EJERCICIO 1

Utilizar la herramienta The Harvester sobre el dominio “ucam.edu”.

• Realizar la búsqueda utilizando todas las fuentes disponibles

• Elegir la opción adecuada para utilizar la base de datos de Shodan para consultar los hosts descubiertos.

• Utilizar la opción –n (DNS reverse query)

• Generar un fichero .html con los resultados del escaneo.

• Indicar además los parámetros de ejecución utilizados

python3 theHarvester.py -d ucam.edu -n -s -b all -f ucam.html

TheHarvester se lanza con python. Con este comando se cumplen todas las opciones mencionadas en el enunciado:

-d ucam.ed = busca el dominio ucam.edu

-b all = Usa todas las fuentes disponibles

-s = usa Shodan para consultar los hosts descbiertos

-n = DNS revers query

-f ucam.html = genera el fichero ucam.html con el informe.

Overall statistics

Domains	Hosts	IP Addresses	Vhosts	Emails	Shodan
5	331554	40015	0	50	0

Plugin	Record	Result
filter column...	filter column...	filter column...
CRTsh	host	smtunadca.ucam.edu
CRTsh	host	smtunadcb.ucam.edu
CRTsh	host	www.ucam.edu
CRTsh	host	sede.ucam.edu
CRTsh	host	catedrarsc.ucam.edu
CRTsh	host	www.escuelaidiomas.ucam.edu
CRTsh	host	www.laudatosi.ucam.edu

EJERCICIO 2

1. Utilizar la herramienta Maltego sobre el dominio “ucam.edu”.

• Incluir la entidad Domain con el dominio “ucam.edu”

• Utilizar la transformada “To Website [using search engine]”

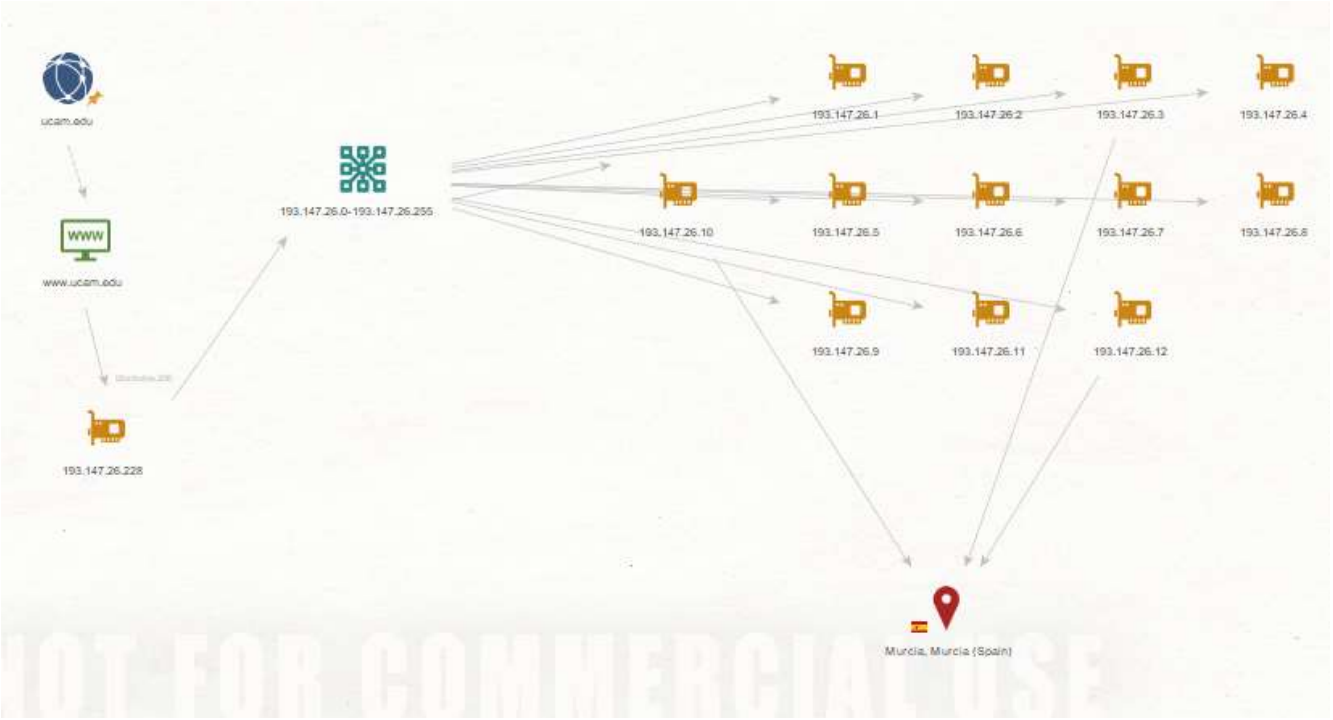
• Sobre el resultado, utilizar la transformada “To IP Address [DNS]”

• Utilizar sobre el resultado la transformada “To Netblock [using natural boundaries]”

• Sobre el netblock creado, utilizar la transformada “To IP addresses”

• Sobre algunas de las IP’s, aplicar la transformada para obtener la dirección (ciudad y país)

• Guardar el resultado y subirlo a la plataforma.



