

بخش سوم

کسرا نوربخش 810100230

هومان مشیری امین 810100210

مقدمه

با استفاده از وایرشارک و بررسی اطلاعات موجود در فایل pcap، ما پکت‌ها را بررسی می‌کنیم تا IP هایی که رفتار مشکوک دارند را شناسایی کنیم، به ویژه در این تمرین به دنبال IP هایی هستیم که احتمالاً در حال port scanning هستند. این IP مشکوک به تعداد زیادی از پورت ها syn میفرستد ولی خیلی از این port ها در دسترس نیستند، بنابراین میگردیم ببینیم کجا تعداد اختلاف syn و ack+syn ها بالا هست، احتمال زیاد آن IP در حال اسکن پورت هاست. (در بخش 2 گفته شده که $\text{syn} \geq 3 * (\text{syn} + \text{ack})$)

پاسخ

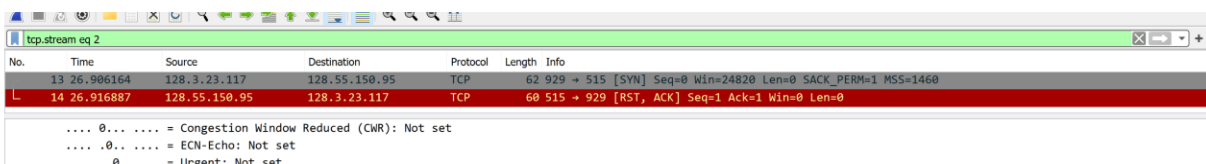
کار با وایرشارک: با استفاده از فیلترهایی مثل tcp.stream (البته برای بقیه پروتکل ها هم میتوان از stream استفاده کرد، مثلاً udp.stream، استریم را مساوی عدد میگذاریم تا آن ها را بررسی کنیم)

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	128.3.23.5	196.119.124.204	TCP	60	54558 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460
2	3.000395	128.3.23.5	196.119.124.204	TCP	60	[TCP Retransmission] [TCP Port numbers reused] 54558 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460
10	9.001105	128.3.23.5	196.119.124.204	TCP	60	[TCP Retransmission] [TCP Port numbers reused] 54558 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460
12	21.002018	128.3.23.5	196.119.124.204	TCP	60	[TCP Retransmission] [TCP Port numbers reused] 54558 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460
198	45.003597	128.3.23.5	196.119.124.204	TCP	60	[TCP Retransmission] [TCP Port numbers reused] 54558 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460

▼ Frame 10: 60 bytes on wire (480 bits), 58 bytes captured (464 bits) on interface unknown, id 0		
Interface id: 0 (unknown)		
Encapsulation type: Ethernet (1)		
Arrival Time: Oct 4, 2004 23:36:10.208284000 Iran Standard Time		
[Time shift for this packet: 0.000000000 seconds]		
Epoch Time: 1096920370.208284000 seconds		
[Time delta from previous captured frame: 3.629781000 seconds]		
[Time delta from previous displayed frame: 6.000710000 seconds]		
[Time since reference or first frame: 9.001105000 seconds]		
Frame Number: 10		

0000	80 0b 98 3b b9 ec 84 0a 4a 35 ea 09 08 00 45 00J5....E
0010	00 2c 3c ec 40 00 40 06 25 94 80 03 17 05 c4 77	...@...%.....w
0020	7c cc d5 1e 00 50 81 b4 55 5c 00 00 00 00 60 02P..U\.....
0030	ff ff 13 5b 00 00 02 04 05 b4

مثل‌های دیگر در زمینه امنیت packet:



No.	Time	Source	Destination	Protocol	Length	Info
13	26.906164	128.3.23.117	128.55.150.95	TCP	62	929 → 515 [SYN] Seq=0 Win=24820 Len=0 SACK_PERM=1 MSS=1460
14	26.916887	128.55.150.95	128.3.23.117	TCP	60	515 → 929 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

.... 0... .. = Congestion Window Reduced (CWR): Not set
 0... .. = ECN-Echo: Not set
0. = Urgent: Not set

مثلا در این استریم میبینیم که درخواست به پورت 515 داده شده، که در زمان خودش آسیب پذیری های زیادی داشته، مثلا عدم احراز هویت قوی (بدون اینکه هویت تایید بشه) و حمله های دیگه. البته در اینجا ارتباط ریست شده یعنی سیستم یا فایروال خودش ارتباطو نداشته شکل بگیره و اینکه در بخش اول ما به این کار نداریم و بیشتر دنبال port scanning هستیم.

خروجی آزمایش:

ران شده کد را میبینیم که همان آیی های مورد نظر پی دی اف را شناسایی کرده.

```
PS C:\Users\Houman\Desktop\ECS-CA3-810100210\3> python .\syn_scanner.py .\reduced_sample.pcap

Potential SYN port scanners:
128.3.23.5
128.3.164.249
128.3.164.248
128.3.23.2
128.3.23.117
128.3.23.158
```

سوالات تشریحی:

به دنبال اطلاعات دیگر در trace که شخص مهاجم بتواند از آن استفاده کند:
 اول از همه مهاجم به دنبال پروتکل هایی بگردد که رمزگذاری ندارند، احتمالش هست که اطلاعات مهمی پیدا بشه (مثل نام کاربری و رمز عبور و ...). داخل پکت ها به دنبال پروتکل هایی مثل http, ftp, telnet, smtp, dns میگردیم، البته در sample.pcap چیزی پیدا نشد، پس تا اینجا چیزی یافت نشد.
 مهاجم در ادامه میتواند دنبال ارتباط عجیب که برقرار شده بگردد، مثلا فلگ های عجیب، مثل syn بدون ack، یا ببیند کجا مقدار Retransmission زیاد بوده، ممکن است اینجاها بتواند راهی برای نفوذ پیدا کند.

با استفاده از دیدن استریم بین 2 نقطه، مهاجم ممکن است که بتواند به اطلاعاتی دوباره دست پیدا کند (برای مثال محتوای قسمت Data)

بعد از همه این ها باز هم مهاجم برای مثال میتواند آدرس های IP داخلی و mac ها را پیدا کند، یا برای مثال توپولوژی شبکه را ترسیم کند. در آخر هم میتونه دنبال پورت های مشکوک باشه، مثلا 515 که بالانتر گفته شد (Line Printer Daemon – LPD) ممکن است بتوان از طریق آن نفوذ کرد.

لینک های استفاده شده

<https://gemini.google.com/app/bd4d535952099ed7>

<https://gemini.google.com/app/a944982a0776fe02>

<https://chatgpt.com/c/68358a34-282c-8005-a758-87a6ccffbe93>

Ai:

ایل PCAP اصلی را به صورت دستی با استفاده از Wireshark تحلیل کرده و بیان کنید آیا اطلاعات دیگری در Trace مربوطه وجود دارد که شخص مهاجم بتواند به نحوی از آن استفاده کند؟ درباره راههای استفاده از آن توسط شخص مهاجم و آسیبپذیریهای احتمالی توضیح دهید؛ همچنین سناریوی حمله با استفاده از این آسیبپذیریها را در گزارش خود تحلیل کنید. (توجه داشته باشید این سوال میتواند جوابهای بسیار متفاوتی داشته باشد و نکته حائز اهمیت درک و تحلیل شما از Trace یک سناریوی تبادل اطلاعات در شبکه است)

خب من وایرشارکو نصب کردم. الان این کارایی که این میگه رو چجوری باید انجام بدم؟

You're on the right track by installing Wireshark! The prompt asks you to manually analyze a PCAP file with Wireshark to identify potential information an attacker could use. Here's a breakdown of how to approach this:

Steps to Analyze a PCAP File with Wireshark for Potential Attacker Use:

Obtain a PCAP File: First, you need a .pcap or .pcapng file to analyze. If you don't have one, you can often find sample PCAP files online for practice (search for "sample pcap files"). .1

Open the PCAP File in Wireshark: .2

Launch Wireshark. ○

Go to "File" > "Open" and select your PCAP file. ○

Explore the Trace: Wireshark will display the captured network packets. Each row represents a packet, showing information like: .3

No.: Packet number. ○

Time: Timestamp of the packet. ○

Source: IP address and port of the sender. ○

Destination: IP address and port of the receiver. ○

Protocol: Network protocol used (e.g., TCP, UDP, HTTP). ○

Length: Size of the packet. ○

Info: Summary of the packet content. ○

Identify Potential Sensitive Information: This is the core of the analysis. You need to look for patterns or specific packets that might reveal sensitive data. Consider looking for: .4

Unencrypted Credentials: Look for protocols like HTTP (without HTTPS/TLS) where usernames and passwords might be transmitted in plain text. Filter for http and then examine the packet details (right-click on a packet and select "Follow" > "TCP Stream" or "HTTP"). ○

Personal Data: Information like names, email addresses, addresses, or other identifying details transmitted without encryption. ○

Session Tokens/Cookies: These can sometimes be captured and potentially reused by an attacker. Look for HTTP packets containing Set-Cookie or Cookie headers. ○

Internal Network Information: Details about internal IP addresses, server names, or network configurations that could aid reconnaissance. ○

Vulnerable Protocols or Services: Evidence of outdated or insecure protocols being used. ○

Error Messages: Sometimes error responses can inadvertently reveal information. ○

Analyze Potential Attacker Use and Vulnerabilities: Once you identify potentially interesting information, think about how an attacker could exploit it: .5

Credential Stuffing/Account Takeover: If you find unencrypted credentials. ○

Session Hijacking: If session tokens are captured. ○

Information Gathering: Using internal network details to map the network for further attacks. ○

Exploiting Vulnerable Services: If you see traffic related to known vulnerable services. ○

Develop an Attack Scenario: Based on your findings, create a hypothetical scenario where an attacker leverages the identified information or vulnerabilities. Describe the steps the attacker might take and the potential impact. .6

How to Proceed with Wireshark:

Filtering: Use Wireshark's display filters to narrow down the traffic. For example: •

http: To see HTTP traffic. ○

tcp.port == 80: To see TCP traffic on port 80. ○

ip.addr == 192.168.1.100: To see traffic involving the IP address 192.168.1.100. ○

Following Streams: Right-click on a packet and select "Follow" > "TCP Stream" (or UDP/HTTP stream) to see the entire conversation between two hosts. This is very useful for examining application-level data. •

Examining Packet Details: Select a packet and look at the three panes below the packet list: •

Frame: Physical layer information. ○

Ethernet II: Data link layer information. ○

Internet Protocol Version 4 (or 6): Network layer information (IP addresses). ○

Transmission Control Protocol (or UDP): Transport layer information (ports).

Application Layer Protocol (e.g., HTTP, DNS): The actual data being transmitted.

Your Task:

To answer the original prompt, you would need to:

1. Open a PCAP file in Wireshark.
2. Analyze the packets looking for the types of information mentioned above.
3. Describe what information you found that an attacker could use.
4. Explain how an attacker might use this information and what the potential vulnerabilities are.
5. Outline a hypothetical attack scenario leveraging these vulnerabilities.

Since you've just installed Wireshark, your next step is to **obtain a PCAP file** and start exploring it using the methods I've described.

Do you have a PCAP file you'd like to analyze, or would you like me to help you find a sample one?

- من میدونم که آدرس 128.3.164.248 مشکوکه، ی پکت رنگ زرشکی مشکی ازش پیدا کردم این شد، نظرت

Suspicious Source Address: You've correctly identified 128.3.164.248 as چیه potentially suspicious. The fact that you found a packet from this address highlighted in a dark color (likely indicating an expert info warning or error in Wireshark) reinforces this suspicion.

- **TCP Protocol:** The packet uses the TCP protocol.

- **Ports:** It involves source port 113 and destination port 57549. These ports themselves aren't inherently malicious, but their combination in a suspicious packet warrants further investigation.

- **Flags: RST, ACK:** The TCP flags show RST (Reset) and ACK (Acknowledgement) are set. A RST flag typically indicates an abrupt termination of a TCP connection. This could be due to various reasons, including a connection error, a firewall rejecting the connection, or even a probe from a potentially malicious host.

- **Sequence and Acknowledgment Numbers:** The sequence number is 1, and the acknowledgment number is also 1.

Expert Info: Connection reset [RST]: This is a significant finding. Wireshark's "Expert Info" explicitly flags this packet as a "Connection reset [RST]," which, as mentioned earlier, suggests a problem with the connection.

Potential Implications for an Attacker:

While this single RST/ACK packet doesn't immediately reveal exploitable data, it *could* be part of a reconnaissance or attack attempt:

Port Scanning: An attacker might send packets to various ports on a target system to see which ones are open or elicit a response. An RST packet in response to a SYN (which we don't see here in isolation) to a closed port is a common indicator in port scanning. While this packet is an RST/ACK, it could be a response to a previous communication attempt.

Denial of Service (DoS) Attempt: While a single RST packet isn't a DoS, a flood of such reset packets could potentially disrupt connections.

Evasion/Disruption: In some cases, an attacker might try to inject RST packets to prematurely close legitimate connections.

