

Wire Shark: MMIT (Man In The Middle Attack)

Members:

- Ahmed Mahmoud Hassan - 2205155
- Mariam Mostafa Amin - 2205084
- Nada Mohamed Abdelsattar - 2205173
- Hannah Emad Eldin - 2205123
- Ali Mohamed Oqab - 2205077

1- HTTP (UNSECURE)

After we login while the wireshark sniffs our network we can see that it's vulnerable and all of the login credentials has been leaked with ease

http.request.method=="POST"						
No.	Time	Source	Destination	Protocol	Length	Info
426	12.070618	192.168.1.110	192.168.1.110	HTTP	8235	POST /upload HTTP/1.1 (
2908	375.595185	192.168.1.110	192.168.1.110	HTTP	789	POST /login HTTP/1.1 (a
3062	387.232567	192.168.1.110	192.168.1.110	HTTP	787	POST /2fa HTTP/1.1 (app
3840	396.832999	192.168.1.110	192.168.1.110	HTTP	8362	POST /profile HTTP/1.1
11451	1678.718919	:::1	:::1	HTTP	1074	POST /phpmyadmin/index.p
12651	1885.171741	192.168.1.110	192.168.1.110	HTTP	842	POST /login HTTP/1.1 (a
12791	1899.115986	192.168.1.110	192.168.1.110	HTTP	787	POST /2fa HTTP/1.1 (app

Content-Length: 62\r\n
Cache-Control: max-age=0\r\n
Origin: http://192.168.1.110:8000\r\n
Content-Type: application/x-www-form-urlencoded\r\n
Upgrade-Insecure-Requests: 1\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/104.0.0.0 Safari/537.36\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8\r\n
Referer: http://192.168.1.110:8000/login\r\n
Accept-Encoding: gzip, deflate\r\n
Accept-Language: en-US,en;q=0.9,ja;q=0.8\r\n
Cookie: ajs_anonymous_id=59eae0ea-5e0b-4e26-85d6-361a05097735; session=0F0u04sCrbpfgZzLih4EX6tvUq5\r\n
[Response in frame: 12690]
[Full request URI: http://192.168.1.110:8000/login]
File Data: 62 bytes
HTML Form URL Encoded: application/x-www-form-urlencoded
Form item: "username_or_email" = "hannahemad@gmail.com"
Key: username_or_email
Value: hannahemad@gmail.com
Form item: "password" = "Hannah123\$"
Key: password
Value: Hannah123\$

We concluded:

Email: hannahemad@gmail.com

Password: Hannah123\$

And it also shows the session key which is also crucial for the attacker to try **sql injection attack**

Problem with HTTP:

- The communication between client and server is unencrypted and vulnerable to the eyes
- Any attacker on the same network can intercept and read the data
- This proves that HTTP is not ideal for handling authentication

Two-Factor Authentication:

Here it shows that even though that the attacker knows the 6 digits the user used to login with he can't really login with it since it regenerates every 30 seconds and it will prevent him from logging into the account. But since we have a 2FA doesn't mean we can overlook the fact that the attacker can already know our Email and Password.

```
12651 1885.171741 192.168.1.110 192.168.1.110 HTTP 842 POST /login HTTP/1.1
12791 1899.115986 192.168.1.110 192.168.1.110 HTTP 787 POST /2fa HTTP/1.1

Content-Length: 11\r\n
  [Content length: 11]
Cache-Control: max-age=0\r\n
Origin: http://192.168.1.110:8000\r\n
Content-Type: application/x-www-form-urlencoded\r\n
Upgrade-Insecure-Requests: 1\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/
Referer: http://192.168.1.110:8000/2fa\r\n
Accept-Encoding: gzip, deflate\r\n
Accept-Language: en-US,en;q=0.9,ja;q=0.8\r\n
Cookie: ajs_anonymous_id=59eae0ea-5e0b-4e26-85d6-361a05097735; session=0F0u04sCrbpfgZzLih4
  Cookie pair: ajs_anonymous_id=59eae0ea-5e0b-4e26-85d6-361a05097735
  Cookie pair: session=0F0u04sCrbpfgZzLih4EX6tvUq50t351MaLN43v6iE
\r\n
[Response in frame: 12825]
[Full request URI: http://192.168.1.110:8000/2fa]
File Data: 11 bytes
HTML Form URL Encoded: application/x-www-form-urlencoded
Form item: "code" = "776778"
  Key: code
  Value: 776778
```

2- HTTPS (Secure)

