

# CISC 5650

## Cyber Security Essentials Lab

Robert Kudyba, System Administrator

Department of Computer and Information Science

Fordham University

Spring 2020

<https://goo.gl/UB1Wra> has some commands to copy/paste for this lab, and <https://www.osboxes.org/guide/> has a tutorial to get Ubuntu installed quickly with VirtualBox

# Outline

- Remote Access: Telnet & SSH
- History of Telnet & SSH
- SSH Encryption
- Security Through Obscurity
- Fail2ban
- Certificate Authority in OpenSSH
- TCP Wrapper with `/etc/hosts.deny` & `badips.com`
- Port Knocking

# Telnet



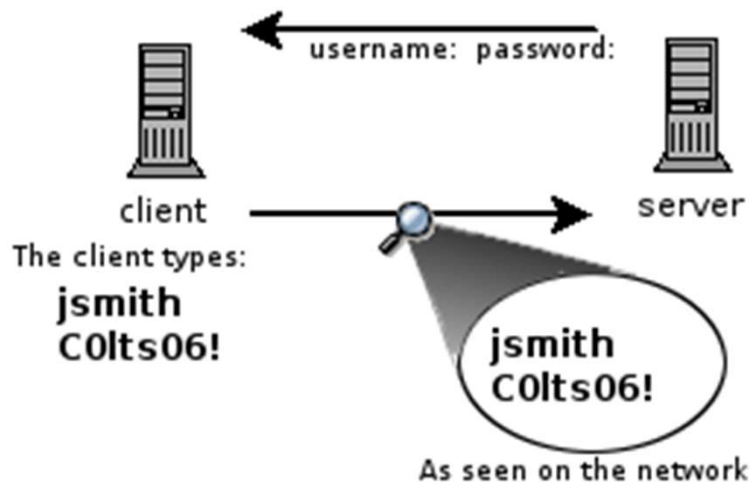
Teletype  
Network

# History of Secure Shell (SSH)

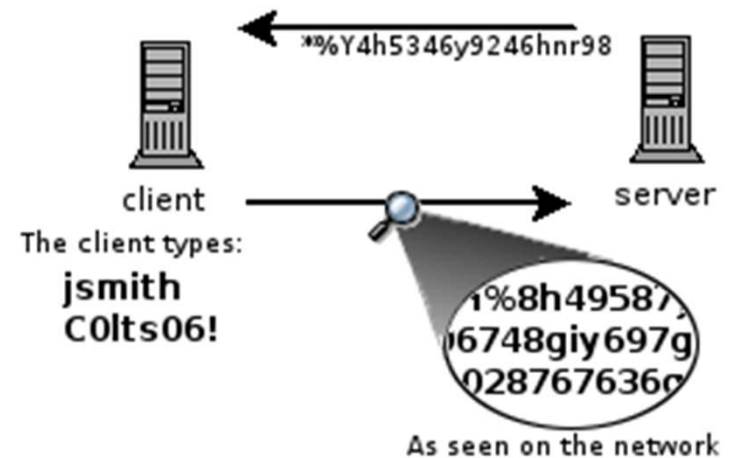


# Encrypted vs. Unencrypted

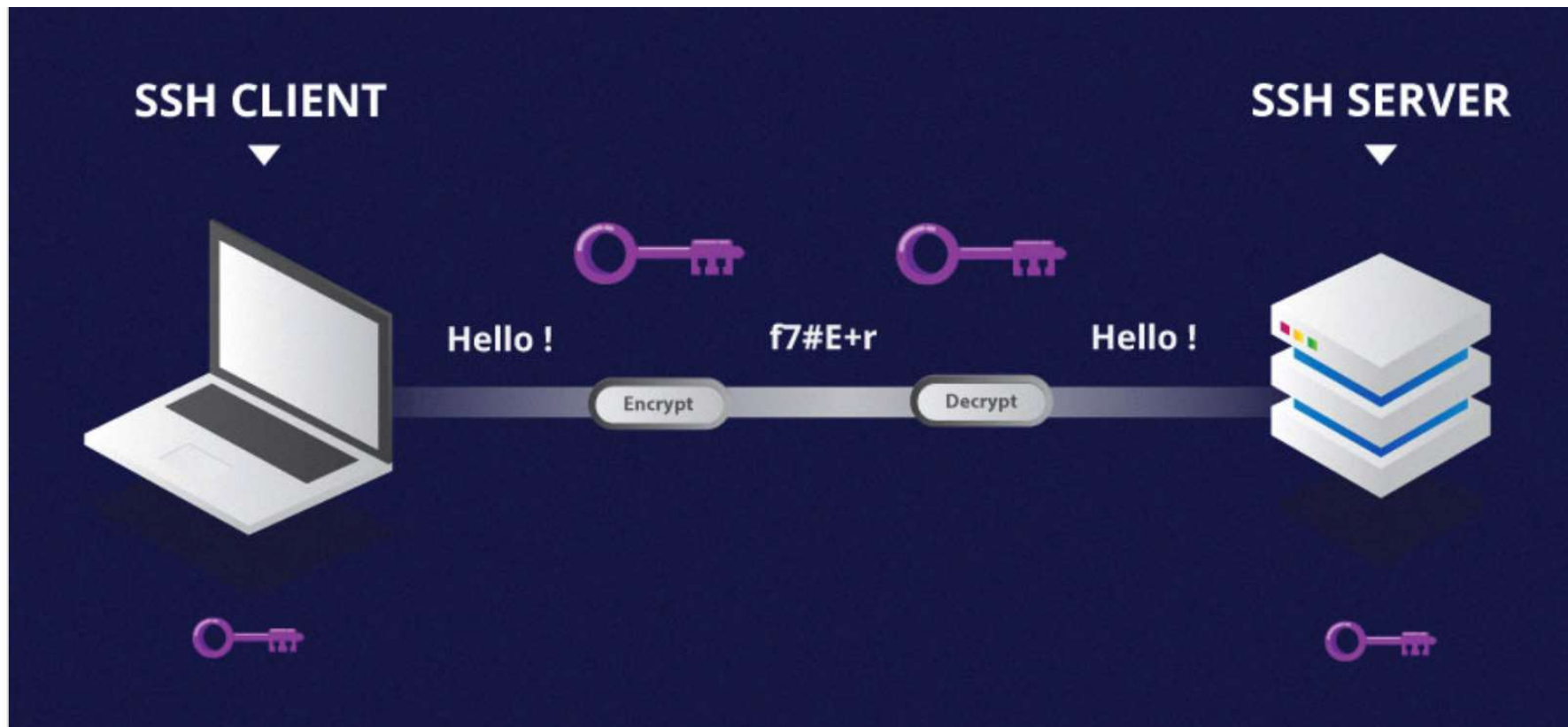
**An unencrypted login session  
such as through telnet**



