به نام خدا

«امنیت داده و شبکه»

دکتر امینی

تمرین چهارم

علیرضا دهقانپور فراشاه

۹۸۱۰۱۵۵۵

سوال اول

views.py 2 | urls.py 1

C: > Users > alireza > Desktop > HW4 > vuln web > xss > views.py > ...

```python
# Create your views here.
def index(request):
    return render(request, "xss.html")

def l1(request):
    if request.method == "GET":
        context = {'name': 'Anonymous User'}
        #template = loader.get_template("./xss/templates/xss_l1.html")
        return render(request, 'xss_l1.html', context)
    elif request.method == "POST":
        name = request.POST.get('name', 'Anonymous User')
        context = {'name': name}
        return render(request, 'xss_l1.html', context)
    else:
        return "Error"

def l2(request):
    if request.method == "GET":
        context = {'name': 'Anonymous User'}
        #template = loader.get_template("./xss/templates/xss_l2.html")
        return render(request, 'xss_l2.html', context)
    elif request.method == "POST":
        name = request.POST.get('name', 'Anonymous User')
        context = {'name': name}
        return render(request, 'xss_l2.html', context)
    else:
        return "Error"

```

همانطور که در کد مشخص است ابتدا صفحه‌ی XSS رندر می‌شود و وقتی وارد چالش اول یا دوم می‌شویم وارد url آن چالش می‌شویم که توابع l1 و l2 برای آن‌ها است.
وقتی وارد آن می‌شویم اولین بار با متد GET وارد می‌شویم که نام ما برابر Anonymous User است. ولی اگر با متد POST باشد نام وارد شده را در یک دیکشنری قرار می‌دهد و صفحه را render می‌کند.
حال در ادامه به کد html دو چالش می‌پردازیم.

views.py 2 | xss_l1.html | urls.py 1

C: > Users > alireza > Desktop > HW4 > vuln web > xss > templates > xss_l1.html > ...

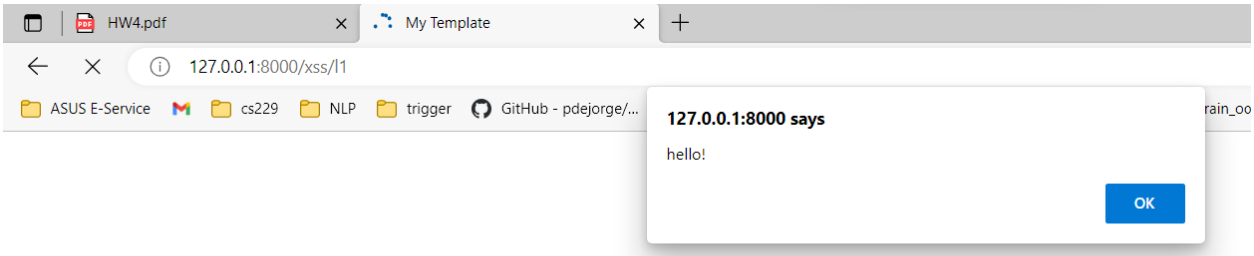
```
1   <!DOCTYPE html>
2   <html>
3   <head>
4       <title>My Template</title>
5   </head>
6   <body>
7       {% autoescape off %}
8       <h1>Hello, {{ name }}!</h1>
9       {% endautoescape %}
10      <form method="POST" action="" >
11          {% csrf_token %}
12          <label for="name">Enter your name:</label>
13          <input type="text" id="name" name="name">
14          <button type="submit">Submit</button>
15      </form>
16      <br>
17      <a href="{% url 'l2' %}">Next XSS Challenge</a>
18  </body>
19  </html>
```

در کد html چالش اول از name در میان یک تگ 1h استفاده شده‌است. می‌توانیم با ورودی دادن یک کد javascript کد خود را اجرا کنیم و یک alert در کد قرار دهیم.

برای چالش اول از ورودی زیر استفاده می‌کنیم.

```
<script>alert("hello!")</script>
```

می‌توانید نتیجه‌ی این چالش را در تصویر زیر مشاهده کنید.



در ادامه کد html چالش دوم را مشاهده می‌کنید.

xss_l1.html | xss_l2.html

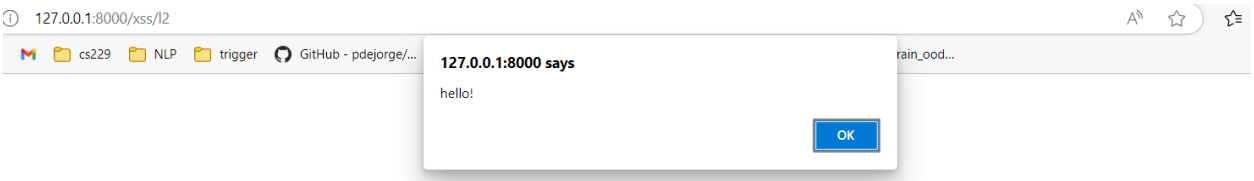
C: > Users > alireza > Desktop > HW4 > vuln web > xss > templates > xss_l2.html > ...

```
<!DOCTYPE html>
<html>
<head>
    <title>My Template</title>
</head>
<body>
    {% autoescape off %}
    <h1 id="header">Level 2: Hello, <script>
        var myText = "Level 2: Hello, {{ name }}!";
        var myParagraph = document.getElementById("header");
        myParagraph.innerHTML = myText;
      </script>
  </h1>
    {% endautoescape %}
    <form method="POST" action="" >
        {% csrf_token %}
        <label for="name">Enter your name:</label>
        <input type="text" id="name" name="name">
        <button type="submit">Submit</button>
    </form>
    <br>
    <a href="{% url 'l1' %}">Previous XSS Challenge</a>
</body>
</html>
```

در اینجا در تعریف متغیر از name استفاده شده‌است. برای حمله به این بخش ابتدا رشته را پایان می‌دهیم و سپس کد html خود را در ادامه می‌گذاریم. برای حمله از کد زیر استفاده می‌کنیم.

"; </script><img src=1 onerror=alert("hello!")>

نتیجه‌ی این حمله را در تصویر زیر مشاهده می‌کنید.



برای چالش SQL ابتدا به کد مراجعه می‌کنیم.