This screenshot shows the same login steps but send over HTTPS instead of HTTP, which uses the TCP secure protocol

tcp.port == 8000

No.	Time	Source	Destination	Protocol	Length	Info
15998	2712.628021	192.168.1.110	192.168.1.110	TLSv1.3	8258	Application Data
15999	2712.628038	192.168.1.110	192.168.1.110	TCP	44	63564 → 8000 [ACK] Seq=2535 Ack=109220
16000	2712.628074	192.168.1.110	192.168.1.110	TLSv1.3	8258	Application Data
16001	2712.628099	192.168.1.110	192.168.1.110	TCP	44	63564 → 8000 [ACK] Seq=2535 Ack=117434
16002	2712.628134	192.168.1.110	192.168.1.110	TLSv1.3	8258	Application Data
16003	2712.628153	192.168.1.110	192.168.1.110	TCP	44	63564 → 8000 [ACK] Seq=2535 Ack=125648
16004	2712.628192	192.168.1.110	192.168.1.110	TLSv1.3	8258	Application Data
16005	2712.628217	192.168.1.110	192.168.1.110	TCP	44	63564 → 8000 [ACK] Seq=2535 Ack=133862
16006	2712.628269	192.168.1.110	192.168.1.110	TLSv1.3	5168	Application Data
16007	2712.628305	192.168.1.110	192.168.1.110	TCP	44	63564 → 8000 [ACK] Seq=2535 Ack=138986
16008	2712.628684	192.168.1.110	192.168.1.110	TCP	44	[TCP Window Update] 63564 → 8000 [ACK]
16009	2712.628973	192.168.1.110	192.168.1.110	TCP	44	63564 → 8000 [FIN, ACK] Seq=2535 Ack=138986
16010	2712.629006	192.168.1.110	192.168.1.110	TCP	44	8000 → 63564 [ACK] Seq=138986 Ack=2536
16011	2712.629136	192.168.1.110	192.168.1.110	TLSv1.3	68	Application Data
16012	2712.629187	192.168.1.110	192.168.1.110	TCP	44	63564 → 8000 [RST, ACK] Seq=2536 Ack=138986
16013	2712.702040	192.168.1.110	192.168.1.110	TCP	44	[TCP Retransmission] 63559 → 8000 [FIN, ACK]
16014	2712.702072	192.168.1.110	192.168.1.110	TCP	44	[TCP ZeroWindow] 8000 → 63559 [ACK] Seq=138986

0101 = Header Length: 20 bytes (5)

- Flags: 0x018 (PSH, ACK)
- Window: 8432
- [Calculated window size: 2158592]
- [Window size scaling factor: 256]
- Checksum: 0xdf3f [unverified]
- [Checksum Status: Unverified]
- Urgent Pointer: 0
- Timestamps
 - [Time since first frame in this TCP stream: 0.009679000 seconds]
 - [Time since previous frame in this TCP stream: 0.000130000 seconds]
- [SEQ/ACK analysis]
 - [iRTT: 0.000108000 seconds]
 - [Bytes in flight: 24]
 - [Bytes sent since last PSH flag: 24]
- TCP payload (24 bytes)
- Transport Layer Security
 - TLSv1.3 Record Layer: Application Data Protocol: Application Data
 - Opaque Type: Application Data (23)
 - Version: TLS 1.2 (0x0303)
 - Length: 19
 - Encrypted Application Data: da47a624eeaa5ab0ef1aa805110b9ebce04e7e

0000 02 00
0010 c0 a0
0020 a6 70
0030 13 d0
0040 bc e0

wireshark_Adapter for loopback traffic captureH7CT62.pcapng

Here we can't really use anything useful in benefit of knowing the credentials of the user its all encrypted, all we can know is the length of the data and as we can see beneath it it's encrypted so using HTTPS is way safer than using just HTTP

Encrypted data be like:

Encrypted Application Data: da47a624eeaa5ab0ef1aa805110b9ebce04e7e

Benefits of HTTPS:

- The connection between client and network is encrypted and secure
- No user data or password are shown in the packets
- Even though the traffic is visible but it's data is encrypted and nearly impossible to break
- HTTPS ensures confidentiality and prevents MITM attack

Conclusion

- HTTP is vulnerable to the network and not safe especially for logging and signing up
- 2FA can protect HTTP by adding another layer of security but it's still risky to insert your data on a vulnerable (HTTP) server
- This experience proves that combining 2FA with HTTPS improves the security of the account and protects from most common attacks