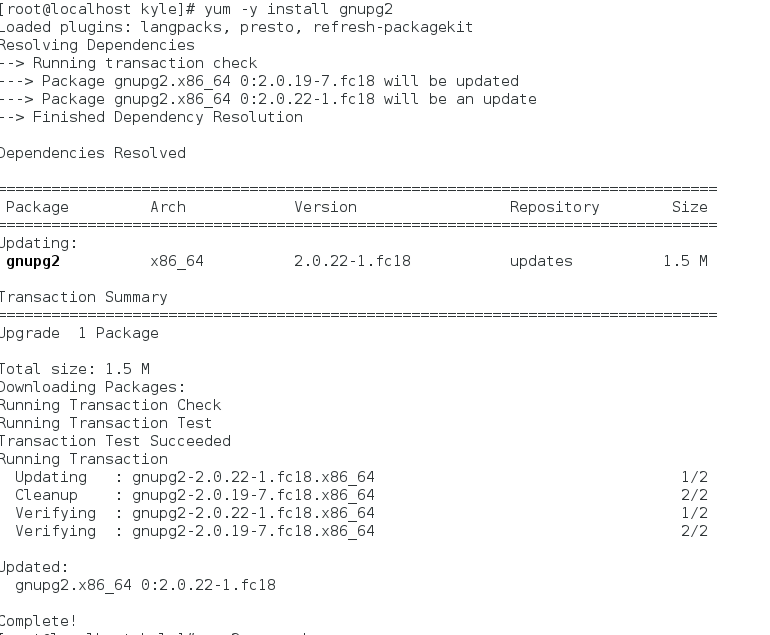
**Project 3: PGP and User Authentication**

**TCSS 431: Network Security**

**Professor Wei Cheng**

**By Eduard Klimenko & Kyle Beveridge**

**2.1) 1.1 Install GnuPG（1 point）**



**2.2.0) Type the following key pair to generate a PGP key pair: （3 points）**

****

**2.2.1) Which folder contains the newly generated key pair? （2 points）**



**2.3.1) After all that setup, we can start signing and verifying documents. Signing a document to prove its authenticity is very simple with GPG2. You only need to provide your private key’s passphrase. Assume you have a file called document.txt. To sign a document, just run: （2 points）**



**2.3.2) You can also pass in --armor, if you need plaintext output. This will create a file called document.txt.gpg or document.txt.asc, depending on whether or not you armor the output. To verify a signature, just run: （2 points）**



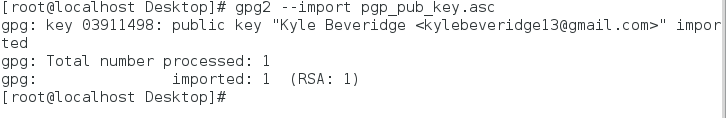
**2.4.0) *gpg2 --export -a KEYID > pgp\_pub\_key.asc*（2 points）**



