ITIS
LEONARDO DA VINCI
Via Toscana, 10 - 43122 PARMA - Tel 0521266511 - Fax 0521266550 - e-mail itis@itis.pr.it - cf.80007330345 - PRTF010006

CORSO DI CYBER SECURITY

Incontro #07 - Mer 17 DIC 24

Prof. Ugolotti 2024-2025

Obiettivi del corso:

OLIMPIADI ITALIANE DI CYBERSICUREZZA

CYBER CHALLENGE.IT

Selezione scolastica
18/01/2025

Preselezione
25/28 Gennaio 2025

Non dimenticate il primo obiettivo....

https://olicyber.it/materiale

La Selezione scolastica (18 Gen 2025)

- networking
- Indirizzamento IPv4
- Funzioni ricorsive
- Operatori bitwise
- Calcolo combinatorio
- Probabilità
- Logica

Cosa faremo Oggi: Web Security

1. Cookies
2. Sessione Utente / la libreria request in python e l’oggetto Session
3. Database
4. SQLi

Sessione utente

- Eseguita l'autenticazione, il server affida all'utente una identità
- L'applicazione web è poi incaricata di *ricordarsi* l'identità dell'utente
- Come il login, anche questa operazione deve essere eseguita in un ambiente trusted, altrimenti un utente malevolo potrebbe spacciarsi per un altro utente

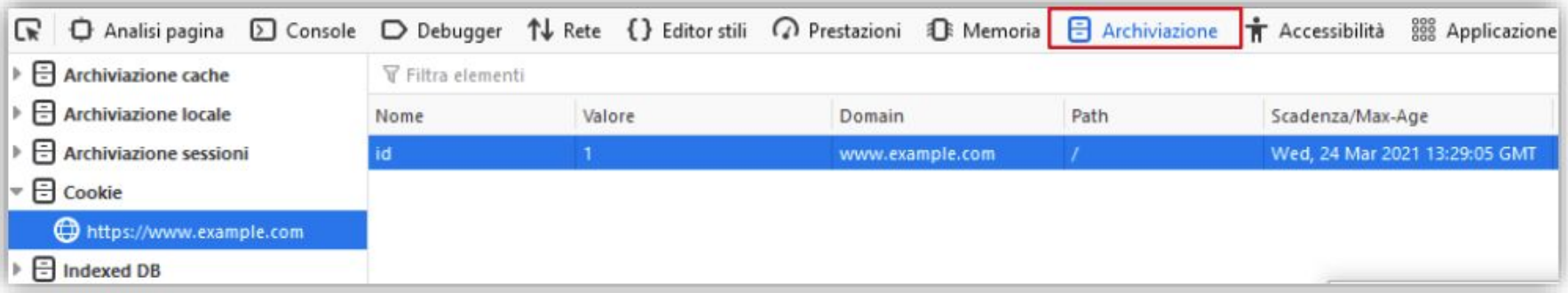
Cookies

- HTTP mette a disposizione i cookies per questa operazione
- I cookies sono una informazione che il browser salva localmente e che invia in ogni richiesta HTTP

I cookies non sono affidabili!

Cookies

- Cliccando sul nome-valore sarà poi possibile modificare il cookie



Cookies

Impostazioni di Sicurezza:

1. I cookie settati da un dominio sono accessibili solo a quel dominio
2. Ogni dominio **non** può settare o leggere i cockie delgi altri domini visitati (non direttamente)

Cookies

- Da browser, i due modi più semplici per creare/modificare/cancellare cookie sono:
  - Usare la console di sviluppo
  - Usare Javascript

```
document.cookie="nome=valore"
```