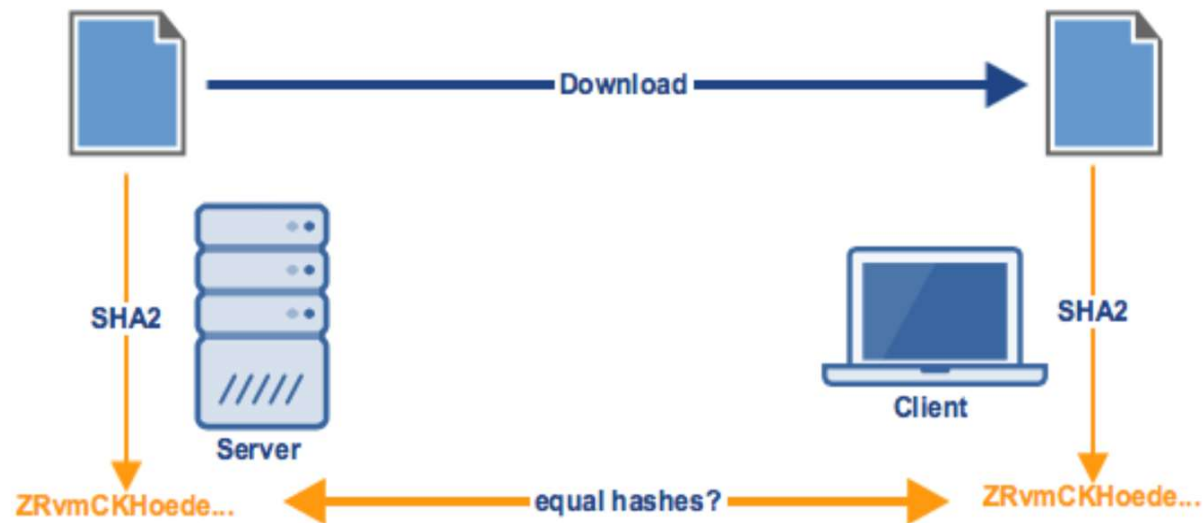
**An encrypted login session  
such as through SSH**



# SSH with Symmetric Encryption



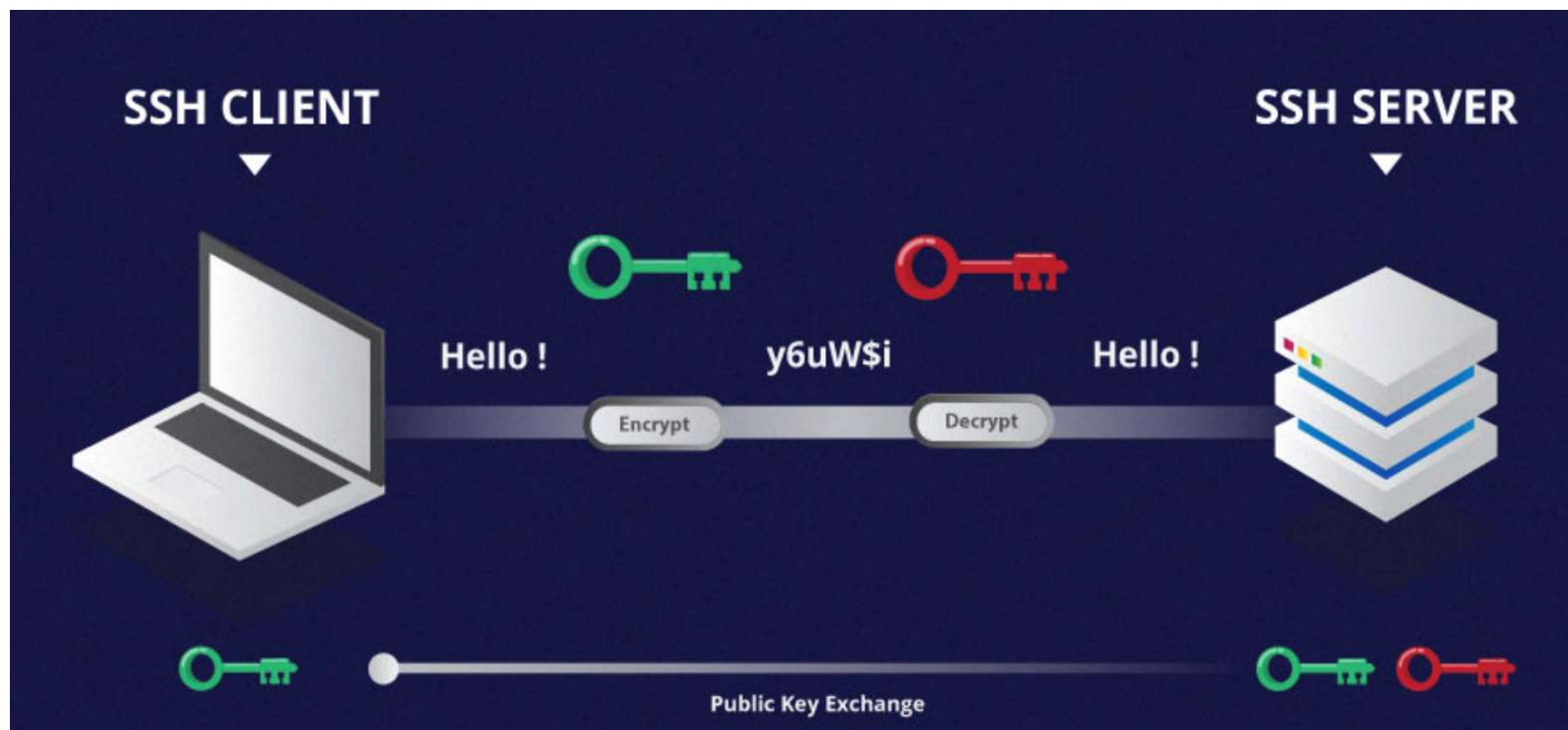
# SSH with HMAC



TCP header	
Seq. & Ack. Numbers	Source port = 3162 TCP->3162
	Destination port = 80 TCP-> HTTP
	Sequence number = 1293906975
	Aknowledgement number = 0
	Header length = 7 (28 bytes)

```
31718], length 0
10:26:40.511970 IP 10.1.1.1.33266 > 10.1.1.2.44444: Flags [P.], se
q 1309556665:1309556677, ack 3626860252, win 107, options [nop,nop
,TS val 4083407334 ecr 2541031718], length 12
10:26:40.530415 IP 10.1.1.2.44444 > 10.1.1.1.33266: Flags [.], ack
1309556677, win 106, options [nop,nop,TS val 2541229169 ecr 40834
07334], length 0
```

# SSH with Asymmetric Encryption





# Trust on first use (TOFU)/trust upon first use (TUFU)

```
[kudyba@dsm ~]$ ssh 150.108.68.128  
The authenticity of host '150.108.68.128 (150.108.68.128)' can't be established.  
ECDSA key fingerprint is SHA256:yVm8V20DZo0nAuvr9k2ydTJv0Rt0gkl8Sp5Mkmp/F0M.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? █
```



# Installing VirtualBox and Ubuntu

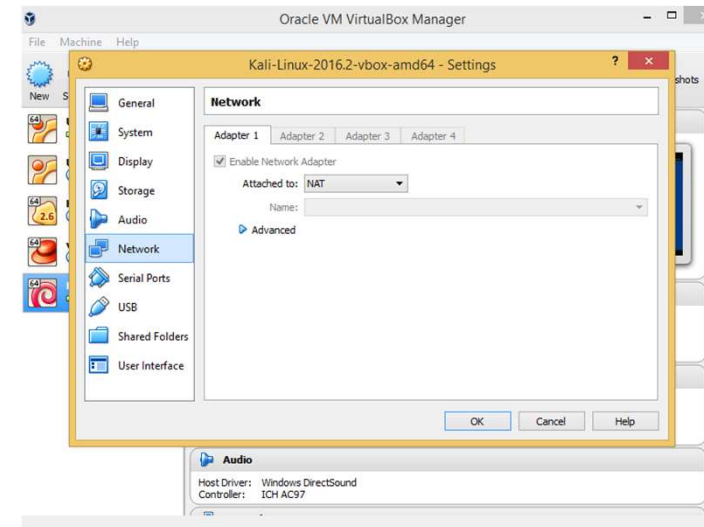
Follow the instructions at <https://www.osboxes.org/guide> to install Ubuntu on VirtualBox.

As of this writing 19.10 is the most stable release.