**2.4.1) What is option to extract the private key? （2 points）**

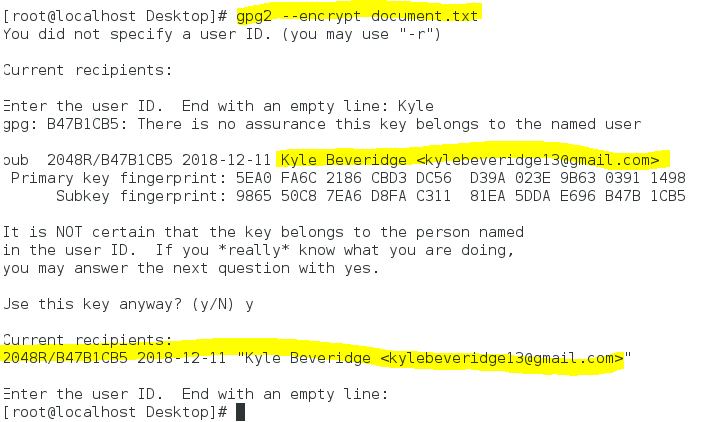


***gpg2 --import pgp\_pub\_key.asc*（2 points）**

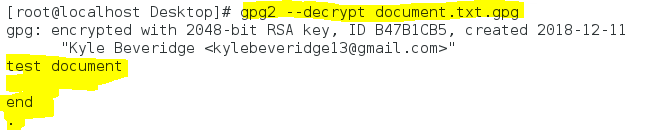


**If you have a file called document.txt on your client that you’d like to encrypt, just run: （1 point）**

**You’ll be prompted to enter the user IDs to whom you’re sending the encrypted document: （2 point）**



**Go back to FC server, to decrypt the transferred document, you can just run: （2 points）**



**2.4.2) Did the decrypted document match the original document on FC client? （2 points）**

Yes, the decrypted version matched the original. Here is the original from client:



**2.4.3) How to register your PGP keys? （2 points）**

--send-keys KeyID’s --keyserver Server

**2.4.4) What is option to delete the PGP keys on your local machine? （2 points）**

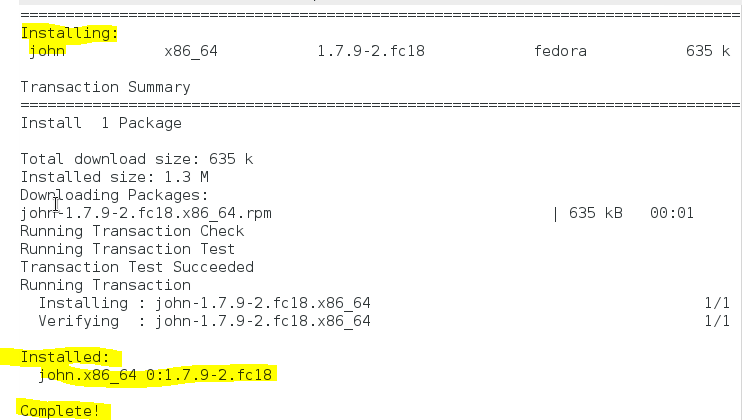
Public key:

--delete-key name

Secret key:

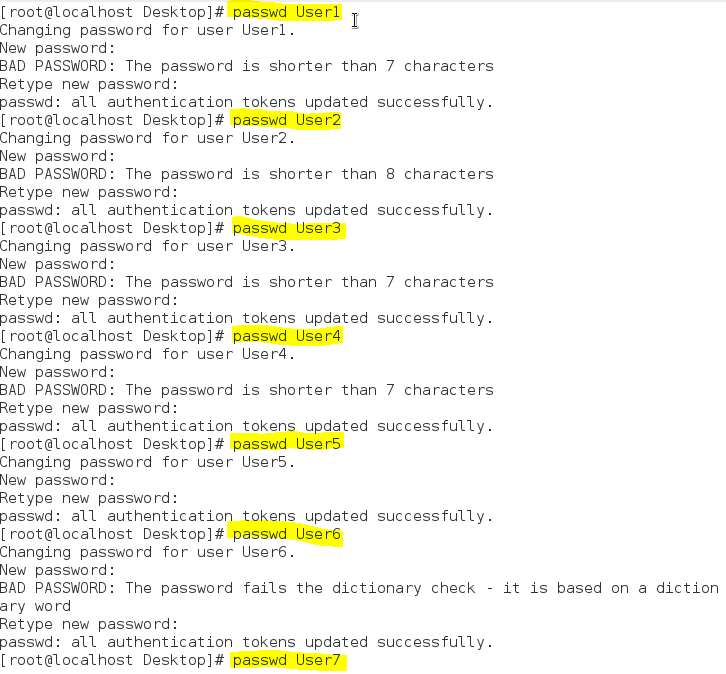
--delete-secrete-key name

**# yum -y install john（1 point）**



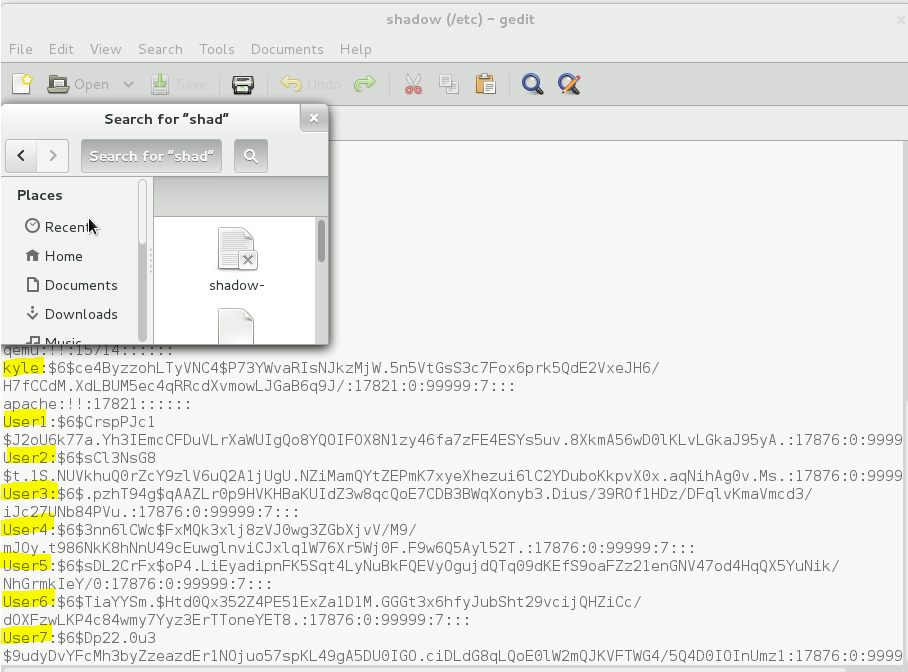
**Create the following user accounts on your Fedora virtual machine using command line. The command to add a user is *useradd*. For example, you can useradd User1 to add User1.**

**The command to set a user’s password is *passwd*. For example, you can passwd User1 to set its password. （1 point）**



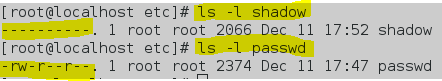
**3.2.1) Where are the usernames and encrypted passwords stored? （2 points）**

They are stored in etc/shadow



**3.2.2) What are the differences between */etc/passwd* and*/etc/shadow*?Check the properties of these two files using the command *ls -l filename*. （2 points）**

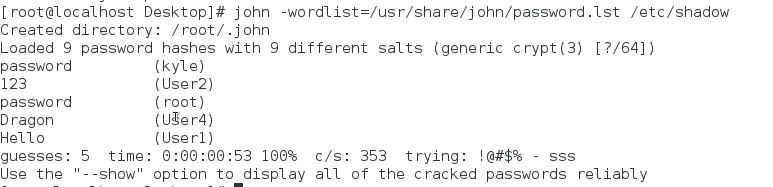
Passwd is more for storing account info, while shadow stores the actual passwords



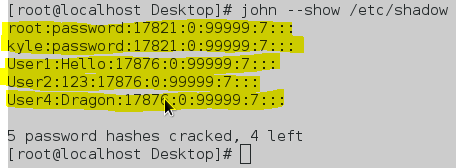
**3.2.3) Why are these two files setup this way? （2 points）**

It is set up this way mostly due to legacy reasons. Passwords used to be stored in passwd, but eventually it was moved out to shadow, which is only readable by the root user. Passwd was kept around for compatability purposes, plus it stored account info

**Run John the Ripper on the shadow password file you just created.（2 points）**



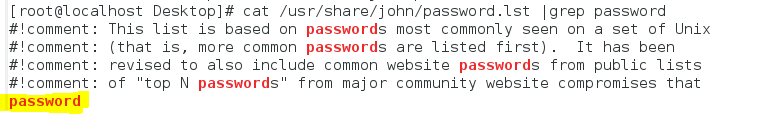
**To display the cracked password（2 points）**

