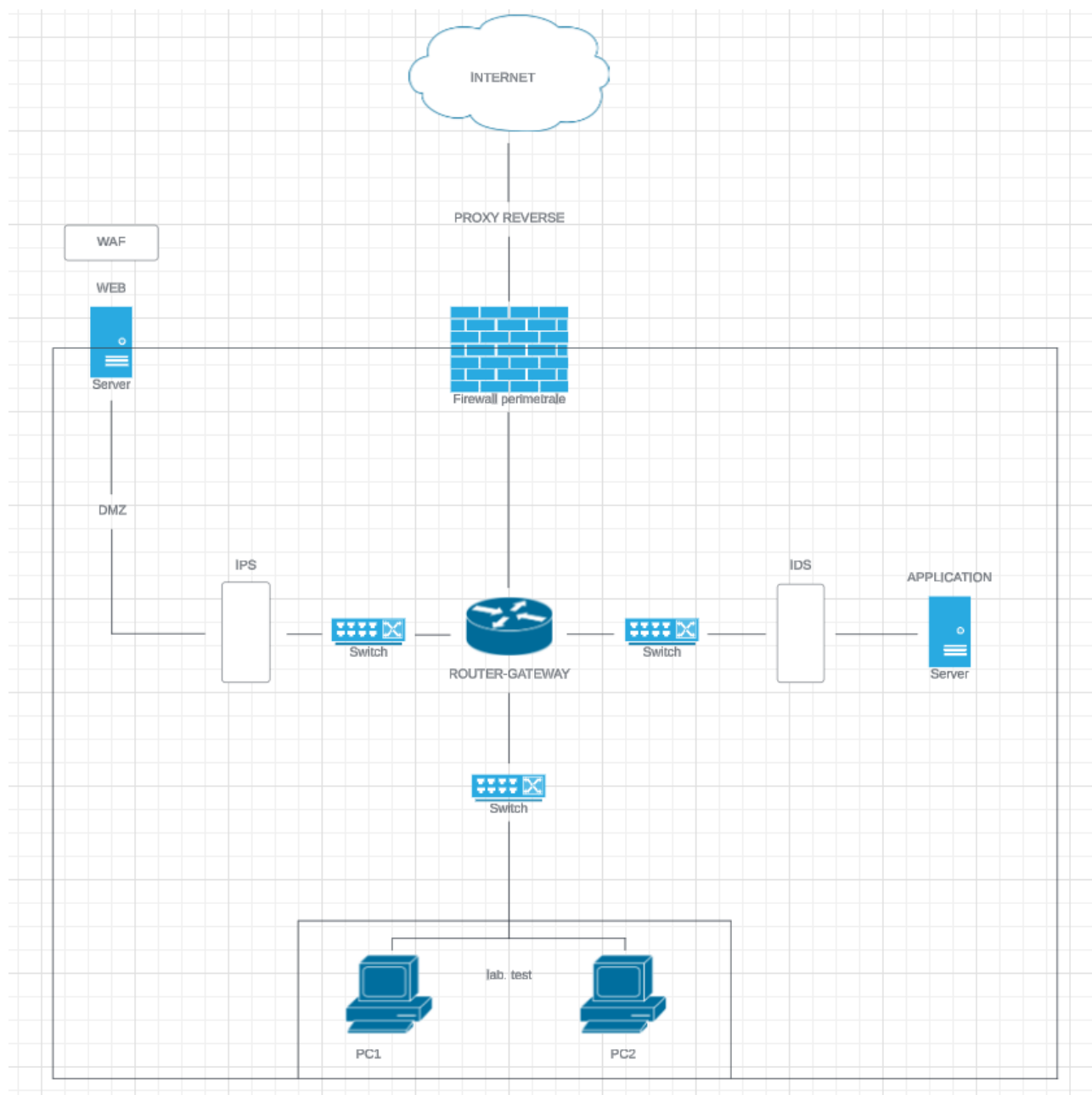


Valutazione sicurezza compagnia Theta

Dalle informazioni fornite dalla compagnia riguardante il perimetro delle attività, il disegno di rete per la messa in sicurezza delle componenti critiche comprende:



Il Web Server – il quale rappresenta un punto di accesso diretto alla compagnia da parte degli utenti su Internet e poiché è esposto, ci serviamo del WAF per proteggere questo server all'esterno. Il WAF esegue un controllo granulare delle richieste HTTP, analizzando i parametri e il contenuto delle richieste in ingresso. Può bloccare le richieste sospette o malformate prima che raggiungano il server web, riducendo il rischio di attacchi basati su input non validi.