Cookies

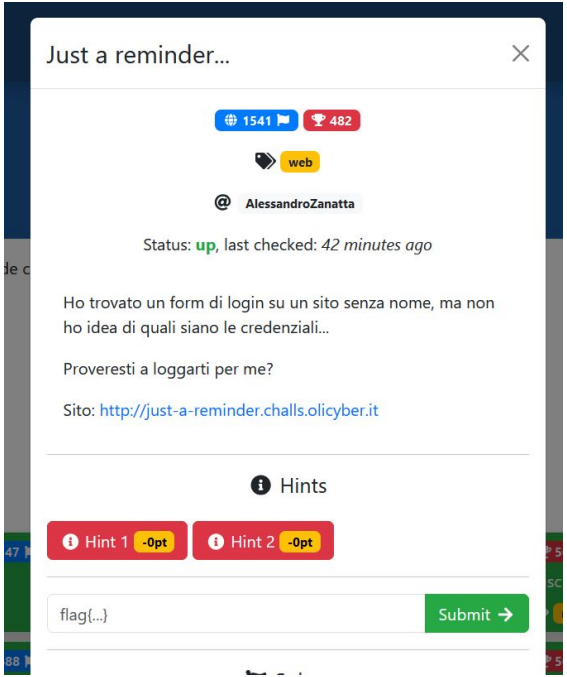
- Come detto prima, tutta la parte di autenticazione deve essere eseguita in un ambiente trusted
- Come è possibile osservare, i cookies non sono affidabili, in quanto è possibile cambiarli a piacimento

Autenticazione sicura

Considera un sito web con meccanismo di autenticazione. Quale tra le seguenti operazioni sono necessarie per avere un processo di login sicuro?

Risposte

(**A**) Verifica username e password lato client
(**B**) Verifica username e password lato server
(**C**) Verifica username e password lato client e server
(**D**) Nessuna delle altre risposte



⚔️"Just a reminder..."🌎

Compiti per casa!

Gestione dati lato client:
https://training.olicyber.it/challenges#challenge-43

Sessione Utente

Per rendere sicuro il cookie di sessione:

1. Firmare digitalmente il cookie
2. Generare una “password” temporanea (session-id) al login, che viene verificata ad ogni richiesta successiva

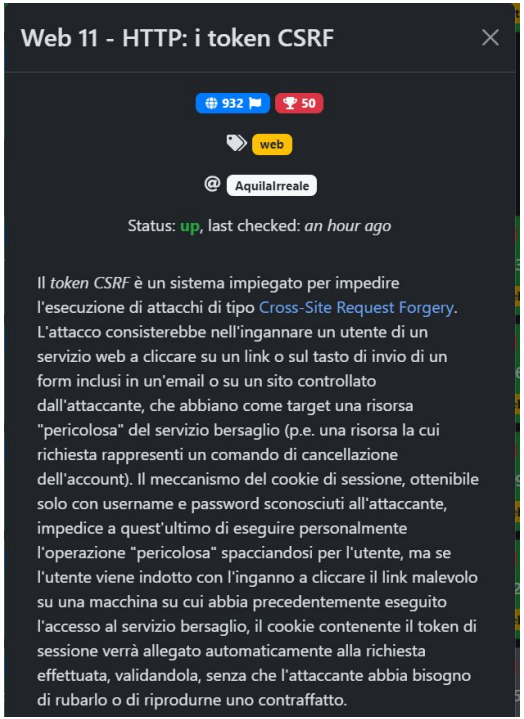
```python
import requests

# URL delle risorse
BASE_URL = "http://web-11.challs.olicyber.it"
LOGIN_URL = f"{BASE_URL}/login"
FLAG_URL = f"{BASE_URL}/flag_piece"

# Credenziali di login
credentials = {
    "username": "admin",
    "password": "admin"
}

# Creazione della sessione
session = requests.Session()

# Login per ottenere il cookie di sessione e il token CSRF iniziale
login_response = session.post(LOGIN_URL, json=credentials)
if login_response.status_code != 200:
    print(f"Errore durante il login: {login_response.status_code}")
    exit()
print(login_response.json())
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
[Running] python -u "c:\Users\rugol\WorkSpace\CSFR.py"
{'status': 'ok', 'csrf': '4feefa60708a81ac'}
```

