



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

گزارش پروژه دوم درس امنیت اطلاعات

دانشجو: مرتضی صفری ۹۸۳۱۰۳۹

استاد: دکتر شهریاری

آبان ۱۴۰۲

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

بخش ۱

سوال ۱)

اسکن (Scanning) در زمینه‌ی امنیت رایانه‌ای به مرحله‌ای از فرایند ارزیابی امنیتی یک سیستم یا شبکه اشاره دارد. در این فاز، هدف اصلی بررسی و بررسی میزان‌ها (سرورها، کامپیوترها و سایر دستگاه‌ها) و شبکه است تا آسیب‌پذیری‌های امنیتی موجود در آنها شناسایی شود.

در فاز اسکن، ابزارها و تکنیک‌های مختلفی برای بررسی سیستم‌ها و شبکه‌ها استفاده می‌شوند. این ابزارها می‌توانند اسکن پورت‌ها، بررسی سرویس‌های فعال، تشخیص آسیب‌پذیری‌ها، جستجوی شناسه‌های سیستم و سرویس‌ها (مانند شماره نسخه و نام سرویس) و اسکن شبکه برای شناسایی دستگاه‌های متصل و نقاط ضعف احتمالی را شامل می‌شوند.

هدف از اسکن، شناسایی نقاط ضعف امنیتی موجود در سیستم یا شبکه است. این اطلاعات می‌توانند شامل لیست پورت‌های باز، سرویس‌های فعال، نقاط ضعف نرم‌افزاری، آسیب‌پذیری‌های ممکن و اطلاعات سیستمی مانند شماره نسخه سیستم عامل و سرویس‌های نصب شده باشد. اطلاعات حاصل از فاز اسکن می‌توانند در فازهای بعدی ارزیابی امنیتی مورد استفاده قرار گیرند. بر اساس نتایج اسکن، تصمیم‌گیری در مورد میزان آسیب‌پذیری سیستم و نحوه‌ی تقویت امنیت آن صورت می‌گیرد.

سوال ۲)

اسکن (Scanning) و Footprinting دو فاز متفاوت در فرایند ارزیابی امنیتی سیستم‌ها هستند. در ادامه تفاوت‌های اصلی بین این دو را توضیح می‌دهم:

Footprinting:

Footprinting فازی از فرایند ارزیابی امنیتی است که در آن اطلاعات جمع‌آوری شده در مورد یک سیستم یا شبکه به منظور شناخت و تجزیه و تحلیل آن استفاده می‌شود. در این فاز، اطلاعاتی مانند آدرس IP، دامنه‌ها، اطلاعات شبکه، سرویس‌ها، نام‌های کاربری و اطلاعات سازمانی جمع‌آوری می‌شوند. Footprinting اغلب با استفاده از منابع عمومی مانند موتورهای جستجو، دستورالعمل‌های WHOIS، اسنیفینگ شبکه و مطالعه اطلاعات عمومی (OSINT) صورت می‌گیرد.

Scanning:

Scanning فاز بعدی از فرایند ارزیابی امنیتی است که در آن سیستم یا شبکه به منظور شناسایی آسیب‌پذیری‌ها و نقاط ضعف امنیتی اسکن می‌شوند. در این فاز، ابزارها و تکنیک‌های مختلفی مانند اسکن پورت‌ها، تشخیص آسیب‌پذیری‌ها، بررسی سرویس‌های فعال و اسکن شبکه برای شناسایی دستگاه‌ها و نقاط ضعف ممکن استفاده شود. هدف اصلی این فاز، شناسایی آسیب‌پذیری‌ها و نقاط ضعف امنیتی است که در فاز بعدی تحلیل و ارزیابی می‌شوند.

به طور خلاصه، Footprinting فازي است که در آن اطلاعات جمع‌آوری و تحلیل می‌شوند تا سیستم‌ها و شبکه‌ها شناسایی شوند، در حالی که Scanning فازي است که در آن سیستم‌ها و شبکه‌ها اسکن و تست می‌شوند تا آسیب‌پذیری‌ها و نقاط ضعف امنیتی شناسایی شوند.