El resultado es Murcia, Murcia (Spain)

2. Partiendo del dominio “ucam.edu”, realizar una investigación propia y adjuntarla a la plataforma

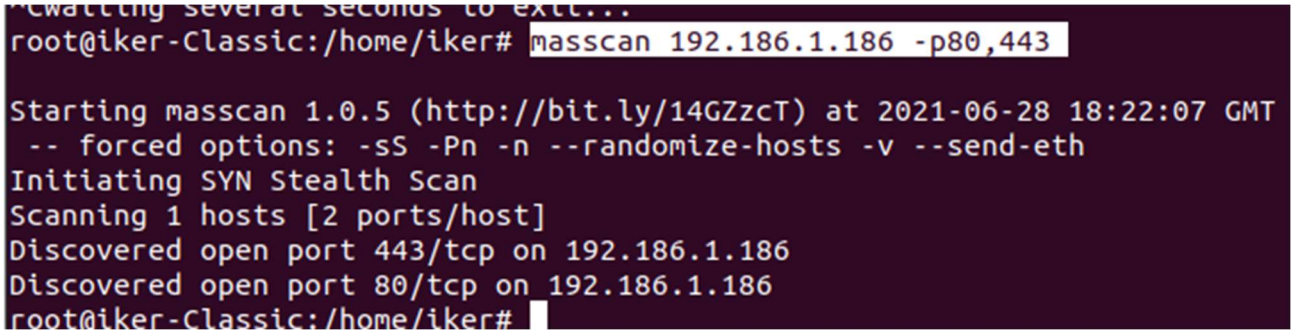
He creado un ejemplo muy sencillo mediante el cual llegamos a obtener información sobre personas de la ucam: he añadido la transformada de To Person, a esta la he pasado a To Email Addresses [PGP] y he obtenido los emails de dos personas de ucam.edu

EJERCICIO 3

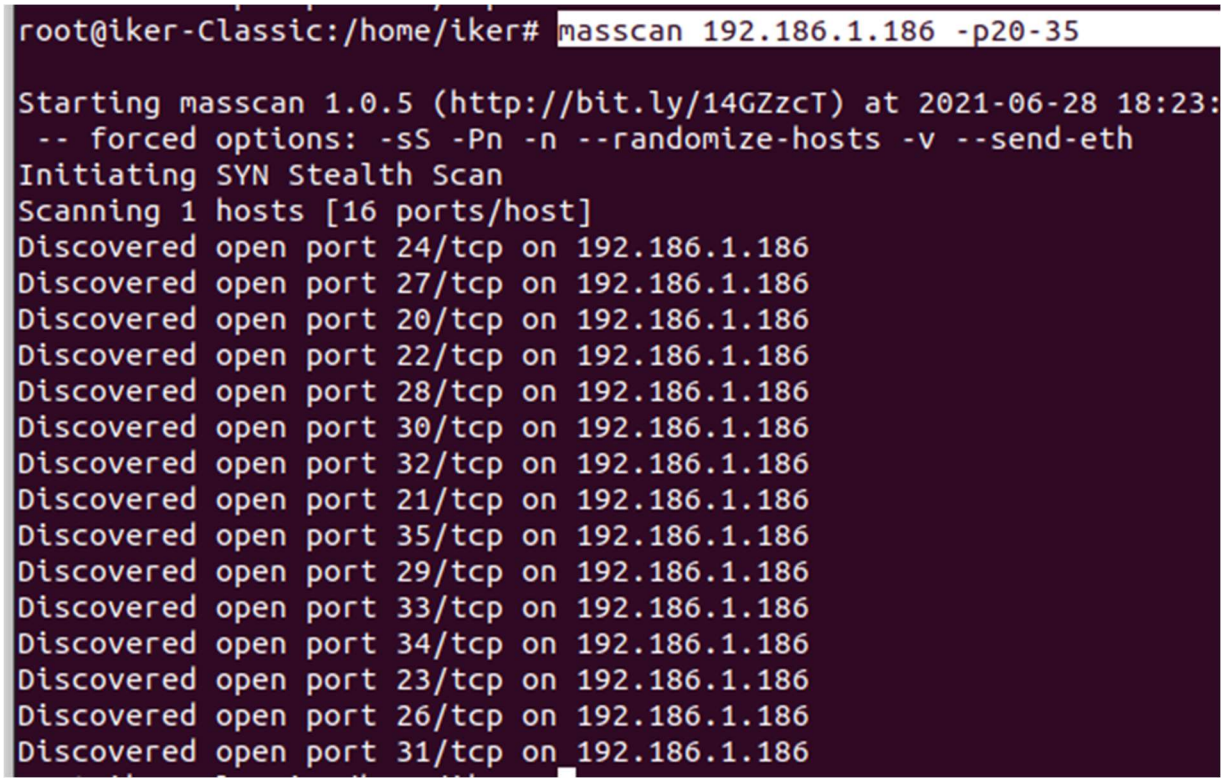
Según la sección anterior, realizar un informe sobre la máquina virtual utilizada en ejercicios anteriores e indicar los resultados que se hayan obtenido.

