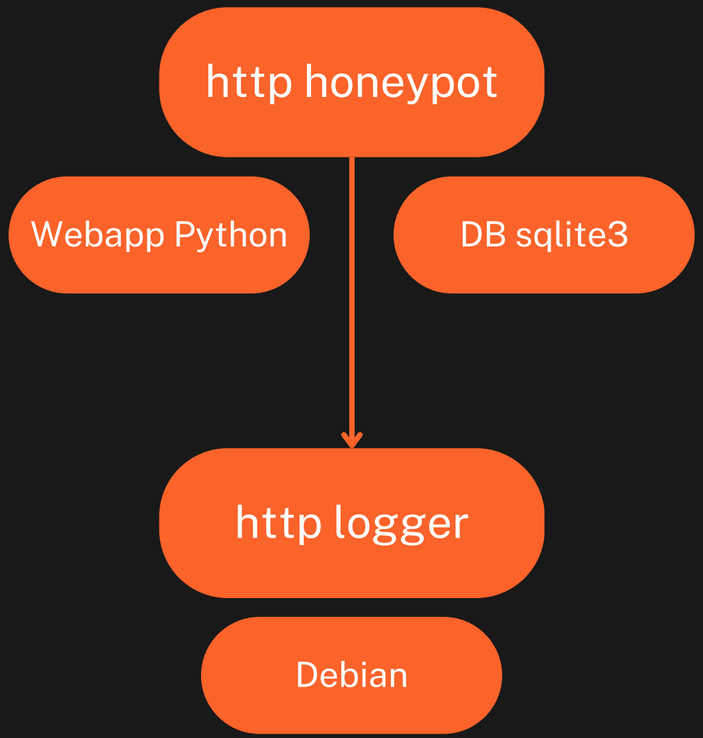
PROGETTO NETWORK SECURITY

# Docker Honeypot

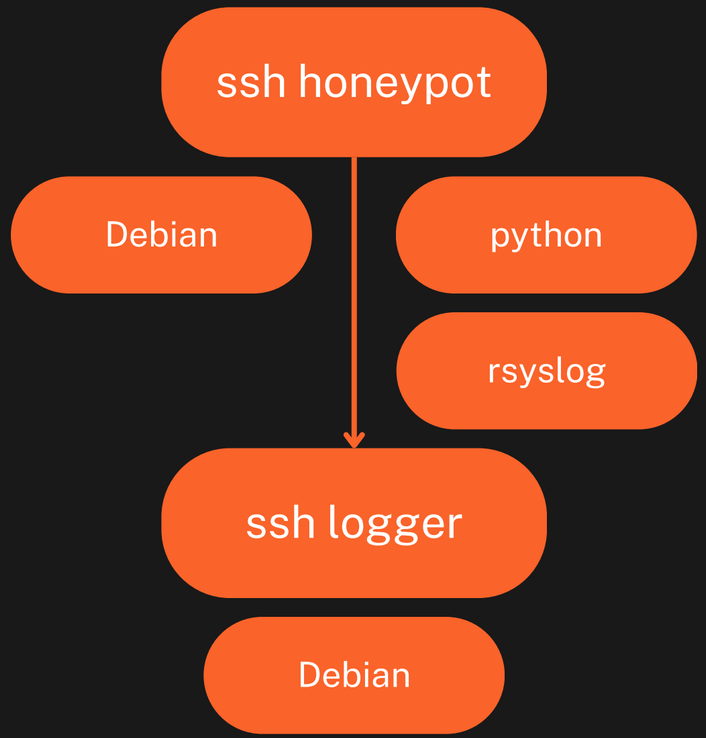
ALESSANDRO FALINO

# COMPOSIZIONE

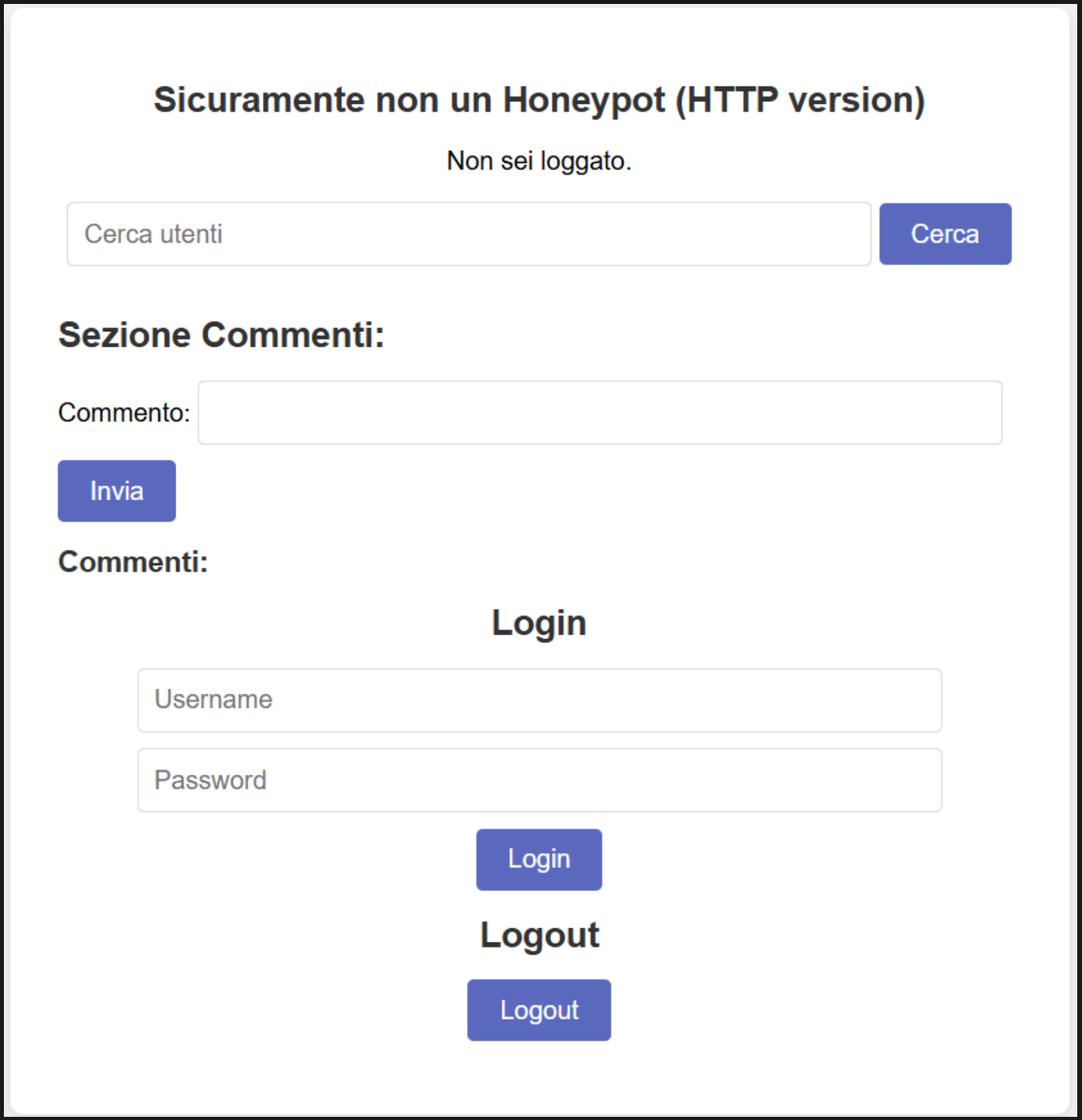
## HONEYPOT WEBAPP




## HONEYPOT SSH



# HTTP

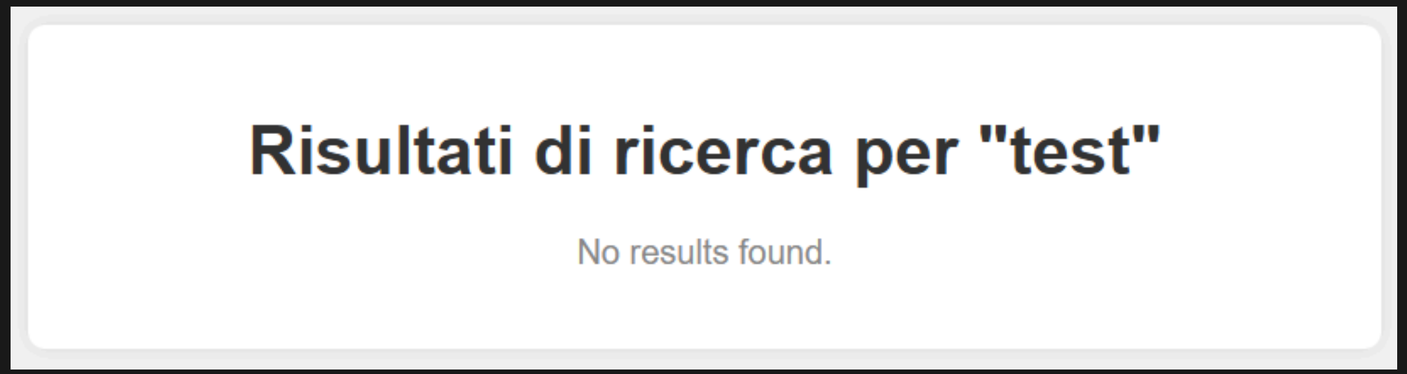



La webapp presenta svariate vulnerabilità.

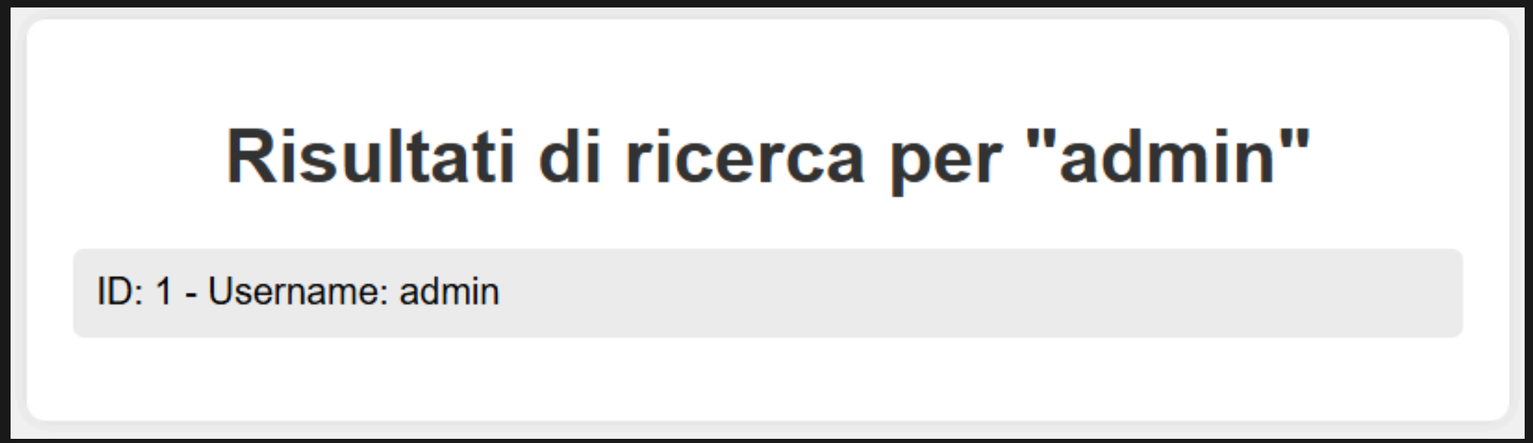
L'idea alla base della webapp è incanalare un eventuale attaccante, per tenerlo occupato il più a lungo possibile

La webapp è collegata ad un semplicissimo DB con dati fittizi, e presenta una sezione "commenti" dove è possibile postare testo, se loggati.
Il flusso ideale che si vorrebbe far seguire è quindi quello di SQLi per trovare user/pass per poter eseguire RCE/XSS

# HTTP




Partiamo dal DB. Possono essere eseguite semplici query di ricerca username semplicemente digitando l'username da cercare nella barra di ricerca della home page




Le query NON sono messe in sicuRezza per cui è possibile visualizzare tutto nel DB, ma la webapp non permette di eseguire più query insieme.

Inoltre la pagina dei risultati mostra solo i primi 2 risultati.

# HTTP

```
<h2>Commenti:</h2>
<div>
    {% for comment in comments %}
        <div class="comment-box">{{ comment|safe }}</div>
    {% endfor %}
</div>
```

**La sezione commenti invece presenta una vulnerabilità di XSS dato che è stato usato lsafe, che tratta il testo come safe, anche se non lo è ( e non lo sarà)**

```
comment = request.form.get("comment", "")
```

