Advanced Homework 10

Due: Wednesday, November 22nd, 11:59PM (Hard Deadline)

Submission Instructions

To receive credit for this assignment you will need to stop by someone's office hours, demo your running code, and answer some questions. **Make sure to check the office hour schedule as the real due date is at the last office hours before the date listed above.** This applies to assignments that need to be gone over with a TA only.

!!! IMPORTANT NOTE !!!

There is a known bug associated with the AFS portion of this exercise. If you download a package during the AFS portion that crashes your Ubuntu machine, then you are afflicted by the bug! :(Perform the following steps to see if the bug would affect you and how to fix it ahead of time.

If the output of running `uname -r` is something that begins with `4.8` you will likely need to follow one of the following actions:

1. If you don't want to upgrade the Linux Kernel or encounter issues when upgrading the Kernel, you can also download and run Ubuntu 17.04 as a new virtual machine. This should come with a newer version of the Linux Kernel that will work for this assignment.
2. If you are running Ubuntu **16.04**, run the following command to upgrade your version of the Linux Kernel.

   ```
   $ sudo apt-get install --install-recommends linux-generic-hwe-16.04 \
     xserver-xorg-hwe-16.04
   ```

   This will probably take about 10-15 minutes to perform. Afterwards, you will need to reboot and run `uname -r` to see if the upgrade took place properly.

The Andrew File System

One thing you've hopefully noticed is that when you log on to any CAEN machine (or ssh to any CAEN login server) all of your files are magically there. That's because your files are stored using AFS, a network file system. When you save files on a CAEN machine, they don't actually save to the local hard disk, instead they go out over the Internet to a machine in a server room somewhere.[1] File reads similarly go over the network to grab the data, so everything stays in sync automatically.[2] Note that while this is similar to Dropbox, it is slightly different as well since Dropbox stores and edits local copies on disk and then syncs them in the background. That wouldn't work at the university scale, imagine if every CAEN machine had to have enough disk space to hold every file for every student! This means that AFS will only work if you are online.

Since AFS is a networked file system, you can also set up your local machine to join the network. This way, whenever you change anything on your machine, it'll automatically update on CAEN and vice-versa. You could run your text editor on your machine and your compiler on CAEN if you like.

[1] Last I checked, for folks with uniqnames starting a-k, this was the MITC Data Center on Oakbrook off of State. I don't recall where the rest of the alphabet is stored. I think it was in the Southfield data center at one point.

[2] This glosses over some details about caching that's done for performance, but the basic idea is there.

Kerberos

To get things working, there are two key things you need to get set up, the first is **Kerberos**. Kerberos is a tool for authentication. You've been using it every time you log into CAEN (or Cosign[3]). With Kerberos, you must periodically (every 24 hours in umich's setup) enter your password to get a new *ticket*. Once you have this ticket on your machine, AFS will use it to gain access to the files that you own.

Big Hint: You will need a configuration file (`/etc/krb5.conf`). Don't try to write this yourself! Instead login to a CAEN machine and grab a copy from that machine.