Los primeros dos comandos nos permiten escanear los puertos 80, 443 y en el segundo el rango del puerto 20 al 35

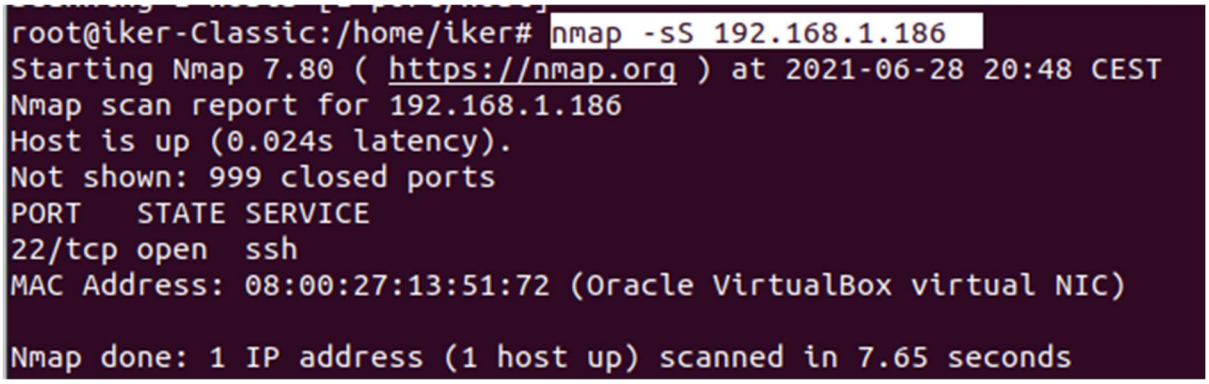
masscan 192.186.1.186 -p80,443



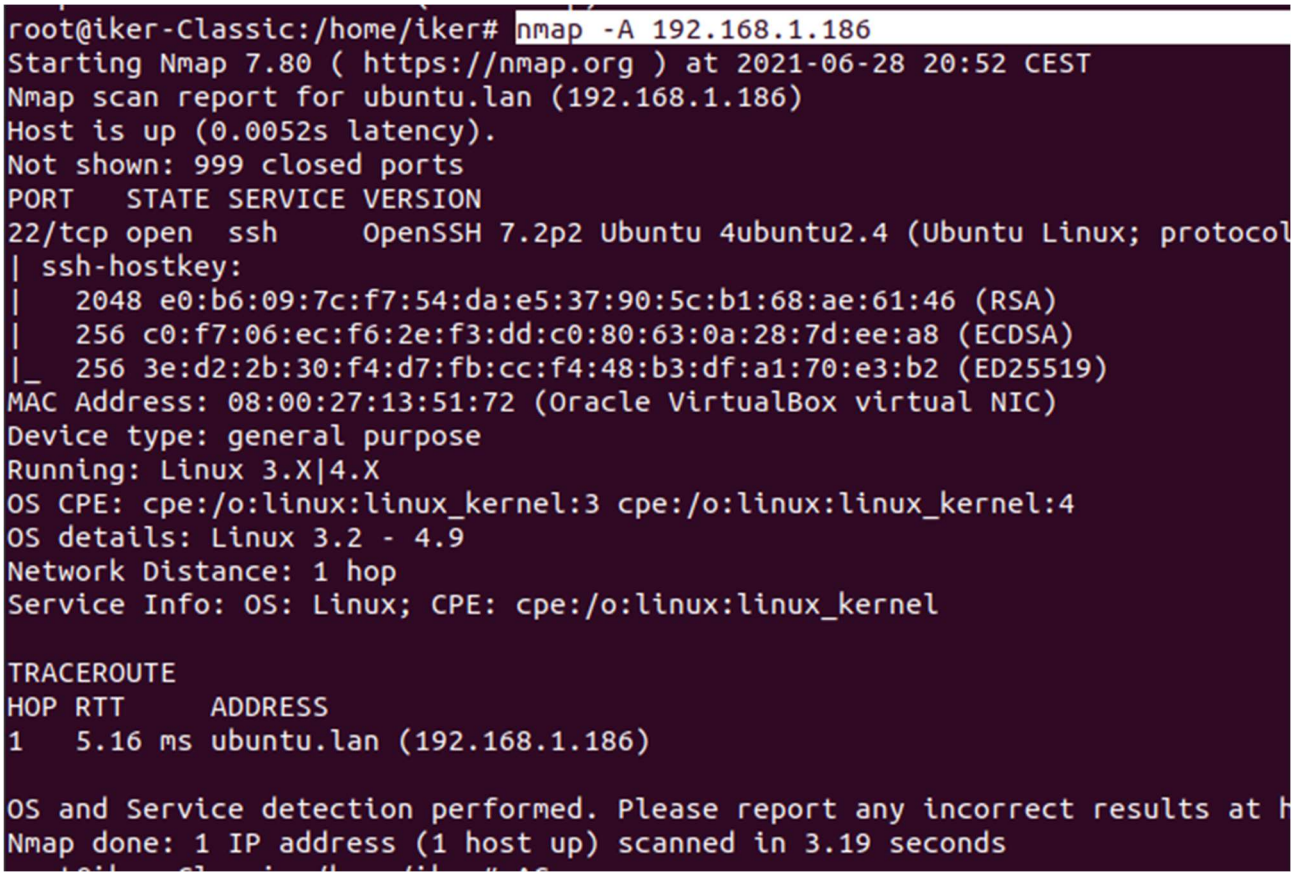
masscan 192.186.1.186 -p20-35



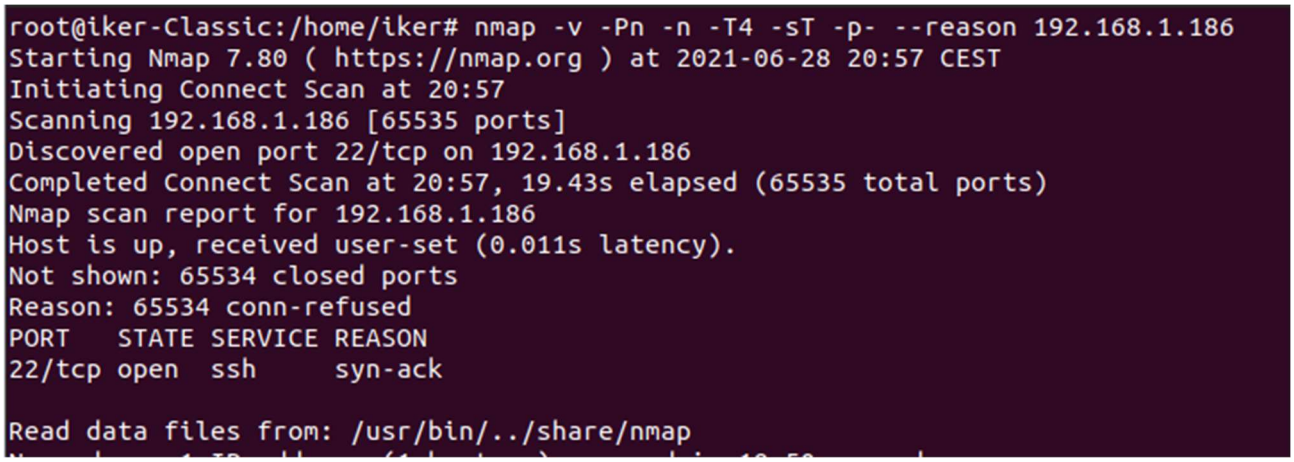
nmap -sS 192.168.1.186



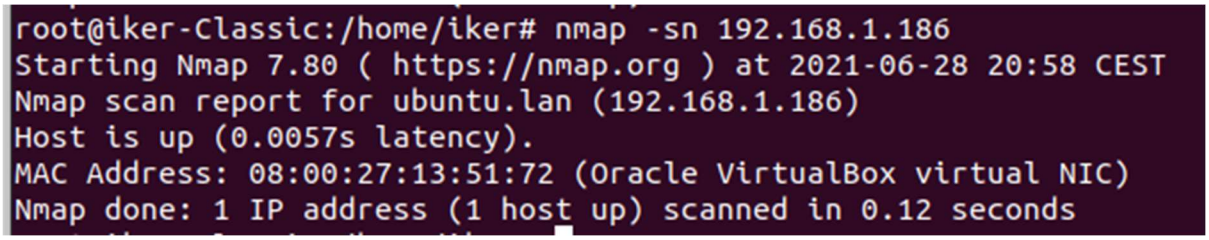
nmap -A 192.168.1.186



nmap -v -Pn -n -T4 -sT -p- --reason 192.168.1.186



nmap -sn 192.168.1.186



En los últimos 4 comandos hemos conseguido la siguiente información:

Tenemos el peurto 22 – ssh abierto.

La MAC del objetivo es 08:00:27:13:51:72 y se corresponde con una VM de Virtual Box

Se trata de un equipo con el S.O Ubuntu, tenemos información del kernel, de algunas ssh-host-key, etc

SEMINARIO 7

*** Como no hay ejercicio he documentado alguno de los pasos ***

Desplegar y configurar Wazuh / OSSEC para trabajar como HIDS

