

# Progetto Reti di calcolatori — Packet Sniffer

---

*di Aldo Fumagalli e Francesco Torregrossa, A.A. 19/20*

## Progetto Reti di calcolatori — Packet Sniffer

- Introduzione

- Sviluppo del programma

  - Protocolli e strutture dati

    - Ethernet

    - IPv4

    - ICMP

    - TCP

    - UDP

  - Ricezione dei pacchetti

  - Analisi dei pacchetti

- Preparazione dell'OrangePi

  - Verifica dei driver della scheda di rete

  - Trovare il nome della scheda di rete

    - Verifica avvenuta creazione interfaccia wireless

  - Attivazione dell'interfaccia

    - Creazione file di configurazione

    - Avvio wpa\_supplicant

    - Avvio del prompt interattivo del tool wpa

  - Richiesta indirizzo IP

  - Risoluzione dei problemi

  - Configurazione del server

    - Installazione e configurazione dei software necessari

- Sviluppo del sito web

  - Database

  - Login, Sessione, gestione database e logout

  - Registrazione e pagina utente

  - Home, ricerca e scheda brano

  - Aggiunzione dei brani

- Prova dello sniffer

# Introduzione

---

Abbiamo realizzato un programma in C che ascolta e analizza tutti i pacchetti ricevuti dal computer, permettendo di mostrare dettagli come la sorgente, il destinatario, i protocolli utilizzati, e anche i contenuti che essi trasportano.

Successivamente, abbiamo caricato il programma su un dispositivo [OrangePi](#) munito di una scheda di rete wireless TP-Link [TL-WN722N](#).

Abbiamo anche preparato un sito web fittizio in PHP, HTML e CSS (Client/Server) che simula una piattaforma di audio streaming con le funzionalità di accesso, registrazione e uso generale.

Così, accedendo e utilizzando il sito web tramite client, abbiamo potuto simulare l'invio di alcuni pacchetti che sono poi stati analizzati dal nostro programma, attivo sull'OrangePi. Questo ci ha permesso di accedere ai dati sensibili e di verificare la correttezza delle informazioni ottenute dal programma stesso.

# Sviluppo del programma

## Protocolli e strutture dati

Ogni pacchetto di rete è stratificato, e ogni strato segue i suoi protocolli, ciascuno con delle proprietà e dei dati diversi. Perciò, partendo dai byte grezzi, abbiamo predisposto delle `struct` adatte a contenere gli header di ciascun protocollo e delle funzioni per analizzarli e visualizzarli. Il tutto è stato raccolto all'interno di vari file `.h` e `.c`, una coppia per protocollo.

Per prima cosa, però, riportiamo i seguenti alias, che abbiamo definito nel file `datatypes.h` per fare chiarezza sui dati

```
1 typedef unsigned char byte;
2 typedef unsigned short word;
3 typedef unsigned int dword;
4 typedef byte *packet;
5 typedef char *string;
```

Abbiamo anche predisposto le seguenti funzioni per facilitare il passaggio tra codifiche `LITTLE_ENDIAN` e `BIG_ENDIAN`

```
1 word switch_encoding_w(word w);
2 dword switch_encoding_dw(dword dw);
```

In questa relazione analizzeremo soltanto i protocolli Ethernet e IP perché le considerazioni fatte su di essi possono essere facilmente riportate per tutti gli altri protocolli presi in esame.

## Ethernet

Il protocollo Ethernet prevede un header di lunghezza fissa di 14 byte, di cui 6 sono dedicati all'indirizzo MAC del destinatario e 6 a quello della sorgente, e gli ultimi 2 indicano il protocollo utilizzato.

```
1  0 1 2 3 4 5 6 7 | 0 1 2 3 4 5 6 7 | 0 1 2 3 4 5 6 7 | 0 1 2 3 4 5 6 7
2  +-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+
3  |               destination MAC (first 4 bytes)               |
4  +-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+
5  | destination MAC (last 2 bytes) | source MAC (first 2 bytes) |
6  +-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+
7  |               source MAC (last 2 bytes)               |
8  +-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+
9  |               type code               |
10 +-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+-+
```