I have some troubleshooting tips on my home page: <https://storm.cis.fordham.edu/~rkudyba>

# Install/Enable SSH & rsyslog

- Prerequisites:
  - Make sure your Virtual Box Network setting is set to NAT on Adapter 1 and that you have Internet access in your VM. Run `sudo apt-get update`
  - Confirm SSH is running with these commands (in Courier font):
    - ❖ `systemctl status ssh`
    - ❖ If not running start with `systemctl start ssh` and then run `systemctl enable ssh` to make it persistent on reboot.
    - ❖ If not installed/not found, as with a default Ubuntu installation, run: `sudo apt-get install ssh`
  - The System Logging Service should be installed and running:
    - ❖ `systemctl status rsyslog`
    - ❖ If not running start with `systemctl start rsyslog`
    - ❖ If not installed/not found, install with `sudo apt-get install rsyslog`, then check status and start, if applicable.



# SSH Login Using a Password and Host Key Checking

- Open another Terminal session and login by running: `ssh <yourusername>@<ipaddress-of-your-vm>`. **Tip:** you can find your IP address with the `ip address` command and it's usually `10.0.2.15`. If you downloaded a VM from OSboxes, the default username is `osboxes` and password is `osboxes.org`, so the command would look like this:  
`ssh osboxes@10.0.2.15`
- TOFU! Before hitting enter type `no` and press return. Let's run `ls -l ~/.ssh`. What's the response? Tip: the up arrow returns your last command.
- If your answer is yes, the SSH client continues login, and stores the host key locally in the user's home directory, i.e., `~/.ssh/known_hosts`. You only need to validate the host key the first time you log in; in subsequent logins, you will not be prompted to confirm it again. Here's what the TOFU will look like:

```
sysadmin@sysadmin-VirtualBox:~$ ssh 10.0.2.15
The authenticity of host '10.0.2.15 (10.0.2.15)' can't be established.
ECDSA key fingerprint is SHA256:VXcb7VhBHzu78VlrwMLtor/0mElKS0IzgIRoYXPIvy4.
Are you sure you want to continue connecting (yes/no)?
```

# SSH Login Using a Password and Host Key Checking

Take note of the TOFU message. \*\*\*ECDSA is Elliptic Curve Digital Signature Algorithm, also used in Bitcoin. Type `who` (press return). What do you see? Now type `exit` (press return).

Let's compare the fingerprint/hash function result of the key. The first `ssh-keygen` command uses the key that was generated when you installed OpenSSH. The second command uses the key dropped into your `known_hosts` file. You can view that file with a command like `cat ~/.ssh/known_hosts`

```
ssh-keygen -lf /etc/ssh/ssh_host_ecdsa_key.pub  
SHA256:ogWgMnHsUaLco3AYLGwErMqtXZE+hUrbcw4nT80Jzsc
```

```
ssh-keygen -lf ~/.ssh/known_hosts  
SHA256:ogWgMnHsUaLco3AYLGwErMqtXZE+hUrbcw4nT80Jzsc|1|NvgK1BIy5Gl74hiOQ  
dDSPhBcAe4=|JVvte8xq9rncqtrN1Ttso6NdNxw= (ECDSA)
```

If the results were different what attack might be occurring?

Man-In-The-Middle! Why? This also can happen when a server is re-provisioned, very common in a virtualized environment.

Did you notice the `SHA256`? That means we are seeing Base64 encoded Secure Hash Algorithm (SHA).

*\*\*\*Note on older Apple Mac's ECDSA is not available. You can use `ssh_host_rsa.pub`*

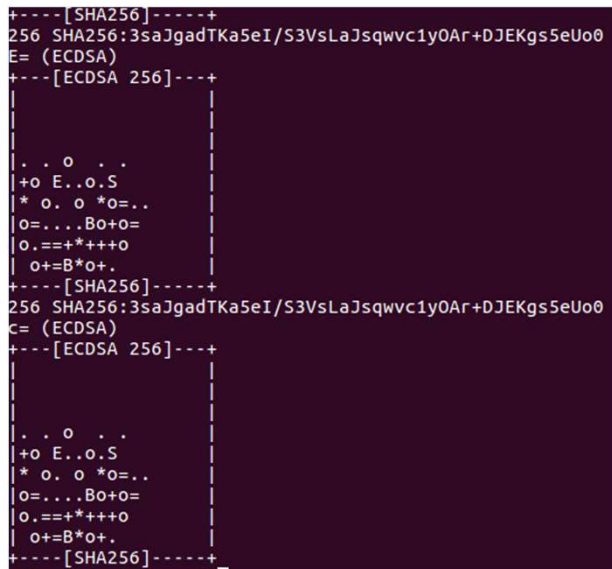
“Visual host keys”: a way of presenting the SSH client user a 2d ASCII art visualization of the host key fingerprint.

```
ssh -o FingerPrintHash=sha256 -o VisualHostKey=yes 10.0.2.15
```

The authenticity of host '10.0.2.15 (10.0.2.15)' can't be established.  
ECDSA key fingerprint is SHA256:ogWgMnHsUaLco3AYLGwErMqtXZE+hUrbcw4nT80Jzsc.  
ctrl-c to quit logging in.

```
ssh-keygen -lv -E sha256 -f ~/.ssh/known_hosts
```

```
256 SHA256:ogWgMnHsUaLco3AYLGwErMqtXZE+hUrbcw4nT80Jzsc  
|1|C50cej0mMqB1MWi2fvATu4HMjFQ=|wV9QCm8jh5DQQhr8jBXiIGG7zzQ= (ECDSA)
```



```
+----[SHA256]-----+  
256 SHA256:3saJgadTKa5eI/S3VsLaJsqwvc1y0Ar+DJEKgs5eUo0  
E= (ECDSA)  
+---[ECDSA 256]---+  
|  
| . . O . .  
| +O E..O.S  
| * O. O *O=.  
| O=...Bo+O=  
| O.=+*++O  
| O+=B*O+.  
|  
+----[SHA256]-----+  
256 SHA256:3saJgadTKa5eI/S3VsLaJsqwvc1y0Ar+DJEKgs5eUo0  
C= (ECDSA)  
+---[ECDSA 256]---+  
|  
| . . O . .  
| +O E..O.S  
| * O. O *O=.  
| O=...Bo+O=  
| O.=+*++O  
| O+=B*O+.  
|  
+----[SHA256]-----+
```

Run both of the above commands and compare the results. Do the pictures look the same? See what happens when you try the first command with a real world server such as `erdos.dsm.fordham.edu`. Note with more host keys in the `known_hosts` file you will have more visual host keys to compare.

# Create Asymmetric SSH Keys

- Open Terminal and type (*note older Apple Mac's do not have ed25519 as an option, just remove -t ed25519*):  
`ssh-keygen -t ed25519 -b 16384`
- Once you have entered the above command, you will get a few more questions:  
Enter file in which to save the key (/home/osboxes/.ssh/id\_rsa):
- You can press enter here, saving the file to the user home (in this case, my example user is called `myuser`). The you will see:  
Enter passphrase (empty for no passphrase):
- It's up to you whether you want to use a passphrase. Entering a passphrase does have its benefits: the security of a key, no matter how encrypted, still depends on the fact that it is not visible to anyone else. Should a passphrase-protected private key fall into an unauthorized users possession, they will be unable to log in to its associated accounts until they figure out the passphrase, buying the hacked user some extra time. The only downside, to having a passphrase, is then having to type it in each time you use the Key Pair. Notice the randomart VisualHostKey image. Copy/paste it some where or screenshot it (hint: it's a lab question).
- Then run (using your username and IP address):  
`ssh-copy-id osboxes@10.0.2.15`  
The authenticity of host '10.0.2.15 (10.0.2.15)' can't be established.  
ECDSA key fingerprint is SHA256:ogWgMnHsUaLco3AYLGwErMqtXZE+hUrbcw4nT80Jzsc.  
Are you sure you want to continue connecting (yes/no)? Yes  
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed  
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys  
osboxes@10.0.2.15's password:  
Number of key(s) added: 1  
Now try logging into the machine, with: "ssh 'osboxes@10.0.2.15'"  
and check to make sure that only the key(s) you wanted were added.
- If you didn't download Ubuntu from osboxes you may have to run `ssh-add` (to work-around an issue on Ubuntu with the GNOME-KEYRING password manager)
- Now you can go ahead and log in with `ssh osboxes@10.0.2.15` and you will not be prompted for a password. However, if you set a passphrase, you will be asked to enter the passphrase at that time (and whenever else you log in in the future).
- Note that if SSH on the server is still configured to accept password authentication we really haven't done any system hardening.