سوال ۳)

می‌توان اقدامات زیر را انجام دهید:

- ۱- غیرفعال‌سازی سرویس‌های غیرضروری: غیرفعال کردن سرویس‌ها و پورت‌های غیرضروری در سیستم‌ها و شبکه‌ها می‌تواند تعداد پورت‌های باز و سرویس‌های قابل دسترس را کاهش دهد و امکان کشف آسیب‌پذیری‌های مربوط به آنها را کاهش دهد.
- ۲- استفاده از فایروال: فایروال‌ها قادرند ترافیک شبکه را کنترل کنند و ترافیک غیرمجاز را مسدود کنند. با تنظیم فایروال به صورت صحیح، می‌توان اسکن پورت‌ها و تلاش‌های نفوذ را محدود کرد.
- ۳- به‌روزرسانی نرم‌افزارها و سیستم‌عامل: با به‌روزرسانی نرم‌افزارها و سیستم‌عامل به نسخه‌های جدید، آسیب‌پذیری‌های شناخته شده را کاهش داده و امکان بهره‌برداری از آنها را کاهش می‌دهید.
- ۴- اجرای شناسایی و جلوگیری از نفوذ (Intrusion Detection and Prevention): استفاده از سیستم‌های شناسایی و جلوگیری از نفوذ می‌تواند فعالیت‌های ناخواسته و مشکوک را تشخیص داده و متوقف کند.
- ۵- استفاده از شبکه‌های خصوصی مجازی (VPN): با استفاده از VPN، ارتباطات شبکه رمزگذاری شده و ترافیک شبکه محافظت می‌شود، که می‌تواند از کشف اطلاعات توسط هکرها در حین اسکن شبکه جلوگیری کند.
- ۶- پیکربندی صحیح دستگاه‌های شبکه: تنظیمات صحیح دستگاه‌های شبکه مانند روترها و سویچ‌ها باعث کاهش نقاط ضعف امنیتی و امکان اسکن شبکه توسط هکرها می‌شود.

۷- آموزش کارکنان: آموزش کارکنان در زمینه بهبود امنیت شبکه و آگاهی از تهدیدات امنیتی، به کاهش خطرات مربوط به اسکن شبکه کمک می‌کند. کارکنان باید درباره مفاهیم امنیتی پایه مانند رمزنگاری، مدیریت رمزعبور و فعالیتهای مشکوک آموزش داده شوند.

تست شناسایی ip

با کد

```
report.txt - Notepad
File Edit Format View Help
IP_Address: 192.168.1.1
IP_Address: 192.168.1.34
IP_Address: 192.168.1.36
IP_Address: 192.168.1.37

Pinging 192.168.1.42 with 32 bytes of data:
Request timed out.

Ping statistics for 192.168.1.42:
    Packets: Sent = 1, Received = 0, Lost = 1 (100% loss),

Pinging 192.168.1.43 with 32 bytes of data:
Request timed out.

Ping statistics for 192.168.1.43:
    Packets: Sent = 1, Received = 0, Lost = 1 (100% loss),

Pinging 192.168.1.44 with 32 bytes of data:
Request timed out.

Ping statistics for 192.168.1.44:
    Packets: Sent = 1, Received = 0, Lost = 1 (100% loss),

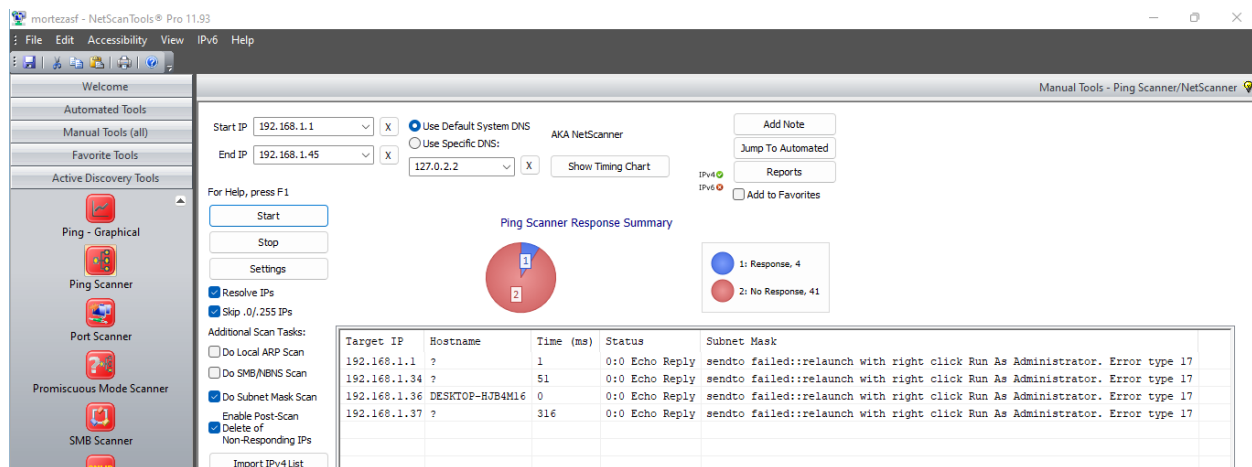
Pinging 192.168.1.45 with 32 bytes of data:
Request timed out.

Ping statistics for 192.168.1.45:
    Packets: Sent = 1, Received = 0, Lost = 1 (100% loss),

['192.168.1.1', '192.168.1.34', '192.168.1.36', '192.168.1.37']
report report.txt saved succussfully.
usage: project2.py [-h] [-p] [-t] [-u] ip start_port end_port
project2.py: error: argument start_port: invalid int value: '192.168.1.1'

C:\Users\Morteza Sf\Desktop\tamri2_security>
```

