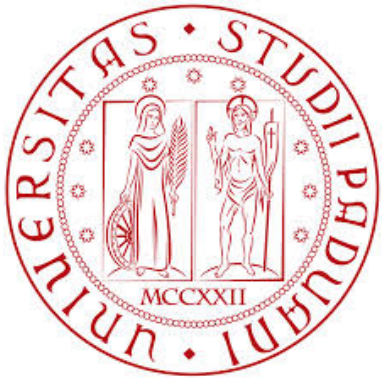
Università degli studi di Padova

Dipartimento di ingegneria dell'informazione



Corso di laurea Magistrale in Ingegneria Informatica

Intercettazione del traffico TCP su reti mobili: tecnica, implementazione e statistiche

Laureando: *Giorgio Zambon*
Matricola: *1082242*

Relatore: *Chia.mo Prof. Ing. Nicola Zingirian*

Anno Accademico 2015/2016

Indice

Capitolo 1

Introduzione

In questo primo capitolo viene data una panoramica dell'idea che ha portato ad indagare i temi trattati dalla seguente tesi di laurea. Si descrive in seguito l'obiettivo che si perseguito nella fase sperimentale.

1.1 Premessa

La seguente tesi di laurea si propone di essere una continuazione del lavoro prodotto per il conseguimento del diploma di laurea triennale in Ingegneria Informatica.
L'evento scatenante di tutto il lavoro svolto è stata l'osservazione di pacchetti di dati ricevuti al momento della connessione di un modem alla rete mobile.
Nel momento in cui avviene l'allacciamento alla rete mobile il modem non ha ancora avuto il tempo materiale di generare traffico, è sorto quindi il dubbio che si trattasse di pacchetti non intenzionalmente indirizzati al dispositivo che aveva appena effettuato l'accesso.
Al fine di dare una risposta al fenomeno osservato, è stato quindi ipotizzato che il traffico ricevuto fosse costituito da pacchetti vacanti, i quali costituiscono il residuo di una connessione precedentemente interrotta. Al dispositivo che dialogava nella connessione interrotta, era associato l'indirizzo IP che poi, essendosi liberato, è passato al modem dov'è stato osservato il fenomeno.
Il lavoro condotto in tale occasione ha validato il fenomeno e la supposizione fatta presentandone un'analisi qualitativa e quantitativa.
In fase conclusiva ci si è soffermati sull'impatto che questo fenomeno ha in termini economici[1] e soprattutto di sicurezza delle informazioni.
Quanto descritto ha ispirato il lavoro svolto in occasione del conseguimento

[1] Nel lavoro si è analizzata la connessione alla rete mobile per messo di SIM M2M nelle quali il traffico dati ha prezzi vantaggiosi solo nel caso dello scambio sporadico di piccole moli di dati

del diploma di laurea magistrale in Ingegneria Informatica. In questa occasione infatti, si è espanso il software prodotto rendendolo non solo capace di ricevere i pacchetti delle connessioni residue, ma in grado anche di rispondere a tali pacchetti con l'obbiettivo di continuare le connessioni, recuperando il maggior numero di dati possibile limitatamente ai pacchetti di tipo TCP[2].

1.2 Obiettivo

Come precedentemente accennato, l'obbiettivo dell'esperimento che verrà descritto è quello di espandere il software creato per ricevere passivamente i pacchetti appartenenti alle connessioni residue in fase di allacciamento alla rete mobile.
L'espansione prevede di rendere il software capace di far avanzare le connessioni interrotte rispondendo ai pacchetti residui che vengono ricevuti. Le differenti connessioni captate devono essere fatte avanzare parallelamente forgiando e indirizzando di volta in volta il pacchetto di risposta al destinatario corretto e facendo in modo che il pacchetto risulti valido.
Dai pacchetti ricetrasmessi si prelevano informazioni e dati posizionando quest'ultimi in maniera contigua così da rendere visualizzabile il contenuto della connessione[3].

[2]Il limite è imposto dal fatto che solamente il tipo TCP del livello 4 ISO-OSI richiede la conferma di ricezione del pacchetto precedente prima dell'invio del successivo, essendo protocollo di comunicazione senza perdita e senza errori nei pacchetti. In tal modo è possibile effettivamente rispondere ad un pacchetto ricevuto, per ottenere i successivi. Gli altri protocolli o non sono utilizzati per il trasposto di dati o, come nel caso dell'UDP, non garantiscono l'affidabilità del dato ricevuto in quanto puntano alla velocità di avanzamento della connessione accettando eventuali corruzioni o perdite dei dati

[3]Il contenuto in questione è per il più delle volte criptato e quindi non comprensibile se visualizzato in un editor di testo. In alcuni casi invece il contenuto viene trasmesso in chiaro (è questo il caso delle connessioni sul protocollo HTTP) e a connessione terminata è quindi possibile visualizzare il contenuto informativo, il quale risulta comprensibile.

Capitolo 2

Nozioni di base

L'intento di questo capitolo è quello di fornire una conoscenza generale sia delle strutture utilizzate all'interno del software che verrà presentato nel capitolo successivo, sia del funzionamento della rete sulla quale viaggiano i pacchetti, concentrando l'attenzione sul protocollo TCP.

2.1 La rete mobile

Qualunque dispositivo capace di generare traffico dati che desideri allacciarsi alla rete mobile, necessita di un indirizzo IP per essere riconoscibile e rintracciabile all'interno di quest'ultima.
L'indirizzo IP viene assegnato in seguito alla richiesta di accesso alla rete effettuata dal dispositivo verso appositi server. L'indirizzo IP specifico viene scelto tra quelli in possesso del particolare operatore telefonico che risultano liberi, ovvero momentaneamente non assegnati a nessun dispositivo.
Minore è il numero di indirizzi IP a disposizione del particolare operatore, maggiore è la frequenza di riciclo di questi ultimi e maggiore è quindi la probabilità che allacciandosi alla rete si ottenga un indirizzo IP che poco prima era associato ad un altro dispositivo.
Questa categoria di indirizzi IP è di grande interesse ai fini dell'esperimento, in quanto la possibilità di ricevere pacchetti appartenenti a connessioni residue da indirizzi IP riassociati poco dopo il loro rilascio, è molto più alta che negli altri casi. Quando uno dei due interlocutori di una connessione non riceve più risposte infatti, entrano in gioco dei meccanismi di timeout che chiudono la comunicazione rendendo impossibile una sua continuazione.

2.2 Firewall

Quando un dispositivo mobile dotato di firewall (ad esempio computer e telefoni cellulari) si connette alla rete mobile, alla ricezione dei pacchetti effettua una valutazione sulla validità o meno degli stessi. Se i pacchetti

vengono classificati come non appartenenti ad alcuna connessione iniziata dal dispositivo, essi vengono scartati ed un pacchetto di reset viene inviato automaticamente al mittente di ciò che è stato ricevuto.
L'aspetto descritto rappresenta un effetto collaterale non desiderato ai fini della buona riuscita dell'esperimento. Si è reso necessario pertanto impostare il firewall in maniera tale che bloccasse tutte le connessioni sia in entrata che in uscita. Il firewall della scheda di rete è stato bypassato in entrambe le direzioni di comunicazione attraverso il modem GPRS esterno utilizzato per la connessione alla rete mobile.
In questo modo è stato quindi possibile ricevere pacchetti estranei al dispositivo senza che venisse automaticamente generato il pacchetto di reset e forgiare la risposta a tali pacchetti.

2.3 Composizione di un pacchetto

In generale un pacchetto di dati che viaggia sulla rete di compone di parti differenti, a loro volta organizzate in campi di lunghezza predefinita e significato che varia in base al valore contenuto da ciascuno di essi.
Una prima suddivisione del pacchetto consente di individuare uno o più header ed una porzione di dati che può anche non essere presente[1]. All'interno del pacchetto, header consecutivi appartengono a livelli ISO-OSI diversi partendo dai più bassi fino ad arrivare ai più alti, inoltre ciascun header può avere composizioni diverse anche nello stesso livello.
In questo esperimento il livello 2 ISO-OSI segue il protocollo ethernet e viene elaborato dal modem, appare quindi trasparente al software. Il primo header di interesse ai fini dell'esperimento e che viene elaborato dal software è il livello 3 ISO-OSI del quale interessa l'header di tipo IP[2]. Al livello 4 ISO-OSI invece, è di interesse l'header che segue il protocollo TCP.
Di seguito vengono analizzati in dettaglio i campi in cui gli header IP e TCP sono suddivisi.

2.3.1 Header IP[3]

L'header del protocollo IP visibile nell'immagine 2.1, appartiene al livello 3 ISO-OSI e costituisce l'inizio dello stream di dati, sia in ricezione che in trasmissione, dei pacchetti elaborati durante l'esperimento.

[1] Esistono protocolli utilizzati solo a scopo di controllo, di conseguenza i pacchetti che seguono tali protocolli non trasmettono dati. Un'altra tipologia di pacchetti privi di dati sono i cosiddetti "keep alive", i quali vengono scambiati tra due dispositivi che stanno comunicando al fine di verificare la reciproca connessione.

[2] Un pacchetto di tipo IP è chiamato anche datagramma

[3] La descrizione completa della struttura e del funzionamento del protocollo IP è definita nell'RFC 791.

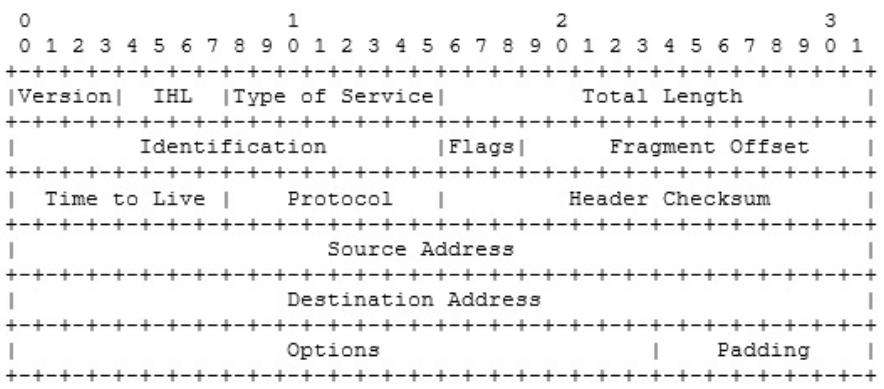


Figura 2.1: Header del protocollo IP del livello 3 ISO-OSI.