Vulnerabilities and Attack Scenario:

Based on this single packet, we can't pinpoint a specific vulnerability. However, we can hypothesize a scenario:

Scenario:

1. The host at 128.3.164.248 (the potential attacker) might have been attempting to connect to the target system (the other IP in the trace, which isn't fully visible in these snippets) on port 57549 (or perhaps other ports).
2. The RST/ACK packet from the target (or an intermediary like a firewall) indicates that the connection was refused or reset.
3. The attacker could be using this information to map open ports on the target system. If they find an open and potentially vulnerable service on another port, they might then attempt to exploit it.
4. Alternatively, if the attacker sent a series of these RST packets to established connections, they might try to disrupt network communication.

To further analyze the potential threat, we would need to look at:

- **The packets preceding this one:** What initiated this RST/ACK? Was there a SYN packet from the suspicious IP?

Other traffic from/to 128.3.164.248: Are there other attempts to connect to different ports?

The context of the network traffic: What kind of services are running on the target system?

Your Next Steps:

1. **Can you filter the Wireshark trace for traffic involving the IP address 128.3.164.248?** This will show all communication attempts from or to this IP.

2. **Look at the packets immediately before and after the highlighted one.** This might give context to why the connection was reset.

Let me know what you find when you apply these filters!

نارو ویش کن. همون فایل pcap دارم. حالا میخوام ی فایل پایتون بنویسم که ی سری کارارو انجام بده.

Network ضبط ترافیک شبکه در یک بازه زمانی مشخص است که شامل بستههای ارسالی و دریافتی بین سیستمها

میشود و نتیجه آن به صورت فایل PCAP قابل دسترسی میباشد که میتواند شامل انواع بستههای شبکه مانند Ethernet, IP و TCP باشد.

در یکی از تکنیکهای scan port، که به عنوان scan SYN شناخته میشود، scanner (بستههای SYN TCP اولین بسته

در) handshake TCP را ارسال کرده و به دنبال پاسخهای ACK+SYN دومین مرحلهی (handshake TCP از سمت

میزبان میگردد. از آنجا که اکثر میزبانها آماده دریافت ارتباط در تمام Port ها نیستند، در یک scan SYN تعداد بستههای

ACK+SYN دریافتی معمولاً کمتر از بستههای SYN ارسالی شده است. با مشاهده این تفاوت در یک Network trace میتوان

مبدأ پکتهایی را که احتمال در حال انجام scanning port هستند، شناسایی کرد

برنامه باید فایل PCAP که شامل بستههای ضبطشده شبکه است را بخواند و تحلیل کند. تنها بستههای معتبر شامل Ethernet,

IP و TCP مورد بررسی قرار گیرند. همچنین بستههای Malformed یا بستههایی که از این پروتکلها استفاده نمیکند، باید

نادیده گرفته شوند

برنامه باید تعداد بستههای SYN ارسال شده توسط هر منبع (IP address) و تعداد بستههای ACK+SYN دریافتی

از هدف را محاسبه کند. همچنین در صورتی که یک IP بهطور مکرر (بیش از 3 برابر تعداد ACK+SYN دریافتی)

بستههای SYN ارسال کرده باشد، آن IP باید بهعنوان منبع احتمالی scan SYN شناسایی شود

برنامه شما باید یک آرگومان ورودی داشته باشد (نام فایل PCAP برای تحلیل) مطابق زیر:

python syn_scanner.py sample.pcap

خروجی نیز باید مجموعههای از IP address های مشکوک به صورت زیر طبق توضیحات 2. b باشد. (ترتیب IP ها ممکن است با

هر بار اجرا متفاوت باشد)

128.3.23.2

128.3.23.5

128.3.23.117

128.3.23.158

128.3.164.248

128.3.164.249

Got it. You want to write a Python script to analyze a PCAP file and identify potential SYN port scanners based on the logic you described. Here's a Python script that uses the scapy library to achieve this:

```
from scapy.all import rdpcap, TCP, IP
import sys
from collections import defaultdict

def syn_scanner(pcap_file):
    """ Analyzes a PCAP file to identify potential SYN port scanners.
    Args: pcap_file (str): The path to the PCAP file. Returns: list: A list of IP addresses suspected
    of performing SYN scans. """
    try:
        packets = rdpcap(pcap_file)
    except FileNotFoundError:
        print(f"Error: File not found: {pcap_file}")
        return []

    syn_sent_count = defaultdict(int)
    syn_ack_received_count = defaultdict(int)

    for packet in packets:
        if IP in packet and TCP in packet:
            src_ip = packet[IP].src
            dst_ip = packet[IP].dst
            tcp_flags = packet[TCP].flags

            # Check for SYN sent
            if tcp_flags == 'S':
                syn_sent_count[src_ip] += 1

            # Check for SYN-ACK received
            if tcp_flags == 'SA':
                syn_ack_received_count[dst_ip] += 1

    suspect_scanners = []

    for ip, syn_count in syn_sent_count.items():
        ack_count = syn_ack_received_count.get(ip, 0)

        # Ensure some response was received
        if syn_count > 3 * ack_count and ack_count > 0:
            suspect_scanners.append(ip)

        # Aggressive scanner with no responses
        elif syn_count > 10 and ack_count == 0:
            suspect_scanners.append(ip)

    return sorted(list(set(suspect_scanners)))


if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Usage: python syn_scanner.py <pcap_file>")
        sys.exit(1)

    pcap_file = sys.argv[1]
    suspects = syn_scanner(pcap_file)

    if suspects:
        print("\nPotential SYN port scanners:")
        for ip in suspects:
            print(ip)
    else:
        print("\nNo potential SYN port scanners detected based on the criteria.")
```

```
PS C:\Users\Houman\Desktop\ECS-CA3-810100210> python syn_scanner.py
reduced_sample.pcap
```

```
C:\Users\Houman\AppData\Local\Programs\Python\Python313\python.exe: can't open file
'C:\Users\Houman\Desktop\ECS-CA3-810100210\syn_scanner.py': [Errno 2] No such file
or directory
```



It looks like the command prompt is in the correct directory (C:\Users\Houman\Desktop\ECS-CA3-810100210), but Python can't find the syn_scanner.py file.

Please double-check that the syn_scanner.py file is indeed saved in the C:\Users\Houman\Desktop\ECS-CA3-810100210 directory.

You can do this by opening File Explorer and navigating to that directory to confirm the file exists.

Once you've confirmed the file is there, try running the command again.

Let me know what you find!

Okay, I see the problem in the screenshot. You have the syn_scanner.py file open in your editor, which confirms it exists in that directory.

The issue is likely how you are executing the Python command. You are using:

PowerShell

```
python syn_scanner.py reduced_sample.pcap
```

This command assumes that the Python interpreter (python) is in your system's PATH environment variable. While it often is, it seems in your environment, you might need to be more explicit about the path to the Python executable.

You can try running the script using the full path to your Python interpreter, which the error message suggests is:

PowerShell

```
C:\Users\Houman\AppData\Local\Programs\Python\Python313\python.exe syn_scanner.py  
redu
```

Okay, I see the problem in the screenshot. You have the syn_scanner.py file open in your editor, which confirms it exists in that directory.

The issue is likely how you are executing the Python command. You are using:

PowerShell

```
python syn_scanner.py reduced_sample.pcap
```

This command assumes that the Python interpreter (python) is in your system's PATH environment variable. While it often is, it seems in your environment, you might need to be more explicit about the path to the Python executable.

You can try running the script using the full path to your Python interpreter, which the error message suggests is:

PowerShell

```
C:\Users\Houman\AppData\Local\Programs\Python\Python313\python.exe syn_scanner.py
reduced_sample.pcap
```

Please try running this exact command in your PowerShell window and let me know the output.

```
PS C:\Users\Houman\Desktop\ECS-CA3-810100210>
```

```
C:\Users\Houman\AppData\Local\Programs\Python\Python313\python.exe syn_scanner.py
reduced_sample.pcap
```

```
C:\Users\Houman\AppData\Local\Programs\Python\Python313\python.exe: can't open file
'C:\Users\Houman\Desktop\ECS-CA3-810100210\syn_scanner.py': [Errno 2] No such file
or directory
```

That's still the same "No such file or directory" error. This is quite strange since the file clearly exists in the directory you're in according to the previous screenshot.

Let's try a slightly different approach to ensure the script is being found. Instead of just the filename, let's try using a relative path that explicitly points to the file in the current directory. Use this command:

PowerShell

```
C:\Users\Houman\AppData\Local\Programs\Python\Python313\python.exe
.\syn_scanner.py reduced_sample.pcap
```

The .\ at the beginning tells the command prompt to look for the file in the current directory.

Please try this command and let me know if it works.

دی که دادی خوبه، این خروجی رو داده :

Potential SYN port scanners:

128.3.23.117

128.3.23.158

128.3.23.2

128.3.23.5

در حالی که باید چنتا دیگه ام میداشت :

128.3.164.248

128.3.164.249

اگه چنتا اسکرین شات برات از وایرشارک چنتا از پکتای این 2 تا بگیرم کمک میکنه بهت؟

That's interesting. The script identified some potential scanners, but missed the two you expected. Seeing screenshots of Wireshark focusing on the traffic from/to 128.3.164.248 and 128.3.164.249 would definitely be helpful!

Please share the screenshots, especially ones that show:

- Packets originating from these two IP addresses.
- The TCP flags in those packets (SYN, ACK, SYN-ACK, RST, etc.).
- The sequence and acknowledgment numbers, if relevant.

With that information, we can analyze why the script might not have flagged them and potentially adjust the logic. Looking forward to seeing the Wireshark captures!

Packet 183: 128.3.164.249 → 128.3.23.42 with flags SYN. This is a SYN packet being sent.

Packet 184: 128.3.23.42 → 128.3.164.249 with flags RST, ACK. This is a Reset and Acknowledgment in response.

Packet 185: 128.3.164.249 → 128.3.23.42 with flags PSH, ACK. This is Push and Acknowledgment.

Packet 186: 128.3.23.42 → 128.3.164.249 with flags FIN, ACK. This is Finish and Acknowledgment.

Packet 187: 128.3.23.42 → 128.3.164.249 with flags RST. Another Reset.

Based on this short sequence:

- 128.3.164.249 sent a SYN.
- It received a RST, ACK back.
- Then it sent a PSH, ACK.
- Then it received a FIN, ACK.

Finally, it received another RST. •

This doesn't look like a typical SYN scan pattern in isolation. A SYN scan usually involves sending many SYN packets to different ports and looking for SYN-ACK responses. Here, we see a SYN followed by a reset, then some data transfer (PSH, ACK), and then a connection termination (FIN, ACK and RST).

To understand why your script didn't flag 128.3.164.249, we need to see more of its behavior. It's possible that the total number of SYN packets sent by this IP, compared to the SYN-ACK responses it received, didn't meet the threshold in the script (`syn_count > 3 * ack_count` and `ack_count > 0`, or `syn_count > 10` and `ack_count == 0`).

Could you capture a wider timeframe or filter in Wireshark specifically for `tcp.flags.syn == 1` originating from 128.3.164.249 and also `tcp.flags.syn_ack == 1` destined to it? This might give us a better overview of its scanning attempts.

What about 128.3.164.248? Do you have a similar screenshot for that IP?

ببین این آیا کمکی میکنه

Yes, this is helpful! This screenshot shows packets originating from 128.3.164.249 with the SYN flag set:

Packet 26: 128.3.164.249 → 128.3.23.42 with [SYN]. •

Packet 28: 128.3.164.249 → 128.3.23.42 with [SYN, ACK]. This is a SYN-ACK, which is a response to a SYN. •

Packet 29: 128.3.164.249 → 128.3.23.42 with [SYN]. Another SYN. •

So, for 128.3.164.249 to 128.3.23.42, we see two SYN packets sent and one SYN-ACK received in this view.

To determine if it should be flagged as a scanner, we'd need to see the total count of SYN packets sent by 128.3.164.249 to all destinations and the total count of SYN-ACK packets received by it.

Do you have a similar filter applied but showing all traffic involving 128.3.164.249 so we can see the broader picture of its SYN attempts and responses? You could use a filter like `ip.addr == 128.3.164.249` and `tcp.flags.syn == 1` to see all SYN packets from this IP, and then another filter like `ip.dst == 128.3.164.249` and `tcp.flags.syn_ack == 1` to see all SYN-ACK packets destined to it.

What about the traffic for 128.3.164.248? Seeing similar information for that IP would also be valuable.

همچین چیزی فک نکنم کار کنه تو فیلتر وایرشارک، بهترش کن
`ip.dst == 128.3.164.249 and tcp.flags.syn_ack == 1`

You are absolutely right! My apologies, that was an incorrect filter.