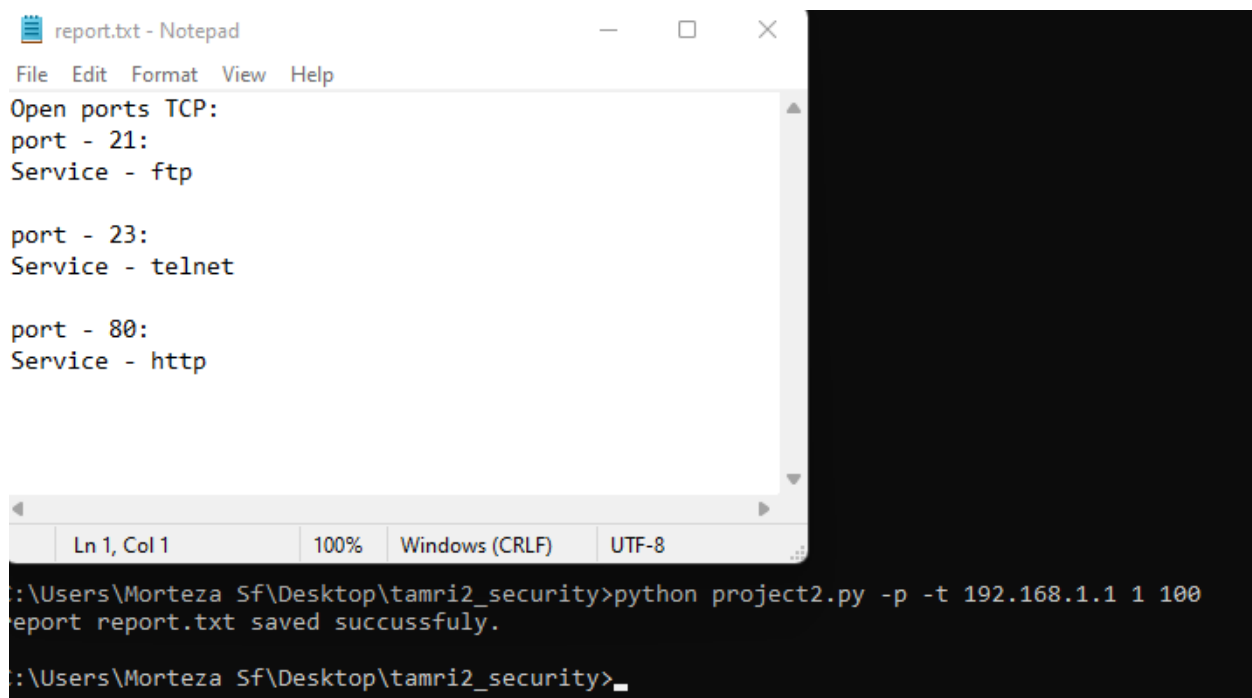
با نرم افزار



تست port

برای TCP:

با کد



mortezaaf - NetScanTools® Pro 11.93

File Edit Accessibility View IPv6 Help

Welcome

Automated Tools

Manual Tools (all)

Favorite Tools

Active Discovery Tools

Ping - Graphical

Ping Scanner

Port Scanner

Promiscuous Mode Scanner

SMB Scanner

SNMP - Core

SNMP Adv

Target Hostname or IP Address

192.168.1.1

Port Range and Scan Mode

☒ TCP Full Connect

☐ UDP Ports Only

☐ TCP Full+UDP Ports

☐ TCP SYN Scan (Half Open)

☐ TCP Custom Scan

Port Range

Start 1

End 100

☐ Use Target List When Scanning

Scan Complete - 100 ports scanned in 4 sec.

Scan Range of Ports

Scan Common Ports

Edit Common Ports List

Edit Target List

Stop

Settings

Defaults

Connect Timeout

2000

Read Timeout

3000

Network Interface (autoselected based on target IP address)

Wi-Fi (192.168.1.36) - Intel(R) Dual Band Wireless-AC 3165

☐ Show All Scanned Port Results

☒ Show TCP Summary

☐ Show UDP Summary

IPv4

IPv6

Add Note

Jump To Automated

Reports

Add to Favorites

TCP Full Connect Response Summary

1: Active TCP Ports, 3

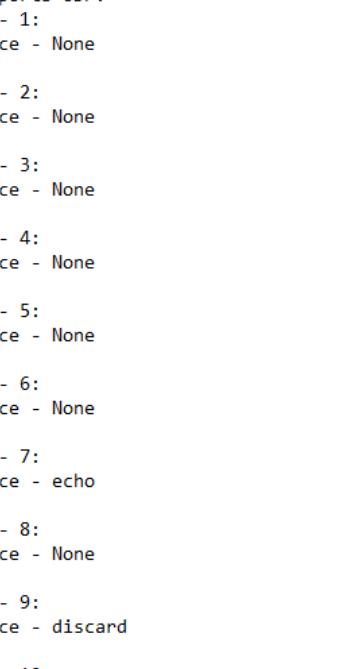
2: Active TCP Ports Returning Data, 2

3: TCP Ports Rejecting Connection, 0

4: No Response - Timeout, 97

IP Address	Port	Port Desc	Protocol	Results	Data Received
192.168.1.1	21	ftp	TCP	Port Active	220 DELI201-T10A FTP version 1.0 ready at Tue Nov 28 12:38:54 2023
192.168.1.1	23	telnet	TCP	Port Active	y0[y0@Password:
192.168.1.1	80	http	TCP	Port Active	