# Disable Password Authentication in SSH

- Open Terminal and type:  
`sudo nano /etc/ssh/sshd_config`
- Look for the line:  
`#PasswordAuthentication Yes`  
change it to, and remove the # sign (it's a comment):  
`PasswordAuthentication No`
- Press `ctl-o`, press enter then `ctl-x` to save and close the file
- Restart ssh with this command: `sudo systemctl restart ssh`
- Try logging in with another username:  
`ssh root@10.0.2.15`
- What happens?
- Another form of system hardening is to disable root logins, but Ubuntu disables that by default. On a system like Kali you would set `PermitRootLogin` to `No` in `/etc/ssh/sshd_config` and then restart ssh.
- To see which ciphers and Message Authentication Codes are available:  

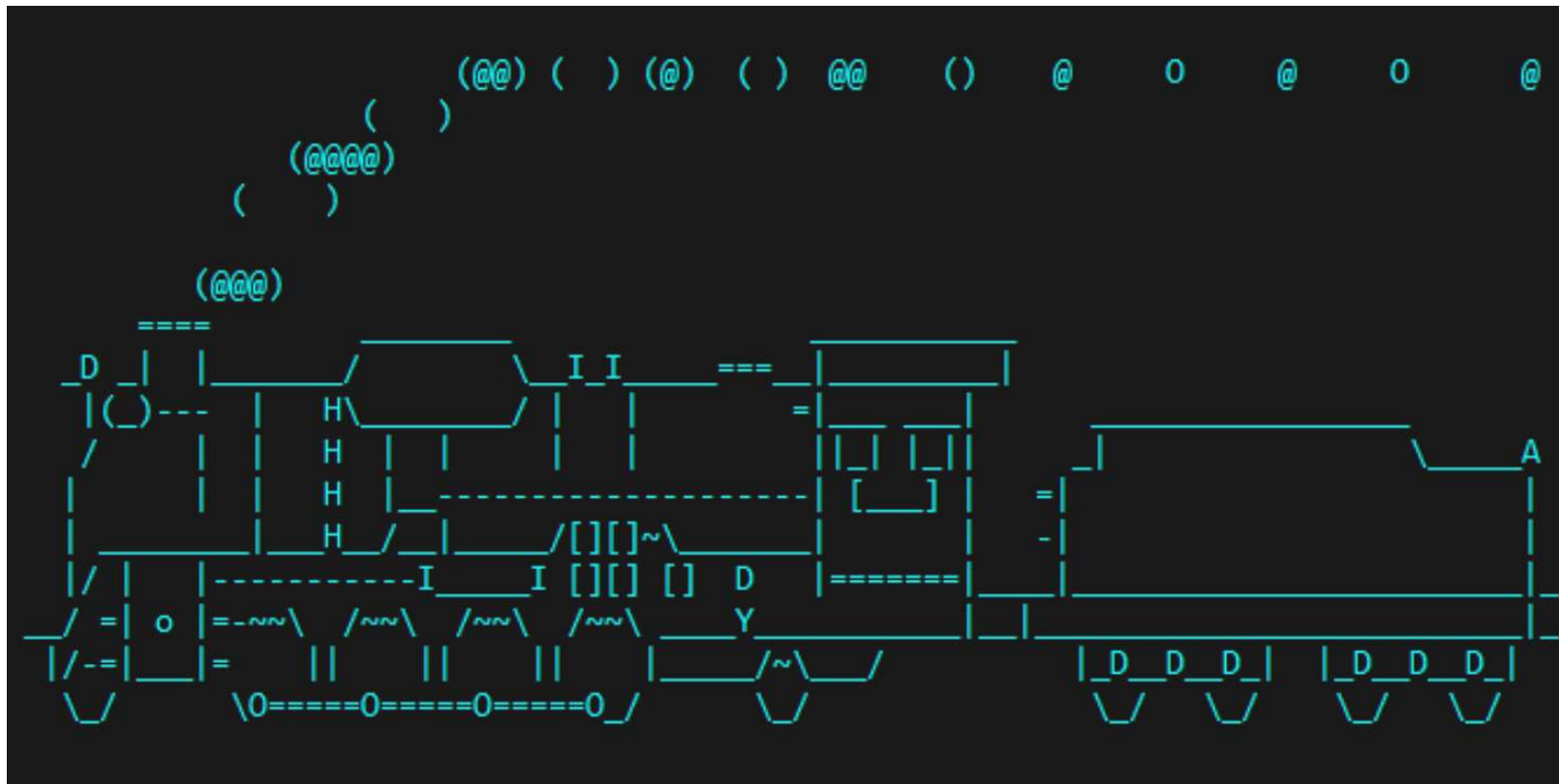
<code>ssh -Q cipher</code>	<code># List supported ciphers</code>
<code>ssh -Q mac</code>	<code># List supported MACs</code>
<code>ssh -Q key</code>	<code># List supported public key types</code>
<code>ssh -Q kex</code>	<code># List supported key exchange algorithms</code>



# Security Through Obscurity



## Brute Force



# Security By Obscurity, Adding a Layer: Fail2ban



## ***FAIL2BAN***

- Log snippet from the monitored log file: `/var/log/auth.log`:
- `osboxes sshd[xx]: Failed password for root from 10.0.2.2 port NNNNN ssh2`
- Action taken by Fail2ban, noted in `/var/log/fail2ban.log`
- `fail2ban.actions[xx]: NOTICE [sshd] Ban 10.0.2.2`

# Fail2ban continued: jails

`/^[Reg]ular[Ex]pression$/`

**BANNED**

# Install & Configure Fail2ban

- I created a web page so you can copy/paste the commands for this slide: <https://goo.gl/UB1Wra>. Might help to open a browser in your VM.
- Run `sudo apt-get install fail2ban`
- Use a text editor like `nano` and create a new file with the following contents:
- `sudo nano /etc/fail2ban/jail.local`
- Copy and paste the below into the new file:
  - `[DEFAULT]`
  - `[ssh]`
  - `enabled = true`
  - `port = ssh`
  - `filter = sshd`
  - `logpath = /var/log/auth.log`
  - `maxretry = 3`
- Press `ctl-o`, press `enter` then `ctl-x` to save and close the file.
- Run `sudo systemctl restart fail2ban`
- Check the logs, if any errors, e.g., typos from above: `tail -n30 -f /var/log/fail2ban.log` (`ctl-c` to quit the tail)