C: > Users > alireza > Desktop > HW4 > vuln web > sqli > views.py > ...

```python
from django.shortcuts import render
from django.db import connection
from django.db.models.expressions import RawSQL
from .models import Student

# Create your views here.


def index(request):
    if request.method == "GET":
        context = {'results': ''}
        return render(request, 'sqli.html', context)
    elif request.method == "POST":
        username = request.POST.get('username', '')
        password = request.POST.get('password', '')
        cursor = connection.cursor()
        print("SELECT * FROM sqli_student WHERE username = '" + username +"' AND password = '" + password + "'")
        cursor.execute("SELECT * FROM sqli_student WHERE username = '" + username +"' AND password = '" + password + "'")
        results = cursor.fetchone()
        return render(request, 'sqli.html', {'results': results})
    else:
        return "Error"
```

برای حمله کافی است در password یک شرط جدید اضافه کنیم.

username=attack

password=attack’ OR 1=1 or ‘

Windows PowerShell

```
PS C:\Users\alireza\Desktop\HW4\vuln web> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
June 16, 2023 - 18:06:02
Django version 4.2, using settings 'vuln.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

[16/Jun/2023 18:06:05] "GET /sqli/ HTTP/1.1" 200 675
SELECT * FROM sqli_student WHERE username = 'attack' AND password = 'attack' or 1=1 or ''
[16/Jun/2023 18:06:23] "POST /sqli/ HTTP/1.1" 200 682
```

نتیجه‌ی اجرای حمله

Enter your username and password to log in

Enter your username:

Enter your password:

Submit

Login Successful

برای جلوگیری از حمله‌های XSS می‌توان از escape استفاده کرد که به این معنی است که شما کاراکترهای کلیدی داده را تبدیل می کنید (یا علامت گذاری می کنید) تا از تفسیر آن در یک زمینه خطرناک جلوگیری کنید.

برای این منظور کدهای html دو چالش را به صورت زیر تغییر می‌دهیم.

C: > Users > alireza > Desktop > HW4 > vuln web > xss > templates > xss_l1.html > html > body > h1

```
<!DOCTYPE html>
<html>
<head>
    <title>My Template</title>
</head>
<body>
    {% autoescape off %}
    <h1>Hello, {{ name | escape }}!</h1>
    {% endautoescape %}
    <form method="POST" action="" >
        {% csrf_token %}
        <label for="name">Enter your name:</label>
        <input type="text" id="name" name="name">
        <button type="submit">Submit</button>
    </form>
    <br>
    <a href="{% url 'l2' %}">Next XSS Challenge</a>
</body>
</html>
```

xss_l1.html | xss_l2.html

C: > Users > alireza > Desktop > HW4 > vuln web > xss > templates > xss_l2.html > html

```
<!DOCTYPE html>
<html>
<head>
    <title>My Template</title>
</head>
<body>
    {% autoescape off %}
    <h1 id="header">Level 2: Hello, <script>
        var myText = "Level 2: Hello, {{ name | escape }}!";
        var myParagraph = document.getElementById("header");
        myParagraph.innerHTML = myText;
    </script>
    </h1>
    {% endautoescape %}
    <form method="POST" action="" >
        {% csrf_token %}
        <label for="name">Enter your name:</label>
        <input type="text" id="name" name="name">
        <button type="submit">Submit</button>
    </form>
    <br>
    <a href="{% url 'l1' %}">Previous XSS Challenge</a>
</body>
</html>
```

برای جلوگیری از SQL Injection از توابع خود Django استفاده می‌کنیم و کد ما به صورت زیر می‌شود.

C: > Users > alireza > Desktop > HW4 > vuln web > sqli > views.py > index

```python
from django.shortcuts import render
from django.db import connection
from django.db.models.expressions import RawSQL
from .models import Student

# Create your views here.


def index(request):
    if request.method == "GET":
        context = {'results': ''}
        return render(request, 'sqli.html', context)
    elif request.method == "POST":
        username = request.POST.get('username', '')
        password = request.POST.get('password', '')
        try:
            student = Student.objects.get(username=username, password=password)
            results = True
        except Student.DoesNotExist:
            results = False

        return render(request, 'sqli.html', {'results': results})
    else:
        return "Error"
```

توجه کنید دیگر Query به صورت مستقیم زده نمی‌شود و Django خودش کوئری را هندل می‌کند. اگر نیاز بود Query بزنیم می‌توانستیم از .raw() برای کلاس مورد نظر استفاده کنیم.

منابع

https://portswigger.net/web-security/cross-site-scripting.

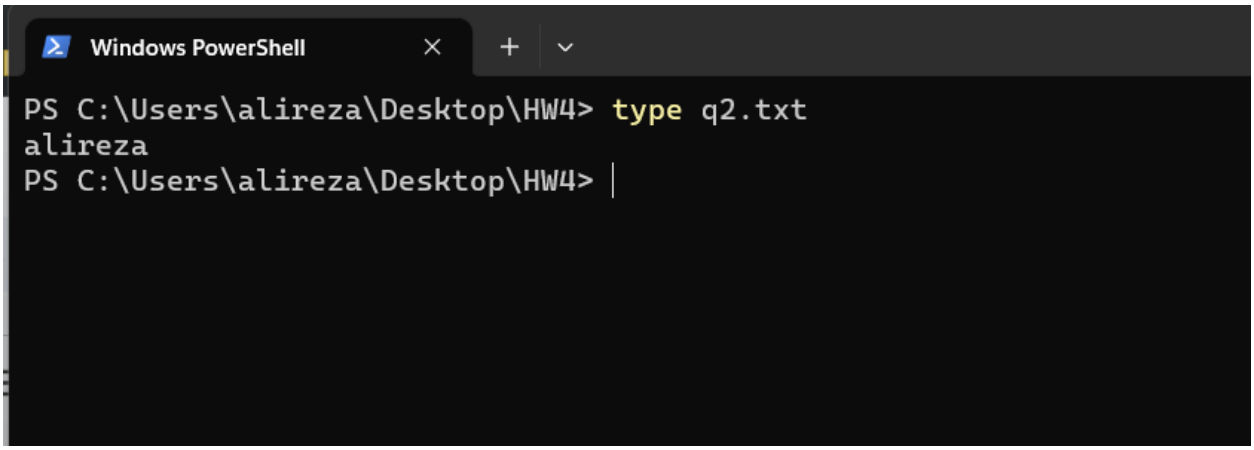
https://portswigger.net/web-security/cross-site-scripting/preventing

https://stackoverflow.com/questions/2159724/how-can-escaping-be-used-to-prevent-xss-attacks

