ثمین مهدی پور ۹۸۳۹۰۳۹

گزارش پروژه سوم – امنیت اطلاعات

بخش ۱)

۱-۱-۱-

در زبان امنیت اطلاعات کامپیوتری، عبارت "scanning" به یک فرآیند اشاره دارد که هدف آن بررسی و تحلیل شبکه‌ها و سیستم‌های کامپیوتری به منظور شناسایی آسیب‌پذیری‌ها، سرویس‌ها، یا دستگاه‌های متصل به شبکه است. این فرآیند می‌تواند به صورت فعال یا غیرفعال انجام شود.

در فاز scanning، اطلاعات مهمی جمع‌آوری می‌شود که به حمله‌کننده کمک می‌کند تا نقاط ضعف و فرصت‌های حمله را شناسایی کند. برخی از اطلاعاتی که در این فاز به‌دست می‌آیند عبارتند از:

۱. پورت‌ها و سرویس‌ها: حمله‌کننده با استفاده از scanning، می‌تواند پورت‌های باز و سرویس‌های در حال اجرا را شناسایی کند. این اطلاعات برای تعیین نقاط ضعف و مسیرهای حمله مهم است.

۲. نقاط ضعف (Vulnerabilities): اسکنرها ممکن است به دنبال نقاط ضعف مربوط به نرم‌افزارها یا سیستم‌عامل مشخص باشند. اطلاعات درباره آسیب‌پذیری‌ها به حمله‌کننده اجازه می‌دهد تا حملات خود را متناسب با آسیب‌پذیری‌های موجود انجام دهد.

۳. توپولوژی شبکه: اطلاعاتی درباره ساختار شبکه، اتصالات، و اجزای مختلف آن جمع‌آوری می‌شود. این اطلاعات ممکن است به حمله‌کننده کمک کند تا مسیرهای مختلف حمله را بشناسد.

۴. سیستم‌های فعال: شناسایی سیستم‌های فعال در شبکه و جزئیات مربوط به آن‌ها نیز جزو اطلاعاتی است که در این فاز قابل استخراج است.پیشگیری از این نوع فعالیت‌ها و اجرای امنیت مناسب می‌تواند کمک کند تا امکان اجرای موفق حملات توسط حمله‌کنندگان کاهش یابد. اجرای به‌روزرسانی‌های امنیتی، پوشش دهی از طریق فایروال‌ها، و استفاده از ابزارهای شناسایی و جلوگیری از نفوذ می‌تواند بهبود امنیت سیستم‌ها کمک کند.

۱-۱-۲-

Scanning و Footprinting دو فعالیت مهم در فرآیند اطلاعاتی (Information Gathering) در حوزه امنیت اطلاعات هستند، اما تفاوت‌های مهمی دارند:

۱. Footprinting (پاگذاری):

- تعریف: Footprinting به معنای جمع‌آوری اطلاعات جامع و مفید درباره سازمان یا هدف مورد نظر است. این اطلاعات ممکن است از منابع عمومی (مثل وب‌سایت‌ها، شبکه‌های اجتماعی، اسناد عمومی) جمع‌آوری شود.

- هدف: هدف Footprinting این است که یک تصویر کلی از هدف را ایجاد کند و به حمله‌کننده اطلاعات لازم برای مراحل بعدی حمله فراهم کند.

- نوع اطلاعات: اطلاعات مربوط به ارگانیزاسیون، شبکه‌ها، ساختار سازمانی، افراد کلیدی، و اطلاعات مربوط به امنیت سازمانی از جمله اطلاعاتی هستند که در این مرحله جمع‌آوری می‌شوند.

۲. Scanning (اسکن):

- تعریف: Scanning به معنای بررسی و اسکن کردن سیستم‌ها یا شبکه‌ها به منظور شناسایی پورت‌ها، سرویس‌ها، و آسیب‌پذیری‌ها است.

