You can check out the source code here:

http://web.engr.oregonstate.edu/cgi-bin/cgiwrap/rosulekm/highlight.cgi?hash-hw.cgi

Similar to the previous homework, this form allows you to create a new account on a system. When creating a new account, you receive a cookie with a cryptographic token that grants access to this account. The "`admin`" username is special. If anyone accesses the page with a valid token for `admin`, some secret administrative announcement will be displayed. However, the page does not allow you to create a new account named `admin`. Nevertheless, your goal in this homework is to obtain the secret admin announcement.

In the last homework, the page used encryption incorrectly, where authentication was called for. In this page, we at least attempt to use some authentication. However, the author of the page couldn't remember how HMAC works, and was too lazy to look it up. Instead, he implemented a MAC algorithm that has some similarities to HMAC but is vulnerable to attack. The page computes $\mathsf{MAC}(k, m)$ as $\mathsf{MD5}(\mathsf{MD5trunc}(m) \| k)$, where "$\|$" is concatenation, and $\mathsf{MD5trunc}(m)$ is the MD5 hash of m truncated to the first 32 bits (4 bytes). *What could possibly go wrong?*

1. Describe clearly the format & contents of the cookie set by the server. How does the page authenticate a user's cookie?

2. When you create an account and get an associated token, what parts of the token can you control completely? What parts can you predict?

3. What must a token contain in order to confer `admin` access? What parts of the token don't matter so much?

4. Explain how a collision under $\mathsf{MD5trunc}$ could be used to gain a valid `admin` token. How much effort should it take to find collisions under $\mathsf{MD5trunc}$?

5. OK, now find a collision under MD5trunc satisfying the properties that you described in the previous questions. Describe your approach for finding the collision (provide your code if it's not too long, but a short description/pseudocode will be enough), and also show the actual values that collide.

   Notes: I chose to use MD5 because it probably has the most support across all programming languages. You should be able to find good support in your preferred coding environment. In lieu of a library, you can compute MD5 hashes from the Linux command line:

   ```
   $ echo -n "hello world" | md5sum
   5eb63bbbe01eeed093cb22bb8f5acdc3  -

   $ echo -n "hello world" | md5sum | cut -c1-8
   5eb63bbb
   ```

   `echo -n` pipes its argument to `md5sum` without adding an extra newline. The `cut` command prints out only the first 8 characters (`md5sum`'s output is hex encoded, so this corresponds to the first 4 raw bytes of the MD5 hash).

6. Describe the mechanics of your final attack to gain `admin` access. What data did you send to the form? What did you get back in response? How did you manipulate it to get admin access?

7. Let's find out how sensitive this problem is to the fact that we truncated the inner MD5 hash to 4 bytes.

   I have set up the page so that when you access it via hash-hw.cgi?5, hash-hw.cgi?6, etc, it will truncate the inner MD5 hash to 5 bytes, 6 bytes, etc, instead of 4 bytes. Authentication tokens generated with one truncation-length will not be compatible with other lengths. Also, the secret admin messages are different for the different truncation-lengths.

   How many of these truncation-lengths can you compromise to gain admin access? For each one that you were able to break (including length=4), list the hash collision and the secret admin message.