CYBERSECURITY LAB #4

Jocelyne Elias
Giulio Tripi – Tutor didattico

<jocelyne.elias, giulio.tripi>@unibo.it

Exercise

Complete the exercises, taking notes of all the steps that you take

Write a **small** report and upload it on Virtuale

Remember: write name, surname and the number of the lab session on the report!

Prerequisites

Virtualbox and the configured Kali VM.
Instructions are on Virtuale!
Also, we will see Wireshark

Wireshark

It is a free and open-source **packet analyzer**.

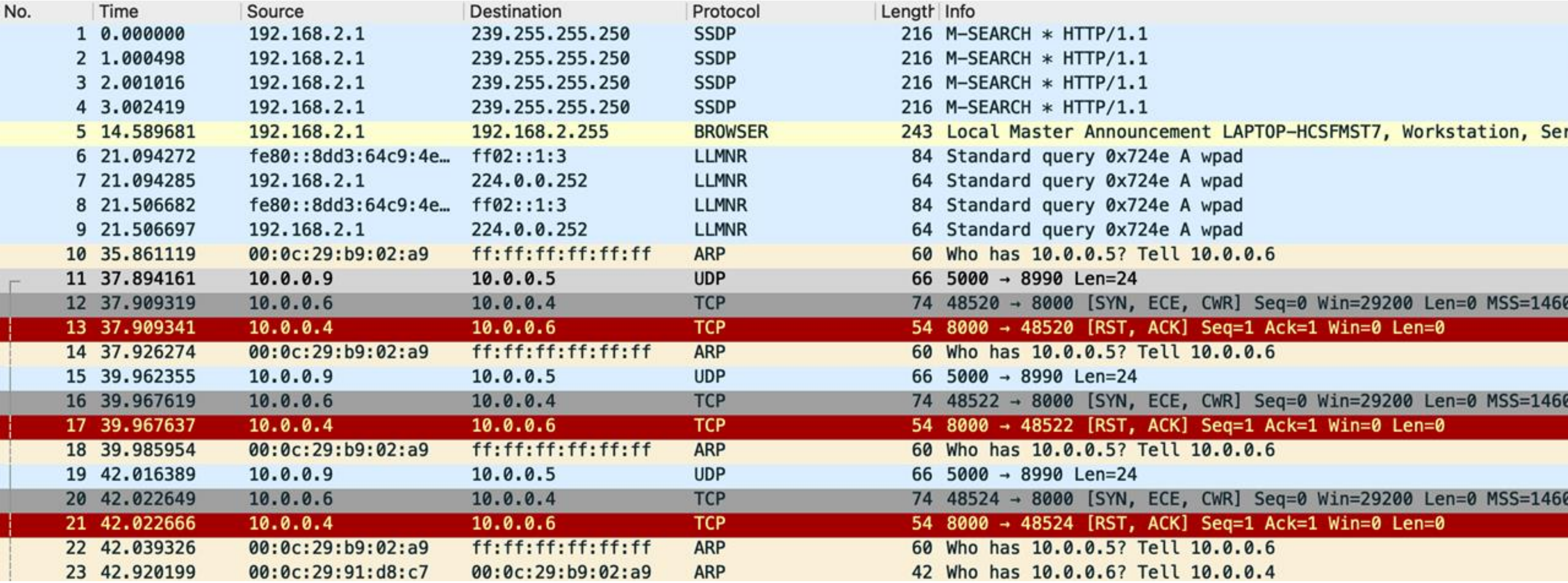
Similar to tcpdump, but it has a **graphical interface** to show sniffed packets.

Has a lot of **filtering capability** to find the packets that you want.

Wireshark

- It can perform **real-time analysis** or on **previously recorded traffic file** (e.g. **PCAP** files)
- It shows a **packet list with a summary** of each of them
  - If you click on one, it will show all the details of every TCP/IP layer with their respective protocol
  - Wireshark could be wrong with the dissection rules (e.g. based on port)
- **Filtering** capability
  - You can select a property of a specific packet and set it as a filtering rule
- You can follow streams

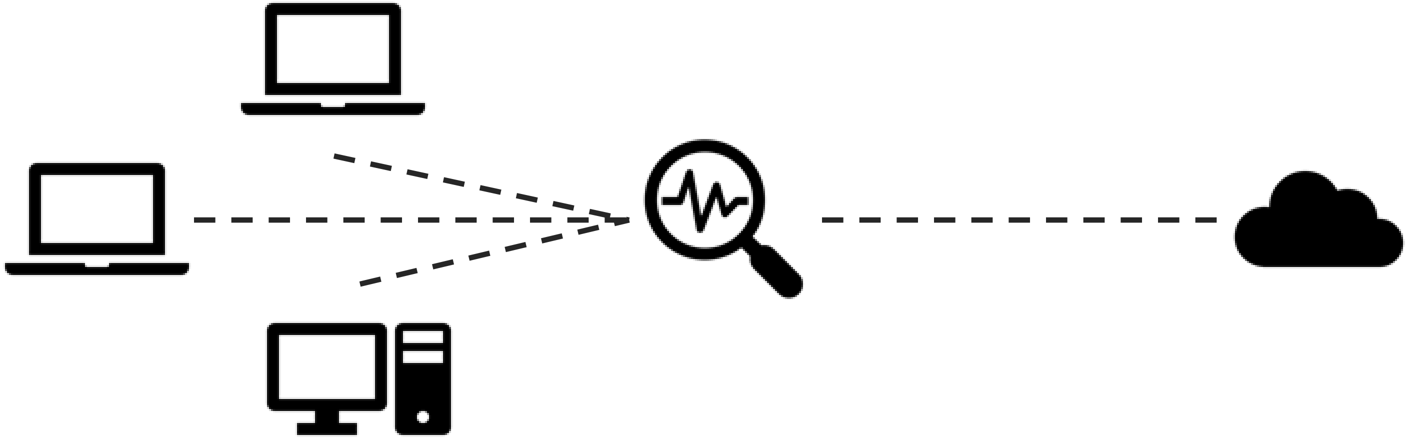
What can we see from a packet analyzer?



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.2.1	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
2	1.000498	192.168.2.1	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
3	2.001016	192.168.2.1	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
4	3.002419	192.168.2.1	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
5	14.589681	192.168.2.1	192.168.2.255	BROWSER	243	Local Master Announcement LAPTOP-HCSFMST7, Workstation, Ser
6	21.094272	fe80::8dd3:64c9:4e…	ff02::1:3	LLMNR	84	Standard query 0x724e A wpad
7	21.094285	192.168.2.1	224.0.0.252	LLMNR	64	Standard query 0x724e A wpad
8	21.506682	fe80::8dd3:64c9:4e…	ff02::1:3	LLMNR	84	Standard query 0x724e A wpad
9	21.506697	192.168.2.1	224.0.0.252	LLMNR	64	Standard query 0x724e A wpad
10	35.861119	00:0c:29:b9:02:a9	ff:ff:ff:ff:ff:ff	ARP	60	Who has 10.0.0.5? Tell 10.0.0.6
11	37.894161	10.0.0.9	10.0.0.5	UDP	66	5000 → 8990 Len=24
12	37.909319	10.0.0.6	10.0.0.4	TCP	74	48520 → 8000 [SYN, ECE, CWR] Seq=0 Win=29200 Len=0 MSS=146
13	37.909341	10.0.0.4	10.0.0.6	TCP	54	8000 → 48520 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
14	37.926274	00:0c:29:b9:02:a9	ff:ff:ff:ff:ff:ff	ARP	60	Who has 10.0.0.5? Tell 10.0.0.6
15	39.962355	10.0.0.9	10.0.0.5	UDP	66	5000 → 8990 Len=24
16	39.967619	10.0.0.6	10.0.0.4	TCP	74	48522 → 8000 [SYN, ECE, CWR] Seq=0 Win=29200 Len=0 MSS=146
17	39.967637	10.0.0.4	10.0.0.6	TCP	54	8000 → 48522 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
18	39.985954	00:0c:29:b9:02:a9	ff:ff:ff:ff:ff:ff	ARP	60	Who has 10.0.0.5? Tell 10.0.0.6
19	42.016389	10.0.0.9	10.0.0.5	UDP	66	5000 → 8990 Len=24
20	42.022649	10.0.0.6	10.0.0.4	TCP	74	48524 → 8000 [SYN, ECE, CWR] Seq=0 Win=29200 Len=0 MSS=146
21	42.022666	10.0.0.4	10.0.0.6	TCP	54	8000 → 48524 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
22	42.039326	00:0c:29:b9:02:a9	ff:ff:ff:ff:ff:ff	ARP	60	Who has 10.0.0.5? Tell 10.0.0.6
23	42.920199	00:0c:29:91:d8:c7	00:0c:29:b9:02:a9	ARP	42	Who has 10.0.0.6? Tell 10.0.0.4

Why should we use a packet analyzer?

- Monitor (***sniffing***) the traffic could be essential for the discovery of attacks and/or anomalies
- But also to perform attacks.....



Where should we use a packet analyzer?

Network

- To listen to all the traffic coming from and to **different hosts**

Host

- To listen to all the traffic coming from and to the **actual host**
  - I can see also **application data layers**!

But.. From which interface?

Your computer could have more than one, in general we can choose **wired or wireless interfaces**.

In the case of wired, you can see only the traffic directed/coming exactly to/from you (Ethernet case).

But, **in the case of wireless, I can “sniff” everything…**

Public wireless networks

In public Wi-Fi network there is **no encryption**, so:

Others can **sniff** your packets!

Or maybe worse, others can perform a **Man in the Middle Attack** (*MitM*).

It depends on algorithms

In Wi-Fi we can choose different algorithms:

- WEP
  - Stream cipher RC4 algorithm, CRC32 checksum, **INSECURE**
- WPA
  - TKIP
- WPA2
  - Not RC4 anymore..
- WPA3

And passwords!!

- As always, weak password can be **cracked**
  - Remember brute-force/dictonary attacks?

"Protected" wireless networks

Encryption makes your packets confidential

...Right?

Attacking WEP

WEP was designed many years ago, now is **obsolete**.

If you know the shared key, you can decrypt every packet of the others.

It used Initializing Vector (IV), sent in plaintext in the packets, with few bits (24), so… Collecting some of them lead you to **crack (*mathematically derive*) the shared key**!

Attacking WPA

By **sniffing** the 4-way handshake we can perform an **offline** attack

Attackers can get all the informations to perform **dictionary and brute force attack**.