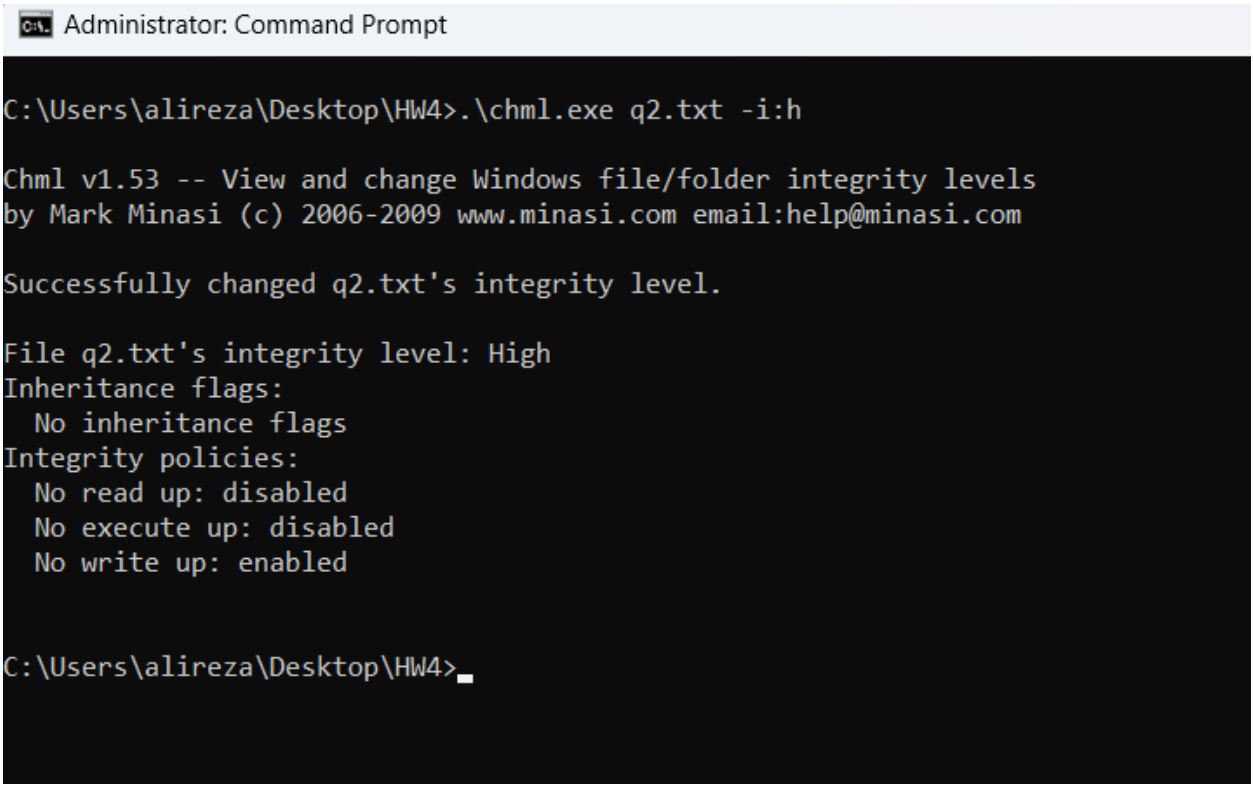
https://www.stackhawk.com/blog/sql-injection-prevention-django/

سوال دوم

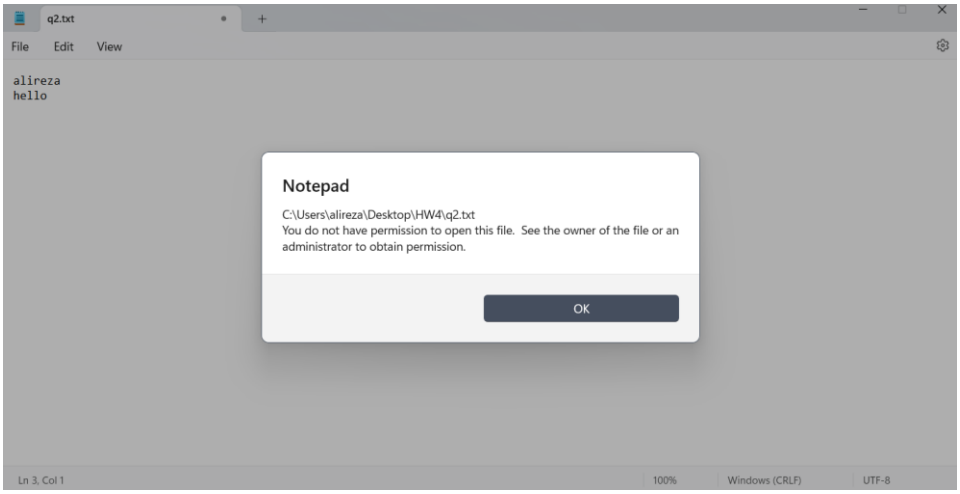
1)یک فایل با نام q2.txt درست کرده و نام خود را در آن نوشته‌ام.



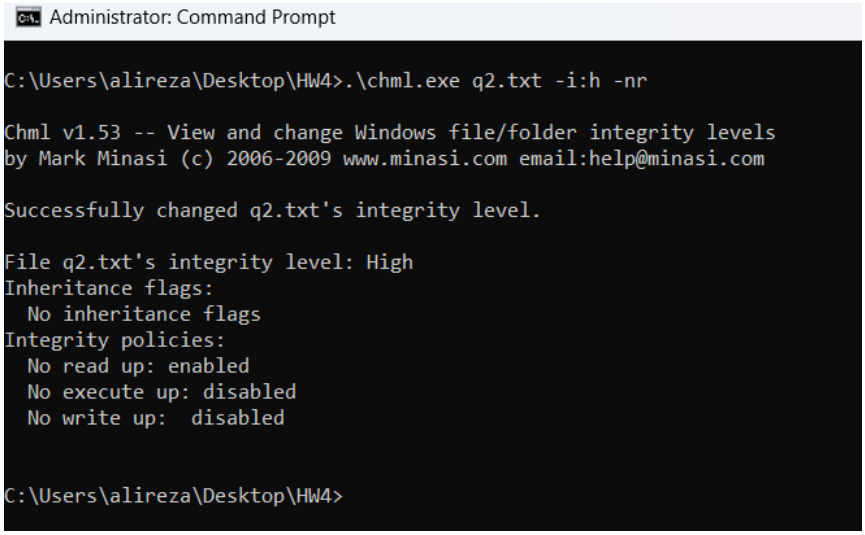
2)حال Integrity Level آن را به high تغییر می‌دهیم.

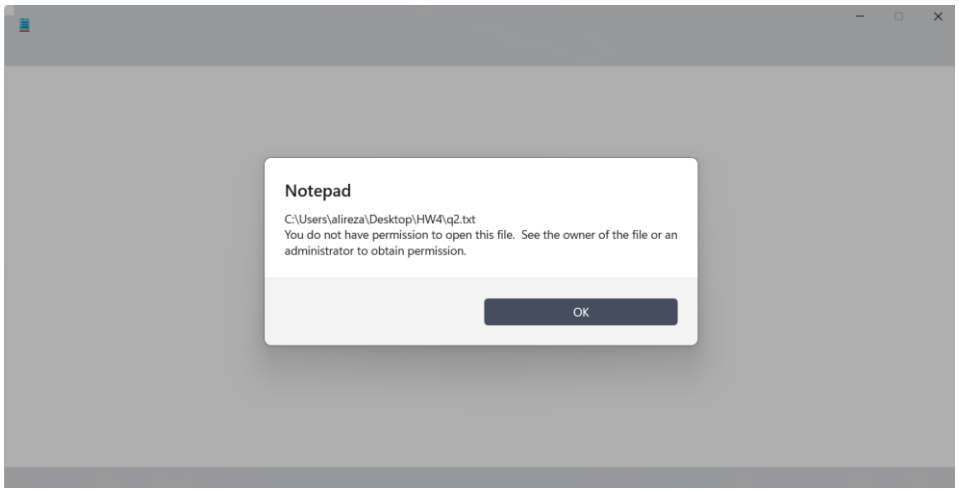


برای خواندن مشکلی ایجاد نمی‌شود و فایل باز می‌شود ولی هنگام نوشتن و ذخیره کردن با خطای زیر رو به رو می‌شویم. علت آن نیز این است که هنگامی که integrity را به high تغییر دادیم همانطور که در تصویر بالا نیز مشخص است سیاست No write up فعال یا enable می‌شود ولی سایر سیاست‌ها تغییر نمی‌کند. کاربر عادی در ویندوز دارای سطح medium است و نمی‌تواند در فایلی که high است و سیاست ننوشتن در آن فعال است چیزی بنویسد.