# IP vs Port



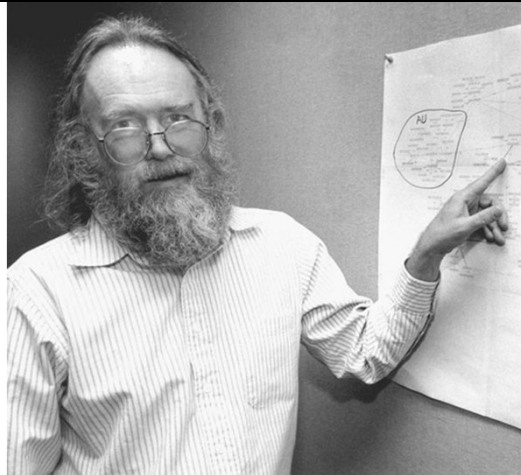
Internet Assigned Numbers Authority



- Run `cat /proc/sys/net/ipv4/ip_local_port_range`

# Loopback or 127.0.0.1

There is no place like  
127.0.0.1\_



# SSH from desktop to VirtualBox

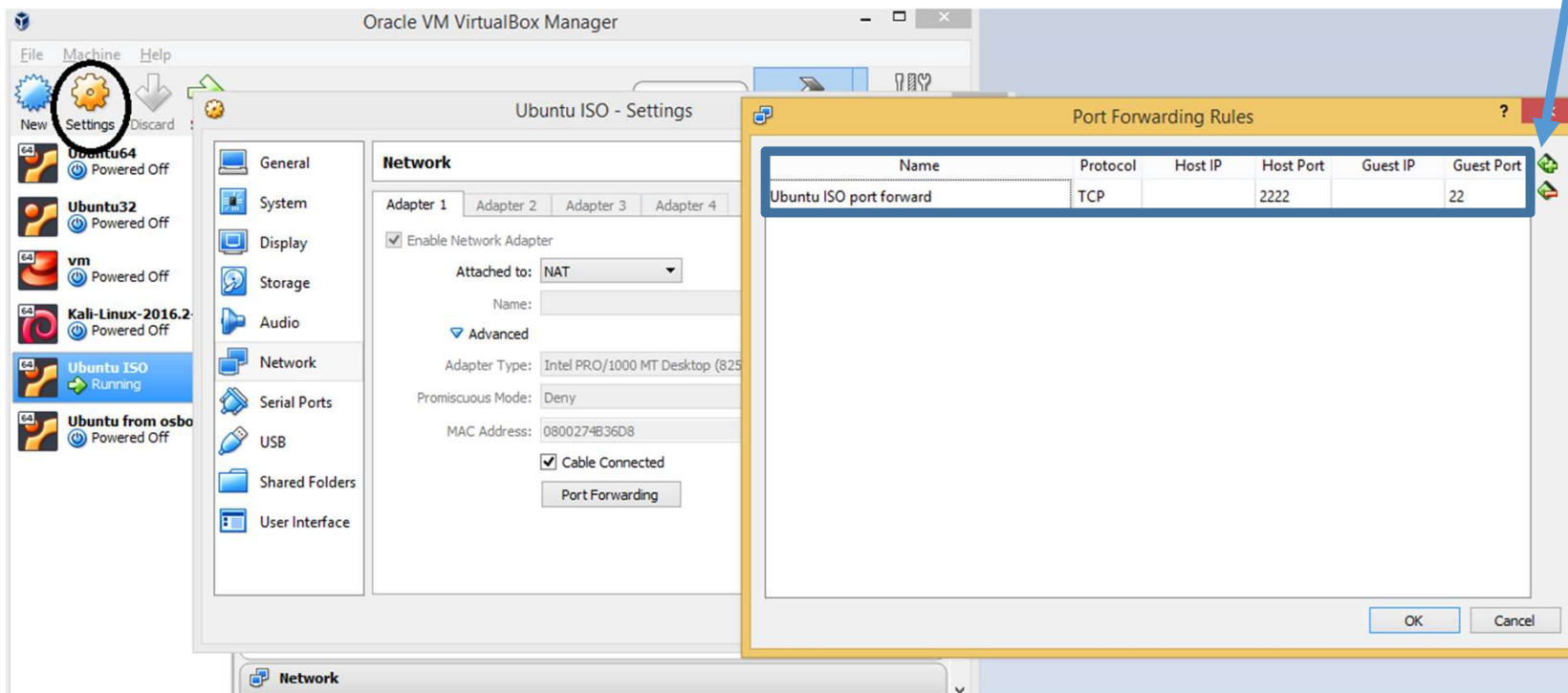
- In Ubuntu within VirtualBox, find the IP address with the command `ip addr`. It usually defaults to 10.0.2.15.
- On a Mac desktop, open Terminal and run `ifconfig`, look for the IP near `inet`.
- On Windows, click the Windows key, press the letter r, type `cmd`, press OK and run `ipconfig`.
- How can we go from an IP address in a different “address space”?
- Port forwarding! "If a TCP connection is received on the Host on TCP port 2222, send it on to the Guest on TCP port 22."



# VirtualBox port forwarding Windows

In Windows VirtualBox,  
→ Settings, select Network  
→ Advanced  
→ Port Forwarding

Press the green plus sign,  
enter 2222 in Host Port and 22 in Guest Port.  
Click OK twice

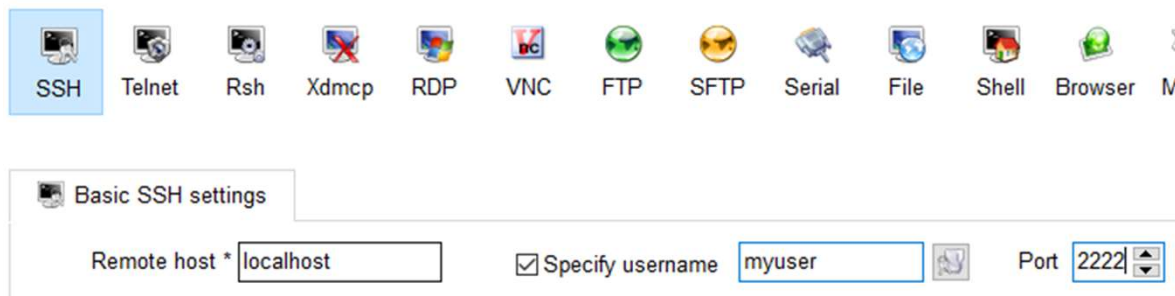


# VirtualBox port forwarding Windows

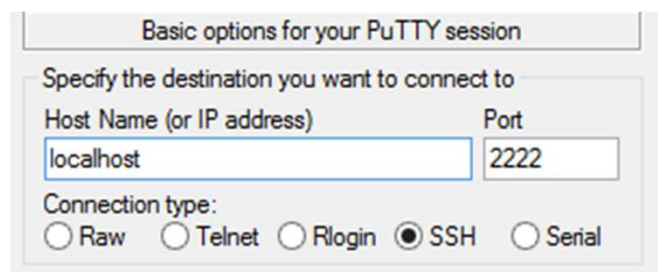
## Putty or MobaXterm

- Enter `localhost` as the IP address and `2222` as the port.

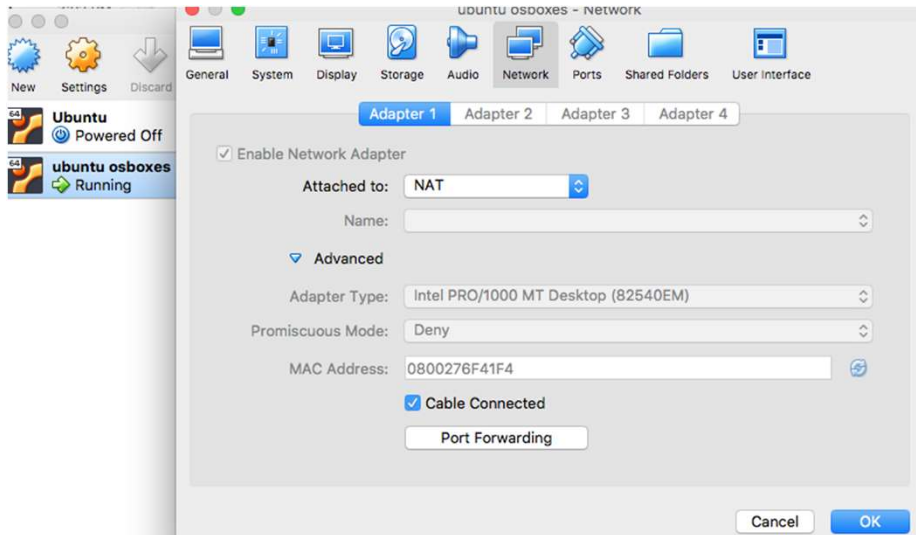
- MobaXterm:



- Putty:



# VirtualBox port forwarding from a Mac



Click Settings  
→ Network  
→ Advanced  
→ then Port Forwarding.