Abbiamo predisposto una struttura con 6 byte, nominati da `a` ad `f`, per rappresentare un singolo indirizzo MAC. Questa struttura è stata utilizzata nella realizzazione di quella principale, `eth_header`.

```
1 #define ETH_HEADER_SIZE 14
2
3 typedef struct
4 {
5     byte a;
```

```

6   byte b;
7   byte c;
8   byte d;
9   byte e;
10  byte f;
11 } mac_address;
12
13 struct eth_header
14 {
15     mac_address destination_host;
16     mac_address source_host;
17     word type_code;
18
19     packet next; // puntatore al pacchetto incapsulato
20 };
21
22 typedef struct eth_header *eth_header;

```

La struttura contiene anche un puntatore `packet next` che serve per raggiungere facilmente il pacchetto incapsulato. Abbiamo definito la seguente funzione per assegnare il valore corretto al puntatore.

```

1  eth_header prepare_eth_header(packet data);
2  {
3      eth_header header = malloc(sizeof(struct eth_header));
4      memcpy(header, data, ETH_HEADER_SIZE);
5      header->next = data + ETH_HEADER_SIZE;
6      return header;
7  }

```

## IPv4

```

1  0 1 2 3 4 5 6 7 | 0 1 2 3 4 5 6 7 | 0 1 2 3 4 5 6 7 | 0 1 2 3 4 5 6 7
2  +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+
3  |  ihl  |  vers  | type of service |          total length          |
4  +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+
5  |          identification          | 0|d|m|    fragment offset    |
6  +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+
7  |  time to live  |   protocol   |          checksum          |
8  +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+
9  |          source IP address          |
10 +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+
11 |          destination IP address          |
12 +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+
13 |          options (variable length)          |
14 +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+ +-+-+-+-+-+-+-+

```

Anche per i pacchetti IPv4 abbiamo seguito la stessa logica, ovvero dopo aver analizzato l'header del pacchetto abbiamo creato una struttura adeguata ad ospitare i dati previsti. In particolare, è stato molto utile rappresentare alcuni dati utilizzando i bit field.

```

1  #define IP_HEADER_SIZE 20
2
3  typedef struct
4  {
5      byte a;
6      byte b;

```

```

7   byte c;
8   byte d;
9 } ip_address;
10
11 struct ip_header
12 {
13     byte
14         header_length : 4,
15         version : 4;
16     byte type_of_service;
17     word total_length;
18
19     word id;
20     word : 1,
21         flag_do_not_fragment : 1,
22         flag_more_fragments : 1,
23         fragment_offset : 13;
24
25     byte time_to_live;
26     byte protocol;
27     word checksum;
28
29     ip_address source_address;
30     ip_address destination_address;
31
32     void *options;
33
34     packet next;
35 };
36
37 typedef struct ip_header *ip_header;

```

Come in precedenza, abbiamo definito la funzione `prepare_ip_header` per stabilire il valore del puntatore al pacchetto incapsulato `next`.

Tuttavia, questo header non ha lunghezza fissa, perché potrebbe contenere delle `options`. Per sapere se le contiene possiamo leggere il parametro `header_length`, che ha un valore minimo di 5 (`options` non presente) e un valore massimo di 15, e va interpretato come numero di *gruppi di 4 byte*. Se vale 5, la lunghezza dell'header IP è di 5 gruppi di 4 byte, cioè 20 byte.

Inoltre, in questo header sono presenti dei dati di tipo `word` che devono essere convertiti prima dell'utilizzo.

```

1  ip_header prepare_ip_header(packet data)
2  {
3      ip_header header = malloc(sizeof(struct ip_header));
4      memcpy(header, data, IP_HEADER_SIZE);
5
6      unsigned int options_length = size_ip_header(header) - IP_HEADER_SIZE;
7      if (options_length)
8      {
9          header->options = malloc(options_length);
10         memcpy(header->options, data + IP_HEADER_SIZE, options_length);
11     }
12     else
13         header->options = NULL;
14
15     header->next = data + IP_HEADER_SIZE + options_length;
16

```

```

17     header->total_length = switch_encoding_w(header->total_length);
18     header->id = switch_encoding_w(header->id);
19     header->checksum = switch_encoding_w(header->checksum);
20
21     return header;
22 }
23
24 dword size_ip_header(ip_header header) {
25     return header->header_length * 4;
26 }