- هدف: هدف Scanning این است که اطلاعات فنی و مربوط به سیستم‌ها و شبکه‌ها را جمع‌آوری کرده و نقاط ضعف ممکن را شناسایی کند.

- نوع اطلاعات: اطلاعات فنی مانند پورت‌های باز، سرویس‌های در حال اجرا، و آسیب‌پذیری‌ها از جمله اطلاعاتی هستند که در این فرآیند به دست می‌آید.

به طور خلاصه، Footprinting به تعمق بیشتری اطلاعات مربوط به هدف را جمع‌آوری می‌کند تا یک تصویر کلی و جامع از سازمان و شبکه ارائه دهد، در حالی که Scanning بیشتر متمرکز بر جزئیات فنی سیستم‌ها و شبکه‌هاست و به شناسایی پورت‌ها و آسیب‌پذیری‌ها می‌پردازد.

برای مقابله با scanning و جلوگیری از اطلاعاتی که حمله‌کننده از طریق اسکن شبکه ممکن است بدست آورد، می‌توان از راهکارهای زیر استفاده کرد:

۱. Firewalls و فیلترهای شبکه:

- استفاده از فایروال‌ها و فیلترهای شبکه باعث محدود کردن دسترسی حمله‌کننده به برخی پورت‌ها و سرویس‌ها می‌شود.

- تنظیم فایروال به گونه‌ای که ترافیک ورودی و خروجی با دقت کنترل شود، می‌تواند کمک کند تا اطلاعات کمتری در معرض اسکن قرار گیرد.

۲. به‌روزرسانی نرم‌افزارها و سیستم‌عامل:

- اطمینان از این که تمام نرم‌افزارها و سیستم‌عامل‌های مورد استفاده به‌روزرسانی شده‌اند. این کار با کاهش آسیب‌پذیری‌ها و امکان اجرای حملات بر روی نقاط ضعف ممکن در سیستم‌ها کمک می‌کند.

۳. پنهان‌سازی اطلاعات:

- از مکانیسم‌های پنهان‌سازی (Stealth) در تنظیمات شبکه و سیستم‌ها استفاده کنید تا اطلاعات غیرضروری پنهان شوند و فرایند اسکن سخت‌تر شود.

۴. مدیریت پورت‌ها:

- تنظیم پورت‌ها به گونه‌ای که تنها پورت‌های لازم برای عملکرد سیستم باز باشند و پورت‌های غیرضروری بسته شوند. این کار با کاهش سطح حمله‌پذیری به برخی از حملات اسکن مرتبط با پورت‌ها کمک می‌کند.

۵. شناسایی و پاسخ به حملات:

- استفاده از سیستم‌های شناسایی نفوذ (IDS) و سیستم‌های مانیتورینگ شبکه می‌تواند به شناسایی فعالیت‌های ناشناخته و مشکوک در شبکه کمک کند. همچنین، پاسخ به سریع به حملات امکان پیشگیری از ادامه آن‌ها را فراهم می‌کند.

۶. آموزش کارکنان:

- آموزش کارکنان در مورد مسائل امنیتی، اهمیت حفاظت از اطلاعات، و رفتارهای امنیتی می‌تواند کمک کند تا حوادث امنیتی کاهش یابد. به کارکنان آموزش دهید که اطلاعات مهم را به دقت مدیریت کنند و از اقدامات

امنیتی استفاده کنند. این راهکارها در کنار یکدیگر مورد استفاده قرار گرفته و باعث تقویت لایه‌های مختلف امنیتی شبکه و سیستم می‌شوند.