CAPTURE THE FLAG

Challenge: Web 11 - HTTP: i token CSRF




```python
# Parsing del token CSRF iniziale
csrf_token = login_response.json().get("csrf")
if not csrf_token:
    print("Token CSRF non trovato nella risposta del login.")
    exit()

# Recupero dei pezzi della flag
flag = ""
for i in range(4):
    # Aggiunta del token CSRF nell'header
    # headers = {"X-CSRF-Token": csrf_token}
    # Richiesta del pezzo di flag
    response = session.get(FLAG_URL, params={"index": i, "csrf": csrf_token})
    if response.status_code != 200:
        print(f"Errore nel recupero del pezzo {i}: {response.status_code}")
        exit()
    # Aggiornamento del token CSRF
    csrf_token = response.json().get("csrf")
    if not csrf_token:
        print(f"Token CSRF non trovato nella risposta per il pezzo {i}.")
        exit()
    # Aggiunta del pezzo alla flag
    flag += response.json().get("flag_piece")

# Output della flag completa
print(f"\nFlag completa: {flag}")
```

Signed Cookie

In pratica il server genere

- Un ID univoco per l'utente loggato
- Un certificato digitale (***sign***) che garantisce l'autenticità del cookie

Session-id …vulnerabili

Chiaramente

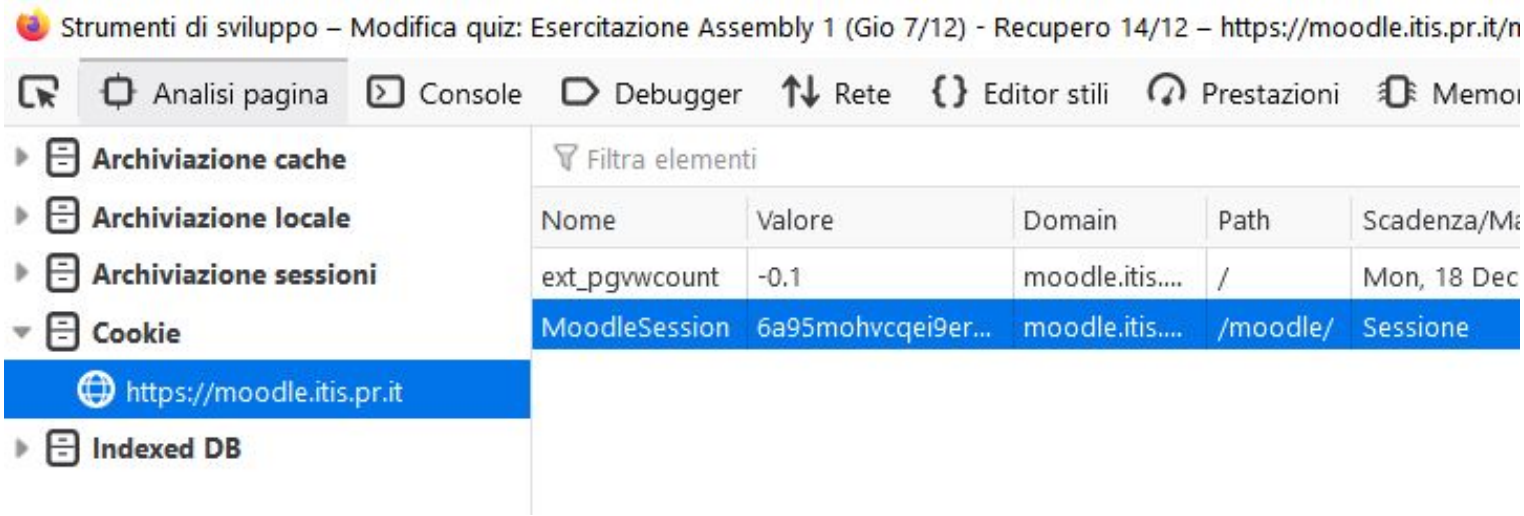
- Session-ID devono avere entropia adeguata
- Altrimenti sono facilmente “GUESS-ABILI”

```
HTTP/1.1 302 FOUND
Date: Wed, 24 Mar 2021 16:36:04 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 223
Connection: close
Server: gunicorn/19.9.0
Location: /cookies
Set-Cookie: session=32; Path=/
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
```

CAPTURE THE FLAG

Challenge: Web??? - ???

