

Wireshark IP Packet Analysis: File Upload using html and node.js

Bibek Timilsina

July 13,2024

Introduction:

This report analyzes an IP packet for a file upload captured using Wireshark.

Node.js File upload:

```
const upload = multer({ storage: storage });

// Create uploads folder if it doesn't exist
const fs = require('fs');
const dir = './uploads';

if (!fs.existsSync(dir)){
  fs.mkdirSync(dir);
}

app.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, 'index.html'));
});

// Handle file upload
app.post('/upload', upload.single('file'), (req, res) => {
  console.log(req)
  console.log(req.file)
  if (!req.file) {
    return res.status(400).send('No file uploaded.');
```

Html for file upload:

```
<body>
  <h1>Upload a File</h1>
  <form id="uploadForm">
    <input type="file" id="fileInput" name="file" required>
    <button type="submit">Upload</button>
  </form>
  <div id="response"></div>

  <script>
    document.getElementById('uploadForm').addEventListener('submit', async (event) => {
      event.preventDefault();
      const fileInput = document.getElementById('fileInput');
      const formData = new FormData();
      formData.append('file', fileInput.files[0]);

      try {
        const response = await fetch('http://localhost:3000/upload', {
          method: 'POST',
          body: formData
        });
        const result = await response.text();
        document.getElementById('response').innerText = result;
      } catch (error) {
        console.error('Error uploading file:', error);
      }
    });
  </script>
</body>
</html>
```

Packet capture details:

The packet capture is performed using Wireshark with the following steps:

1. **Open Wireshark** and select the network interface used for local communication.(Adapter for loopback traffic capture in windows)
2. Start the capture.
3. Go to your browser and navigate to <http://localhost:3000> and upload a file using the html form.
4. Stop packet capture.
5. Filter to capture only the HTTP POST requests by using `http.request.method == "POST"`

The capture packet is analyzed in the following sections:

```

▶ Frame 337: 4434 bytes on wire (35472 bits), 4434 bytes captured (35472 bits) on interface \Device\NPF{...}
▶ Null/Loopback
▼ Internet Protocol Version 6, Src: ::1, Dst: ::1
    0110 .... = Version: 6
    ▼ .... 0000 0000 .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
        .... 0000 00.. .... = Differentiated Services Codepoint: Default (0)
        .... ..00 .... = Explicit Congestion Notification: Not ECN-Capable
    .... 1101 0100 0111 1110 0001 = Flow Label: 0xd47e1
    Payload Length: 4390
    Next Header: TCP (6)
    Hop Limit: 128
    Source Address: ::1
    Destination Address: ::1
▶ Transmission Control Protocol, Src Port: 51416, Dst Port: 3000, Seq: 443044, Ack: 1, Len: 4370

```

Figure: Headers Fields

```

[29 Reassembled TCP Segments (447413 bytes): #279(675), #281(16384), #283(16384), #285(16384)]
▶ Hypertext Transfer Protocol
4
00000000 50 4f 53 54 20 2f 75 70 6c 6f 61 64 20 48 54 54 POST /up load HT
00000010 50 2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 6c 6f 63 P/1.1..H ost: loc
00000020 61 6c 68 6f 73 74 3a 33 30 30 30 0d 0a 43 6f 6e alhost:3 000..Con
00000030 6e 65 63 74 69 6f 6e 3a 20 6b 65 65 70 2d 61 6c nection: keep-al
00000040 69 76 65 0d 0a 43 6f 6e 74 65 6e 74 2d 4c 65 6e ive..Con tent-Len
00000050 67 74 68 3a 20 34 34 36 37 33 38 0d 0a 73 65 63 gth: 446 738..sec
00000060 2d 63 68 2d 75 61 3a 20 22 4e 6f 74 2f 41 29 42 -ch-ua: "Not(A)B
00000070 72 61 6e 64 22 3b 76 3d 22 38 22 2c 20 22 43 68 rand";v= "8", "Ch
00000080 72 6f 6d 69 75 6d 22 3b 76 3d 22 31 32 36 22 2c romium"; v="126",
00000090 20 22 4d 69 63 72 6f 73 6f 66 74 20 45 64 67 65 "Micros oft Edge
000000a0 22 3b 76 3d 22 31 32 36 22 0d 0a 73 65 63 2d 63 ";v="126 " ..sec-c
000000b0 68 2d 75 61 2d 70 6c 61 74 66 6f 72 6d 3a 20 22 h-ua-pla tform: "
000000c0 57 69 6e 64 6f 77 73 22 0d 0a 73 65 63 2d 63 68 Windows" ..sec-ch
000000d0 2d 75 61 2d 6d 6f 62 69 6c 65 3a 20 3f 30 0d 0a -ua-mobi le: ?0..
000000e0 55 73 65 72 2d 41 67 65 6e 74 3a 20 4d 6f 7a 69 User-Age nt: Mozi
000000f0 6c 6c 61 2f 35 2e 30 20 28 57 69 6e 64 6f 77 73 lla/5.0 (Windows
00000100 20 4e 54 20 31 30 2e 30 3b 20 57 69 6e 36 34 3b NT 10.0 ; Win64;
00000110 20 78 36 34 29 20 41 70 70 6c 65 57 65 62 4b 69 x64) Ap pleWebKi
00000120 74 2f 35 33 37 2e 33 36 20 28 4b 48 54 4d 4c 2c t/537.36 (KHTML,
00000130 20 6c 69 6b 65 20 47 65 63 6b 6f 29 20 43 68 72 like Ge cko) Chr
00000140 6f 6d 65 2f 31 32 36 2e 30 2e 30 2e 30 20 53 61 ome/126. 0.0.0 Sa
00000150 66 61 72 69 2f 35 33 37 2e 33 36 20 45 64 67 2f fari/537 .36 Edg/
00000160 31 32 36 2e 30 2e 30 2e 30 0d 0a 43 6f 6e 74 65 126.0.0. 0..Conte
00000170 6e 74 2d 54 79 70 65 3a 20 6d 75 6c 74 69 70 61 nt-Type: multipa
00000180 72 74 2f 66 6f 72 6d 2d 64 61 74 61 3b 20 62 6f rt/form- data; bo
00000190 75 6e 64 61 72 79 3d 2d 2d 2d 2d 57 65 62 4b 69 undary=-- --WebKi
000001a0 74 46 6f 72 6d 42 6f 75 6e 64 61 72 79 5a 32 54 tFormBou ndaryZ2T
000001b0 6d 42 75 59 6b 68 70 71 71 66 34 66 36 0d 0a 41 mBuYkhpq qf4f6..A
000001c0 63 63 65 70 74 3a 20 2a 2f 2a 0d 0a 4f 72 69 67 ccept: * /*..Orig
000001d0 69 6e 3a 20 68 74 74 70 3a 2f 2f 6c 6f 63 61 6c in: http ://local
000001e0 68 6f 73 74 3a 33 30 30 30 0d 0a 53 65 63 2d 46 host:300 0..Sec-F

```

Frame (4434 bytes) Reassembled TCP (447413 bytes)

Figure: Hex for Ip packet

IP Header Data:

60 0d 47 e1 11 26 06 80 00 00 00 00 00 00 00 00 00 00 00 01 00
00 00 00 00 00 00 00 00 00 00 00 00 00 01

IP Header Analysis:

Field	Value(hex)	Length	Details
Version	6	4	IPv6 packet.
Traffic Class	00	8	Used to classify & prioritize packets. 0 for default traffic class
Flow Level	0d 47 e1	20	for special handling of particular flows of data.
Payload Length	11 62	12	Length of the payload following the IPv6 header (1162 bytes).
Next Header	06	8	Indicates the type of header following the IPv6 header
Hop Limit	80	8	Maximum number of hops the packet can traverse (similar to TTL in IPv4).
Source Address	::1	128	IPv6 address of the packet's source (loopback address).
Destination Address	::1	128	IPv6 address of the packet's destination (loopback address).

TCP Header Data:

Ce 5c 0b b8 c5 a2 19 fd e8 62 c3 97 50 18 27 f6 09 af 00 00

TCP Header details:

field Name	Length (bits)	Value	Explanation
Source Port	16	51416	Port number from which the packet was sent on the source device.
Destination Port	16	3000	Port number to which the packet is

			being sent on the destination device.
Sequence Number	32	443044	Sequence number of the first byte of data in this segment.
Acknowledgment Number	32	1	Next sequence number the sender expects to receive (if ACK flag is set).
Data Offset	4	5	Indicates where the data begins; number of 32-bit words in the TCP header.
Reserved	3	0	Reserved for future use; should be set to zero.
Flags	9	Various	Control flags (e.g., SYN, ACK, FIN, RST).
Window Size	16	4370	Size of the sender's receive window (buffer space available).
Checksum	16	Varies	Used for error-checking the header and data.
Urgent Pointer	16	0	Indicates the offset from the sequence number where the urgent data is (if URG flag is set).