```

## ICMP

```

1  0 1 2 3 4 5 6 7 | 0 1 2 3 4 5 6 7 | 0 1 2 3 4 5 6 7 | 0 1 2 3 4 5 6 7
2  +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+
3  |      type      |      code      |      checksum      |
4  +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+
5  |                                     options                                     |
6  +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+

```

## TCP

```

1  0 1 2 3 4 5 6 7 | 0 1 2 3 4 5 6 7 | 0 1 2 3 4 5 6 7 | 0 1 2 3 4 5 6 7
2  +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+
3  |      source port      |      destination port      |
4  +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+
5  |      sequence number      |
6  +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+
7  |      acknowledgment number      |
8  +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+
9  | data |      | c|e|u|a|p|r|s|f |
10 |offset |0 0 0 0 | w|c|r|c|s|s|y|i |      window      |
11 |      |      | r|e|g|k|h|t|n|n |
12 +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+
13 |      checksum      |      urgent pointer      |
14 +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+
15 |      options      ... | ...      padding      |
16 +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+

```

## UDP

```

1  0 1 2 3 4 5 6 7 | 0 1 2 3 4 5 6 7 | 0 1 2 3 4 5 6 7 | 0 1 2 3 4 5 6 7
2  +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+
3  |      source port      |      destination port      |
4  +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+
5  |      length      |      checksum      |
6  +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+ + +-+-+-+-+-+-+-+

```

## Ricezione dei pacchetti

Mediante le librerie `sys/socket.h` e `netinet/if_ether.h`, abbiamo richiesto al sistema la creazione di una socket in grado di ricevere qualsiasi pacchetto, la cui lettura viene realizzata tramite il metodo `read` della libreria `unistd.h`.

Nel dettaglio, la socket viene creata con questi parametri

```
1 | int socket(int family, int type, int protocol)
```

- `family` viene impostato a `AF_PACKET`, così la comunicazione può avvenire al livello di collegamento, quindi si riceveranno i frame Ethernet
- `type` viene impostato a `SOCK_RAW`, per accettare pacchetti con qualsiasi tipo di socket
- `protocol` viene impostato a `ETH_P_ALL`, per indicare che pacchetti con qualsiasi protocollo della `family` sono accettati (anche se successivamente analizzeremo solo quelli contenenti IPv4)

Pertanto, il codice che esegue la lettura è il seguente, in cui la funzione `ntohs` esegue il passaggio da `LITTLE_ENDIAN` a `BIG_ENDIAN` e viceversa (se necessario)

```
1 | int sock = socket(AF_PACKET, SOCK_RAW, ntohs(ETH_P_ALL));
2 | if (sock < 0)
3 |     perror("Socket creation error");
4 |
5 | while (read(sock, buffer, PKT_LEN) > 0)
6 |     analyze(buffer);
```

## Analisi dei pacchetti

Ogni volta che arriva un pacchetto qualsiasi, esso viene analizzato e scomposto utilizzando le funzioni e le strutture relative ai vari protocolli.

Di seguito riportiamo lo scheletro della funzione `analyze`, che mostra la logica con cui i pacchetti vengono estratti a vari livelli. La funzione vera, nel file `sniffer.c`, gestisce, tra le altre cose, anche i filtri sull'output a video.

```
1 | void analyze(packet buffer)
2 | {
3 |     eth_header eh = prepare_eth_header(buffer);
4 |     describe_eth_header(eh);
5 |
6 |     if (eh->type_code == ntohs(ETH_P_IP))
7 |     {
8 |         ip_header iph = prepare_ip_header(eh->next);
9 |         describe_ip_header(iph);
10 |
11 |         switch (iph->protocol)
12 |         {
13 |             case IPPROTO_ICMP:
14 |             {
15 |                 icmp_header icmph = prepare_icmp_header(iph->next);
16 |                 describe_icmp_header(icmph);
17 |                 print_plaintext(icmph->next, iph->total_length - size_ip_header(iph) -
size_icmp_header(icmph));
18 |                 break;
19 |             }
20 |             case IPPROTO_TCP:
21 |             {
22 |                 tcp_header tcph = prepare_tcp_header(iph->next);
23 |                 describe_tcp_header(tcph);
24 |                 print_plaintext(tcph->next, iph->total_length - size_ip_header(iph) - size_tcp_header(tcph));
25 |                 break;
26 |             }
27 |             case IPPROTO_UDP:
```

```
28     {
29         udp_header udph = prepare_udp_header(iph->next);
30         describe_udp_header(udph);
31         print_plaintext(udph->next, iph->total_length - size_ip_header(iph) - size_udp_header(udph));
32         break;
33     }
34     default:
35         print_plaintext(iph->next, iph->total_length - size_ip_header(iph));
36         break;
37     }
38 }
39 }
```