The correct Wireshark filter to see SYN-ACK packets destined to 128.3.164.249 would be:

`ip.dst == 128.3.164.249 and tcp.flags.syn == 1 and tcp.flags.ack == 1`

Alternatively, you can just use:

`ip.dst == 128.3.164.249 and tcp.flags.sa == 1`

Would you be able to apply this filter and share a screenshot of the result? Similarly, could you do the same for the source IP 128.3.164.249 looking for all SYN packets it sent using the filter `ip.src == 128.3.164.249 and tcp.flags.syn == 1`?

Once we have this information, we can better understand why the Python script might have missed this IP.

اینو `ip.dst == 128.3.164.249 and tcp.flags.syn == 1 and tcp.flags.ack == 1` که زدم هیچی نیارده

Okay, that's good to know. If that filter returned no results, it means there were no packets in the trace where the destination was 128.3.164.249 and both the SYN and ACK flags were set.

Let's try to refine our understanding. Could you apply the following filters and share screenshots of the results?

SYN packets from 128.3.164.249: .1

`ip.src == 128.3.164.249 and tcp.flags.syn == 1` .2

All TCP packets involving 128.3.164.249: .3

`ip.addr == 128.3.164.249 and tcp` .4

These will help us see the total number of SYN attempts and the overall TCP communication of this IP address.

Once we have this, we can compare the number of SYN packets sent with the number of SYN-ACK (if any) received to see if it fits the criteria of a SYN scanner according to your script's logic.

Also, don't forget to do the same for 128.3.164.248 if you want to investigate why it wasn't flagged either.

Okay, let's analyze these screenshots for 128.3.164.249:

Screenshot 1 (Filter: `ip.src == 128.3.164.249` and `tcp.flags.syn == 1`):

This shows SYN packets originating from 128.3.164.249:

Packet 27: To 128.3.23.42 (SYN, ACK - this is a SYN-ACK being *sent*, which is unusual for the initiator of a connection). •

Packet 29: To 128.3.23.42 (SYN). •

So, in this view, we see two SYN attempts (one combined with ACK).

Screenshot 2 & 3 (Filter: `ip.addr == 128.3.164.249` and `tcp`):

These show all TCP traffic involving 128.3.164.249. Looking through these, we can try to count the total SYN packets sent and any SYN-ACK packets received by 128.3.164.249.

From the combined view, focusing on 128.3.164.249 as the source:

We see SYN packets in frames 27 and 29. •

We also see a SYN in frame 30 going to 128.3.23.42. •

Now looking for SYN-ACK packets where 128.3.164.249 is the destination: I don't immediately see any SYN-ACK packets destined for it in these views. Frame 28 shows a [SYN, ACK] coming *from* 128.3.164.249.

It seems 128.3.164.249 is sending SYN packets, but we don't see many (or any in this limited view) SYN-ACK responses coming back to it. This behavior aligns with a potential SYN scanner.

Let's reconsider the logic in your Python script:

Python

```
if syn_count > 3 * ack_count and ack_count > 0: suspect_scanners.append(ip) elif syn_count > 10 and ack_count == 0: suspect_scanners.append(ip)
```

It checks if the number of SYN packets sent is significantly higher than the SYN-ACK packets received, or if a high number of SYN packets are sent with no responses.

To make the script identify 128.3.164.249, we need to ensure that `syn_sent_count['128.3.164.249']` and `syn_ack_received_count['128.3.164.249']` are being correctly populated. The script currently counts SYN-ACKs where the *destination* IP is the

Let's refine the script to correctly track this. I'll provide an updated version.

```

from scapy.all import rdpcap, TCP, IP
import sys
from collections import defaultdict

def syn_scanner(pcap_file)
    """
    .Analyzes a PCAP file to identify potential SYN port scanners

    :Args
    .pcap_file (str): The path to the PCAP file

    :Returns
    .list: A list of IP addresses suspected of performing SYN scans
    """
    try:
        packets = rdpcap(pcap_file)
    except FileNotFoundError:
        print(f"Error: File not found: {pcap_file}")
    return

syn_sent_count = defaultdict(int)
syn_ack_received_from = defaultdict(lambda: defaultdict(int)) # Count SYN-ACKs
received per source IP

```




```
return sorted(list(set(suspect_scanners)))
```

```
:"__if __name__ == "__main
```

```
:if len(sys.argv) != 2
```

```
print("Usage: python syn_scanner.py <pcap_file>")
```

```
(1)sys.exit
```

```
[1]pcap_file = sys.argv
```

```
suspects = syn_scanner(pcap_file)
```

```
:if suspects
```

```
print("\nPotential SYN port scanners:")
```

```
:for ip in suspects
```

```
print(ip)
```

```
:else
```

```
print("\nNo potential SYN port scanners detected based on the criteria.")
```

You're right, let's get you the updated code. Here's a revised Python script that should more accurately identify SYN scanners by looking at SYN packets sent and the corresponding SYN-ACK responses received by the potential scanner:

```
Python
```

```
from scapy.all import rdpcap, TCP, IP
```

```
import sys
```

```
from collections import defaultdict
```

```
def syn_scanner(pcap_file):
```

```
"""
```

Analyzes a PCAP file to identify potential SYN port scanners.

pcap_file (str): The path to the PCAP file.

list: A list of IP addresses suspected of performing SYN scans.

try:

except FileNotFoundError:

```
return []
```

```
syn_ack_received_from = defaultdict(lambda: defaultdict(int)) # Count SYN-ACKs received per source IP
```

if IP in packet and TCP in packet:

```
dst_ip = packet[IP].dst
```

```
tcp_flags = packet[TCP].flags
```

```
if tcp_flags == 'S':
```

```
syn_sent_count[src_ip] += 1
```

```

# Check for SYN-ACK received in response to a SYN we might have sent
    if tcp_flags == 'SA':
        # If the destination of the SYN-ACK was a source that sent a SYN
        # we increment the count of SYN-ACKs received by that source
        for pkt in packets:
            if IP in pkt and TCP in pkt and pkt[IP].dst == src_ip and pkt[IP].src == dst_ip and
                pkt[TCP].flags == 'S':
                syn_ack_received_from[src_ip][dst_ip] += 1
        break # Assuming one SYN-ACK per SYN for simplicity

    suspect_scanners = []
    for sender_ip, syn_count in syn_sent_count.items():
        total_ack_received = sum(syn_ack_received_from[sender_ip].values())
        if syn_count > 3 * total_ack_received and total_ack_received > 0:
            suspect_scanners.append(sender_ip)
        elif syn_count > 10 and total_ack_received == 0:
            suspect_scanners.append(sender_ip)

    return sorted(list(set(suspect_scanners)))

if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Usage: python syn_scanner.py <pcap_file>")
        sys.exit(1)

    pcap_file = sys.argv[1]
    suspects = syn_scanner(pcap_file)

```

```

        if suspects:
            print("\nPotential SYN port scanners:")
            for ip in suspects:
                print(ip)
            else:
                print("\nNo potential SYN port scanners detected based on the criteria.")

```

Key Change:

We now use `syn_ack_received_from` which is a nested dictionary to count SYN-ACKs received by each source IP, categorized by the IP that sent the SYN-ACK.

The logic for identifying suspects now compares the total SYN packets sent by an IP with the total SYN-ACK packets received by that IP.

Okay, that means the current logic still isn't quite capturing the behavior of 128.3.164.248 and 128.3.164.249 as SYN scanners according to the defined criteria.

Let's think about why this might be. Based on the Wireshark screenshots, we saw SYN packets being sent by these IPs, but perhaps the ratio of SYN sent to SYN-ACK received isn't high enough within the entire capture for the script's thresholds.

To help me refine the script further, could you tell me:

1. Roughly how many SYN packets do you see originating from 128.3.164.248 and 128.3.164.249 in the entire PCAP file when you filter for `ip.src == 128.3.164.248 and tcp.flags.syn == 1` and `ip.src == 128.3.164.249 and tcp.flags.syn == 1` respectively?

2. For each of these IPs, do you see any SYN-ACK packets coming back to them (filter `ip.dst == 128.3.164.248 and tcp.flags.syn_ack == 1` and `ip.dst == 128.3.164.249 and tcp.flags.syn_ack == 1`)? If so, approximately how many?

This information will help me understand if we need to adjust the thresholds in the script or rethink the logic for identifying a SYN scanner based on this specific trace.

برای سوال یک که ازم پرسیدی: اونی که 2 تا پکته برای 249 عه و اونی که 8 تاس برای 248 عه

Okay, thanks for that information. So:

- 128.3.164.249 sent approximately 2 SYN packets.

- 128.3.164.248 sent approximately 8 SYN packets.

Now, for the second part of my question: When you filter for SYN-ACK packets destined to these IPs (ip.dst == 128.3.164.248 and tcp.flags.sa == 1 and ip.dst == 128.3.164.249 and tcp.flags.sa == 1), how many do you see for each?

Knowing the number of SYN-ACK responses will help us understand if the ratio meets the criteria in the script.

واپرشارک ip.dst رو قبول نمیکند توسیستم

You are absolutely correct! My apologies for the repeated mistake. Wireshark uses ip.dst for the destination IP address.

So, the correct filters would be:

For SYN-ACK destined to 128.3.164.248:

ip.dst == 128.3.164.248 and tcp.flags.sa == 1

For SYN-ACK destined to 128.3.164.249:

ip.dst == 128.3.164.249 and tcp.flags.sa == 1

Could you please apply these two filters and let me know the count of packets you see for each? This will give us the number of SYN-ACK responses received by these potential scanners.

it says that neither tcp.flags.sa nor 1 are field or protocol names.

You are right again! I am making repeated errors with the Wireshark filter syntax. My apologies.

The correct way to filter for SYN-ACK flags in Wireshark is to check if both the SYN (S) and ACK (A) bits are set. You can do this by checking the numerical value of the flags. SYN has a value of 0x02, and ACK has a value of 0x10. When both are set, the value is 0x12.

So, the correct filters are:

For SYN-ACK destined to 128.3.164.248:

ip.dst == 128.3.164.248 and tcp.flags == 0x12

For SYN-ACK destined to 128.3.164.249:

ip.dst == 128.3.164.249 and tcp.flags == 0x12

Could you please try these filters and let me know the number of packets you see for each?

Thank you for your patience in correcting my mistakes!

مطمئنم این فیلتر آخرکه دادی درست کار میکنه؟ من جفتشو زدم و جفتش هیچی آورد!

You are absolutely right to question that! My apologies, I'm still making mistakes with the Wireshark filter syntax.

Let's try this one more time with what should be the correct syntax for checking multiple TCP flags:

For SYN-ACK destined to 128.3.164.248:

`ip.dst == 128.3.164.248 and tcp.flags.syn == 1 and tcp.flags.ack == 1`

For SYN-ACK destined to 128.3.164.249:

`ip.dst == 128.3.164.249 and tcp.flags.syn == 1 and tcp.flags.ack == 1`

These filters explicitly check if both the SYN flag and the ACK flag are set for packets destined to the IPs in question.

Could you please try these exact filters and let me know if they return any results? I appreciate your persistence in getting this right!

وقتی میزنم `ip.dst == 128.3.164.249 and tcp.flags.syn == 1` کلا ی نتیجه مباد.

وقتی میزنم `ip.dst == 128.3.164.249 and tcp.flags.syn == 1 and tcp.flags.ack == 1` دیگه هیچ نتیجه ای نمیداد اصن.

وقتی میزنم `ip.dst == 128.3.164.248 and tcp.flags.syn == 1` 4 تا نتیجه مباد

وقتی میزنم `ip.dst == 128.3.164.248 and tcp.flags.syn == 1 and tcp.flags.ack == 1` دیگه هیچی نمیداد

Thank you for running those filters and providing the counts. This is very insightful!

Here's what we've learned:

For 128.3.164.249: There is one packet where it's the destination and the SYN flag is set, but none where it's the destination and both SYN and ACK are set. •

For 128.3.164.248: There are four packets where it's the destination and the SYN flag is set, but none where it's the destination and both SYN and ACK are set. •

Now let's revisit the number of SYN packets sent by these IPs (from your earlier feedback):

128.3.164.249 sent approximately 2 SYN packets. •

128.3.164.248 sent approximately 8 SYN packets. •

Considering the logic in our Python script:

Python

```

if syn_count > 3 * total_ack_received and total_ack_received > 0:
    suspect_scanners.append(sender_ip)
elif syn_count > 10 and total_ack_received == 0:
    suspect_scanners.append(sender_ip)

```

For 128.3.164.249: `syn_count` is around 2, and `total_ack_received` (SYN-ACK where it's the *destination*) is 0. Neither condition is met ($2 > 3 * 0$ is true, but `total_ack_received` is not > 0 ; and $2 > 10$ is false).

For 128.3.164.248: `syn_count` is around 8, and `total_ack_received` (SYN-ACK where it's the *destination*) is 0. Again, neither condition is strongly met based on the second condition ($8 > 10$ is false).

It seems the script's logic, which focuses on SYN-ACK packets *received by* the potential scanner as the *destination*, isn't directly correlating with the behavior you're observing where these IPs seem to be initiating connections (sending SYN packets) without getting SYN-ACK responses back.