Especificaciones:

• Instalación de los diferentes componentes de Wazuh / OSSEC

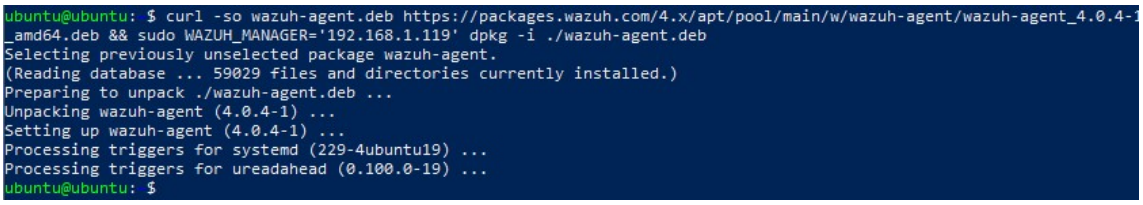
Uso de Wazuh mediante la máquina virtual que ofrecen en su página web:

https://documentation.wazuh.com/4.0/virtual-machine/virtual-machine.html

• Instalación y configuración de un agente

Una vez entramos en la plataforma, está nos indica que comandos tenemos que lanzar en la máquina que queremos instalar el agente. En este caso una VM Ubuntu

Lanzar el comando. En este caso desde una powershell conectados por ssh a la VM de Ubuntu:



Levantar el servicio del agente:

Ya tenemos nuestro agente en Wazuh:

• Repaso de los diferentes puntos de configuración

• Repaso de la funcionalidad más común del panel de administración

SEMINARIO 8

Desplegar Snort + Kippo (NIDS + HoneyPot).