3) اکنون سیاست No read up را فعال می‌کنیم و نتایج را در تصاویر زیر مشاهده می‌کنید.





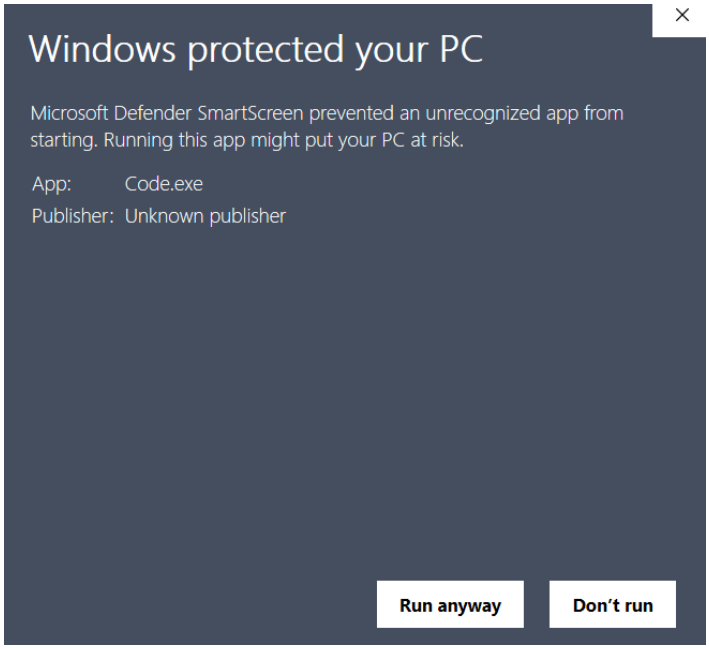
4) کد به زبان C را در تصویر زیر مشاهده می‌کنید.

```c
int main() {
    FILE *ptr;
    char ch;
    ptr = fopen("q2.txt", "r");
    if (NULL == ptr) {
        printf("can not open the file to read!\n");
    }
    if (ptr != NULL) {
        printf("content of the file is:\n");
        while ((ch = fgetc(ptr)) != EOF) {
            printf("%c", ch);
        }
        printf("\nend of the file\n");
        fclose(ptr);
    }
    ptr = fopen("q2.txt", "w");
    if (ptr == NULL) {
        printf("can not open the file to write!\n");
    }
    if (ptr != NULL) {
        fprintf(ptr, "%s", "modified!");
        fclose(ptr);
        printf("write successfully to the file\n");
    }
    ch = getchar();
    return 0;
}
```

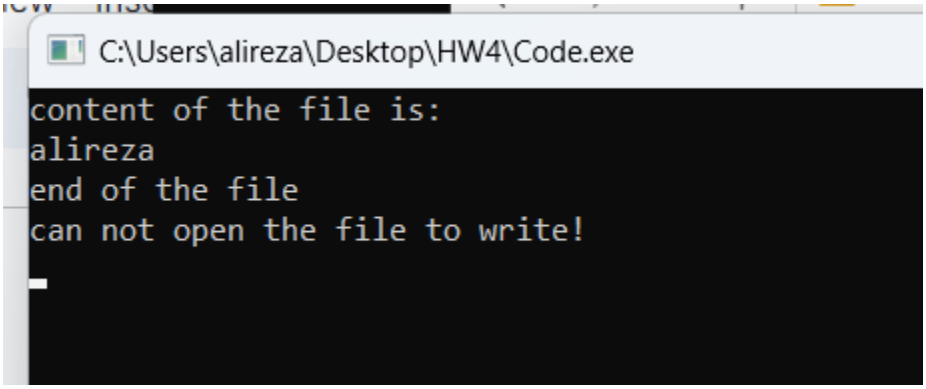
5) ابتدا سطح integrity دو فایل را تغییر می‌دهیم و در شکل زیر مشاهده می‌کنید.

```
Select Administrator: Command Prompt

C:\Users\alireza\Desktop\HW4>.\chml.exe q2.txt -i:m

Chml v1.53 -- View and change Windows file/folder integrity levels
by Mark Minasi (c) 2006-2009 www.minasi.com email:help@minasi.com

Successfully changed q2.txt's integrity level.

File q2.txt's integrity level: Medium
Inheritance flags:
  No inheritance flags
Integrity policies:
  No read up: disabled
  No execute up: disabled
  No write up: enabled


C:\Users\alireza\Desktop\HW4>.\chml.exe Code.exe -i:l

Chml v1.53 -- View and change Windows file/folder integrity levels
by Mark Minasi (c) 2006-2009 www.minasi.com email:help@minasi.com

Successfully changed Code.exe's integrity level.

File Code.exe's integrity level: Low
Inheritance flags:
  No inheritance flags
Integrity policies:
  No read up: disabled
  No execute up: disabled
  No write up: enabled
```

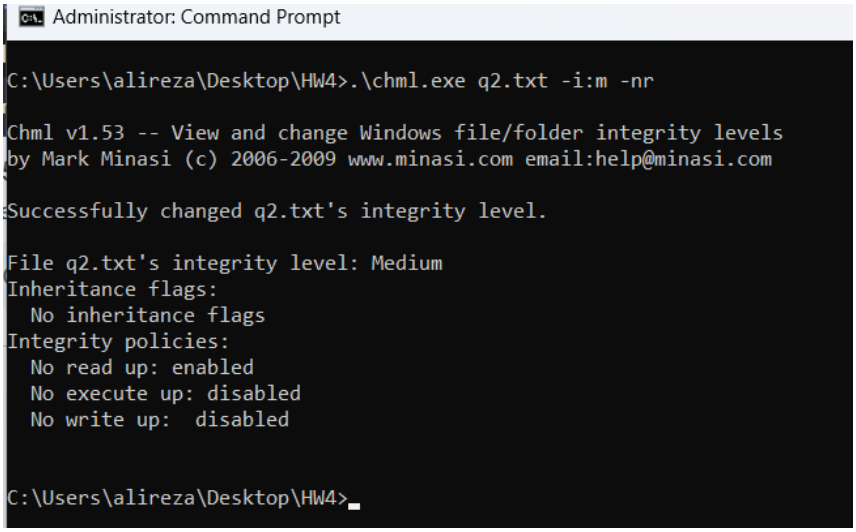
حال فایل exe را اجرا می‌کنیم.



