BÀI TẬP THỰC HÀNH TUẦN 20/2/2023

Môn học: Kỹ thuật lập trình

Bài 1: Thu thập thông tin với Shodan

Bài 1.1:

Lấy thông tin về một địa chỉ IP cụ thể, chẳng hạn như máy chủ DNS và vị trí địa lý,…

Cần đăng ký tài khoản trên developer.shodan.io để lấy API key cho từng tài khoản

Tham khảo:

```python
#!/usr/bin/env python
import requests
import os
SHODAN_API_KEY = os.environ['SHODAN_API_KEY']
ip = '1.1.1.1'
def ShodanInfo(ip):
    try:
        result = requests.get(f"https://api.shodan.io/shodan/host/{ip}?key={SHODAN_API_KEY}&minify=True").json()
    except Exception as exception:
        result = {"error":"Information not available"}
    return result
print(ShodanInfo(ip))
```

```
{'region_code': None, 'tags': [], 'ip': 16843009, 'area_code':
None, 'domains': ['one.one'], 'hostnames': ['one.one.one.one'],
'postal_code': None, 'dma_code': None, 'country_code': 'AU',
'org': 'Cloudflare', 'data': [], 'asn': 'AS13335', 'city':
None, 'latitude': -33.494, 'isp': 'CRISLINE', 'longitude':
143.2104, 'last_update': '2020-06-25T15:29:34.542351',
'country_code3': None, 'country_name': 'Australia', 'ip_str':
'1.1.1.1', 'os': None, 'ports': [53]}
```

Bài 1.2:

Tìm danh sách địa chỉ IP cho các máy chủ FTP đăng nhập ở chế độ Anonymous.

Tham khảo:

```python
#!/usr/bin/env python
import shodan
import re
import os
servers =[]
SHODAN_API_KEY = os.environ['SHODAN_API_KEY']
shodanApi = shodan.Shodan(shodanKeyString)
results = shodanApi.search("port: 21 Anonymous user logged in")
print("hosts number: " + str(len( results['matches'])))
for result in results['matches']:
    if result['ip_str'] is not None:
        servers.append(result['ip_str'])
for server in servers:
    print(server)
```

```
In [16]: runfile('D:/Google D
hosts number: 100
70.40.210.79
153.127.37.14
101.98.62.139
50.87.180.25
162.241.245.46
162.144.214.131
192.185.133.85
144.202.0.37
158.69.166.71
134.119.56.145
67.20.80.193
209.59.144.69
67.20.80.149
51.75.186.62
69.25.107.19
```

Bài 2: Tạo một máy khách DNS bằng Python và xem cách máy khách này lấy thông tin về máy chủ định danh, máy chủ thư và địa chỉ IPV4/IPV6.

Tham khảo:

```
import dns.resolver
hosts = ["oreilly.com", "yahoo.com", "google.com", "microsoft.
com", "cnn.com"]
for host in hosts:
    print(host)
    ip = dns.resolver.query(host, "A")
    for i in ip:
        print(i)
```

```
$ python3 dns_resolver.py
oreilly.com
199.27.145.65
199.27.145.64
yahoo.com
98.137.246.8
72.30.35.9
98.137.246.7
72.30.35.10
98.138.219.232
98.138.219.23
...
```

Bài 3: Tìm kiếm các địa chỉ dễ bị tổn thương trong máy chủ với fuzzing

Lý thuyết:

Tiến trình fuzzing

Fuzzer là một chương trình có một tệp chứa các URL dùng để dự đoán cho một ứng dụng hoặc máy chủ cụ thể. Chúng ta tạo request cho từng URL dự đoán và nếu chúng ta nhận được response thành công => tìm thấy một URL không công khai hoặc bị ẩn.