Il secondo componente è Application server , dedicato a un'applicazione di e-commerce e risulta accessibile solo internamente. La sua sicurezza è cruciale per proteggere dati sensibili dei clienti e dell'azienda.

Proponiamo : un Firewall perimetrale per l'intera protezione della rete interna e per controllare l'accesso alle porte critiche . Si dovrebbe effettuare una configurazione che filtra il traffico in ingresso e limita l'accesso solo a porte essenziali (ad esempio, 80 per HTTP, 443 per HTTPS).

la DMZ, detta anche «demilitarized zone», è un segmento di rete che espone servizi raggiungibili da internet, per esempio un servizio di shopping online, o il servizio webmail. Poiché i flussi sono provenienti da internet è critico che questa rete sia protetta efficacemente.

Intrusion Prevention System (IPS) e Intrusion Detection System (IDS):

L'IPS, collocato dopo la DMZ, blocca attivamente le minacce, mentre l'IDS, posizionato prima dell'Application Server, monitora il traffico interno per rilevare comportamenti sospetti o attività malevole.

Virtual Private Network (VPN):

La VPN offre una connessione sicura per gli utenti autorizzati che necessitano di accedere ai servizi interni senza esporre direttamente il Web server a Internet. La crittografia e l'autenticazione garantiscono un accesso remoto sicuro.

Switch e Router Gateway:

Questi componenti forniscono la connettività di rete essenziale. Il router gateway collega la rete interna al firewall perimetrale, gestendo il traffico in ingresso e uscita.

VLAN

Una tecnologia di rete che consente di suddividere una rete fisica in più reti logiche indipendenti. Ogni VLAN è un gruppo di dispositivi che possono comunicare tra loro come se fossero collegati alla stessa rete fisica, indipendentemente dalla loro posizione fisica nella rete.

Benefici delle VLAN:

Segmentazione Logica: Le VLAN consentono di suddividere una rete in segmenti logici, migliorando la sicurezza e l'efficienza.

Isolamento del Traffico: Dispositivi in VLAN diverse non possono comunicare direttamente tra loro senza passare attraverso un router, fornendo un livello di isolamento del traffico.

Gestione Semplificata: La gestione della rete può essere semplificata, in quanto è possibile assegnare VLAN in base a criteri logici piuttosto che alla posizione fisica dei dispositivi.

Programma in Python per l'enumerazione dei metodi HTTP abilitati su un determinato target.

```
1 import requests
2
3 def enumerate_http_methods(target_url):
4     http_methods = ["GET", "POST", "PUT", "DELETE", "OPTIONS", "HEAD", "TRACE", "PATCH"]
5
6     for method in http_methods:
7         try:
8             response = requests.request(method, target_url)
9             print(f"Metodo: {method}, Stato: {response.status_code}")
10
11         except requests.RequestException as e:
12             print(f"Errore durante la richiesta con il metodo {method}: {e}")
13
14 if __name__ == "__main__":
15     # Chiedi all'utente di inserire l'URL di destinazione
16     target_url = input("Inserisci l'URL di destinazione: ")
17
18     try:
19         enumerate_http_methods(target_url)
20
21     except Exception as ex:
22         print(f"Errore generale: {ex}")
23
```

```
ter you become. The more you are able to hear"
(kali@kali)-[~/Desktop]
$ python verbshttp.py
Inserisci l'URL di destinazione: 
```

```
you are able to hear"
(kali@kali)-[~/Desktop]
$ python verbshttp.py
Inserisci l'URL di destinazione: http://192.168.50.5/phpMyAdmin/
```

```
(kali@kali)-[~/Desktop]
$ python verbshttp.py
Inserisci l'URL di destinazione: http://192.168.50.5/phpMyAdmin/
Metodo: GET, Stato: 200
Metodo: POST, Stato: 200
Metodo: PUT, Stato: 200
Metodo: DELETE, Stato: 200
Metodo: OPTIONS, Stato: 200
Metodo: HEAD, Stato: 200
Metodo: TRACE, Stato: 200
Metodo: PATCH, Stato: 200 e you are able to hear"
```

Importazione delle librerie:

```
import requests
```

Importa il modulo requests, che è una libreria Python comunemente utilizzata per effettuare richieste HTTP.