L'header IP si compone di svariati campi che vengono di seguito descritti in dettaglio:

- "Version" [4 bit]: indica la versione del pacchetto IP. Nel caso del protocollo IPv4 e il campo assume il valore 4;
- "IHL"[4 bit]: acronimo di "Internet Header Lenght", rappresenta la lunghezza dell'header IP in word da 32 bit. La lunghezza è direttamente correlata alla presenza o meno dei campi opzionali alla fine dell'header. Il valore minimo assumibile da questo campo è 5 che corrisponde all'header privo di campi opzionali;
- "Type of Service" [8 bit]: nella formulazione originale dell'RFC di riferimento questo campo era suddiviso in porzioni, le quali permettevano al mittente del pacchetto di definire il modo e la precedenza con cui il destinatario doveva trattare il pacchetto. Attualmente tali bit vengono utilizzati nelle nuove tecnologie basate sullo streaming in tempo reale come ad esempio il VoIP utilizzato per lo scambio dei dati vocali;
- "Total Lenght" [16 bit]: indica la dimensione in byte del datagramma comprendendo sia gli header che i dati. Segue da quanto detto per il campo IHL che questo campo può avere valore minimo di 20. Tale valore rappresenta un datagramma dov'è presente il solo header IP privo di campi opzionali e di dati;
- "Identification" [16 bit]: viene utilizzato per distinguere in modo univoco il frammento di un datagramma nel caso in cui il datagramma sia stato spezzato in più parti all'origine;
- "Flags" [3 bit]: questi bit vengono utilizzati per il controllo del protocollo e della frammentazione del datagramma. Il primo bit è riservato

ed è impostato sempre a 0. Il secondo, se impostato a 1 indica che il datagramma non può essere frammentato[4], viceversa impostato a 0 indica che la frammentazione è permessa. Il terzo e ultimo bit indica, se impostato a 1, che ci sono frammenti del datagramma successivi da ricevere; viceversa impostato a 0 indica che si tratta dell'ultimo frammento o dell'unico frammento nel caso in cui il datagramma non sia stato partizionato;

- "Fragment Offset" [13 bit]: indica l'offset di un particolare frammento al fine di poter ricostruire la corretta successione di frammenti. Il conteggio del valore è effettuato prendendo blocchi di 8 bit dal campo dati e rappresenta la posizione successiva all'ultimo byte di dati ricevuto rispetto al datagramma originale (il primo frammento ha valore 0 in questo campo);

- "Time to Live" [8 bit]: indica il tempo di vita di un pacchetto e rappresenta un modo per evitare la persistenza infinita nella rete di un pacchetto che non può essere consegnato. Tale campo è un misuratore dei rimbalzi del pacchetto nei nodi della rete il cui valore viene decrementato di 1 ogni volta che il pacchetto transita attraverso un nodo intermedio, viene infine scartato solo al raggiungimento del valore 0;

- "Protocol" [8 bit]: indica il codice associato al protocollo utilizzato nel campo dati del pacchetto IP. Valori degni di nota sono ad esempio il 6 per il protocollo TCP e il 17 per il protocollo UDP;

- "Header Checksum" [16 bit]: tramite il valore contenuto in questo campo si determina l'integrità perlomeno dell'header IP del pacchetto. Il valore contenuto in questo campo è costituito dal complemento della somma in complemento a uno di gruppi da 16 bit alla volta appartenenti all'header IP[5]. Durante la compilazione dell'header il campo viene impostato a 0, viene calcolato e memorizzato solamente quando tutti i valori dei campi sono stati inseriti. La verifica dell'integrità del campo data è deputata ai livelli superiori;

[4] Se il pacchetto non può essere inviato dall'host mittente senza essere frammentato viene semplicemente scartato. Questa funzionalità è sfruttata per conoscere le capacità comunicative di un host.

[5] L'effettivo procedimento del calcolo del checksum si effettua innanzitutto prendendo porzioni successive di 16 bit o 4 cifre esadecimali a seconda della base scelta per il calcolo e sommandole tra di loro, effettuata la somma, se il numero di cifre binarie che la compongono supera il valore limite di 16, significa che c'è un riporto; il riporto viene tagliato e sommato al valore rimanente ripetendo l'operazione nel caso in cui la nuova somma generi ancora una cifra con più di 16 bit; ottenuta una cifra di soli 16 bit, il valore scritto in binario viene complementato. Il controllo di integrità avviene sommando il valore di checksum calcolato all'arrivo del pacchetto con quello già presente nel campo checksum dello stesso, se la somma da come risultato un insieme di 16 uno l'header IP è integro, in caso contrario non lo è e il pacchetto viene scartato

- "Source Address" [32 bit]: in questo campo vengono memorizzate in successione le quattro cifre che compongono l'indirizzo IP del mittente del pacchetto;

- "Destination Address" [32 bit]: in questo campo vengono memorizzate in successione le quattro cifre che compongono l'indirizzo IP del destinatario del pacchetto;

- "Options"/"Padding"/"Data" [numero variabile di bit]: questi tre campi compaiono nella successione elencata in coda all'header IP e sono tutti e tre facoltativi. Una loro assenza infatti è consentita dai valori assumibili dai campi dell'header. I campi "Options" se presenti possono essere uno o più di uno e a causa dell'arbitrarietà della loro dimensione, si sfrutta se necessario il campo "Padding" per completare i 32 bit. Il campo "Padding" è composto da soli 0 e il suo unico scopo è quello descritto. Il campo "Data" se presente, contiene in generale un header di livello superiore seguito da ulteriori dati anch'essi facoltativi.

2.3.2 Header TCP[6]

L'header TCP è un header di livello 4 ISO-OSI e nello stream di dati segue direttamente l'header IP nei pacchetti appartenenti a tale protocollo.

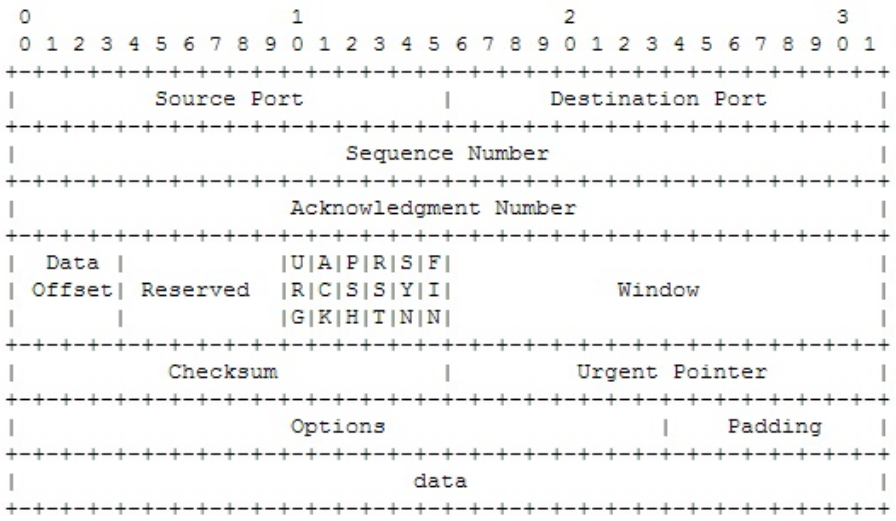


Figura 2.2: Header del protocollo TCP del livello 4 ISO-OSI.

[6]La descrizione completa della struttura e del funzionamento del protocollo TCP è definita nell'RFC 793.

L'header TCP come è possibile osservare nell'immagine 2.2 si compone di svariati campi che vengono di seguito descritti in dettaglio:

- "Source port" [16 bit]: indica il valore della porta che l'host mittente sta utilizzando per inviare il pacchetto TCP;
- "Destination port" [16 bit]: indica il valore della porta alla quale l'host destinatario riceverà il pacchetto TCP;
- "Sequence number" [32 bit]: indica lo scostamento espresso in byte dell'inizio del segmento TCP all'interno del flusso completo, partendo da un valore iniziale negoziato in fase di apertura della connessione;
- "Acknoledgement number" [32 bit]: questo campo viene considerato solamente quando il flag ACK che verrà discusso in seguito è fissato a 1. Il valore di questo campo rappresenta la conferma della ricezione di una porzione di dati indicando il valore del prossimo "Sequence number" che l'host mittente del segmento TCP si aspetta di ricevere;
- "Data offset" [4 bit]: indica la lunghezza dell'header TCP in dword da 32 bit. Tale lunghezza può variare da 5 a 15 dword in base alla lunghezza del campo facoltativo "Option";
- "Reserved" [4 bit]: questo campo contiene dei bit che non sono utilizzati e che quindi vanno settati a zero. La presenza di questi bit è predisposta per eventuali usi futuri. Nella prima formulazione del protocollo TCP questo campo conteneva 6 bit, due dei quali sono stati assegnati nel corso degli anni;
- "Flags" [8 bit]: in questo campo sono presenti i bit di controllo del protocollo. Ognuno dei bit svolge un compito specifico a seconda di come è settato. Rispetto alla formulazione originale del protocollo TCP, a questo campo sono stati aggiunti 2 bit (presi dal campo "Reserved") per consentire nuove funzionalità. In ordine i bit significano:
  - "CWR[7]": se settato a 1 indica che l'host sorgente ha ricevuto un segmento TCP con il flag "ECE" settato a 1;
  - "ECE[8]": se settato a 1 indica che l'host supporta ECN nella fase di negoziazione;

[7]Questo flag è stato introdotto in un momento successivo alla prima formulazione del protocollo TCP e il suo utilizzo è specificato nell'RFC 3168. La sigla è l'acronimo di "Congestion Window Reduced". L'effetto prodotto dal settaggio a 1 di questo flag è la riduzione della dimensione della "Window size". Tale riduzione viene solitamente invocata in presenza di congestione e permette un recupero più veloce dei pacchetti persi in quanto meno pesanti.

[8]Questo flag è stato introdotto in un momento successivo alla prima formulazione del protocollo TCP e il suo utilizzo è specificato nell'RFC 3168. La sigla è l'acronimo di "ECN", che a sua volta è acronimo di "Explicit Congestion Notification", ed "Echo". Il

- "URG": se settato a 1 indica che nel flusso sono presenti dati urgenti all'offset indicato nel campo "Urgent Pointer". Quest'ultimo campo indica la fine dei dati urgenti;
- "ACK": se settato a 1 indica che il campo "Acknoledgement number" è valido;
- "PSH": se settato a 1 indica che i dati non devono essere bufferizzati ma passati direttamente ai livelli superiori;
- "RST": se settato a 1 indica che la connessione non è valida. Questo bit viene utilizzato sia in caso di errore grave, sia assieme al flag "ACK" per forzare la chiusura di una connessione;
- "SYN": settato a 1 indica che l'host mittente del pacchetto vuole aprire una connessione TCP con il destinatario. Il mittente che invia un pacchetto con il flag "SYN" settato a 1, attende dal destinatario un pacchetto con i flag "SYN" e "ACK" settati a 1. La fase descritta è detta di negoziazione[9] ed è in questa fase che si determina il "Sequence number" iniziale al quale si accennava precedentemente;
- "FIN": se settato a 1 indica che il mittente del pacchetto vuole chiudere la connessione TCP aperta con il destinatario. Una volta inviato il pacchetto, il canale risulta chiuso a metà in quanto il mittente non può più inviare dati sul canale ma il destinatario si. Quando anche l'host destinatario invia un pacchetto nel quale il flag "FIN" è settato a 1, la conseguente risposta con i flag "FIN" e "ACK" settati a 1 determina la chiusura definitiva della connessione tra i due host.

suo settaggio avviene nella fase di negoziazione: se entrambi gli host inviano un pacchetto con questo flag settato a 1 significa che durante la connessione si accetta la possibilità di settare a 1 il flag CWR in caso di congestione al fine di ridurre la "Window size" senza che i pacchetti vengano scartati.

[9] La fase di negoziazione è anche chiamata "3-way handshake" e comprende appunto tre fasi nelle quali i due host negoziano i parametri della connessione:

* l'host A invia un segmento SYN all'host B. Il flag "SYN" è impostato a 1 e il campo "Sequence number" contiene il valore x che specifica l'"Initial Sequence Number" di A;
* B invia un segmento "SYN"/"ACK" ad A. I flag "SYN" e "ACK" sono impostati a 1, il campo "Sequence number" contiene il valore y che specifica l'"Initial Sequence Number" di B e il campo "Acknowledgment number" contiene il valore x+1 confermando la ricezione del ISN di A;
* A invia un segmento ACK a B - il flag ACK è impostato a 1 e il campo "Acknowledgment number" contiene il valore y+1 confermando la ricezione del "ISN" di B.