# Preparazione dell'OrangePi

L'OrangePi è una scheda **Open-Source** basata su architettura **ARM**, su questa scheda possono essere eseguite distribuzioni linux e quella che abbiamo utilizzato noi è **Armbian**, una distribuzione di linux creata per dispositivi che funzionano con architetture **ARM**.

Ciò che segue è la configurazione della scheda di rete wireless e quindi delle impostazioni di rete dell'OrangePi. Essa è suddivisa in più fasi procedurali, dove la non riuscita di una qualsiasi di esse causa il non poter procedere alla fase successiva, e quindi in queste situazioni bisogna intervenire con strumenti di risoluzione per poter, poi, riprendere con la normale procedura.

## Verifica dei driver della scheda di rete

Prima di tutto bisogna verificare che i driver installati nel nostro dispositivo siano compatibili con la nostra scheda di rete. Questo si può fare con il seguente comando:

```
1 $ lspci -k
2 06:00.0 Network controller: Intel Corporation WiFi Link 5100
3   Subsystem: Intel Corporation WiFi Link 5100 AGN
4   Kernel driver in use: iwlwifi
5   Kernel modules: iwlwifi
```

In questo caso i driver sono correttamente installati, ma in caso contrario bisognerebbe ricorrere a comandi risolutivi per l'installazione dei driver corretti, se esistenti.

## Trovare il nome della scheda di rete

Dopo la verifica dei driver bisogna immediatamente identificare la nostra scheda di rete, e successivamente controllare che l'interfaccia wireless si sia creata correttamente autonomamente, in maniera tale da poter incominciare la nostra vera e propria configurazione.

```
1 $ iwconfig
2 wlan0  unassociated  Nickname:"<WIFI@REALTEK>"
3          Mode:Auto   Frequency=2.412 GHz  Access Point: Not-Associated
4          Sensitivity:0/0
5          Retry:off   RTS thr:off   Fragment thr:off
6          Power Management:off
7          Link Quality:0  Signal level:0  Noise level:0
8          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
9          Tx excessive retries:0  Invalid misc:0  Missed beacon:0
```

*wlan0* è il nome della nostra scheda di rete

## Verifica avvenuta creazione interfaccia wireless

```
1 $ ip link set dev wlan0 up
```

Non restituendo nessun errore, il terminale conferma l'avvenuta creazione.

## Attivazione dell'interfaccia

Il primo passo è quello di abilitare l'interfaccia tramite il comando *iw* o alternativamente *ifconfig*. Questi due sono rispettivamente comandi per la gestione delle interfacce wireless (*iw*) e per le interfacce in generale (*ifconfig*).

```
1 | $ iw wlan0 up
```

## Creazione file di configurazione

Il secondo passo è quello di creare un file di configurazione, dove i nostri tool che ci permetteranno di configurare la rete andranno a salvare informazioni che servono ad accedere alla rete e a mantenere la connessione con essa.

```
1 | /etc/wpa_supplicant.conf
2 | ctrl_interface=/run/wpa_supplicant
3 | update_config=1
```

## Avvio wpa\_supplicant

*wpa\_supplicant* è un tool che ci permette di stabilire una connessione wireless con chiave **WPA**. Quindi esso deve essere avviato specificando il file di configurazione dove lui si dovrà appoggiare.

```
1 | $ wpa_supplicant -B -i interface -c /etc/wpa_supplicant.conf
```

## Avvio del prompt interattivo del tool wpa

*wpa\_cli* è un prompt interattivo che funziona tramite *wpa\_supplicant*. Da esso avverrà la configurazione della connessione tra la nostra scheda di rete Wi-Fi e il router della rete Wi-Fi su cui vogliamo collegarci. Per semplicità l'SSID della rete wireless sarà *MIOSSID*.

```
1 | $ wpa_cli
```

