Homework 06

**Due: 1:00 pm, Friday March 6th**

Please hand in your homework as a single PDF file by running `cs1951e_handin hw06` in a folder containing only that file on a department machine. While we will not take points off for turning in extraneous files, it makes our lives difficult. **Handins that are not in PDF format will not be accepted.**

Note that homework is due before class, that is, at 1:00 pm, not 1:00 am.

Malware

Problem 1

It's very difficult to figure out what a program will do, and that includes deciding whether its behavior is malicious (in fact, in the general case, it's provably impossible). This makes malware detection a daunting task.

Below, you are given a simple Go function which sanitizes user logins by hashing passwords so they can be safely transmitted over the network. However, it doesn't behave exactly as you might think. What is its malicious behavior? Explain how this is achieved. While we can't stop you from running the code, please try to figure it out on your own first.

```go
import (
    "crypto/md5"
    "encoding/hex"
    "strings"
)

// sanitizeLoginString takes an input of the form user:pass:[domain]
// (domain is optional) and returns the string mutated so that the
// password is hashed with MD5.
//
// For example,
// foo:bar:baz -> foo:37b51d194a7513e45b66f6524f2d51f2:baz
// foo:bar: -> foo:37b51d194a7513e45b66f6524f2d51f2:
func sanitizeLoginString(s string) string {
    parts := strings.Split(s, " ")
    passIdx := len(parts) - 1
    domain := s[len(s)-1] != ':
    if domain {
        passIdx = len(parts) - 2
    }
    hash := md5.Sum([]byte(parts[passIdx]))
    parts[passIdx] = hex.EncodeToString(hash[:])
    return strings.Join(parts, ":")
}
```
DoS

Problem 2

You know one of your CS1951-E classmates has sent you an envelope full of glitter, but you don’t know what the outside of the envelope will look like. So, logically, you decide to DoS your mailman, as that is the only way you can be sure you won’t receive that letter. Assume that the mailman follows the protocol below, and is otherwise naïvely designed. List five methods that may manage to disrupt the service. You should include at least one of each of the three main sub-types: volume-based, protocol-layer, and application-layer attacks. Be creative!

Protocol:

- Each envelope has an addressee’s name on it, and a return-address name.
- While the mailman continues to be able, he will:
  1. Walk up to each house in order down the street, and knock on the door.
  2. If someone appears within a brief period of time, he will (otherwise he will skip these steps):
     1. Look through the letters he has to check if there is mail addressed to this person, or if there is any mail sent from this person which is also marked as return to sender, and if so, give each one to the person.
     2. Ask if they have mail to send, and if so, wait to receive the letters from them.
     3. Once he has visited all the houses, he checks if any letters were unable to be delivered, and marks them as to return to sender.
     4. If he was unable to return a letter to sender, he drops it in a trash bin at the post-office.

Trust

Problem 3

A huge amount of trust is involved in driving a car - trust in the car and its manufacturer, trust in the roads, trust in the traffic signaling systems, trust in other drivers and other drivers’ cars, etc. Answer the following questions about the trust involved in driving a car, and explain your answers.

(a) Provide one example of implicit transitive trust. How might we make that trust explicit?
(b) Provide one example of explicit transitive trust. In what way is this trust explicit?
(c) Would it be feasible to make the trust involved in your answer to either (a) or (b) intransitive? How would you enforce this?
(d) Provide one example of intransitive trust.

Problem 4

Authentication and authorization are closely related to trust. Provide one example of an authentication or authorization system which uses each of the following:

(a) Implicit transitive trust
(b) Explicit transitive trust
(c) Intransitive trust
Web Servers Pre-work

These are questions to which we do not expect you to know the answer. We don’t expect you to know the details underlying the questions. As long as you state any (reasonable) assumptions you are making, and be explicit about what your understanding of the background material is, we will give you most of the credit so long as the reasoning based on that understanding is sound. Think of it like a math problem - show your work, and we won’t take off too much for minor calculation mistakes.

Problem 5

Many web sites written in interpreted programming languages place code files throughout a directory tree which mirrors the URL structure of the website. For example, if the web root of www.foo.com is /var/www, requesting http://www.foo.com/about.php might execute the PHP script at /var/www/about.php, and requesting http://www.foo.com/users/list.php might execute the PHP script at /var/www/users/list.php. Additionally, some websites store uploaded files in directories inside the web root so that they can be requested. For example, a web server might place uploads in /var/www/uploads, so that if you uploaded bar.jpg, requesting http://www.foo.com/uploads/bar.jpg would result in being served the image stored at /var/www/uploads/bar.jpg. How might a malicious user gain unauthorized access to a web server employing both of these techniques?

Problem 6

For this problem, you are not permitted to do any research or consult any outside sources other than the course staff. Imagine that you have a web server with two components: one which decides whether a user is permitted to access a given URL (the “enforcement engine”, and one which actually serves URLs (the “serve engine”. When a user requests a URL, the enforcement engine first checks to make sure they have permissions. If they do, the enforcement engine invokes the serve engine to serve the URL to the user.

Some web servers are configured to interpret URLs corresponding to folders on the filesystem as referring to a listing of the files and folders inside that folder. For example, if I were to request http://www.foo.com/folder/ on such a server, I would get back a list of the files and folders in /var/www/folder (assuming /var/www was the web root like in the previous problem).

Imagine that I have a web server with separate enforcement and serve engines, and that the policy for the website is that users are not allowed to see directory listings (in fact, if they request URLs corresponding to folders, they will get an error). Also imagine that the serve engine is written in a language with null-terminated strings (like C), but that the enforcement engine is written in a language without null-terminated strings (like Java or Go). How might a malicious user cause the server to give themselves a listing of files and folders in a directory?