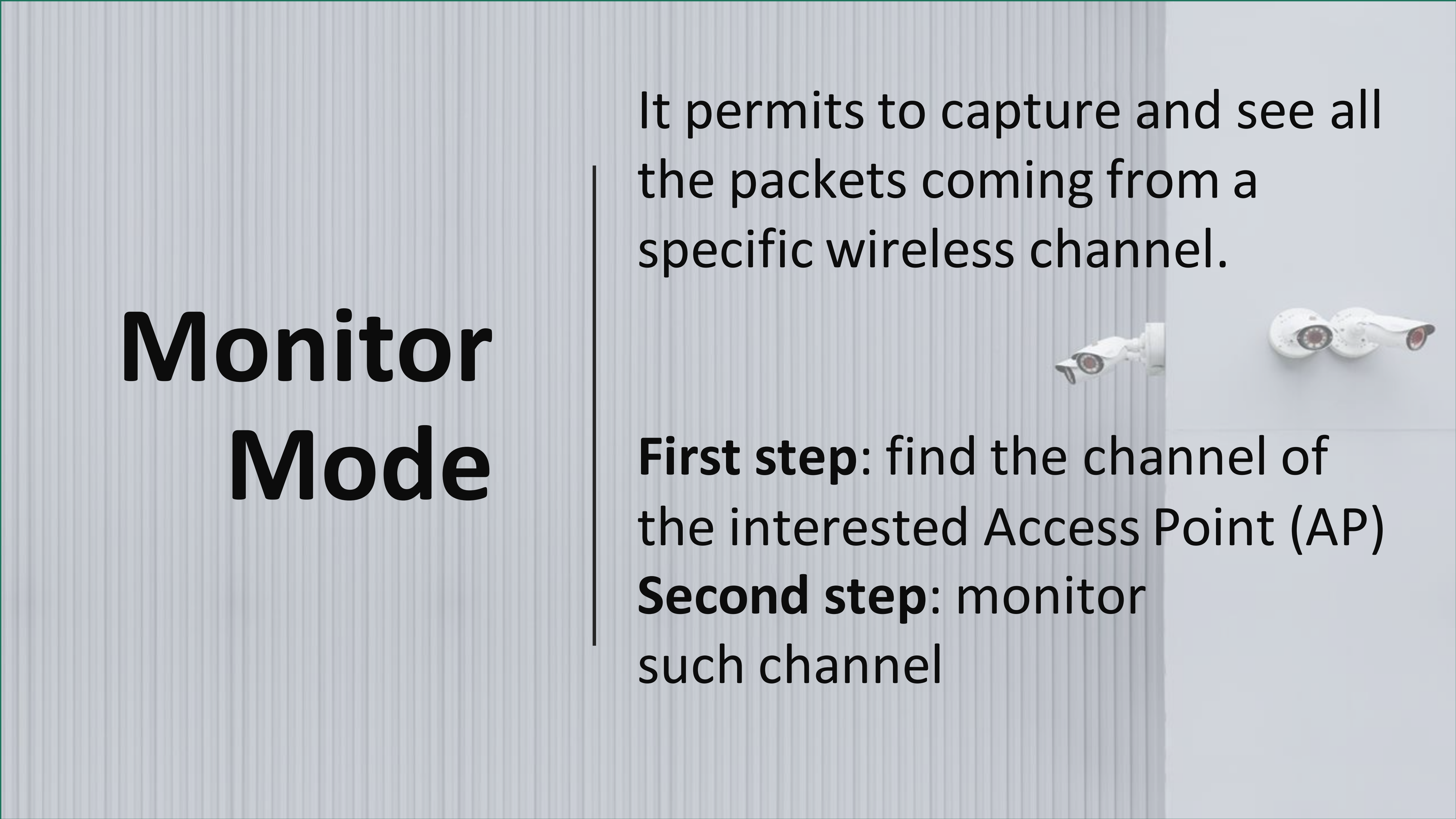
So, in this case, the probability of success it’s more password-related…

How to sniff packets “*in the air*”

- With physical cable, not considering the (old) *hub topologies*, we cannot see packets unless we are physically connected to the specific cables!
- With wireless instead, everything is in the *air*. So **we can simply listen to receive all the packets**.

That’s what is called **Monitor Mode**

Monitor Mode



It permits to capture and see all the packets coming from a specific wireless channel.

First step: find the channel of the interested Access Point (AP)
Second step: monitor such channel

Monitor mode in linux

- To enable monitor mode on your pc, it depends on the wireless card. You can follow the **aircrack-ng guide**: *https://www.aircrack-ng.org/doku.php?id=cracking_wpa*

- With **airmon-ng**, once you **capture a WPA handshake**, you can stop the analysis and go to the next step with the PCAP file obtained.
  - For the exercise, I will give you an example PCAP file.

Capturing the handshake

- With monitor mode we can capture the packets sent for the handshake

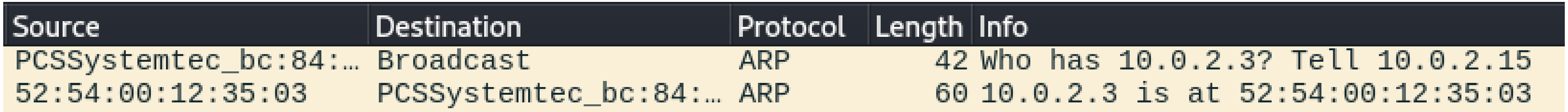
- Then, we save the .pcap file that contains the packets and we can try an **offline** dictionary/brute-force-based **attack**. You can also use Wireshark to investigate the packets.

Attacking

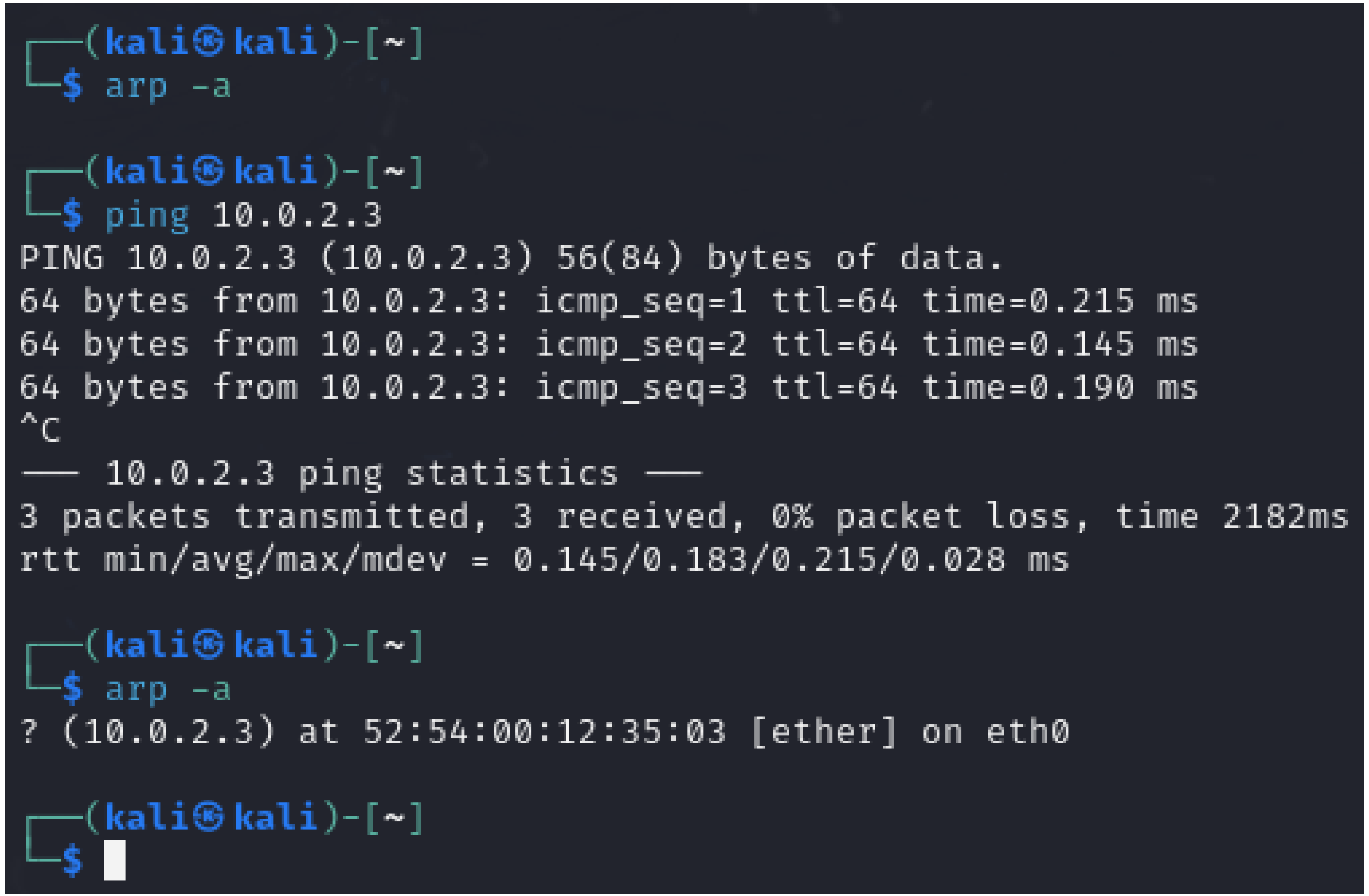
- You can use **aircrack-ng suite**, able to attack WEP and WPA/WPA2

- Usage: **aircrack-ng `(-w wordlist) (-b BSSID) (pcap file)`** `[OPTIONS]`
  - **Wordlist:** pass the wordlist to use for cracking the password
  - **BSSID:** MAC address of the interested AP
  - **PCAP file:** the file containing the sniffed handshake
  - **In OPTIONS:** settings for optimized WEP or WPA/2 attacks..

ARP protocol



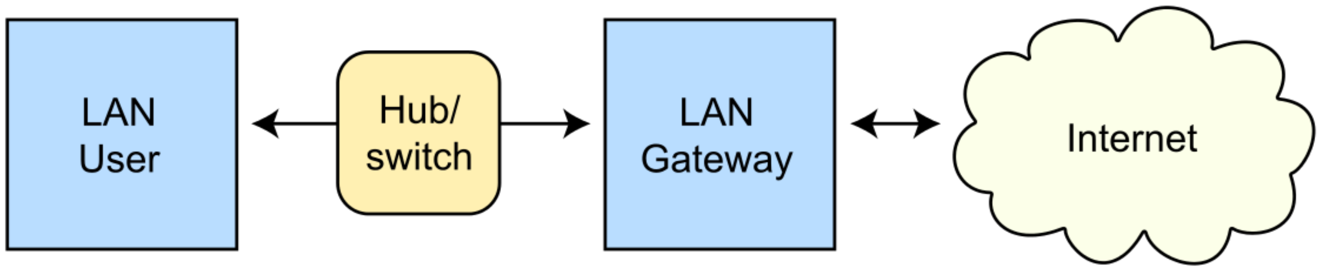
Source	Destination	Protocol	Length	Info
PCSSystemtec_bc:84:…	Broadcast	ARP	42	Who has 10.0.2.3? Tell 10.0.2.15
52:54:00:12:35:03	PCSSystemtec_bc:84:…	ARP	60	10.0.2.3 is at 52:54:00:12:35:03



```
┌──(kali㉿kali)-[~]
└─$ arp -a

┌──(kali㉿kali)-[~]
└─$ ping 10.0.2.3
PING 10.0.2.3 (10.0.2.3) 56(84) bytes of data.
64 bytes from 10.0.2.3: icmp_seq=1 ttl=64 time=0.215 ms
64 bytes from 10.0.2.3: icmp_seq=2 ttl=64 time=0.145 ms
64 bytes from 10.0.2.3: icmp_seq=3 ttl=64 time=0.190 ms
^C
--- 10.0.2.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2182ms
rtt min/avg/max/mdev = 0.145/0.183/0.215/0.028 ms

┌──(kali㉿kali)-[~]
└─$ arp -a
? (10.0.2.3) at 52:54:00:12:35:03 [ether] on eth0

┌──(kali㉿kali)-[~]
└─$ 
```