Cookie di Sessione



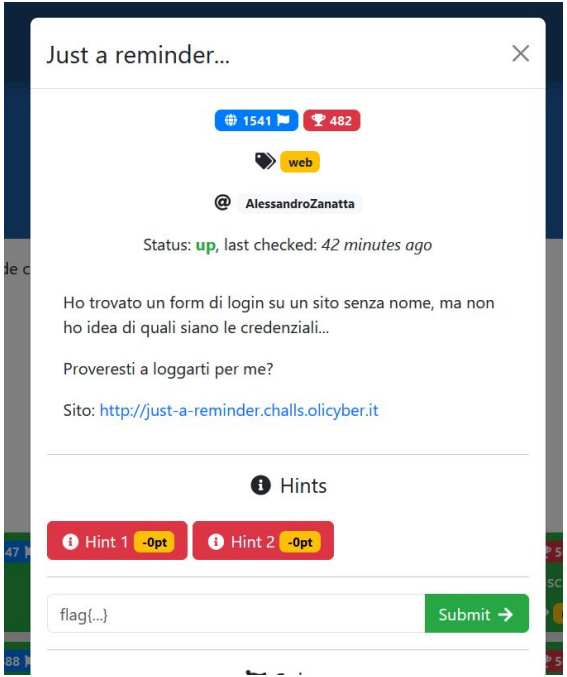
Durante la lezione il prof. mostra (incautamente) il proprio session_id del login moodle... e a fine lezione si trova il profilo hackerato

Autenticazione sicura

Considera un sito web con meccanismo di autenticazione. Quale tra le seguenti operazioni sono necessarie per avere un processo di login sicuro?

Risposte

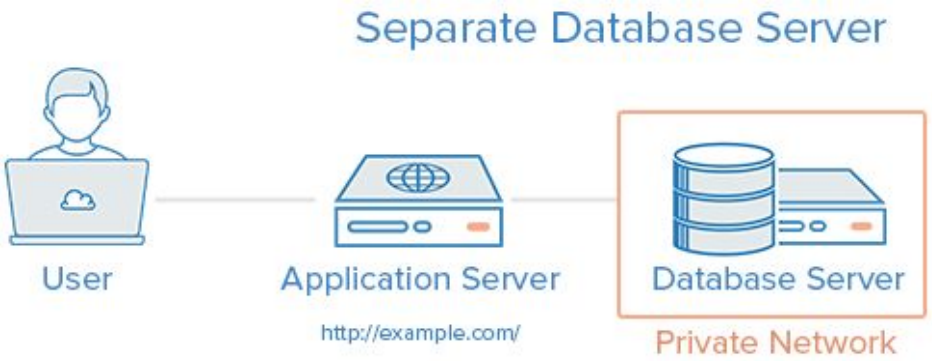
(**A**) Verifica username e password lato client
(**B**) Verifica username e password lato server
(**C**) Verifica username e password lato client e server
(**D**) Nessuna delle altre risposte