Scansione delle reti con il comando *scan* e successiva stampa del risultato tramite *scan\_result*

```
1 | > scan                #scansione reti
2 | OK
3 | <3>CTRL-EVENT-SCAN-RESULTS
4 | > scan_results          #stampa
5 | bssid / frequency / signal level / flags / ssid
6 | 00:00:00:00:00:00 2462 -49 [WPA2-PSK-CCMP][ESS] MIOSSID
7 | 11:11:11:11:11:11 2437 -64 [WPA2-PSK-CCMP][ESS] ALTROSSID
```

Aggiunta della rete con inserimento delle credenziali tramite: *add\_network*, *set\_network* e *enable\_network*. Per semplicità la password della rete wireless sarà: *passphrase*

```

1 > add_network          #aggiunta rete
2 0
3 > set_network 0 ssid "MIOSSID"
4 > set_network 0 psk "passphrase"
5 > enable_network 0
6 <2>CTRL-EVENT-CONNECTED - Connection to 00:00:00:00:00:00 completed (reauth) [id=0 id_str=]

```

Questa combinazione di comandi non fa altro che creare una configurazione di rete nel file

`/etc/wpa_supplicant.conf` come segue:

```

1 network={
2     ssid="MIOSSID"
3     #psk="passphrase"
4     psk=59e0d07fa4c7741797a4e394f38a5c321e3bed51d54ad5fcbd3f84bc7415d73d
5 }

```

Se la connessione è avvenuta con successo, allora bisogna salvare la configurazione con il comando `save_config`:

```

1 >save_config
2 OK

```

## Richiesta indirizzo IP

Ultimo passaggio è quello della richiesta dell'indirizzo IP tramite il comando `dhcpcd` per finalmente partecipare alla rete, se non presente il comando deve essere installato, poichè necessario per la comunicazione con il server `dhcpd` del router.

```

1 $ dhcpcd wlan0

```

## Risoluzione dei problemi

Se per qualche motivo dopo la configurazione di `wpa_supplicant` la connessione dovesse fallire allora necessitano dei comandi per la gestione delle sessioni di `wpa_supplicant`:

- Il comando `ps ax | grep "wpa_supplicant -B" | grep -v grep` permette di visualizzare l'id di tutte le sessioni che sono state aperte con `wpa_supplicant`, in maniera tale da poterle gestire, visto che per esistere una connessione con questo tool, deve essere attiva un'unica sessione di `wpa_supplicant`
- il comando `kill $(pgrep -f "wpa_supplicant -B")` chiuderà tutte i processi di `wpa_supplicant`, così da darmi la possibilità di riconfigurare la rete in maniera diversa per far avvenire una effettiva connessione

## Configurazione del server

Il sito web che abbiamo creato è una piattaforma mock di condivisione di tracce audio generate dagli utenti. Prevede che gli utenti si registrino prima di poter effettuare delle ricerche o dei nuovi caricamenti. Il server sarà eseguito sull'OrangePi e la comunicazione avverrà sul protocollo HTTP.

## Installazione e configurazione dei software necessari

Il server farà uso di Apache, MySQL, PHP5 e phpmyadmin.

```
1 | sudo apt-get install mysql-server
2 | sudo apt-get install apache2
3 | sudo apt-get install php libapache2-mod-php
4 | sudo apt-get install phpmyadmin
```

Accedendo dal browser all'indirizzo `MIOIP/phpmyadmin` è possibile eseguire la configurazione iniziale automatica, alla quale sarà richiesta la scelta di una password per l'utente `phpmyadmin`.

Successivamente, sarà necessario creare un database per il nostro servizio, per il quale l'utente `phpmyadmin` avrà tutti i permessi.

```
1 | sudo mysql -u root
```

```
1 | CREATE DATABASE catalogomusica;
2 | GRANT ALL PRIVILEGES ON catalogomusica.* TO 'phpmyadmin'@'localhost';
```