Tiến trình fuzzing bao gồm các giai đoạn sau:

1. Xác định mục tiêu: Để fuzz một ứng dụng, chúng ta phải xác định ứng dụng mục tiêu.
2. Xác định đầu vào: Lỗ hổng tồn tại do ứng dụng đích chấp nhận đầu vào không đúng định dạng và xử lý nó không đúng cách.
3. Tạo dữ liệu fuzz: Sau khi nhận được tất cả các tham số đầu vào, chúng ta phải tạo dữ liệu đầu vào không hợp lệ để gửi đến ứng dụng đích.
4. Fuzzing: Sau khi tạo dữ liệu fuzz, chúng ta phải gửi nó đến ứng dụng đích. Chúng ta có thể sử dụng dữ liệu fuzz để theo dõi các trường hợp ngoại lệ khi gọi dịch vụ.
5. Xác định khả năng khai thác: Sau khi fuzzing, chúng ta phải kiểm tra đầu vào gây ra sự cố.

FuzzDB là một dự án cung cấp tài nguyên để kiểm tra các lỗ hổng trong máy chủ và ứng dụng web. FuzzDB bao gồm một tập hợp các thư mục chứa các kiểu tấn công đã biết được thu thập trong nhiều thử nghiệm:

https://github.com/fuzzdb-project/fuzzdb

Các danh mục FuzzDB được tách thành các thư mục khác nhau chứa các mẫu tài nguyên có thể dự đoán, nghĩa là các mẫu để phát hiện các lỗ hổng với payload độc hại hoặc các tuyến (routes) dễ bị tấn công

Một trong các bài tập là sử dụng nó để hỗ trợ xác định các lỗ hổng trong các ứng dụng web thông qua các phương pháp brute-force.

Bài tập:

Bài 3.1: Tìm kiếm trang login: với một URL (domain) đang phân tích, thực hiện kiểm tra kết nối cho từng mục (domain + predictable URL) và nếu request trả về mã 200, thì điều đó có nghĩa là trang login đã được tìm thấy trong máy chủ.

predictable URL trong fuzzdb:

https://github.com/fuzzdb-project/fuzzdb/blob/master/discovery/predictable-filepaths/login-file-locations/Logins.txt

Tham khảo:

```python
#!/usr/bin/env python
import requests
logins = []
with open('Logins.txt', 'r') as filehandle:
    for line in filehandle:
        login = line[:-1]
        logins.append(login)
domain = "http://testphp.vulnweb.com"
for login in logins:
    print("Checking... "+ domain + login)
    response = requests.get(domain + login)
    if response.status_code == 200:
        print("Login resource detected: " +login)
```

Bài 3.2: Tìm lỗi SQL injection đối với một trang web.

Theo cách tương tự, sử dụng dự án FuzzDB để xây dựng một tập lệnh nhằm kiểm tra lỗi SQL-injection.

Sử dụng một tệp cung cấp danh sách các chuỗi để kiểm tra tính dễ bị tổn thương:

https://github.com/fuzzdb-project/fuzzdb/tree/master/attack/sql-injection/detect

Tham khảo:

snippet code

```python
import requests

domain = "http://testphp.vulnweb.com/listproducts.php?cat="

mysql_attacks = []

# open file and read the content in a list
with open('MySQL.txt', 'r') as filehandle:
    for line in filehandle:
        attack = line[:-1]
        mysql_attacks.append(attack)
```

Bài 4: python nmap

Python cung cấp mô-đun python-nmap để thực hiện quét cổng bằng công cụ Nmap. Python-nmap là một công cụ được sử dụng rất nhiều trong phạm vi kiểm tra bảo mật hoặc kiểm tra xâm nhập. Ngoài ra, nó cho phép quản trị viên hệ thống hoặc nhà tư vấn bảo mật máy tính thực hiện tự động hóa các quy trình kiểm tra thâm nhập.