۱-۲-

برای پیاده سازی این بخش گام به گام جلو میرویم:

در اولین قدم برای استفاده از کامندلاین با کتابخانه argparse تابعی به نام parse_arguments را پیاده سازی میکنیم:

```
def parse_arguments():
    parser = argparse.ArgumentParser(description="Network Scanner CLI Tool")

    # IP Scan Command
    ip_scan_parser = parser.add_argument_group("IP Scan Options\n")
    ip_scan_parser.add_argument("--ipscan", action="store_true",
help="Perform IP scanning")
    ip_scan_parser.add_argument("-m", "--subnet-mask", type=int, help="Subnet
mask (e.g., 24)")
    ip_scan_parser.add_argument("start_ip", help="Start IP address")
    ip_scan_parser.add_argument("end_ip", help="End IP address", nargs="?")
# Make end_ip optional for portscan

    # Port Scan Command
    port_scan_parser = parser.add_argument_group("Port Scan Options")
    port_scan_parser.add_argument("--portscan", action="store_true",
help="Perform port scanning")
    port_scan_parser.add_argument("--tcp", action="store_true", help="Use TCP
protocol")
    port_scan_parser.add_argument("--udp", action="store_true", help="Use UDP
protocol")
    port_scan_parser.add_argument("target_ip", help="Target IP address for
port scanning")

    args = parser.parse_args()
    return args
```

همانطور که مشاهده میشود در این تابع برای اسکن کردن پورت و آی پی کامندهایی در نظر گرفته شده است.

در گام بعدی میخواهیم تابعی رو پیاده سازی کنیم که اسکن آی پی را انجام میدهد برای این کار آی پی آغازین و پایانی دریافت میشود و با برقراری یک اتصال TCP با پورت ۸۰ دسترسی پذیری آی پی بررسی میشود:

```
def ip_to_int(ip):
    return struct.unpack("!I", socket.inet_aton(ip))[0]


def int_to_ip(ip_int):
    return socket.inet_ntoa(struct.pack("!I", ip_int))


def ip_scan(start_ip, end_ip, subnet_mask):
    active_machines = []

    start_int = ip_to_int(start_ip)

    # If end_ip is provided, scan the IP range; otherwise, scan only the
specified IP
    if end_ip:
        end_int = ip_to_int(end_ip)
        subnet_int = 2 ** (32 - subnet_mask)

        for i in range(subnet_int):
            current_ip_int = start_int + i
            current_ip = int_to_ip(current_ip_int)

            # Perform socket connection to check if the IP is reachable
            try:
                socket.create_connection((current_ip, 80), timeout=1)
                active_machines.append(current_ip)
            except (socket.timeout, socket.error):
                pass
    else:
        try:
            socket.create_connection((start_ip, 80), timeout=1)
            active_machines.append(start_ip)
        except (socket.timeout, socket.error):
            pass

    return active_machines

    return active_machines
```

حال میخواهیم به اسکن پورت های فعال یک ماشین فعال بپردازیم:

```
def port_scan(target_ip, tcp_scan=True, udp_scan=True):
    open_ports = []

    for port in range(1, 1025):  # Scan common ports, adjust as needed
        try:
            if tcp_scan:
                socket.create_connection((target_ip, port), timeout=1)
                open_ports.append((port, "TCP"))
```

```
            elif udp_scan:
                udp_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
                udp_socket.settimeout(1)
                udp_socket.sendto(b'', (target_ip, port))
                open_ports.append((port, "UDP"))
        except (socket.timeout, socket.error):
            pass

    return open_ports
```

در این تابع با بررسی پورت هایی که اتصال های آنها از نوع TCP,UDP است و برقراری ارتباط با آنها بررسی میکنیم اگر پاسخی دریافت شده باشد فعال هستند پس آنها را در لیستی ذخیره کرده و نهایتا این لیست پورتهای فعالی که از پروتکل های مدنظر استفاده میکنند را برمیگردانیم.

در تابع بعدی به بررسی سرویس های فعال روی این پورت های فعال یک ماشین فعال میپردازیم:

```
def identify_services(target_ip, open_ports):
    services = []

    for port, protocol in open_ports:
        try:
            with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
                s.settimeout(1)
                s.connect((target_ip, port))
                service_info = s.recv(1024).decode('utf-8')
                services.append((port, protocol, service_info))
        except (socket.timeout, socket.error):
            pass

    return services
```

همانطور که مشاهده میشود تابع بالا با دریافت لیست پورتهای فعال و آی پی مقصد با برقراری اتصال اطلاعاتی دریافت میکند که حاوی اطلاعات سرویس است و آن را در لیستی ذخیره کرده و برمیگرداند.