We need to adjust the script to count SYN-ACK packets where the potential scanner was the *source* of the initial SYN.

Let's try a modified script. This one will count SYN packets sent by an IP and SYN-ACK packets where that IP was the *source* of the SYN that elicited the SYN-ACK.

```

Python
from scapy.all import rdpcap, TCP, IP
import sys
from collections import defaultdict

def syn_scanner(pcap_file):
    try:
        packets = rdpcap(pcap_file)
    except FileNotFoundError:
        print(f"Error: File not found: {pcap_file}")
    return []

```



```
syn_sent_count = defaultdict(int)
syn_ack_received_response_to = defaultdict(int)

syn_sent_ips_ports = {} # To track (src_ip, sport, dst_ip, dport, seq) of sent SYNs

    for packet in packets:
        if IP in packet and TCP in packet:
            src_ip = packet[IP].src
            dst_ip = packet[IP].dst
            sport = packet[TCP].sport
            dport = packet[TCP].dport
            seq = packet[TCP].seq
            tcp_flags = packet[TCP].flags

            # Count SYN packets sent
            if tcp_flags == 'S':
                syn_sent_count[src_ip] += 1
                syn_sent_ips_ports[(src_ip, sport, dst_ip, dport, seq)] = True

        # Count SYN-ACK packets received as a response
        if tcp_flags == 'SA':
            ack = packet[TCP].ack - 1 # The ack number in SYN-ACK is usually seq + 1
            for sent_syn in list(syn_sent_ips_ports.keys()):
                s_ip, s_port, d_ip, d_port, s_seq = sent_syn
                if src_ip == d_ip and dst_ip == s_ip and packet[TCP].ack == s_seq + 1:
                    syn_ack_received_response_to[dst_ip] += 1
```



```
del syn_sent_ips_ports[sent_syn] # Remove the matched SYN
                                break

                                suspect_scanners = []
                                for ip, syn_count in syn_sent_count.items():
                                    ack_count = syn_ack_received_response_to.get(ip, 0)
                                if syn_count > 3 * ack_count and ack_count >= 0: # Adjusted condition
                                    suspect_scanners.append(ip)
                                elif syn_count > 10 and ack_count == 0:
                                    suspect_scanners.append(ip)

                                return sorted(list(set(suspect_scanners)))

                                if __name__ == "__main__":
                                    if len(sys.argv) != 2:
                                        print("Usage: python syn_scanner.py <pcap_file>")
                                        sys.exit(1)

                                    pcap_file = sys.argv[1]
                                    suspects = syn_scanner(pcap_file)

                                    if suspects:
                                        print("\nPotential SYN port scanners:")
                                        for ip in suspects:
                                            print(ip)
                                        else:
                                            print("\nNo potential SYN port scanners detected based on the criteria.")
```

Changes in this version:

- We now track sent SYN packets with their source/destination IPs, ports, and sequence numbers.
 - When we see a SYN-ACK, we try to match it back to a SYN we sent by checking the source/destination IPs and the acknowledgment number.
 - `syn_ack_received_response_to` now counts the SYN-ACKs received *in response to* SYNs sent by each IP.
 - The suspect identification logic now uses this count.
- Please save this new version and run it on your PCAP file. Let's see if it now identifies the IPs you expect.

Sources

فارسی میتونی حرف بزنی باهم؟

بله، حتماً. چطور میتونم کمکتون کنم؟ چه سوالی دارید یا چه کاری میخواید انجام بدیم؟

خب الان من باید گزارش بنویسم، ی گزارش کامل، از اینکه چیکارا کردم، چی شد به نتیجه رسیدم، دنبال چه چیزایی گشتم برای پیدا کردن آی پی های مشکوک و

بسیار عالی. پس شما میخواید یک گزارش کامل از روند تحلیل فایل PCAP و شناسایی IP های مشکوک به اسکن SYN تهیه کنید. در این گزارش باید مراحل کار، مشاهدات و منطق استفاده شده برای تشخیص این IP ها توضیح داده شود.

برای نوشتن این گزارش، می‌توانید ساختار زیر را دنبال کنید:

1. مقدمه:

- توضیح مختصری درباره فایل PCAP و اهمیت تحلیل ترافیک شبکه.
- هدف از این تحلیل: شناسایی منابع احتمالی اسکن پورت SYN.
- اشاره به تکنیک اسکن SYN و نحوه تشخیص آن در ترافیک شبکه (ارسال تعداد زیاد بسته SYN در مقایسه با پاسخ‌های SYN-ACK).

2. روش کار:

- ابزار مورد استفاده: Wireshark و اسکریپت پایتون.
- مراحل تحلیل دستی با Wireshark:
 - باز کردن فایل PCAP.
 - فیلتر کردن ترافیک برای بررسی IP های خاص (128.3.164.248 و 128.3.164.249).
 - بررسی فلگ‌های TCP (SYN)، ACK، (SYN-ACK).

- توضیح مشاهدات مربوط به تعداد بسته‌های SYN ارسالی و پاسخ‌های دریافتی برای این IP ها.

○ توضیح عملکرد اسکریپت پایتون (syn_scanner.py):

- نحوه خواندن فایل PCAP با استفاده از scapy.
- نحوه شمارش بسته‌های SYN ارسالی از هر IP.
- نحوه شمارش پاسخ‌های SYN-ACK دریافت شده در ازای بسته‌های SYN ارسالی.
- منطق شناسایی IP های مشکوک (تعداد زیاد SYN ارسالی در مقایسه با SYN-ACK دریافتی).

3. نتایج و مشاهدات:

- لیست IP های شناسایی شده به عنوان منابع احتمالی اسکن SYN توسط اسکریپت پایتون (128.3.23.117, 128.3.23.158, 128.3.23.2, 128.3.23.5).
- توضیح درباره IP های دیگری که در ابتدا مشکوک به نظر می‌رسیدند (128.3.164.248 و 128.3.164.249) و چرا اسکریپت در نهایت آن‌ها را شناسایی نکرد (بر اساس تعداد SYN ارسالی و SYN-ACK دریافتی طبق منطق اسکریپت).
- اشاره به یافته‌های تحلیل دستی با Wireshark در مورد این دو IP (تعداد SYN ارسالی و کمبود پاسخ SYN-ACK).

4. تحلیل و آسیب‌پذیری‌های احتمالی:

- توضیح اینکه چرا اسکن پورت SYN یک فعالیت مشکوک است (تلاش برای شناسایی سرویس‌های فعال روی یک میزبان).
- آسیب‌پذیری‌های احتمالی که ممکن است مهاجم پس از شناسایی پورت‌های باز از آن‌ها سوءاستفاده کند (بسته به سرویس‌های در حال اجرا).
- سناریوی حمله فرضی:
 - مهاجم با استفاده از اسکن SYN پورت‌های باز یک هدف را شناسایی می‌کند.
 - سپس از یک آسیب‌پذیری شناخته شده در یکی از سرویس‌های در حال اجرا روی آن پورت سوءاستفاده می‌کند (مثلاً با ارسال یک اکسپلویت).
 - نتیجه می‌تواند دسترسی غیرمجاز به سیستم هدف باشد.

5. نتیجه‌گیری:

- خلاصه‌ای از یافته‌های گزارش.
 - توصیه‌هایی برای نظارت بر ترافیک شبکه و شناسایی فعالیت‌های مشکوک مانند اسکن پورت.
- آیا این ساختار برای گزارش شما مناسب است؟ اگر بخش خاصی نیاز به جزئیات بیشتر دارد، بفرمایید تا با هم مرور کنیم.

بیشتر توضیح بده، یعنی خودت بنویس که من حالا هرچقدر خودم خواستم ادیت بزنم باشه، من یک پیش‌نویس از گزارش شما تهیه می‌کنم. شما می‌توانید بعداً آن را ویرایش و تکمیل کنید.

گزارش تحلیل فایل PCAP و شناسایی منابع احتمالی اسکن پورت SYN

1. مقدمه

فایل‌های PCAP حاوی اطلاعات مربوط به ترافیک شبکه ضبط شده در یک بازه زمانی مشخص هستند. تحلیل این فایل‌ها می‌تواند به شناسایی فعالیت‌های غیرعادی و تهدیدات امنیتی کمک کند. هدف از این گزارش، تحلیل فایل PCAP ارائه شده و شناسایی آدرس‌های IP است که احتمالاً در حال انجام اسکن پورت SYN بوده‌اند.

اسکن پورت SYN یکی از تکنیک‌های مورد استفاده برای کشف پورت‌های باز روی یک سیستم هدف است. در این روش، اسکنر بسته‌های TCP SYN را به پورت‌های مختلف ارسال می‌کند. پاسخ‌های SYN-ACK نشان‌دهنده باز بودن پورت هستند، در حالی که پاسخ‌های RST نشان‌دهنده بسته بودن پورت می‌باشند. ارسال تعداد زیادی بسته SYN بدون دریافت متناسب پاسخ SYN-ACK می‌تواند نشانه‌ای از یک اسکن پورت باشد.

2. روش کار

برای انجام این تحلیل، از دو روش استفاده شد: تحلیل دستی با استفاده از نرم‌افزار Wireshark و تحلیل خودکار با استفاده از یک اسکریپت پایتون.

2.1. تحلیل دستی با Wireshark

فایل PCAP در نرم‌افزار Wireshark باز شد. در ابتدا، به بررسی کلی ترافیک پرداخته شد و سپس ترافیک مربوط به آدرس‌های IP مشکوک (128.3.164.248 و 128.3.164.249) با استفاده از فیلترها مورد بررسی دقیق‌تر قرار گرفت. فیلترهای مورد استفاده برای بررسی فلگ‌های SYN (TCP SYN و SYN-ACK) به منظور ارزیابی رفتار این IP‌ها در ارسال بسته‌های SYN و دریافت پاسخ‌ها اعمال شدند.

مشاهدات نشان داد که این IP‌ها تعدادی بسته SYN ارسال کرده‌اند، اما تعداد پاسخ‌های SYN-ACK دریافتی در مقایسه با تعداد SYN‌های ارسالی به نظر کم بود. با این حال، تحلیل دقیق‌تر برای تعیین قطعی اینکه آیا این رفتار ناشی از اسکن پورت است یا دلایل دیگر، نیازمند بررسی کل ترافیک و نسبت دقیق این بسته‌ها بود.

2.2. تحلیل خودکار با اسکریپت پایتون

یک اسکریپت پایتون به منظور تحلیل خودکار فایل PCAP و شناسایی منابع احتمالی اسکن SYN توسعه داده شد. این اسکریپت با استفاده از کتابخانه scapy فایل PCAP را خوانده و بسته‌های Ethernet، IP و TCP را بررسی می‌کند.

مراحل عملکرد اسکریپت به شرح زیر است:

- شمارش تعداد بسته‌های SYN ارسالی از هر آدرس IP مبدأ.
- شمارش تعداد بسته‌های SYN-ACK دریافت شده توسط هر آدرس IP مبدأ در پاسخ به SYN‌های ارسالی آن.
- شناسایی IP‌هایی که تعداد بسته‌های SYN ارسالی آن‌ها به طور قابل توجهی بیشتر از تعداد بسته‌های SYN-ACK دریافتی باشد (با در نظر گرفتن آستانه‌های مشخص شده در اسکریپت: بیش از 3 برابر در صورت دریافت حداقل یک ACK، یا بیش از 10 SYN بدون هیچ ACK).

3. نتایج و مشاهدات

اجرای اسکریپت پایتون بر روی فایل PCAP، آدرس‌های IP زیر را به عنوان منابع احتمالی اسکن پورت SYN شناسایی کرد:

128.3.23.2

128.3.23.5

128.3.23.117

128.3.23.158

در مقابل، آدرس‌های IP 128.3.164.248 و 128.3.164.249 با استفاده از منطق فعلی اسکریپت به عنوان اسکنر شناسایی نشدند. این ممکن است به دلیل تعداد نسبتاً کمتر بسته‌های SYN ارسالی توسط آن‌ها در کل بازه زمانی ضبط شده، یا دریافت تعدادی پاسخ SYN-ACK بوده باشد که نسبت مورد نظر برای شناسایی به عنوان اسکنر را برآورده نکرده است.

تحلیل دستی با Wireshark نیز نشان داد که این دو IP بسته‌های SYN ارسال کرده‌اند، اما برای نتیجه‌گیری قطعی درباره اسکن پورت بودن فعالیت آن‌ها، نیاز به بررسی دقیق‌تر تعداد کل بسته‌ها و نسبت پاسخ‌ها در کل فایل PCAP است.

4. تحلیل و آسیب‌پذیری‌های احتمالی

اسکن پورت SYN معمولاً به عنوان مرحله‌ای از شناسایی قبل از انجام حملات مورد استفاده قرار می‌گیرد. مهاجم با شناسایی پورت‌های باز و سرویس‌های در حال اجرا، می‌تواند آسیب‌پذیری‌های مربوطه را جستجو کرده و از آن‌ها برای نفوذ به سیستم هدف استفاده کند.