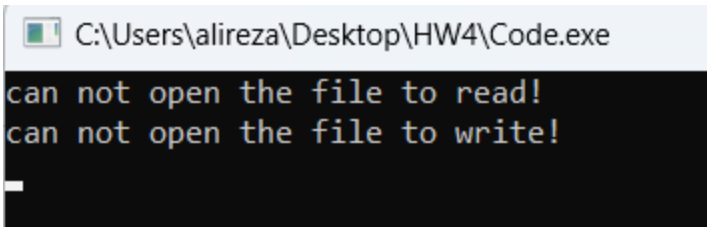
و نتیجه‌ی اجرا را مشاهده می‌کنید.



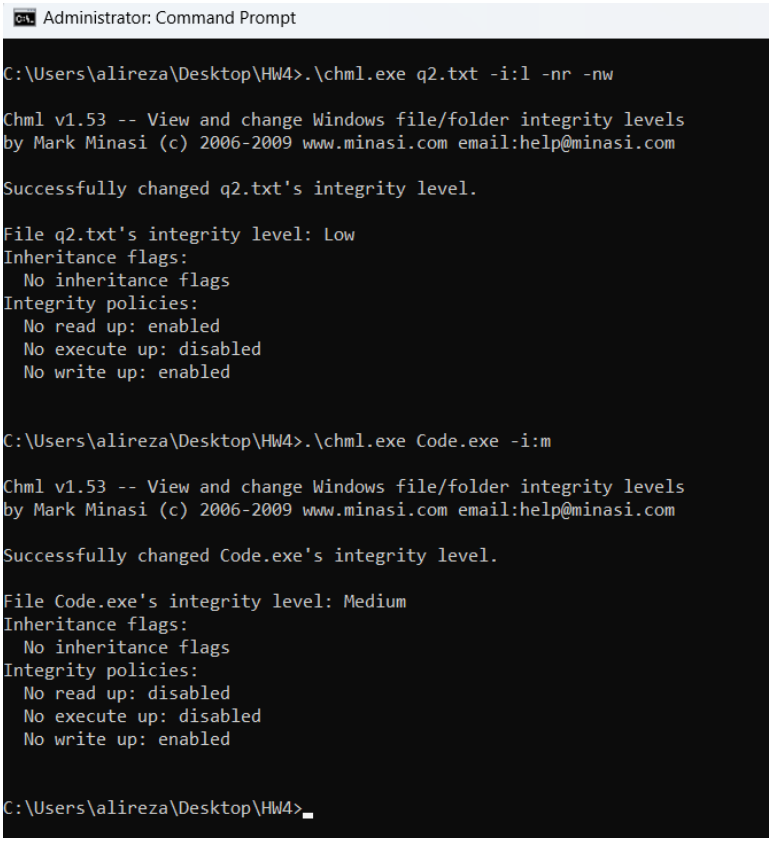
حال No read up را فعال می‌کنیم.



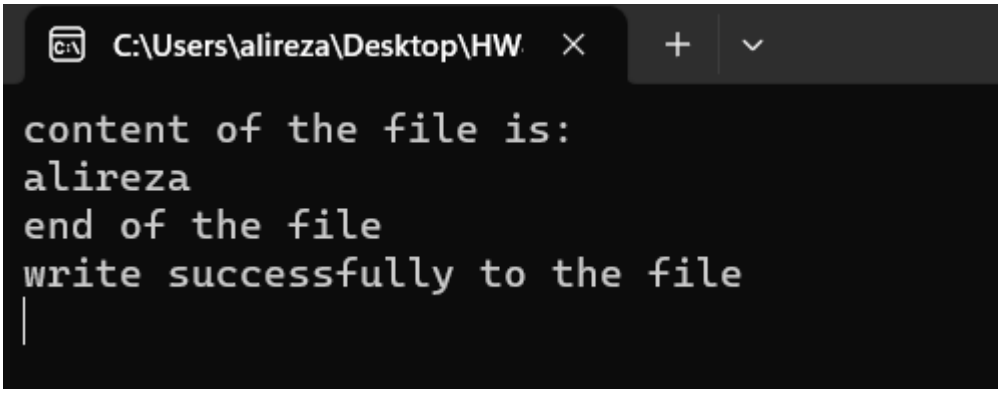
حال نتیجه‌ی اجرای فایل اجرایی را مشاهده می‌کنید که دیگر اجازه‌ی خواندن نیز ندارد.



6) برای این بخش ابتدا integrity فایل اجرایی برابر medium و فایل متنی را برابر low قرار می‌دهیم.



همانطور که مشخص است هم نوشتن هم خواندن بدون مشکل انجام شده است.



سوال سوم

ابتدا جداول گفته شده را با دستورات زیر می‌سازیم.

Query Query History

```
1  CREATE TABLE PHYSICIANS (
2      name VARCHAR(64),
3      specialist VARCHAR(64),
4      phone_number VARCHAR(64),
5      national_id VARCHAR(10),
6      PRIMARY KEY (national_id)
7  );
```

Query Query History

```
1  CREATE TABLE PATIENTS (
2      name VARCHAR(64),
3      sickness VARCHAR(64),
4      phone_number VARCHAR(64),
5      national_id VARCHAR(10),
6      PRIMARY KEY (national_id)
7  );
```

Query Query History

```
1  CREATE TABLE SICKNESS (
2      sickness VARCHAR(64),
3      national_id VARCHAR(10),
4      FOREIGN KEY (national_id) REFERENCES PATIENTS (national_id),
5      PRIMARY KEY (national_id, sickness)
6  );
```

Query Query History

```
1  CREATE TABLE RECOGNITIONS (
2      patient_name VARCHAR(64),
3      physician_name VARCHAR(64),
4      sickness VARCHAR(64)
5  );
```

سپس با دستور زیر تعدادی داده وارد جدول می‌کنیم.

```
Query  Query History
1  INSERT INTO PHYSICIANS (name, specialist, phone_number, national_id)
2  VALUES ('John Doe', 'Nephrologist', '5555555555', '1234567890');
3
4  INSERT INTO PATIENTS (name, sickness, phone_number, national_id)
5  VALUES ('Jane Smith', 'Diabetes', '1234567890', '1234567891');
6
7  INSERT INTO SICKNESS (sickness, national_id)
8  VALUES ('Diabetes', '1234567891');
9
10 INSERT INTO RECOGNITIONS (patient_name, physician_name, sickness)
11 VALUES ('Jane Smith', 'John Doe', 'Diabetes');
```

حال با دستورات زیر خواسته‌ی سوال را ایجاد می‌کنیم.

```
Query  Query History
1  CREATE ROLE Normal_User;
2  GRANT SELECT(name, specialist) ON TABLE PHYSICIANS TO Normal_User;
3  CREATE USER normaluser with password '98101555';
4  GRANT Normal_User to normaluser;
5
6  CREATE ROLE Nurse;
7  GRANT ALL PRIVILEGES ON TABLE PATIENTS, SICKNESS, RECOGNITIONS TO Nurse;
8  GRANT SELECT ON TABLE PHYSICIANS TO Nurse;
9  CREATE USER nurse1 with password '98101555';
10 GRANT Nurse to nurse1;
11
12 CREATE ROLE Physician;
13 GRANT SELECT, UPDATE ON TABLE PATIENTS, SICKNESS TO Physician;
14 CREATE USER physician1 with password '98101555';
15 GRANT Physician to physician1;
16
```

حال باید با user های ساخته شده وارد شویم و تست کنیم.