⚔️“Just a reminder...”🌎

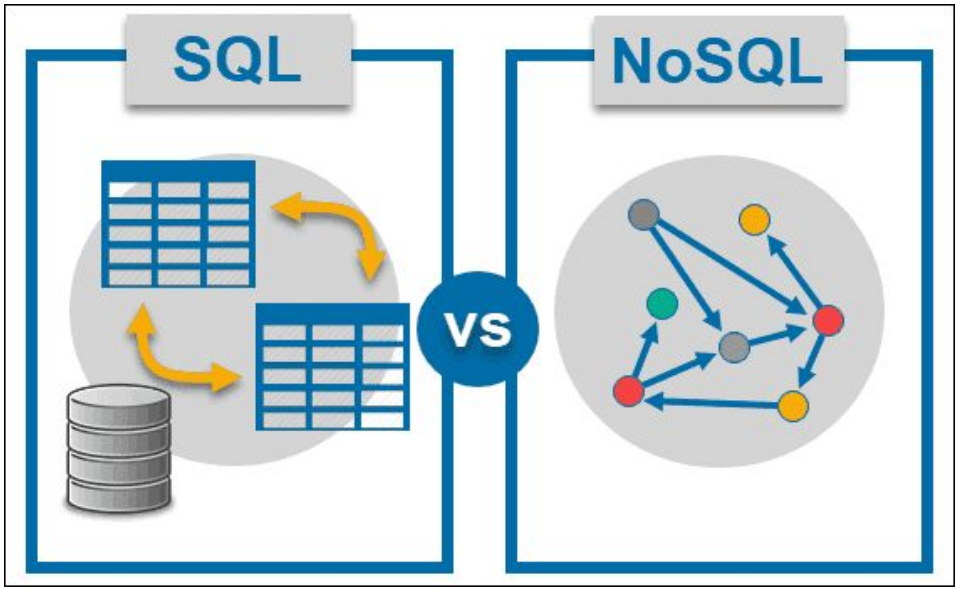
Database

- Il server WEB è ***stateless***
  - Ha bisogno di un server scripting per la gestione delle pagine dinamiche (php, python, node.js, ecc)
  - Ha bisogno di un DBMS per memorizzare (input utente, scelte, ecc) e per recuperare informazioni



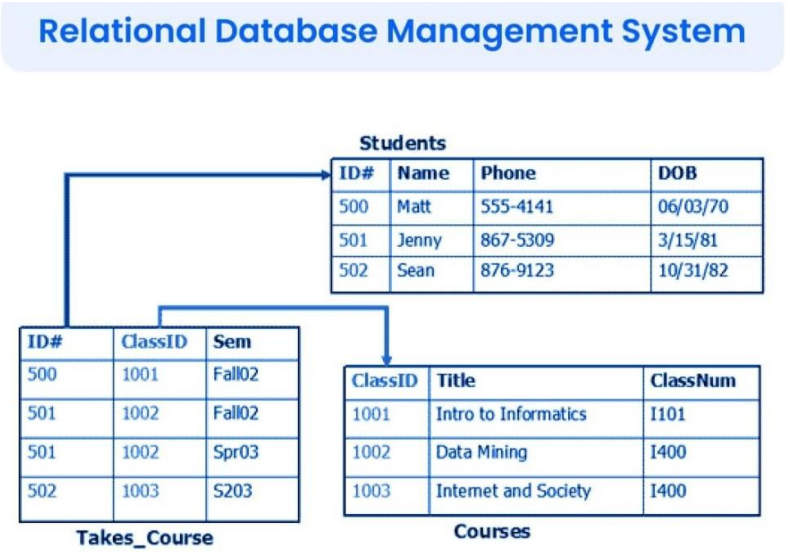
Database …più tipi

- RELAZIONALI (SQL)
- NoSQL
- BigData
- …



Database Relazionali

- E’ un insieme di tabelle
- Talvolta collegate tra loro medianti “Chiavi”
- Il DBMS fornisce:
  - meccanismi di autenticazione
  - gestione permessi (cosa possiamo fare e su quali tabelle)



Database Relazionale… Un Esempio

- Vediamo la struttura del DB di una diffusissimo CMS: Wordpress
- Esempi di Query
-

SQL Injection

- Tecnica per sfruttare vulnerabilità in applicazioni mal programmate

Users

	UserName	Passwd	UserID
1	A-JayBibbins637	uslwfpua	10000
2	A-JayTorain976	iyvqxzrq	10001
3	AadamDobbin507	aufvxyuy	10002
4	AadamHaws649	vonolnnv	10003
5	AadamThiengtham0	ydypixak	10004
6	AadhishAyon371	fbecsnda	10005
7	AadiAccardi419	imkboytx	10006
8	AadiGnau621	12345	10007
9	AadilMarinko33	juhktina	10008
10	AadishivLathon60	ohhiumw	10009
11	AadishivPrioletti873	dragon	10010
12	AadityaTerre708	123456	10011
13	AadmClary990	uqqbdmdr	10012
14	AadvikAllman584	qwerty	10013
15	AadvikMannheimer804	aforxkuf	10014
16	AadvaRogan194	dragon	10015