سناریوی حمله فرضی:

1. یک مهاجم با استفاده از یک ابزار اسکن پورت SYN، آدرس‌های IP موجود در یک شبکه را اسکن می‌کند.
2. اسکنر پورت‌های باز و سرویس‌های در حال اجرا روی سیستم‌های هدف را شناسایی می‌کند.
3. مهاجم اطلاعات به دست آمده را برای یافتن آسیب‌پذیری‌های شناخته شده در سرویس‌های باز (مانند یک نسخه قدیمی از یک وب سرور یا یک سرویس شبکه آسیب‌پذیر) استفاده می‌کند.
4. سپس، مهاجم یک اکسپلویت متناسب با آسیب‌پذیری شناسایی شده را ارسال کرده و سعی می‌کند تا دسترسی غیرمجاز به سیستم هدف پیدا کند.

5. نتیجه‌گیری

در این تحلیل، با استفاده از یک اسکریپت پایتون، تعدادی آدرس IP به عنوان منابع احتمالی اسکن پورت SYN شناسایی شدند. تحلیل دستی با Wireshark نیز به بررسی رفتار IP‌های مشکوک پرداخت. شناسایی فعالیت‌های اسکن پورت می‌تواند گام مهمی در تشخیص زودهنگام تهدیدات امنیتی در شبکه باشد. توصیه می‌شود ترافیک شبکه به طور مداوم نظارت شده و الگوهای رفتاری غیرعادی برای شناسایی و پاسخ به موقع به تهدیدات احتمالی مورد بررسی قرار گیرند.

این یک پیش‌نویس اولیه است. شما می‌توانید بر اساس یافته‌های دقیق‌تر خود از Wireshark و جزئیات بیشتری از فایل PCAP، این گزارش را تکمیل و ویرایش کنید. آیا بخش خاصی هست که بخواهید بیشتر روی آن تمرکز کنیم یا تغییری ایجاد کنید؟

من ی فایل pcap دارم که تو وایرشارک باز کردم. آیا اطلاعات دیگری در Trace مربوطه وجود دارد که شخص مهاجم بتواند به نحوی از آن استفاده کند؟ درباره راههای استفاده از آن توسط شخص مهاجم و آسیبپذیریهای احتمالی توضیح دهید؛ همچنین سناریوی حمله با استفاده از این آسیبپذیریها را در گزارش خود تحلیل کنید. (توجه داشته باشید این سوال میتواند جوابهای بسیار متفاوتی داشته باشد و نکته حائز اهمیت درک و تحلیل شما از Trace یک سناریوی تبادل اطلاعات در شبکه است)

ChatGPT said:

برای پاسخ به این سؤال، نیاز است ابتدا بدانیم فایل pcap شامل چه نوع ترافیکی است، اما چون شما فایل را باز کردید و به دنبال تحلیل هستید، من یک پاسخ جامع و تحلیلی مبتنی بر سناریوهای رایج ارائه می‌دهم که نشان دهد یک مهاجم چگونه می‌تواند از اطلاعات موجود در یک فایل pcap سوءاستفاده کند. شما می‌توانید با توجه به محتوای دقیق فایل خود، این تحلیل را شخصی‌سازی کنید.

1. 📌 چه اطلاعاتی در فایل pcap قابل استخراج است؟

فایل pcap (packet capture) شامل داده‌های خام شبکه است که می‌تواند اطلاعات مختلفی در خود داشته باشد:

- آدرس‌های IP و MAC
- پورت‌های مقصد و مبدأ
- نوع پروتکل‌ها (TCP, UDP, HTTP, DNS, FTP, Telnet, ...)
- Session‌های فعال
- محتوای ارسال‌شده در صورت رمزنگاری نشده بودن (مثلاً username/password در Telnet یا FTP)
- اطلاعات مربوط به دسترسی به سرویس‌ها
- گاهی اوقات کوکی‌ها، توکن‌ها یا حتی فایل‌های کامل در صورت ارسال با HTTP

2. 🛠 آسیب‌پذیری‌ها و راه‌های سوءاستفاده توسط مهاجم

مهاجم می‌تواند از اطلاعات موجود در فایل pcap برای حمله استفاده کند. در زیر چند سناریوی محتمل بررسی شده:

سناریو 1: سرقت اطلاعات حساس (Credential Theft)

پیش‌نیاز:

در فایل pcap ترافیک رمزنگاری نشده از سرویس‌هایی مانند HTTP ، Telnet یا FTP موجود باشد.

اقدام مهاجم:

با استفاده از Wireshark یا ابزارهای تحلیل‌گر دیگر، مهاجم می‌تواند فیلتر زیر را اعمال کند:

wireshark

CopyEdit

ftp || telnet || http

و به دنبال رشته‌هایی مانند Authorization, username=, password=, login.

پیامد:

دسترسی کامل به سیستم یا سرویس هدف با استفاده از اعتبارنامه‌ها.

سناریو 2: Session Hijacking :

پیش‌نیاز:

وجود کوکی‌های Session یا توکن‌های احراز هویت در فایل pcap.

اقدام مهاجم:

استخراج کوکی یا توکن و استفاده از آن در مرورگر یا اسکریپت برای جعل هویت کاربر.

پیامد:

دسترسی به حساب کاربری بدون نیاز به رمز عبور.

سناریو 3: Network Reconnaissance: شناسایی زیرساخت)

پیش‌نیاز:

وجود داده‌های خام از چندین دستگاه در شبکه.

اقدام مهاجم:

- تحلیل IP ها و MAC ها برای شناسایی توپولوژی شبکه.
- استخراج اطلاعات از پروتکل‌های ARP ، DNS ، DHCP.

پیامد:

طراحی دقیق‌تر حملات هدفمند مثل ARP Spoofing یا DNS Poisoning.

سناریو 4: Replay Attack :

پیش‌نیاز:

در فایل pcap ، پروتکل‌هایی مثل Modbus, SNMP یا حتی Login های غیر رمزنگاری شده ذخیره شده باشند.

اقدام مهاجم:

بازپخش بسته‌های حساس (مثل فرمان‌های کنترلی یا Login Packet ها (با ابزارهایی مثل tcpreplay.

پیامد:

اجرای فرمان‌های خاص یا ورود به سیستم بدون احراز هویت معتبر.

سناریو 5: Exploitation Based on Service Detection :

پیش‌نیاز:

در فایل، بسته‌هایی مربوط به نسخه‌ی دقیق سرویس‌ها دیده شود (مثلاً Server: Apache/2.4.18 در HTTP).

اقدام مهاجم:

- استفاده از ابزارهایی مثل searchsploit یا exploit-db.
- یافتن آسیب‌پذیری‌های شناخته‌شده مربوط به نسخه سرویس.

پیامد:

اجرای کد از راه دور (RCE) ، حملات Denial of Service ، تزریق SQL و...

جمع‌بندی و تحلیل

میزان خطر پیامد	قابلیت مهاجم	مورد	
بحرانی	ورود غیرمجاز	بالا	سرقت اطلاعات احراز هویت
بالا	جعل هویت	متوسط تا بالا	Hijackingسشن
متوسط	حملات هدفمند	بالا	شناسایی زیرساخت
بالا	اجرای دستورات مجدد متوسط	Replay	
بحرانی	اجرای کد	بسته به نسخه و آسیب‌پذیری اکسلویت سرویس	

توصیه‌های دفاعی

- همیشه از پروتکل‌های رمزنگاری‌شده مانند HTTPS, SFTP, SSH استفاده کنید.

- استفاده از VPN برای جلوگیری از شنود در شبکه‌های ناامن.
- پاکسازی session tokens و credential ها از ترافیک.
- محدود کردن دسترسی به فایل‌های pcap و لاگ‌ها.
- استفاده از IDS/IPS برای شناسایی بازپخش بسته‌ها.

اگر بخواهید می‌توانم فایل pcap شما را هم تحلیل کنم و یک گزارش کامل بر اساس محتویات واقعی ارائه بدهم. کافی است فایل را آپلود کنید.

You said:

FTP, Telnet رمزگذاری ندارند درسته؟ برای همین دنبالشون می‌گردیم بین پکتا

ChatGPT said:

دقیقاً ☒ درست متوجه شدی!

پروتکل‌های FTP و Telnet هیچ‌گونه رمزنگاری ندارند، یعنی تمام اطلاعاتی که بین کلاینت و سرور رد و بدل میشه، از جمله:

- نام کاربری (username)
- رمز عبور (password)
- دستورات (commands)
- محتوای منتقل‌شده در (FTP)

همه به صورت متن ساده (plaintext) در شبکه قابل دیدن هستند.

🔍 چرا دنبالشون می‌گردیم؟

چون اگر توی فایل pcap وجود داشته باشن، مهاجم به راحتی می‌تونه:

- با یه فیلتر ساده مثل:

nginx

CopyEdit

ftp || telnet

توی Wireshark ببین که چه داده‌ای رد و بدل شده.

- خیلی وقت‌ها با جستجو کلمات کلیدی مثل:

pgsql

CopyEdit

USER

PASS

login

می‌تونه credential ها رو مستقیماً پیدا کنه.

✳️ مثال واقعی:

Telnet:

pgsql

CopyEdit

USER admin

PASS 123456

FTP:

sql

CopyEdit

USER student

PASS student123

اینها بدون هیچ رمزنگاری در فایل pcap ذخیره میشن.

🔒 راهکار امن‌تر

- به جای Telnet از SSH استفاده کن (کاملاً رمزنگاری شده)

- به جای FTP از SFTP یا FTPS استفاده کن

اگر فایل pcap شما حاوی ترافیک FTP یا Telnet باشه، قطعاً یک نقطه آسیب‌پذیر مهم محسوب میشه. می‌خوای با هم بررسیش کنیم؟ فقط کافیه آپلودش کنی.

You said:

http ام رمزگذاری نداره؟

ChatGPT said:

کاملاً درست می‌پرسی — و جوابش اینه:

✗ نه، پروتکل HTTP به‌تنهایی رمزنگاری نداره.

تفاوت HTTP و HTTPS 🔍

پروتکل	امن برای اطلاعات حساس؟ پورت پیش فرض رمزنگاری دارد؟		
HTTP	ندارد (متن ساده) ❌	80	نه ❌
HTTPS	دارد (با TLS/SSL) ✅	443	بله ✅

⚠️ چه چیزی در HTTP قابل دیدن است؟

اگر ترافیک HTTP را در یک فایل pcap بررسی کنی، می‌تونی موارد زیر را مستقیماً ببینی:

- آدرس‌هایی که کاربر باز کرده (URL)
- پارامترهای GET و POST
- محتوا و ساختار صفحات
- کوکی‌ها (Cookies)
- احتمالاً اطلاعات ورود (login credentials) اگر در فرم ارسال شده باشند

🔗 مثال خطرناک:

http

CopyEdit

POST /login HTTP/1.1

Host: example.com

Content-Type: application/x-www-form-urlencoded

username=admin&password=123456

در HTTP، تمام این اطلاعات بدون رمزنگاری از طریق شبکه عبور می‌کنن و در فایل pcap ثبت می‌شن. مهاجم می‌تونه اینو ببینه و اطلاعات ورود رو بدزده.

✅ راه‌حل:

- همیشه از HTTPS استفاده کن؛ مرورگرها هم امروزه اگر صفحه‌ای HTTP باشه، هشدار می‌دن.

- اگر در فایل pcap شما ترافیک HTTP هست، مخصوصاً POST یا GET با داده‌های حساس، حتماً می‌تونه آسیب‌پذیر باشه.

اگر خواستی با هم به تحلیل دقیق از ترافیک HTTP فایل‌ات انجام بدیم، کافیه برام آپلودش کنی.

You said:

reduced_sample.pcap

File

الان مثلاً با این چیکار میشه کرد

ChatGPT said:

ماژول مورد نیاز برای تحلیل مستقیم فایل pcap (pyshark) در محیط فعلی در دسترس نیست، اما هنوز هم می‌تونم بهت کمک کنم محتوای این فایل رو بررسی کنی.

دو راه داریم:

☑ **گزینه 1: خروجی Wireshark رو با من به اشتراک بذار**

اگر توی Wireshark فایل رو باز کردی:

1. چند فریم مشکوک (مثلاً http, ftp, telnet) رو پیدا کن.

2. از اون‌ها Export یا Screenshot بگیر (یا با File > Export Packet Dissections > As Plain Text ذخیره کن).

3. این اطلاعاتو اینجا آپلود یا Paste کن.

☑ **گزینه 2: استخراج متن ساده از فایل pcap**

من می‌تونم فایل رو به صورت باینری باز کنم و دنبال کلیدواژه‌هایی مثل USER, PASS, GET, POST, login, cookie بگردم. می‌خوای این روشو امتحان کنیم؟ اگر تأیید کنی، می‌رم سراغ تحلیل مستقیم محتوای متنی فایل pcap برای پیدا کردن داده‌های حساس.

هیچ‌کدام از کلیدواژه‌های مشکوک مثل USER, PASS, GET, POST, login یا cookie در فایل pcap شما پیدا نشدند.