- بررسی Normal User

در شکل زیر مشاهده می‌کنید که دسترسی به نام دکترها را دارد.

```
Query  Query History
1  select name from physicians;
```

Data Output Messages Notifications

	name character varying (64)
1	John Doe

اما نمی‌تواند کل جدول را بخواند.

Query Query History

```
1  select * from physicians;
```

Data Output Messages Notifications

```
ERROR:  permission denied for table physicians

SQL state: 42501
```

- بررسی Physician

پزشک می‌تواند از جدول بیماران بخواند.

Query Query History

```
1  select * from patients;
```

Data Output Messages Notifications

	name character varying (64)	sickness character varying (64)	phone_number character varying (64)	national_id [PK] character varying (10)
1	Jane Smith	Diabetes	1234567890	1234567891

توانایی آپدیت روی جدول را نیز دارد.

Query Query History

```
1  update patients set phone_number = '4444' where national_id = '1234567891';
```

Data Output Messages Notifications

```
UPDATE 1

Query returned successfully in 518 msec.
```

اما امکان insert در جدول را ندارد. توجه کنید برای جدول بیماری‌ها نیز به همین صورت است.

Query Query History

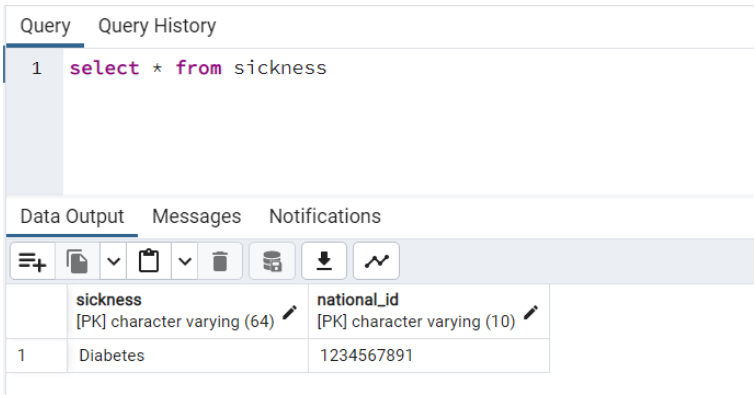
```
1  INSERT INTO PATIENTS (name, sickness, phone_number,national_id)
2  VALUES ('alireza', 'flu', '0142', '1234567899');
```

Data Output Messages Notifications

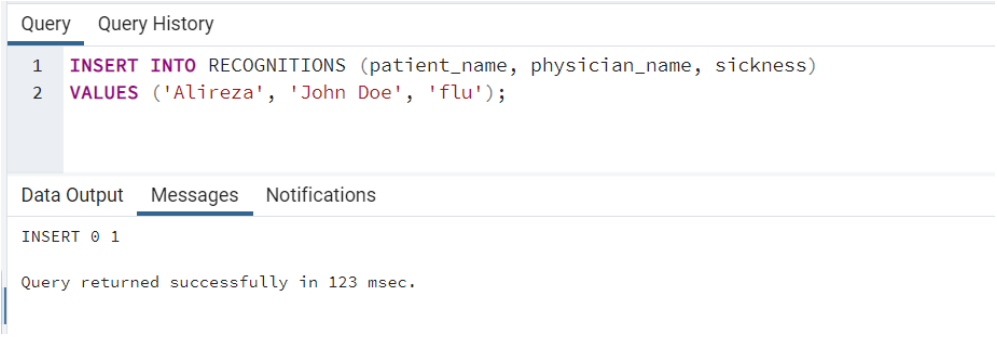
```
ERROR:  permission denied for table patients

SQL state: 42501
```

- بررسی Nurse

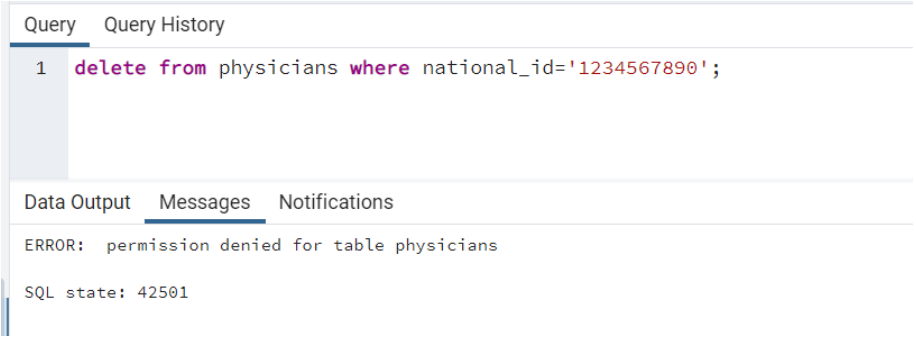
برای نمونه توانایی خواندن روی جدول بیماری را دارند.



همچنین امکان اضافه کردن داده به جدول تشخصیات را نیز دارند.



ولی برای مثال امکان حذف از جدول پزشکان ندارند.



منابع

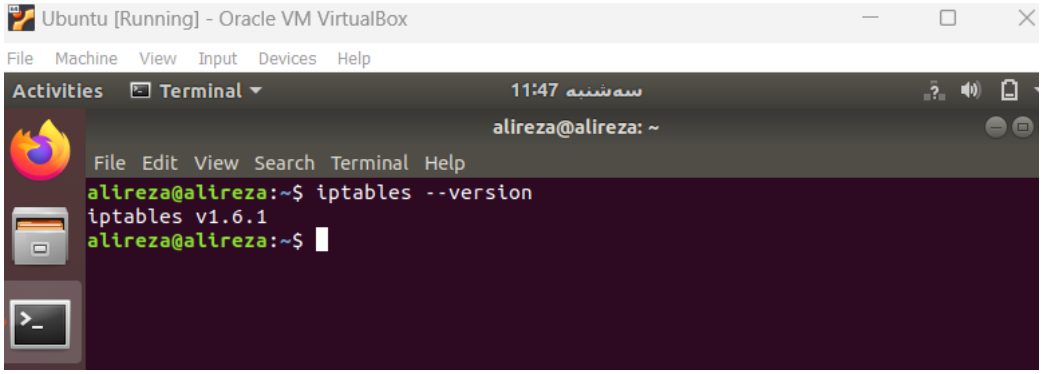
https://tableplus.com/blog/2018/04/postgresql-how-to-grant-access-to-users.html

https://dba.stackexchange.com/questions/151674/how-to-change-username-to-connect-to-server-in-pgadmin4-in-query-tool