ARP poisoning

Routing under normal operation



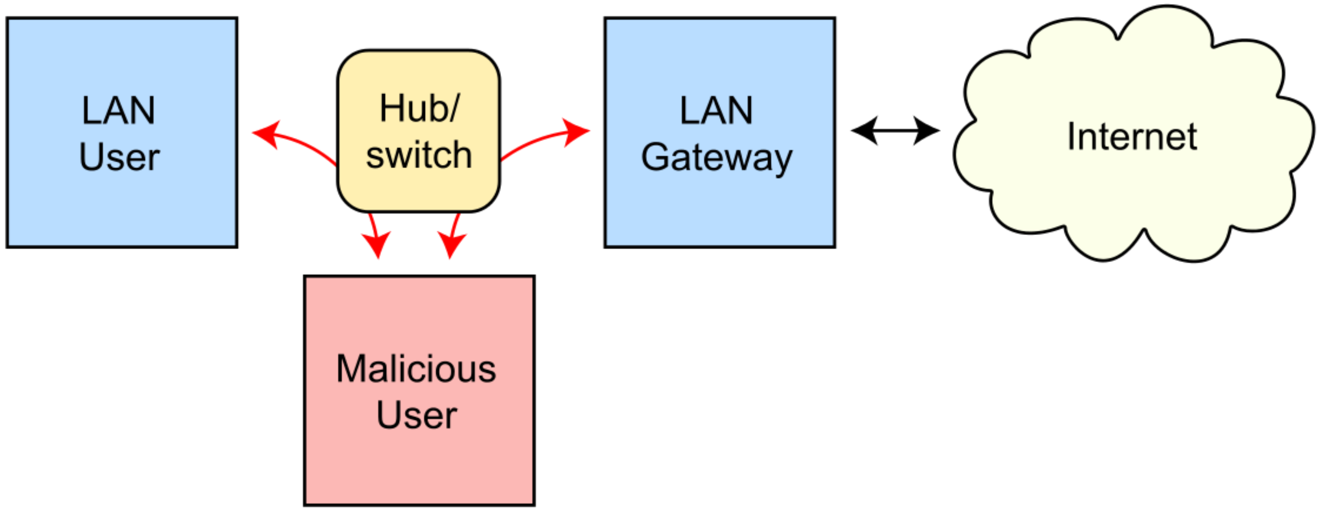
LAN User

Hub/ switch

LAN Gateway

Internet

Routing subject to ARP cache poisoning



LAN User

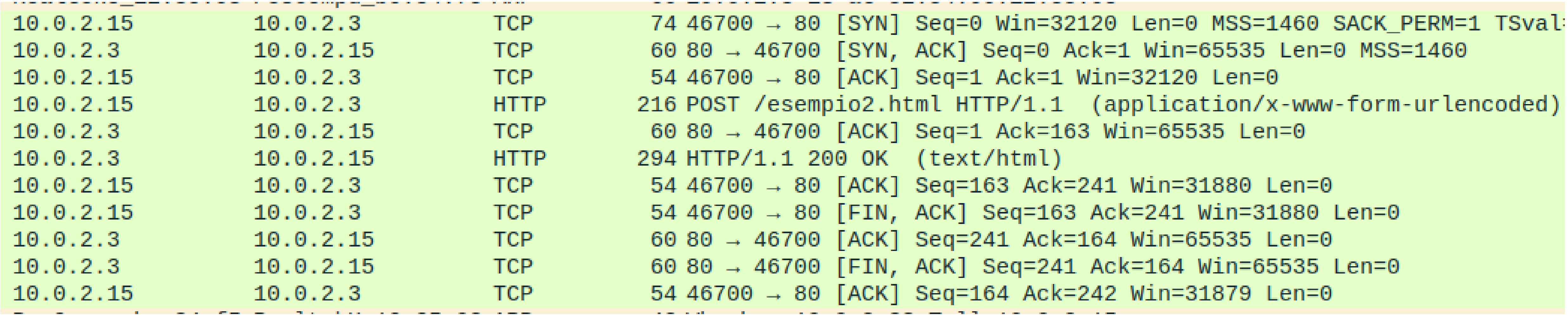
Hub/ switch

LAN Gateway

Internet

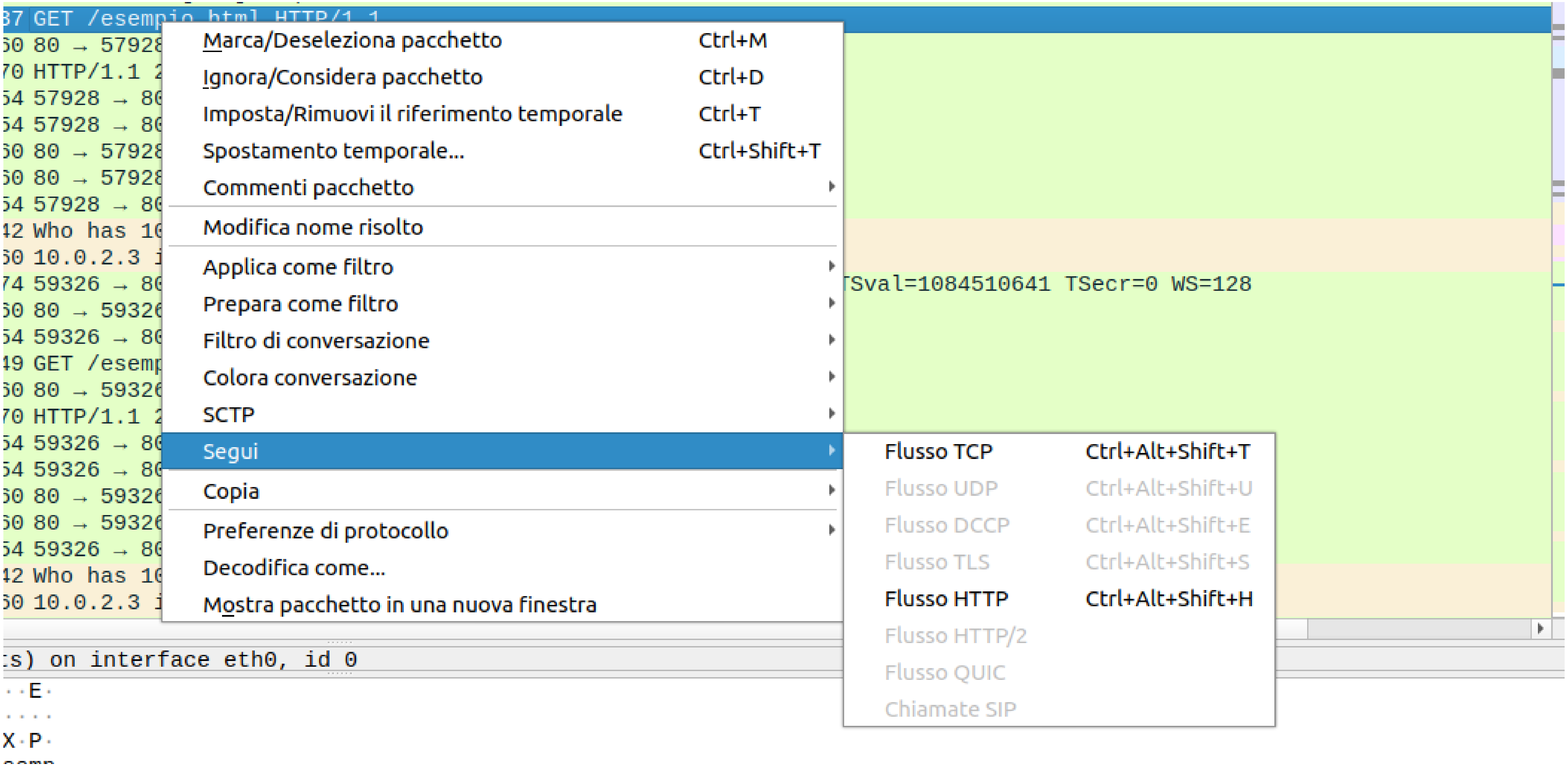
Malicious User

HTTP protocol



```
10.0.2.15        10.0.2.3         TCP          74 46700 → 80 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM=1 TSval
10.0.2.3         10.0.2.15        TCP          60 80 → 46700 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
10.0.2.15        10.0.2.3         TCP          54 46700 → 80 [ACK] Seq=1 Ack=1 Win=32120 Len=0
10.0.2.15        10.0.2.3         HTTP        216 POST /esempio2.html HTTP/1.1  (application/x-www-form-urlencoded)
10.0.2.3         10.0.2.15        TCP          60 80 → 46700 [ACK] Seq=1 Ack=163 Win=65535 Len=0
10.0.2.3         10.0.2.15        HTTP        294 HTTP/1.1 200 OK  (text/html)
10.0.2.15        10.0.2.3         TCP          54 46700 → 80 [ACK] Seq=163 Ack=241 Win=31880 Len=0
10.0.2.15        10.0.2.3         TCP          54 46700 → 80 [FIN, ACK] Seq=163 Ack=241 Win=31880 Len=0
10.0.2.3         10.0.2.15        TCP          60 80 → 46700 [ACK] Seq=241 Ack=164 Win=65535 Len=0
10.0.2.3         10.0.2.15        TCP          60 80 → 46700 [FIN, ACK] Seq=241 Ack=164 Win=65535 Len=0
10.0.2.15        10.0.2.3         TCP          54 46700 → 80 [ACK] Seq=164 Ack=242 Win=31879 Len=0
```

Follow HTTP flow



```
37 GET /esempio.html HTTP/1.1
50 80 → 57928
70 HTTP/1.1 2
54 57928 → 80
54 57928 → 80
50 80 → 57928
50 80 → 57928
54 57928 → 80
42 Who has 10
50 10.0.2.3 i
74 59326 → 80                                                TSval=1084510641 TSecr=0 WS=128
50 80 → 59326
54 59326 → 80
49 GET /esemp
50 80 → 59326
70 HTTP/1.1 2
54 59326 → 80
54 59326 → 80
50 80 → 59326
50 80 → 59326
54 59326 → 80
42 Who has 10
50 10.0.2.3 i
```