Enter 2222 in Host Port and 22 in Guest Port.  
Open Terminal, type  
`ssh -p 2222 osboxes@localhost`

Name	Protocol	Host IP	Host Port	Guest IP	Guest Port
Rule 1	TCP		2222		22

# Fail2ban testing

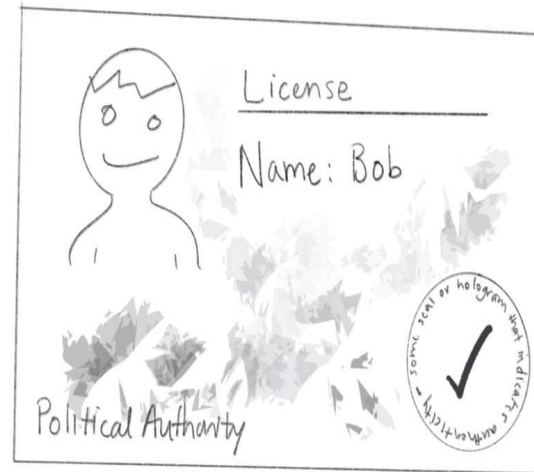
- In Slide 16 we disabled Password Authentication in SSH, let's enable that again:  
`sudo nano /etc/ssh/sshd_config`, find `PasswordAuthentication No` **replace the** No with Yes.  
**Restart ssh:** `sudo systemctl restart ssh`
- Let's follow the Fail2ban log file in real-time, run this command in your VM: `sudo tail -n10 -f /var/log/fail2ban.log`
- From your actual desktop (not the VM), try logging in using SSH.
- From a Mac, using Terminal run `ssh -p 2222 root@localhost` and press return 3 times. Repeat.
- From Windows, using Putty or Mobaxterm, enter the hostname as `localhost` and Port 2222. Enter a blank password. Repeat. `ctrl-c` will stop the real-time follow.
- What happens? What does the log file show?

# SSH with a Certificate Authority

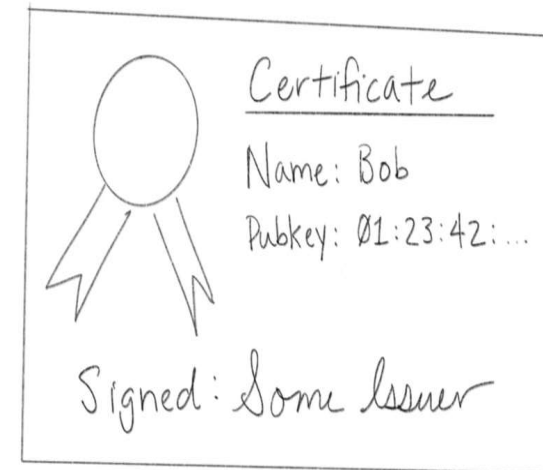
This is a cert.  
It's not that fancy...



Interpretation: "Some Issuer (issuer) says Bob's (subject) public key is 01:23:42:..."



Issued by DMV (Political Authority)  
Verified by Checking holograms & stuff  
Trusted b/c Trust DMV (lol)  
Used to Authenticate person / figure out name using picture



Certificate Authority  
Checking signature & stuff  
Trust CA  
Authenticate entity / figure out name using public key

# WSL for Windows and Installing Ubuntu

For Windows 10, follow [this tutorial](#) to install Ubuntu on Windows. This uses a feature called “Windows Subsystem for Linux”.

URL is <https://tutorials.ubuntu.com/tutorial/tutorial-ubuntu-on-windows>

1. Use the Start menu to launch the Microsoft Store application.
2. Search for Ubuntu and select the first result, 'Ubuntu', published by Canonical Group Limited.
3. Click on the Install button.
4. When launched for the first time, Ubuntu will inform you that it's 'Installing' and you'll need to wait a few moments.
5. When complete, you'll be asked for a username and password specific to your Ubuntu installation. These don't need to be the same as your Windows 10 credentials. With this step complete, you'll find yourself at the Ubuntu bash command line.

# Configuring SSH with a Certificate Authority

First, we will create our own CA, which is essentially just a normal key pair. Think of this step as similar to the license stamper machine at the DMV. In Ubuntu Terminal:

```
mkdir ~/my-ca && cd ~/my-ca  
ssh-keygen -C CA -f ca
```

Leave passphrase empty and press return 2x. Two files are created, `ca` (the private key) and `ca.pub` (the public key). Let's place `ca.pub` in `/etc/ssh/ca.pub`.

```
sudo cp ca.pub /etc/ssh/
```

Now configure SSH to trust it by adding this single line change:

```
sudo nano /etc/ssh/sshd_config
```

Add this line at the bottom of the file:

```
TrustedUserCAKeys /etc/ssh/ca.pub
```

We also have to re-enable SSH password authentication so in

```
/etc/ssh/sshd_config change: PasswordAuthentication No to  
PasswordAuthentication Yes
```

Press `ctl-o` to save changes, press enter, and then press `ctl-x` to exit.

Restart ssh: 

```
sudo systemctl restart ssh
```

Let's stop Fail2ban: 

```
systemctl stop fail2ban
```

# Configuring SSH with a Certificate Authority 2

\*In Ubuntu on Windows, or Terminal on Mac, we generate a key (that the “DMV” will need to sign):

```
ssh-keygen -t ecdsa
```

Change into the `.ssh` directory and list the files with these 2 commands:

```
cd .ssh
```

```
ls -lt
```

You should see `id_ecdsa` and `id_ecdsa.pub` which are your private key and public key, respectively. We will secure FTP the public key to Ubuntu. From your desktop “client”, i.e., Terminal in a Mac or Ubuntu on Windows run these commands:

```
sftp -P 2222 osboxes@localhost
```

```
ls -l
```

```
pwd
```

```
cd my-ca
```

```
put id_ecdsa.pub
```

```
ls -l
```

```
exit
```

*\*Note on older Apple Mac's ECDSA is not available, replace `ecdsa` with `rsa`*



# Sign the new keys with the Certificate Authority

Now in the Ubuntu VM Certificate Authority we are going to sign the public key. Make sure you are in the newly created `my-ca` directory.

```
cd ~/my-ca  
ssh-keygen -s ca -I guesthost -n osboxes -V +1w -z 1  
id_ecdsa.pub
```

Let's inspect the newly created certificate with this command:

```
ssh-keygen -Lf id_ecdsa-cert.pub
```

The certificate ID will be `guesthost` and the only principal it has will be `osboxes`. Principal refers to a system user, i.e., `osboxes`. It's valid for one week and has the serial number 1.

Key ID: **"guesthost"**

Serial: 1

**Valid: from 2019-09-27T13:52:00 to 2019-10-04T13:53:24**

Principals:

**osboxes**

# Using the Signed CA Certificate SSH Key

From the desktop, download from your VM the signed certificate created via the `ssh-keygen` command.

```
cd ~/.ssh
sftp -P 2222 osboxes@localhost
ls -l
cd my-ca
ls -l
get id_ecdsa-cert.pub
ls -l
exit
```

For Mac OS X users, create a file that will prepend the port number and hostname and act as a work-around:

```
nano ~/.ssh/config
host localhost
HostName localhost
Port 2222
User osboxes
Save changes and exit the file: ctrl-o, press enter, ctrl x
```

Now try to ssh in:

```
ssh localhost
exit
```

Try with a non-existing user:

```
ssh test@localhost
```

What happens?