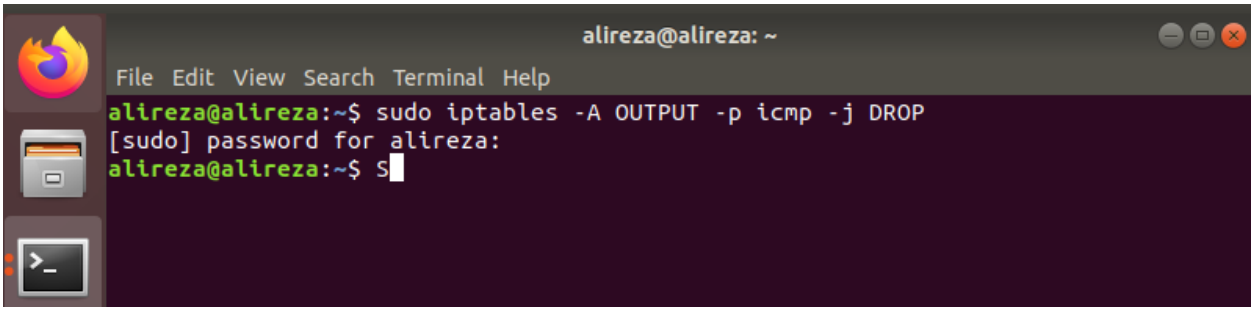
سوال چهارم

بخش اول

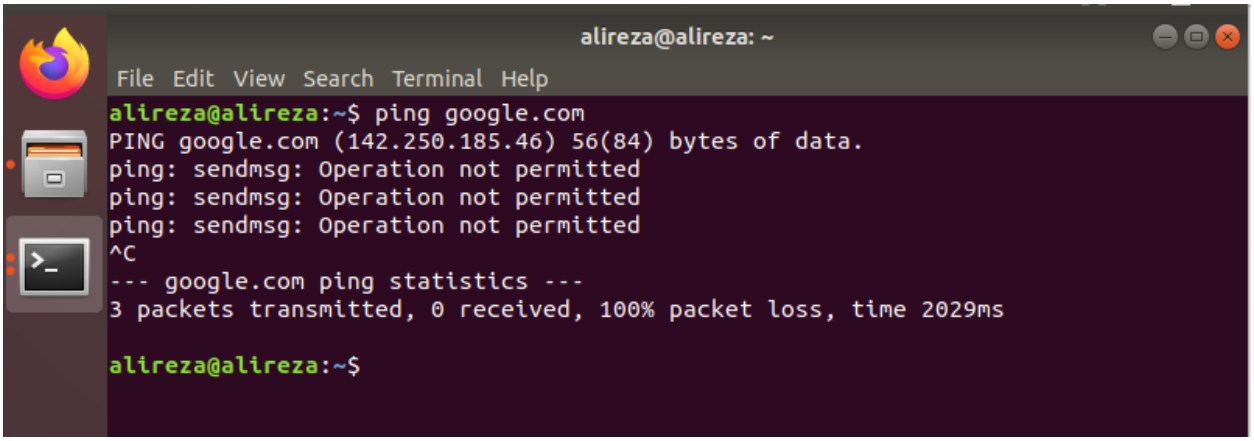
در ابتدا ورژن iptables را در ماشین مجازی مشاهده می‌کنیم.



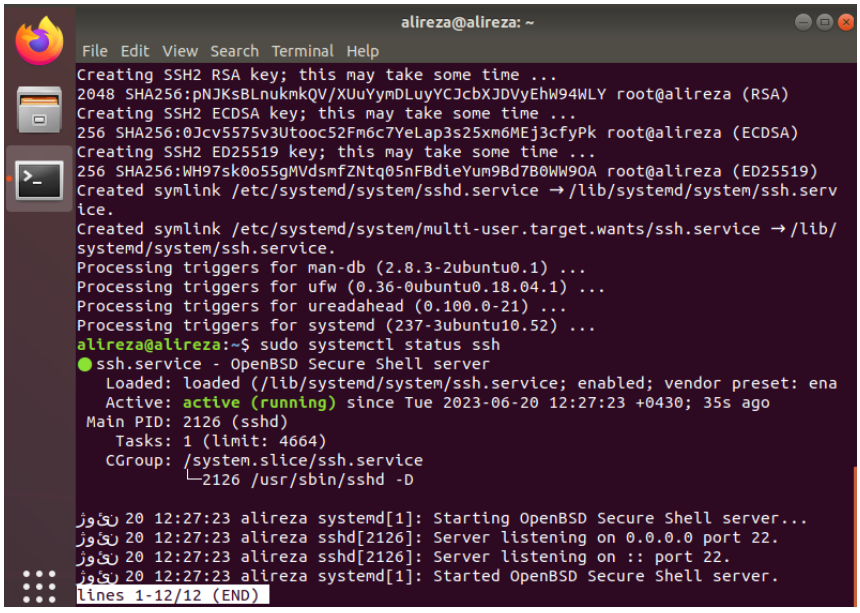
- برای این بخش برای تست از دستور ping استفاده می‌کنیم که این دستور بسته‌های ICMP ارسال می‌کند. سپس دستور زیر را می‌زنیم:



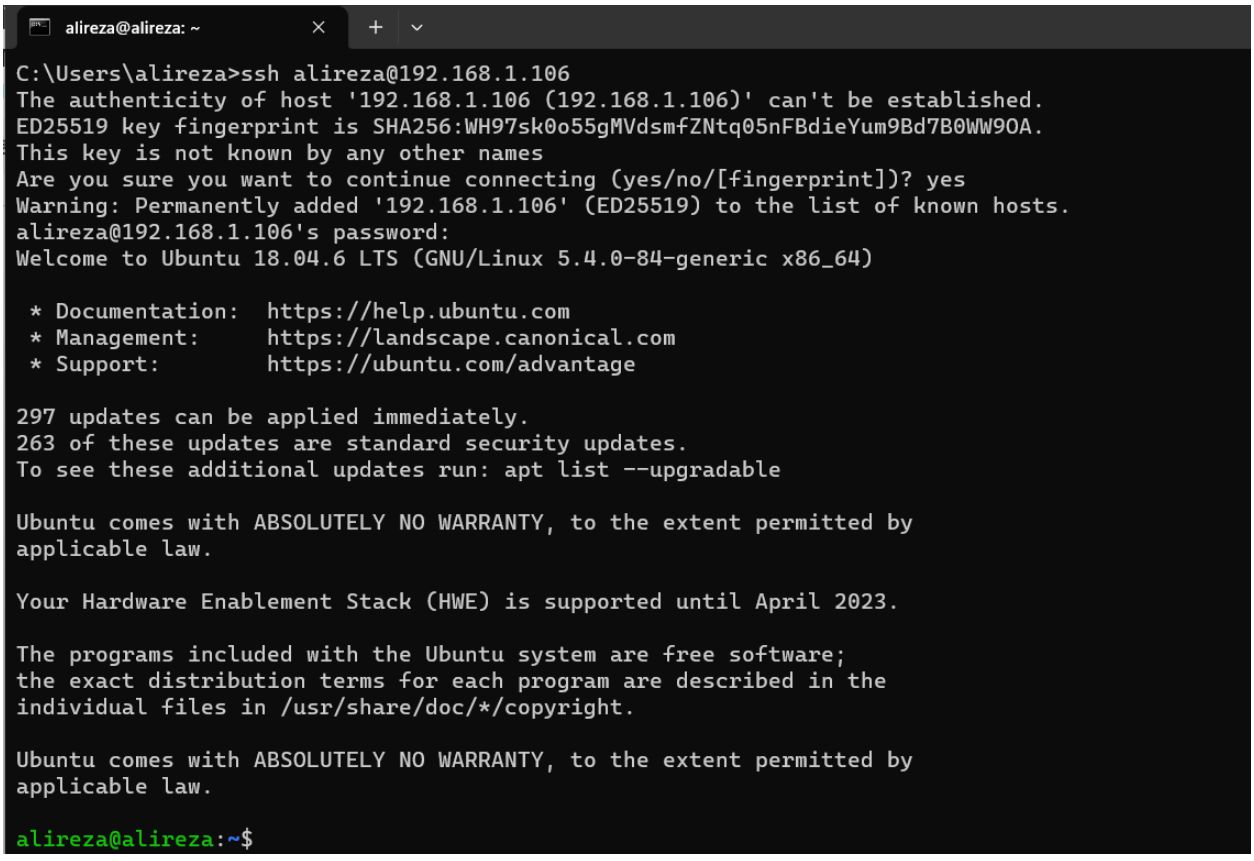
سپس google را ping می‌کنیم و پیغام خطای زیر را دریافت می‌کنیم.