Especificaciones:

• Instalación y configuración de Snort como sistema de detección de intrusos basado en red

• Cambio del puerto por defecto del servidor SSH, pasaría del 22 al 10.000

• Instalación del SSH honeypot Kippo y Kippo-Graph sobre el puerto 22

EJERCICIO 4

Basándose en el seminario anterior, definir una regla de Snort para generar alertas por cada conexión establecida desde cualquier IP y puerto hacia el puerto 22 de la máquina donde está ejecutándose Kippo. El protocolo a aplicar en la regla deberá ser TCP. Describir la sintaxis de la regla, así como los pasos necesarios para incluirla en Snort. Adicionalmente se debe incluir una captura donde se observen las alertas generadas por la regla anterior, para ello, será necesario intentar acceder por SSH al puerto 22.

1. Editamos en el fichero de configuración de snort las variables que apuntan a las rutas donde tenemos nuestras reglas (imagen 1) y también indicamos los ficheros de reglas locales que tenemos en nuestra máquina: local.rules y community.rules (imagen 2)

nano /etc/snort/snort.conf

```
# Path to your rules files (this can be a relative path)
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\rules
var RULE_PATH /etc/snort/rules
var SO_RULE_PATH /etc/snort/so_rules
var PREPROC_RULE_PATH /etc/snort/preproc_rules
```

```
##################################################
# Step #7: Customize your rule set
# For more information, see Snort Manual, Writing Snort Rules
#
# NOTE: All categories are enabled in this conf file
##################################################

# site specific rules
include $RULE_PATH/local.rules
include $RULE_PATH/community.rules
```

2. Editamos el fichero local.rules para añadir la regla:

   nano /etc/snort/local.rules

```
alert tcp any any -> 192.168.1.129 22 (msg:"Snort Kippo Test"; sid:999999999; rev:999)
```

   En la alerta indicamos que recoja el trafico por el protocolo tcp desde cualquier IP, hacia la IP donde esta Kippo: 192.168.1.129 y hacia el puerto 22 de la máquina.

   Entre paréntesis se indica el mensaje que muestra, un id de la regla y un id para las revisiones.

Se puede comprobar que la regla funciona de dos maneras.

Levantando el snort en modo consola:

```
alert tcp any any -> 192.168.1.129 22 (msg:"Snort Kippo Test"; sid:999999999; rev:999)
```

Y tras hacer un intento de conexión por ssh a la máquina donde esta Kippo, vemos lo que captura snort:

```
Commencing packet processing (pid=1199)
06/25-12:26:41.288753  [**] [1:999999999:999] Snort Kippo Test [**] [Priority: 0] {TCP} 192.168.1.248:49467 -> 192.168.1.129:22
06/25-12:26:41.789823  [**] [1:999999999:999] Snort Kippo Test [**] [Priority: 0] {TCP} 192.168.1.248:49467 -> 192.168.1.129:22
06/25-12:26:42.291224  [**] [1:999999999:999] Snort Kippo Test [**] [Priority: 0] {TCP} 192.168.1.248:49467 -> 192.168.1.129:22
06/25-12:26:42.792324  [**] [1:999999999:999] Snort Kippo Test [**] [Priority: 0] {TCP} 192.168.1.248:49467 -> 192.168.1.129:22
06/25-12:26:43.292852  [**] [1:999999999:999] Snort Kippo Test [**] [Priority: 0] {TCP} 192.168.1.248:49467 -> 192.168.1.129:22
```

Y también como tenemos a snort corriendo como servicio, revisando el log que este nos genera:

tail -f /var/log/snort/alert

```
root@ubuntu:/etc/snort/rules# tail -f /var/log/snort/alert
******S* Seq: 0x67D668F4  Ack: 0x0  Win: 0xFAF0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP WS: 8 NOP NOP SackOK

[**] [1:999999999:999] Snort Kippo Test [**]
[Priority: 0]
06/25-12:28:41.292485 192.168.1.248:50426 -> 192.168.1.129:22
TCP TTL:128 TOS:0x0 ID:25993 IpLen:20 DgmLen:52 DF
******S* Seq: 0x67D668F4  Ack: 0x0  Win: 0xFAF0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP WS: 8 NOP NOP SackOK
```

SEMINARIO 9

Desplegar y configurar Vulns / VulnsRepo

Especificaciones:

• Instalación de los diferentes componentes de Vulns

• Instalación de los diferentes componentes de VulnsRepo

• Generar reportes de las diferentes máquinas

• Analizar los diferentes reportes y genarar gráficos en función de ellos

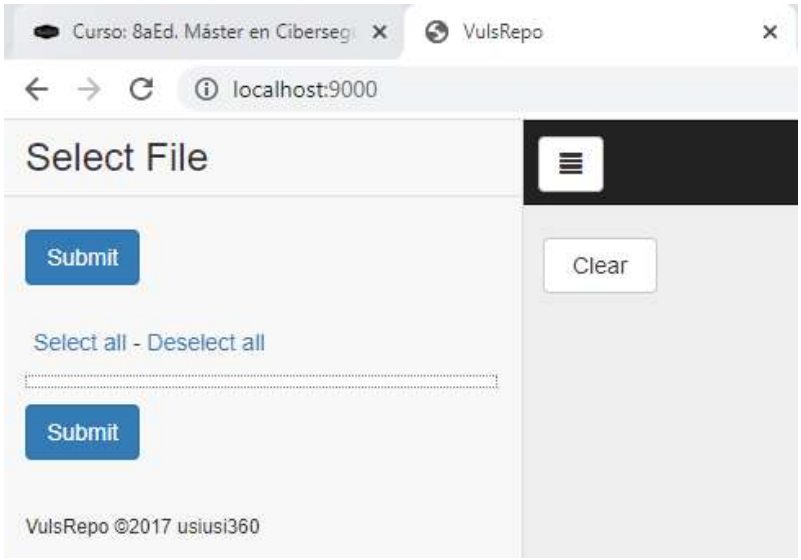
EJERCICIO 5

Según la diapositiva anterior, realizar un informe sobre la máquina virtual utilizada en ejercicios anteriores e indicar los resultados que se hayan obtenido.