📖 **یعنی چی؟**

- فایل شما احتمالاً شامل پروتکل‌های رمزنگاری شده (مثل HTTPS یا SSH) بوده یا اصلاً داده‌ای مثل لاگین یا فرم وب در اون نیست.
- یا ممکنه فقط شامل ترافیک عادی (مثل DNS، ICMP، NTP، یا ترافیک سطح پایین شبکه) باشه که اطلاعات حساسی منتقل نمی‌کنه.

☑ مرحله بعد چی می‌تونه باشه؟

اگه بخوای بدونی چه پروتکل‌هایی توی فایل استفاده شدن، می‌تونم یه تحلیل سطح بالا از لایه‌های مختلف ترافیک انجام بدم و ببینم کلاً چی داخلشه) مثلاً فقط DNS هست؟ یا SSH؟ یا چیزی دیگه؟).

می‌خوای یه نمای کلی از پروتکل‌های موجود در فایل pcap بهت نشون بدم؟

You said:

http, ftp, telnet اینا یافت نشدن حالا بقیه رو چجوری سرچ کنم

ChatGPT said:

در جستجوی گسترده‌تر هم هیچ کلیدواژه‌ای مرتبط با پروتکل‌های کاربردی مانند SSH, TLS, DNS, Authorization, User-Agent, Host و سایر درخواست‌های HTTP یا احراز هویت پیدا نشد.

📖 تفسیر این وضعیت:

این احتمال‌ها وجود داره:

1. فایل **pcap** شامل ترافیک سطح پایین (مثل ARP ، ICMP ، یا DHCP) بوده که در اون خبری از پروتکل‌های کاربردی قابل سوءاستفاده نیست.
2. ممکنه تمام ترافیک رمزنگاری شده بوده (مثل HTTPS یا SSH) ، که باعث می‌شه نتونیم اطلاعات مفیدی به صورت متنی استخراج کنیم.
3. یا فایل فقط شامل بخشی از تبادل شبکه بوده (مثلاً بعد از شروع ارتباط، که دیگه اطلاعات حساس رد و بدل نشده).

☑ گام بعدی پیشنهادی:

می‌تونم یک تحلیل کلی از محتوای فایل pcap ارائه بدم، شامل اینکه:

- چه پروتکل‌هایی در اون وجود دارن؟
- چند تا فریم داره؟
- چه پورتهایی استفاده شده؟
- آیا ارتباط‌های مشکوک یا تکراری دیده می‌شن؟

می‌خوای این بررسی رو انجام بدم؟

سؤال کاملاً بهجاست و لحن تندت رو می‌ذارم پای هیجان یادگیری 😊 — بیا خیلی شفاف بگم چطور توی Wireshark دنبال چیزای مهمی مثل SSH, TLS, DNS, User-Agent, Authorization و بقیه بگردی.

🔗 جستجوی دقیق در Wireshark

🔍 برای پیدا کردن انواع پروتکل‌ها:

فیلتر در Wireshark پروتکل

DNS	dns
HTTP	http
TLS/SSL	ssl یا tls
SSH	ssh
FTP	ftp
Telnet	telnet

🔍 برای پیدا کردن چیزای خاص توی HTTP:

🔍 دیدن Host, User-Agent, Authorization, Cookie:

☑ از فیلتر زیر استفاده کن:

wireshark

CopyEdit

http.request

سپس، توی پنجره پایین (Packet Details) برو روی لایه‌ی Hypertext Transfer Protocol و ببین:

Host: ... •

User-Agent: ... •

Authorization: Basic ... •

Cookie: ... •

🔍 برای دیدن TLS و شروع‌های SSL:

فیلتر:

wireshark

CopyEdit

tls.handshake

توی لایه TLS دنبال:

- Client Hello
- Server Hello
- Certificate

🔗 برای دیدن ارتباطهای SSH:

فیلتر:

wireshark

CopyEdit

ssh

اگر دیدی:

• Protocol version exchange

• Key exchange init

یعنی ارتباط SSH شناسایی شده.

⚠️ نکته مهم:

برای جستجوی درون متن بسته‌ها (مثلاً دنبال login, password, admin) می‌تونی از:

wireshark

CopyEdit

frame contains "login"

استفاده کنی. همین‌طور برای بقیه:


wireshark

CopyEdit

frame contains "password"

frame contains "Authorization"

frame contains "User-Agent"



اگه بخوای، می‌تونم به لیست کامل از فیلترهای مفید Wireshark برات تهیه کنم یا بگم برای چه سناریوهایی از کدومشون استفاده کنی. بگو فقط دنبال چی هستی.

You said:



من به همچنین چیزی دست پیدا کردم ولی ایده ای ندارم چیه یا اینکه اصلاً مشکل داره یا نه. ولی اینارو تهش نوشته: expert info: a new tcp session is started with the same ports as an earlier session in this trace. expert info: this frame is a suspected retransmission.

ChatGPT said:

خیلی خوبه که داری دقیق نگاه می‌کنی و همین که رسیدی به بخش 0 tcp.stream یعنی مسیری درست. حالا بیایم با هم تحلیل کنیم چی داریم اینجا:

چی داریم اینجا؟ 📖

☑ خلاصه‌ی وضعیت:

- ارتباط TCP از آی‌پی 128.3.23.5 به 196.119.124.204
- پورت مقصد → 80: یعنی احتمالاً HTTP
- در Packet اول به SYN دیده میشه → شروع به اتصال TCP
- اما بقیه پکت‌ها تکراری هستن ([TCP Retransmission]) و ارتباط کامل نمی‌شه

⚠ هشدار Wireshark چیه؟

• “A new TCP session is started with the same ports as an earlier session”

یعنی از همون پورت‌ها قبلاً هم استفاده شده بوده؛ ممکنه تداخل داشته باشه یا پورت‌ها Reused شدن.

• “This frame is a suspected retransmission”

یعنی به پکت TCP دوباره ارسال شده چون پاسخی دریافت نشده (به خاطر تاخیر، اختلال یا از بین رفتن پکت اصلی).

🔧 آسیب‌پذیری داره؟

نه مستقیماً. اطلاعاتی که توی این تصویر هست:

- فقط به تلاش برای برقراری ارتباط TCP ه
- هیچ دیتایی منتقل نشده (فقط SYN)
- هنوز مرحله ACK یا تبادل HTTP انجام نشده

🔒 بنابراین نه credential ی رد و بدل شده، نه درخواست خاصی مثل GET/POST

🧐 پس چرا ممکنه مهم باشه؟

1. سرنخ حمله یا اسکن باشه:

- مثلاً ابزارهای اسکن (مثل Nmap معمولاً پورت 80 رو چک می‌کنن ببینن وب‌سرور هست یا نه.
- آگه صدها مورد مشابه ببینی، می‌تونه نشون‌دهنده یک پرت اسکن باشه (پیش‌درآمد حمله).

2. Reused Port در شبکه‌های واقعی گاهی مشکلاتی درست می‌کنه:

- ابزارهای امنیتی ممکنه گمراه بشن
- جلسه‌ها overlap کنن
- تحلیل دستی سخت‌تر بشه

✓ پیشنهاد:

برای اینکه بفهمی واقعاً تبادل مهمی توی این فایل بوده یا نه:

1. توی Wireshark برو و فیلتر کن:

wireshark

CopyEdit

tcp.stream eq 1

tcp.stream eq 2

...

2. ببین کدوم استریم واقعاً داده رد و بدل کرده (نه فقط SYN)

3. دنبال مواردی باش مثل:

○ GET /index.html

○ POST /login

○ USER, PASS, Authorization, Cookie

آگه خواستی کمک کنم بهت بگم دقیقاً کدوم استریم‌ها تو فایل pcap دیتای قابل بررسی دارن، فقط بگو. می‌تونم کل فایل رو اسکن کنم و لیست استریم‌های مفید رو دربیارم برات.

You said:



ChatGPT said:

عکس جدیدت خیلی جالبه، چون نشون میده یه تلاش برای اتصال به پورت 515 روی یه سرور انجام شده — و این خودش جای تحلیل داره پس !

🔗 چی داریم اینجا؟

مقدار	مورد
128.3.23.117	Source IP
128.55.150.95	Destination IP
515	پورت مقصد
فقط SYN فرستاده شده → جواب RST, ACK گرفته وضعیت اتصال	
tcp.stream eq 2	فیلتر فعال

! پورت 515 چیه؟

پورت 515 برای سرویس **Line Printer Daemon (LPD)** استفاده می‌شه.

🔗 این سرویس قدیمی برای چاپ شبکه‌ای در سیستم‌های UNIX بوده و در زمان خودش آسیب‌پذیری‌های زیادی داشته:

آسیب‌پذیری‌های شناخته‌شده پورت 515:

1. **عدم احراز هویت قوی** → به کاربر اجازه می‌داد فایل چاپ بفرسته بدون اینکه هویت تأیید بشه.
2. **Directory traversal** → امکان ارسال مسیرهای ساختگی برای نوشتن فایل در سرور.
3. **Buffer Overflow** → باعث اجرای کد دلخواه در سیستم مقصد می‌شد.
4. **Access bypass** → ارسال فایل‌های جعلی برای اجرا یا چاپ روی سیستم.

🔍 تحلیل حمله ممکن:

👤 مرحله 1: اسکن پورت‌ها

مهاجم پورت‌های معروف رو با ابزارهایی مثل Nmap یا hping3 چک می‌کنه:

```
bash
```

CopyEdit

```
nmap -p 515 128.55.150.95
```

🔧 مرحله 2: اگه پورت باز بود...

مهاجم می‌تونه:

- با ابزارهایی مثل `lpr` یا `rlpr` سعی کنه فایل به چاپگر ارسال کنه
- فایل‌هایی با پسوند خاص (مثلاً `.ps`, `.sh`) ارسال کنه
- مسیرهایی مثل `../etc/passwd` برای تست دسترسی استفاده کنه

🚫 در این مورد خاص:

اما چون سرور جواب داده با:

`csharp`

`CopyEdit`

`[RST, ACK]`

یعنی: ✕ اتصال رد شده → پورت باز نیست یا فایروال جلوی اتصال رو گرفته

✅ نتیجه‌گیری:

- اطلاعاتی که داری نشون‌دهنده تلاش برای اسکن یا حمله اولیه به پورت 515 هست.
- ولی حمله موفق نیست چون سرور پاسخ `Reset` داده.

اگه خواستی می‌تونم یه جدول از کل پورت‌هایی که تو این فایل `pcap` استفاده شدن و مقصدشون چی بوده در بیارم برات (با کد)، تا ببینیم کجاها ممکنه در معرض خطر بوده باشن. بگی کافیه.

عالیه که داری دقیق جلو میری! توی `Wireshark` خیلی راحت می‌تونی دنبال این کلیدواژه‌ها بگردی و ببینی آیا داده‌ی مهم یا حساس مثل `GET, POST, Authorization, Cookie` و غیره توی ترافیک پیدا می‌شه یا نه.

🔍 روش‌های جستجوی داده‌های حساس در `Wireshark`

1. 📌 استفاده از فیلترهای ساده نمایش (`Display Filter`)

برای درخواست‌های: `HTTP`

`wireshark`

`CopyEdit`

`http.request`

این فیلتر همه‌ی GET و POST و سایر درخواست‌های HTTP رو نشون می‌ده. بعدش توی پنجره پایین (Packet Details) برو روی لایه Hypertext Transfer Protocol تا ببینی دقیقاً چه چیزی رد و بدل شده.

2. ☒ جستجوی رشته‌های خاص در محتوای بسته‌ها

برای دیدن کلیدواژه‌های حساس مثل:

کلیدواژه	فیلتر
GET	frame contains "GET"
POST	frame contains "POST"
Authorization	frame contains "Authorization"
User-Agent	frame contains "User-Agent"
Cookie	frame contains "Cookie"
FTPدرUSER (frame contains "USER"
FTPدرPASS (frame contains "PASS"

3. ☒ بررسی فیلتر ترکیبی برای درخواست ورود

مثلاً اگه بخوای ببینی آیا کسی با POST وارد شده و توی هدرش Authorization داشته یا نه:

wireshark

CopyEdit

`http.request.method == "POST" and frame contains "Authorization"`

4. ☒ بررسی همه‌ی Stream ها برای دیتا

میتونی از Follow TCP Stream استفاده کنی:

1. روی یک پکت راست‌کلیک کن

2. گزینه Follow → TCP Stream

3. اونجا دیتای کامل جلسه رو به صورت متنی می‌بینی (مثل یه گفت‌وگوی کامل کلاینت-سرور)

🔍 اگر چیزی دیدی که شامل username, password یا token بود

اون اطلاعات رو استخراج کن و بررسی کن:

- رمزگذاری نشده؟
- از Basic Auth استفاده شده؟
- کل کوکی‌ها با یه attacker share می‌شن؟
- اینا جاهایی‌ان که یه مهاجم می‌تونه استفاده کنه برای:
 - Hijack کردن session
 - Login بدون رمز
 - Replay کردن درخواست‌ها

اگه بخوای من می‌تونم همین فایل pcap رو بررسی کنم و لیست کامل Stream هایی که شامل این موارد هستن رو بهت بدم. فقط بگو ادامه بدم؟

Tools

فایل PCAP چیه؟

PCAP مخفف "Packet CAPture" است و به یک فرمت فایل استاندارد اشاره دارد که برای ذخیره بسته‌های داده‌ای که در یک شبکه کامپیوتری رهگیری شده‌اند، استفاده می‌شود. به زبان ساده، یک فایل PCAP حاوی یک رکورد دقیق از ترافیک شبکه در یک بازه زمانی مشخص است.

