SSH SECURITY

If you've never used SSH before on a computer, the chances are very high that when you tried to play along with the previous section you encountered a strange notification that may have looked like an error, asking you to enter **yes** or **no**. It probably looked something like:

> *The authenticity of host '192.168.10.20 (192.168.10.20)' can't be established.*
>
> *RSA key fingerprint is 29:b0:59:4f:ef:2e:6d:ee:81:97:40:04:aa:03:f7:66.*
>
> *Are you sure you want to continue connecting (yes/no)?*

Firstly, it's safe to hit **yes** if you are connecting to a server for the first time.

It's still important that we understand what the message meant, and why it's OK to say yes on your first connection to a server.

When SSHing to a remote computer, your computer tries its best to authenticate the remote computer in order to protect you from man-in-the-middle attacks.

Web servers solve this problem using Certificates signed by Certificate Authorities and validated by trust anchors installed in our computers. If SSH had been designed the same way, we would need to apply for a certificate for each computer we wanted to SSH to. This would create a major barrier to the adoption of SSH, so thankfully the SSH protocol solves the man-in-the-middle problem in a very different way.

The solution SSH has chosen works without the need for any central authorities like the CAs that underpin security on the web, but the price we pay for that convenience is that we have to deal with prompts like the one above. Because there are no central authorities to rely on, the end user has to take responsibility for their own security.

When the SSH service is installed on a computer, a random asymmetric key-pair is generated. One half of that pair is designated the server's private key, and the other the server's public key.

The first time a client connects to a server via SSH, the client saves the server's public key in a special file, along with the server's IP address and DNS name (if the client connected by DNS name rather than IP).

When the client re-connects with a server on an IP address or at a DNS name it has saved details for, it uses the saved public key to validate the server. A man-in-the-middle will not have the server's private key, and so will not be able to pass the client's security check.

Once you understand this process, the message you get when you first connect to a server makes more sense. You are being asked if you want to trust a server for which there is no saved public key, and hence who's identity cannot be confirmed. When you say yes a second message will pop up, telling you the key has been saved, it will look something like:

Warning: Permanently added '192.168.10.20' (RSA) to the list of known hosts.

On future connections to the server you should not see any more messages, because they key will be saved, and the server should pass validation by the client. The database of public keys is stored in a plain text file, **~/.ssh/known_hosts**, one entry per line. You can view the content of this file with the command:

```
cat ~/.ssh/known_hosts
```

If for some reason the server validation fails, you’ll see an error message something like:

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
29:b0:59:4f:ef:2e:6d:ee:81:97:40:04:aa:03:f7:66.
Please contact your system administrator.
Add correct host key in /Users/bart/.ssh/known_hosts to get rid of this message.
Offending RSA key in /Users/bart/.ssh/known_hosts:14
```

> RSA host key for 192.168.10.20 has changed and you have requested strict checking.
>
> Host key verification failed.

This **could** mean there is a man-in-the-middle attack in progress. But before you assume the worst, remember that there are legitimate reasons a server's public and private keys could change.

Firstly, if you re-install the OS on a computer, a new set of SSH keys will be generated, so the server will legitimately change identity.

Secondly, if you regularly connect to multiple servers on a network that has dynamically assigned IPs, then sooner or later you'll get this error because you once saw one computer at this IP, and now a different one has randomly been assigned it. It's largely to avoid problems like this that I like to set static DHCP leases for all my computers on my home network.

Once you have satisfied yourself that the warning message is innocent, the solution is to edit **~/.ssh/known_hosts** with your favourite text editor and remove the line containing the old key. Conveniently, the line number is given in the error message, it's the number after the **:**, so in the example above, the offending key is on line 14, so that's the line I need to delete.

Update: an alternative to manually editing the file is to use the **ssh-keygen**command to delete the offending key for you. You do this using the **-R** flag (R for remove) to pass the IP or hostname who's key you need to remove:

```
ssh-keygen -R computer_name_or_ip
```

Source: https://www.bartbusschots.ie/s/2015/02/14/taming-the-terminal-part-29-of-n-intro-to-ssh/