با کد



```
report.txt - Notepad
File Edit Format View Help
Open ports UDP:
port - 1:
Service - None

port - 2:
Service - None

port - 3:
Service - None

port - 4:
Service - None

port - 5:
Service - None

port - 6:
Service - None

port - 7:
Service - echo

port - 8:
Service - None

port - 9:
Service - discard

port - 10:
Service - None

Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

با نرم افزار

morteasf - NetScanTools® Pro 11.93

File Edit Accessibility View IPv6 Help

Welcome

Manual Tools - Port Scanner

Target Hostname or IP Address: 192.168.1.1 X

Port Range and Scan Mode

Port Range: Start 1 End 16

Scan Mode: ☒ UDP Ports Only ☐ TCP Full Connect ☐ TCP Full+UDP Ports ☐ TCP SYN Scan (Half Open) ☐ TCP Custom Scan

Use Target List When Scanning: ☐

Scan Complete - 16 ports scanned in 4 sec.

Network Interface (autoselected based on target IP address): Wi-Fi (192.168.1.36) - Intel(R) Dual Band Wireless-AC 3165

Show All Scanned Port Results: ☐

Show TCP Summary: ☒ Show UDP Summary: ☐

UDP Scan Response Summary

1: Possibly Active UDP Ports, 16

2: Closed UDP Ports (ICMP 3:3), 0

3: Filtered UDP Ports (other ICMP), 1

Connect Timeout: 2000

Read Timeout: 3000

IP Address	Port	Port Desc	Protocol	Results	Data Received
192.168.1.1	1	-	UDP	Port Active - Unconfirmed	
192.168.1.1	2	-	UDP	Port Active - Unconfirmed	
192.168.1.1	3	-	UDP	Port Active - Unconfirmed	
192.168.1.1	4	-	UDP	Port Active - Unconfirmed	
192.168.1.1	5	-	UDP	Port Active - Unconfirmed	
192.168.1.1	6	-	UDP	Port Active - Unconfirmed	
192.168.1.1	7	echo	UDP	Port Active - Unconfirmed	
192.168.1.1	8	-	UDP	Port Active - Unconfirmed	
192.168.1.1	9	discard	UDP	Port Active - Unconfirmed	
192.168.1.1	10	-	UDP	Port Active - Unconfirmed	

بخش ۲:

تحقیق کنید هریک از سویچ های زیر چه کاربردی دارند.

-ss در Nmap یک پارامتر است که برای تعیین نوع اسکن استفاده می شود. این پارامتر نشان می دهد که Nmap از اسکن SYN استفاده می کند. اسکن SYN یکی از رایج ترین و مورد استفاده ترین روش های اسکن شبکه در Nmap است. در اسکن SYN، Nmap یک ساختار پیام TCP SYN برای برقراری ارتباط با دستگاه ها ارسال می کند. این پیام SYN برای برقراری اتصال با دستگاه مقصد ارسال می شود و در صورتی که پورت مقصد باز باشد، دستگاه مقصد پیام SYN-ACK را در پاسخ ارسال می کند. در نهایت، Nmap یک پیام RST ارسال می کند تا اتصال را ببندد.

-sv در Nmap برای اسکن و تشخیص نسخه سرویس های در حال اجرا استفاده می شود. با استفاده از این پارامتر، Nmap سعی می کند نسخه های دقیق سرویس های در حال اجرا را تشخیص داده و به شما ارائه دهد. با استفاده از -sv، Nmap در هنگام اسکن پورت ها، درخواست های خاصی به سرویس ها ارسال می کند تا نسخه سرویس را تشخیص دهد. این درخواست ها ممکن است شامل درخواست های پروتکل خاصی مانند HTTP GET در صورت اسکن پورت ۸۰ (HTTP) باشد یا درخواست های پروتکل SMTP در صورت اسکن پورت ۲۵ (SMTP) و غیره. با تحلیل پاسخ های دریافتی از سرویس ها، Nmap سعی می کند نسخه سرویس را شناسایی کند و آن را نمایش دهد.

-st در Nmap برای اسکن TCP Connect استفاده می شود. اسکن TCP Connect یکی از روش های پرکاربرد و ساده ترین روش های اسکن در Nmap است. در اسکن TCP Connect، Nmap برای هر پورت مقصد یک اتصال TCP کامل برقرار می کند. اگر اتصال TCP برقرار شود و پورت مورد نظر باز باشد، این به معنی این است که پورت در دستگاه مقصد باز است. در غیر این صورت، اگر اتصال TCP برقرار نشود یا پورت بسته باشد، Nmap به این نتیجه می رسد که پورت در دستگاه مقصد بسته است. استفاده از پارامتر -st (اسکن TCP Connect) ممکن است در مواردی که فایروال ها و سیستم های امنیتی اسکن های SYN را مسدود کرده اند، مفید باشد. با این روش، Nmap به عنوان یک اتصال TCP معمولی عمل می کند و احتمالاً قادر است پورت های باز را تشخیص دهد.