Tham khảo hướng dẫn khi thực hành bị lỗi:

1. Cài đặt nmap trên win
2. Bổ sung path trong biến môi trường (restart lại máy)
3. Cml: pip install python-nmap

Bài 4.1:

Kiểm tra các port với địa chỉ host xác định (VD: dantri.com.vn)

Tham khảo:

```
import nmap
portScanner = nmap.PortScanner()
host_scan = input('Host scan: ')
portlist="21,22,23,25,80"
portScanner.scan(hosts=host_scan, arguments='-n -p'+portlist)
print(portScanner.command_line())
hosts_list = [(x, portScanner[x]['status']['state']) for x in
portScanner.all_hosts()]
for host, status in hosts_list:
    print(host, status)
for protocol in portScanner[host].all_protocols():
    print('Protocol : %s' % protocol)
    listport = portScanner[host]['tcp'].keys()
    for port in listport:
        print('Port : %s State : %s' % (port,portScanner[host][protocol][port]['state']))
```

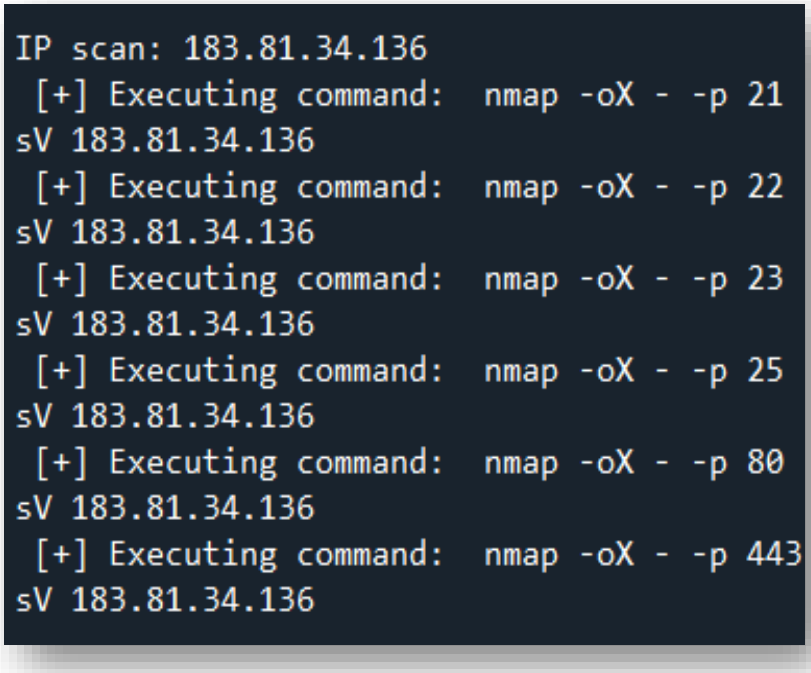
```
In [4]: runfile('D:/Google Drive/ml_waf/
untitled0.py', wdir='D:/Google Drive/ml_wa

Host scan: 183.81.34.136
nmap -oX - -n -p21,22,23,25,80 183.81.34.1
183.81.34.136 up
Protocol : tcp
Port : 21 State : closed
Port : 22 State : closed
Port : 23 State : closed
Port : 25 State : closed
Port : 80 State : open
```

Bài 4.2: Scan port ở chế độ đồng bộ (xem slide)

Tham khảo:

```python
import nmap
class NmapScanner:
    def __init__(self):
        self.portScanner = nmap.PortScanner()
    def nmapScan(self, ip_address, port):
        self.portScanner.scan(ip_address, port)
        print(" [+] Executing command: ", self.portScanner.command_line())
def main():
    ip_address = input('IP scan: ')
    ports = ["21","22","23","25","80","443"]
    for port in ports:
        NmapScanner().nmapScan(ip_address, port)
if __name__ == "__main__":
    main()
```



Bài 4.3: Scan port ở chế độ không đồng bộ.

Tham khảo:

```
class PortScannerAsync(object):
    """
    PortScannerAsync allows to use nmap from python asynchronously
    for each host scanned, callback is called with scan result for the host

    """
    def __init__(self):
        """
        Initialize the module

        * detects nmap on the system and nmap version
        * may raise PortScannerError exception if nmap is not found in the path

        """
        self._process = None
        self._nm = PortScanner()
        return

    def __del__(self):
        """
        Cleanup when deleted

        """
        if self._process is not None:
```