# Sviluppo del sito web

## Database

Le tabelle del database sono state create tramite phpmyadmin. Di seguito riportiamo la tabella degli utenti, quella degli autori e quella dei loro brani.

```
1  # utenti
2  +-----+-----+-----+-----+-----+-----+-----+-----+
3  | id | username | password | email | Nome | Cognome | admin | avatar |
4  +-----+-----+-----+-----+-----+-----+-----+-----+
5  | 1 | ikros | linux | aldof98@hotmail.it | Aldo | Fumagalli | 1 | NULL |
6  | 12 | frank | linux | frango@pr.it | Frango | Torregrossa | 0 | NULL |
7  +-----+-----+-----+-----+-----+-----+-----+-----+
8
9  # autori
10 +-----+-----+-----+-----+
11 | id | Nome | Genere | Eta |
12 +-----+-----+-----+-----+
13 | 1 | Pink Floyd | Rock Psichedelico | NULL |
14 +-----+-----+-----+-----+
15
16 # brani
17 +-----+-----+-----+-----+-----+-----+-----+-----+
18 | id | Titolo | Album | Anno | autore | file | immagine |
19 +-----+-----+-----+-----+-----+-----+-----+-----+
20 | 1 | wish you were here | wish you were here | 1975 | 1 | a.mp3 | a.png |
21 +-----+-----+-----+-----+-----+-----+-----+-----+
```

## Login, Sessione, gestione database e logout

Quando un client per la prima volta fa una richiesta al server effettuerà una connessione al database tramite uno script php.

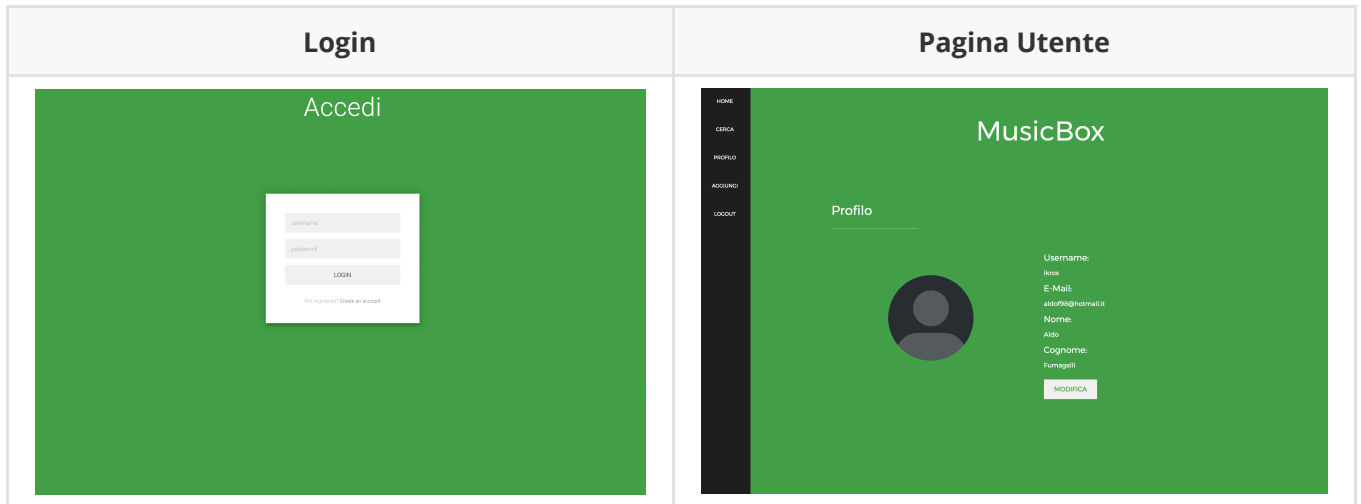
La sessione viene creata non appena un utente compila correttamente il form di login tramite l'email e la password del suo profilo. Essa serve per identificare l'utente all'interno delle varie pagine del sito web.

Il logout viene effettuato da uno script php che cancella la sessione, non permettendo all'utente di entrare nelle pagine interne della pagina.

## Registrazione e pagina utente

Il form all'interno della pagina di registrazione permette di effettuare una registrazione all'interno del sito, creando un profilo utente. Successivamente l'utente sarà reindirizzato alla Home page.

Nella pagina utente è presente un riepilogo delle informazioni sensibili che compongono l'account compresa l'immagine del profilo. All'interno di questa pagina è presente un pulsante "Modifica" che permette di modificare le informazioni del profilo dell'utente, compresa la password. La pagina di modifica è stata realizzata tramite un form pre-compilato dalle informazioni dell'utente, che esso stesso può modificare e poi infine farne il submit.