در گام نهایی تابعی داریم که مسئولیت نمایش اطلاعات بدست امده توسط توابع قبلی و ذخیره آن در یک فایل متنی را دارد:

```
def display_and_save_report(active_machines, open_ports, services):
    print("\n=== Network Scan Report ===")

    # Display active machines
    print("\nActive Machines:")
    for machine in active_machines:
        print(f"- {machine}")

    # Display open ports
    print("\nOpen Ports:")
    for port, protocol in open_ports:
        print(f"- Port {port} ({protocol}) is open")

    # Display identified services
    print("\nIdentified Services:")
```

```
    for port, protocol, service_info in services:
        print(f"- Port {port} ({protocol}): {service_info}")

    # Save the report to a text file
    with open("NetworkScanReport.txt", "w") as report_file:
        report_file.write("=== Network Scan Report ===\n\n")

        # Write active machines
        report_file.write("\nActive Machines:\n")
        for machine in active_machines:
            report_file.write(f"- {machine}\n")

        # Write open ports
        report_file.write("\nOpen Ports:\n")
        for port, protocol in open_ports:
            report_file.write(f"- Port {port} ({protocol}) is open\n")

        # Write identified services
        report_file.write("\nIdentified Services:\n")
        for port, protocol, service_info in services:
            report_file.write(f"- Port {port} ({protocol}):
{service_info}\n")

    print("\nReport saved to NetworkScanReport.txt")
```

حالا برنامه را تست میکنیم:

```
PS C:\Users\Samin\Desktop\University\Term 8\Information Security\projects\project2> python scanner.py --ipscan -m 24 192.168.1.1 192.168.1.254

=== Network Scan Report ===

Active Machines:
- 192.168.1.1
- 192.168.1.3
```

مشاهده میشود که ماشین های فعال ذکر شده اند و در فایل هم ذخیره شده اند.

NetworkScanReport.txt - Notepad

File Edit Format View Help

```
=== Network Scan Report ===

Active Machines:
- 192.168.1.1
- 192.168.1.3
- 192.168.1.4
- 192.168.1.5
- 192.168.1.6
- 192.168.1.9
- 192.168.1.10
- 192.168.1.11
- 192.168.1.12
- 192.168.1.13
- 192.168.1.14
- 192.168.1.15
- 192.168.1.16
- 192.168.1.17
- 192.168.1.18
- 192.168.1.19
- 192.168.1.20
- 192.168.1.21
- 192.168.1.22
- 192.168.1.23
- 192.168.1.24
- 192.168.1.25
- 192.168.1.26
- 192.168.1.27
```

اسکن پورت را چک میکنیم:

NetworkScanReport-2.txt - Notepad

File Edit Format View Help

```
=== Network Scan Report ===


Active Machines:

Open Ports:
- Port 80 (TCP) is open
- Port 443 (TCP) is open

Identified Services:
```

بخش دوم)

۲-۱-۱-

- `sS-`: این سوئیچ در nmap به معنای "TCP SYN Scan" است. با استفاده از این سوئیچ، nmap تلاش می‌کند با ارسال یک پیام SYN به هر پورت تعریف شده برای یک میزبان، تشخیص دهد که آیا پورت باز است یا نه. این اسکن به عنوان یک اسکن سریع و کم‌هزینه شناخته می‌شود.

- `sV-`: این سوئیچ برای "Version Detection" یا تشخیص نسخه سرویس‌های در حال اجرا استفاده می‌شود. با استفاده از این سوئیچ، nmap سعی می‌کند نسخه سرویس‌هایی که در پورت‌های باز پیدا می‌شوند را شناسایی کرده و نمایش دهد.

