Cybersecurity Lab 04

Ritik Tiwari (B21CS098)

Step 1: Capture Network Traffic

5184	64.829829	157.240.239.60	172.31.48.119	TLSv1.2	126	Application Data
5185	64.873630	172.31.48.119	157.240.239.60	TCP	54	57822 → 443 [ACK] Seq=211 Ack=217 Win=516 Len=0
5186	65.559755	20.192.44.78	172.31.48.119	TLSv1.2	81	Application Data
5187	65.605855	172.31.48.119	20.192.44.78	TCP	54	57265 → 443 [ACK] Seq=212 Ack=136 Win=512 Len=0
5188	66.437770	172.31.48.119	172.16.100.160	HTTP	630	GET /ERP_IITJ/logout.do HTTP/1.1
5189	66.604660	172.16.100.160	172.31.48.119	TCP	54	8080 → 50889 [ACK] Seq=848221 Ack=5937 Win=64128 Len=0
5190	66.604660	172.16.100.160	172.31.48.119	TCP	1514	8080 → 50889 [ACK] Seq=848221 Ack=5937 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
5191	66.604660	172.16.100.160	172.31.48.119	TCP	1514	8080 → 50889 [ACK] Seq=849681 Ack=5937 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
5192	66.604874	172.31.48.119	172.16.100.160	TCP	54	50889 → 8080 [ACK] Seq=5937 Ack=851141 Win=131328 Len=0
5193	66.605135	172.16.100.160	172.31.48.119	TCP	1514	8080 → 50889 [ACK] Seq=851141 Ack=5937 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
5194	66.605135	172.16.100.160	172.31.48.119	TCP	1514	8080 → 50889 [ACK] Seq=852601 Ack=5937 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
5195	66.605135	172.16.100.160	172.31.48.119	TCP	1514	8080 → 50889 [PSH, ACK] Seq=854061 Ack=5937 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
5196	66.605309	172.31.48.119	172.16.100.160	TCP	54	50889 → 8080 [ACK] Seq=5937 Ack=855521 Win=131328 Len=0
5197	66.804223	172.16.100.160	172.31.48.119	HTTP	212	HTTP/1.1 200 (text/html)
5198	66.828160	172.31.48.119	34.120.195.249	TLSv1.3	211	Application Data
5199	66.828507	172.31.48.119	34.120.195.249	TLSv1.3	544	Application Data
5200	66.856016	172.31.48.119	172.16.100.160	TCP	54	50889 → 8080 [ACK] Seq=5937 Ack=855679 Win=131072 Len=0
5201	66.874462	172.16.100.160	172.31.48.119	TCP	212	[TCP Spurious Retransmission] 8080 → 50889 [PSH, ACK] Seq=855521 Ack=5937 Win=64128 Len=158 [TCP segment of
5202	66.874546	172.31.48.119	172.16.100.160	TCP	66	[TCP Dup ACK 5200#1] 50889 → 8080 [ACK] Seq=5937 Ack=855679 Win=131072 Len=0 SLE=855521 SRE=855679
5203	66.875462	34.120.195.249	172.31.48.119	TCP	66	[TCP Window Update] 443 → 50909 [ACK] Seq=1237 Ack=2099 Win=72448 Len=0 SLE=2256 SRE=2746
5204	66.875462	34.120.195.249	172.31.48.119	TCP	54	443 → 50909 [ACK] Seq=1237 Ack=2746 Win=74240 Len=0
5205	67.012920	34.120.195.249	172.31.48.119	TLSv1.3	125	Application Data
5206	67.012920	34.120.195.249	172.31.48.119	TLSv1.3	87	Application Data
5207	67.012920	34.120.195.249	172.31.48.119	TLSv1.3	93	Application Data
5208	67.013115	172.31.48.119	34.120.195.249	TCP	54	50909 → 443 [ACK] Seq=2746 Ack=1380 Win=129792 Len=0
5209	67.013808	172.31.48.119	34.120.195.249	TLSv1.3	89	Application Data
5210	67.013852	172.31.48.119	34.120.195.249	TLSv1.3	93	Application Data
5211	67.118680	34.120.195.249	172.31.48.119	TCP	66	[TCP Window Update] 443 → 50909 [ACK] Seq=1380 Ack=2746 Win=76032 Len=0 SLE=2781 SRE=2820

Step 2: Apply Display Filters

Apply filter to show only the HTTP packets.

507	18.570013	210.148.85.30	172.31.48.119	HTTP	353	HTTP/1.1 200 OK (image/jpeg)
1045	47.214428	172.31.48.119	172.16.100.160	HTTP	497	GET /ERP_IITJ/ HTTP/1.1
1066	48.335023	172.16.100.160	172.31.48.119	HTTP	316	HTTP/1.1 200 (text/html)
1067	48.336400	172.31.48.119	172.16.100.160	HTTP	483	GET /ERP_IITJ/css/bootstrap.css HTTP/1.1
1084	48.337430	172.31.48.119	172.16.100.160	HTTP	479	GET /ERP_IITJ/css/style.css HTTP/1.1
1085	48.337625	172.31.48.119	172.16.100.160	HTTP	486	GET /ERP_IITJ/css/font-awesome.css HTTP/1.1
1086	48.337766	172.31.48.119	172.16.100.160	HTTP	488	GET /ERP_IITJ/css/SidebarNav.min.css HTTP/1.1
1087	48.337873	172.31.48.119	172.16.100.160	HTTP	483	GET /ERP_IITJ/css/googleapi.css HTTP/1.1
1154	48.357328	172.31.48.119	172.16.100.160	HTTP	480	GET /ERP_IITJ/css/custom.css HTTP/1.1
1480	48.590163	172.16.100.160	172.31.48.119	HTTP	1322	HTTP/1.1 200 (text/css)
1481	48.590163	172.16.100.160	172.31.48.119	HTTP	699	HTTP/1.1 200 (text/css)
1482	48.590163	172.16.100.160	172.31.48.119	HTTP	116	HTTP/1.1 200 (text/css)
1484	48.593028	172.31.48.119	172.16.100.160	HTTP	474	GET /ERP_IITJ/js/jquery-1.11.1.min.js HTTP/1.1
1485	48.593217	172.31.48.119	172.16.100.160	HTTP	473	GET /ERP_IITJ/js/modernizr.custom.js HTTP/1.1
1486	48.593354	172.31.48.119	172.16.100.160	HTTP	470	GET /ERP_IITJ/js/metisMenu.min.js HTTP/1.1
1524	48.609846	172.16.100.160	172.31.48.119	HTTP	717	HTTP/1.1 200 (text/css)
1529	48.611171	172.31.48.119	172.16.100.160	HTTP	463	GET /ERP_IITJ/js/custom.js HTTP/1.1
1588	48.625453	172.16.100.160	172.31.48.119	HTTP	152	HTTP/1.1 200 (text/css)
1591	48.628937	172.16.100.160	172.31.48.119	HTTP	960	HTTP/1.1 200 (text/css)
1593	48.643438	172.31.48.119	172.16.100.160	HTTP	534	GET /ERP_IITJ/images/beta-banner.png HTTP/1.1
1594	48.646843	172.31.48.119	172.16.100.160	HTTP	532	GET /ERP_IITJ/images/iitrlogo1.jpg HTTP/1.1
1620	48.840716	172.16.100.160	172.31.48.119	HTTP	678	HTTP/1.1 200 (text/javascript)
1621	48.840716	172.16.100.160	172.31.48.119	HTTP	630	HTTP/1.1 200 (text/javascript)
1623	48.841208	172.31.48.119	210.148.85.30	HTTP	204	GET /api/check/online?t=1708444754 HTTP/1.1
1627	48.868815	172.16.100.160	172.31.48.119	HTTP	1060	HTTP/1.1 200 (text/javascript)

Step 3: Examine HTTP Requests and Responses.

1) Get Requests from the client side.

1066	48.335023	172.16.100.160	172.31.48.119	HTTP	316	HTTP/1.1 200 (text/html)
1067	48.336400	172.31.48.119	172.16.100.160	HTTP	483	GET /ERP_IITJ/css/bootstrap.css HTTP/1.1
1084	48.337430	172.31.48.119	172.16.100.160	HTTP	479	GET /ERP_IITJ/css/style.css HTTP/1.1
1085	48.337625	172.31.48.119	172.16.100.160	HTTP	486	GET /ERP_IITJ/css/font-awesome.css HTTP/1.1
1086	48.337766	172.31.48.119	172.16.100.160	HTTP	488	GET /ERP_IITJ/css/SidebarNav.min.css HTTP/1.1
1087	48.337873	172.31.48.119	172.16.100.160	HTTP	483	GET /ERP_IITJ/css/googleapi.css HTTP/1.1
1154	48.357328	172.31.48.119	172.16.100.160	HTTP	480	GET /ERP_IITJ/css/custom.css HTTP/1.1
1480	48.590163	172.16.100.160	172.31.48.119	HTTP	1322	HTTP/1.1 200 (text/css)

2) Post Request from the client side.

2223	54.086848	172.31.48.119	172.16.100.160	HTTP	843 POST /ERP_IITJ/login.do HTTP/1.1 (application/x-www-form-urlencoded)
2235	54.374552	172.16.100.160	172.31.48.119	HTTP	217 HTTP/1.1 200 (text/html)
2255	56.999669	172.31.48.119	172.16.100.160	HTTP	799 POST /ERP_IITJ/login.do HTTP/1.1 (application/x-www-form-urlencoded)
2294	57.322154	172.31.48.119	172.16.100.160	HTTP	470 GET /ERP_IITJ/js/Chart.js HTTP/1.1
2295	57.326510	172.31.48.119	172.16.100.160	HTTP	478 GET /ERP_IITJ/js/jquery-1.12.4.js HTTP/1.1

3) Status Codes with 200 (Ok), 404 (Not Found) and 500 (Internal Server Error)

HTTP requests with 200 response status code.

.528620	172.31.48.119	210.148.85.30	HTTP	204 GET /api/check/online?t=1708445517 HTTP/1.1
.761523	210.148.85.30	172.31.48.119	HTTP	349 HTTP/1.1 200 OK (image/jpeg)
.638155	172.31.48.119	172.16.100.160	HTTP	656 GET /ERP_IITJ/logout.do HTTP/1.1
.030921	172.16.100.160	172.31.48.119	HTTP	212 HTTP/1.1 200 (text/html)
.880144	172.31.48.119	172.16.100.5	HTTP	431 GET /images/logo.ico HTTP/1.1
.172223	172.16.100.5	172.31.48.119	HTTP	515 HTTP/1.1 302 Found (text/html)
.694928	172.31.48.119	172.16.100.160	HTTP	800 POST /ERP_IITJ/login.do HTTP/1.1 (application/x-w
.965157	172.16.100.160	172.31.48.119	HTTP	217 HTTP/1.1 200 (text/html)
.461824	172.31.48.119	172.16.100.160	HTTP	799 POST /ERP_IITJ/login.do HTTP/1.1 (application/x-w
.663715	172.31.48.119	172.16.100.5	HTTP	446 GET /dept_faculty_pic/suchetana.jpg HTTP/1.1
.663883	172.31.48.119	172.16.100.5	HTTP	437 GET /staff_pics/he.jpg HTTP/1.1

HTTP request with 404 response status code.

.528620	172.31.48.119	210.148.85.30	HTTP	204 GET /api/check/online?t=1708445517 HTTP/1.1
.761523	210.148.85.30	172.31.48.119	HTTP	349 HTTP/1.1 200 OK (image/jpeg)
.638155	172.31.48.119	172.16.100.160	HTTP	656 GET /ERP_IITJ/logout.do HTTP/1.1
.030921	172.16.100.160	172.31.48.119	HTTP	212 HTTP/1.1 200 (text/html)
.880144	172.31.48.119	172.16.100.5	HTTP	431 GET /images/logo.ico HTTP/1.1
.172223	172.16.100.5	172.31.48.119	HTTP	515 HTTP/1.1 302 Found (text/html)
.694928	172.31.48.119	172.16.100.160	HTTP	800 POST /ERP_IITJ/login.do HTTP/1.1 (application/x-w
.965157	172.16.100.160	172.31.48.119	HTTP	217 HTTP/1.1 200 (text/html)
.461824	172.31.48.119	172.16.100.160	HTTP	799 POST /ERP_IITJ/login.do HTTP/1.1 (application/x-w
.663715	172.31.48.119	172.16.100.5	HTTP	446 GET /dept_faculty_pic/suchetana.jpg HTTP/1.1
.663883	172.31.48.119	172.16.100.5	HTTP	437 GET /staff_pics/he.jpg HTTP/1.1

4) Post Request Header

Here post request is done by the client side on the ERP/IITJ/LOGIN_do

```
00 00 5e 00 01 0a c8 94  02 37 aa 53 08 00 45 00   ··^····· ·7·S··E·
03 11 89 8f 40 00 80 06  81 10 ac 1f 30 77 ac 10   ····@··· ····0w··
64 a0 c8 9f 1f 90 0b 10  67 6f 5b 0c 60 f6 50 18   d····· go[·`·P·
02 00 28 a0 00 00 50 4f  53 54 20 2f 45 52 50 5f   ··(···PO ST /ERP_
49 49 54 4a 2f 6c 6f 67  69 6e 2e 64 6f 20 48 54   IITJ/log in.do HT
54 50 2f 31 2e 31 0d 0a  48 6f 73 74 3a 20 31 37   TP/1.1·· Host: 17
32 2e 31 36 2e 31 30 30  2e 31 36 30 3a 38 30 38   2.16.100 .160:808
30 0d 0a 43 6f 6e 6e 65  63 74 69 6f 6e 3a 20 6b   0··Conne ction: k
65 65 70 2d 61 6c 69 76  65 0d 0a 43 6f 6e 74 65   eep-aliv e··Conte
6e 74 2d 4c 65 6e 67 74  68 3a 20 33 38 0d 0a 43   nt-Lengt h: 38··C
61 63 68 65 2d 43 6f 6e  74 72 6f 6c 3a 20 6d 61   ache-Con trol: ma
78 2d 61 67 65 3d 30 0d  0a 55 70 67 72 61 64 65   x-age=0· ·Upgrade
2d 49 6e 73 65 63 75 72  65 2d 52 65 71 75 65 73   -Insecur e-Reques
74 73 3a 20 31 0d 0a 4f  72 69 67 69 6e 3a 20 68   ts: 1··O rigin: h
74 74 70 3a 2f 2f 31 37  32 2e 31 36 2e 31 30 30   ttp://17 2.16.100
2e 31 36 30 3a 38 30 38  30 0d 0a 43 6f 6e 74 65   .160:808 0··Conte
05 · Time: 72.700561 · Source: 172.31.48.119 · Destination: 172.16.100.160 · Protocol: HTTP · Length: 799 · Info: POST /ERP_IITJ/login.do HTTP/1.1 (application/x-www-form-urlencoded)
ow packet bytes
```

Step 4: Check for Unusual URIs and Parameters

```
Hypertext Transfer Protocol
  POST /ERP_IITJ/login.do HTTP/1.1\r\n
    [Expert Info (Chat/Sequence): POST /ERP_IITJ/login.do HTTP/1.1\r\n]
      Request Method: POST
      Request URI: /ERP_IITJ/login.do
      Request Version: HTTP/1.1
    Host: 172.16.100.160:8080\r\n
    Connection: keep-alive\r\n
    Content-Length: 38\r\n
    Cache-Control: max-age=0\r\n
    Upgrade-Insecure-Requests: 1\r\n
    Origin: http://172.16.100.160:8080\r\n
    Content-Type: application/x-www-form-urlencoded\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 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,application/signed-exchange;
    Referer: http://172.16.100.160:8080/ERP_IITJ/login.do\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: en-US,en;q=0.9\r\n
    Cookie: JSESSIONID=649B2E5D9E05F31EA406F7817C0A9F27; SERVER=tomcat3\r\n
    \r\n
    [Full request URI: http://172.16.100.160:8080/ERP_IITJ/login.do]
    [HTTP request 3/4]
    [Prev request in frame: 2766]
    [Response in frame: 2814]
    [Next request in frame: 2837]
    File Data: 38 bytes
HTML Form URL Encoded: application/x-www-form-urlencoded
  Form item: "userid" = "tiwari.26"
      Key: userid
      Value: tiwari.26
  Form item: "password" = "tiwari.26531"
      Key: password
      Value: tiwari.26531
```

The vulnerability increases in the case of POST request. For example, in the case of http i.e. insecure request without ssl certificate the payload of the POST request can be seen in the wireshark. It means that the payload is readable and is not encrypted so intruder can see your credentials or other sensitive information if the request is done on a http URL.

Above is the screenshot of the [ERP portal](), while logging using the LDAP username and password.

Step 5: Analyse the Status Codes:

Here are some of the commonly encountered HTTP status code ranges 2xx- Success:

● These status codes indicate that the request was successfully received, understood, and accepted.

● Example: 200 OK- There quest was successful, and the server has returned the requested data.

3xx- Redirection:

● Theses tatus codes indicate that further action needs to be taken to complete the request. The client may need to follow a different URI or take additional steps.

● Example: 301 Moved Permanently- The requested resource has been permanently moved to a new location, and the client should use the new URI.

4xx- Client Error:

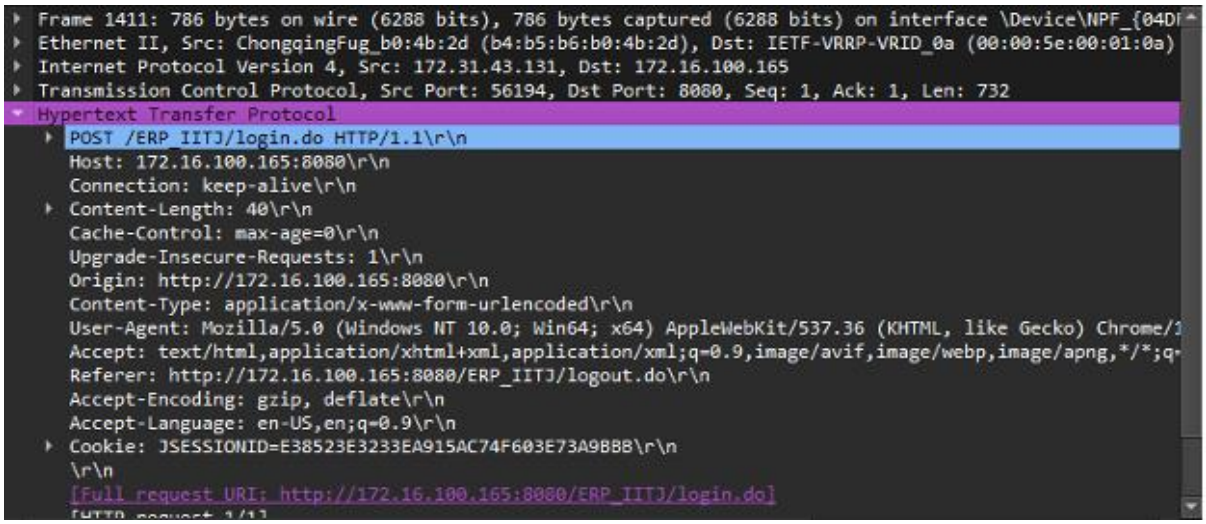
● These status codes indicate that the client seems to have made an error in the request, and the server cannot or will not process it.

● Example: 404 Not Found- The requested resource could not be found on the server. 5xx- Server Error:

● These status codes indicate that the server has encountered an error or is incapable of performing the request.

● Example: 500 Internal Server Error- A generic error message indicating that the server encountered an unexpected condition that prevented it from fulfilling the request.

Step 6: Look for Non-standard Headers



```
0000  00 00 5e 00 01 0a b4 b5  b6 b0 4b 2d 08 00 45 00   ··^····· ··K-··E·
0010  03 04 d1 6d 40 00 80 06  3e 2e ac 1f 2b 83 ac 10   ···m@··· >.··+···
0020  64 a5 db 82 1f 90 da dc  a2 bf 3d 31 c9 3d 50 18   d····· ··=1·=P·
0030  01 00 33 f8 00 00 50 4f  53 54 20 2f 45 52 50 5f   ··3···PO ST /ERP_
0040  49 49 54 4a 2f 6c 6f 67  69 6e 2e 64 6f 20 48 54   IITJ/log in.do HT
0050  54 50 2f 31 2e 31 0d 0a  48 6f 73 74 3a 20 31 37   TP/1.1·· Host: 17
0060  32 2e 31 36 2e 31 30 30  2e 31 36 35 3a 38 30 38   2.16.100 .165:808
0070  30 0d 0a 43 6f 6e 6e 65  63 74 69 6f 6e 3a 20 6b   0··Conne ction: k
0080  65 65 70 2d 61 6c 69 76  65 0d 0a 43 6f 6e 74 65   eep-aliv e··Conte
0090  6e 74 2d 4c 65 6e 67 74  68 3a 20 34 30 0d 0a 43   nt-Lengt h: 40··C
00a0  61 63 68 65 2d 43 6f 6e  74 72 6f 6c 3a 20 6d 61   ache-Con trol: ma
00b0  78 2d 61 67 65 3d 30 0d  0a 55 70 67 72 61 64 65   x-age=0·  ·Upgrade
00c0  2d 49 6e 73 65 63 75 72  65 2d 52 65 71 75 65 73   -Insecur e-Reques
00d0  74 73 3a 20 31 0d 0a 4f  72 69 67 69 6e 3a 20 68   ts: 1··O rigin: h
00e0  74 74 70 3a 2f 2f 31 37  32 2e 31 36 2e 31 30 30   ttp://17 2.16.100
00f0  2e 31 36 35 3a 38 30 38  30 0d 0a 43 6f 6e 74 65   .165:808 0··Conte
0100  6e 74 2d 54 79 70 65 3a  20 61 70 70 6c 69 63 61   nt-Type:  applica
0110  74 69 6f 6e 2f 78 2d 77  77 77 2d 66 6f 72 6d 2d   tion/x-w ww-form-
0120  75 72 6c 65 6e 63 6f 64  65 64 0d 0a 55 73 65 72   urlencod ed··User
0130  2d 41 67 65 6e 74 3a 20  4d 6f 7a 69 6c 6c 61 2f   -Agent:  Mozilla/
0140  35 2e 30 20 28 57 69 6e  64 6f 77 73 20 4e 54 20   5.0 (Win dows NT 
0150  31 30 2e 30 3b 20 57 69  6e 36 34 3b 20 78 36 34   10.0; Wi n64; x64
```

```
Origin: http://172.16.100.165:8080\r\n
Content-Type: application/x-www-form-urlencoded\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 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,application/signed-exchange;v=b3;q=0.7\r\n
Referer: http://172.16.100.165:8080/ERP_IITJ/logout.do\r\n
Accept-Encoding: gzip, deflate\r\n
```

In an API request, headers play a crucial role in conveying additional information about the request, the client, and how the server should handle the response. Headers are key-value pairs included in the HTTP request or response, and they provide metadata that supplements the basic information transmitted in the request or response body. Here are some common use cases for headers in API requests:

1. Authentication: Headers often include authentication information to verify the identity of the client making the request. Common authentication headers include Authorization, which may contain tokens or credentials. Example:
Authorization: Bearer YOUR_ACCESS_TOKEN

2. Content Type: The Content-Type header specifies the format of the data in the request body. This is important for the server to correctly interpret and process the incoming data.