- "Window size" [16 bit]: indica il numero di byte che l'host mittente è in grado di riceve a partire da quello specificato dall'"acknoledgement number";
- "Checksum" [16 bit]: la funzione ed il processo di calcolo di questo campo è uguale a quello descritto nella sottosezione 2.3.1 relativa al checksum presente nell'header IP. I campi utilizzati per il calcolo del checksum in questo caso comprendono: l'header TCP, i dati del pacchetto TCP e i campi inseriti in uno pseudoheader, il quale verrà descritto successivamente nella sezione 2.4;
- "Urgent pointer" [16 bit]: è un puntatore che punta all'ultimo dato urgente nel campo data. Tale campo ha valore solamente quando il flag "URG" è settato a uno. Tutti i byte del campo data che precedono quello puntato da questo campo devono essere considerati urgenti;
- "Options"/"Padding"/"Data" [numero variabile di bit]: come nel caso dell'header IP, anche in questo caso i tre campi sono facoltativi. Per quanto riguarda la compilazione dei primi due si fa riferimento alla sottosezione 2.3.1, il campo "Data" invece fa eccezione in quanto contiene dati anziché i campi di un header di livello superiore.

2.4 Pseudoheader

Lo pseudoheader è un insieme di campi che esistono contigui solo in forma concettuale, in una struttura come quella rappresentata nell'immagine 2.3. Essi sono infatti utilizzati per il solo calcolo del valore del campo "Checksum" nell'header TCP e non compaiono assieme in nessuna struttura dati.

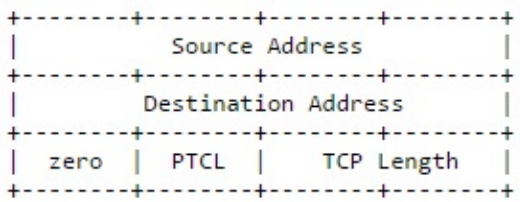


Figura 2.3: Campi dello pseudoheader per il calcolo del checksum del protocollo TCP.

In dettaglio, i campi che compongono lo pseudoheader hanno lunghezza totale pari a 12 ottetti e sono:

- i campi "Source Address" e "Destination Address" dell'header IP;

- un ottetto composto da soli 0;
- il campo "Protocol" dell'header IP;
- "TCP Lenght" è un valore che rappresenta la lunghezza in ottetti dell'header TCP sommata a quella del campo "Data" del TCP, vengono esclusi dal calcolo i 12 ottetti che costituiscono lo pseudoheader a prova della sua esistenza solo in forma concettuale.

2.5 Man in the middle (MITM)

In ambito di sicurezza informatica, con la locuzione "man in the middle" si intende un tipo di attacco nel quale si verifica un'intrusione di un terzo attore durante la comunicazione tra due host nella rete. Sfruttando questa circostanza, un attaccante può essere in grado di leggere, modificare, reindirizzare il contenuto della conversazione. Quando nessuna delle due parti facenti parte della comunicazione riesce a percepire che il proprio collegamento stato compromesso dalla presenza del terzo attore, l'attacco è da considerarsi efficace.

Capitolo 3

Fasi di realizzazione dell'esperimento

In questo capitolo vengono formalizzati i fenomeni osservati nella fase precedente all'implementazione dell'esperimento. Tali osservazioni hanno permesso di ideare una procedura di sfruttamento dei fenomeni, nonché l'individuazione delle fasi che ne hanno costituito il risvolto pratico.

3.1 L'osservazione del fenomeno

Riprendendo quando accennato all'interno del capitolo introduttivo, il fenomeno che ha portato alla scrittura di questo elaborato è stato inizialmente osservato al momento della connessione di un modem alla rete mobile.
In una fase immediatamente successiva a quella di negoziazione dell'indirizzo IP e degli altri parametri di connessione tra il modem e il server al quale questo di connette, si è notato del traffico inaspettato provenire in ingresso al modem.
Il traffico ricevuto è da considerarsi anomalo, in quanto dopo aver superato con successo la fase di negoziazione, il client non ha ancora generato alcun tipico di traffico in uscita e non deve pertanto riceverne in ingresso.
Nel caso in cui un dispositivo riceva del traffico che non era ad esso destinato, le normali implementazioni protocolli di rete quali ad esempio il TCP, provvedono automaticamente a far terminare la comunicazione inviando un opportuno pacchetto di reset al mittente.

3.2 Formulazione della procedura di sfruttamento del fenomeno osservato

In seguito ad una prima fase di verifica nella quale si è constatato che il traffico ricevuto dopo l'ottenimento dell'indirizzo IP dal dispositivo non gli

apparteneva, si è analizzata la situazione ipotizzando dei metodi di sfruttamento della stessa.
L'analisi del fenomeno ha permesso di definirne le cause, le quali sono da imputare alla scarsità di indirizzi IP disponibili rispetto al numero maggiore di dispositivi che li utilizzano.
Nel momento in cui un dispositivo si interfaccia sulla rete mobile, gli viene assegnato un indirizzo IP tra quelli che risultano inutilizzati in quel momento.
In alcune situazioni, le quali sono alla base dell'esperimento, capita che l'assegnazione di un indirizzo IP avvenga pochi istanti dopo che quest'ultimo si è liberato. Quando ciò si verifica, è possibile che tale indirizzo IP abbia ancora associato del traffico in entrata, il quale viene ricevuto dal nuovo dispositivo assegnatario dell'IP.
La maggioranza dei dispositivi che sono in grado di accedere alla rete mobile sono in grado di terminare automaticamente le connessioni non volute sfruttando i protocolli in essi implementati e forgiando appositi pacchetti di reset da inoltrare al mittente del traffico indesiderato. Mediante l'utilizzo di un modem privo di interfaccia software in grado di inviare automaticamente pacchetti di reset, è stato possibile meditare un modo per sfruttare i pacchetti in ingresso al fine di continuare le connessioni ricevute.
Il focus dell'operazione di risposta è stato incentrato sulle connessioni di tipo TCP, in quanto tale protocollo di trasferimento prevede l'invio di pacchetti di conferma di avvenuta ricezione da parte del client. In accordo con le caratteristiche del protocollo TCP, il traffico in ingresso è stato quindi opportunamente filtrato. Sfruttando in un secondo momento gli RFC relativi a IP e TCP, i campi dei pacchetti ricevuti sono stati opportunamente manipolati al fine di forgiare i pacchetti di risposta, i quali sono stati di volta in volta inviati.
Sì è così potuto notare che, in seguito all'invio dei pacchetti di risposta, l'interlocutore continuava le connessioni nei casi in cui questo risultava possibile[1].

3.3 Ottimizzazione della procedura

Una volta verificata la reale possibilità di continuare connessioni forgiando un'adeguata risposta ai pacchetti indesiderati di volta in volta ricevuti, il focus dell'esperimento si è spostato nell'ideare un metodo di sfruttamento di tale fenomeno. L'aspetto fondamentale sul quale si è posta l'attenzione

[1]Non tutte le connessioni hanno generato traffico in ingresso dopo l'avvenuta connessione. Di quelle che hanno generato traffico inoltre, una parte dei pacchetti ricevuti presentava il flag di "FIN" attivo ad indicare che il pacchetto ricevuto costituiva la fine della connessione.

è stato quello di massimizzare il numero di differenti indirizzi IP ottenuti in fase di negoziazione. Per fare ciò, è stato necessario definire degli spazi temporali all'interno dei quali la connessione potesse fare il suo corso e al termine dei quali questa potesse essere riavviata in maniera ciclica, dopo aver completato le connessioni attive individuate.

3.4 Risultati attesi

In primo luogo, quanto si intende dimostrare mediante questo esperimento, è la possibilità di entrare in possesso di dati facenti parte di connessioni altrui rispondendo ai pacchetti che talvolta vengono ricevuti non appena avviene la connessione alla rete mobile. Tale riposta implica che è necessario bypassare il meccanismo automatico di reset.
In secondo luogo, in merito all'acquisizione dei dati, è stata riservata un particolare attenzione nella ricerca di connessioni in chiaro, nelle quali fosse quindi possibile vedere il traffico non criptato. Un tale fenomeno avrebbe aggravato la violazione di dati che questo esperimento già costituisce per sua natura.

Capitolo 4

Il software

Per la strumentazione hardware necessaria per poter effettuare la connessione alla rete mobile, è stato scritto del software con l'obiettivo di sfruttare le potenzialità hardware ai fini dell'obbiettivo dell'esperimento. Il software si occupa dunque sia di iniziare la connessione impostando adeguatamente i parametri del modem, sia di gestire il traffico dati in ingresso operando opportuni controlli e scelte sui pacchetti ricevuti e forgiando in un secondo momento i pacchetti di risposta ad hoc che generano il traffico in uscita.

4.1 Wvdial

Wvdial è un servizio che consente di connettere il modem alla rete, dopo aver impostato alcuni parametri relativi all'operatore telefonico e alla linea.
Le impostazioni dei parametri vengono memorizzate in un file di configurazione che viene invocato automaticamente dal servizio al suo avvio.
I parametri presenti nel file vengono eseguiti in sequenza ed hanno tutti la struttura: "Nome Parametro"="Valore parametro".
Tra i parametri impostabili sono presenti:

- fino a 3 stringhe anche non consecutive contenenti comandi AT inviati al modem per l'inizializzazione[1] e la connessione[2];

- il nome utente e la password, i quali devono necessariamente essere dichiarati anche se il loro valore può essere nullo;

[1] Nella fase di inizializzazione il primo comando inviato al modem è generalmente ATZ il quale resetta tutte le impostazioni del modem al valore predefinito. Un altro comando che è obbligatorio inviare al modem è "AT+CGDCONT", il quale si compone di diversi campi e permette di definire l'APN utilizzata per la connessione (variabile da operatore a operatore) e il tipo di protocollo utilizzato (nello specifico, IP).

[2] La fase di connessione consiste quasi essenzialmente nella chiamata ATD*99***1# al numero che consente l'aggancio alla rete.

- l'interfaccia alla quale il modem è collegato. Nel caso specifico il modem è stato connesso al computer attraverso una porta USB pertanto l'interfaccia impostata è "ttyUSB0";
- la velocità di connessione espressa in baud[3] che nell'esperimento è stata impostata a 7200 baud/s;
- l'ultimo parametro impostabile e necessario ai fini dell'esperimento è "stupid mode". Tale parametro, se impostato a on permette di effettuare traffico nel momento immediatamente successivo alla connessione alla rete.

4.2 Continuatore di connessioni

Accanto al software responsabile di impostare il modem ed effettuare la chiamata per la connessione alla rete mobile, è stato necessario scrivere del software che gestisse la creazione di un canale comunicativo, la ricezione dei pacchetti, la memorizzazione del contenuto informativo di interesse e infine la costruzione delle risposte ai pacchetti ricevuti.

4.2.1 Operazioni preliminari

Nella prima parte del programma vengono elencate le firme dei metodi ausiliari chiamati nel caso di reperimento o calcolo ripetitivo di alcuni dati, vengono inoltre dichiarare le strutture dati e le variabili che verranno utilizzate all'interno del metodo *main()* principale. Come accennato, i metodi ausiliari si occupano di espletare funzioni ricorrenti e limitate ad un compito che non influisce con altre parti del *main()*. Nello specifico sono stati implementati metodi per:

- ricavare l'indice dell'interfaccia alla quale è allacciato il modem al fine di aprire su di essa il canale comunicativo;
- riunire i campi che costituiscono lo pseudoheader in un unico buffer;
- compilare di volta in volta la struttura del pacchetto di risposta sfruttando i dati di input costituiti dai campi del pacchetto ricevuto;

Le strutture dati dichiarate contengono i campi dei vari livelli ISO-OSI che il programma è predisposto ad analizzare quali: header IP, header TCP, pacchetto di risposta e pseudoheader. Per quanto riguarda le ultime due

[3] baud è sinonimo di simboli/s, ovvero rappresenta il numero di simboli che viene trasmesso in un secondo. È l'unità di misura del 'baud rate' che indica il numero di simboli trasmessi al secondo in un sistema di trasmissione come un modem. Questa unità di misura viene spesso scambiata con bit/s che è una misura differente in quanto un simbolo può essere composto da un numero differente di bit a seconda della modulazione.

strutture dati, pur non essendo necessaria un'implementazione dedicata delle stesse, si è optato comunque per la loro costruzione al fine di renderne più immediato l'impiego.
Le strutture vengono sempre utilizzate sotto forma di puntatori al relativo buffer[4], il quale rappresenta lo stream di dati in ingresso, uscita o locale ad esempio nel caso del calcolo del checksum. Operando in questo modo è possibile suddividere virtualmente il buffer nei campi necessari, mantenendo composizione e dimensione della struttura associata.
Tutte le strutture dati utilizzate sono descritte in dettaglio nella sezione sec:composizione di un pacchetto.
Riguardo le variabili dichiarate in questa fase preliminare, accanto ai vari indici e variabili di supporto ai diversi calcoli è degna di nota la tabella dedicata alla memorizzazione dei dati rilevanti delle connessioni.
La struttura a tabella è stata implementata al fine di memorizzare in maniera ordinata i dati relativi ad ogni connessione captata seguendo l'avanzare di ognuna di esse senza interferenze. La tabella in questione viene inizializzata a 0 nella sua interezza non appena è dichiarata, in maniera tale da evitare errori dovuti al contenuto casuale dei suoi campi, nelle fasi successive del programma.
Ogni riga della tabella rappresenta una diversa connessione captata all'interno di una sessione ed è suddivisa in campi contenenti i dati rilevanti della stessa. In particolare i primi campi tengono traccia dei valori di *sequence number*, *acknoledgement number*, indirizzo IP sorgente, porta sorgente e di destinazione, numero dei pacchetti ricevuti, ricezione del flag di FIN e timer di attività della connessione. In un ultimo campo si memorizzano in maniera consecutiva e contigua i dati reperiti di volta in volta nei pacchetti della connessione. Ad ogni nuova ricezione, i campi della riga relativa alla connessione dov'è stato ricevuto il pacchetto, vengono inizialmente utilizzati per effettuare i diversi controlli di validità del pacchetto e in un secondo momento alcuni di essi vengono aggiornati coerentemente con l'avanzamento della comunicazione.

4.2.2 La creazione dei socket e l'ascolto

Il socket è il canale comunicativo attraverso il quale è possibile trasmettere i pacchetti in ingresso e uscita sfruttando apposite funzioni. Per poter aprire il canale comunicativo, ad esso devono essere associate informazioni riguardo al modo in cui deve funzionare e l'interfaccia attraverso la quale scambiare i pacchetti di dati.

[4] I buffer utilizzati cono degli array di *unsigned char* di lunghezza che può essere prefissata nel caso in cui si conosca la lunghezza esatta della struttura che ne farà riferimento, oppure sovrastimata se la struttura che riferisce il buffer ha lunghezza sconosciuta a priori. Il tipo di dato che costituisce i buffer è stato scelto in quanto una variabile di questo tipo ha dimensione pari ad un byte.

La creazione del canale comunicativo avviene per mezzo dell'omonima funzione *socket()*, la quale riceve in input i valori:

- socket_family: rappresenta la famiglia alla quale appartiene il socket che nel caso specifico è stata impostata a *AF_PACKET* che rappresenta la famiglia dei pacchetti di basso livello. Altri tipi noti di famiglie di socket sono ad esempio le *AF_INET* e *AF_INET6*, le quali riguardano i pacchetti dal livello 3 ISO_OSI, rispettivamente nei protocolli IPv4 e IPv6;

- socket_type: è il parametro che rappresenta il tipo di socket che verrà creato. Il socket utilizzato nel programma è di tipo *SOCK_RAW*, questa tipologia permette ad un utente di creare manualmente i pacchetti che transitano sul canale comunicativo a partire da un livello ISO-OSI basso. In questo modo un utente esperto può accedere a funzionalità e parametri che in altri casi potrebbero rimanere trasparenti.
  Altri tipi di socket sono ad esempio *SOCK_DGRAM*, solitamente utilizzato nelle comunicazioni UDP in quanto supporta una comunicazione bidirezionale senza garanzia di assenza di pacchetti duplicati, sequenzialità nella ricezione degli stessi ed effettiva ricezione di tutti i pacchetti che compongono la comunicazione. L'unico aspetto garantito da questa tipologia di socket è il mantenimento delle porzioni di dati inviati. Tutto ciò che non viene garantito dalla precedente tipologia di socket, è invece assicurato da *SOCK_STREAM*, il quale è infatti preferito quando è fatto uso di trasmissioni TCP;

- protocol: tramite questo parametro è possibile specificare il protocollo che il socket deve supportare nella ricezione dei pacchetti. Solitamente ad ogni socket è richiesto di supportare un solo specifico protocollo durante la comunicazione, il quale viene specificato mediante un apposita parola chiave differente per ogni protocollo. Dato che i controlli sulla tipologia e correttezza dei pacchetti sono stati implementati dal software, in questo esperimento il socket trasmette qualsiasi tipo di protocollo ed il parametro in questione viene pertanto compilato con la parola chiave "ETH_P_ALL".

Il socket, a parametri impostati, non è ancora pronto per essere utilizzato in quanto necessità di essere associato ad una interfaccia sulla quale ricevere e trasmettere i dati. L'associazione del socket all'interfaccia avviene attraverso la funzione *bind()*, la quale riceve in input, oltre allo stesso socket appena configurato, una struttura di supporto di tipo *sockaddr_ll*. I campi di quest'ultima struttura sono molteplici e la loro compilazione è differente a seconda dell'applicazione implementata. Nel software in questione è sufficiente compilare correttamente solo i campi obbligatori descritti in seguito lasciando gli altri inizializzati a 0:

- famiglia: valore e significato di questo campo sono equivalenti a quelli del parametro socket_family del socket, pertanto il campo viene compilato con *AF_PACKET*;

- protocollo: il valore indicato esplicita la natura dei pacchetti che vengono trasmessi. Nel caso specifico i pacchetti sono dei datagrammi e viene pertanto impostato il valore a *0X0008* (valore del protocollo IP);

- indice: il valore di quest'ultimo campo obbligatorio indica l'indice dell'interfaccia che deve essere utilizzata durante la trasmissione. Tale parametro è ricavabile attraverso una funzione dedicata, la quale ha il compito di interrogare apposite strutture di sistema. Per ricavare il valore dell'indice è sufficiente conoscere almeno il nome dell'interfaccia che si intende utilizzare (l'interfaccia utilizzata dal modem in questo esperimento è la *PPP0*).

La funzione *bind()* ha quindi il compito di unire il socket alla struttura appena descritta non appena entrambi sono stati debitamente compilati. Associando il socket ad una specifica interfaccia e tipologia di dati, lo si rende in grado di trasmettere su di essa uno stream di dati in ingresso e uscita. Il software procede pertanto mettendo il socket in ascolto sull'interfaccia prestabilita, in attesa che qualche dato vena ricevuto.
L'ascolto sul canale comunicativo avviene per mezzo della funzione *recvfrom()* la quale impone al socket di rimanere in attesa di ricevere dati. La funzione appena dichiarata interroga ripetutamente il canale comunicativo restituendo il valore -1 in caso di errore, 0 nel caso in cui l'interrogazione riveli che non è ancora stato ricevuto nessun dato oppure il numero di byte ricevuti nel caso in cui venga ricevuto qualcosa. La chiamata a tale funzione è bloccante, ovvero il software non può procedere oltre finché non viene ricevuto qualche dato. Questo fatto rappresenta un effetto non voluto ed è quindi stato implementato un meccanismo di chiusura automatica del canale comunicativo nel caso in cui ad ogni ascolto l'attesa dei dati superi un certo lasso temporale.

4.2.3 Ricezione e controllo della correttezza dei pacchetti

In accordo con le impostazioni del canale comunicativo descritte precedentemente, la funzione di ascolto del socket permette l'ingresso di tutte le tipologie di pacchetti lasciando al software l'onere di effettuare i dovuti controlli.
Ogni pacchetto ricevuto viene immediatamente scomposto nei campi che lo compongono ai livelli 3 e 4 ISO-OSI così da rendere facile la consultazione e la manipolazione dei dati in essi contenuti. Il primo controllo effettuato dal software consiste nel verificare l'appartenenza o meno del pacchetto al protocollo TCP, protocollo utilizzato in questo esperimento, verificando che

il valore del campo "Protocol" dell'header IP sia pari a 6. Tutti i pacchetti ricevuti che non rispettano tale vincolo vengono immediatamente scartati, gli altri invece avanzano ai controlli e manipolazioni successivi.
Appurato che un pacchetto è di tipo TCP, si procede con il controllo dei campi "Destination Address" dell'header IP e "Source Port" e "Destination Port" dell'header TCP, i quali costituiscono insieme una identificativo univoco di ogni connessione all'interno di una sessione. Questo controllo viene effettuato confrontando i valori dei suddetti tre campi con i corrispettivi valori memorizzati nella tabella delle connessioni che è stata accennata nella sottosezione 4.2.1.
Un ultimo controllo permette di stabilire se un pacchetto appartenente ad una connessione già avviata è adiacente alla successione di dati memorizzata confrontando il campo "Sequence Number" dell'header TCP con il valore "Acknoledgement Number" presente nella tabella delle connessioni. Il valore in tabella rappresenta il prossimo bit dello stream di dati che ci si aspetta di ricevere mentre il valore presente nel pacchetto rappresenta il primo bit di dati dello stream presente all'interno del pacchetto stesso. Com'è facile intuire un pacchetto risulta allineato e supera il controllo solamente se i due valori sono concordi.

4.2.4 Memorizzazione e aggiornamento dei dati

Riprendendo quanto detto sulla tabella di memorizzazione delle connessioni, si ricorda che appena creata ogni sua cella viene inizializzata a 0 per evitare conflitti durante i controlli effettuati nel corso della sessione. Ogni riga della tabella si suddivide in diverse parti necessarie a memorizzare i dati di una sola connessione, l'ultima di queste parti ha dimensioni rilevanti per poter memorizzare il campo dati dei pacchetti di volta in volta captati e appartenenti alla connessione.
Nel momento in cui un pacchetto TCP viene sottoposto agli ulteriori controlli descritti nella sottosezione precedente, è necessario scorrere le righe della tabella confrontando ad ogni ciclo i campi per determinarne la natura.
Per ogni pacchetto ricevuto possono verificarsi 3 situazioni differenti che necessitano di essere gestite in maniera coerente con il caso:

- il pacchetto appartiene ad una nuova connessione. Se scorrendo le righe della tabella, a partire dalla prima, in cerca di una corrispondenza con i tre campi che identificano univocamente la connessione si incorre in una riga costituita da celle con il solo valore 0, allora significa che il pacchetto in analisi non appartiene a nessuna delle connessioni precedentemente memorizzate o è il primo pacchetto della sessione ricevuto[5].

[5] All'interno del ciclo si controlla anche che la tabella non sia piena al fine di non andare in errore cercando di scrivere su celle di memori estranee alla tabella. Tuttavia il valore massimo delle righe è tale da consentire che in nessun caso si raggiunga la fine della tabella.

Al verificarsi di questa situazione, la riga in questione sarà destinata alla memorizzazione delle sole informazioni della nuova connessione rilevata. Essendo il primo pacchetto della connessione, da esso verranno estratti tutti i campi necessari ad identificare la connessione che verranno utilizzati per il confronto con i successivi pacchetti ricevuti. Oltre ai campi identificativi, vengono memorizzate anche altre informazioni necessarie alla localizzazione del pacchetto all'interno dello stream di dati della connessione e l'eventuale campo dati del suddetto pacchetto;

- il pacchetto appartiene ad una connessione già iniziata ma non è adiacente al precedente pacchetto ricevuto all'interno della stessa connessione. In questo caso il pacchetto verrà scartato in quanto i dati in esso contenuti non rappresentano una naturale continuazione dello stream di dati della connessione. Questa tipologia di pacchetti supera con successo il controllo dei tre campi univoci che rappresentano la connessione trovando la riga corrispondente nella tabella di memorizzazione delle connessioni. A questo punto fallisce però l'ultimo controllo descritto, il quale rileva una non contiguità dei dati con conseguente eliminazione del pacchetto;

- il pacchetto appartiene ad una connessione già iniziata e contiene dati contigui a quelli già memorizzati. In questo caso il pacchetto supera tutti i controlli e trova la propria collocazione all'interno della tabella. L'azione intrapresa dal programma quando viene rilevato un pacchetto con tali caratteristiche è quella di aggiornare i campi della tabella che risulterebbero obsoleti all'arrivo di un eventuale pacchetto successivo. Nello specifico nella riga della tabella si aggiornano i valori di "Acknoledgement Number" e "Sequence Number" prelevandoli dal pacchetto, si incrementano il numero di pacchetti ricevuti e il numero di ottetti di dati totali all'interno della connessione e si aggiungono infine i dati contenuti nel pacchetto a partire dalla prima posizione libera del buffer dedicato[6].

Quando il pacchetto supera i controlli e il software memorizza i dati in esso contenuti, viene incrementato anche un indice globale[7] che conta il numero totale di pacchetti TCP corretti ricevuti durante la sessione.

[6] Al fine di evitare celle vuote o sovrapposizioni durante la memorizzazione dei dati all'interno del buffer, il campo della tabella che tiene traccia del numero totale di ottetti ricavati all'interno della connessione viene sfruttato anche come indice e rappresenta sempre il valore della prima cella libera del buffer.

[7] Tale valore consente di personalizzare la notifica di timeout della funzione *recvfrom()* specificando se il timeout è stato raggiunto perché non sono più arrivate risposte a continuazione delle connessioni captate o se non è stato ricevuto alcun pacchetto durante la sessione.

4.2.5 Costruzione della risposta ai pacchetti

La comunicazione tra due dispositivi che dialogano in rete seguendo il protocollo TCP avanza per mezzo di pacchetti informativi contenenti dati e conferme di ricezione, il protocollo TCP infatti garantisce per sua natura la correttezza e la ricezione dei pacchetti durante ogni connessione.
Nel momento in cui avviene la ricezione di un pacchetto che supera tutti i controlli imposti dal software senza essere scartato, si rende quindi necessario forgiare una risposta a tale pacchetto in maniera coerente con le regole del protocollo TCP e tale che l'interlocutore, una volta ricevuta la stessa, proceda con l'invio del pacchetto successivo.
La composizione del pacchetto di risposta è stata affidata ad un funzione dedicata, la quale riceve in input i dati necessari prelevati dal pacchetto ricevuto. I dati passati come argomento vengono inseriti nei campi di destinazione all'interno della struttura del pacchetto di risposta, eventualmente dopo opportuna manipolazione.
Il buffer che costituisce il pacchetto di risposta è stato scomposto in un'unica struttura per praticità di utilizzo. In particolare la struttura si compone dell'header IP, seguito dall'header TCP, non sono presenti né i campi facoltativi options né il campo dati successivo all'header TCP in quanto il pacchetto costituisce solamente una conferma di quanto è stato ricevuto.
I campi della struttura vengono compilati automaticamente dal software coerentemente con la descrizione fornita per ognuno di essi nelle sottosezioni 2.3.1 e 2.3.2. Ogni campo della struttura deve essere compilato in maniera corretta, in caso contrario si corre il rischio che l'interlocutore scarti il pacchetto di risposta facendo terminare la connessione.
Trattandosi di un pacchetto di risposta ad uno stream di dati, è importante che in tale pacchetto avvenga l'inversione degli indirizzi IP e delle porte rispetto al pacchetto al quale si riferisce la risposta. Un'altra operazione di interesse è quella che riguarda la compilazione dei campi "Sequence Number" e "Acknoledgement Number"; anche in questo caso infatti avviene un'inversione degli stessi rispetto al pacchetto ricevuto, con la differenza che il campo "Acknoledgement Number" viene incrementato di un valore pari al numero di byte di dati successivi all'header TCP presenti nel pacchetto ricevuto. Il campo in questione infatti serve a confermare all'interlocutore qual'è l'ultimo byte ricevuto e viene aggiornato anche nella tabella di memorizzazione delle connessioni in quanto risulta necessario nella valutazione dell'eventuale pacchetto successivo.
Quando la struttura del pacchetto di risposta risulta completa in tutte le sue parti, il software procede con l'invio del pacchetto utilizzando lo stesso socket creato per la ricezione ed invocando la funzione *send()*[8]. La funzione

[8]Prima di ogni invocazione sia della funzione *send()* che della funzione *recvfrom* è necessario fare un refresh della struttura di supporto *sockaddrll* reimpostandone i valori. Questa operazione risulta obbligatoria in quanto ogni utilizzato della struttura nelle fun-

send(), sulla falsa riga di quanto accade per la corrispettiva *recvfrom()*, riceve in input il buffer contenente il pacchetto di risposta, la sua dimensione e la struttura *sockaddr_ll* di supporto e relativa dimensione. Sempre coerentemente con la sua controparte, la funzione dedicata all'invio del pacchetto restituisce il valore -1 nel caso in cui venga rilevato un errore, 0 nel caso in cui non venga inviato alcun dato, il numero totale di byte inviati nel caso in cui l'operazione vada a buon fine.

4.2.6 Chiusura della connessione e salvataggio dei dati

La terminazione del software, salvo l'incorrere di errori non previsti dall'esecuzione, è delegata ad opportuni timeout i quali rilevano l'inattività del software, sia essa dovuta alla sessione o alle singole connessioni.
Un primo timeout, il quale è già stato presentato nella parte iniziale dedicata alla presentazione del software, ha il compito di terminare il programma qualora venga rilevata l'inattività della funzione *recvfrom()*. Si ricorda infatti che tale funzione per sua natura bloccherebbe il software finché non viene ricevuto un pacchetto e per questo motivo è stata affiancata da un timer parallelo, il quale ne monitora l'attività e che, alla sua scadenza, impone la terminazione del software.
Quando interviene il primo timeout il programma consulta la variabile globale che conta il numero totale di pacchetti ricevuti all'interno della sessione, se questa variabile ha mantenuto il valore di inizializzazione pari a 0, una stringa di output segnala all'utente che la sessione non ha generato traffico. In questo caso specifico il software termina in quanto non è presente alcun dato da memorizzare.
Durante le sessioni di connessione vengono ricevuti pacchetti di svariata natura oltre alla tipologia TCP, tali pacchetti benché non siano di interesse ai fini dell'esperimento, attivano la funzione *recvfrom()* in quanto costituiscono uno stream di dati. La ricezione di questi dati bypassa il timeout appena descritto e manifesta il rischio che il programma rimanga attivo solamente per scartare i pacchetti non desiderati perdendo tempo e rischiando il blocco in loop.
Al fine di prevenire il rischio appena descritto, è stato impostato un secondo timer che chiude il canale comunicativo qualora il tempo trascorso dall'ultima ricezione di un pacchetto TCP superi una determinata soglia. Così facendo, se dopo aver terminato lo scambio di pacchetti di tipo TCP il programma continua a ricevere solamente pacchetti di altra natura, si ha comunque la certezza che questo termini evitando il loop di inattività.
A scopo puramente informativo riguardo lo stato delle connessioni nel corso di ogni sessione, è stato inserito un timeout anche nella tabella di memoriz-

zioni citate, provoca una modifica della stessa che, non intervenendo con il refresh dei valori, si tradurrebbe in un errore al successivo utilizzo con conseguente terminazione del software.

zazione delle connessioni in corrispondenza di ogni riga utilizzata. Ad ogni pacchetto TCP ricevuto, il timer di una connessione viene azzerato nel caso in cui si stia memorizzando il primo pacchetto della stessa oppure si abbia ricevuto un altro pacchetto ad essa appartenente. Per tutte le connessioni già iniziate che non corrispondono al pacchetto appena ricevuto, il timer di ognuna viene incrementato del tempo trascorso. Quando un timer raggiunge il valore specificato all'interno del programma, il timeout entra in azione segnando la corrispondente connessione come chiusa e impostando il valore del timer a -1. Tramite questa scelta progettuale è possibile monitorare l'attività delle connessioni attraverso stringhe di output dedicate.
Al raggiungimento di un qualsiasi timeout, per costringere il programma ad andare in fase di terminazione, è sufficiente chiudere il canale comunicativo tramite la funzione *close(socket)*. L'effetto di questa operazione è quello di mandare volutamente in errore la funzione *recvfrom()*, errore che viene gestito nei vari casi in maniera automatica dal software avviando la fase di terminazione.
Nel caso in cui venga ricevuto almeno un pacchetto[9] TCP durante ogni sessione, al raggiungimento di uno dei timeout predisposti per terminare il programma, quest'ultimo prima di terminare procede con una fase di salvataggio dei dati acquisiti: attraverso un ciclo si scorrono una alla volta le righe della tabella di memorizzazione delle connessione. Nel caso in cui la riga analizzata contenga dati relativi ad una connessione, viene creato un nuovo file al cui interno il software memorizza i valori identificativi e riassuntivi della connessione e l'intero stream di dati che è stato ricevuto. Prima di procedere con la riga successiva della tabella, il file viene salvato impostando il nome con il template: "porta sorgente-data-ora".
Appena il programma raggiunge la prima riga della tabella che non è stata utilizzata durante la sessione, può finalmente terminare.

4.3 Script di coordinazione

I due programmi descritti nelle sezioni precedenti hanno funzioni complementari e devono quindi funzionare assieme in maniera coordinata al fine di ottenere quando voluto dall'esperimento. Riassumendo: Wvdial è il software che si occupa di effettuare la connessione del modem alla rete attraverso la SIM card inserita all'interno di quest'ultimo; il continuatore di connessioni si occupa invece di creare un canale comunicativo capace di trasmettere e ricevere dati sfruttando la connessione stabilita dall'altro software e manipolare i dati che transitano su tale canale.
Il coordinamento di questi due software viene gestito attraverso uno script

[9] Il caso particolare di una sessione senza che venga registrata la ricezione di alcun pacchetto è stato esposto nella descrizione del primo timeout. Non essendoci alcuna informazione memorizzata, il programma termina senza affrontare fasi successive.

in linguaggio bash. Lo script è, ad accezione delle dichiarazioni di variabili, completamente racchiuso in un ciclo infinito di tipo *while(1)*. Ogni esecuzione del ciclo per intero costituisce una delle sessioni nominate nella sezione precedente e si compone di più fasi distinte:

- viene innanzitutto eseguito in background il software Wvdial[10], che procede all'abilitazione dell'interfaccia PPP0 utilizzata nel programma e all'avvio della fase di negoziazione con il server al fine di ottenere un indirizzo IP valido ed instaurare la connessione;
- immediatamente dopo l'esecuzione di Wvdial viene eseguito il continuatore di connessioni. Il continuatore di connessioni può essere eseguito anche se l'interfaccia PPP0 non è ancora abilitata o se la fase di negoziazione da parte di Wvdial è ancora in corso. Il primo caso è stato gestito tramite un ciclo che controlla ripetutamente l'avvenuta abilitazione dell'interfaccia, il secondo invece è risolto automaticamente dalla funzione *bind()* che rimane in attesa dell'effettiva disponibilità del socket sull'interfaccia desiderata;
- per quanto dichiarato nella sezione precedente si ha la certezza che il continuatore di connessioni termini in ogni caso. La terminazione di questo programma, il quale non è eseguito in background, permette allo script di eseguire la funzione successiva relativa alla terminazione del programma Wvdial;
- l'ultima operazione svolta dallo script è uno *sleep(s)* di alcuni secondi. La funzione mette in pausa lo script per un tempo pari agli s secondi specificati nell'argomento concedendo al modem il tempo di disconnettersi. I secondi di pausa sono necessari e sono stati introdotti dopo aver registrato errori di acquisizione di indirizzo IP tentando di effettuare una riconnessione repentina del modem;
- concluse le operazioni descritte, lo script ricomincia dalla prima istruzione del ciclo. La terminazione dello script avviene attraverso una combinazione di tasti manuale da parte dell'utente.

[10] L'esecuzione in background di questo software consente allo script di passare alle istruzioni successive senza attendere la terminazione del programma. Il programma rimane comunque attivo e funzionante finché non viene terminato attraverso apposita funzione.

Capitolo 5

Il processo di esecuzione

In questo capitolo viene presentato lo svolgimento complessivo dell'esperimento fornendo una visione ad alto livello del software descritto nel capitolo precedente.
Al fine di favorire la comprensione del procedimento seguito dal software, nella pagina successiva è stato rappresentato un diagramma di flusso che sintetizza le fasi del quale esso si compone, nonché le dipendenze per mezzo delle quali queste sono in comunicazione tra loro.
La rappresentazione tramite diagramma di flusso permette di suddividere in macro-fasi il software complessivo sollevando il lettore dalla comprensione dei dettagli del codice presentati nel capitolo precedente e concentrando l'attenzione sulla logica alla base del programma.

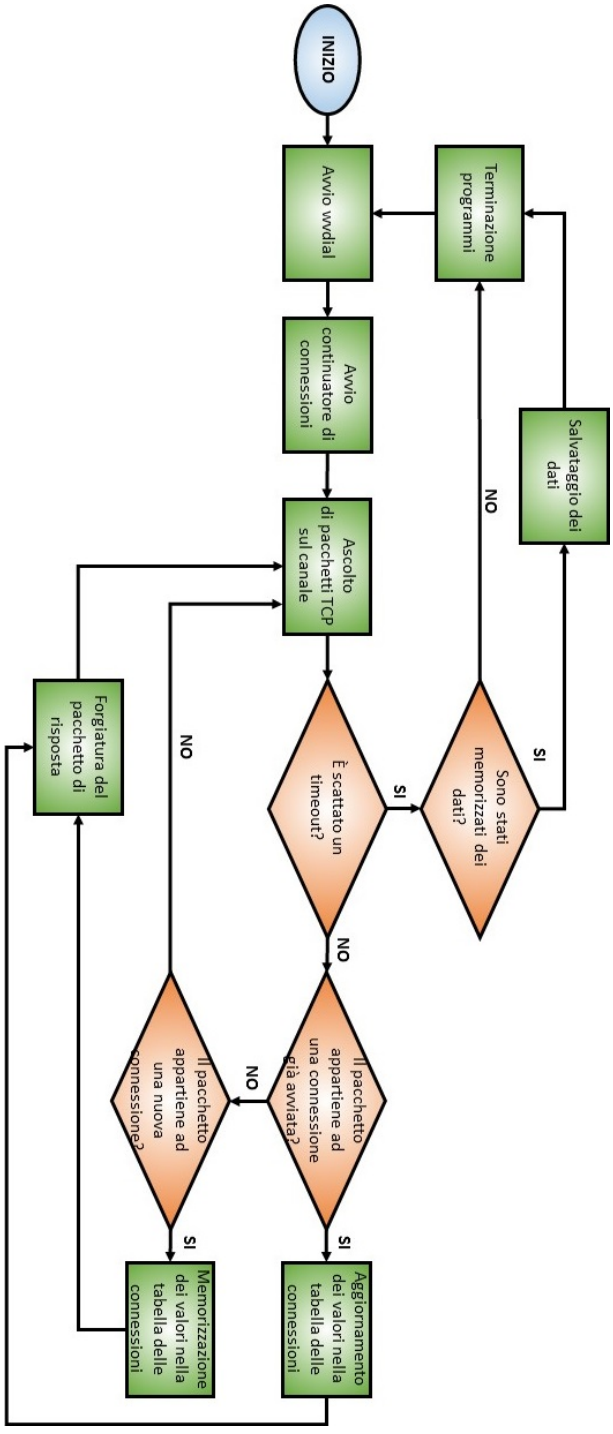
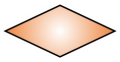


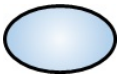
Figura 5.1: Diagramma di flusso complessivo del software.

Viene di seguito fornita una legenda esplicativa riguardo il significato dei simboli dei quali si compone il diagramma di flusso rappresentato:

Nel diagramma di flusso, tramite questa figura si rappresentano le funzioni svolte dal software. Ogni figura rappresenta la versione ad alto livello del set di istruzioni che la rappresenta nel codice;



Attraverso questa figura si rappresentano nel diagramma di flusso, i casi nei quali il software deve compiere delle decisioni sulla base della valutazione di specifici parametri. Questo genere di figura rappresenta la versione ad alto livello degli "if-then-else" che consentono al software di adattarsi alle varie casistiche;



Questa figura è stata introdotta solamente con l'obiettivo di indicare dove effettivamente fosse collocabile l'inizio del software relativo all'esecuzione delle prime istruzioni;

Tali figure indicano una relazione di dipendenza che intercorre tra altre due differenti oggetti. In questo modo è possibile fornire una versione ad alto livello riguardo il modo con cui sono collegate tra loro le fasi chiave del software.

L'inizio del programma nella sua totalità, avviene per mezzo dell'avvio del software dedicato alla creazione dell'interfaccia PPP e alla negoziazione dei dati di connessione con il server. In una fase immediatamente successiva avviene quindi l'esecuzione del software continuatore delle connessioni. Quest'ultimo software si occupa di creare un socket comunicativo sull'interfaccia PPP appena inizializzata e si mette in ascolto su di esso in attesa di ricevere eventuali informazioni.
Come anticipato nel capitolo dedicato alla descrizione dettagliata del software impiegato, sono stati definiti diversi timeout al fine di evitare gli sprechi di tempo e massimizzare i risultati ottenibili. La scadenza di uno qualsiasi dei timeout implica la terminazione del software continuatore delle connessioni con conseguente avvio di un nuovo ciclo del bash di coordinazione. Ogni nuovo ciclo si traduce in pratica con la creazione di una nuova interfaccia PPP sulla quale creare un socket per l'ascolto.
Nel caso in cui venga ricevuto un pacchetto TCP in tempo utile, il timer predisposto non scade, facendo procedere il software con l'analisi del pacchetto. In particolare viene analizzata la natura della connessione alla quale il pacchetto appartiene (nuova connessione o connessione già avviata) agendo di conseguenza nella fase di memorizzazione dei nuovi dati. Se il pacchetto non corrisponde a nessuna delle due categorie appena elencate, questo viene scartato e il software torna in ascolto sul canale. Nell'eventualità che il pacchetto sia coerente con una delle due casistiche invece, in seguito all'aggiornamento o memorizzazione dei dati in esso contenuti nella matrice di memorizzazione delle connessioni, il software procede con la fase dedicata alla forgiatura del pacchetto di risposta. In un momento successivo alla forgiatura, il nuovo pacchetto di risposta viene inviato e il programma si rimette in ascolto sul canale comunicativo.
Al controllo relativo alla scadenza del timer, nel caso in cui questo dia esito positivo, viene effettuata un'ulteriore verifica riguardo la presenza di eventuali dati memorizzati prima della scadenza dello stesso. Tale controllo permette di rilevare e provvedere alla necessità di produrre in output un report per ogni connessione avvenuta, ciascuno contenente le informazioni sensibili che la caratterizzano.

Capitolo 6

Risultati statistici

L'obiettivo di questo capitolo è quello di fornire una serie di misurazioni fisiche che formalizzino il procedimento descritto a parole nei capitoli precedenti dell'elaborato.
L'output prodotto dal software nel suo complesso, si compone dei report contenenti i dati relativi ad ogni connessione catturata e dei file di log prodotti durante l'esecuzione. Al contrario dei primi, dei quali sono stati ampiamente descritti composizione e contenuto, per quanto riguarda la categoria dei file di log è opportuno dare una visione sommaria del contenuto in essi memorizzato. Tali file vengono aggiornati in parallelo all'esecuzione del software complessivo ed in essi è possibile ricercare tutti gli eventi intercorsi durante l'esecuzione e per i quali è stato previsto un opportuno output.
In fase di implementazione del software o di nuove funzionalità ad esso correlate, i file di log svolgono una funzione di supporto permettendo ad esempio di rilevare esecuzioni di porzioni di codice impreviste oppure monitorare il contenuto di determinate variabili in punti chiave del flusso di esecuzione al fine di verificarne la correttezza. Durante la fase di esecuzione ininterrotta del software, ovvero quando si è verificato che tutte le funzionalità in esso contenute operano nella maniera corretta, i file di log sono stati utilizzati anche al fine di archiviare:

- le informazioni salienti riguardo lo stato assunto nel tempo dal software;
- la registrazione di eventuali interruzioni impreviste e della relativa descrizione;
- lo stream di dati generato dal software sia in input che in output.

Analizzando il contenuto di tutti i file prodotti durante l'esecuzione del software è stato possibile effettuare delle stime sia quantitative per quanto riguarda le connessioni rilevate e la relativa dimensione che qualitative estraendone il contenuto e partizionando determinati parametri.

