# WS1: Introduction to Web Security

## Terms and Concepts

- Hypertext: formatted text designed for electronic display that contains references ("hyperlinks") to other texts and multimedia content (images, sounds, etc.).
- World Wide Web (WWW): a global network of interlinked hypertext documents; a subset of the Internet.
- **Protocol**: a set of rules governing data transmission between two computing components (hardware or software), e.g., HTTP.
- **Encoding**: a way to represent information.

## Examples of digital encodings:

Images: JPG, PNG, etc.Audio: MP3, FLAC, etc.

o Text: UTF-8, ASCII, ISO-8859-2, etc.

Binary data: Base64, etc.Hypertext: HTML, etc.

**Note**: there are also analog encodings of digital data, such as frequency, amplitude, and/or phase modulation of a signal for transmitting bits in modems or specific voltages (e.g., 0/5V for logical values 0/1) in CMOS logic gates.

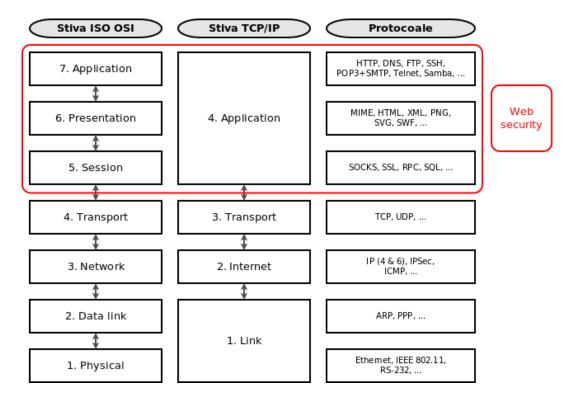
• **Encryption**: encoding that hides information from unauthorized entities (those without the decryption key).

Important: encryption ≠ encoding

• **File Format**: a protocol for encoding data in a file. Typically, files of a certain format have a specific extension, but this is not mandatory (e.g., a file named ana.txt may contain an image in PNG format).

# Web Protocols

### ISO OSI and TCP/IP Stacks



#### HTTP Protocol

- Client-server model (browser web server)
- Request-response mechanism, initiated by the client
- Status codes (numerical encoding of results):
  - 2xx: Success (e.g., 200 = OK)
  - 3xx: Redirection (e.g., 301 = Moved Permanently, 302 = Temporary Redirect)
  - 4xx: Client error (e.g., 400 = Bad Request, 401 = Unauthorized authentication, 403 = Forbidden - authorization, 404 = Not Found)
  - 5xx: Server error (e.g., 500 = Internal Server Error, 501 = Not Implemented, 503 = Service Unavailable)
  - Standardized statuses: <u>RFC 2616</u>

### Request exmple:

```
1. $ nc example.com 80
2. GET / HTTP/1.1
3. Host: example.com
```

### Response example:

```
HTTP/1.1 200 OK
   Accept-Ranges: bytes
3.
   Cache-Control: max-age=604800
   Content-Type: text/html
4.
5.
   Date: Thu, 19 Nov 2015 12:57:34 GMT
6.
    Etag: "359670651"
   Expires: Thu, 26 Nov 2015 12:57:34 GMT
7.
8.
   Last-Modified: Fri, 09 Aug 2013 23:54:35 GMT
9.
    Server: ECS (lga/13BA)
10. X-Cache: HIT
11. x-ec-custom-error: 1
12. Content-Length: 1270
13.
14. <!doctype html>
15. <html>
16. <head>
17. <title>Example Domain</title>
18. [...]
19. </html>
```

## Example using Curl:

```
$ curl -v example.com
    * Rebuilt URL to: example.com/
    * Trying 93.184.216.34...
3.
4.
    * Connected to example.com (93.184.216.34) port 80 (#0)
5.
    GET / HTTP/1.1
6.
7.
   Host: example.com
8.
   User-Agent: curl/7.43.0
9.
    Accept: */*
10.
11. HTTP/1.1 200 OK
12. Accept-Ranges: bytes
13. Cache-Control: max-age=604800
14. Content-Type: text/html
15. Date: Thu, 19 Nov 2015 12:57:50 GMT
16. Etag: "359670651"
17. Expires: Thu, 26 Nov 2015 12:57:50 GMT
18. Last-Modified: Fri, 09 Aug 2013 23:54:35 GMT
19. Server: ECS (ewr/15BD)
20. Vary: Accept-Encoding
21. X-Cache: HIT
22. x-ec-custom-error: 1
23. Content-Length: 1270
24.
25. <!doctype html>
26. <html>
27. <head>
28. <title>Example Domain</title>
29. [...]
30. </html>
```