Definizione della funzione enumerate_http_methods:

Definisce una funzione denominata enumerate_http_methods che accetta un parametro target_url. Questa funzione sarà responsabile per l'invio delle richieste HTTP con vari metodi e la gestione delle risposte.

Definizione degli HTTP methods:

```
http_methods = ["GET", "POST", "PUT", "DELETE", "OPTIONS", "HEAD", "TRACE", "PATCH"]
```

Crea una lista di metodi HTTP che verranno eseguiti nella successiva iterazione.

Iterazione attraverso i metodi HTTP:

```
for method in http_methods:
```

Inizia un ciclo for che itera attraverso ogni metodo nella lista http_methods.

Invio di richieste HTTP:

```
response = requests.request(method, target_url)
```

Per ogni metodo, invia una richiesta HTTP all'URL di destinazione specificato, utilizzando il metodo corrente. La risposta viene memorizzata nella variabile response.

Gestione delle risposte HTTP:

```
print(f"Metodo: {method}, Stato: {response.status_code}")
```

Stampa il metodo HTTP utilizzato e lo stato della risposta ottenuta.

Gestione delle eccezioni:

```
except requests.RequestException as e:
```

```
    print(f"Errore durante la richiesta con il metodo {method}: {e}")
```

Nel caso in cui si verifichi un'eccezione durante l'invio della richiesta (ad esempio, una connessione di rete interrotta), gestisce l'eccezione stampando un messaggio di errore.

Sollevamento di un'eccezione per stati HTTP non 2xx:

```
response.raise_for_status()
```

Dopo l'invio di ciascuna richiesta, questa istruzione solleva un'eccezione se lo stato della risposta HTTP non è 2xx (successo).

Gestione specifica degli errori HTTP:

```
except requests.HTTPError as e:
```

```
    print(f"Errore HTTP durante la richiesta con il metodo {method}: {e}")
```

Aggiunge una clausola except specifica per gestire gli errori HTTP sollevati da raise_for_status(). Questo fornisce una gestione più dettagliata degli errori HTTP.

Blocco principale del programma:

```
if __name__ == "__main__":
```

Verifica se lo script è eseguito direttamente (non importato come modulo).

Input dell'utente: `target_url = input("Inserisci l'URL di destinazione: ")`

Chiede all'utente di inserire l'URL di destinazione.

Chiamata della funzione `enumerate_http_methods`:

`enumerate_http_methods(target_url)`

Esegue la funzione `enumerate_http_methods` con l'URL di destinazione inserito dall'utente come argomento.

Gestione delle eccezioni globali:

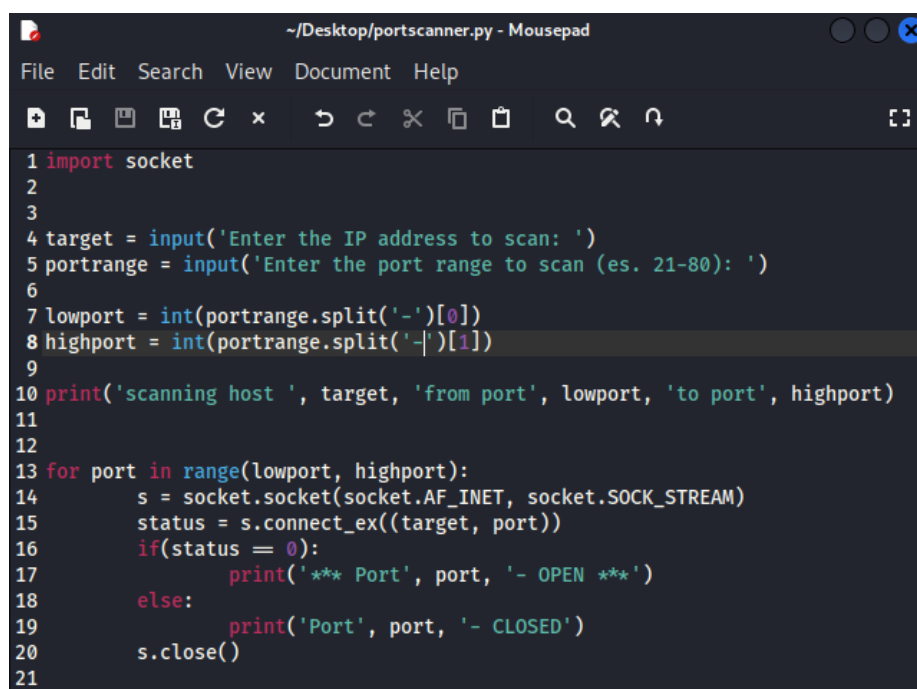
`except Exception as ex:`

`print(f"Errore generale: {ex}")`

Gestisce eventuali eccezioni globali che potrebbero verificarsi durante l'esecuzione del programma e stampa un messaggio di errore generale.

Quando un server supporta metodi come PUT o DELETE, che sono utilizzati per modificare o eliminare risorse sul server, ma tali operazioni non sono necessarie per il corretto funzionamento dell'applicazione, si crea un potenziale rischio di sicurezza. Questo perché un malintenzionato potrebbe sfruttare questi metodi per compiere azioni dannose, come la modifica o l'eliminazione non autorizzata di dati.

Programma in Python per la valutazione dei servizi attivi (port scanning).



The image shows a screenshot of a text editor window titled `~/Desktop/portscanner.py - Mousepad`. The editor contains a Python script for port scanning. The script imports the `socket` module and prompts the user for an IP address and a port range. It then iterates through the specified port range, attempting to connect to each port. If a connection is successful, it prints the port as 'OPEN'; otherwise, it prints the port as 'CLOSED'.

```
1 import socket
2
3
4 target = input('Enter the IP address to scan: ')
5 portrange = input('Enter the port range to scan (es. 21-80): ')
6
7 lowport = int(portrange.split('-')[0])
8 highport = int(portrange.split('-')[1])
9
10 print('scanning host ', target, 'from port', lowport, 'to port', highport)
11
12
13 for port in range(lowport, highport):
14     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
15     status = s.connect_ex((target, port))
16     if(status == 0):
17         print('*** Port', port, '- OPEN ***')
18     else:
19         print('Port', port, '- CLOSED')
20     s.close()
21
```

```
kali@kali: ~/Desktop
File Actions Edit View Help
(kali@kali)~[~/Desktop]
python portscanner.py
Enter the IP address to scan: 192.168.50.101
Enter the port range to scan (es. 21-80): 21-81
scanning host 192.168.50.101 from port 21 to port 81
*** Port 21 - OPEN ***
*** Port 22 - OPEN ***
*** Port 23 - OPEN ***
Port 24 - CLOSED
*** Port 25 - OPEN ***
Port 26 - CLOSED
Port 27 - CLOSED
Port 28 - CLOSED
Port 29 - CLOSED
Port 30 - CLOSED
Port 31 - CLOSED
Port 32 - CLOSED
Port 33 - CLOSED
Port 34 - CLOSED
Port 35 - CLOSED
Port 36 - CLOSED
Port 37 - CLOSED
Port 38 - CLOSED
Port 39 - CLOSED
Port 40 - CLOSED
Port 41 - CLOSED
Port 42 - CLOSED
```

```
File Actions Edit View Help
Port 57 - CLOSED
Port 58 - CLOSED
Port 59 - CLOSED
Port 60 - CLOSED
Port 61 - CLOSED
Port 62 - CLOSED
Port 63 - CLOSED
Port 64 - CLOSED
Port 65 - CLOSED
Port 66 - CLOSED
Port 67 - CLOSED
Port 68 - CLOSED
Port 69 - CLOSED
Port 70 - CLOSED
Port 71 - CLOSED
Port 72 - CLOSED
Port 73 - CLOSED
Port 74 - CLOSED
Port 75 - CLOSED
Port 76 - CLOSED
Port 77 - CLOSED
Port 78 - CLOSED
Port 79 - CLOSED
*** Port 80 - OPEN ***
(kali@kali)~[~/Desktop]
$
```

importa il modulo socket:

```
import socket
```

Qui viene importato il modulo socket, che consente di creare socket e gestire le connessioni di rete.

Input dell'utente:

```
target = input('Inserisci l'indirizzo IP da analizzare: ')
```

```
portrange = input('Inserisci l'intervallo di porte da analizzare (es. 21-80): ')
```

L'utente viene richiesto di inserire l'indirizzo IP di destinazione e l'intervallo di porte da analizzare.

Estrazione delle porte bassa e alta:

```
lowport = int(portrange.split('-')[0]) highport = int(portrange.split('-')[1])
```

L'intervallo di porte inserito viene diviso al trattino e i valori risultanti vengono convertiti in interi. Questi valori rappresentano gli estremi bassi e alti dell'intervallo di porte.

Stampa informazioni sulla scansione:

```
print('Scansione dell'host', target, 'dalla porta', lowport, 'alla porta', highport)
```

Questa riga stampa informazioni sulla scansione, inclusi l'host di destinazione e l'intervallo di porte che verrà analizzato.

Ciclo attraverso le porte:

```
for port in range(lowport, highport):
```

Questo ciclo itera attraverso ciascuna porta nell'intervallo specificato.

Creazione di un socket:

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Viene creato un nuovo socket utilizzando la famiglia di indirizzi IPv4 (**AF_INET**) e il tipo di socket TCP (**SOCK_STREAM**).

Verifica dello stato della porta:

```
status = s.connect_ex((target, port))
```

Il metodo **connect_ex** tenta di connettersi alla coppia (**target, port**) specificata. Se la connessione ha successo, **status** sarà 0; in caso contrario, sarà un codice di errore.

Stampa dello stato della porta:

```
if status == 0: print('*** Porta', port, '- APERTA ***') else: print('Porta', port, '- CHIUSA')
```

A seconda del valore di **status**, viene stampato se la porta è aperta o chiusa.

Chiusura del socket:

```
s.close()
```

Il socket viene chiuso dopo aver verificato lo stato della porta corrente.

Report degli attacchi Brute Force sulla pagina phpmyadmin con evidenza della coppia

```
licenses at https://nmap.org/oem/.
Trying: root -
FAILED: Login attempt with: root _
Trying: root - 123456
FAILED: Login attempt with: root _ 123456
Trying: root - 12345
FAILED: Login attempt with: root _ 12345
Trying: root - 123456789
FAILED: Login attempt with: root _ 123456789
Trying: root - password
FAILED: Login attempt with: root _ password
Trying: root - iloveyou
FAILED: Login attempt with: root _ iloveyou
Trying: root - princess
FAILED: Login attempt with: root _ princess
Trying: root - 12345678
FAILED: Login attempt with: root _ 12345678
Trying: root - 1234567
FAILED: Login attempt with: root _ 1234567
Trying: root - abc123
```

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'root'@'192.168.50.101' WITH GRANT OPTION;
Query OK, 0 rows affected (0.00 sec)

