Cheat Sheet - Cloud Security-1

Key Terms

- **Term**: IaaS or 'Infrastructure as a service'

  > **Definition**: A vendor provides clients pay-as-you-go access to storage, networking, servers and other computing resources in the cloud.

- **Term**: PaaS or 'Platform as a Service'

  > **Definition**: A service provider offers access to a cloud-based environment in which users can build and deliver applications. The provider supplies underlying infrastructure.

- **Term**: SaaS or 'Software as a Service'

  > **Definition**: A service provider delivers software and applications through the internet. Users subscribe to the software and access it via the web or vendor APIs.

- **Term**: XaaS or 'Anything as a Service'

  > **Definition**: Refers to the *all* the offerings via cloud computing as opposed to being provided locally, or on premises.

- **Term**: DBaaS or 'Database as a Service'

  > **Definition**: Refers to cloud software that enables users to setup, operate and scale databases without having to know about the exact implementations of the specific database.

- **Term**: CaaS or 'Communications as a Service'

  > **Definition**: Refers to an outsourced communications solution. Such communications can include voice over IP (VoIP or Internet telephony), instant messaging (IM), collaboration and video conference applications.

- **Term**: DaaS or 'Data as a Service'

> **Definition**: Refers to the concept that a companies' data product can be provided to the user on demand, regardless of geographic or organizational separation between provider and consumer.

- **Term**: Virtual Computing Or Virtual Machine

> **Definition**: Refers to an emulation of an entire physical computer system. Virtual machines are based on computer architectures and provide the functionality of a physical computer.

- **Term**: Container Computing

> **Definition**: A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.

- **Term**: Provisioning

> **Definition**: In general, provisioning means "providing" or making something available. Though the term is used in a variety of contexts in IT, in cloud computing, we are referring to software that automatically configures a virtual machine or container. Examples include: Ansible, Chef and Puppet.

- **Term**: CI or Continuous Integration

> **Definition**: Continuous integration is the practice of automating the integration of code changes from multiple contributors into a single software project.

- **Term**: CD or Continuos Delivery

> **Definition**: Continuous delivery is a software release approach in which development teams produce and test code in short cycles, usually with a high degree of automation.

Key Commands

SSH

Secure Shell sets up an encrypted connection between two machines. Commands given on the first machine are executed on the second machine and output from the second machine is sent back to the first machine.

The result is the ability to control a remote machine using the command line while keeping all your actions private from any would be attacker or snooper.

ssh-keygen

The `ssh-keygen` command creates a private/public key pair that you can use to authenticate your SSH connections. Once created, the public key is copied to the server into the `~/.ssh/known_hosts` file.

```
# Create an ssh public/private key pair
ssh-keygen
```

You can create a password for your SSH key if you wish, but this isn't recommended if the SSH key is going to be used for automation purposes.

ssh-copy-id

The `ssh-copy-id` command will copy the public ssh key into the correct location on the remote server.

```
# copy the public key for 'mykey' to the ~/.ssh/known_hosts file on a rem
ote machine
ssh-copy-id -i ~/.ssh/mykey user@<host ip>
```

ssh

The `ssh` command creates an ssh connection to a remote machine. If you do not have an SSH key in place, you will be prompted for a password.

```
# Connect to the machine with the IP address 10.10.0.4 using the 'admin'
user.
ssh admin@10.10.0.4
```

You can create multiple keys for different purposes and different machines. To specify a particular key, use the `-i` flag.

```
# Connect to the machine at 10.10.0.4 using a specific 'mykey' identity
ssh -i mykey.pub admin@10.10.0.4
```

Docker

Docker allows you to run Linux containers on any server or local machine. Containers are similar to a virtual machine, except they only run the resources necessary to complete their specific task.