## Home, ricerca e scheda brano

Nella Home page, come in ogni pagina all'interno del sito è presente una navigation-bar che permette all'utente di navigare all'interno del sito.

La pagina di ricerca è composta da un form dove è possibile inserire diverse informazioni riguardo al brano o all'autore da ricercare. Dopo aver effettuato il submit lo script comporrà una query per il database e successivamente ne saranno mostrati i risultati componendo codice HTML e CSS. I risultati sono collegati alle proprie pagine di "scheda brano" dove è appunto possibile ascoltarli tramite HTML5. Di seguito le parti principali del codice della pagina di ricerca:


```

1  <html>
2  <head>
3
4  <!-- ... -->
5
6  <!-- Search form -->
7  <form action="<?php $_SERVER['PHP_SELF'] ?>" method="post" autocomplete="off">
8      <p><input type="text" placeholder="Titolo" name="titolo"></p>
9      <p><input type="text" placeholder="Autore" name="autore"></p>
10     <p><input type="text" placeholder="Album" name="album"></p>
11     <p><input type="text" placeholder="Anno" name="anno" maxlength="4"></p>
12     <p>
13         <button type="submit" name="cerca">
14             <i class="fa fa-search"></i> CERCA
15         </button>
16         <button type="reset">
17             <i class="fa fa-trash-o" aria-hidden="true"></i> CANCELLA
18         </button>
19     </p>
20 </form>
21
22 <!-- ... -->
23
24 <!-- Script PHP che comporrà la query in base alle informazioni del form -->
25 <?php
26 //qua viene generata la mia query
27 if(isset($_POST["cerca"])) {
28     require "database.php";
29     $sql= "SELECT B.id, B.titolo, A.Nome AS autore, B.album, B.anno," .
30         " A.id AS id_aut FROM brani AS B, autori AS A WHERE B.id=A.id AND ";

```

[illegible]

```
91
92 </body>
93 </html>
```

Ricerca	Scheda brano
<div><div>HOME CERCA PROFILO ACCESSIONE LOGOUT</div><div><h1>MusicBox</h1><h2>Cerca brano</h2><div><div>Titolo</div><div>Autore</div><div>Album</div><div>Anno</div><div>CERCA</div><div>CANCELLA</div></div></div></div>	<div><div>HOME CERCA PROFILO ACCESSIONE LOGOUT</div><div><h1>MusicBox</h1><h2>Brano</h2><div><div><p>Titolo: wish you were here</p><p>Album: wish you were here</p><p>Anno: 1975</p><p>Autore: Pink Floyd</p><p>Riproduci:</p><div><div></div><div></div><div>0:00</div></div><div>MODIFICA</div></div></div></div></div>

# Aggiunzione dei brani

Gli admin hanno la possibilità di aggiungere brani. Questo è possibile tramite l'apposita pagina composta da un form in cui è possibile inserire tutti i dettagli del brano. Inoltre è predisposta una una sezione dove è possibile effettuare l'upload del brano.



# Prova dello sniffer

Per testare il funzionamento dello sniffer, lo abbiamo avviato insieme al server web. Dal momento che stiamo cercando pacchetti di tipo HTTP contenenti dei dati sensibili, possiamo lanciare lo sniffer da remoto (tramite *ssh*) con i seguenti filtri

```
1 gcc sniffer.c protocols/*.c protocols/*.h -o sniff
2 sudo ./sniff --noicmp --noudp --nunknown --noplainempty --port 80
```

Così facendo, l'output conterrà solamente pacchetti TCP (anche con gli header IP ed Ethernet) che hanno un contenuto non vuoto. Inoltre, sappiamo che la porta del server sarà 80, perciò terremo solo conto delle comunicazioni da e verso quella porta. Una volta individuato l'indirizzo IP del server, potremmo anche filtrare l'output con `--ip`.

In questo esempio, un amministratore del sito sta eseguendo il login con le sue credenziali. Dallo sniffer possiamo vedere la richiesta POST contenente i dati dell'utente, separata in due pacchetti. Inoltre, possiamo leggere il codice di sessione PHP che questo utente sta usando.

