CYBER CHALLENGE.IT

Attack/Defense demo

Introduction

Welcome to final Attack/Defense competition for the participants of the CyberChallenge.IT training program. The competition follows the typical Attack/Defense format.

The rules listed on this page may change as more issues are raised by the participants. Also, the organisers keep the right to change them at any time. Keep in mind that it is not feasible to list all the rules and exceptions to the rules that apply to the CTF competition. When in doubt, ask to the competition admins.

Schedule

The competition will take place on June 30th. A detailed schedule for the day is provided below (CEST time):

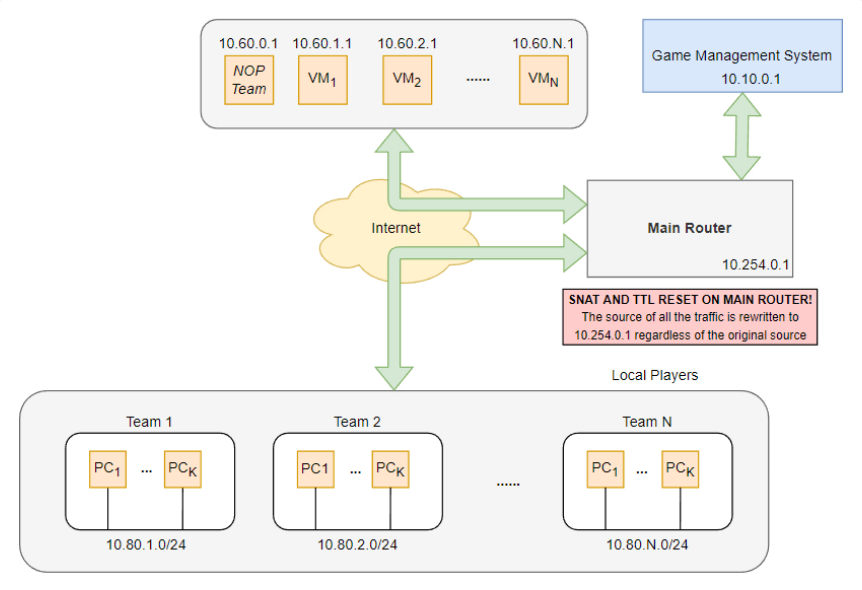
- **10:00 - 11:00**: access to the VMs is granted, but the network is closed. Teams should use this time to analyse services and customize their VMs before everyone can attack each other.
- **11:00 - 18:00**: the network is open! Flags are dispatched to each service by the Game System and teams can earn points by submitting them.

Network and Setup

The game is played within the `10.0.0.0/8` subnet. Each team has its own vulnerable machine located at `10.60.team_id.1`, while players connecting to the game network are assigned an ip in the `10.80.team_id.1/24` subnet.

The ip address `10.60.0.1` is assigned as *NOP Team* (Non-playing team) vulnerable VM, this VM will not be patched during the competition and its flags will not count towards the scoreboard you can use it to test your attacks.

All vulnerable VMs will be hosted by organisers and will have enough resources in terms of CPU and memory to run the pre-installed services.



The Game System is responsible for dispatching flags to the vulnerable machines, checking services integrity, hosting the scoreboard and updating scores. Participants are asked to attack vulnerable machines of other teams to retrieve proofs of successful exploitation (flags). Flags must be submitted to the flag submission service hosted by the organisers to score points. At the same time, teams must defend the vulnerable services installed on their VMs. Teams can do whatever they want within their network segment.

Internet access is granted to install new software on the VM and on the laptops of participants, if needed. Organisers discourage interaction between CTF network and remote servers (e.g., starting attacks from cloud): bruteforce attacks or large computational resources are not required to succeed at the competition.

Beware that if you mess up your vulnerable machine, all we can do is reset it to its original state (backup your exploits, tools and patches!). Resetting a vulnerable machine and return to a valid game state can take a long time and may lead to a high loss of points in the

competition.

Default SSH user for the VM is `root` and the password will be sent to the teams before the competition start.

Inside each vulnbox a public key owned only by the admins is inserted. It won't be used by the admins to interfere with the challenge, but only to provide any support (push updates, solve possible issues, ...). You can freely remove it (at your own risk). The valid admin public key in `/root/.ssh/authorized_keys` file is:

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDk6j1qfO45eXfyBekQcps7Qml1A/zHgLOYSAdbImwUnO+SzR5KZMFtRIoSqzzJ6/tzfixGJOrWtzJZUO12ThwRQgym/oKtOpgXn2g
```

Scoring

The game is divided in rounds (also called ticks) with the duration of 120 seconds. During each round, a bot will add new flags to your vulnerable machine. Moreover it will check the integrity of services by interacting with them and by retrieving the flags through legitimate accesses.

Your team gains points by attacking other teams and by keeping services up and running. The total score is the sum of the individual scores for each service. The score per service is made of two components:

- Offense: Points for flags captured from other teams and submitted to the Game System within their validity period
- SLA: % for the availability and correct behavior of your services (up tick / total tick)

The score for a `flag` of a `service` stolen by an attacking team `attacker` from a victim team `victim` will be assigned dynamically according to the formula:

```
scale = 15 * sqrt(5)
norm = ln(ln(5)) / 12
offense_points[flag] = scale / (1+exp((sqrt(score[attacker][service]) - sqrt(score[victim][service]))*norm))
defense_points[flag] = min(victim_score, offense_points)
```

According to the previous pseudocode, the attacking team will be assigned `offense_points` points and the victime team will loose `defense_points` points:

```
# Service base points
score[team][service] = 5000

# Sum offensive points
for flag in stolen_flags[team][service]:
  score[team][service] += offense_points[flag]

# Substract defensive points
for flag in lost_flags[team][service]:
  score[team][service] -= defense_points[flag]
```

The final team score will be the sum of the score for each service multiplied by its service SLA %.

```
total_score[team] = 0

for service in services:
  # Compute SLA of the service
  sla[team][service] = ticks_up[team][service] / ticks[team][service]
  # Limit scores to 0
  score[team][service] = max(0, score[team][service])
  # Add service score
  total_score[team] += score[team][service] * sla[team][service]
```

Some consideration about the scoring system:

- SLA % it is not added to the final score but it is a multiplying factor of the total score
- The score of a valid flag is correlated by the difference in service score of the two teams
- You will gain more points stealing flags from teams with higher service score
- You will gain less points stealing flags from teams with lower service score

For each team, the scoreboard will list the total score and for each service its flag points, the number of captured and lost flags, the SLA in % and three different status indicators for the various types of possible failures that can occured in your services.

In each round, at most three different kind of checks will take place on your services:

- Check SLA: it checks the availability of the service
- Put flag: it puts a new flag in the service
- Get flag: it try to retrieve a valid previously inserted flag. Please note that this check is not going to be performed if previous `N` put flag checks failed (where `N` is the flag lifetime).

The status of each check will also be included in the scoreboard, the possible status values are:

OK: everything works

DOWN: something's wrong

SYSTEM_ERROR: error in platform worker

Please note that, regarding the SLA %, a service is considered up (`ticks_up[team][service]`) only when every check performed in a round is successful (`OK`).

Since there are countless ways to break a service, the scoreboard may be not provide full information if a service is marked as corrupt or mumble. Try to restore the service from your backup (please backup it before applying any patch) and check if the service is marked as up in few minutes.

Flags

A flag is a string made up of 31 uppercase alphabetic or numeric chars, followed by =. Each flag is matched by the regular expression `/^[A-Z0-9]{31}=$/`.

You can submit stolen flags by performing an HTTP PUT request to the Game System at `http://10.10.0.1:8080/flags`. The flags must be submitted as an array of strings and the requests must contains the header `X-Team-Token` set to the team token given to the participants.

As an example, we provide a simple python snippet that accounts for the submission of two flag using.

```
import requests

TEAM_TOKEN = '4242424242424242'

flags = ['AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA=', 'BBBBBBBBBBBBBBBBBBBBBBBBBBBBB=']

print(requests.put('http://10.10.0.1:8080/flags', headers={
    'X-Team-Token': TEAM_TOKEN
}, json=flags).text)
```

The request will return a json array, for each sent flag you will receive an object in the form of:

```
{
    'msg': f'[{flag}] {message}',
    'flag': flag,
    'status': true/false
}
```

Where the message can be:

- Accepted: X flag points
- Denied: invalid flag
- Denied: flag from nop team
- Denied: flag is your own
- Denied: flag too old
- Denied: flag already claimed

Please note that the request status code 500 means that the request is malformed, team token header is not valid or the game is ended. In any case there is going to be a description of the problem in the response body.

Flags are considered expired after 5 rounds. It means that teams have up to 10 minutes to steal a flag and submit it. At the same time, the check bot will try to retrieve one of the last 5 flags from a service to check if the intended functionalities have been preserved and mark it as up.

Flag IDs

Some services have "Flag ID"s, additional information that you might need for an exploit. Usually this is the username of the Game System's account that stores the flag. The flag ids are only given for flags that are still valid.

The endpoint http://10.10.0.1:8081/flagIds gives you a JSON with all information you need during the competition. It is updated after each tick. Format:

```
{
    "service_1": {
        "10.60.1.1": [
            "flag_id_service_1_team_1",
            "flag_id_service_1_team_1",
            "flag_id_service_1_team_1"
        ],
        ...
        "10.60.8.1": [
            "flag_id_service_8_team_8",
            "flag_id_service_8_team_8",
            "flag_id_service_8_team_8"
        ]
    }
}
```

Technical and Human Behaviour

We'd like everyone to enjoy a fair game. For this reason we ask you to follow these simple rules:

- You are only allowed to attack targets on the subnet `10.60.0.0/16`
- No attacks against the infrastructure including denial-of-service (DoS), floods, DNS poisoning, ARP spoofing, MITM, etc...
- The only permitted attack targets are the vulnerable machines! Players are not allowed to attack each other (e.g., you can't break into rivals' laptops)
- Destructive behavior are severely prohibited. These include removing flags or deleting files on compromised hosts, creating fake flags to break legitimate attacks, DoS against vulnerable services
- Unfair practices are not allowed. These include any action aimed at hindering others or exploiting advantages not related to skills and preparation
- Network vulnerability scanners (with the exception of a few manual runs of nmap) are not allowed, do something better!

- Sharing flags, exploits or hints between teams is severely prohibited and will grant you the exclusion from the competition
- When in doubt, ask to the organizers

Before attempting to break one of the aforementioned rules, remember that all the network traffic is logged. Violations to these rules will be evaluated by the organisers who reserve the right to penalize or exclude teams and individual from the competition.

Tips for the CTF

Change your password

The first thing you have to do when the competition starts is to change the root password of your virtual machine : since you may connect directly as root with ssh, it is enough to use the command passwd.

```
root@diff:~# passwd
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Docker and Docker-Compose

All services will run under docker and docker-compose. Docker provides a platform for OS-level virtualization and enables developers to package applications in containers.

docker cheat-sheet

- `docker ps`: list running containers
- `docker ps -a`: list running and non running containers
- `docker image ls`: list the docker images stored in the local machine
- `docker image rm imagename`: remove a docker images
- `docker run -it ubunutu:20.04 bash`: execute and save a container running bash in ubuntu 20.04
- `docker start containername`: start a non running container
- `docker stop containername`: stop a running container
- `docker rm containername`: remove a container

NOTE: instead of containername is possible to use the container id

docker-compose cheat-sheet

In the directory that contains the file docker-compose.yml.

- `docker-compose ps`: container status
- `docker-compose build`: build all the images of the compose
- `docker-compose up`: start all the containers in the compose
- `docker-compose up --build`: start all the containers in the compose and rebuild the images
- `docker-compose up [--build] -d`: start all the containers in the compose and exit (start in the background)
- `docker-compose stop`: stop all the containers in the compose but keep them as stopped
- `docker-compose down`: stop all the containers in the compose and remove all the containers

Note that during the CTF you will need to rebuild the docker images to apply the patches. Switch to the service folder and execute: `docker-compose up --build -d`

This command will rebuild all the images and run the containers in the background.

Switch user

In your machine there might be a different user for each service: since it is a good habit to use root only when strictly necessary, you may consider the possibility of switching to the user that owns the service on which you want to work. This can be done with the command `su -`; if you want to go back to the previous user, you can use the combination CTRL + D or the command exit.

```
root@diff:~# su -
pin pin@diff:~$ id uid=1001(pin) gid=1001(pin) groups=1001(pin)
pin@diff:~$ exit logout
root@diff:~#
```

Files search

find is a very powerful tool for searching files inside a given path with some particular properties: for instance you can ask to list files that belong to a given user, search for files on which you have particular permissions and so on. You can also combine different properties using logical operators.

Network connections

Information about active network connections can be seen using netstat as follows:

```
diff@diff:~$ netstat -natup
(Not all processes could be identified, non-owned process info
will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name
tcp 0 0 0.0.0.0:8000 0.0.0.0:* LISTEN 32098/python
tcp 0 0 0.0.0.0:42765 0.0.0.0:* LISTEN 238/python
tcp 0 0 0.0.0.0:42766 0.0.0.0:* LISTEN -
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN -
tcp 0 672 192.168.69.22:22 192.168.69.159:51987 ESTABLISHED -
tcp6 0 0 :::22 :::* LISTEN -
```

Column Proto reports the transport protocol in use, Local Address and Foreign Address denote, respectively, addresses of the parts involved in the connection, State is the state of the connection (except for connectionless protocols) and PID/Program name shows the PID and the name of the program that has created the connection. The last column shows information only of processes owned by the user that gives the command: if you want to see them all, you have to run the command as root.

Network monitoring

you may want to see the traffic you're receiving: in this case, use the tcpdump command:

```
$ tcpdump -i game -s0 -w traffic.pcap not port 22
```

This command saves the network traffic from interface game into the traffic.pcap file that can be later analyzed with tools like wireshark. The not port 22 filter prevents tcpdump from saving your ssh session traffic, of which you are probably not interested.

Saving the network traffic on the vulnbox can lead to saturate the available space and lead to request a vm reset, pay attention to the space left!