تحقیق کنید کنید که هریک از سویچ های زیر مربوط به چه مودی از Scan هستند و تفاوت آن ها در چیست؟

F-: این سوئیچ برای اسکن TCP FIN استفاده می‌شود. در این روش، Nmap پیام FIN (پایان) را به دستگاه مقصد ارسال می‌کند. اگر پورت بسته باشد و هیچ پاسخی برگردانده نشود، Nmap نتیجه می‌گیرد که پورت بسته است. این روش به طور کلی کمتر قابل تشخیص توسط سیستم‌های امنیتی است، اما ممکن است در برخی موارد به دلیل پاسخ‌های متفاوت دستگاه‌های مقصد، نتایج دقیقی را به ارمغان بیاورد.

O-: این سوئیچ برای تشخیص سیستم‌عامل مقصد استفاده می‌شود. با استفاده از این سوئیچ، Nmap سعی می‌کند اطلاعاتی در مورد سیستم‌عامل در دستگاه مقصد را جمع‌آوری کند. برای این منظور، Nmap از تحلیل پاسخ‌های برگشتی به پیام‌های SYN و ACK استفاده می‌کند. این سوئیچ می‌تواند اطلاعات مفیدی را در مورد نوع سیستم‌عامل (مانند ویندوز، لینوکس، مک، و غیره) در دستگاه مقصد فراهم کند.

A-: این سوئیچ باعث اجرای یک اسکن جامع (Comprehensive Scan) در Nmap می‌شود. با استفاده از این سوئیچ، Nmap تعدادی از روش‌های اسکن را اجرا می‌کند، از جمله SYN Scan، TCP Connect Scan، UDP Scan، و سایر روش‌ها، و همچنین مشخصات سیستم‌عامل و نسخه سرویس‌ها را تشخیص می‌دهد. این سوئیچ معمولاً برای بررسی جامع و کامل شبکه و سیستم‌ها مورد استفاده قرار می‌گیرد.

تحقیق کنید که سوئیچ‌های sn- و pn- چه تفاوتی دارند

سوئیچ sn- یا sn-- برای اسکن شبکه به صورت پینگ (Ping Scan) استفاده می‌شود. در این روش، Nmap پیام‌های ICMP Echo Request را به دستگاه‌های مقصد ارسال می‌کند و منتظر پاسخ ICMP Echo Reply می‌ماند. اگر پاسخ دریافت شود، Nmap نتیجه می‌گیرد که دستگاه مقصد زنده است. این روش به عنوان یک روش سریع برای تشخیص دستگاه‌های موجود در شبکه استفاده می‌شود، اما نمی‌تواند اطلاعات جزئی‌تری مانند پورت‌ها و سرویس‌ها را تشخیص دهد.

سوئیچ pn- یا pn-- برای نادیده گرفتن پینگ (Ping Sweep) استفاده می‌شود. با استفاده از این سوئیچ، Nmap پینگ کردن دستگاه‌های مقصد را نادیده می‌گیرد و بدون ارسال پیام‌های ICMP، به تست پورت‌ها می‌پردازد. این روش برای تست پورت‌ها و تشخیص سرویس‌ها در دستگاه‌هایی که ممکن است برخی پینگ‌ها را بلاک کنند، مفید است. با استفاده از این روش، نتایج اسکن برای دستگاه‌هایی که پینگ را بلاک می‌کنند، نیز قابل مشاهده است.

تصاویر بخش ۲-۲ در بخش قبل قرار داده شد. در ادامه شرح کد را میبینیم:

گزارش کد

این گزارش برای کدی تهیه شده است که امکان اسکن محدوده آی پی و پورتها را فراهم می کند. این برنامه به شما امکان می دهد تا محدوده آی پی مورد نظرتان را اسکن کنید و ماشین های فعال را شناسایی کنید. همچنین، شما می توانید پورت های مشخصی را برای یک آی پی خاص اسکن کنید و سرویس هایی که بر روی آن پورتها در حال اجرا هستند را شناسایی کنید.

کتابخانه ها استفاده شده

برای پیاده سازی این کد، از دو کتابخانه socket و os استفاده شده است. کتابخانه socket برای برقراری ارتباط با سوکتها و اسکن پورتها استفاده می شود و کتابخانه OS برای اجرای دستورات سیستم عامل مورد نیاز در اسکن آی پی استفاده می شود.

توابع اصلی

تابع scan_ip_range(start_ip, end_ip)

این تابع برای اسکن محدوده آی پی استفاده می شود. آی پی ها را به ترتیب از start_ip تا end_ip پینگ می زند و آی پی های فعال را در لیست active_hosts ذخیره می کند. در نهایت، لیست active_hosts را برمی گرداند.