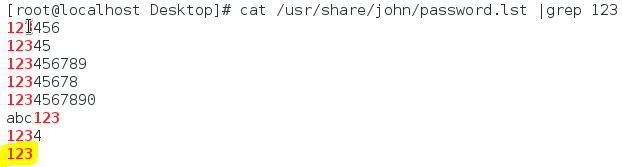


**3.3.1) Write down how many passwords has been cracked and what passwords they were. （2 points）**

5 passwords were cracked: root, my main user account, and users 1, 2, and 4

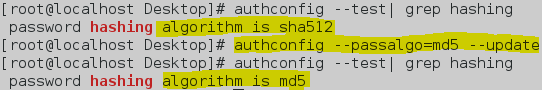
**3.3.2) Do you see any passwords that were just created on the FC machine? （2 points）**







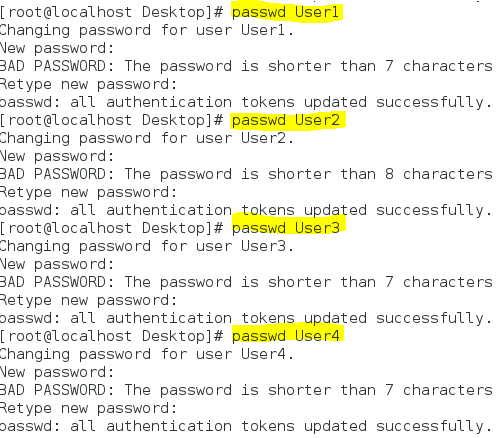
**We can tell the system to create users’ passwords in MD5 format. On another Fedora VM, type the following commands. （2 point）**

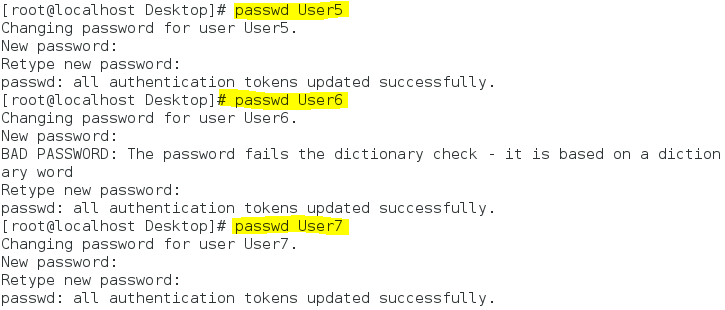


**Repeat the steps in section 3.2 and 3.3. （6 points）**

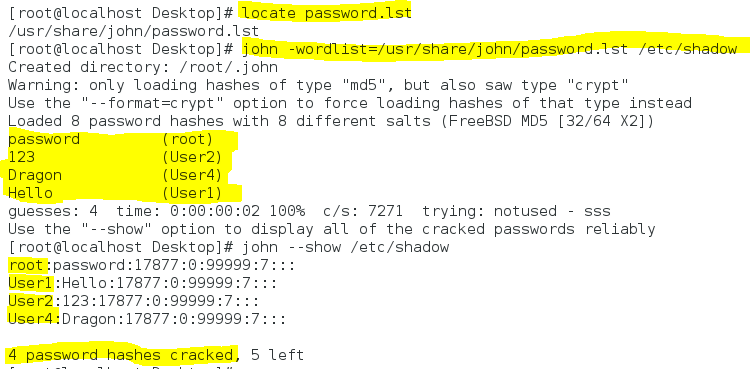
Making users:





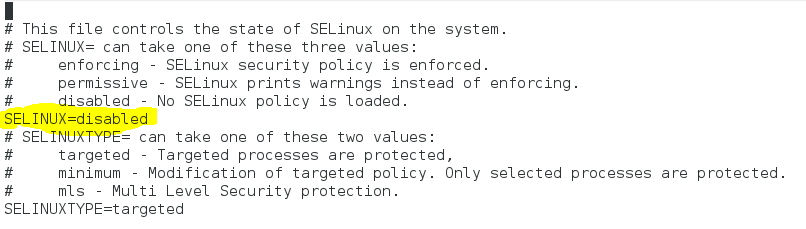


Cracking the passwords:

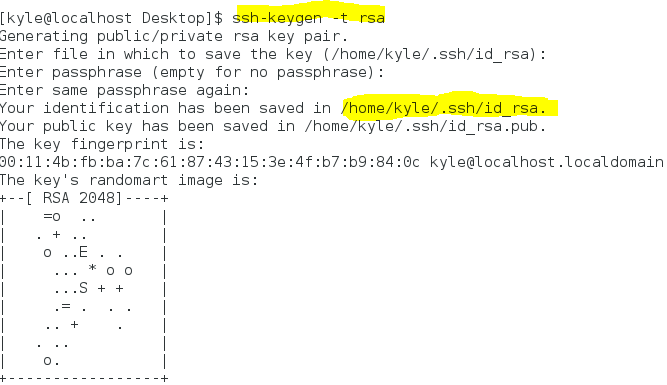


It was only able to crack 4 passwords this time, compared to 5 last time, even though the user account password was still password

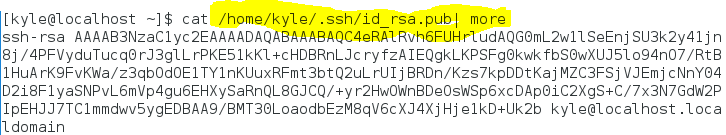
## **Disable the SELinux（2 points）**



### **Create a 2048 bit RSA key on your client VM as a regular user (e.g., student):（5 points）**



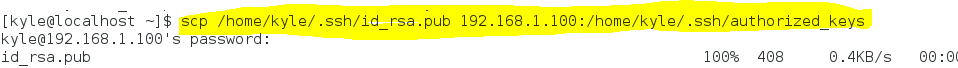
**You can view the public key by:（2 points）**



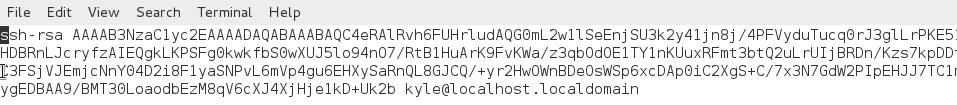
**You can view the prive key by:（2 points）**



**Transfer the /home/user/.ssh/id\_rsa.pub file to the server you want to connect to. You can use scp to copy.（2 points）**

****

**Concatenate the contents of id\_rsa.pub to /home/user/.ssh/authorized\_keys（3 points）**



**Remove the public key file: （1 point）**

****

**Start the sshd service on the serevr:（1 point）**



**Login to the client where you created the RSA key pairs. Try to connect to the server where you copied the public key to using the ssh command.（2 points）**

It asks for a key password since i set up the key pair with a password

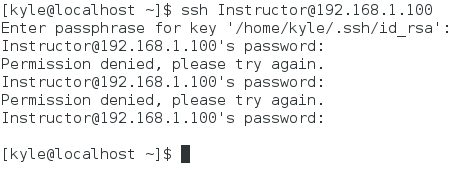


**You can see more information about what is happening by typing:（2 points）**



**In addition, you can place the user’s public key into another user’s (e.g.,Instructor or root) .ssh directory. Test what will happen when you try to log in as another user. For example,（2 points）**

Cant log in since the user’s public key isn’t on the server



**4.4.1) What are the pros and cons of this type of authentication approach?（2 points）**

If you didn’t use a password on your private key, it would be easier to log in to the server than without. However in this case, someone could steal your private key to get into the server, which would be easier than cracking a password in many cases

## **Use Digital Signature Algorithm (DSA)（10 points）**

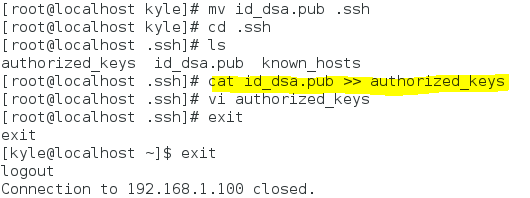
Generate key:



Move key to server:



Add to authorized keys:



Testing login:

