Università degli studi di Modena e Reggio Emilia

Dipartimento di Scienze Fisiche, Informatiche e Matematiche

Corso di Laurea in informatica

Titolo: prima riga Seconda riga Terza riga Quarta riga

Relatore:
Luca Ferretti

Candidato:
Fabio Zanichelli

Anno Accademico 2021/2022

Indice

Capitolo 1

Introduzione

1.1 OS Fingerprinting

L'OS fingerprinting consiste nel rilevare da remoto il sistema operativo di un dispositivo analizzandone i pacchetti inviati. Le differenze di implementazione dello stack TCP/IP, infatti, determinano comportamenti diversi che, analizzati, consentono di ottenere informazioni importanti.
Il fingerprinting può essere effettuato in due modalità: attiva e passiva.
Nella prima si analizzano le risposte ricevute in seguito ad alcuni pacchetti inviati; questi ultimi sono appositamente costruiti in modo da massimizzare le informazioni che si possono ottenere dalla risposta. Nella seconda, invece, viene ispezionato il normale traffico del dispositivo target; si tratta quindi di una tecnica meno invasiva e che si espone meno al rischio di essere scoperti.

1.2 Stack TCP/IP

Per effettuare comunicazioni tramite internet, vi è il bisogno che tutti i dispositivi connessi rispettino determinati meccanismi; questo si rende necessario a causa dell'elevata eterogeneità derivata da hardware e software differenti. Questi meccanismi, che prendono il nome di *protocolli*, sono strutturati secondo diversi layer (livelli) formando lo stack TCP/IP.

Sebbene l'idea originale prevedesse un modello composto da sette livelli, de facto lo schema attualmente in uso ne prevede solamente quattro. Nonostante ciò, nella terminologia informatica la numerazione è rimasta quella precedente

Livello 7	Applicativo
Livello 4	Trasporto
Livello 3	Rete
Livello 2	Fisico

Tabella 1.1: Livelli dello stack TCP/IP

1.2.1 Funzionamento dello stack TCP/IP

Si consideri l'esempio dell'invio di una lettera tramite il servizio di poste. La procedura da seguire è la seguente:

1. Il mittente scrive il contenuto del messaggio su un foglio e successivamente lo inserisce nella busta.
2. Il mittente scrive, nella busta, il nominativo del destinatario.
3. Il mittente scrive il CAP e l'indirizzo del ricevente.
4. Il mittente, dopo aver incollato il francobollo, consegna la busta al servizio di poste.

Si noti che la sequenza di eventi che si verifica per la ricezione della lettera riflette le operazioni effettuate per l'invio, ma in ordine inverso:

1. Viene controllata la presenza del francobollo.
2. Il postino legge l'indirizzo del destinatario e consegna la lettera.
3. Verrà letto il nominativo per individuare a quale inquilino è diretta.
4. Il destinatario apre la busta e legge il contenuto del messaggio.

La sequenza di eventi descritta rappresenta ciò che avviene quando si comunica tramite internet. Ad ogni livello dello stack, infatti, vi è un protocollo che aggiunge una sua intestazione (*header*) al pacchetto del mittente, ad iniziare dal livello più alto disponibile per quel dispositivo e il ricevente le valuterà in ordine inverso. Ad ogni layer il protocollo utilizzato può essere differente; fondamentale, ovviamente, che questo sia supportato da entrambi gli attori della conversazione.
La procedura dell'inserimento degli header prende il nome di *incapsulamento*.

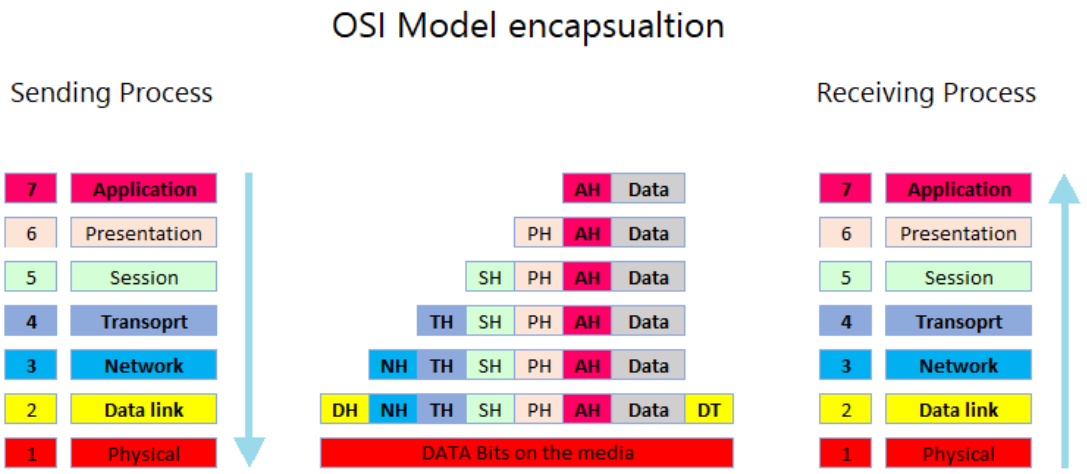

Figura 1.1: METTERE RIFERIMENTO IMMAGINE: http://infodoc.altervista.org/sistemi-e-reti/incapsulamento/

1.2.2 Header per il fingerprinting

Gli header aggiunti ad ogni livello sono formati da vari campi contententi informazioni utili per la comunicazione, e il valore che questi assumono in determinate situazioni è dipendente dal sistema operativo che si sta utilizzando.

Si prenda ad esempio l'header del protocollo TCP:

Il campo *option* permette di segnalare al ricevente l'uso di alcune opzioni di comunicazione, come il window scaling; il loro supporto e l'effettivo utilizzo, essendo queste

TCP Header				
Bits	0-15			16-31
0	Source port			Destination port
32	Sequence number			
64	Acknowledgment number			
96	Offset	Reserved	Flags	Window size
128	Checksum			Urgent pointer
160	Options			

Figura 1.2: METTERE RIFERIMENTO IMMAGINE: https://www.ionos.it/digitalguide/server/know-how/presentazione-tcp/

facoltative e quindi peculiari di specifici sistemi operativi, rivestono quindi particolare importanza ai fini del fingerprinting. Esempi analoghi si possono trovare nei protocolli di tutti i livelli dello stack, e l'unione delle informazioni acquisite consente di poter individuare con una discreta precisione il sistema operativo del dispositivo target.

1.3 Strumenti per il fingerprinting

Esistono numerosi tool per effettuare il fingerprinting, che eseguono anche attività correlate e fondamentali per quest'ultimo come ad esempio il port scanning. Nel corso di questo documento si farà riferimento a due strumenti:

- Nmap, per il fingerprinting attivo
- p0f, per il fingerprinting passivo

Capitolo 2

Esperienza tirocinio sul fingerprinting

2.1 Obiettivo

L'obiettivo dell'esperienza consisteva nell'analizzare le differenze che portavano all'individuazione del sistema operativo, e successivamente utilizzare le conoscenze acquisite per modificare dei parametri del sistema operativo in modo da riuscire ad ingannare i principali strumenti per il fingereprinting. I risultati ottenuti da quest'ultimi, quindi, dovevano essere errati e portare all'individuazione di un sistema operativo differente rispetto a quello realmente in uso.
La procedura è stata effettuata su server HTTP, ovvero con pacchetti di dati non cifrati; si è successivamente cercato di effettuare un fingerprinting sull'handshake TLS, ovvero sullo scambio di messaggi che precede una comunicazione cifrata.

2.2 Strumenti e sistemi operativi utilizzati

Per la realizzazione dell'obiettivo sono stati utilizzati due differenti sistemi operativi: Windows 11 e Kali (una distribuzione Linux basata su Debian). La motivazione risiede nel fatto che Windows sia il sistema più diffuso al mondo e quindi un risultato che individui quello come sistema operativo risulti plausibile agli occhi di chi vuole effettuare il fingerptinging.

I tool utilizzati, oltre ai già citati Nmap e p0f, sono stati i seguenti:

- **Wireshark**, per l'analisi dei pacchetti
- **Server Apache** installati su entrambi i sistemi operativi per simulare le risposte
- **Scapy**, una libreria Python in grado di inviare specifici pacchetti modificabili in ogni campo, e di mostrare i quelli ricevuti
- **nftables**, per la parte successiva all'analisi

Capitolo 3

Attività di analisi

L'attività di analisi iniziale, ovvero senza l'ausilio di librerie specifiche per l'invio di determinati pacchetti, è stata effettuata installando un Server Apache sia su Windows 11 che su sistema operativo Kali. Si è quindi proceduto ad inviare richieste ai server tramite browser web e all'analisi dei pacchetti ricevuti in risposta con l'utilizzo di Wireshark. Successivamente sono state analizzate anche le risposte a determinati pacchetti inviati con Scapy. Di seguito sono riportate le differenze più importanti rilevate durante l'intera attività di analisi.

3.1 Livello 3: protocolli IP e ICMP

È stata effettuata in primis l'analisi del TTL, essendo un valore estremamente semplice da analizzare. Questo campo contiene un numero intero che viene decrementato ad ogni *hop*, cioè ad ogni router che incontra nel percorso verso l'host ricevente, e viene eliminato dalla rete quando questo raggiunge lo 0. Il protocollo, però, non impone nessun valore di partenza, lasciandolo decidere al sistema operativo del mittente; l'unico limite è rappresentato dagli 8 bit riservati a quel campo (ovvero ad un valore massimo di 255). L'analisi ha portato ad evidenziare la seguente differenza riguardo al TTL:

Windows 11	128
Kali	7

Tabella 3.1: TTL iniziale

È inoltre stata effettuata l'analisi del protocollo ICMP, e in particolare è stata notata una differenza nel campo *code* in caso di risposta a determinati pacchetti inviati utilizzando la libreria di Python scapy.

METTERE SCREEN DEL MINISCRIPT IN PYTHON.

Questo comando invia un pacchetto ICMP con il campo *code* impostato ad un numero diverso da zero: in questo specifico contesto, Windows 11 risponde inviando un pacchetto in cui *code*=0, mentre Kali copia il valore che ha ricevuto nella richiesta.

Windows 11	0
Kali	Stesso valore inviato nella richiesta

Tabella 3.2: Campo code quando nella richiesta è diverso da 0

3.2 Livello 4: analisi protocollo TCP

Il livello 4 dello stack è formato da due protocolli: Transmission Control Protocol (TCP) e User Datagram Protocol (UDP). Entrambi forniscono informazioni utili per il fingerprinting, ma il TCP possiede un numero maggiore di campi e di opzioni utilizzabili, pertanto è stata data priorità all'analisi di quest'ultimo.

Analizzando l'handshake TCP, si può notare una discrepanza nell'utilizzo del Window Scale; quest'opzione consente di aumentare la Window Size oltre il valore che si otterrebbe settando tutti i bit di quel campo a 1. L'utilizzo di questa opzione viene concordato nei segmenti SYN e SYN+ACK dell'handshake e il suo valore non viene più modificato per il resto della connessione. Confrontando i valori ottenuti da Windows 11 e Kali, si ottiene il seguente risultato:

Windows 11	8
Kali	7

Tabella 3.3: Window Scaling Factor

```
1  from scapy.all import *
2
3  pkt=sr1(IP(dst='192.168.63.1')/TCP(dport=80, flags='SCE'))
4  pkt.show()
```

Figura 3.1: Comando python per l'invio del pacchetto specifico

Continuando l'analisi si possono notare ulteriori differenze riguardanti il flag sulla notifica esplicita di congestione (ECN). Se infatti si invia un pacchetto simulando l'inizio di un handshake (SYN) con i flag sulla congestione attivi, si otterranno risultati differenti tra Windows 11 e Kali nel pacchetto di risposta (che sarà ovviamente un SYN+ACK, secondo passo dell'handshake):

Windows 11	0
Kali	1

Tabella 3.4: ECN in risposta a specifico pacchetto

3.3 Livello 7: analisi protocollo HTTP

Sebbene HTTP sia un protocollo a livello applicativo, esso consente ugualmente di ricavare alcune informazioni utili per l'OS fingerprinting: il campo *user-agent*, infatti, contiene informazioni esplicite riguardanti il browser che si sta utilizzando e il sistema operativo in utilizzo. Questo tipo di situazione prende il nome di *banner grabbing*.

A questo livello dello stack, si può inoltre tentare un fingerprinting che vada oltre l'individuazione del sistema operativo, ponendo come obiettivo quello di indovinare il tipo di applicativo in uso. Questo è reso possibile dal fatto che HTTP è un protocollo di tipo testuale, pertanto non vi sono campi prefissati per ogni funzione. Ogni header contiene varie coppie, secondo lo schema:

chiave:valore

Questo, a differenza dei protocolli analizzati precedentemente, permette un diverso ordine con le quali le coppie vengono elencate, essendo questo ininfluente per una cor-

retta comunicazione. Analizzare l'ordinamento è molto importante se si vuole tentare di individuare, ad esempio, il tipo di server che sta rispondendo.