Example: Content-Type: application/Json

3. Accept: The Accept header in a request indicates the preferred media types (content types) that the client can understand. It helps the server in providing the response in a format that the client can handle.
Example: Accept: application/Json

4. Custom Headers: APIs may define custom headers to carry specific information related to the application or the request. These headers are not standardized and are defined by the API itself.

Example: X-Custom-Header: Some Value

5. Conditional Requests: Headers like If-Match, If-None-Match, If-Modified-Since, and If-Unmodified-Since are used for conditional requests, allowing the client to specify conditions under which the request should be processed.
Example: If-None-Match: "etag123"

6. Caching: Headers such as Cache-Control and Expires control caching behaviour, helping to optimize network usage by specifying how long the response should be considered fresh.

Example: Cache-Control: max-age=3600

7. Compression: Headers like Accept-Encoding and Content-Encoding are used to negotiate content compression, helping to reduce the size of data transmitted over the network.

Example: Accept-Encoding: grip, deflate

8. User-Agent: The User-Agent header provides information about the client application or device making the request. It helps servers tailor responses based on the client type.

Example: User-Agent: MyAPIClient/1.0

Headers play a crucial role in enhancing the security of web applications by helping to prevent and mitigate various vulnerabilities. Here are some ways in which headers contribute to handling vulnerabilities:

● Cross-Origin Resource Sharing (CORS) Headers:

○ Vulnerability Addressed: Cross-Site Request Forgery (CSRF), Cross-Site Scripting (XSS)

○ Header: Access-Control-Allow-Origin, Access-Control-Allow-Methods, Access-Control-Allow-Headers

○ Description: CORS headers control which domains are allowed to make requests to your server, preventing unauthorized cross-origin requests. Properly configured CORSheaders can mitigate CSRF and XSS attacks.

● Content Security Policy (CSP) Header:

○ Vulnerability Addressed: Cross-Site Scripting (XSS)

○ Header: Content-Security-Policy ○ Description: CSP headers define a policy for the types of content that a browser should execute. They mitigate XSS attacks by controlling the sources from which scripts, styles, and other resources can be loaded.

● HTTP Strict Transport Security (HSTS) Header:

○ Vulnerability Addressed: Man-in-the-Middle (MITM) attacks, session hijacking

○ Header: Strict-Transport-Security

○ Description: HSTS headers enforce the use of HTTPS, preventing attackers from downgrading a secure connection to an insecure one. This helps mitigate MITM attacks and enhances the overall security of the communication channel.

● X-Content-Type-Options Header:

○ Vulnerability Addressed: MIME sniffing attacks ○ Header: X-Content-Type-Options: no sniff

○ Description: This header prevents browsers from interpreting files as a different MIME type than declared by the server. It helps prevent MIME sniffing attacks where an attacker may attempt to exploit inconsistencies in MIME type handling

Step 7: Follow-Up Actions:

Following up on the findings from the analysis of protocol-specific traffic is a crucial step in maintaining the security of your network. Here are some suggested follow-up actions based on the example scenario:

1. Blocking Suspicious IP Addresses: Identify the external IP addresses that were responsible for the unusual GET requests targeting specific paths (e.g., /admin.php, /login.jsp, /config.bin). You can block these IP addresses at the firewall or network perimeter to prevent further access attempts.

2. Tightening Firewall Rules: Review and update your firewall rules to restrict access to sensitive endpoints and directories. Implement rules that explicitly allow only necessary and authorized traffic while blocking or restricting access to potentially vulnerable areas.

3. Investigating Security of Targeted Endpoints: Examine the security configurations and vulnerabilities of the targeted endpoints (e.g., /admin.php, /login.jsp, /config.bin). Conduct security assessments, vulnerability scans, or penetration tests to identify and address any weaknesses.

4. Enhancing Web Application Security: If the targeted endpoints are part of a web application, consider implementing additional security measures such as input validation, parameterized queries, and proper session management to protect against common web application vulnerabilities