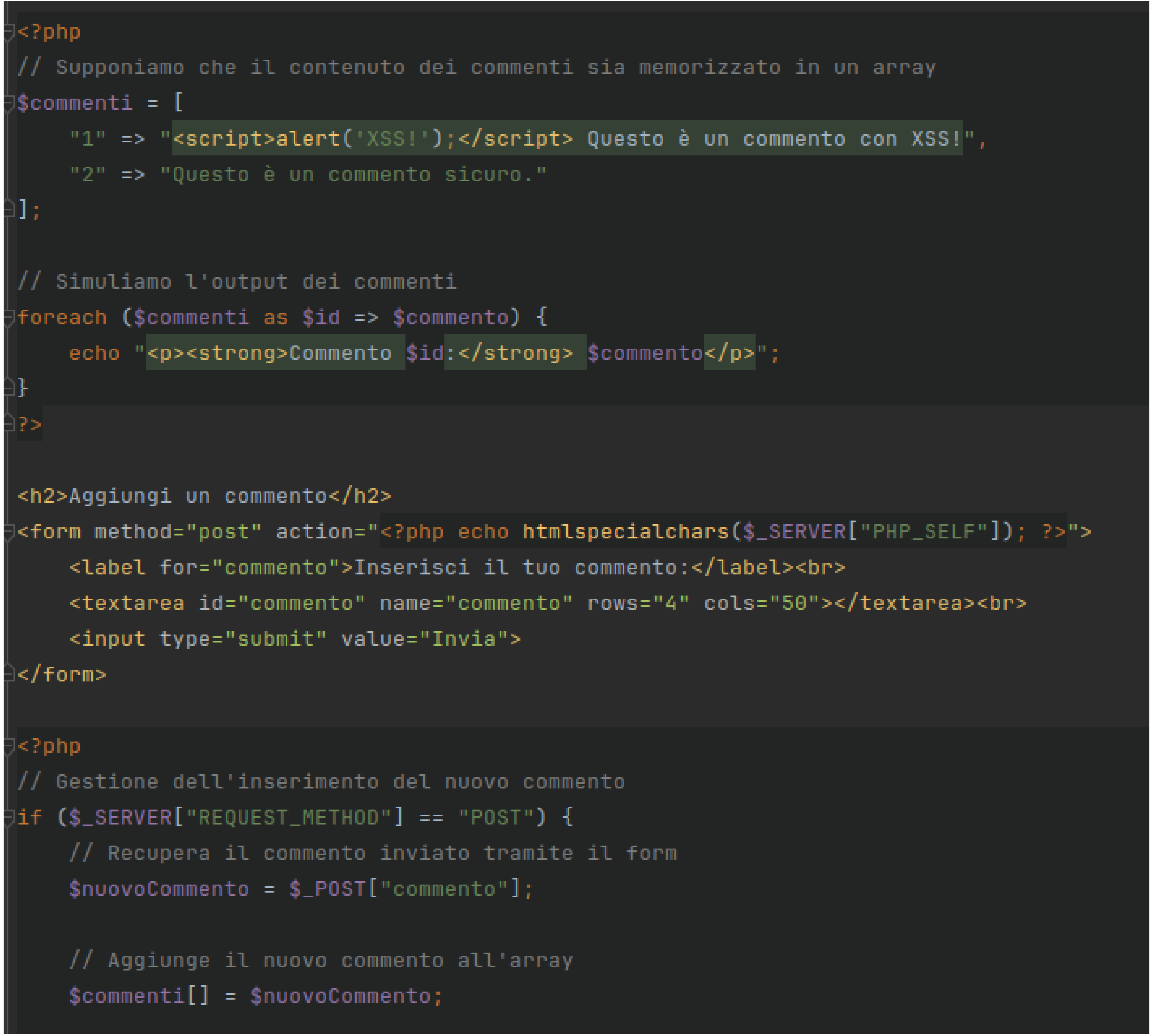
```
s) on interface eth0, id 0
··E·
····
X·P·
```

Marca/Deseleziona pacchetto Ctrl+M
Ignora/Considera pacchetto Ctrl+D
Imposta/Rimuovi il riferimento temporale Ctrl+T
Spostamento temporale... Ctrl+Shift+T
Commenti pacchetto
Modifica nome risolto
Applica come filtro
Prepara come filtro
Filtro di conversazione
Colora conversazione
SCTP
Segui
Copia
Preferenze di protocollo
Decodifica come...
Mostra pacchetto in una nuova finestra
Flusso TCP Ctrl+Alt+Shift+T
Flusso UDP Ctrl+Alt+Shift+U
Flusso DCCP Ctrl+Alt+Shift+E
Flusso TLS Ctrl+Alt+Shift+S
Flusso HTTP Ctrl+Alt+Shift+H
Flusso HTTP/2
Flusso QUIC
Chiamate SIP

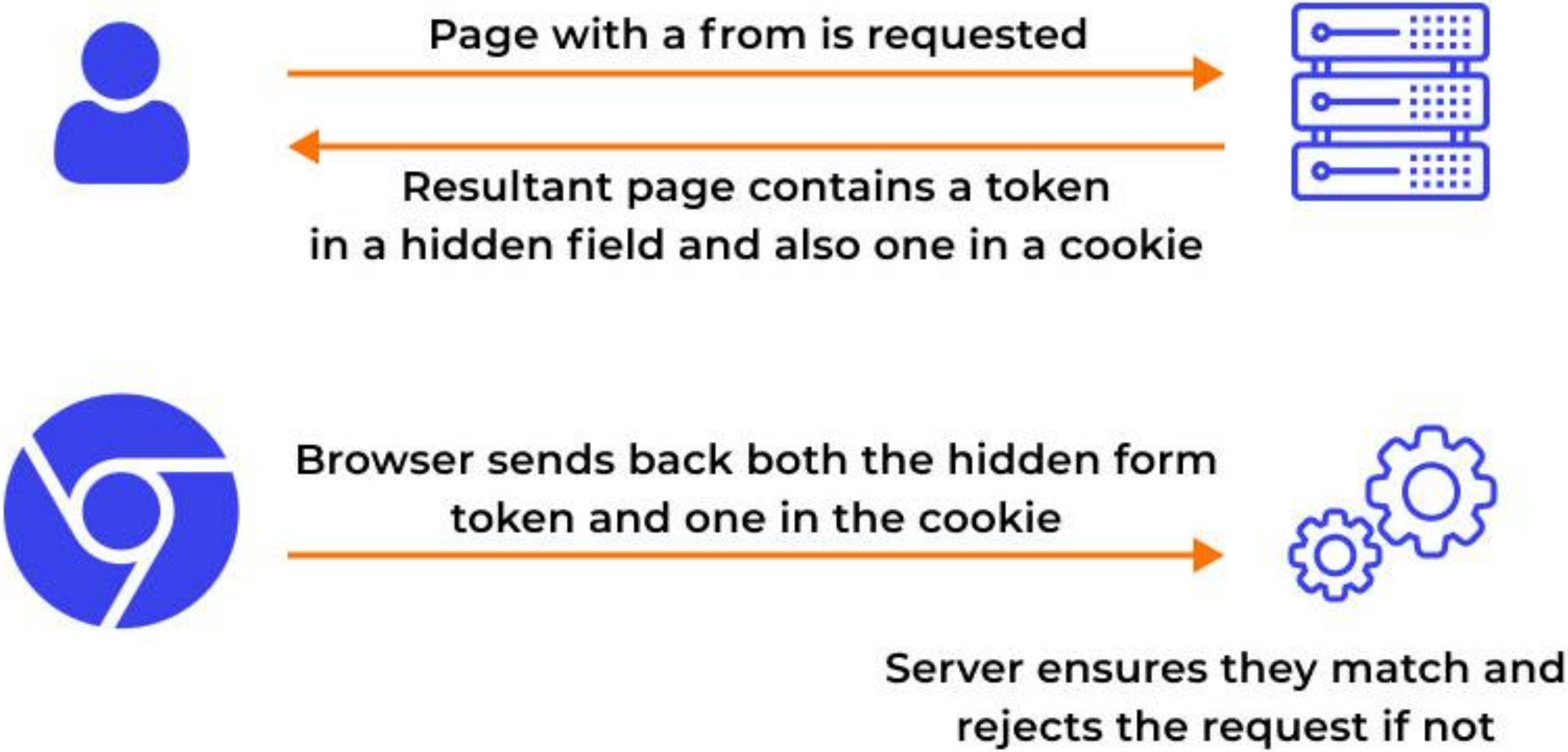
Wireshark – some filters

- tls.handshake.extension.type == "server_name" - check if SNI exists
- ssl.handshake.extension.data contains "example.com"
- http
- http && http.request.method == GET
- http && http.request.method == GET && http.request.full_uri contains unibo

XSS attacks



```
<?php
// Supponiamo che il contenuto dei commenti sia memorizzato in un array
$commenti = [
    "1" => "<script>alert('XSS!');</script> Questo è un commento con XSS!",
    "2" => "Questo è un commento sicuro."
];

// Simuliamo l'output dei commenti
foreach ($commenti as $id => $commento) {
    echo "<p><strong>Commento $id:</strong> $commento</p>";
}
?>

<h2>Aggiungi un commento</h2>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>">
    <label for="commento">Inserisci il tuo commento:</label><br>
    <textarea id="commento" name="commento" rows="4" cols="50"></textarea><br>
    <input type="submit" value="Invia">
</form>

<?php
// Gestione dell'inserimento del nuovo commento
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Recupera il commento inviato tramite il form
    $nuovoCommento = $_POST["commento"];

    // Aggiunge il nuovo commento all'array
    $commenti[] = $nuovoCommento;
```