```
def scan_ip_range(start_ip, end_ip):
    active_hosts = []

    try:
        start_ip_split = start_ip.split('.')
        end_ip_split = end_ip.split('.')

        start_ip_int = int(start_ip_split[3])
        end_ip_int = int(end_ip_split[3])

        for i in range(start_ip_int, end_ip_int + 1):
            ip =
            f"{start_ip_split[0]}.{start_ip_split[1]}.{start_ip_split[2]}.{i}"
            response = os.system(f"ping -n 1 -w 1000 {ip}")
            print(response)

            if response == 0:
                active_hosts.append(ip)

        print(active_hosts)
```

```

return active_hosts

except Exception as e:
    print(f"Error in scan ip range: {str(e)}")
    return active_hosts

```

با استفاده از حلقه `for` بر روی محدوده آی‌پی‌ها پیمایش می‌شود. با استفاده از دستور `os.system` یک پینگ بر روی آی‌پی جاری ارسال می‌شود و نتیجه پینگ در متغیر `response` ذخیره می‌شود. در انتها، اگر نتیجه پینگ برابر با ۰ باشد، یعنی پینگ موفقیت‌آمیز بوده و آی‌پی فعال است. در این صورت، آی‌پی فعال به لیست `active_hosts` اضافه می‌شود. در نهایت، لیست `active_hosts` که شامل آی‌پی‌های فعال است، چاپ می‌شود و به عنوان خروجی تابع برگردانده می‌شود.

تابع `scan_ports(ip_address, start_port, end_port, protocol)`

این تابع برای اسکن پورت‌ها استفاده می‌شود. با استفاده از تابع `socket`، اتصال به هر پورت از `start_port` تا `end_port` را بررسی می‌کند. پورت‌های باز را در لیست `open_ports` ذخیره می‌کند و در نهایت آن را برمی‌گرداند.

```

def scan_ports(ip_address, start_port, end_port, protocol):
    open_ports = []

    try:
        for port in range(start_port, end_port + 1):
            if protocol == 'tcp':
                sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            else:
                sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
            sock.settimeout(1)
            result = sock.connect_ex((ip_address, port))

            if result == 0:
                open_ports.append(port)

            sock.close()

        return open_ports

    except Exception as e:
        print(f"Error in port scan: {str(e)}")
        return open_port

```

این تابع چهار آرگومان `ip_address`، `start_port`، `end_port` و `protocol` را دریافت می‌کند. در ابتدا، یک لیست خالی به نام `open_ports` تعریف می‌شود که در آن پورت‌های باز ذخیره خواهند شد. سپس با استفاده از حلقه `for` بر روی محدوده پورت‌ها پیمایش

می‌شود. در هر مرحله، یک سوکت ایجاد می‌شود. اگر پروتکل برابر با 'tcp' باشد، سوکت به صورت TCP ایجاد می‌شود و در غیر این صورت، سوکت به صورت UDP ایجاد می‌شود. سپس تایم‌اوت سوکت به ۱ ثانیه تنظیم می‌شود و با استفاده از تابع `connect_ex` اتصال به آدرس آی‌پی و پورت جاری برقرار می‌شود. نتیجه اتصال در متغیر `result` ذخیره می‌شود.

در صورتی که نتیجه اتصال برابر با ۰ باشد، یعنی اتصال موفقیت‌آمیز بوده و پورت باز است. در این صورت، پورت جاری به لیست `open_ports` اضافه می‌شود. در نهایت، سوکت بسته می‌شود و لیست `open_ports` که شامل پورت‌های باز است، به عنوان خروجی تابع برگردانده می‌شود. در صورت بروز خطا، خطا در اسکن پورت‌ها چاپ می‌شود و لیست `open_ports` خالی برگردانده می‌شود.

تابع `save_report(filename, content)`

این تابع برای ذخیره گزارش در یک فایل استفاده می‌شود. آن یک نام فایل و محتوای گزارش را دریافت می‌کند و محتوا را در فایل مورد نظر ذخیره می‌کند.