**Input non sanitizzato**

# HTTP

**Il sistema di logging è molto semplice, ed è collegato al container vulnerabile, i due condividono un volume.**

**Il sistema vulnerabile carica sul volume il file di log, ed il sistema di logging lo legge e lo copia su un secondo file, per evitare che un attaccante possa cancellarlo.**

```
[2025-03-11 15:33:17,922] - INFO - Request: GET / - IP: 172.21.0.1
[2025-03-11 15:33:17,922] - INFO - Pagina principale visitata
[2025-03-11 15:33:18,454] - INFO - 172.21.0.1 - - [11/Mar/2025 15:33:18] "GET / HTTP/1.1" 200 -
[2025-03-11 15:33:25,361] - INFO - Request: GET /search - IP: 172.21.0.1
[2025-03-11 15:33:25,382] - INFO - Query eseguita: SELECT * FROM users WHERE username LIKE 'admin'
[2025-03-11 15:33:25,417] - INFO - 172.21.0.1 - - [11/Mar/2025 15:33:25] "GET /search?q=admin HTTP/1.1" 200 -
[2025-03-11 15:33:34,699] - INFO - Request: POST /login - IP: 172.21.0.1
[2025-03-11 15:33:34,707] - INFO - 172.21.0.1 - - [11/Mar/2025 15:33:34] "ESC[32mPOST /login HTTP/1.1ESC[0m" 302 -
[2025-03-11 15:33:34,722] - INFO - Request: GET / - IP: 172.21.0.1
[2025-03-11 15:33:34,722] - INFO - Pagina principale visitata
[2025-03-11 15:33:35,211] - INFO - 172.21.0.1 - - [11/Mar/2025 15:33:35] "GET / HTTP/1.1" 200 -
[2025-03-11 15:33:48,360] - INFO - Request: POST / - IP: 172.21.0.1
[2025-03-11 15:33:48,360] - INFO - Pagina principale visitata
[2025-03-11 15:33:48,828] - INFO - New comment added: admin: sto postando un commento
[2025-03-11 15:33:48,829] - INFO - 172.21.0.1 - - [11/Mar/2025 15:33:48] "POST / HTTP/1.1" 200 -
```

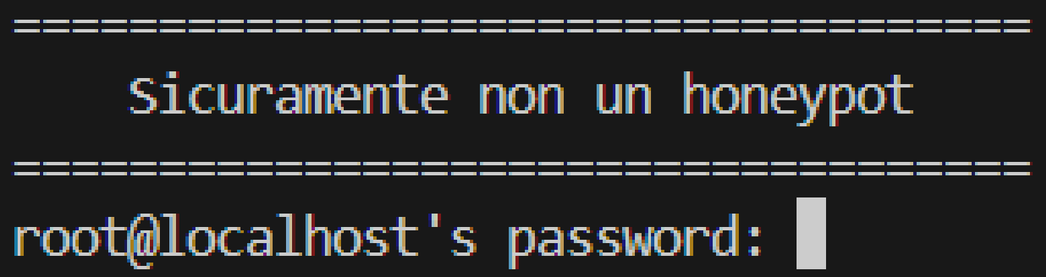
```
- Request: GET /search - IP: 172.21.0.1
- Query eseguita: SELECT * FROM users WHERE username LIKE 'admin'
- 172.21.0.1 - - [11/Mar/2025 15:33:25] "GET /search?q=admin HTTP/1.1" 200 -
```

```
- Request: POST / - IP: 172.21.0.1
- Pagina principale visitata
- New comment added: admin: sto postando un commento
- 172.21.0.1 - - [11/Mar/2025 15:33:48] "POST / HTTP/1.1" 200 -
```


**Infine viene mandato un messaggio ogni volta che la pagina principale (/) viene acceduta**

# SSH

IL server ssh è un semplice server che logga tutti gli input di tutti gli utenti




il principio di base per il sistema di logging è lo stesso della webapp, un secondo container che prende i log dal primo e li copia per evitare manomissioni

```
root:[2025-02-02 14:00:34] User connected, IP(172.22.0.1), Port(35000)
root:[2025-02-02 14:00:37] test 05
root:[2025-02-02 14:00:41] cd ..
root:[2025-02-02 14:00:44] cd var/log
```

# SSH

Il container vulnerabile utilizza open_ssh e rsyslog+python per loggare le connessioni

Possono essere impostati utenti fittizi con user e password facili (o difficili) da indovinare in modo che un eventuale attaccante debba perdere tempo per attuare una escalation of privilege

Il logging registra tutti i comandi eseguiti con un wrapper bash mentre rsyslog tiene traccia delle connessioni, ed uno script python forma il file di log finale