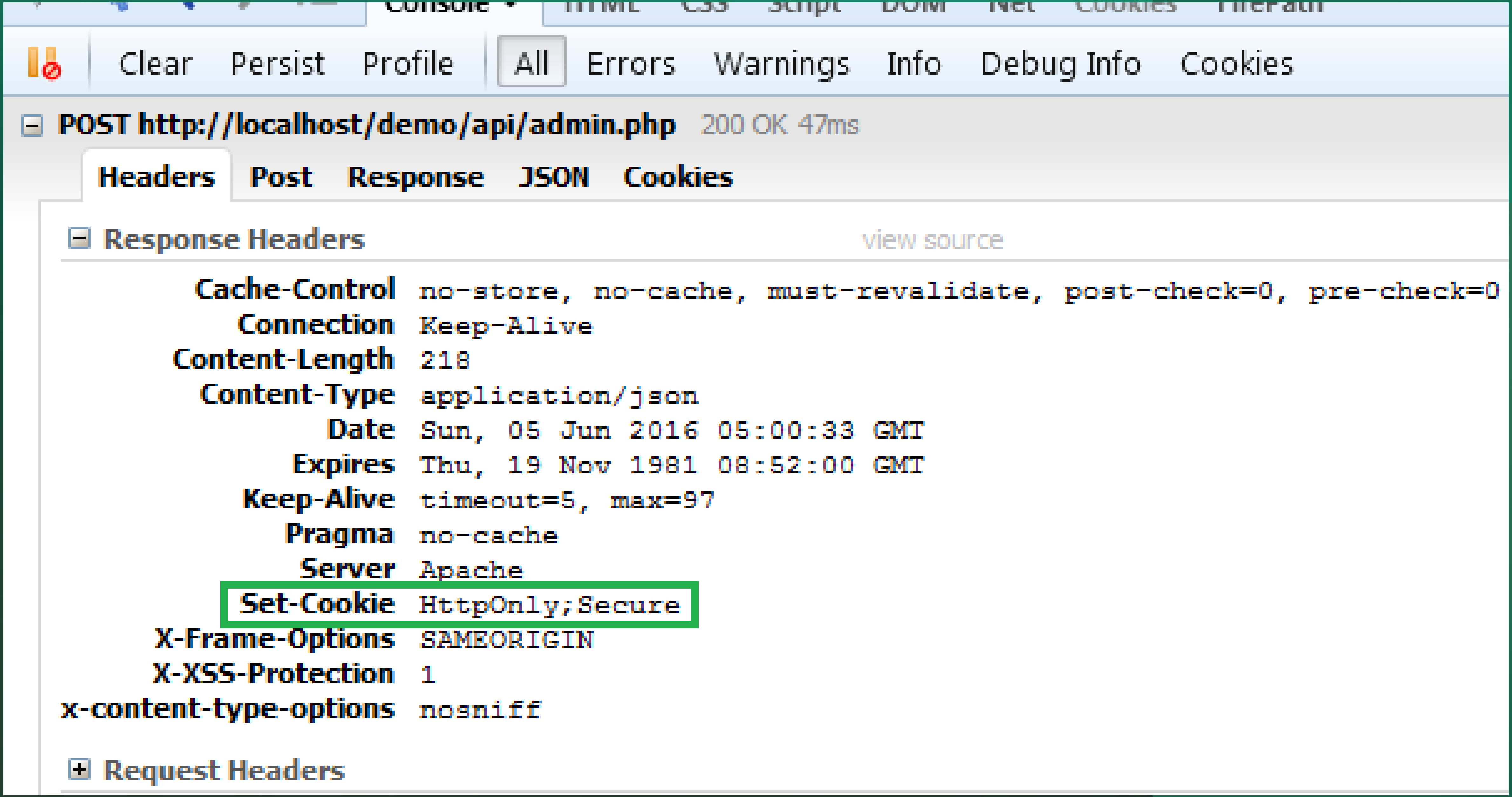


Page with a from is requested

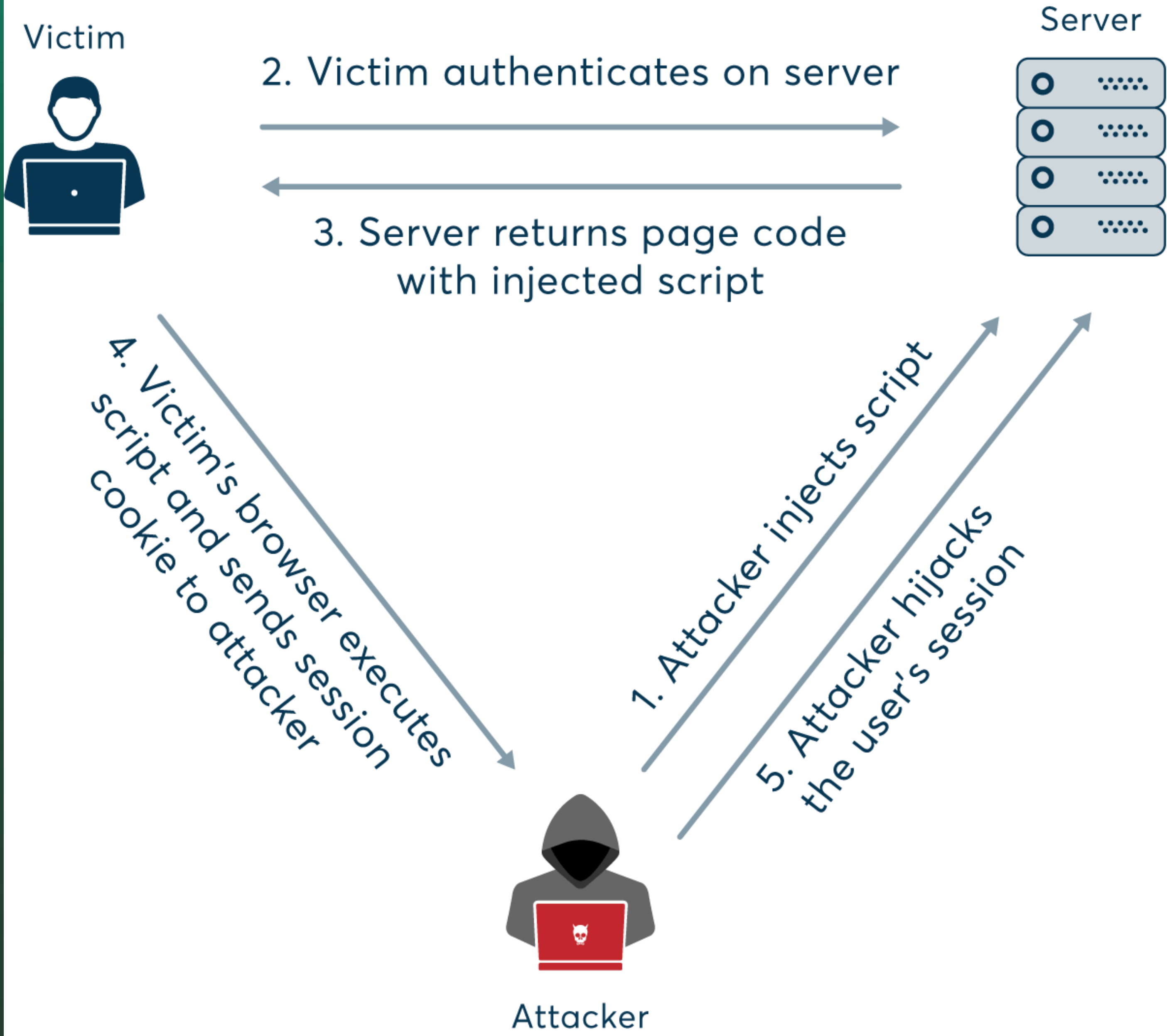
Resultant page contains a token
in a hidden field and also one in a cookie

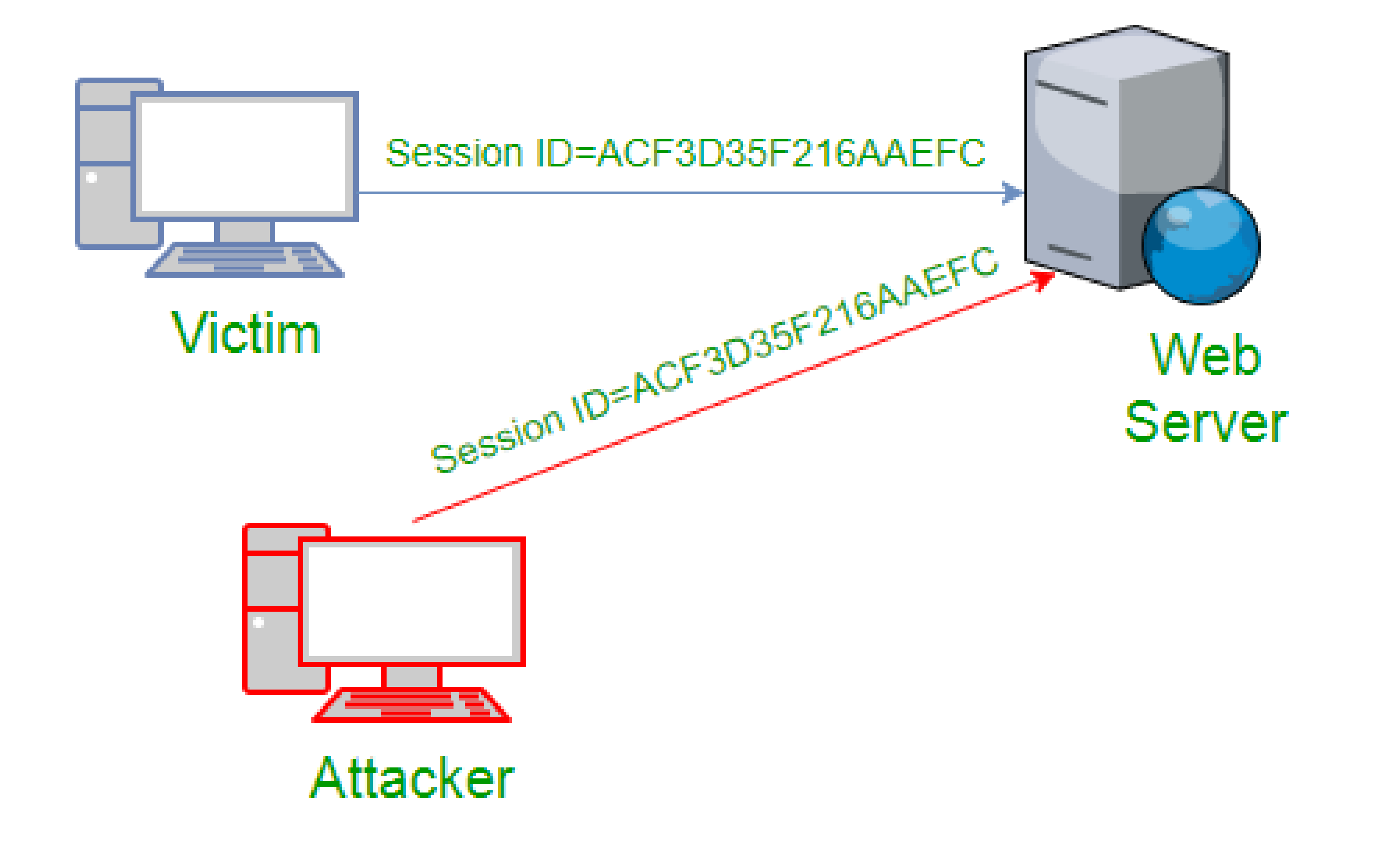
Browser sends back both the hidden form
token and one in the cookie

Server ensures they match and
rejects the request if not

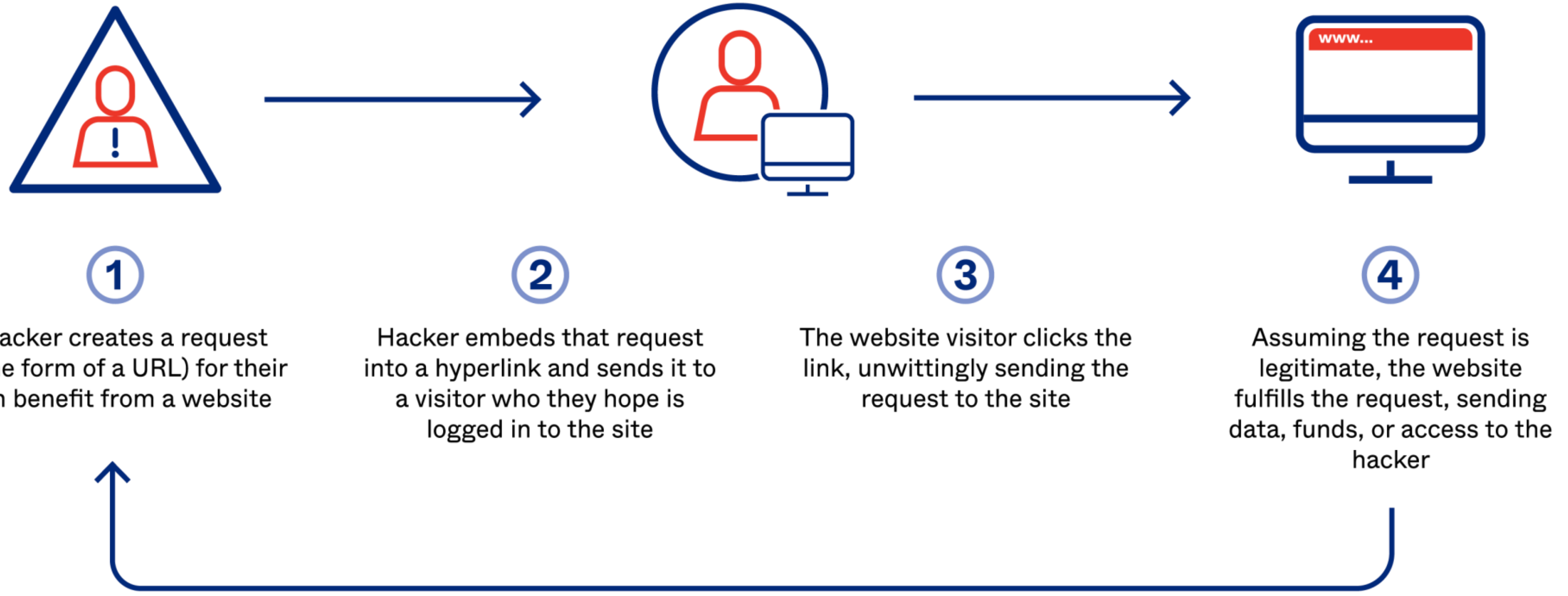


```
⊟ POST http://localhost/demo/api/admin.php   200 OK 47ms

⊟ Response Headers

        Cache-Control  no-store, no-cache, must-revalidate, post-check=0, pre-check=0
           Connection  Keep-Alive
       Content-Length  218
         Content-Type  application/json
                 Date  Sun, 05 Jun 2016 05:00:33 GMT
              Expires  Thu, 19 Nov 1981 08:52:00 GMT
           Keep-Alive  timeout=5, max=97
               Pragma  no-cache
               Server  Apache
           Set-Cookie  HttpOnly;Secure
      X-Frame-Options  SAMEORIGIN
     X-XSS-Protection  1
x-content-type-options  nosniff

⊞ Request Headers
```