AFS

The next thing you'll need to install and set up is AFS. Notice that once you install AFS you will have *list* permissions on much of the directory hierarchy. This will let you explore a little, such as seeing that files exist, but not their contents.[4] manage For example, while holding credentials for `cgagnon`, I try to look at thealex's (a past IA for the course) files, I can see the files in his home directory, but not what's in them:

```
$ ls /afs/umich.edu/user/m/m/thealex
ls: cannot access /afs/umich.edu/user/t/h/thealex/Shared: Permission denied
ls: cannot access /afs/umich.edu/user/t/h/thealex/Private: Permission denied
ls: cannot access /afs/umich.edu/user/t/h/thealex/Desktop: Permission denied
ls: cannot access /afs/umich.edu/user/t/h/thealex/Downloads: Permission denied
ls: cannot access /afs/umich.edu/user/t/h/thealex/Templates: Permission denied
ls: cannot access /afs/umich.edu/user/t/h/thealex/Documents: Permission denied
ls: cannot access /afs/umich.edu/user/t/h/thealex/Music: Permission denied
ls: cannot access /afs/umich.edu/user/t/h/thealex/Pictures: Permission denied
ls: cannot access /afs/umich.edu/user/t/h/thealex/Videos: Permission denied
ls: cannot access /afs/umich.edu/user/t/h/thealex/hs_err_pid12482.log: Permission denied
ls: cannot access /afs/umich.edu/user/t/h/thealex/Game.cpp: Permission denied
ls: cannot access /afs/umich.edu/user/t/h/thealex/os_x_fuse_demo: Permission denied
ls: cannot access /afs/umich.edu/user/t/h/thealex/rawr.cpp: Permission denied
ls: cannot access /afs/umich.edu/user/t/h/thealex/repos: Permission denied
ls: cannot access /afs/umich.edu/user/t/h/thealex/c4cs: Permission denied
ls: cannot access /afs/umich.edu/user/t/h/thealex/krb5.conf: Permission denied
ls: cannot access /afs/umich.edu/user/t/h/thealex/test.txt: Permission denied
ls: cannot access /afs/umich.edu/user/t/h/thealex/484: Permission denied
ls: cannot access /afs/umich.edu/user/t/h/thealex/dotfiles_old: Permission denied
ls: cannot access /afs/umich.edu/user/t/h/thealex/oradiag_thealex: Permission denied
$ cat /afs/umich.edu/user/t/h/thealex/test.txt
cat: /afs/umich.edu/user/t/h/thealex/test.txt: Permission denied
```

If you find yourself getting a similar set of permission denied issues in your own home directory, it means you are connected to AFS correctly, but you haven't connected Kerberos and AFS correctly.

The Assignment

Get AFS installed and working on your local machine. You should have read/write access to your own files.

The writeup in this assignment gives you some intuition for what's going on and the steps you'll have to take, but it is not a step-by-step guide – there are plenty of those out there (I even found a few umich-specific ones for the Kerberos half, not that there's anything special you have to do to set things up to work here, but they use umich as their example).

Note for users doing this on their Mac: OpenAFS has not released a binary compatible with the latest few macOS versions. I've seen some people use a commercial package, but have not tried it myself.

[3] You can actually set up your web browser to not need your password either. It can also use the same Kerberos ticket. I know this is do-able in Firefox, it may be do-able in other browsers as well.

[4] You may notice the POSIX permissions don't line up. That's because AFS uses ACLs to control permissions. Run a `fs help` or `man fs` for more information.

Submission checkoff

- ☐ Show the commands necessary to get AFS working locally
- ☐ Show that creating/modifying a file via the local AFS directory is also reflected when logged into CAEN

`Expect` Respect

This sections touches on a handy scripting language called `expect` that can be used to automate interactions with remote servers. What we'll be doing is partially automating the login procedure for Michigan servers. We say partially because of the two-factor step during the login procedure.

First you'll need to install `expect` for your system (Ubuntu and MacOS both have packages available for download).

Some guiding steps this script should accomplish are:

1. Prompt the user for their uniqname password and store it in a variable. This output should **not** be displayed on the screen when the user types in their password!
2. Initiate a connection to CAEN and `send` the password previously provided by the user.
3. Select one of the Duo options to fire off the push/call/text automatically.

Although the creating of this script doesn't save too much time during the login process for CAEN, this tool can be useful for a lot of other tasks where you are using key-based access to servers and aren't prompted for passwords or Duo options.

As always, there are multiple ways to accomplish some of the tasks above. A potential solution requires no more than about 20 lines of `expect` scripting.

Submission checkoff

- ☐ Explain why it's a bad idea to hardcode the password for the user into the expect script
- ☐ You'll almost certainly have `\r` in your script. Explain what this means/does
- ☐ Show off your script working
  - ☐ Prompting the user for their password (and not displaying it on the screen)
  - ☐ Automatically selecting one of the Duo options
  - ☐ Successfully logging the user into CAEN