Una prima analisi statistica ha avuto come obiettivo l'individuazione della percentuale di sessioni all'interno delle quali è stata catturata almeno una connessione rispetto al totale.
Mediante l'output prodotto nei fil di log precedentemente descritti, è stato possibile individuare le sessioni nelle quali l'inattività del canale comunicativo ha fatto scattare il timeout del software continuatore di connessioni. Lo sfruttamento di questo parametro in correlazione ai dati appartenenti alle sessioni produttive ha permesso di effettuare una stima dei tentativi di connessione andati a buon fine.
Attraverso i parametri volumetrici di traffico memorizzati in fase di chiusura delle singole connessioni è stato inoltre possibile stabilire all'interno delle sessioni che sono risultate produttive, che porzione di queste ha anche generato uno scambio di dati.
In molti casi infatti, lo scambio di pacchetti in una connessione interessava solamente gli header dei livelli 3 e 4 ISO-OSI privi di dati, è il caso ad esempio dei pacchetti keep-alive[1] i quali sono risultati essere abbastanza frequenti.
I risultati della stima appena descritta sono rappresentati nel grafico 6.1.

[1] Il segnale keep-alive viene solitamente inviato ad intervalli regolari da un dispositivo che ha instaurato una connessione con un altro e svolge un ruolo importante nella rete permettendo di rilevare l'indisponibilità di un host con il quale si sta effettuando o si è effettuata una comunicazione. Dopo che un segnale keep-alive viene inviato, se non giunge nessuna risposta dall'interlocutore quest'ultimo viene considerato non disponibile. Il verificarsi di questa condizione indica all'host mittente che future comunicazioni dovranno essere instradate utilizzando percorsi alternativi fino a che il canale che risulta non disponibile ritorna operativo.
Attraverso i pacchetti keep-alive è possibile espletare anche un'altra importante funzione in ambito della comunicazione tra due host.Tramite tali pacchetti è infatti possibile preservare il canale comunicativo impedendo che questo venga automaticamente chiuso per rilevata inattività. In questo modo due host che hanno necessità di scambiarsi informazioni ad intervalli temporali prolungati possono così avere la certezza della disponibilità nel tempo del canale comunicativo.
Quanto descritto è tuttavia stato uno sviluppo successivo alla nascita di questa tipologia di pacchetti. Inizialmente infatti, la loro finalità era stata pensata per il solo test della raggiungibilità di canali e host.

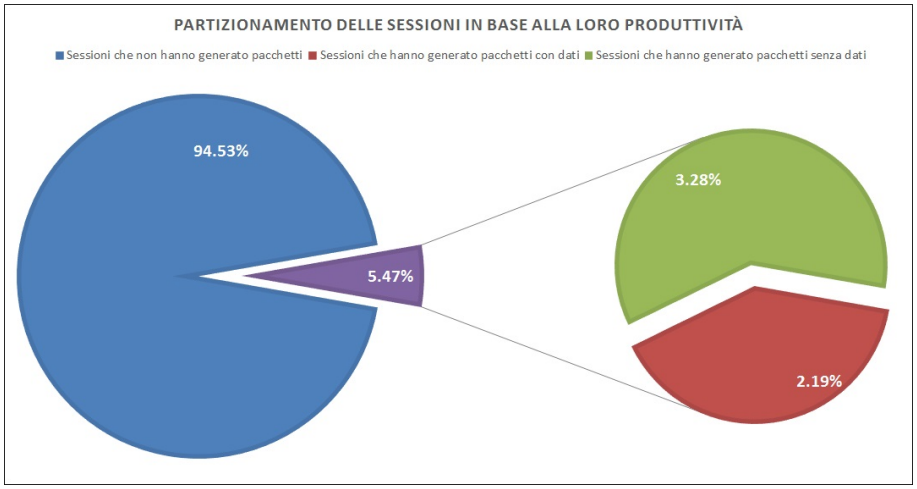


Figura 6.1: Distribuzione delle sessioni di connessione dapprima in base all'attività ed in seguito rispetto alla presenza o meno di volume di traffico.

Dal grafico 6.1 si evince che in media solamente cinque sessioni su cento hanno generato uno scambio di dati. Delle sessioni nelle quali è avvenuta una comunicazione tra host inoltre, solamente due quinti di queste hanno costituito un effettivo traffico di informazioni.
Nel grafico, ogni sessione appartenente ai sottoinsiemi che hanno generato uno scambio di pacchetti è stata considerata come un unico elemento. Ciascuno di questi elementi può comunque essere costituito da più connessioni al suo interno, le quali possono essere tutte prive di dati, tutte con contenuto informativo od una via di mezzo tra le due casistiche.

Un'ulteriore analisi statistica ha riguardato la distribuzione oraria delle connessioni catturate. In tale analisi sono state pertanto considerate solamente le connessioni che hanno generato almeno un pacchetto di traffico, a prescindere dal volume di quest'ultimo.
Le connessioni sono state poi suddivise in base all'ora del giorno nella quale sono state ricevute ed il loro conteggio è stato rappresentato nel grafico 6.2 per osservarne l'andamento.

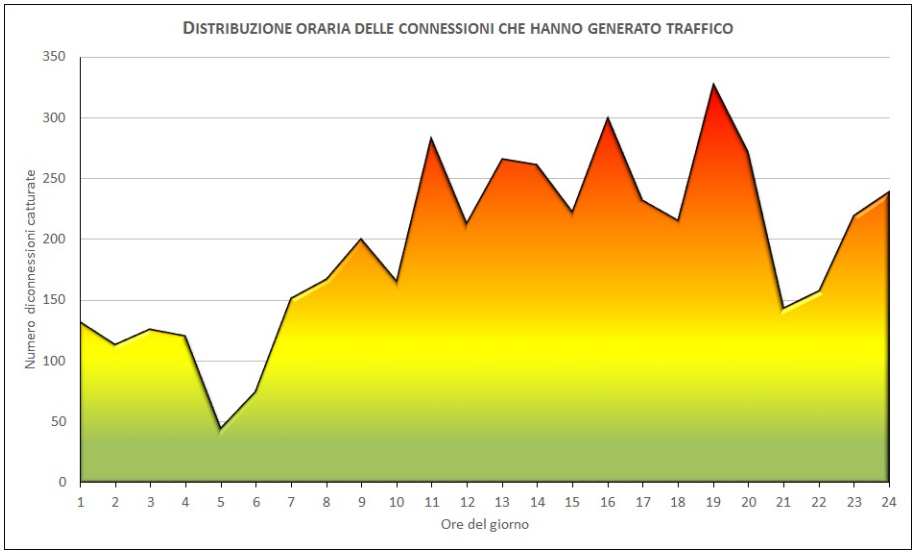


Figura 6.2: Distribuzione oraria delle sessioni nelle quali sono state catturate delle connessioni durante il periodo di esecuzione del software.

La distribuzione oraria rappresentata nel grafico fa riferimento al fuso orario italiano ed ogni fascia oraria rappresenta l'aggregazione delle connessioni catturate durante tutti i giorni di esecuzione del software.
Com'è possibile notare dal grafico, la maggior parte delle connessioni catturate si concentrano nelle ore pomeridiane, fascia oraria nella quale si ha una densità di connessioni che si mantiene costantemente elevata, con la presenza di sporadici picchi.
In aggiunta alla fascia pomeridiana, un picco di connessioni è stato rilevato alla mezzanotte.
In una seconda analisi, spostando l'attenzione sulle connessioni clusterizzate dalle sessioni, è stato possibile analizzarne alcuni spetti.
Tramite i parametri memorizzati nei file, è stato possibile suddividere le connessioni in base al numero di pacchetti di cui è costituito il loro traffico. La suddivisione è stata effettuata partizionando l'insieme delle connessioni in quelle che sono state interessate da un traffico costituito dal un solo pacchetto è quelle che invece si sono protratte per un numero superiore di questi.
Il risultato dell'analisi è osservabile nel grafico 6.3.

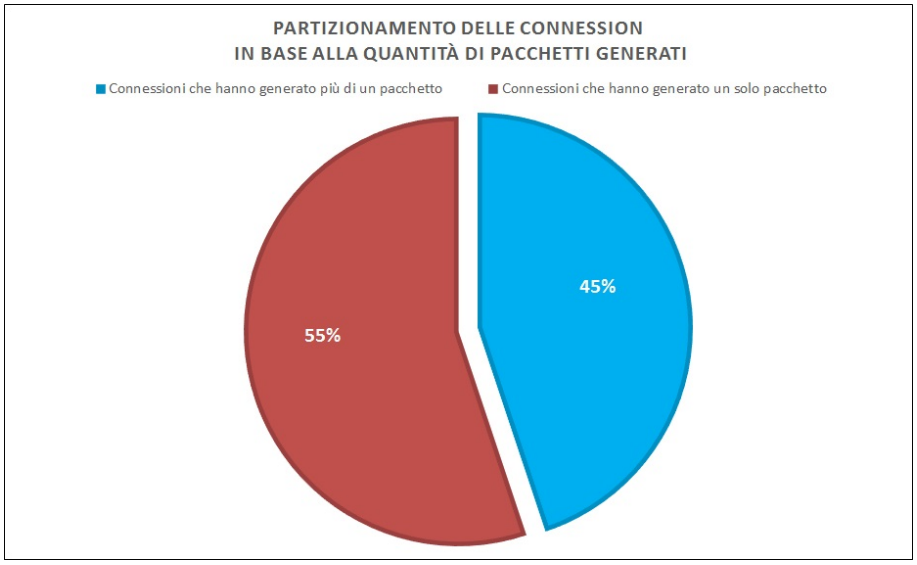


Figura 6.3: Distribuzione delle connessioni in base al numero di pacchetti generati dal traffico.

Il grafico mostra come poco più della metà delle connessioni siano state costituite solamente da un pacchetto al quale non è seguita altra comunicazione[2]. Di contro, il 45% circa delle connessioni ha proseguito la comunicazione dopo aver ricevuto risposta al primo pacchetto.
Accanto all'analisi del numero di pacchetti generati, è stato possibile effettuare anche una stima dei volumi di traffico di dati generato in ciascuna connessione. Il volume è stato calcolato utilizzando solamente gli effettivi dati generati dalla comunicazione, al netto degli header dei pacchetti.
Il grafico 6.4 rappresenta la sopraccitata distribuzione volumetrica delle connessioni catturate. Le fasce individuate partizionano le connessioni dimensionalmente indicando per ogni sottoinsieme il range di byte nel quale si collocano le connessioni.
Ogni fascia volumetrica è stata poi internamente suddivisa in base al fatto che la connessione abbia o meno registrato la ricezione del flag di FIN, indicatore della fine della connessione. In accordo con quanto specificato precedentemente, una comunicazione che avviene all'interno di una connessione tra due host non deve considerarsi necessariamente terminata non appena si

[2]Non è da considerarsi un evento eccezionale il fatto che non si sia ricevuta risposta al successivo pacchetto per più della metà delle connessioni catturate. Nel caso ad esempio dei pacchetti keep-alive descritti in precedenza, l'interlocutore che invia un pacchetto appartenente a questa tipologia, attende solamente una risposta senza alcun interesse nel a proseguire immediatamente la comunicazione. Esistono inoltre casi nei quali l'interlocutore che ha inviato il pacchetto catturato è in attesa di un input da parte del destinatario; in tali situazioni rispondere al pacchetto con una conferma di ricezione non ha alcun effetto sulla prosecuzione della comunicazione, la quale viene chiusa.

interrompe il flusso costituito dallo scambio di pacchetti. Ci sono infatti casi in cui ad esempio uno dei due host attende input da parte dell'altro interlocutore; in tali casi il flag di FIN non viene settato perché la comunicazione non deve terminare.