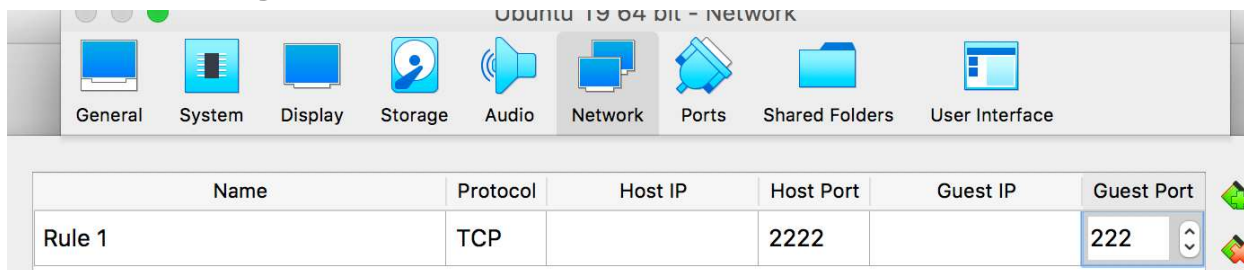
# System Hardening : Change SSH Port

- In the Ubuntu VM open `/etc/ssh/sshd_config` again with nano . Run:  
`sudo nano /etc/ssh/sshd_config`
- Look for the directive `Port 22` which may have a hashtag/pound sign in front of it like this:  
`#Port 22`. Note that the hashtag (#) indicates a comment.
- Change the value to <sup>1</sup>222 and remove the hashtag (if present) so it looks like this:  
`Port 222` `ctl-o`, press enter to save, `ctl-x` to exit.
- Restart `sshd` (Kali & Ubuntu do not require the “d” in this command, but Fedora/RedHat do):
  - `sudo systemctl restart ssh` then to confirm the status run
  - `systemctl status ssh`
- Optional step. Open a new Terminal window, you can check the system logs in real time by running the following:
  - `tail -n30 -f /var/log/syslog`
    - ❖ `-n30` option specifies the last 30 lines and the `-f` option specifies to watch (or follow) the log in real time .  
Press `ctl-c` to quit tailing.

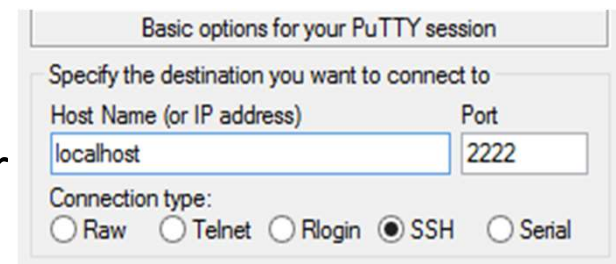
<sup>1</sup>222 is technically reserved for Berkeley `rshd` (remote shell) with SPX authentication.

# Logging in SSH with new port number

- In Ubuntu Terminal, type `ssh -p 222 osboxes@10.0.2.15`.
- OPTIONAL: Remembering the non-standard ssh port can be annoying, but if you have a standard set of workstations that you use for access your servers, just utilize a file called `~/.ssh/config` to specify certain ports for certain servers. We created this file on our desktop, now we can do the same in the VM. To create the file run `nano ~/.ssh/config` containing:  
`Host 10.0.2.15`  
`Port 222`
- Now we will try to SSH from your desktop. In VirtualBox settings, update the Port Forwarding Guest Port number to 222:



- From a Mac: `ssh -p 2222 osboxes@localhost`
- From Windows Putty or MobaXterm use localhost for the Host Name and 2222 as the Port:



# SSH logins with new port

- Restart Fail2ban: `systemctl restart fail2ban`
- If you forgot the IP address, open Ubuntu's Terminal and run the command `ip addr` and look for the IP address usually 10.0.2.15. You can also use System Settings (Gear icon) and click on Network.
- In Ubuntu Terminal watch (or follow) the fail2ban log file in real time (ctl-c to quit):  
`tail -n30 -f /var/log/fail2ban.log`
- What change in Fail2ban's config file needs to be made?
- In another Terminal window try logging in to your own workstation with `ssh -p 222 10.0.2.15` and put in a few passwords until you see the log has a "Ban" in it.
- You can whitelist your IP by adding a new directive in the `/etc/fail2ban/jail.local` file under [DEFAULT]:  
`ignoreip = 10.0.2.15`
- Any changes to the `jail.local` file requires a restart of Fail2ban.  
`sudo systemctl restart fail2ban`
- SSH from your desktop  
`ssh -p 222 osboxes@localhost`
- What's the IP of your host/desktop? It will show in `/var/log/fail2ban.log`

# Install and Enable sendmail For Email Notifications

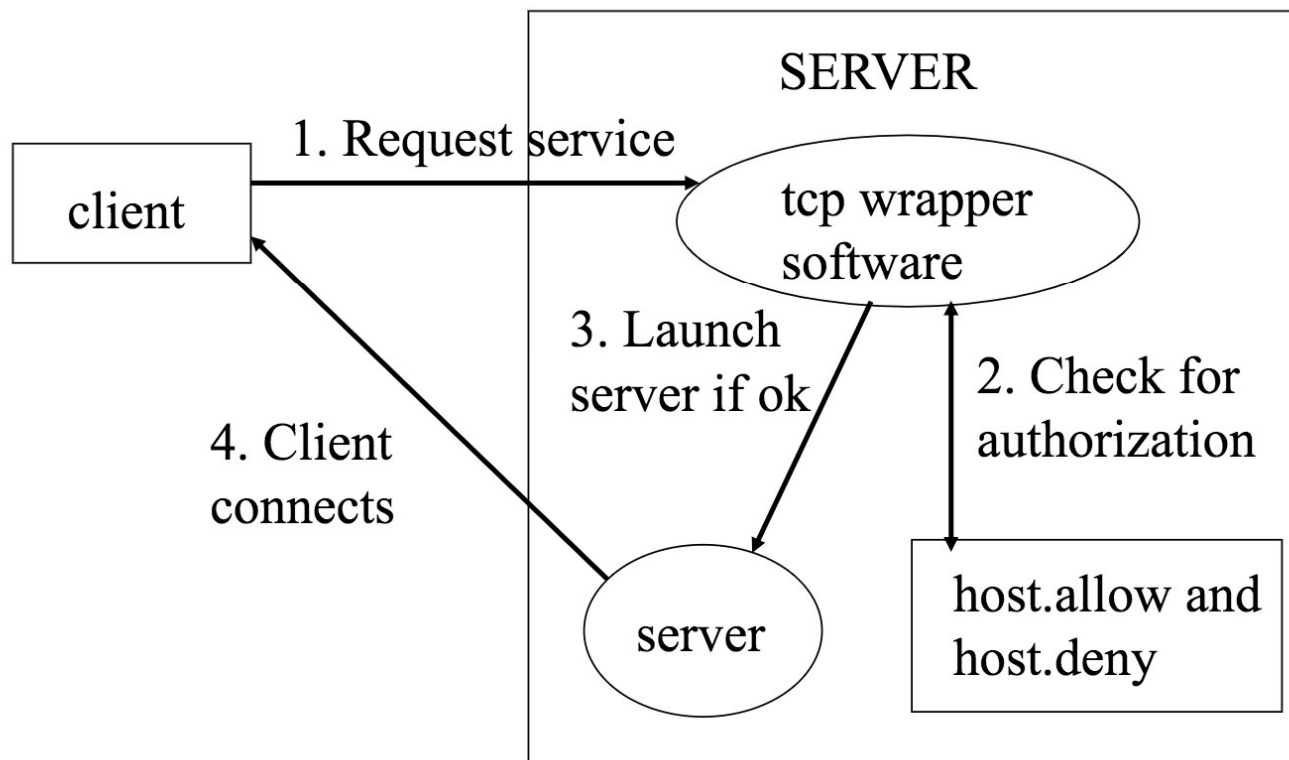
- In Terminal run (this takes a couple of minutes):  
`sudo apt-get install sendmail sendmail-bin`
- Run `sudo nano /etc/hosts` and replace the first line with the following:  
`127.0.0.1 localhost localhost.localdomain osboxes`  
Then add this new line to the end of the file:  
`10.0.2.15 localhost localhost.localdomain osboxes`
- `ctl-o`, press save, `ctl-x`
- **\*\*\*NOTE: If you are on Kali or changed the VM hostname, replace osboxes with the results of the command 'hostname'. Emails will not be sent without this change.\*\*\***
- Run:  
`sudo systemctl enable sendmail`  
`sudo systemctl start sendmail`  
Check logs:  
`tail -n20 -f /var/log/mail.log`
- Check status:  
`systemctl status sendmail`  
Note: you may see a message like unable to qualify my own domain name, this is ok. Your @fordham.edu email should accept the email, check the log file for any messages, such as "Real domain name required for sender address".