A container typically only runs one task. This could be a web server, or a particular application or any program you choose. They are similar to an `app` on your phone. A container is completely self sufficient and has everything it needs to run.

docker pull

`docker pull` will copy a container to the server so you can run it.

```
# download the 'dvwa' container from the 'cyberxsecurity' docker reposito
ry
docker pull cyberxsecurity/dvwa
```

docker image ls

`docker image ls` lists all of the container images that are copied to the server. Each image can be used to create any number of containers.

```
docker image ls
REPOSITORY                TAG                 IMAGE ID            CREATED
            SIZE
cyberxsecurity/ansible    latest              6657e0b22542        11 days
ago         303MB
ubuntu                    18.04               775349758637        7 weeks
ago         64.2MB
```

docker container list -a

`docker container list -a` will list *all* of the containers on the system, including containers that are not running.

docker run

`docker run` will create a container from the specified image.

```
# Run the cyberxsecurity/ansible container
docker run cyberxsecurity/ansible

# Run the container using the image ID 6657e0b22542
docker run 6657e0b22542
```

docker start

`docker start container_name` will start a stopped container. This does not *create* a container from a container image. Instead, it will only start a container that you already have created.

docker stop

`docker stop` will stop a running container. This is similar to shutting down a VM.

```
# stop the ansible container
docker stop ansible
```

docker exec

`docker exec` will execute a command directly on a container and return the output of that command to you. This is commonly used to get a bash shell on the container by executing the bash command.

```
# run the `bash` command on a container named 'my-container'
docker exec -it my-container /bin/bash
```

Here the `-it` flags stand for `interactive` and `terminal` or `interactive terminal` all together. The result is that this command returns a bash shell.

docker attach

`docker attach` will give you a shell on the specified container. This works similarly to an SSH connection and can be used instead of the `exec` command above to get a bash shell on a container.

```
# connect to the container named ansible
docker attach ansible
```

docker image rm

`docker image rm` removes an image from the server

```
# Remove the dvwa container image from the server
docker image rm dvwa
```

Ansible

Ansible is a provisioner that can be used to configure any Linux machine. There is significant documentation for how to use Ansible on their website HERE.

ansible all -m ping

Ansible usually needs a file full of commands called a 'playbook' in order to complete it's tasks. However, you can also execute single commands if you wish using the `-m` flag to specify the `module` you want to use.

With this command we are using the 'ping' module (`-m`) on 'all' of the machines listed in the `hosts` file.

```
# Ping all of the hosts in the hosts file using the ping module
ansible all -m ping
```

ansible-playbook

The `ansible-playbook` command will run the contents of a `playbook.yml` file. The playbook file can be named anything you wish as long as it ends in `.yml` and it has the correct formatting.

```
# Run the playbook 'my-playbook.yml'
ansible-playbook my-playbook.yml
```

Ansible playbooks

Playbooks always carry the `.yml` extension and begin with `---` on the first line to signify that it is a YAML file.

The first few lines of the playbook will give global settings for the file and notate what machines the playbook is to use, if you would like to run the commands as `root` , and what `tasks`

are to be completed.

```
---
- name: Config Web VM with Docker
  hosts: web
  become: true
  tasks:
```

In this Definition, we give the playbook a descriptive `name`, specify that is to run on the machines listed under the `web` headding in the `hosts` file, specify that all commands should be run as root, and then start listing the `tasks`. - `name:` can be anything you wish - `hosts:` specify what machines are to be used. Here we are using the `web` hosts. - `become: true` means: 'Become the root user for all the following commands' - `tasks:` starts the task section where all other commands/tasks will be listed.

Ansible apt module

Ansible modules can easily be found by googling 'Ansible module <name-of-thing-you-want-to-do>'

The `apt` module lets you install applications using the apt-get commands.

```
# Install nano. force `apt-get` to be used instead of just `apt`
  - name: Install nano
    apt:
      force_apt_get: yes
      name: nano
      state: present
```

See all of the `apt` module options HERE

Ansible pip module

The pip module allows you to install python packages as if you were using the command `pip install`

The syntax is just like the `apt` module.

```
  - name: Install Docker python module
    pip:
      name: docker
      state: present
```

See all of the pip module options HERE

Ansible docker-container

The `docker-container` module can be used to download and manage Docker containers.

Here we are downloading the container `cyberxsecurity/dvwa`, staring the container, and forwarding the host port 80 to the container port 80

```
- name: download and launch a docker web container
  docker_container:
    name: dvwa
    image: cyberxsecurity/dvwa
    state: started
    published_ports: 80:80
```

To see all of the `docker-container` module options, click HERE