Intrusion Detection Systems Assignments

1. Configuration of Snort

You need 2 VMs for this assignment: - an attacker VM (Kali prefered) - a VM with Snort installed

1. Run `ifconfig` to see all network interfaces
2. Get your network interface and the corresponding IP address
3. Open `/etc/snort/snort.config` and modify `$_HOME_NET` with your IP address
4. Open `/etc/snort/rules/local.rules` and insert following rule: `alert icmp any any -> $HOME_NET any (msg:"machine was pinged"; sid:1000001; rev:1; classtype:icmp-event;)`
5. Run `sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i enp0s3`
6. Ping IDS VM from attacker VM
7. IDS VM shows the configured alert

2. Interaction with a FTP Server

You need 2 VMs for this assignment: - an attacker VM with vsftpd installed - a VM with Snort and vsftpd installed. Add a user via `sudo adduser test`

Hint: If Snort doesn´t show alerts, maybe start it with option `-k none`

1. Get IP from attacker VM from configuration task
   `solution:`
   `use ipconfig`
2. Create an alert which detects connections to ftp server on IDS VM from attacker VM
   `solution:`
   `alert tcp <attacker-ip> any -> $HOME_NET 21 (msg:"connection attempt from suspicious IP"; flags:S; sid:1000002; rev:1;)`
3. Create an alert which detects failed authentications on IDS VM's ftp server
   `solution:`
   `alert tcp $HOME_NET 21 -> 192.168.178.65 any (msg:"FTP login failed by suspicious IP"; content:"Login incorrect."; sid:1000003; rev:1;)`

3. Port scans

1. Write a rule that is able to detect a TCP Scan on Port 22
   `solution:`
   `Rule: alert tcp any any -> $HOME_NET 22 (msg: "NMAP TCP Scan";sid:10000005; rev:2; )`
   `Run TCP scan with: nmap -sT -p22 <IP-address>`
2. Write a rule that is able to detect a FIN Scan on Port 80
   `solution:`
   `Rule: alert tcp any any -> $HOME_NET any (msg:"NMAP FIN Scan"; flags:F; sid:1000008; rev:1;)`

`Run FIN scan with: nmap -sF -p80 <IP-address>`

Honeypot Assignments

In order to execute these tasks, docker and docker-compose must be installed and the two container images `cowrie` and `honeytrap` have to be installed.
`Solution:`
`Pull cowrie container: docker pull cowrie/cowrie`
`Pull honeytrap container: docker pull honeytrap/honeytrap && docker network create honeytrap`

1. Medium Interactive SSH Honeypot

Use `cowrie` to set up a SSH "vulnerability". Start the `cowrie` docker container and configure it to open a weak protected SSH port.

1. Run the Cowrie container and use docker publish parameter to bind port 22 from the hostmachine to port 2222 from the cowrie container
   [Hint: docker run -p host_port:container_port cowrie/cowrie]
   `Solution: docker run 22:2222 cowrie/cowrie`
2. Open a new terminal and ssh into the machine [hint ssh root@localhost] ... But what is the password? Maybe try the most commonly used passwords ....
   `Solution: Cowrie will accept every password entered, you can even enter no password ssh localhost`
3. When you have access to the machine look around, find out what kind of machine this is, check that we are root and download some maleware (or wget any website). Did you find any hints you are in a Honeypot?
   `Solution:`
   `Type of machine: uname -a`
   `Root-check: whoami`
   `"Malware"-Download: wget google.de`
   `Detect the honeypot: Many solutions, e.g. when reconnecting the last login date will be wrong, created files will be deleted after a connection reset, ....`
4. Now switch to the defender point of view: As you can see in the window, where honeypot was started, cowrie logs everything we do.
5. When we use docker to access the container with `docker exec -ti cowrie bash` we can access the log files under `~/cowrie-git/var/log/cowrie/cowrie.json`
6. Downloaded files are located under `~/cowrie-git/var/lib/cowrie/downloads/`
7. Stop the container `docker stop cowrie`

2. Low Interactive HoneyTrap

1. Start the HoneyTrap container using docker-compose. Navigate to the directory `./config-files/honeypot/task2` and start the container(s) with `docker-compose up`

[Note: Make sure you stopped the cowrie container, otherwise docker can't publish port 22 as the cowrie container is still publishing it]

2. Start a ssh-service on port 22 and a telnet- service on port 23. Take a look at the slides or look at the manuals [https://docs.honeytrap.io/services/]. The config file is located in the same directory and is called `config.toml`
   [Note: You don't need to edit the `docker-compose.yml` , it is configured for all tasks.]
   [Note 2: You need to restart the docker honeytrap containe`docker-compose restart honeytrap`]
   `Solution: Can be found in ./config-files/honeypot/task2/2.1-2.6/config.toml`
3. Log in as root with SSH.
   [Note: What is the password? Maybe take a second look at the `config.toml` .]
   `Solution There are two passwords configured for user root: "root" and "password"`
   `ssh localhost`
4. How does it differ from `cowrie` ?
   `Solution: You can't use any services.`
5. Now setup services `tcp/80` (for HTTP) and `tcp/443` (for HTTPS) to imitate an Webserver.
   [Note 1: You need to restart the honeytrap server afterwards]
   [Note 2: Try to access these sites from a browser]
   `Solution: Can be found in ./config-files/honeypot/task2/2.1-2.6/config.toml`
6. Access these sites. What do you see?
   `Solution: on http://localhost you only received the minimum HTTP-Headers, which your browser will display as a blank site. At https://localhost you get the initiation of the SSH-handshake wich won't be continued by honeytrap`
7. [Optional] Visualize the honeypot events with Kibana. Use therefore the files in the config and docker-compose files in the `./config-files/honeypot/task2/2.7` -folder
   `Solution:`
   `Stop the currently running honeytrap-container: docker kill honeytrap && docker rm honeytrap`
   `Execute steps 2.7.1 - 2.7.3`
   `Head to directory ./config-files/honeypot/task2/2.7/`
   `Create Docker-Network "honeytrap": docker network create honeytrap`
   `Start honeytrap-container with new configuration: docker-compose up`

    1. Switch off the VM and give it the maximum of power you can afford.
    2. Add to file `/etc/sysctl.conf` the following content: `vm.max_map_count=262144` , to increaese the OS's version maximum mmap size. Elasticsearch uses mmapfs directory by default to store its indices.
    3. Execute `sudo sysctl -p` to apply this changes
    4. Execute the docker-compose-file `Head to directory ./config-files/honeypot/task2/2.7/`
       `Execute: docker-compose up`
    5. Wait until all three containers are running.
    6. Head to `localhost:5601` in a browser
    7. Set `honeytrap` as Index Pattern.

8. Press on `Discover` on the left side and take a look at the events.
9. In order to filter the cyclic heartbeat messages, use `NOT category: heartbeat` as a filter