- `sT-`: این سوئیچ به معنای "TCP Connect Scan" است. در این نوع اسکن، nmap یک اتصال TCP کامل به هر پورت مورد نظر ایجاد می‌کند. این روش اسکن اطلاعات دقیق‌تری از وضعیت پورت‌ها ارائه می‌دهد اما ممکن است برخی از سیستم‌ها به دلیل ایجاد اتصال از پیش، فعالیت اسکن را تشخیص دهند.

۲-۱-۲-

- `F-`: سوئیچ `F-` در nmap به معنای "Fast Scan" یا اسکن سریع است. این نوع اسکن سعی دارد با اسکن تعداد محدودی از پورت‌ها به سرعت اطلاعاتی از میزبان هدف جمع‌آوری کند. معمولاً این اسکن بر روی پورت‌های مهم و شناخته‌شده انجام می‌شود.

- `O-`: این سوئیچ به معنای "OS Detection" یا تشخیص سیستم عامل است. با استفاده از این سوئیچ، nmap سعی می‌کند نوع سیستم عاملی که بر روی میزبان اجرا می‌شود را تشخیص دهد. این اطلاعات می‌تواند در تحلیل امنیتی و شناخت محیط شبکه مفید باشد.

- `A-`: این سوئیچ "Aggressive Scan" یا اسکن فشرده را نمایان می‌کند. با استفاده از این سوئیچ، nmap تلاش می‌کند اطلاعات بیشتری از میزبان را جمع‌آوری کند، از جمله نوع سیستم عامل، نسخه سرویس‌ها، و دیگر اطلاعات. این اسکن ممکن است برخی از فایروال‌ها و سیستم‌های حفاظتی را فراموش کند و در نتیجه به نتایج دقیق‌تری منجر شود. اما به عنوان یک اسکن فشرده، ممکن است دیده‌شود و در برخی از موارد محدودیت‌هایی

داشته باشد.

۲-۱-۳-

در ابزار nmap، سوئیچ‌های `-sn` و `-pn` به ترتیب به معنای "Ping Scan" و "No Ping" هستند و تفاوت‌های زیر را دارند:

۱. `-sn` (Ping Scan):

- این سوئیچ به معنای اسکن پینگ است. با استفاده از این سوئیچ، nmap سعی می‌کند با ارسال پیغام‌های پینگ (ICMP Echo Requests) به میزبان‌های هدف، ایستگاه‌های فعال را شناسایی کند.

- این نوع اسکن به سرعت اجرا می‌شود و تنها میزبان‌هایی را نشان می‌دهد که پینگ پاسخ می‌دهند.