Creamos el túnel ssh:

ssh -L 9000:localhost:5111 centos@192.168.18.128

localhost:9000



Todo lo que llega por el puerto 9000, pasa por el puerto 22 a la máquina y es redirigido al puerto 5111,

Para escanear la máquina virtual de Ubuntu editamos el fichero config.toml

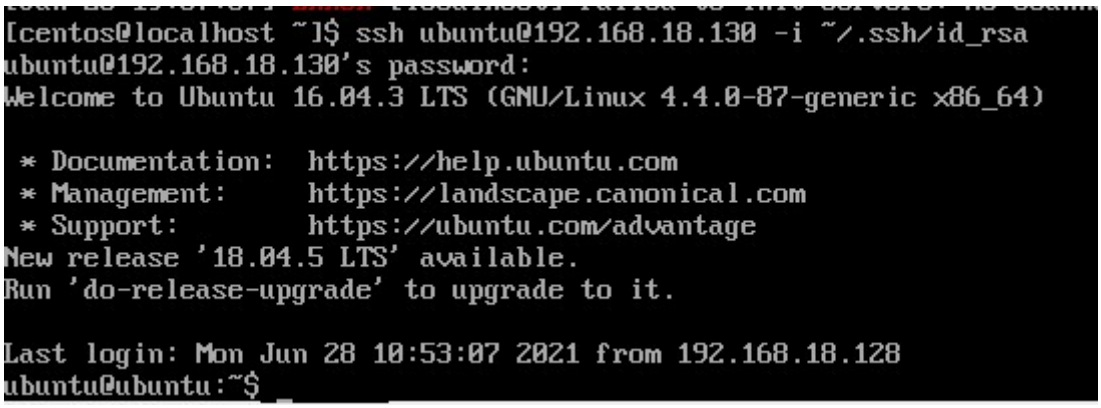
```
[servers.localhost]
host = "localhost"
port = "local"


[servers.ubuntu]
host         = "192.168.18.130"
port         = "22"
user         = "ubuntu"
keyPath      = "/home/centos/.ssh/id_rsa"
```

*** Aquí he tenido un problema que no he conseguido resolver ***

He seguido la guía oficial para configurar la conexión por ssh a la máquina que quiero escanear: https://vuls.io/docs/en/tutorial-remote-scan.html

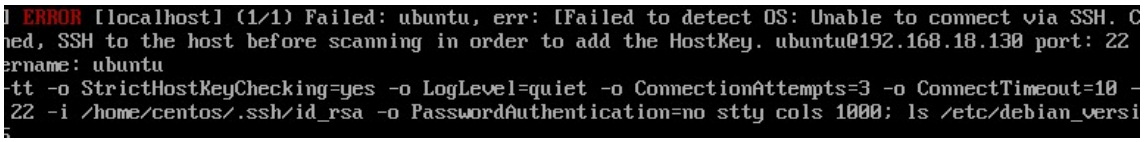
Compruebo que la conexión por ssh es correcta



Dicha conexión queda reflejada en el fichero known_host:

Pero no consigo escanear dicha máquina con vuls:

Tengo todo el rato este error:



He comprobado la red con netstat, tengo el tune ssh, el servicio levantado escuchando en el puerto 5111:

He matado el servicio, reiniciado el mismo, cambiado configuraciones para intentar en otras máquinas, etc pero no he conseguido hacer el escaneo.

Usando el comando vuls configtest ubuntu veo todo el rato el error mencionado arriba, lo mismo que al hacer vuls scan ubuntu. En localhost si que funciona:

EJERCICIO 6

Describir detalladamente que hace el siguiente rol de Ansible.

Entrar en detalle sobre los recursos (files y templates) que se encuentran en él, así como el uso que se le da. Para cada instrucción dentro del fichero main.yml, se debe indicar su cometido, así como los recursos que utiliza.

https://github.com/ansible/ansible-examples/tree/master/mongodb/roles/mongoc

Rol mongoc:

Este rol soluciona errores en las plantillas de mongoc y mongod y también errores orden de los hosts.

Files:

Secret

Contiene las claves de autenticación usadas en la plantilla: mongoc.conf.j2 y en el fichero main.yml

Templantes:

Ficheros de configuración usados por el rol

adduser.j2: Añade usuario Admin con su contraseña.

mongoc.conf.j2: contiene la configuración sobre los logs que vamos a usar en mongo. Indica en varias variables el destino de los logs, que estos se añadan a los existentes, el puerto donde escucha, el path a la base de datos de mongo, usando el fichero secret le pasa la clave, etc

mongoc.j2: es el script para levantar mongod. En el se configuran las variables y rutas necesarias para ello.