```
def save_report(filename, content):
    try:
        with open(filename, 'w') as file:
            file.write(content)
        print(f"report {filename} saved succussfully.")
    except Exception as e:
        print(f"Error in save report. {str(e)}")
```

تابع `identify_service(ip, port)`

این تابع برای شناسایی سرویس متصل به یک پورت استفاده می‌شود. با استفاده از تابع `socket.getservbyport`، نام سرویس مرتبط با پورت مشخص شده را برمی‌گرداند.

```
def identify_service(ip, port):
    try:
        service_name = socket.getservbyport(port)
        return service_name
    except Exception as e:
        print(f"Error identifying the service on the port: {port}: {str(e)}")
        return None
```

تابع main()

این تابع اصلی برنامه است. از ماژول `argparse` برای پردازش آرگومان‌ها استفاده می‌شود. برنامه با استفاده از آرگومان‌های ارائه شده، تشخیص می‌دهد کدام بخش از برنامه باید اجرا شود. در نهایت، گزارش مربوطه با استفاده از توابع اسکن و شناسایی سرویس‌ها تهیه و در یک فایل ذخیره می‌شود.

```
def main():
    import argparse
    parser = argparse.ArgumentParser(description='اسکن محدوده آی‌پی و یافتن ماشین‌های فعال یا اسکن پورت‌ها')
    parser.add_argument('--ipscan', action='store_true', help='اسکن محدوده آی‌پی')
    parser.add_argument('-m', '--subnet_mask', type=int, help='ماسک زیرشبکه')
    parser.add_argument('-ip', '--ip_range', nargs='+', help='محدوده آی‌پی (شروع و پایان)')
    parser.add_argument('-p', '--portscan', action='store_true', help='اسکن پورت‌ها')
    parser.add_argument('-t', '--tcp', action='store_true', help='اسکن پورت‌های TCP')
    parser.add_argument('-u', '--udp', action='store_true', help='اسکن پورت‌های UDP')
    parser.add_argument('ip', type=str, nargs='?', help='اسکن آی‌پی')
    parser.add_argument('start_port', type=int, nargs='?', help='پورت شروع')
    parser.add_argument('end_port', type=int, nargs='?', help='پورت پایان')

    args = parser.parse_args()

    if args.ipscan:
        if args.subnet_mask and args.ip_range:
            start_ip = args.ip_range[0]
            end_ip = args.ip_range[1]
            subnet_mask = args.subnet_mask

            active_hosts = scan_ip_range(start_ip, end_ip)

            report_content = ""

            for host in active_hosts:
                report_content += f"Ip address {host}\n"

            save_report("report.txt", report_content)
        else:
            print('Please specify the IP range and subnet mask')

    elif args.portscan:
```



```

if args.tcp:
    protocol = 'tcp'
elif args.udp:
    protocol = 'udp'
else:
    print('To scan ports, you must choose one of TCP and UDP protocols.')
    return

open_ports = scan_ports(args.ip, args.start_port, args.end_port,
protocol)
if len(open_ports) > 0:
    report_content = f"Open ports {protocol.upper()}:\n"
    for port in open_ports:
        report_content += f"port - {port}:\n"
        report_content += f"Service - {identify_service(args.ip,
port)}\n\n"
    else:
        report_content = f"open port {protocol.upper()} not found"

save_report("report.txt", report_content)

```

نتیجه‌گیری

کد ارائه شده امکان اسکن محدوده آی‌پی و پورت‌ها را فراهم می‌کند و به شما امکان می‌دهد تا ماشین‌های فعال در یک محدوده آی‌پی را شناسایی کنید. همچنین، می‌توانید پورت‌های مشخصی را برای یک آی‌پی خاص اسکن کرده و سرویس‌هایی که بر روی آن پورت‌ها در حال اجرا هستند را شناسایی کنید.

با تشکر از توجه شما