- برای این کار بایست از یک سیستم دیگر به ماشین مجازی ssh بزنیم. برای این کار openssh-server را نصب می‌کنیم و وضعیت آن را با دستور زیر مشاهده می‌کنیم.

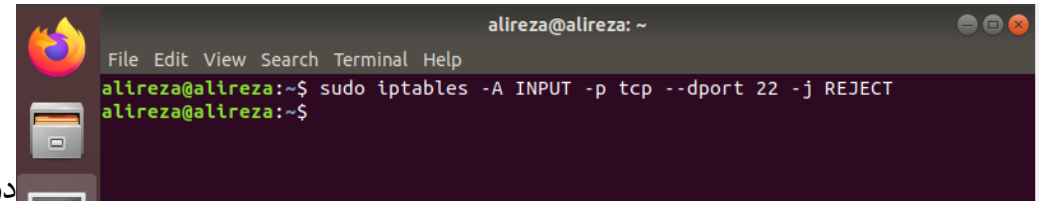


حال از دستگاه host به ماشین مجازی ssh می‌زنیم و نتیجه‌ی آن به شرح زیر است.



حال دستور زیر را در Iptables می‌زنیم که ترافیک پروتکل tcp به پورت ورودی ۲۲ را reject کند.

دوباره

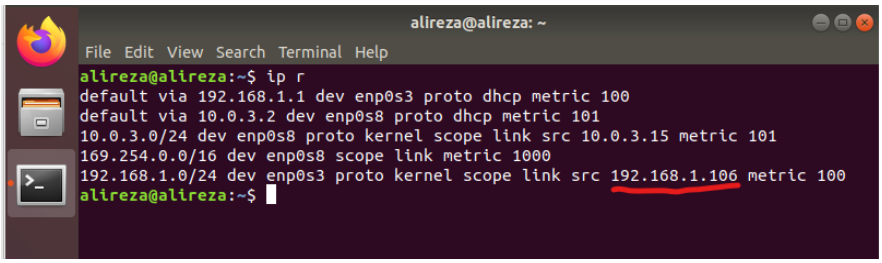


تلاش می‌کنیم تا ssh بزنیم ولی نتیجه به شرح زیر است.

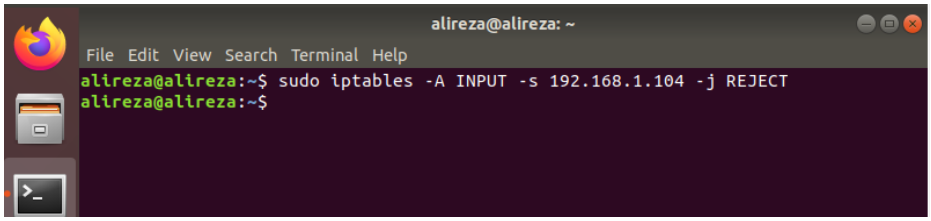
```
C:\Users\alireza>ssh alireza@192.168.1.106
ssh: connect to host 192.168.1.106 port 22: Connection timed out

C:\Users\alireza>
```

- ابتدا ip دستگاه host و ماشین مجازی را بدست می‌آوریم

```
Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::3120:e8b9:7b1b:30d7%20
   IPv4 Address. . . . . . . . . . . : 192.168.1.104
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1

C:\Users\alireza>
```



سپس با دستو زیر ترافیک ورودی از این ip را ریجکت می‌کنیم.



نتیجه‌ی Ping کردن ماشین مجازی به شرح زیر است.

```
C:\Users\alireza>ping 192.168.1.106

Pinging 192.168.1.106 with 32 bytes of data:
Reply from 192.168.1.106: Destination port unreachable.
Reply from 192.168.1.106: Destination port unreachable.
Reply from 192.168.1.106: Destination port unreachable.
Reply from 192.168.1.106: Destination port unreachable.

Ping statistics for 192.168.1.106:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

C:\Users\alireza>
```

بخش دوم

- iptables -A INPUT -p tcp –dport 22 ! -s 192.168.1.0/24 -j ACCEPT
- iptables -A INPUT -m state ! –state ESTABLISHED -j REJECT
- iptables -t nat -I PREROUTING -p tcp --dport 80 -j DNAT --to-destination <ip:port>
- iptables -t nat -I PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 8080
- برای این بخش سه rule به صورت زیر تعریف می‌کنیم. در قانون اول اگر تعداد connection برای یک IP مشخص بیشتر از ۱۱۱ باشد بسته REJECT می‌شود. در ادامه برای دنبال کردن اتصالات از conntrack استفاده می‌کنیم که اتصالات را ردیابی می‌کند. این ردیابی معمولاً به صورت یک جدول بزرگ با حداقل 6 ستون اجرا می شود: پروتکل IP مبدا، پورت مبدا، IP مقصد، پورت مقصد و وضعیت اتصال. در قانون دوم چک می‌شود تعداد connection های جدید حداکثر ۲۰ تا باشد و تعداد درخواست‌های ارتباط جدید باید کمتر از ۶۰ تا در ثانیه باشد. اگر این شرایط برقرار بود accept می‌شود. در غیر این صورت بسته drop می‌شود.

  iptables -A INPUT -p tcp -m connlimit --connlimit-above 111 -j REJECT

  iptables -A INPUT -p tcp -m conntrack --ctstate NEW -m limit --limit 60/s --limit-burst 20 -j ACCEPT

  iptables -A INPUT -p tcp -m conntrack --ctstate NEW -j DROP

منابع

https://averagelinuxuser.com/ssh-into-virtualbox/
https://gist.github.com/mattia-beta/bd5b1c68e3d51db933181d8a3dc0ba64
https://superuser.com/questions/1071656/whats-the-difference-between-iptables-state-and-ctstate