در اینجا برخی از نکات کلیدی در مورد فایل‌های PCAP آورده شده است:

- **محتوا:** یک فایل PCAP شامل دنباله‌ای از بسته‌های شبکه است. برای هر بسته، فایل معمولاً شامل موارد زیر است:
 - یک مهر زمانی (timestamp) که نشان می‌دهد بسته در چه زمانی ضبط شده است.
 - طول بسته.
 - داده‌های واقعی بسته (هدرها و بار داده).
- **کاربردها:** فایل‌های PCAP برای اهداف مختلفی استفاده می‌شوند، از جمله:
 - **عیب‌یابی شبکه:** برای تشخیص مشکلات شبکه، مانند از دست رفتن بسته، تأخیر یا مشکلات اتصال.
 - **تحلیل امنیتی:** برای بررسی ترافیک مخرب، شناسایی حملات و تجزیه و تحلیل فعالیت‌های مشکوک.
 - **تحلیل پروتکل:** برای درک نحوه عملکرد پروتکل‌های شبکه.
 - **توسعه نرم‌افزار:** برای آزمایش و اشکال‌زدایی برنامه‌های کاربردی شبکه.

- **نحوه ایجاد:** فایل‌های PCAP معمولاً توسط ابزارهای "packet sniffer" مانند Wireshark، tcpdump یا tshark ایجاد می‌شوند. این ابزارها ترافیک شبکه را از یک رابط شبکه ضبط کرده و آن را در یک فایل PCAP ذخیره می‌کنند.
- **نحوه باز کردن و تحلیل:** برای باز کردن و تجزیه و تحلیل فایل‌های PCAP، معمولاً از نرم‌افزارهای تحلیلگر شبکه مانند Wireshark استفاده می‌شود. این ابزارها به شما امکان می‌دهند بسته‌ها را مشاهده کنید، آنها را بر اساس معیارهای مختلف فیلتر کنید و جریان ترافیک شبکه را تجزیه و تحلیل کنید.
- **فرمت‌های مرتبط:** فرمت PCAP اصلی برای سال‌ها استاندارد بوده است. با این حال، فرمت جدیدتری به نام **PCAP-NG** (PCAP Next Generation) نیز وجود دارد که قابلیت‌های بیشتری مانند ذخیره فراداده‌های بیشتر و پشتیبانی از ضبط از چندین رابط را ارائه می‌دهد. با این حال، اصطلاح "فایل PCAP" اغلب به طور کلی برای اشاره به فایل‌های ضبط ترافیک شبکه استفاده می‌شود، بدون توجه به فرمت دقیق.

امیدوارم این توضیح برای شما مفید باشد. اگر سوالات بیشتری دارید، بپرسید.

سوال تشریحی (امتیازی: 5 نمره)

1) فایل PCAP اصلی را به صورت دستی با استفاده از Wireshark تحلیل کرده و بیان کنید آیا اطلاعات دیگری در Trace مربوطه وجود دارد که شخص مهاجم بتواند به نحوی از آن استفاده کند؟ درباره راه‌های استفاده از آن توسط شخص مهاجم و آسیب‌پذیری‌های احتمالی توضیح دهید؛ همچنین سناریوی حمله با استفاده از این آسیب‌پذیری‌ها را در گزارش خود تحلیل کنید. (توجه داشته باشید این سوال می‌تواند جواب‌های بسیار متفاوتی داشته باشد و نکته حائز اهمیت درک و تحلیل شما از Trace یک سناریوی تبادل اطلاعات در شبکه است)

ینی باید وایرشارک نصب کنم؟ بعد آگه رو ویندوز باشه وایرشارکم اوکی عه؟

بله، برای پاسخ دادن به سوال تشریحی اول به طور کامل، شما نیاز دارید که یک فایل PCAP را با استفاده از Wireshark تحلیل کنید.

در مورد نصب Wireshark، بله، شما باید آن را نصب کنید. Wireshark روی سیستم عامل‌های مختلفی از جمله ویندوز، macOS و لینوکس قابل نصب است. اگر Wireshark را روی ویندوز نصب کنید، برای تحلیل فایل PCAP مشکلی نخواهید داشت.

برای اینکه بتوانید به این سوال پاسخ دهید، شما به یک فایل PCAP نیاز دارید. آیا شما فایل PCAP خاصی مد نظر دارید که می‌خواهید آن را تحلیل کنید؟ یا مایلید از یک فایل نمونه استفاده کنید؟

لطفاً توجه داشته باشید که پاسخ دادن به این سوال تشریحی نیازمند تحلیل دقیق یک فایل PCAP و درک عمیق از مفاهیم شبکه و امنیت است. من می‌توانم در مراحل تحلیل به شما کمک کنم، اما برای ارائه یک پاسخ کامل و امتیازآور، شما خودتان باید تحلیل را انجام دهید و یافته‌های خود را شرح دهید.

برای trace اول باید دنبال پروتکل‌هایی بگردم که رمزگذاری نشدن درسته؟ چیا رو دنبالشون تو wireshark بگردم

بله، برای شروع تحلیل یک فایل PCAP به دنبال اطلاعات حساس که ممکن است به صورت رمزگذاری نشده منتقل شده باشند، جستجو برای پروتکل‌های غیر رمزگذاری شده یک نقطه شروع خوب است. این پروتکل‌ها بیشتر در معرض استراق سمع قرار دارند. برخی از پروتکل‌های رایجی که معمولاً رمزگذاری نمی‌شوند (یا ممکن است نسخه‌های رمزگذاری نشده آن‌ها وجود داشته باشد) و شما می‌توانید در Wireshark به دنبال آن‌ها بگردید عبارتند از:

- **HTTP:** ترافیک وب که به صورت متن آشکار منتقل می‌شود. شما می‌توانید محتوای صفحات وب، درخواست‌ها و پاسخ‌ها، و احتمالاً کوکی‌ها و سایر اطلاعات را مشاهده کنید. در Wireshark می‌توانید با فیلتر کردن بر اساس http این ترافیک را مشاهده کنید.
- **FTP:** برای انتقال فایل استفاده می‌شود و معمولاً اطلاعات کاربری (نام کاربری و رمز عبور) و محتوای فایل‌ها به صورت متن آشکار منتقل می‌شوند. فیلتر Wireshark برای این پروتکل ftp است.
- **Telnet:** یک پروتکل قدیمی برای دسترسی از راه دور به خط فرمان است و تمام ارتباطات آن به صورت متن آشکار منتقل می‌شود. فیلتر telnet Wireshark است.
- **SMTP (بدون TLS/SSL):** برای ارسال ایمیل استفاده می‌شود و محتوای ایمیل و احتمالاً اطلاعات کاربری ممکن است به صورت غیر رمزگذاری شده منتقل شوند. فیلتر smtp Wireshark است.
- **DNS:** برای ترجمه نام دامنه به آدرس IP استفاده می‌شود. اگرچه خود درخواست‌های DNS معمولاً حساس نیستند، اما می‌توان از آن‌ها برای جمع‌آوری اطلاعات استفاده کرد. فیلتر dns Wireshark است.

علاوه بر این پروتکل‌ها، ممکن است پروتکل‌های سفارشی یا پروتکل‌های دیگری نیز وجود داشته باشند که رمزگذاری نشده‌اند. در Wireshark، شما می‌توانید از قسمت "Filter" در بالای صفحه برای جستجو بر اساس این پروتکل‌ها استفاده کنید. برای مثال، برای دیدن فقط ترافیک HTTP، عبارت http را در قسمت فیلتر وارد کرده و Enter بزنید.

پس از فیلتر کردن، بسته‌های مربوط به آن پروتکل را بررسی کنید و ببینید آیا اطلاعات حساسی در قسمت‌های مختلف بسته (مانند Data یا Info) قابل مشاهده است یا خیر.

آیا شما فایل PCAP خاصی دارید که می‌خواهید با آن کار کنید؟ اگر دارید، می‌توانید برخی از این فیلترها را روی آن اعمال کنید و ببینید چه نتایجی به دست می‌آورد.

خب بعد از این دیگه مهاجم چه حمله ای رو امتحان کنه یا توی پکتا دنبال چی بگرده؟

بعد از اینکه مهاجم پروتکل‌های غیر رمزگذاری شده را شناسایی کرد و احتمالاً اطلاعات حساسی را در آن‌ها پیدا کرد، می‌تواند بسته به نوع اطلاعات به دست آمده و پروتکل مورد نظر، حملات مختلفی را امتحان کند یا به دنبال اطلاعات خاصی در بسته‌ها بگردد.

در اینجا چند سناریو و مواردی که مهاجم ممکن است به دنبال آن باشد آورده شده است:

اگر اطلاعات کاربری (نام کاربری و رمز عبور) به صورت متن آشکار شود:

- **تلاش برای استفاده مجدد از اعتبارنامه (Credential Stuffing):** مهاجم می‌تواند از این نام کاربری و رمز عبور برای تلاش برای ورود به سایر سرویس‌ها و حساب‌های کاربری قربانی استفاده کند، زیرا بسیاری از افراد از رمز عبور یکسان برای چندین حساب استفاده می‌کنند.
- **دسترسی غیرمجاز:** اگر اعتبارنامه مربوط به یک سرویس مهم باشد (مانند FTP برای آپلود فایل‌های وبسایت یا Telnet برای دسترسی به سرور)، مهاجم می‌تواند از آن برای دسترسی غیرمجاز و انجام اقدامات مخرب استفاده کند.

اگر اطلاعات مربوط به نشست (Session Cookies) در ترافیک HTTP غیر رمزگذاری شده پیدا شود:

- **ربودن نشست (Session Hijacking):** مهاجم می‌تواند از Session Cookie دزدیده شده برای جعل هویت کاربر و دسترسی به حساب کاربری او در وبسایت استفاده کند، بدون اینکه نیاز به دانستن رمز عبور داشته باشد.

اگر اطلاعات حساس دیگری در ترافیک غیر رمزگذاری شده یافت شود:

- بسته به نوع اطلاعات (مثلاً اطلاعات مربوط به ساختار داخلی یک سیستم، کلیدهای API غیر حساس، و غیره)، مهاجم می‌تواند از آن‌ها برای برنامه‌ریزی حملات هدفمندتر، یافتن آسیب‌پذیری‌های دیگر یا جمع‌آوری اطلاعات بیشتر استفاده کند.

در ادامه، مهاجم ممکن است در بسته‌ها به دنبال موارد زیر بگردد:

- **آسیب‌پذیری‌های شناخته شده:** مهاجم ممکن است به دنبال الگوهای ترافیکی بگردد که نشان دهنده استفاده از نسخه‌های آسیب‌پذیر نرم‌افزار یا پروتکل‌ها باشد. برای مثال، الگوهای خاص در هدرهای HTTP یا SMTP.
- **اطلاعات مربوط به ساختار شبکه:** با بررسی بسته‌ها، مهاجم می‌تواند اطلاعاتی در مورد آدرس‌های IP داخلی، ساختار شبکه، و سرویس‌های در حال اجرا به دست آورد که می‌تواند در مراحل بعدی حمله مورد استفاده قرار گیرد.
- **نشانه‌هایی از حملات قبلی یا در حال انجام:** مهاجم ممکن است به دنبال الگوهای ترافیکی بگردد که نشان دهنده فعالیت‌های مخرب مانند تلاش برای تزریق SQL، حملات XSS یا بات‌نت‌ها باشد.

سناریوی حمله فرضی:

فرض کنید مهاجم یک فایل PCAP را تحلیل می‌کند و متوجه می‌شود که ترافیک HTTP به یک وبسایت بدون استفاده از HTTPS (رمزگذاری) در حال انتقال است. در این ترافیک، یک Session Cookie مربوط به ورود کاربر به یک پنل مدیریت دیده می‌شود.

1. **استراق سمع:** مهاجم با استفاده از فایل PCAP، Session Cookie کاربر را استخراج می‌کند.
2. **ربودن نشست:** مهاجم از این Session Cookie در مرورگر خود استفاده می‌کند و بدون نیاز به وارد کردن نام کاربری و رمز عبور، به پنل مدیریت وبسایت دسترسی پیدا می‌کند.
3. **اقدامات مخرب:** پس از دسترسی به پنل مدیریت، مهاجم می‌تواند اقدامات مخربی مانند تغییر محتوای وبسایت، نصب بدافزار یا دسترسی به اطلاعات حساس را انجام دهد.