# Fail2ban continued

- The given example below will block the offending remote IP address on the 6th ban in the same day and will then ban that IP from all port of your server for a period of 1 week. I created a web page so you can copy/paste the commands for this slide: <https://goo.gl/UB1Wra>. Notice what's different with the port option.
- Run `sudo nano /etc/fail2ban/jail.local` file and replace with the following (ctrl k deletes 1 line at a time):
- ```
[DEFAULT]
destemail = YOUREMAIL@fordham.edu
sendername = Fail2Ban
mta = sendmail
banaction = iptables-multiport
action = %(action_mw)s
protocol = tcp
[ssh]
enabled = true
port = 22,222
filter = sshd
logpath = /var/log/auth.log
maxretry = 3
[recidive]
enabled = true
filter = recidive
logpath = /var/log/fail2ban.log
action = iptables-allports[name=recidive]
        sendmail-whois-lines[name=recidive, logpath=/var/log/fail2ban.log]
bantime = 604800 ; 1 week
findtime = 86400 ; 1 day
maxretry = 5
```
- Restart Fail2ban: `sudo systemctl restart fail2ban`
- Check the status, `sudo systemctl status fail2ban`, fix any errors/typos, view the `/var/log/fail2ban.log` file with the 'more' command, e.g., `more /var/log/fail2ban.log`.
- If you do this outside Fordham's network, or you are using an alternate Linux than OSBoxes, the email might not go out. Check the logs in `/var/log/mail.log`. Check your Spam mail folder.

# TCP Wrapper /etc/hosts.deny & badips.com

## How tcp wrappers works





# Add badips Script

I created a web page so you can copy/paste the commands for this slide: <https://goo.gl/UB1Wra>

In Terminal, run the following commands and note the long URL below can get cut off when copy/pasting:

```
cd ~
```

```
sudo wget https://raw.githubusercontent.com/mitchellkrogza/fail2ban-useful-scripts/master/f2b-badips-to-hostsdeny.sh
```

```
sudo chmod 755 f2b-badips-to-hostsdeny.sh
```

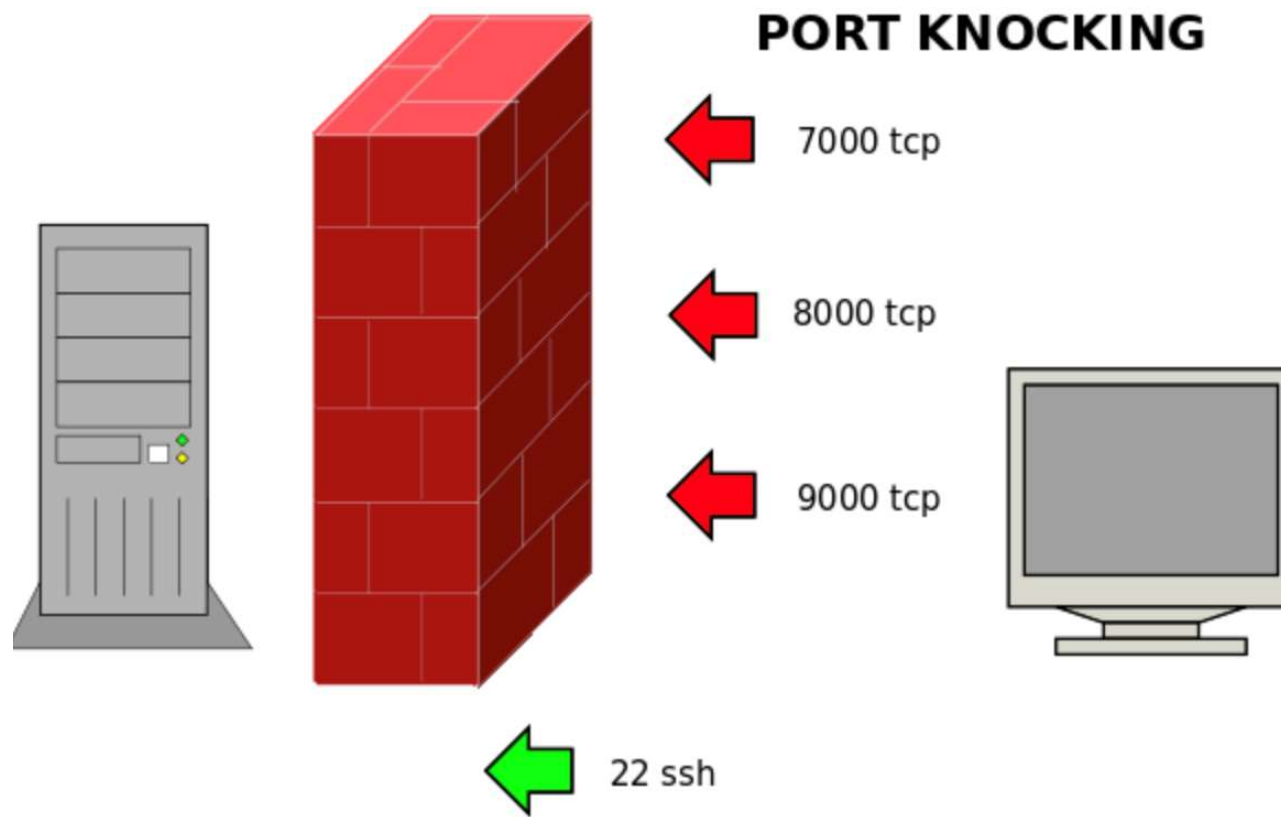
```
sudo nano f2b-badips-to-hostsdeny.sh
```

Look for `_keyservice=` and add `9f0f68f96dad4815715b22bd260eaa90bc3be9af`

Save & close (ctl-O, press enter, ctl-X in nano).

- Type the following to run the script: `sudo ./f2b-badips-to-hostsdeny.sh` (ignore the syntax error/invalid number of lines)
- View the updated file with the new IP addresses, e.g., `more /etc/hosts.deny`
- Optional, add it to cron so it updates every night, in this example at 10:55 PM, the next line assume 'vi' is used rather than 'nano':
- Type `crontab -e` (press Enter), then press "i" (for insert) and add (replacing with your home directory):
- `55 22 * * * /home/osboxes/f2b-badips-to-hostsdeny.sh`
- `:wq!` will save and exit the file.

# Port Knocking



# Recap

- Which CIA tenant(s) does symmetric key encryption guarantee?
- Confidentiality!
- Does symmetric key cryptography implement nonrepudiation?
- No! Because any communicating party can encrypt and decrypt messages with the shared secret key, there is no way to prove where a given message originated.
- Provide an example of “security by obscurity” with SSH.
- What layers/system hardening did we implement?
- Changed SSH port, disabled password authentication, installed Fail2ban, used a Certificate Authority, added “recidive” jail for Fail2ban, and installed TCP Wrapper script. Also showed you Port Knocking.
- Remember to delete the `~/.ssh/config` file!