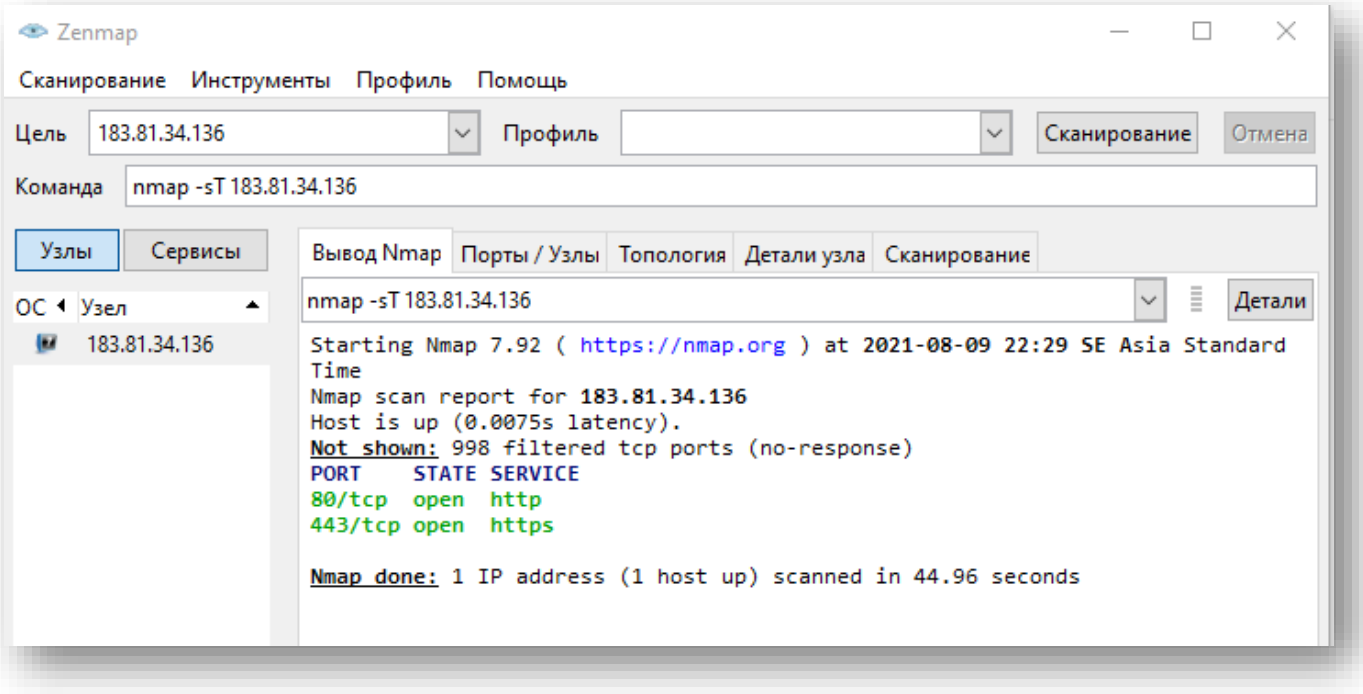
```
import nmap
portScannerAsync = nmap.PortScannerAsync()
def callback_result(host, scan_result):
    print(host, scan_result)
portScannerAsync.scan(hosts='scanme.nmap.org', arguments='-p 21', callback=callback_result)
portScannerAsync.scan(hosts='scanme.nmap.org', arguments='-p 22', callback=callback_result)
portScannerAsync.scan(hosts='scanme.nmap.org', arguments='-p 23', callback=callback_result)
portScannerAsync.scan(hosts='scanme.nmap.org', arguments='-p 80', callback=callback_result)
while portScannerAsync.still_scanning():
    print("Scanning >>>")
    portScannerAsync.wait(None)
```

Bài 4.4: Làm việc với Nmap thông qua mô đun os và subprocess

Thực hiện lệnh nmap với mô-đun os, sẽ không cần cài đặt thêm bất kỳ phần phụ thuộc nào và đó là cách dễ nhất để khởi chạy lệnh nmap thông qua shell.

Tham khảo:

```python
import os
nmap_command = "nmap -sT 127.0.0.1"
os.system(nmap_command)
```



Yêu cầu: Các em thực hiện lệnh nmap sao cho thể hiện nhiều thông tin nhất có thể về target.