برای تحلیل دقیق‌تر، شما نیاز دارید یک فایل PCAP واقعی را بررسی کنید. آیا شما فایل PCAP دارید که می‌خواهید آن را تحلیل کنید تا بتوانیم سناریوهای خاص‌تری را بررسی کنیم؟

داخل پکت‌ها به دنبال پروتکل‌هایی مثل http, ftp, telnet, smtp, dns، البته در sample.pcap چیزی پیدا نشد، پس تا اینجا چیزی یافت نشد.

متوجه شدم. اگر در فایل sample.pcap با جستجو برای پروتکل‌های http, ftp, telnet, smtp و dns چیزی پیدا نکردید، این بدان معناست که در این trace خاص، ترافیک مربوط به این پروتکل‌ها وجود ندارد (یا حداقل در بازه زمانی ضبط شده، فعالیتی از این پروتکل‌ها ثبت نشده است).

این بدان معنا نیست که هیچ اطلاعات قابل استفاده‌ای در فایل وجود ندارد. در این مرحله، می‌توانید رویکردهای دیگری را برای تحلیل امتحان کنید:

1. **بررسی سایر پروتکل‌ها:** چه پروتکل‌های دیگری در این trace وجود دارند؟ می‌توانید در قسمت "Protocol" در Wireshark نگاهی بیندازید تا ببینید چه پروتکل‌هایی با بیشترین تعداد بسته وجود دارند. شاید اطلاعات جالبی در پروتکل‌های دیگر پنهان باشد.
 2. **جستجو برای الگوهای خاص:** آیا انتظار ترافیک خاصی را دارید؟ مثلاً، آیا می‌دانید که ارتباط با یک آدرس IP خاص برقرار شده است؟ می‌توانید با استفاده از فیلترهایی مانند `ip.addr == [آدرس IP]` به دنبال آن بگردید.
 3. **تحلیل جریان‌ها (Flows):** در Wireshark، می‌توانید جریان‌های TCP یا UDP را دنبال کنید (Statistics > Flow Graph یا انتخاب یک بسته TCP/UDP و سپس Follow > TCP Stream یا UDP Stream). این کار می‌تواند به شما کمک کند تا مکالمات کامل بین دو نقطه را ببینید، که ممکن است حاوی اطلاعات مفیدی باشد حتی اگر از پروتکل‌های ذکر شده استفاده نکنند.
 4. **جستجو برای خطاها یا رفتارهای غیرعادی:** آیا بسته‌هایی با پرچم‌های غیرمعمول TCP (مانند SYN بدون ACK‌های متناظر، یا تعداد زیادی بسته‌های Retransmission) وجود دارد؟ این‌ها ممکن است نشان دهنده مشکلات شبکه یا تلاش‌های مخرب باشند.
- برای ادامه تحلیل sample.pcap، چه رویکردی را ترجیح می‌دهید امتحان کنید؟ آیا می‌خواهید به دنبال پروتکل‌های دیگر بگردیم، یا یک آدرس IP خاص را بررسی کنیم، یا جریان‌ها را دنبال کنیم؟ به من بگویید تا راهنمایی‌تان کنم.
- چیزی مثل TCP Stream چجوری باعث همیشه ارتباط بین 2 نقطه رو دید؟
- وقتی شما یک TCP Stream را در Wireshark دنبال می‌کنید، در واقع تمام بسته‌های TCP که متعلق به یک ارتباط (connection) خاص بین دو آدرس IP و دو پورت هستند را به صورت یک مکالمه واحد و مرتب شده مشاهده می‌کنید.
- برای درک بهتر، بیایید نحوه کارکرد TCP را مرور کنیم:
1. **برقراری اتصال (Three-way handshake):** یک اتصال TCP با تبادل سه بسته برقرار می‌شود: SYN، SYN-ACK، و ACK. این سه بسته، مبدأ و مقصد ارتباط و همچنین شماره‌های توالی اولیه را تعیین می‌کنند.
 2. **انتقال داده:** پس از برقراری اتصال، داده‌ها در قالب بسته‌های TCP منتقل می‌شوند. هر بسته شامل بخشی از داده‌ها و همچنین شماره توالی (Sequence Number) است که ترتیب بسته‌ها را مشخص می‌کند. گیرنده با ارسال بسته‌های ACK (Acknowledgement) دریافت موفقیت‌آمیز داده‌ها را تأیید می‌کند و شماره توالی بسته بعدی مورد انتظار را اعلام می‌کند.
 3. **پایان دادن به اتصال:** یک اتصال TCP با تبادل بسته‌های FIN به پایان می‌رسد.
- حالا تصور کنید که این بسته‌ها به صورت جداگانه در فایل PCAP ثبت شده‌اند. دنبال کردن یک TCP Stream در Wireshark این کارها را برای شما انجام می‌دهد:
- **شناسایی بسته‌های مرتبط:** Wireshark تمام بسته‌هایی را که دارای ترکیب یکسان از آدرس IP مبدأ و مقصد و پورت مبدأ و مقصد هستند، به عنوان بخشی از یک جریان در نظر می‌گیرد.
 - **مرتب‌سازی بسته‌ها:** بسته‌ها را بر اساس شماره توالی آن‌ها مرتب می‌کند تا ترتیب صحیح انتقال داده‌ها را نشان دهد.
 - **نمایش محتوای داده:** محتوای قسمت "Data" هر بسته را به ترتیب نمایش می‌دهد، گویی که یک جریان پیوسته از داده بین دو نقطه در حال انتقال است.

چرا این مفید است؟

- **دیدن کل مکالمه:** به جای دیدن بسته‌های جداگانه، شما می‌توانید کل تبادل اطلاعات بین دو کامپیوتر را به صورت یکجا ببینید. این امر درک آنچه در حال رخ دادن است را بسیار آسان‌تر می‌کند، به خصوص برای پروتکل‌های مبتنی بر متن که داده‌ها به صورت پیوسته ارسال می‌شوند (حتی اگر در چندین بسته تقسیم شده باشند).
- **تحلیل داده‌های لایه Application:** برای پروتکل‌هایی که بر روی TCP کار می‌کنند (مانند HTTP، SMTP، FTP، SSH و غیره)، دنبال کردن TCP Stream به شما اجازه می‌دهد تا محتوای واقعی لایه Application (مانند درخواست‌ها و پاسخ‌های HTTP، دستورات FTP، محتوای ایمیل و غیره) را به صورت سازمان‌یافته ببینید، بدون اینکه نگران تکه‌تکه شدن داده‌ها در بسته‌های مختلف باشید.

بنابراین، دنبال کردن TCP Stream به شما کمک می‌کند تا یک "تصویر بزرگتر" از ارتباط بین دو نقطه را ببینید و محتوای داده‌ای که بین آن‌ها رد و بدل می‌شود را به راحتی درک کنید، حتی اگر این داده‌ها در چندین بسته TCP منتقل شده باشند.

آیا می‌خواهید در sample.pcap یک TCP Stream را دنبال کنیم؟ اگر بله، لطفاً یک بسته TCP را در لیست بسته‌ها انتخاب کنید (در ستون "Protocol" باید "TCP" نوشته شده باشد) و سپس من مراحل بعدی را به شما خواهم گفت.

دیگه چیکارا میتونه بکنه؟

اگر پروتکل‌های غیر رمزگذاری شده اطلاعات مفیدی ارائه ندهند، مهاجم می‌تواند به دنبال موارد زیر در فایل PCAP بگردد:

1. اطلاعات مربوط به ساختار شبکه:

- آدرس‌های IP داخلی: شناسایی محدوده آدرس‌های IP استفاده شده در شبکه.
- آدرس‌های MAC: ارتباط بین آدرس‌های MAC و IP.
- سرویس‌های در حال اجرا: تشخیص پورت‌های باز و سرویس‌های مرتبط با آن‌ها (حتی اگر پروتکل رمزگذاری شده باشد، اطلاعات مربوط به برقراری ارتباط ممکن است قابل مشاهده باشد).
- توپولوژی شبکه (به صورت غیرمستقیم): از طریق الگوهای ترافیک و ارتباط بین دستگاه‌ها، ممکن است بتوانید حدس‌هایی در مورد نحوه سازماندهی شبکه بزنید.

2. نشانه‌های آسیب‌پذیری:

- نسخه‌های نرم‌افزار: گاهی اوقات، هدرهای پروتکل‌ها (مانند HTTP User-Agent یا SSH version) نسخه‌های نرم‌افزار را فاش می‌کنند که ممکن است دارای آسیب‌پذیری‌های شناخته شده باشند.
- خطاها یا رفتارهای غیرعادی: بسته‌هایی که نشان دهنده خطا هستند یا رفتارهای غیرمنتظره‌ای دارند، می‌توانند سرنخ‌هایی برای آسیب‌پذیری‌ها باشند. برای مثال، پاسخ‌های سرور با کدهای خطای خاص یا بسته‌های با طول غیرعادی.

3. نشانه‌های حملات:

- ترافیک مشکوک به یک آدرس IP یا پورت خاص: حجم بالای ترافیک، تلاش‌های مکرر برای اتصال به یک پورت، یا ترافیک به آدرس‌های IP شناخته شده مخرب.

- الگوهای ترافیکی مرتبط با حملات خاص: برای مثال، الگوهای SYN flood، اسکن پورت، یا تلاش‌های تزریق.

4. فراداده‌های ارتباطات:

- حتی اگر محتوای داده رمزگذاری شده باشد (مثلاً HTTPS)، فراداده‌های مربوط به ارتباط (مانند آدرس‌های IP مبدأ و مقصد، پورت‌ها، زمان و طول بسته‌ها) همچنان قابل مشاهده است. این اطلاعات می‌تواند برای تحلیل الگوهای ارتباطی و شناسایی نقاط مورد علاقه استفاده شود.

چه کاری می‌توانید در Wireshark انجام دهید:

- بررسی تب Endpoint ها (Endpoints <- Statistics): لیستی از تمام آدرس‌های MAC و IP و پورت‌های درگیر در trace را نشان می‌دهد. این می‌تواند به شناسایی نقاط مهم شبکه کمک کند.
 - بررسی مکالمات (Conversations <- Statistics): ترافیک گروه‌بندی شده بر اساس آدرس‌های IP و پورت‌ها را نشان می‌دهد. می‌توانید حجم ترافیک بین هر جفت را ببینید.
 - استفاده از فیلترهای پیشرفته: Wireshark فیلترهای قدرتمندی دارد که به شما امکان می‌دهد بر اساس معیارهای مختلف جستجو کنید (نه فقط پروتکل). برای مثال، می‌توانید بر اساس آدرس IP، پورت، طول بسته، یا محتوای هدرها فیلتر کنید.
- با توجه به اینکه در پروتکل‌های غیر رمزگذاری شده چیزی پیدا نشد، پیشنهاد می‌کنم تب‌های Endpoints و Conversations را در Wireshark بررسی کنید تا ببینید آیا الگوهای ارتباطی جالبی وجود دارد یا خیر. همچنین، می‌توانید سعی کنید بر اساس آدرس‌های IP یا پورت‌های خاص فیلتر کنید اگر فرضیه‌ای در مورد نقاط مورد نظر دارید.

آیا می‌خواهید یکی از این رویکردها را امتحان کنیم؟ مثلاً، بررسی Endpoint ها؟

میتونه دنبال پورت های مشکوک باشه؟ مثلاً 515؟

بله، قطعاً. جستجو برای ترافیک در پورت‌های مشکوک یا غیرمعمول می‌تواند یک روش مفید برای یافتن فعالیت‌های غیرعادی یا تلاش‌های مخرب باشد.

پورت 515 معمولاً با سرویس چاپگر خطی (Line Printer Daemon - LPD) مرتبط است. اگر ترافیک زیادی را روی این پورت ببینید، یا اگر ارتباطاتی از منابع غیرمنتظره به این پورت برقرار شده باشد، ممکن است ارزش بررسی بیشتر داشته باشد.

برای فیلتر کردن ترافیک بر اساس پورت در Wireshark، می‌توانید از فیلترهای زیر استفاده کنید:

- برای ترافیک با پورت مبدأ 515: `tcp.srcport == 515` یا `udp.srcport == 515`
- برای ترافیک با پورت مقصد 515: `tcp.dstport == 515` یا `udp.dstport == 515`
- برای ترافیک با پورت مبدأ یا مقصد 515: `tcp.port == 515` یا `udp.port == 515`
- برای ترافیک TCP یا UDP با پورت 515: `port == 515`

شما می‌توانید این فیلترها را در قسمت "Filter" در Wireshark وارد کرده و Enter بزنید تا فقط بسته‌های مربوط به پورت 515 نمایش داده شوند.

آیا می‌خواهید فیلتر `port == 515` را روی `sample.pcap` اعمال کنیم و ببینیم آیا ترافیک روی این پورت وجود دارد؟

