SSH and Certificates

1 SSH with public-key crypto

- SSH is a program that allows you to log into another system over the Internet. You may have used it to log into basin.cs.middlebury.edu remotely. Read the AUTHENTICATION part of the ssh(1) manpage to learn how you can create and use a public/private keypair to authenticate rather than a password. What are the tradeoffs?

  (The ssh program comes in the openssh package.)

We typed man ssh in cmd. Here is the most important part:

```
AUTHENTICATION
     ...
     ...
     ...
     Public key authentication works as follows: The scheme is based on public-key cryptography, using cryptosystems where encryption and decryption
     are done using separate keys, and it is unfeasible to derive the decryption key from the encryption key.  The idea is that each user creates a public/private key pair for authentication purposes.  The server knows the public key, and only the user knows the private key.  ssh implements public key authentication protocol automatically, using one of the DSA, ECDSA, Ed25519 or RSA algorithms.  The HISTORY section of ssl(8) contains a brief discussion of the DSA and RSA algorithms.

     The file ~/.ssh/authorized_keys lists the public keys that are permitted for logging in.  When the user logs in, the ssh program tells the server which key pair it would like to use for authentication.  The client proves that it has access to the private key and the server checks that the corresponding public key is authorized to accept the account.
     The server may inform the client of errors that prevented public key authentication from succeeding after authentication completes using a different method.  These may be viewed by increasing the LogLevel to DEBUG or higher (e.g. by using the -v flag).

     The user creates their key pair by running ssh-keygen(1).  This stores the private key in ~/.ssh/id_dsa (DSA), ~/.ssh/id_ecdsa (ECDSA), ~/.ssh/id_ecdsa_sk (authenticator-hosted ECDSA), ~/.ssh/id_ed25519 (Ed25519), ~/.ssh/id_ed25519_sk (authenticator-hosted Ed25519), or ~/.ssh/id_rsa (RSA) and stores the public key in ~/.ssh/id_dsa.pub (DSA), ~/.ssh/id_ecdsa.pub (ECDSA), ~/.ssh/id_ecdsa_sk.pub (authenticator-hosted  ECDSA), ~/.ssh/id_ed25519.pub (Ed25519), ~/.ssh/id_ed25519_sk.pub (authenticator-hosted Ed25519), or ~/.ssh/id_rsa.pub (RSA) in the users home directory.  The user should then copy the public key to ~/.ssh/authorized_keys in their home directory on the remote machine. The authorized_keys file corresponds to the conventional ~/.rhosts file, and has one key per line, though the lines can be very long.  After this, the user can log in without giving the password.
```

```
     ...
     ...
     ...
```

We logged in into basin (weathertop here):

```
[majd@majd 10]$ ssh majdh@weathertop.cs.middlebury.edu
S
Kernel 3.10.0-1160.36.2.el7.x86_64 on an x86_64

(majdh@weathertop.cs.middlebury.edu) Password:
Last login: Wed Feb  2 13:21:33 2022 from 140.233.163.245
You have logged on to the server weathertop.middlebury.edu

This device is the property of Middlebury College.
[majdh@weathertop ~]$ ls
cs431  public_html  ZhjdnRXX  ZhkdnRXX
```

Then created ssh/authorized_keys:

```
[majdh@weathertop ~]$ mkdir .ssh
mkdir: cannot create directory    . s s h  : File exists
[majdh@weathertop ~]$ ls -a
.       .bash_history  .bashrc  .k5login  public_html  ZhjdnRXX
..      .bash_logout   cs431    .mozilla  .ssh         ZhkdnRXX
.atom   .bash_profile  .emacs   .pki      .xemacs      .zshrc
[majdh@weathertop ~]$ cd .ssh
[majdh@weathertop .ssh]$ ls
[majdh@weathertop .ssh]$ touch authorized_keys
[majdh@weathertop .ssh]$ ls
authorized_keys
```

We then created the key pairs on our computer:

```
[majd@majd 10]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/majd/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/majd/.ssh/id_rsa
Your public key has been saved in /home/majd/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:2gghFGm+5gZNRoW9F4C27Eh/KYGGBnJVTodsDhVAW2Y majd@majd
The key's randomart image is:
+---[RSA 3072]----+
|  oX**Eo.        |
|o.O oB=.         |
|+Bo.o=..         |
|.=*o..o          |
|+*..o.. S        |
|o =o + +         |
| +  o o .        |
```

```
|  o              |
| .               |
+----[SHA256]-----+
```

We copied the content of "/home/majd/.ssh/id_rsa.pub" to "majdh@weathertop/.ssh/authorized_keys" and changed the permissions on that file r and w only by the user majd:

```
[majdh@weathertop .ssh]$ cat authorized_keys
ssh-rsa ............= majd@majd

[majdh@weathertop .ssh]$ chmod 600 authorized_keys
[majdh@weathertop .ssh]$ exit
logout
Connection to weathertop.cs.middlebury.edu closed.
[majd@majd 10]$ ssh majdh@weathertop.cs.middlebury.edu
S
Kernel 3.10.0-1160.36.2.el7.x86_64 on an x86_64

Last login: Wed Feb  2 13:32:16 2022 from 140.233.163.245
You have logged on to the server weathertop.middlebury.edu

This device is the property of Middlebury College.
```

Success!

2 Browser certificates

- Figure out how to find the list of certificate authorities (CAs) that are magically trusted by your browser. Be amazed and afraid at how long the list is. Examine some of the certificates; see what sorts of data they contain.

  Visit a few sites available using HTTPS, which adds encryption and certificate verification on top of HTTP. Find the certificate for each, poke around its contents, see the chain of certificate authorities that lead to the root.

SSH and Certificates