Se marca donde esta el fichero de configuración, que opciones le pasamos, donde esta el fichero de configuración del servicio en el sistema, etc

Posteriormente configuramos que hace el servicio cuando lo iniciamos, paramos o re-iniciamos.

Tasks:main.yml:

Este playbook despliega la configuración de mongd en lo servidores.

- name: Create data directory for mongoc configuration server

file: path={{ mongodb_datadir_prefix }}/configdb state=directory owner=mongod group=mongod

Crea el directorio de datos para la configuración de mongoc en el servidor. Indica el path donde lo va a crear, que usuario es el propietario y el grupo del mismo.

- name: Create the mongo configuration server startup file

template: src=mongoc.j2 dest=/etc/init.d/mongoc mode=0655

Crea el fichero de configuración para iniciar el servicio de mongo. En el src indica que plantilla va a utitizar: mongoc.js el destino en la máquina y en el mode los permisos que otorga a la misma.

- name: Create the mongo configuration server file

template: src=mongoc.conf.j2 dest=/etc/mongoc.conf

Crea el fichero de configuración del servicio de mongo. En el src indica que plantilla va a usar: mongoc.conf.j2 y el destino del mismo en el server.

- name: Copy the keyfile for authentication

copy: src=roles/mongod/files/secret dest={{ mongodb_datadir_prefix }}/secret owner=mongod group=mongod mode=0400

Copia el fichero con las claves de autenticación. En el src indica desde donde las copia: /files/secret, el destino en el server e indica usuario propietario, grupo y permisos sobre el fichero.

- name: Start the mongo configuration server service

command: creates=/var/lock/subsys/mongoc /etc/init.d/mongoc start

Inicia el servicio de mongo en el servidor, mediante la orden start sobre el fichero que hemos copiado antes en /etc/init/d/mongoc

- name: pause

pause: seconds=20

Una pausa de 20 segundos.

- name: add the admin user

mongodb_user: database=admin name=admin password={{ mongo_admin_pass }} login_port={{ mongoc_port }} state=present

ignore_errors: yes

Añade el usuario administrador, indicando el nombre, su contraseña, el puerto por el que hace login y el estado.

SEMINARIO 10

Desplegar un servidor MySQL usando Vagrant

Especificaciones:

• Inicializar un VagrantFile a partir de la box Ubuntu/trusty64

• Configurar Vagrant para trabajar con una receta de ansible

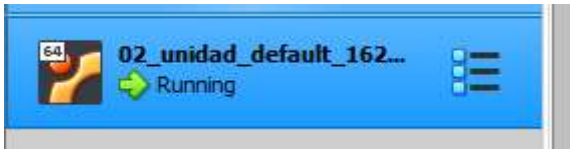
Desplegar el entorno y usar algunos de los comandos típicos.

Creamos el fichero vagrant

```
vagrant init ubuntu/trusty64
```

```
PS C:\Users\Admin\Documents\Tech\ENIIT\Ciberseguridad\04_Arquitectu
A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.
```

Con el comando: vagrant up creamos en virtual box la VM:



Con vagrant ssh entramos en la maquina:

```
Welcome to Ubuntu 14.04.6 LTS (GNU/Linux 3.13.0-170-generic

 * Documentation:  https://help.ubuntu.com/

  System information as of Sun Jun 27 06:49:42 UTC 2021

  System load:  0.63              Processes:           81
  Usage of /:   3.6% of 39.34GB   Users logged in:     0
  Memory usage: 25%               IP address for eth0: 10.0
  Swap usage:   0%

  Graph this data and manage this system at:
    https://landscape.canonical.com/

UA Infrastructure Extended Security Maintenance (ESM) is no

0 updates can be installed immediately.
0 of these updates are security updates.

Enable UA Infrastructure ESM to receive 64 additional secur
See https://ubuntu.com/advantage or run: sudo ua status

New release '16.04.7 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


vagrant@vagrant-ubuntu-trusty-64:~$
```

Para desplegar el servidor de MySQL usando Ansible, lo primero que tenemos que editar es el fichero Vagrantfile:

```
Vagrant.configure("2") do |config|

  config.vm.box = "ubuntu/trusty64"

  config.vm.network :private_network, ip: "192.168.33.33"
  config.vm.network "forwarded_port", guest: 3306, host: 3306

   config.vm.provision "ansible" do |ansible|
    ansible.playbook = "site.yml"
    ansible.verbose = "vv"
   end
end
```

En él, a parte de marcar la ip y hacer u port forwarding, le indicamos que vamos a usar Ansible y le decimos cual es el nombre de la receta que vamos a usar: site.yml

En esta receta site.yml configuramos lo siguiente:

```
- name: deploy MySQL and configure the databases
  hosts: all
  remote_user: root

  vars_files:
    - inventory/group_vars/all

  roles:
    - db
```

El nombre de la receta, en el cual indicamos que vamos a desplegar mysql. El fichero de variables que tenemos configurado en el fichero all y el rol de ansible que vamos a utilizar: db

El fichero all, configuramos la conexión al servicio de mysql.

```
mysqlservice: mysql
mysql_port: 3306
dbuser: prueba
dbname: foodb
upassword: 123456
```

Y en el rol de ansible tenemos dos ficheros que tenemos que tener en cuenta: main.yml y la plantilla my.cnf.j2

main.yml

```yaml
- name: Install Mysql package
  package: name={{ item }} state=installed
  become: yes
  become_user: root
  with_items:
    - mysql-server
    - python-mysqldb

- name: Create Mysql configuration file
  template: src=my.cnf.j2 dest=/etc/mysql/my.cnf
  become: yes
  become_user: root
```

En este rol de ansible vamos instalar mysql-server y Python-mysqldb y vamos a configurar mysql mediante la plantilla my.cnf.j2

También arrancamos el servicio, creamos una base de datos y un usuario.

my.cnf.j2

```
[client]
port            = 3306
socket          = /var/run/mysqld/mysqld.sock
bind-address    = 0.0.0.0


[mysqld_safe]
socket          = /var/run/mysqld/mysqld.sock
nice            = 0


[mysqld]
user            = mysql
pid-file        = /var/run/mysqld/mysqld.pid
socket          = /var/run/mysqld/mysqld.sock
port            = {{ mysql_port }}
```

En la plantilla tenemos configuradas las variables del servicio.

Simplemente tenemos que ejecutar vagrant up para que descargue la box requerida y por medio de ansible instale y despliegue mysql.

SEMINARIO 11

Desplegar y configurar Nagios

Especificaciones:

• Instalación de los diferentes componentes de Nagios

• Instalación de los diferentes componentes de NRPE

• Activar la monitorización de la máquina virtual remota

• Analizar el estado de monitorización de las máquinas

EJERCICIO 7

Basándose en el seminario anterior, ajustar la configuración para monitorizar el número de procesos zombie existentes en la máquina de la unidad 1, configurada al final del seminario. Describir todos los pasos

Una vez instalado nrpe en el servidor y en la sonda. Comprobamos que accedemos desde el servidor a la misma:

Para poder monitorizar servicios en la máquina sonda, tenemos que editar dos servicios en la máquina servidor. El primero lo hemos llamado unidad1.cfg y se encuentra en /usr/local/nagios/etc/servers En el cual vamos a definir el host (sonda) al que nos vamos a conectar

Creamos el fichero de configuración en server con la información de la máquina sonda:

nano /etc/local/nagios/servers/unidad1.cfg

```
File: /usr/local/nagios/etc/servers/unidad1.cfg
```

```
define host{
        use                             linux-server
        host_name                       Munidad1
        alias                           MAliasUnidad1
        address                         192.168.1.186
        max_check_attempts              5
        check_period                    24x7
        notification_interval           30
        notification_period             24x7
}
```

En el definimos el host (sonda) y configuramos las variables deseadas.

En este caso indicamos que es un Linux-server (requerido), usamos un nombre de host para el mismo e indicamos la ip. También le marcamos los periodos e intervalos de monitorización.

También tenemos que definir los servicios que vamos a monitorizar:

```
define service {
        use                             generic-service
        host_name                       Munidad1
        service_description             Zombie procs
        check_command                   check_nrpe!check_zombie_procs
}
```

Le indicamos el uso, el nombre del host, una descripción para el servicio y el comando que va a lanzar check_nrpe para monitorizar.

En esta ultima línea le estamos diciendo a nagios que en la máquina server, lance con check_nrpe el comando check_zombie_procs configurado en la máquina sonda.

Dicho comando se puede consultar en el host sonda, en la siguiente ruta:

/usr/local/nagios/etc/nrpe.cfg

```
 GNU nano 2.5.3                    File: /usr/local/nagios/etc/nrpe.cfg

# are in the following format:
#
# command[<command_name>]=<command_line>
```

Para que podamos lanzar el comando check_nrpe desde el server, tenemos que configurarlo en el fichero: /usr/local/nagios/etc/objects/commands.cfg

```
 GNU nano 2.5.3                File: /usr/local/nagios/etc/objects/commands.cfg
```

```
define command{
        command_name check_nrpe
        command_line /usr/local/nagios/libexec/check_nrpe -H $HOSTADDRESS$ -c $ARG1$
}
```

Le indicamos el nombre del mismo y lo que ejecuta al ser lanzado.

Una vez tenemos estos dos ficheros configurados en el servidor, reiniciamos el servicio de nagios y en un intervalo de tiempo no muy largo, ya tenemos los dos servicios (he dejado también cpu load activado) sobre la VM de la unidad 1 activados.

Host	Service	Status	L
Munidad1	CPU Load	OK	0
	Zombie procs	OK	0

En el caso de los procesos zombies, podemos ver que actualmente no hay ninguno:

Zombie procs	OK	06-26-2021 23:40:34	0d 0h 0m 57s	1/3	PROCS OK: 0 processes with STATE = Z

Service State Information

Current Status:	OK (for 0d 0h 1m 33s)
Status Information:	PROCS OK: 0 processes with STATE = Z
Performance Data:	procs=0;5;10;0;
Current Attempt:	1/3 (HARD state)
Last Check Time:	06-26-2021 23:40:34
Check Type:	ACTIVE
Check Latency / Duration:	0.000 / 0.032 seconds
Next Scheduled Check:	06-26-2021 23:50:34
Last State Change:	06-26-2021 23:40:34
Last Notification:	N/A (notification 0)
Is This Service Flapping?	NO (0.00% state change)
In Scheduled Downtime?	NO
Last Update:	06-26-2021 23:41:59 (0d 0h 0m 8s ago)