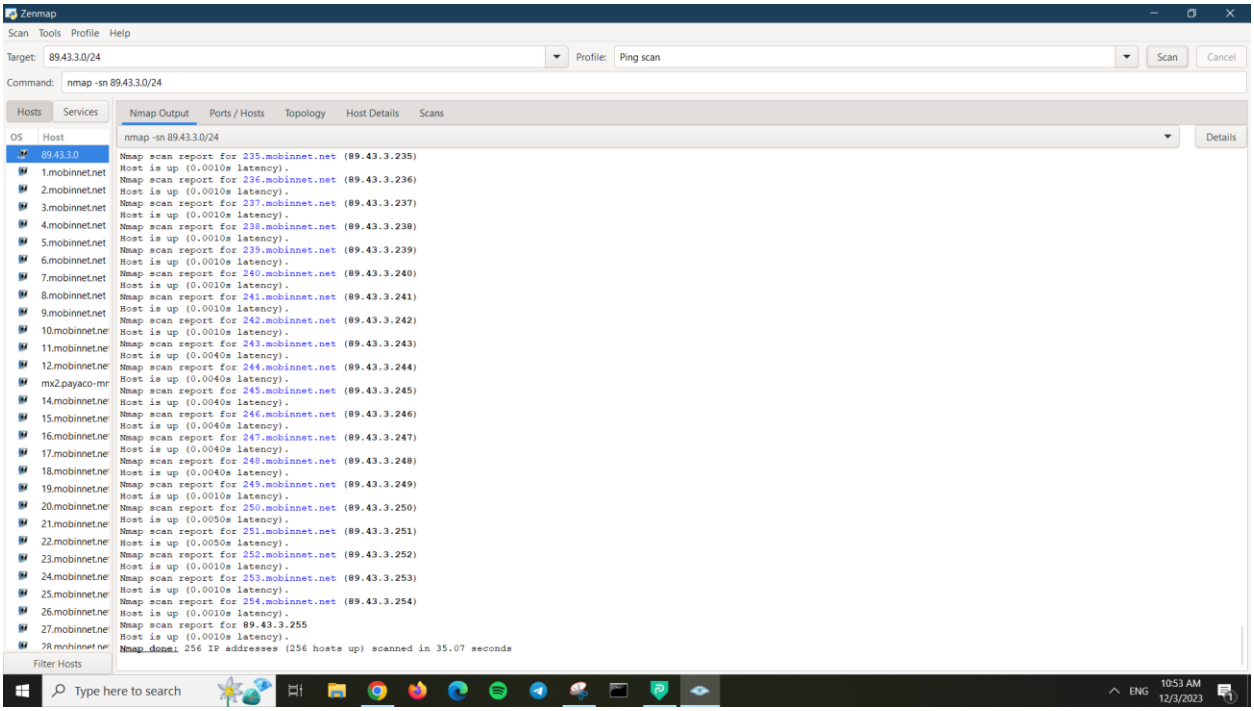
۲. `-pn` (No Ping):

- این سوئیچ به معنای "No Ping" یا "بدون پینگ" است. با استفاده از این سوئیچ، nmap هیچ پینگی ارسال نمی‌کند و به تنهایی به اسکن پرداخته و سرویس‌های فعال را شناسایی می‌کند.

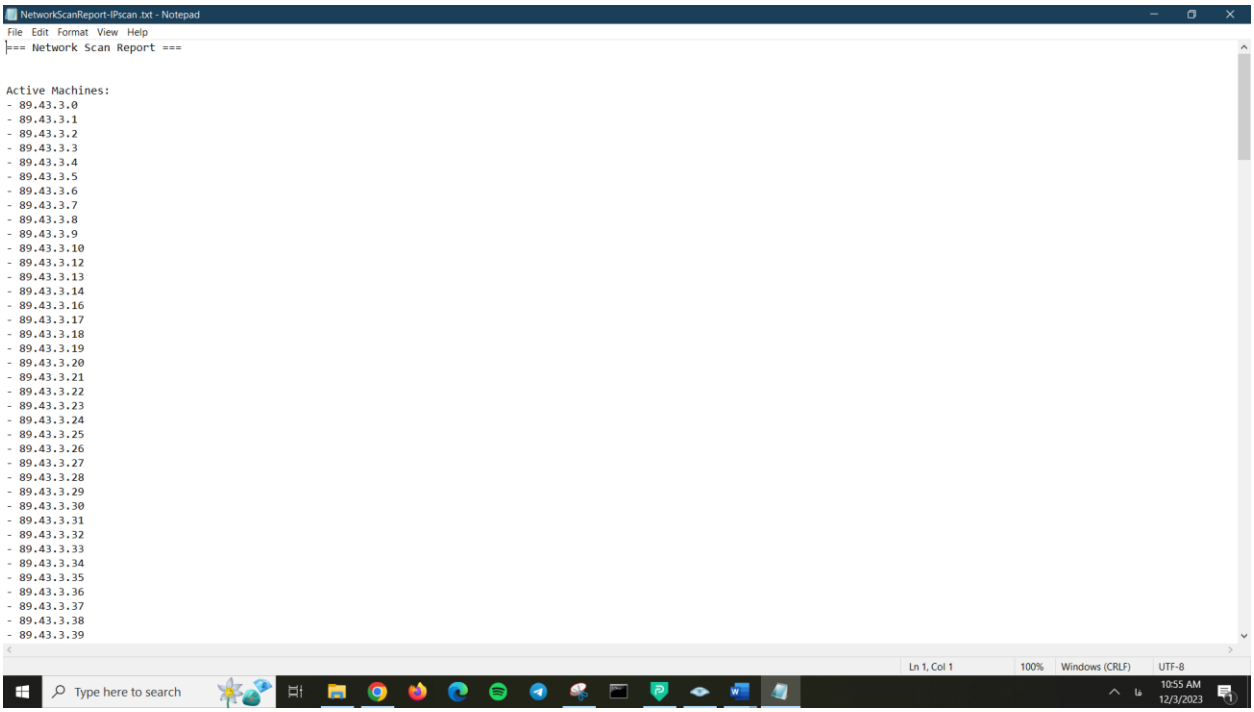
- این نوع اسکن ممکن است در مواردی که میزبان‌ها پینگ را ممنوع کرده‌اند یا به دلایل امنیتی، پینگ پاسخ نمی‌دهند، مفید باشد.