Victim
Server
2. Victim authenticates on server
3. Server returns page code with injected script
4. Victim's browser executes script and sends session cookie to attacker
1. Attacker injects script
5. Attacker hijacks the user's session
Attacker


Victim
Session ID=ACF3D35F216AAEFC
Web Server
Session ID=ACF3D35F216AAEFC
Attacker

How Cross Site Request Forgeries (CSRFs) Work

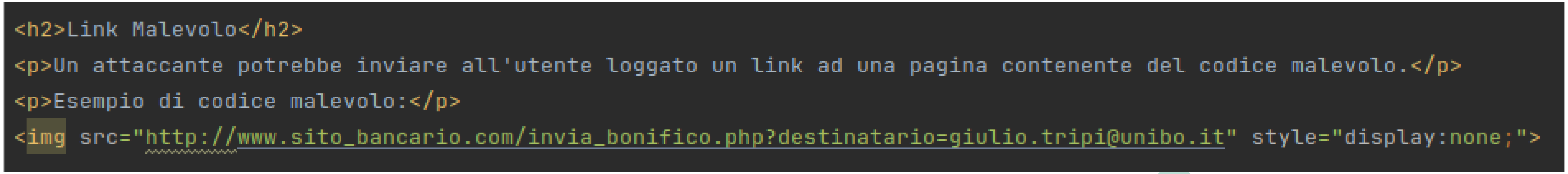


1. A hacker creates a request (in the form of a URL) for their own benefit from a website

2. Hacker embeds that request into a hyperlink and sends it to a visitor who they hope is logged in to the site

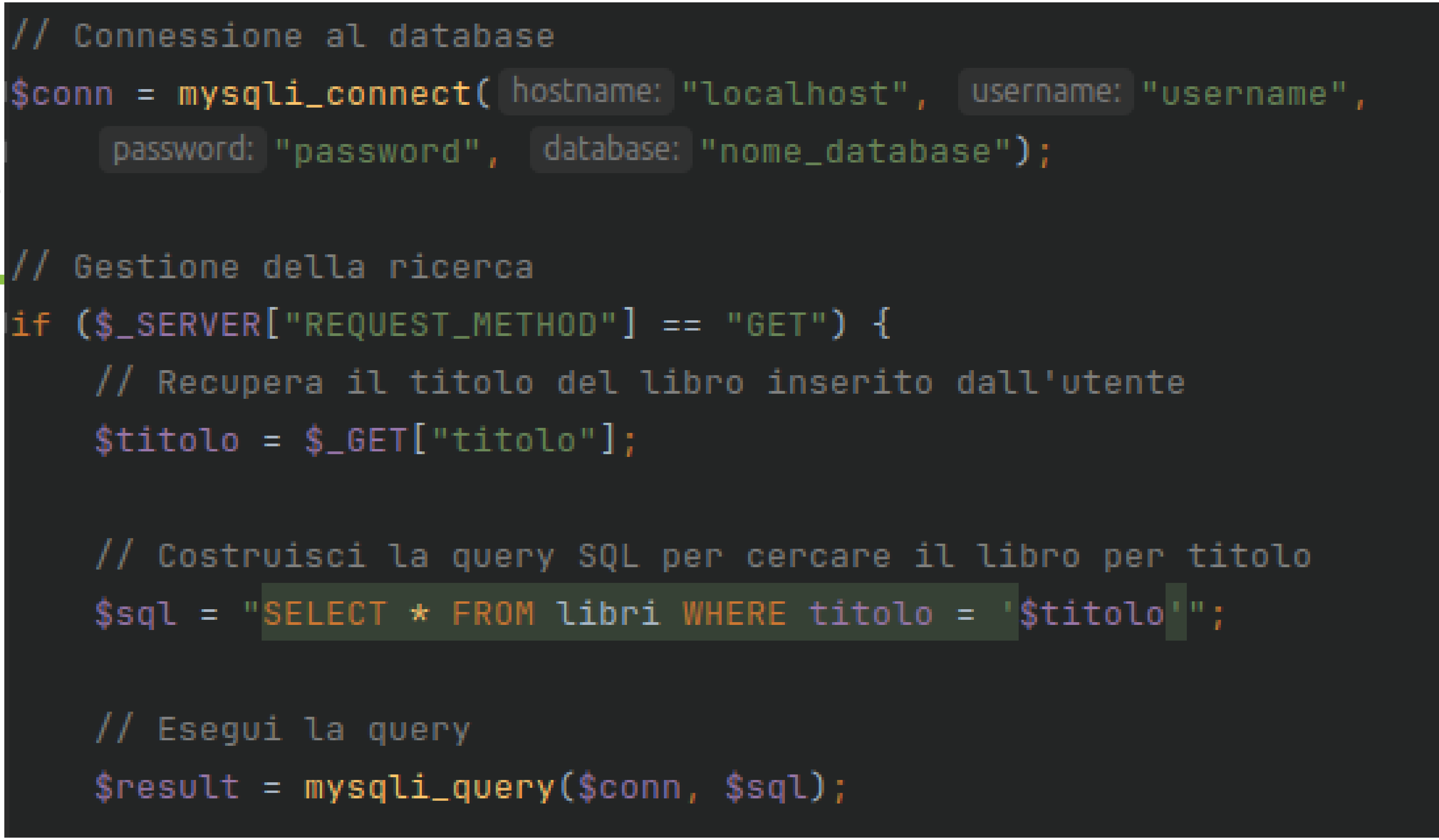
3. The website visitor clicks the link, unwittingly sending the request to the site

4. Assuming the request is legitimate, the website fulfills the request, sending data, funds, or access to the hacker



```html
<h2>Link Malevolo</h2>
<p>Un attaccante potrebbe inviare all'utente loggato un link ad una pagina contenente del codice malevolo.</p>
<p>Esempio di codice malevolo:</p>
<img src="http://www.sito_bancario.com/invia_bonifico.php?destinatario=giulio.tripi@unibo.it" style="display:none;">
```

SQL injections



```
// Connessione al database
$conn = mysqli_connect( hostname: "localhost",  username: "username",
    password: "password",  database: "nome_database");

// Gestione della ricerca
if ($_SERVER["REQUEST_METHOD"] == "GET") {
    // Recupera il titolo del libro inserito dall'utente
    $titolo = $_GET["titolo"];

    // Costruisci la query SQL per cercare il libro per titolo
    $sql = "SELECT * FROM libri WHERE titolo = '$titolo'";

    // Esegui la query
    $result = mysqli_query($conn, $sql);
```