## **Network Packet Structure**

## Example - Wireshark capture

```
Frame 69: 369 bytes on wire (2952 bits), 369 bytes captured (2952 bits) on interface 0
Ethernet II, Src: IntelCor_21:53:e0 (a0:36:9f:21:53:e0), Dst: Dell_
Internet Protocol Version 4, Src: 93.184.216.34 (93.184.216.34), Dst: 172.21.
Transmission Control Protocol, Src Port: 80 (80), Dst Port: 59326 (59326), Seq: 1289, Ack: 341, Len: 303
 [2 Reassembled TCP Segments (1591 bytes): #67(1288), #69(303)]
Hypertext Transfer Protocol
Line-based text data: text/html
 <!doctype html>\n
  <html>\n
  <head>\n
        <title>Example Domain</title>\n
        <meta charset="utf-8" />\n
        <meta http-equiv="Content-type" content="text/html; charset=utf-8" />\n
        <meta name="viewport" content="width=device-width, initial-scale=1" />\n
        <style type="text/css">\n
       body {\n background-color: #f0f0f2;\n
             margin: 0;\n
             padding: 0;\n
              font-family: "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;\n
        }\n
           88 54 54 50 2f 31 2e 31 20 32 30 30 20 4f 4b 0d
9a 41 63 63 65 70 74 2d 52 61 6e 67 65 73 3a 20
92 79 74 65 73 0d 0a 43 61 63 68 65 2d 43 6f 6e
0000
                                                                                    HTTP/1.1 200 OK.
.Accept- Ranges:
bytes..C ache-Con
0010
0020
         62 79 74 65 73 0d 0a 43 61 63 68 65 2d 43 61 66 74 72 61 6c 3a 20 6d 61 78 2d 61 67 65 3d 36 30 34 38 30 30 0d 0a 43 6f 6e 74 65 6e 74 2d 54 79 70 65 3a 20 74 65 78 74 2f 68 74 6d 6c 0d 0a 44 61 74 65 3a 20 57 65 64 2c 20 31 38 20 4e 6f 76 20 32 30 31 35 20 31 33 3a 35 32 3a 32 30 20 47 4d 54 0d 0a 45 74 61 67 3a 20 22 33 35 39 36 37
                                                                                    trol: ma x-age=60
4800..Co ntent-Ty
0030
0040
0050
                                                                                    pe: text /html..D
                                                                                    ate: Wed , 18 Nov
2015 13 :52:20 G
0060
0070
0080
                                                                                    0651"..E xpires:
Wed, 25 Nov 2015
13:52:2 0 GMT..L
         30 36 35 31 22 0d 0a 45 78 70 69 72 65 73 3a 20
57 65 64 2c 20 32 35 20  4e 6f 76 20 32 30 31 35
0090
00a0
                                             30 20 47 4d 54 0d 0a 4c
00b0
00c0
         69 2c 20 30 39 20 41 75  67 20 32 30 31 33 20 32
33 3a 35 34 3a 33 35 20  47 4d 54 0d 0a 53 65 72
00d0
00e0
00f0
0100
Frame (369 bytes) Reassembled TCP (1591 bytes)
```

### Diagram: packets structure

HTML response

		Ethernet frame		
Ethernet header	IP packet			
	IPv4 header	TCP segment		
		TCP header	HTTP response	
			HTTP header	HTML document
MAC addrs (src / dst) Next type: IP 	Ver: IPv4 IP addrs (src / dst) Flags, checksum Packet ID Next type: TCP 	Ports (src / dst) Flags, checksum Sequence number Options 	HTTP ver. Status Content-Type Caching information 	HTML head
				HTML body
	OS network stack (Linux kernel)		Web server (Apache)	Content generator (PHP)
OS network stack (Linux kernel)		Web browser (Firefox)		

## Unauthorized data modification (data tampering)

Tamper Dev is a browser add-on for Chromium-based browsers that intercepts communication between the client and web server, allowing modification of message content before it is sent to or received from the server. A user-initiated request from the browser can be modified using Tamper Dev and then sent to the server or canceled (default action).

#### Lab Exercises

1. Connect using netcat to www.utcluj.ro:80, send a GET request to fetch data from /universitatea/educatie/, and explain the result. Obtain the same content using wget and compare results. Follow the instructions from the page accessed with netcat; what is the result page containing?

**Hint:** you can use html2text to remove HTML tags and see only the actual text content from the document.

2. Write the following HTML code, save it in /var/www/html as test.html, and open it in a browser:

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4. <meta charset="utf-8">
5. <title>Login Form</title>
6. </head>
7. <body>
8. <section class="container">
9.
    <div class="login">
10.
      <h1>Login to Web App</h1>
      <form method="get" action="index.html">
11.
12.
       <input type="text" name="login" placeholder="Username or Email">
       <input type="password" name="password" placeholder="Password">
13.
       <input type="submit" name="commit" value="Login">
14.
15.
      </form>
     </div>
16.
17. </section>
18. </body>
19. </html>
```

Access the page using a browser: <a href="http://localhost/test.html">http://localhost/test.html</a>. Click the Login button, intercept the request using Tamper Dev, and analyze the URL and headers. Modify the request type from GET to POST, intercept the request again, and explain the differences.

**Hint:** to complete this exercise, you need a locally installed web server. Installing and configuring a complex server (e.g., Apache) is not necessary; instead, you can use built-in extensions from Python or Node. For example, to serve resources from a directory using Python, run the following command in that directory: **python3 -m http.server**. This will start a mini web server on the default port 8000. You can access the web page by navigating to: http://localhost:8000/

3. Authenticate on <a href="https://websinu.utcluj.ro">https://websinu.utcluj.ro</a> and analyze transmitted packets using Wireshark. Observe username and password parameters. Perform the same research using Tamper Dev and explain differences between browser-level parameters and intercepted packets.

# References

- ISO OSI: <a href="https://en.wikipedia.org/wiki/OSI\_model">https://en.wikipedia.org/wiki/OSI\_model</a>
- HTTP 1.1 standard: https://datatracker.ietf.org/doc/html/rfc2616
- HTTP Status Codes: <a href="https://en.wikipedia.org/wiki/List\_of-HTTP\_status\_codes">https://en.wikipedia.org/wiki/List\_of-HTTP\_status\_codes</a>
- Tamper Dev Tamper Dev