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

mysql> EXIT;
Bye
```

```
mysql> SELECT user, host FROM mysql.user WHERE user = 'root' AND host = 'http://
192.168.50.101/phpMyAdmin/index.php';
+-----+-----+
| user | host |
+-----+-----+
| root | http://192.168.50.101/phpMyAdmin/index.php |
+-----+-----+
```


Report totale che include i risultati trovati e le contromisure

Report di Sicurezza - Analisi delle Porte e Metodi HTTP

Scansione delle Porte:

Porta 21 (FTP): Aperta
Porta 22 (SSH): Aperta
Porta 25 (SMTP): Aperta
Porta 80 (HTTP): Aperta

Analisi dei Metodi HTTP:

PUT: Abilitato (non necessario per l'applicazione web). Deve essere Disabilitato
DELETE: Abilitato (non necessario per l'applicazione web). Deve essere Disabilitato
GET: Abilitato (utilizzato per richiedere dati dal server).
PATCH: Abilitato (potenziale rischio di sicurezza se non necessario). Deve essere Disabilitato
POST: Abilitato (utilizzato per inviare dati al server).
OPTIONS: Abilitato (riduzione della superficie di attacco). Deve essere Disabilitato o limitato
HEAD: Abilitato (utilizzato per richiedere solo l'intestazione della risposta).
TRACE: Abilitato (potenziale rischio di Cross-Site Tracing). Deve essere Disabilitato
Contromisure Consigliate:

Porta 21 (FTP):

Contromisure:

Utilizzare connessioni FTP sicure (FTPS) o considerare l'utilizzo di alternative sicure come SFTP (SSH File Transfer Protocol).
Implementare l'accesso tramite credenziali sicure e monitorare l'attività FTP per rilevare eventuali comportamenti anomali.
Limitare l'accesso FTP solo a indirizzi IP autorizzati.
Crittografare il traffico FTP per proteggere i dati trasmessi.

Porta 22 (SSH):

Contromisure:

Implementare autenticazione a due fattori (2FA).
Disabilitare l'accesso root e limitare gli utenti autorizzati.
Monitorare i log SSH per rilevare tentativi di accesso non autorizzato.
Utilizzare chiavi SSH complesse e regolarmente aggiornate.
Abilitare solo protocolli SSH sicuri (preferibilmente SSHv2).

Porta 25 (SMTP):

Contromisure:

Limitare l'accesso ai server SMTP solo ai server di posta autorizzati.
Implementare autenticazione e cifratura per evitare il rischio di spam o accesso non autorizzato.

Monitorare i log SMTP per rilevare attività sospette.

Configurare SPF (Sender Policy Framework) , DKIM e DMARC per prevenire l'invio di email spoofate.

Crittografare le comunicazioni SMTP per proteggere la privacy e l'integrità delle email.

Porta 80 (HTTP):

Contromisure:

Implementare un Firewall perimetrale per filtrare e monitorare il traffico HTTP.

Configurare un Web Application Firewall (WAF) per proteggere da attacchi web.

Assicurarsi che i metodi HTTP siano configurati in modo sicuro (es. limitare l'uso di PUT, DELETE, TRACE, etc.).

Aggiornare regolarmente il software del server web per correggere vulnerabilità.

Implementare HTTPS per crittografare il traffico tra il client e il server.

Utilizzare certificati SSL/TLS validi e di qualità per garantire una comunicazione sicura.

Altre Contromisure Consigliate:

Il Firewall di Nuova Generazione integrato sia nella DMZ che nella rete interna, offre funzionalità avanzate di sicurezza come l'ispezione profonda dei pacchetti e la protezione contro attacchi DDoS. La sua integrazione contribuisce a fornire una difesa coesa su tutti i livelli della rete. Questo firewall con Proxy reverse integrata . che andrebbe a creare una rete sicura siccome il firewall della prossima generazione funge sia da IDS che da IPS , ma essendo ad un elevato costo lo mettiamo come un consiglio se la azienda desidera questa tipologia di rete .

Monitoraggio del Traffico:

Implementare sistemi di monitoraggio del traffico per rilevare attività anomale e comportamenti sospetti.

Utilizzare sistemi di rilevamento delle intrusioni per analizzare il traffico in tempo reale e identificare potenziali minacce.

Pianificazione delle Patch:

Mantenere il sistema operativo e il software aggiornati con le ultime patch di sicurezza.

Automatizzare il processo di applicazione delle patch per garantire tempestività ed efficacia.

Controllo degli Accessi:

Implementare controlli di accesso rigorosi sia per il Web server che per l'Application server.

Monitorare e registrare gli accessi per rilevare accessi non autorizzati.

Implementare la gestione degli accessi in base al principio del privilegio minimo necessario.

1. Gestione delle Identità e degli Accessi (IAM):

Definizione dei Ruoli: Assegna ruoli specifici agli utenti in base alle loro responsabilità. Ad esempio, amministratori di sistema, utenti standard .

Principio del Minimo Privilegio: Concede agli utenti solo i privilegi di cui hanno bisogno per svolgere il proprio lavoro. Limita l'accesso ai dati e alle risorse.

2. Autenticazione Sicura:

Utilizzo di Password Forti: L'utilizzo di password complesse e la loro regolare rotazione.
Autenticazione Multifattore (MFA/2FA): Richiede un secondo metodo di verifica oltre alle password, aumentando significativamente la sicurezza.

Backup Regolari:

Eseguire backup regolari dei dati critici per garantire la disponibilità e l'integrità dei dati in caso di incidente.

Verificare periodicamente la correttezza dei backup e la possibilità di ripristino.

Formazione e Consapevolezza:

Formare regolarmente gli utenti e gli amministratori di sistema sulle migliori pratiche di sicurezza.

Promuovere la consapevolezza sulla sicurezza informatica attraverso campagne educative e simulazioni di phishing.