SQL injections

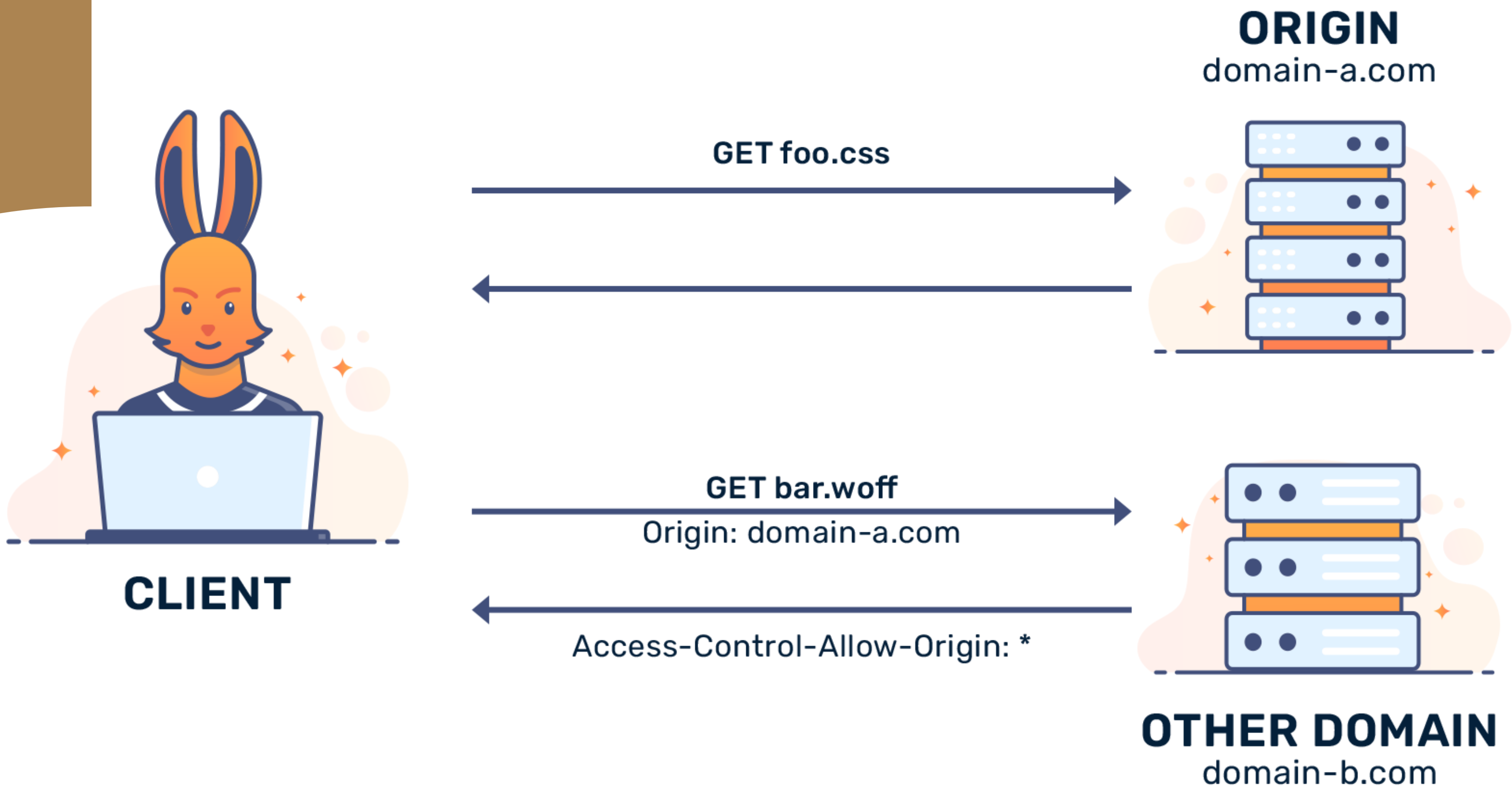
Se

```
$titolo = "' OR '1' = '1'";
```

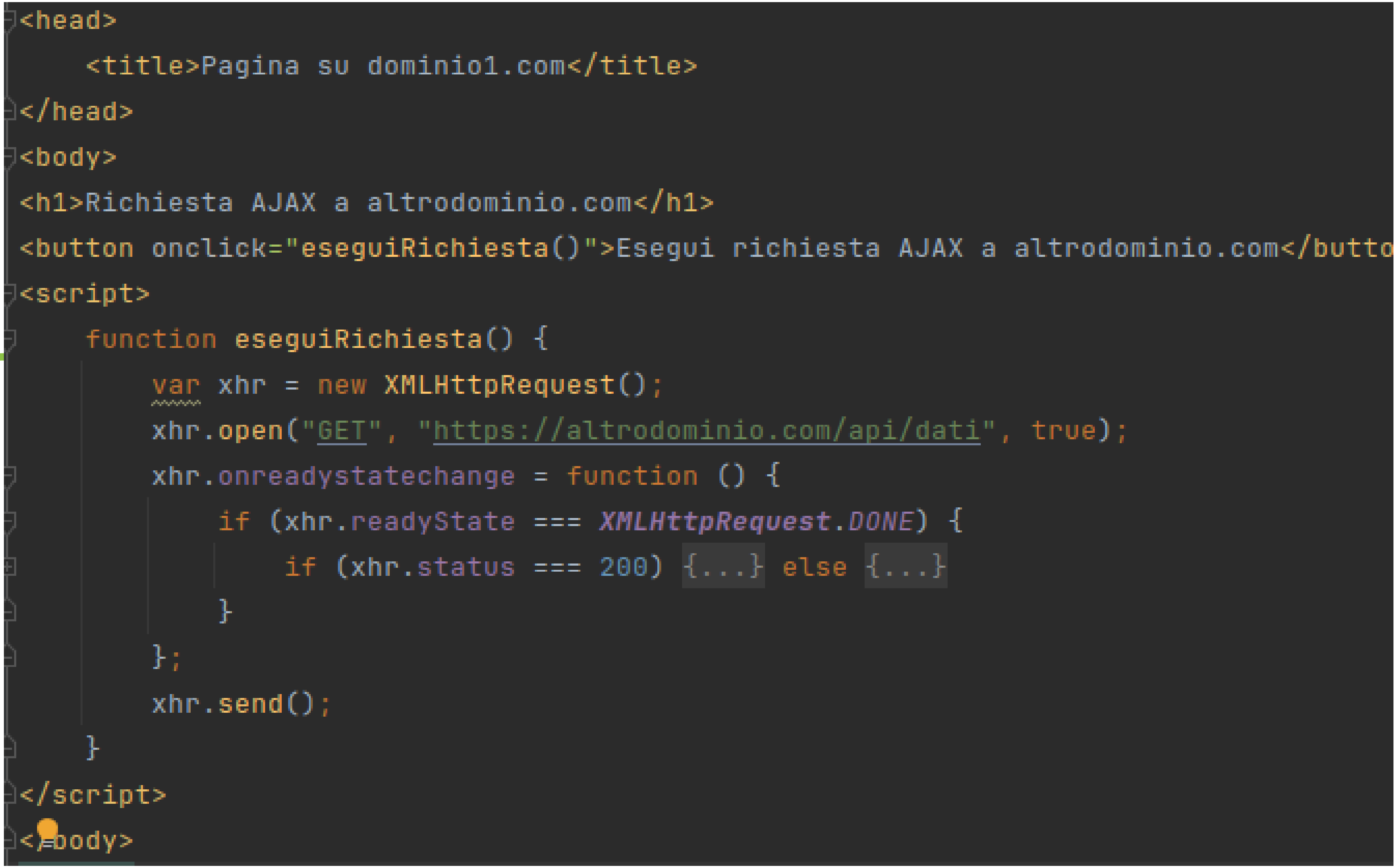
La query diventa:

```
SELECT * FROM libri WHERE titolo = '' OR '1'='1'
```

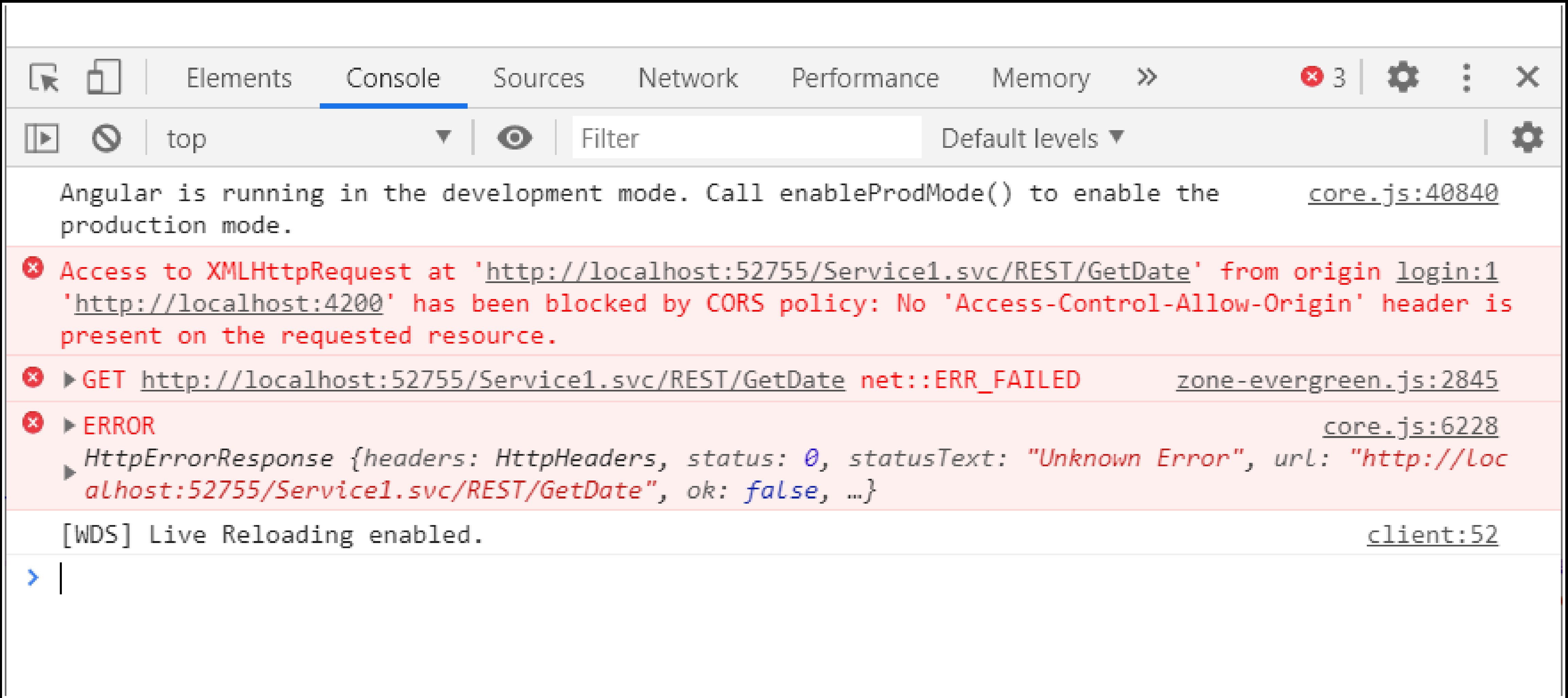
CORS


ORIGIN
domain-a.com
GET foo.css
CLIENT
GET bar.woff
Origin: domain-a.com
Access-Control-Allow-Origin: *
OTHER DOMAIN
domain-b.com

CORS



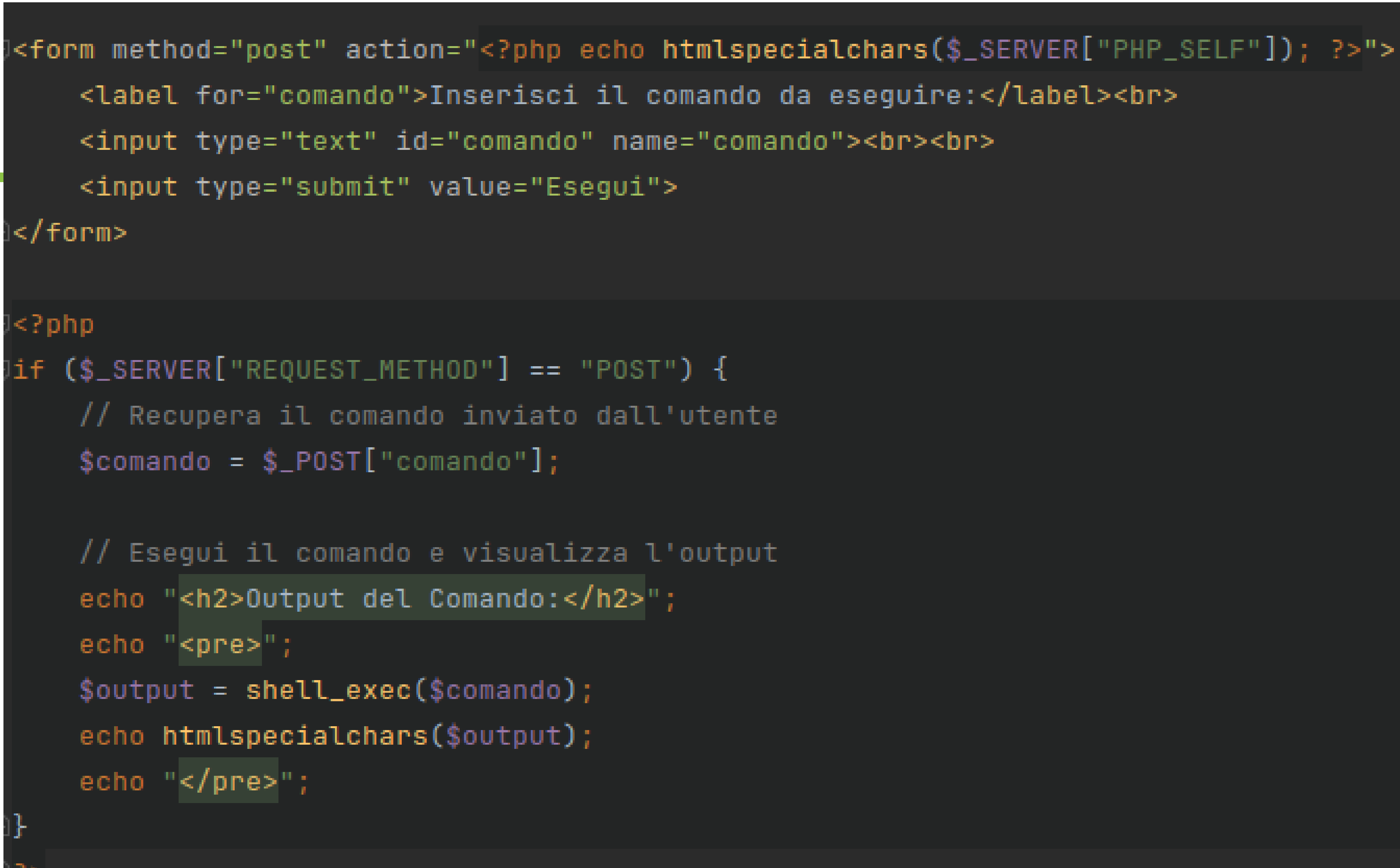
```
<head>
    <title>Pagina su dominio1.com</title>
</head>
<body>
<h1>Richiesta AJAX a altrodominio.com</h1>
<button onclick="eseguiRichiesta()">Esegui richiesta AJAX a altrodominio.com</butto
<script>
    function eseguiRichiesta() {
        var xhr = new XMLHttpRequest();
        xhr.open("GET", "https://altrodominio.com/api/dati", true);
        xhr.onreadystatechange = function () {
            if (xhr.readyState === XMLHttpRequest.DONE) {
                if (xhr.status === 200) {...} else {...}
            }
        };
        xhr.send();
    }
</script>
</body>
```



```
Angular is running in the development mode. Call enableProdMode() to enable the      core.js:40840
production mode.
Access to XMLHttpRequest at 'http://localhost:52755/Service1.svc/REST/GetDate' from origin login:1
'http://localhost:4200' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is
present on the requested resource.
▶GET http://localhost:52755/Service1.svc/REST/GetDate net::ERR_FAILED       zone-evergreen.js:2845
▶ERROR                                                                                 core.js:6228
▶HttpErrorResponse {headers: HttpHeaders, status: 0, statusText: "Unknown Error", url: "http://loc
alhost:52755/Service1.svc/REST/GetDate", ok: false, …}
[WDS] Live Reloading enabled.                                                            client:52
>
```