## Richiesta POST tramite il form di accesso

```
-----
Ethernet Header:
1<P... - Destination: 7C:8B:CA:05:62:A8, Source: 38:2C:4A:BD:CC:45, Protocol: 8
0P0... IP Header:
80... - Destination: 192.168.1.121, Source: 192.168.1.124, Protocol: 6
      - Version: 4, Header Length: 5, TTL: 64
      - Id: 0, Total Length: 585, Type of Service: 0
ADCA... TCP Header:
SUM... - Source Port: 65052, Destination Port: 80
:bd... - Sequence Number: 95222284, Acknowledgment: 153007706
:520... - Data Offset: 8, Window Size: 4117
1<P... - CWR: 0, ECE: 0, URG: 0, ACK: 1
ect... - PSH: 1, RST: 0, SYN: 0, FIN: 0
-----
Data Length: 533, Raw Data:
POST / HTTP/1.1..Host: 192.168.1.121..Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8..Accept-Encoding: gzip, deflate..Accept-Language: en-gb..Content-Type: application/x-www-form-urlencoded..Origin: http://192.168.1.121..User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/12.1.1 Safari/605.1.15..Connection: keep-alive..Upgrade-Insecure-Requests: 1..Referer: http://192.168.1.121/..Content-Length: 29..Cookie: PHPSESSID=9167ne2vd42cvhah84pfj4bp6h....
-----
```

```
-----
Ethernet Header:
- Destination: 7C:8B:CA:05:62:A8, Source: 38:2C:4A:BD:CC:45, Protocol: 8
IP Header:
- Destination: 192.168.1.121, Source: 192.168.1.124, Protocol: 6
- Version: 4, Header Length: 5, TTL: 64
- Id: 0, Total Length: 81, Type of Service: 0
TCP Header:
- Source Port: 65052, Destination Port: 80
- Sequence Number: 95222817, Acknowledgment: 153007706
- Data Offset: 8, Window Size: 4117
- CWR: 0, ECE: 0, URG: 0, ACK: 1
- PSH: 1, RST: 0, SYN: 0, FIN: 0
-----
Data Length: 29, Raw Data:
username=ikros&password=linux
-----
```

Abbiamo poi avviato un altro browser e caricato il sito web. Anche se ancora non abbiamo fatto l'accesso, PHP ci fornisce un cookie di sessione.

PHPSESSID di un utente non registrato

Signup

Not Secure | 192.168.1.121

Incognito

Outlook - aldof98...

YouTube - Broadc...

Facebook

Swift - Apple (IT)

What Is My IP Add...

Google Traduttore

speed test

PayPal: Riepilogo

informatica

tesina

Accedi

username

password

LOGIN

Not registered? Create an account

Application

Manifest

Service Workers

Clear storage

Storage

Local Storage

Session Storage

IndexedDB

Web SQL

Cookies

http://192.168.1.121

Cache

Cache Storage

Application Cache

Background Services

Background Fetch

Background Sync

Notifications

Payment Handler

Push Messaging

Frames

top

Filter

Name	Value	Domain	Path	Expir...	S...	Http...	Sec...	Sa...
PHPSESSID	sp9mp1v3kdra247kep2q09i1td	192.168...	/	Sessi...	3...			

È possibile modificare il valore del cookie copiando quello ottenuto dallo sniffer, così facendo quando la pagina sarà ricaricata il web server non potrà distinguerci dall'amministratore.

Sostituzione del PHPSESSID

Signup

Not Secure | 192.168.1.121

Incognito

Outlook - aldof98...

YouTube - Broadc...

Facebook

Swift - Apple (IT)

What Is My IP Add...

Google Traduttore

speed test

PayPal: Riepilogo

informatica

tesina

Accedi

username

password

LOGIN

Not registered? Create an account

Application

Manifest

Service Workers

Clear storage

Storage

Local Storage

Session Storage

IndexedDB

Web SQL

Cookies

http://192.168.1.121

Cache

Cache Storage

Application Cache

Background Services

Background Fetch

Background Sync

Notifications

Payment Handler

Push Messaging

Frames

top

Filter

Name	Value	Domain	Path	Expires...	S...	Http...	Secure	Same...
PHPSESSID	9i67ne2vd42cvhah84pf4bp6h	192.168.1....	/	Session	35			