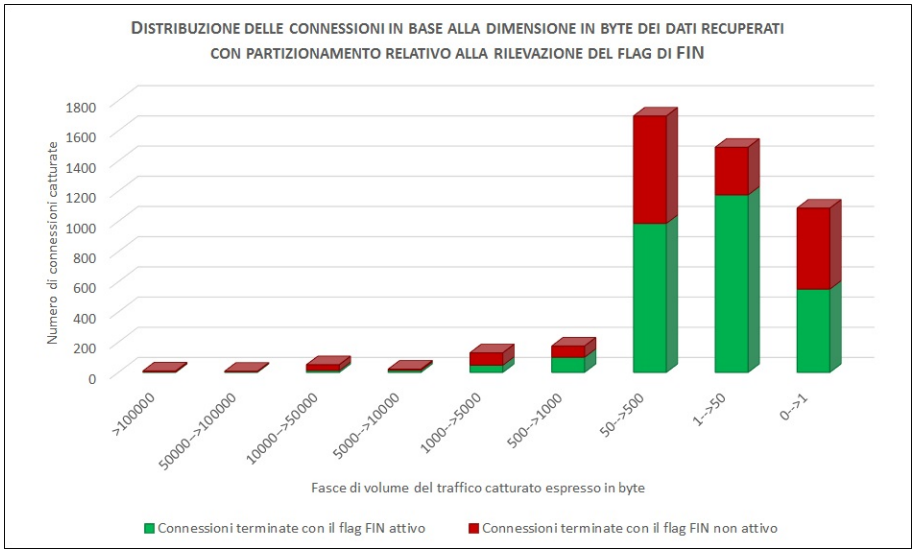


Figura 6.4: Distribuzione volumetrica delle connessioni per fasce di dimensione con suddivisione interna dettata dalla rilevamento del flag di FIN a fine connessione.

Osservando il grafico e coerentemente con quanto esposto nelle stime precedenti, si nota che una porzione considerevole delle connessioni catturate presenta dimensione del traffico di dati nulla. In questo segmento ricadono ad esempio le connessioni di tipo keep-alive.
Per quanto riguarda le connessioni rimanenti, a parte qualche sottoinsieme di connessioni che supera il megabyte di traffico, la maggior parte di queste si concentra nelle fasce che non superano i 500byte.
In merito al flag di FIN che identifica la fine della connessione, le stime individuano che la maggior parte delle connessioni sarebbero potenzialmente in grado di continuare in seguito all'immissione di opportuni input (ad esempio in un applicativo).
In un'ultima stima riguardo le connessioni ci si è occupati di studiare la suddivisione del traffico sulle differenti porte.
Il grafico 6.5 rappresenta per l'appunto la suddivisione percentuale del traffico in base alla porta cui esso era destinato di volta in volta in ogni connessione catturata.
Come avviene per le normali connessioni internet, a meno di casi persona-

lizzati, ogni porta è destinata a ricevere una specifica tipologia di traffico in base all'assegnazione[3] cui ciascuna è stata sottoposta.

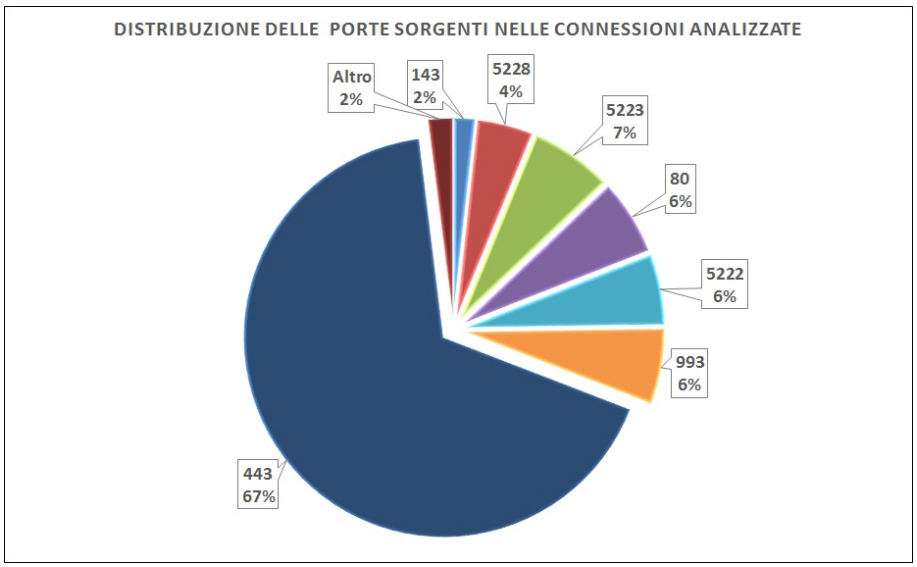


Figura 6.5: Distribuzione delle connessioni ricevute in base alle porte di destinazione.

Durante la fase sperimentale sono state catturate connessioni su 53 porte di destinazione differenti. Nel grafico 6.5 è possibile osservare la distribuzione delle porte in base al volume di connessioni indirizzate verso ciascuna di esse. Per favorire una migliore rappresentazione ed una più facile lettura, le porte destinatarie di un numero limitato di connessioni sono state raggruppate in un'unica categoria.
La maggioranza delle connessioni che sono state catturate durante lo svolgimento dell'esperimento, erano destinate alla porta 443 assegnata al traffico HTTPS. Il protocollo HTTPS costituisce la versione sicura del protocollo HTTP, il cui traffico viaggia in chiaro sulla porta 80. Le connessioni che seguono il protocollo HTTPS sono considerate sicure in quanto il traffico viene criptato e non è quindi direttamente leggibile da un eventuale utente in ascolto sul canale comunicativo. Data la natura del protocollo, nonostante la grande quantità di pacchetti di questa tipologia accumulati, non è stato possibile carpire da esse alcuna informazione.

[3]Le porte note sono le porte TCP e UDP nell'intervallo 0-1023 e sono assegnate a specifici servizi dalla IANA. I numeri delle porte registrate sono quelli nell'intervallo 1024-49151. I numeri di porta dell'intervallo 49152-65535 appartengono a porte private o dinamiche e non sono utilizzati da una applicazione in particolare. La IANA non impone questa suddivisione, è semplicemente un insieme di utilizzi raccomandati. Talvolta le porte possono essere usate per protocolli o applicazioni diverse dalla designazione ufficiale IANA.

Un'altra grossa fetta di dati catturati è stata quella destinata alla porta 993, utilizzata dal protocollo IMAP sicuro. Anche in questo caso ci si è trovati di fronte a traffico criptato che poteva essere letto direttamente.
Non tutto il traffico catturato è costituito da dati criptati, è stato infatti possibile talvolta catturare connessioni sulle porte 80 (HTTP), 143 (IMAP non criptato) o altre porte prive di un particolare protocollo. In questi casi è stato possibile in qualche occasione leggere in chiaro il traffico catturato.
Il fatto che le connessioni catturate siano per la maggior parte criptate sulla porta 443 non è da considerarsi una condizione anomala. Dal momento in cui è stato introdotto l'HTTP/2.0 nell'aprile del 2014, è di fatto resa obbligatoria la migrazione ad un sito HTTPS per poter sfruttare le nuove funzionalità messe a disposizione dal nuovo protocollo. Dato che i siti web i quali rimangono in formato HTTP sono supportati solamente dall'HTTP/1.0, si è fortemente accentuata questa tendenza migratoria diminuendo ma non azzerando la possibilità di catturare connessioni in chiaro.

Capitolo 7

Conclusioni

Riprendendo quanto affermato nei primi capitoli del presente elaborato, l'obiettivo dell'esperimento consiste nella dimostrazione pratica della possibilità di intromettersi nel traffico di altri utenti proseguendo la comunicazione.
In questo esperimento viene analizzata tale possibilità, espletando l'intromissione al momento della connessione alla rete mobile e di conseguenza all'assegnazione dell'indirizzo IP.
Attraverso le prove effettuate continuativamente durante i giorni di esecuzione del software è stata dimostrata la fattibilità di quanto prefissato come obiettivo. Sia l'osservazione del fenomeno durante l'esecuzione del software che l'analisi postuma dei dati ricavati alla fine di ogni intervallo di esecuzione hanno dimostrato che nel caso in cui ad un utente che effettua la connessione alla rete mobile venga assegnato un indirizzo IP che fino a qualche istante prima era assegnato ad un altro utente e nei confronti del quale è avvenuta una disconnessione forzata del canale comunicativo, è possibile protrarre eventuali comunicazioni chiuse forzatamente, continuando a ricevere lo stream di dati a partire dal punto di disconnessione.
Il fatto di poter continuare a ricevere il rimanente stream di dati è possibile grazie a due aspetti fondamentali:

- è necessario bypassare l'invio automatico del flag reset da parte del protocollo TCP implementato nella maggior parte dei calcolatori inibendo le operazioni automatiche della scheda di rete[1];
- i pacchetti di risposta a quanto ricevuto all'interno delle varie connessioni devono essere forgiati appositamente in maniera tale che dall'altro lato della comunicazione non venga percepita l'avvenuta sostituzione dell'interlocutore.

Rispettando le condizione appena elencate, l'esperimento è da considerarsi riuscito e l'obiettivo è stato quindi raggiunto. Ad avvalorare quanto afferma-

[1] Durante questo esperimento l'inibizione è stata implementata forzando il firewall a bloccare tutto il traffico sia in ingresso che in uscita.

to, sono presenti gli output generati dal software e memorizzati nei diversi file di log. All'interno di questi file è infatti possibile scorrere il traffico che è stato prodotto ed è quindi possibile osservare come l'interlocutore continui a rispondere al modem dopo che il primo pacchetto è stato catturato.
Accanto all'obiettivo primario di fattibilità e riuscita dell'esperimento, era nascosta la presenza di un altro sotto obiettivo direttamente dipendente dal principale. Sarebbe stata infatti desiderabile l'intercettazione di un pacchetto appartenente ad una connessione in chiaro al quale poter rispondere al fine di ottenere eventuali dati sensibili.
Sebbene siano state catturate un numero adeguato di connessioni in chiaro, in nessuna di queste è tata rilevata la presenza di informazioni particolarmente sensibili, il che ha causato un parziale fallimento del sotto obiettivo.
Concludendo, se da un lato la possibilità di catturare traffico appartenente ad altri utenti semplicemente connettendosi alla rete e rispondendo ai pacchetti che vengono passivamente ricevuti è un fatto provato ed una minaccia reale alla protezione dei dati, dall'altro la presenza nella maggior parte dei casi di traffico criptato, rende necessario il possesso di elevate competenze in ambito di decrittazione dei dati da parte di un malintenzionato che ha l'obiettivo di carpire informazioni sensibili attraverso la procedura qui esposta.
Non è da escludere infine che un funzionamento molto più prolungato del software presentato in questo elaborato possa materializzare la possibilità di ricevere informazioni interessanti in chiaro.

Bibliografia e sitografia

[1] Andrew S. Tannembaum, David J. Wetherall. *Reti di calcolatori, 5ˆ edizione*, 2011 Pearson Italia, Milano-Torino;

[2] RFC 791, *Internet Protocol (IP)*, http://www.ietf.org/rfc/rfc791.txt;

[3] RFC 793, *Transmission Control Protocol (TCP)*, http://www.ietf.org/rfc/rfc793.txt;

[4] RFC 1661, *The Point-to-Point Protocol (PPP)*, http://www.ietf.org/rfc/rfc1661.txt;

[5] Wikipedia http://it.wikipedia.org/;

[6] AT Commands Reference Guide, http://m2m.pp.fi/data/Telit_AT_Commands_Reference_Guide_r5.pdf.

Indice analitico