CORS blocked by browser

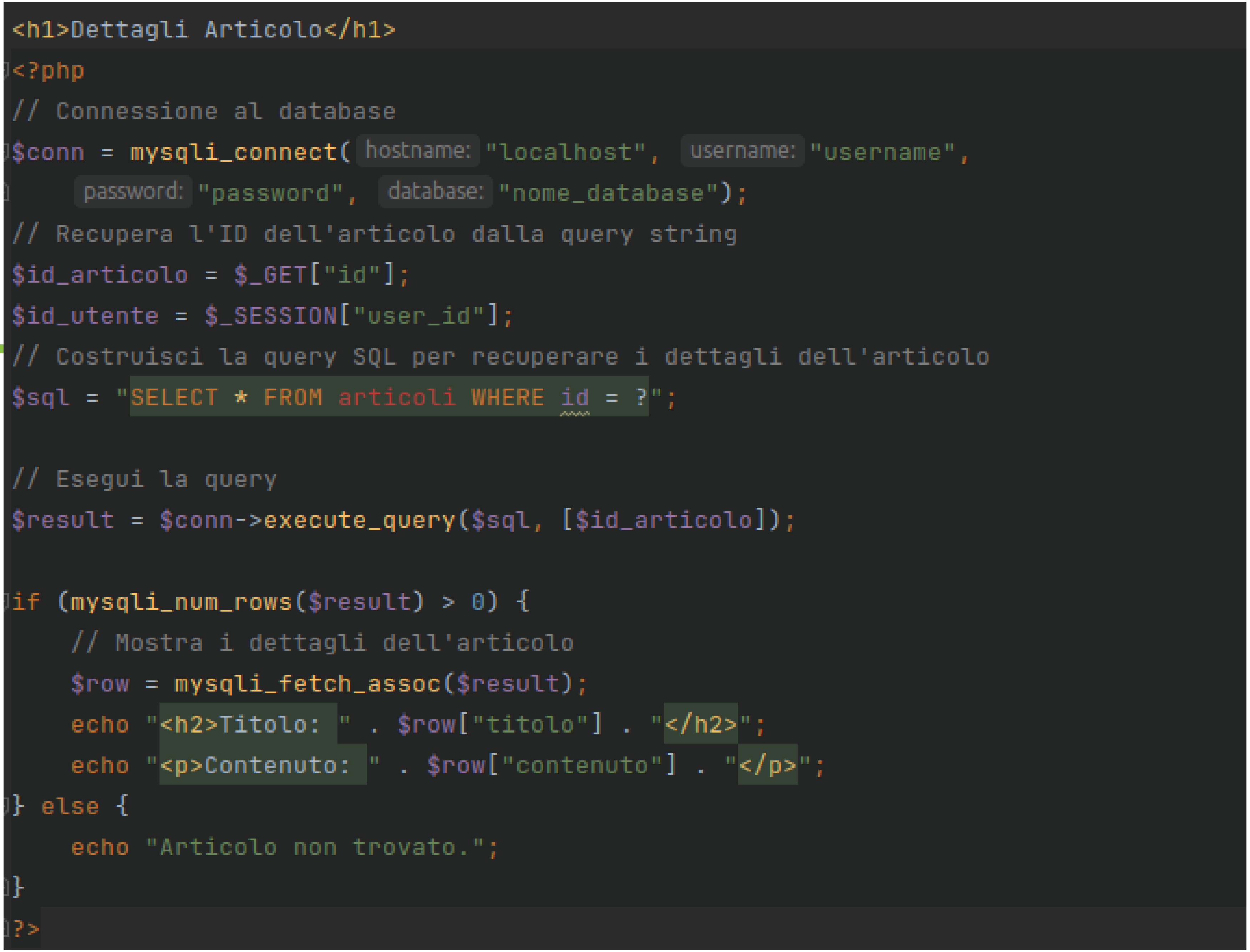
Command injection



```
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>">
    <label for="comando">Inserisci il comando da eseguire:</label><br>
    <input type="text" id="comando" name="comando"><br><br>
    <input type="submit" value="Esegui">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Recupera il comando inviato dall'utente
    $comando = $_POST["comando"];

    // Esegui il comando e visualizza l'output
    echo "<h2>Output del Comando:</h2>";
    echo "<pre>";
    $output = shell_exec($comando);
    echo htmlspecialchars($output);
    echo "</pre>";
}
```

Information disclosure

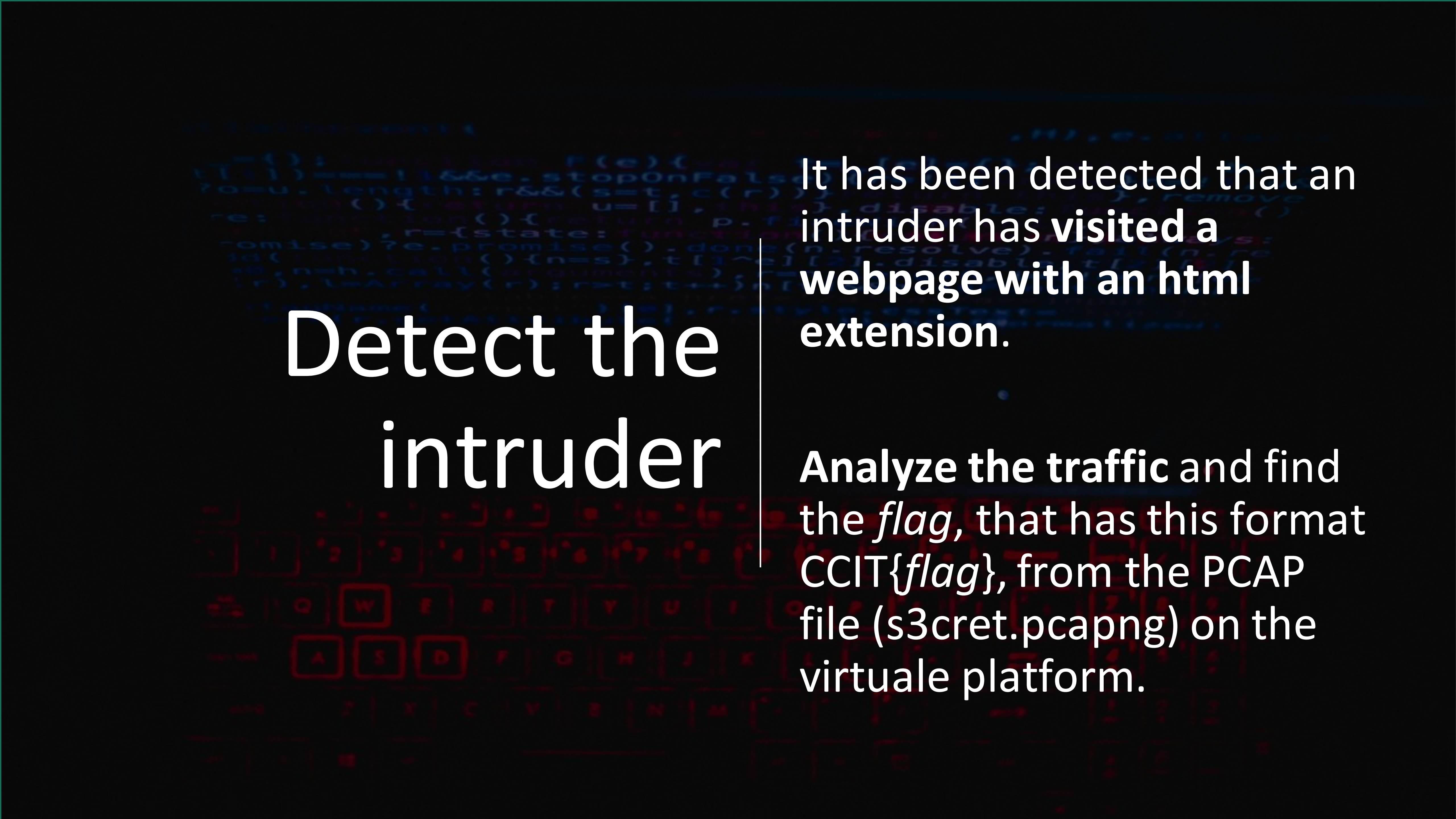
dettagli_articolo.php?id=123



```
<h1>Dettagli Articolo</h1>
<?php
// Connessione al database
$conn = mysqli_connect( hostname: "localhost",  username: "username",
    password: "password",  database: "nome_database");
// Recupera l'ID dell'articolo dalla query string
$id_articolo = $_GET["id"];
$id_utente = $_SESSION["user_id"];
// Costruisci la query SQL per recuperare i dettagli dell'articolo
$sql = "SELECT * FROM articoli WHERE id = ?";

// Esegui la query
$result = $conn->execute_query($sql, [$id_articolo]);

if (mysqli_num_rows($result) > 0) {
    // Mostra i dettagli dell'articolo
    $row = mysqli_fetch_assoc($result);
    echo "<h2>Titolo: " . $row["titolo"] . "</h2>";
    echo "<p>Contenuto: " . $row["contenuto"] . "</p>";
} else {
    echo "Articolo non trovato.";
}
?>
```

And more...

- Server-Side Request Forgery (SSRF)
- Path traversal
- Brute force attack (especially on login)



Detect the intruder

It has been detected that an intruder has **visited a webpage with an html extension**.

Analyze the traffic and find the *flag*, that has this format CCIT{*flag*}, from the PCAP file (s3cret.pcapng) on the virtuale platform.

A/D

Create some web pages that suffer some of the previously explained attacks (XSS, CSRF, SQLi, command injection and information disclosure).

Explain how to exploit them and how to fix them.