بنابراین، تفاوت اصلی بین این دو سوئیچ در این است که `-sn` از پینگ برای شناسایی میزبان‌های فعال استفاده می‌کند، در حالی که `-pn` از پینگ صرف نظر می‌کند و تلاش می‌کند مستقیماً سرویس‌ها را شناسایی کند.

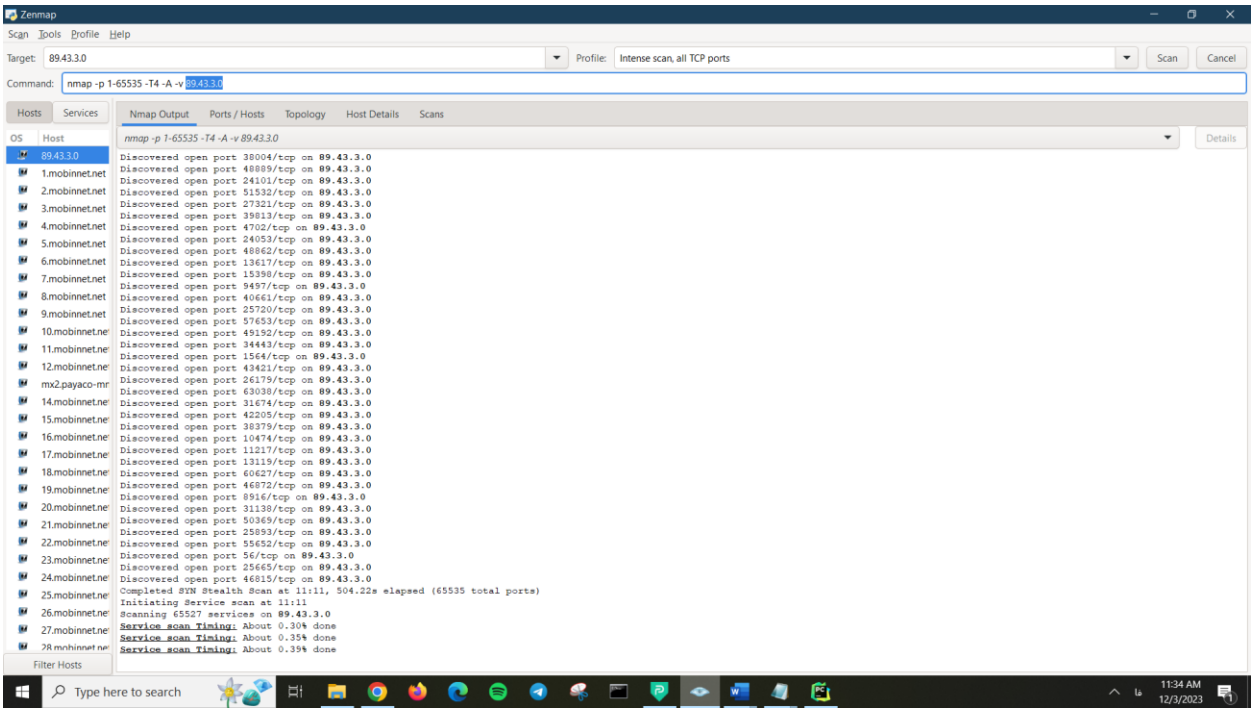
۲-۲- ابتدا آی پی اسکن را انجام میدهیم:



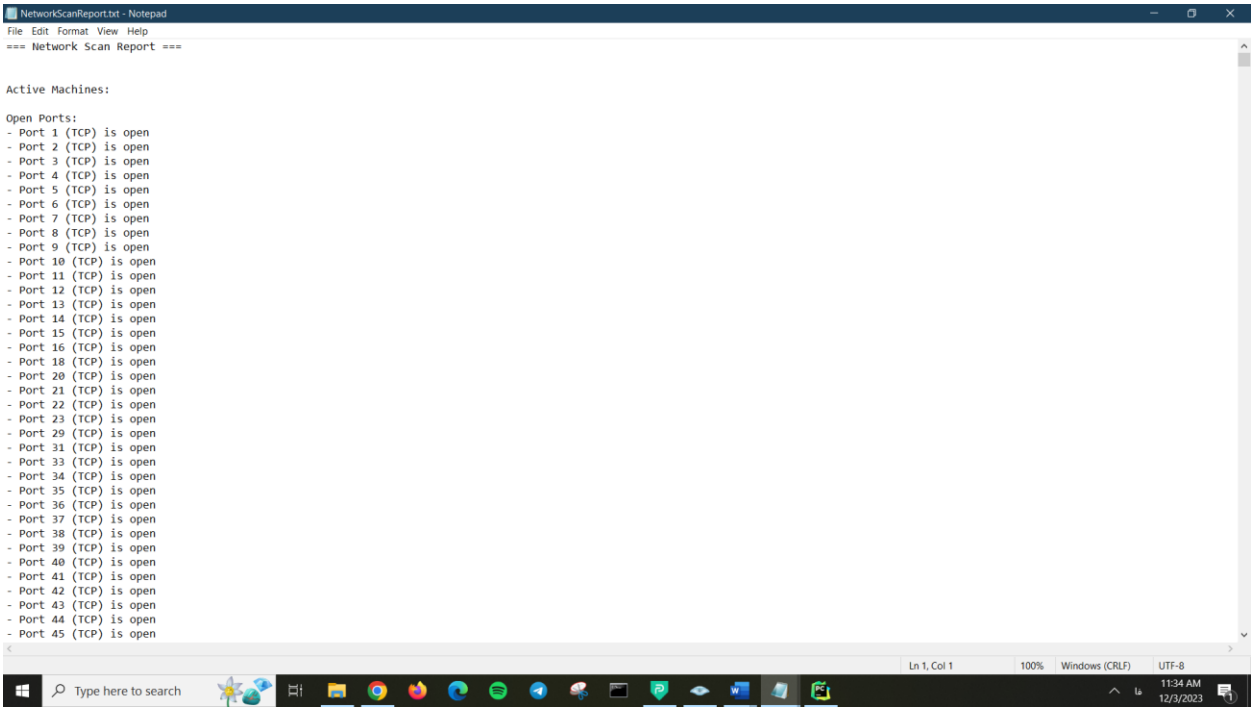
مشاهده میشود که ۲۵۶ هوست بالا شناسایی شده اند که در کد هم همین خروجی را داشتیم:



حالا پورت های فعال را برای آی پی ۸۹.۴۳.۳.۰ شناسایی میکنیم:



مشاهده میشود کد هم همین خروجی را داده است:



همچنین در هردو اسکن سرویسی شناسایی نشده: