Università di Perugia
Dipartimento di Matematica e Informatica

Tesi triennale in informatica

Una Rassegna sui Canali Nascosti Temporizzati

Relatore
Prof. Francesco Santini

Candidato
Edoardo Tommasi

Anno Accademico 2023-2024

Dedicato a Marcello che, come una stella cadente, ha illuminato il cielo ovunque il suo sentiero si posasse. La stella si è poi spenta toccando l'oceano, la luce in cielo ormai estinta. Eppure, tra i racconti delle gracili conchiglie ebbre di sale e i canti di vecchie sirene erranti riesco ancora a vederla, dispersa nelle sfumature di blu.

Keep the light shining.

Contents

Capitolo 1

Introduzione

La steganografia è uno strumento estremamente potente che, sin dall'antichità, è stato utilizzato dall'uomo per nascondere informazioni segrete all'interno di dati apparentemente innocui. Già nei tempi antichi, tecniche rudimentali di steganografia erano impiegate per trasmettere messaggi segreti attraverso mezzi inaspettati come testi scritti, pergamene e persino nei disegni. La parola stessa, che deriva dal greco antico, significa "scrittura nascosta" a indicare l'obiettivo principale di questa pratica: celare l'esistenza del messaggio stesso. Con il passare dei secoli, e in particolare con l'avanzamento delle tecnologie, gli ambiti di applicazione della steganografia si sono moltiplicati esponenzialmente. L'avvento dell'era digitale ha aperto nuove frontiere, permettendo di nascondere dati all'interno di immagini, audio, video e persino nei protocolli di comunicazione di rete. Questo è possibile grazie alla capacità dei computer di manipolare e alterare i dati a livello binario in modo da renderli invisibili all'occhio umano, ma significativi dal punto di vista del contenuto macchina. Le applicazioni moderne della steganografia sono incredibilmente varie: nei file di immagini, le informazioni segrete possono essere integrate nei pixel senza alterare visibilmente l'aspetto dell'immagine; nel caso degli audio, i dati possono essere nascosti dato nelle frequenze non percepibili dall'orecchio umano; nei video, le informazioni possono essere disseminate tra i fotogrammi, rendendo praticamente impossibile individuarle senza gli strumenti adeguati.

Nel 2003, è stato introdotto il concetto di Network Steganography, un'innovazione che ha rivoluzionato il campo della sicurezza informatica, che utilizza le reti di comunicazione come mezzo per trasmettere informazioni nascoste. A differenza della

steganografia tradizionale, che inserisce dati segreti in file digitali come immagini o audio, la Network Steganography sfrutta i protocolli di rete per celare informazioni. Questo approccio è particolarmente efficace perché sfrutta la naturale complessità e la vastità delle reti di comunicazione moderne, rendendo estremamente difficile la rilevazione dei messaggi nascosti. Dato l'enorme volume di traffico dati e la grande varietà di protocolli e applicazioni, offrono molteplici punti di inserimento per informazioni nascoste.

La Network Steganography è diventata una delle tipologie più usate in ambito pratico. Le sue implementazioni prendono il nome di Canali Coperti o Covert Channel e possono essere utilizzate per bypassare le tradizionali misure di sicurezza, trasmettendo informazioni senza destare sospetti. Questi canali nascosti dato operano sfruttando risorse di comunicazione che non sono destinate alla trasmissione di dati segreti, eludendo così i meccanismi di controllo e sorveglianza standard. In un ambiente di rete controllato, dove tutto il traffico è monitorato, un Covert Channel può consentire la comunicazione non autorizzata tra due parti, sfruttando le peculiarità del protocollo di rete per celare i dati.

Esistono diversi tipi di Covert Channel, ciascuno con le proprie peculiarità e metodi di implementazione. Uno dei tipi principali è il Covert Storage Channel, che nasconde informazioni modificando i valori in una risorsa condivisa. Nei protocolli di rete vengono sfruttati i campi non utilizzati o opzionali all'interno degli header dei pacchetti di rete, i dati segreti possono essere inseriti in campi che non sono tipicamente monitorati, come il campo di identificazione in un pacchetto IP o i bit di flag in un pacchetto TCP. Un'altra possibilità riguarda l'uso delle etichette di controllo del flusso in pacchetti di dati, che possono essere modificate per trasmettere informazioni nascoste senza influenzare il funzionamento del protocollo. Questi metodi sono efficaci perché si integrano perfettamente nei normali flussi di traffico di rete, rendendo molto difficile la loro rilevazione.

Un'altra tipologia di Covert Channel, che sarà anche il focus di questa tesi, è quella dei Covert Timing Channel o canali temporizzati, che sfruttano le variazioni temporali nei protocolli di comunicazione per trasmettere dati. Il tempo di risposta variabile in un protocollo di rete può essere manipolato per codificare informazioni segrete: l'idea su cui si basa il Covert Timing Channel è che le informazioni non siano trasmesse attraverso i dati stessi, ma piuttosto attraverso il tempo intercorso tra l'invio di questi

dati. Ciò significa che un osservatore casuale potrebbe vedere un flusso di dati apparentemente innocuo, mentre in realtà ogni piccolo ritardo o anticipo nella trasmissione può rappresentare un bit di informazione nascosta, in modo particolarmente subdolo perché sfrutta il comportamento naturale e legittimo delle reti di comunicazione, rendendo molto difficile la rilevazione del canale nascosto. In una comunicazione HTTP, il tempo tra le richieste successive può essere deliberatamente modificato per rappresentare dati binari, con tempi più brevi indicanti uno 0 e tempi più lunghi indicanti un 1. Questa tecnica può essere applicata a vari protocolli di rete, inclusi TCP, IP, HTTP, ICMP e DNS, ciascuno dei quali offre diverse opportunità per variare i tempi di trasmissione. Inoltre, i Covert Timing Channel (ma anche i Covert Storage Channel) possono essere implementati non solo a livello di applicazione ma anche a livelli più bassi della pila di protocolli, come il livello di trasporto o il livello di rete, aumentando ulteriormente la complessità della loro rilevazione. L'uso dei Covert Timing Channel non si limita alla trasmissione di informazioni segrete; può anche essere impiegato per costruire sistemi di comunicazione resilienti che resistono alla sorveglianza e alla censura. Tuttavia questa stessa capacità li rende strumenti potenzialmente pericolosi nelle mani di attori malevoli che possono utilizzarli per esfiltrare dati sensibili o orchestrare attacchi coordinati senza essere rilevati. La difficoltà nella rilevazione dei Covert Timing Channel risiede nella necessità di distinguere tra variazioni temporali legittime, dovute a congestione di rete o altri fattori benigni, e quelle introdotte deliberatamente per nascondere informazioni. Di conseguenza la ricerca in questo campo si concentra non solo sulla creazione di tecniche sempre più sofisticate per la realizzazione di questi canali, ma anche sullo sviluppo di metodi avanzati per la loro rilevazione e mitigazione. La capacità di un Covert Timing Channel di operare sotto il radar della maggior parte delle tecnologie di sicurezza attuali lo rende una delle aree più critiche e affascinanti della steganografia di rete, rappresentando una continua sfida.

Comprendere i covert channel diventa dunque fondamentale per migliorare la sicurezza delle reti di comunicazione. Con l'aumento delle minacce cibernetiche e la crescente sofisticazione degli attacchi, è essenziale essere in grado di identificare e mitigare i metodi che gli attori malevoli possono utilizzare per nascondere le loro attività. L'esplorazione dei covert channel può portare allo sviluppo di nuove tecniche di protezione dei dati e di comunicazione sicura, contribuendo a costruire reti più

resilienti e affidabili. La conoscenza dei covert channel e delle loro implementazioni pratiche offre un vantaggio competitivo ai professionisti della sicurezza informatica, fornendo loro gli strumenti necessari per affrontare le sfide attuali e future.
In questa tesi andremo a trattare i covert channel e i protocolli che ne permettono l'implementazione con un focus sui covert timing channel. La tesi inizierà con una discussione dettagliata sui principali protocolli utilizzati dai covert channel: TCP/IP, DNS, HTTP, e ICMP. Ogni protocollo verrà descritto nel funzionamento e modalità di implementazione di covert timing channel. I covert channel saranno poi introdotti nelle loro caratteristiche fondamentali, nella tassonomia che li organizza e nelle diverse e possibili implementazioni. Concetto fondamentale sarà quello di dirimere la differenza tra la normale criptazione dei dati, che si occupa di occultare il contenuto ma non la comunicazione, e i covert channel, che invece occultano le comunicazioni nascondendole nella rete in chiaro. Saranno poi descritte le proposte presenti in letteratura di covert channel sperimentali e sistemi di difesa possibili che avranno come obiettivo quello di mitigarne la presenza e costruire reti più sicure e affidabili per gli utenti. Infine saranno descritte le possibili implementazioni e sistemi di rilevazioni presenti oggi in rete che effettivamente possono essere utilizzati dagli utenti per costruire una propria covert channel o per garantirne la rilevazione.
In particolare questo documento è così organizzato:

- *Capitolo 1*: l'introduzione corrente che offre una descrizione generica della tesi e di ciò di cui andremo a trattare;
- *Capitolo 2*: il background contenente le informazioni relative ai protocolli utilizzati dai covert channel e alla storia, ai tipi di steganografia, la terminologia utilizzata e la tassonomia degli stessi con un focus sui covert timing channel;
- *Capitolo 3*: contiene le proposte presenti in letteratura di covert timing channel con implementazioni sperimentali, strategie di difesa, applicazioni in contesti specifici;
- *Capitolo 4*: le implementazioni pratiche e i sistemi di rilevamento di covert timing channel presenti oggi in rete;
- *Capitolo 5*: la conclusione nella quale andremo a chiudere gli argomenti trattati nella tesi e ad introdurre possibili sbocchi futuri inerenti all'argomento.

Capitolo 2

Background

In questa sezione verrà introdotto e descritto l'argomento della "Steganografia" e dei "Covert Channel", con un focus sui "Covert Timing Channel", categoria che utilizza fattori di "Timing" ossia temporali per mascherare comunicazioni all'interno di una rete. Verranno anche descritti alcuni protocolli di rete per facilitare la comprensione dell'argomento.

2.1 La steganografia

Steganografia deriva dal greco ed è composto da due parole "στεγανοζ "e "γραφια", che significano rispettivamente "nascosto" e "scrittura". L'idea è quindi quella di andare ad occultare una comunicazione, di rendere ignota la sua esistenza [31]. Già nell'antichità esistevano tecniche di steganografia: in Grecia esistevano tavolette di legno incise e poi ricoperte di cera, sulla quale era riportato un messaggio falso che nascondesse il vero messaggio. Altra testimonianza è quella dello schiavo persiano a cui fu tatuato un messaggio sulla testa e, una volta ricresciuti i capelli, fu inviato a destinazione con il compito di tagliarseli e recapitare il messaggio. Anche la macchina Enigma utilizzata dalle forze naziste durante la seconda guerra mondiale può definirsi strumento steganografico, permise infatti ai tedeschi di occultare comunicazioni militari importanti dietro a finti comunicati di valore bellico irrilevante.

2.2 I protocolli utilizzati

Un protocollo è un insieme di regole per i formati di messaggi e procedure che consentono di scambiare informazioni. Sono regole che devono essere eseguite su ogni macchina coinvolta nella comunicazione e definiscono un metodo comune di comprensione dei messaggi. Per l'implementazione di canali coperti sono state pubblicate proposte che sfruttano tutti i protocolli di rete, a partire dal livello Applicazione fino al Data Link. I protocolli che generalmente vengono più utilizzati in questo ambito sono TCP [19], UDP [36], IP [10], DNS [27], ICMP [37] e HTTP [40]. Se da un lato i primi tre sono i più "forti" dato che la maggior parte del traffico di rete è generato da questi livelli, dall'altro sono richiesti i diritti di root per operare ai livelli superiori della pila ISO/OSI, cosa che rende meno possibile un'applicazione su vasta scala. DNS ed ICMP sono invece molto più sfruttabili dato che hanno un payload più facilmente modificabile e non richiedono diritti di root per avviare le comunicazioni. Esistono però anche moltissime misure di monitoraggio per questi due protocolli data la loro diffusione che sfruttano l'intelligenza artificiale per garantirne sicurezza ed efficienza. Infine abbiamo il protocollo HTTP che, dall'avvento di internet, è diventato sempre più diffuso ed utilizzato. Dato il grandissimo utilizzo è molto difficile trovare servizi di monitoraggio che riescano ad analizzare i pacchetti: principalmente vanno a focalizzarsi sulla presenza di anomalie nella comunicazione, ad esempio se il client invia più dati del server. Le implementazioni generalmente non sono di steganografia pura ma tecniche di tunneling che sfruttano HTTPS per ottenere una connessione che non solo sia nascosta ma anche sicura.

2.2.1 Rete TCP/IP

TCP/IP, come mostrato in Figura 2.1 sotto, è una famiglia di protocolli di comunicazione che vengono utilizzati per collegare sistemi di computer in una rete dove IP (Internet Protocol) fornisce un servizio di trasmissione dati senza connessione e TCP (Transmission Control Protocol) fornisce un servizio di trasmissione full duplex orientato alla connessione a livello pubblico. Il protocollo IP [10] fu sviluppato per l'uso in sistemi interconnessi per lo scambio di dati, chiamati datagram, da una sorgente ad una destinazione. Sorgenti e destinazioni sono identificate da un indirizzo di lunghezza fissa. Il protocollo implementa due funzionalità di base:

- *Indirizzamento*: vengono assegnati indirizzi trasportati nell'header internet per trasmettere datagram verso una specifica destinazione.

- *Frammentazione*: vengono usati dei campi nell'header internet per frammentare e riassemblare i datagram per il passaggio in reti "piccole".

Il protocollo TCP [19], dedicato alla connessione, si occupa di offrire servizi di collegamento affidabili tra coppie di processi all'interno di una rete. Si occupa dunque di offrire affidabilità negli scambi di informazioni lavorando a stretto contatto col protocollo IP, che invece si occupa di dare una destinazione e scompattare e ricompattare i datagram di dati. I dati all'interno di una rete TCP/IP sono divisi in:

- *Segmenti*: il protocollo TCP numera i pacchetti e provvede ad inserire in ogni pacchetto un header, che contiene le informazioni necessarie per realizzare il servizio.

- *Datagram*: il protocollo IP trasforma il pacchetto TCP in un datagram e inserisce un header, che contiene le informazioni necessarie (ad esempio l'indirizzo del computer di destinazione) per trasferire l'informazione attraverso la rete.

- *Frame*: dati preparati per il passaggio sul mezzo hardware di trasporto.

Una rete TCP/IP [23] può essere più facilmente compresa se suddivisa in termini di layer (o strati/livelli) come mostrato nella Figura 2.1. Dall'alto sono presenti i vari livelli:

- *Application layer*: rappresenta l'interfaccia con l'utente. Permette di consultare pagine web gestendo le sessioni tra client, browser e server web.

- *Transport Layer*: gestisce il controllo degli errori, i numeri di sequenza e il controllo di flusso per un collegamento attraverso una rete.

- *Network Layer*: si occupa delle route per i pacchetti di dati, del controllo per vedere se un server in un'altra rete è attivo e funzionante e dell'indirizzamento e la ricezione di pacchetti IP da altre reti.

- *Network Interface Layer*: ottiene i dati dai livelli superiori e li traduce in segnali da mandare attraverso un medium fisico.

- *Hardware*: mezzo di trasporto per i dati.

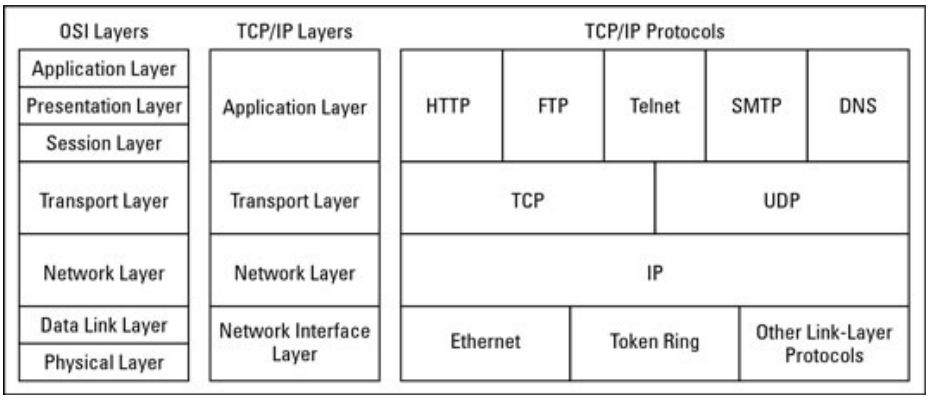

Figura 2.1: Suite Protocolli TCP/IP [14]

Il flusso dei dati, come mostrato in Figura 2.2, inizia al livello dei programmi applicativi che inviano messaggi ad uno dei protocolli di trasporto internet, UDP (User Datagram Protocol) o TCP (Transmission Control Protocol). Questi dati vengono divisi in segmenti con aggiunto l'indirizzo di destinazione e vengono dunque passati al livello successivo. A questo punto, nel livello di rete il segmento viene racchiuso in un datagramma IP che aggiunge l'intestazione (header) ed il trailer. Il datagramma viene passato dunque al livello di interfaccia di rete. Qui i datagrammi IP verranno trasmessi come frame su uno specifico hardware di rete come l'ethernet e mandati verso la destinazione. I pacchetti arrivati al destinatario passeranno dunque tutti gli strati di protocollo in reverse, come possiamo vedere in Figura 2.2, cioè dal basso verso l'alto: i frame vengono ricevuti dallo strato di interfaccia di rete (come un adattatore ethernet) che rimuove l'intestazione ethernet e invia il datagramma scompattato allo strato superiore. Nel livello di rete viene eliminata l'intestazione IP e il segmento viene mandato al livello di trasporto dove viene rimossa l'intestazione TCP/UDP. A questo punto i dati, arrivati al livello di applicazione, sono facilmente fruibili all'utente che ne ottiene una rappresentazione tramite interfaccia. Questo processo di costruzione/decostruzione dei dati viene fatto in contemporanea, gli host inviano e ricevono informazioni simultaneamente.

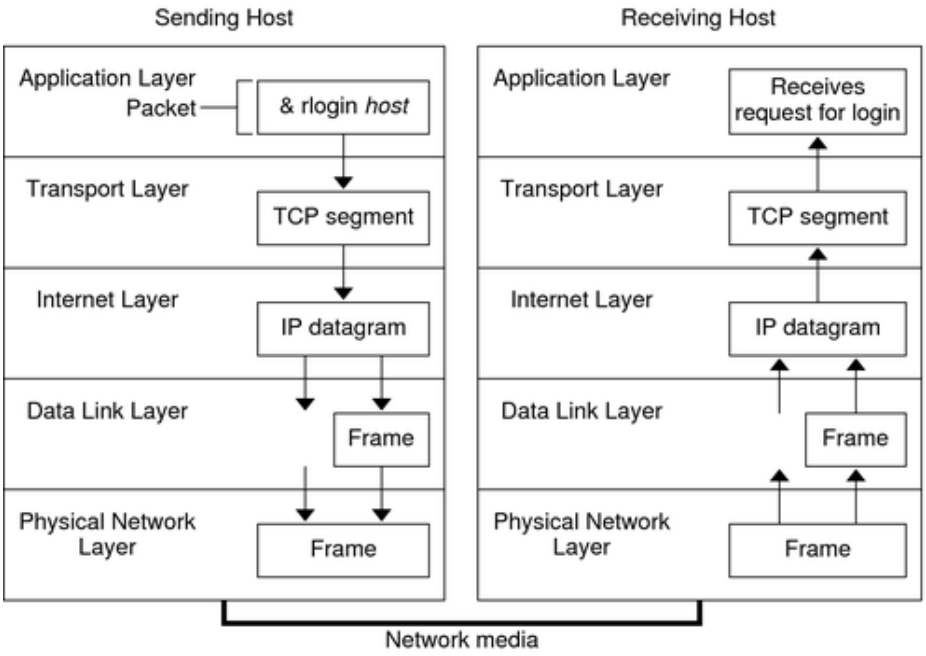

Figura 2.2: Movimento dei dati in una rete TCP/IP [30]

Diverse delle tipologie di Covert Storage Channel sfruttano proprio gli header TCP/UDP ed IP per iniettare bit nascosti all'interno della comunicazione in chiaro, bit che verranno poi ricostruiti dal ricevente in modo tale da decifrare il messaggio nascosto. Ovviamente questo genere di comunicazione è incredibilmente lento, vengono trasportati pochi kilobyte e byte al secondo e ciò è dato dal fatto che gli header trasportano informazioni minimali. Questo però è un vantaggio e al tempo stesso uno svantaggio, perchè se da un lato limita la velocità delle comunicazioni e la loro efficienza, dall'altro nelle comunicazioni in incognito bastano anche messaggi semplici per poter passare istruzioni più complesse che solo il ricevente può comprendere.

2.2.2 DNS

L'obiettivo del protocollo DNS [27] (Domain Name System) è garantire un meccanismo per nominare risorse di rete in modo tale che queste possano essere utilizzate da host, reti, organizzazioni. Garantisce quindi che, ad esempio, l'utente non debba interagire con IP, ma con dei nomi quando effettua ricerche su internet, semplificandone l'uso. Il protocollo si basa su un modello client/server dove vengono eseguite

richieste e risposte inerenti alle associazioni IP/Nomi di dominio. Vengono usati dei "demoni", ovvero programmi attivi in background che si occupano di svolgere le varie mansioni rimanendo sempre in ascolto.

- *Client*: Utilizza il demone "Resolver" per recuperare le informazioni associate ad un determinato nome di dominio, ad esempio l'indirizzo dell'host. Per effettuare le richieste vengono quindi passate delle query al resolver con il nome di dominio.
- *Server*: Utilizza il demone "Named" per passare al Resolver l'informazione oppure l'indirizzo di un altro "Name" server che possiede quell'informazione. Possono essere "Primary", nel caso in cui contengano l'informazione originale, o "Secondary", se l'informazione contenuta è una copia di un Primary server, o "Caching only" se l'informazione è contenuta nella macchina stessa per un accesso veloce.

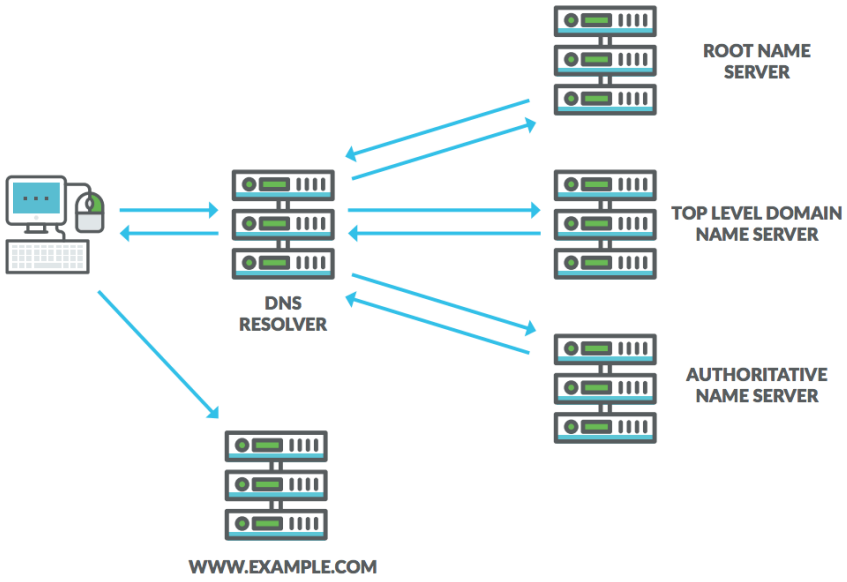

Figura 2.3: Funzionamento DNS [25]

Una richiesta DNS parte nel momento in cui l'utente va ad interagire, ad esempio, con un sito web. I siti su internet sono indicizzati con il loro nome, non con l'indirizzo

IP, questo perchè sarebbe improponibile per gli utenti ricordarsi ogni singolo indirizzo IP. Una volta che l'utente clicca sul link del sito web viene fatta una richiesta per tradurre il nome di dominio nell'effettivo indirizzo associato a quel dominio, in Figura 2.3 viene mostrato tutto il processo di funzionamento DNS. Il client dapprima controlla se questa "associazione" è già salvata nella cache locale, questo viene fatto normalmente per indirizzi già utilizzati e garantisce velocità maggiori. La cache locale però è limitata nello spazio e, nel caso in cui non dovesse contenere l'associazione adeguata, viene mandata una richiesta dal demone client Resolver al root server di riferimento, che esso sia organizzativo (.gov, .edu, .net, .org) o statale (.it, .uk, .fr e così via). A questo punto il root server risponde tramite i server primari di TLD (Top Level Domain), ovvero i primi in ordine gerarchico dopo il root server, per verificare che essi abbiano i file di zona con le associazioni necessarie. Le contengono? Allora viene mandata dal demone server Named una risposta al client che riceve l'associazione e si connette al server di dominio. Non le contengono? Allora vengono mandati al Resolver altri server (che essi siano sempre primari o secondari) a cui connettersi per trovare le associazione richieste. Questo processo fatto dal Resolver è ricorsivo e continua fintanto che non vengono trovati i dati. La Figura 2.4 mostra l'organizzazione semplificata dell'albero dns inerente ai domini .com e .edu e come vengono quindi gestite le richieste e le eventuali risposte per garantire un servizio funzionale e distribuito.

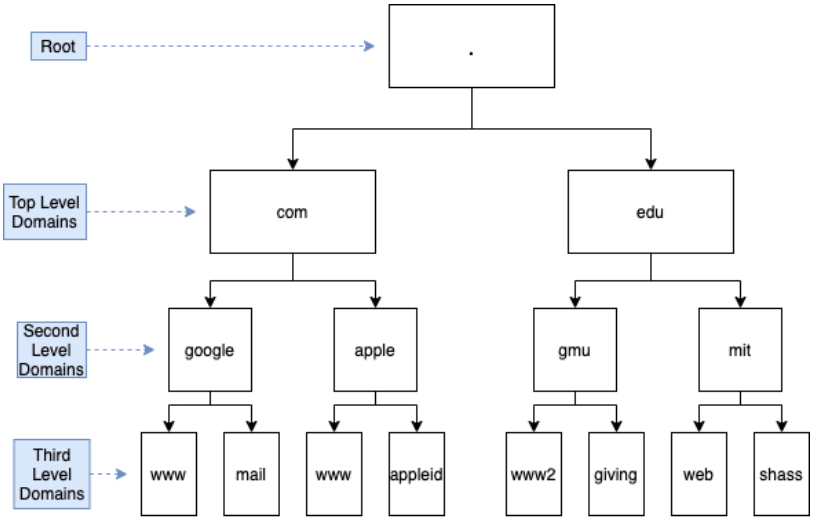

Figura 2.4: Albero DNS [5]

Anche il procollo DNS è ampiamente utilizzato nelle Covert Channel data la sua estensione e facilità d'uso. Normalmente vengono sfruttati Canali Temporizzati, nei quali vengono mandati messaggi da decodificare in base ai delay di invio come bit di 0 ed 1 che contengono messaggi nascosti. Un'altra tecnica ampiamente utilizzata è quella del tunneling: si va a dividere il messaggio in tanti piccoli pacchetti che si mascherano all'interno dei nomi di dominio delle query DNS, queste informazioni saranno poi decodificate dal destinatario.

2.2.3 HTTP

Il protocollo HTTP [40] (Hypertext Transfer Protocol), come mostrato in Figura 2.5, è un protocollo a livello applicazioni per i sistemi di informazione distribuiti, collaborativi ed ipermediali, che permette la comunicazione di dati attraverso Internet.

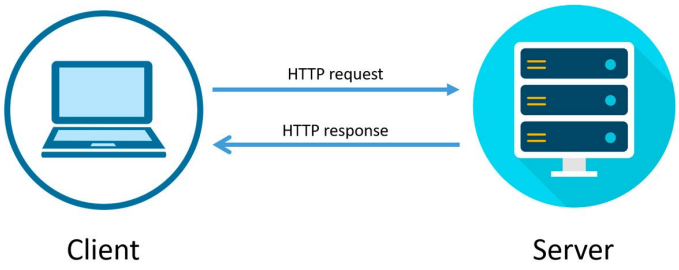

Figura 2.5: Funzionamento semplificato richiesta e risposta HTTP [9]

Nasce nel 1990 e negli anni è stato aggiornato a nuove versioni, mostrate nella Figura 2.6, che hanno aggiornato e aggiunto nuove funzionalità:

- *HTTP 0.9*: è la versione iniziale di HTTP. Non aveva alcun numero di versione inizialmente ma fu rinominata 0.9 per differenziarla dalle successive. Non aveva alcun tipo di header HTTP e venivano trasmessi solo file senza codici di errore o status. Era incredibilmente semplice: le richieste consistevano in una singola linea di testo che doveva iniziare con GET, unico metodo disponibile, seguito dall'indirizzo della risorsa. La risposta era il file stesso.

- *HTTP 1.0*: in questa versione furono introdotti il numero di versione, il codice di status della richiesta e gli header HTTP garantendo che anche i metadati

potessero essere trasmessi e rendendo il protocollo più flessibile ed estensibile. Finalmente anche documenti diversi da file HTML potevano essere trasmessi tramite l'header Content-Type.

- *HTTP 1.1*: la prima versione del protocollo ad essere standardizzata. Introdusse la possibilità di mantenere connessioni già stabilite, il pipelining, migliori controlli della cache, negoziazione del contenuto e la possibilità di "hostare" differenti domini dallo stesso indirizzo IP tramite l'header Host.

- *HTTP 2.0*: la prima versione del protocollo ad essere binaria e non più testuale. Introdusse il multiplexing, che garantiva connessioni multiple verso la stessa connessione, la compressione degli header e permise ai server di interagire con la cache del client.

- *HTTP 3.0*: l'ultima versione del protocollo in ordine di uscita, ancora non implementata a livello globale. Sostituisce il protocollo TCP con QUIC, ovvero una versione modificata di UDP che implementa il rilevamento della perdita di pacchetti e la ritrasmissione indipendente per ogni flusso di dati.

Nella Figura 2.6 sono presenti anche SPDY 1.0 e SPDY 2.0 che furono dei protocolli a livello applicativo per il trasporto dei contenuti web ideati da Google e da utilizzare in contemporanea con HTTP [16]. L'obiettivo era quello di ridurre la latenza e selezionare i file da condividere in modo tale da utilizzare una sola connessione TCP per client. Questi protocolli però saranno soppiantati da HTTP 2.0 che implementò la maggior parte delle loro caratteristiche.

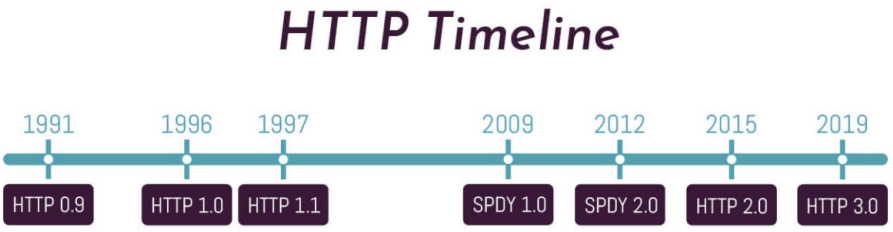

Figura 2.6: Timeline delle versioni HTTP [41]

2.2.4 ICMP

Il protocollo ICMP [37] (Internet Control Message Protocol), Nella Figura 2.7 vediamo come ICMP viene utilizzato in sinergia con il protocollo IP e si occupa di mandare messaggi di diagnostica della connessione. I messaggi ICMP vengono mandati per diverse motivazioni: quando un datagram non riesce a raggiungere una destinazione, quando il gateway non ha la capacità di poter mandare un datagram, quando esiste una strada più breve per mandare il traffico. Riassumendo, dato che protocollo IP non è affidabile in ogni casistica, c'è bisogno di ICMP per dare un feedback dei problemi che possono insorgere. Un esempio di messaggio largamente utilizzato è quello di "Echo Reply Message" ed "Echo Request" che servono a testare la connettività di una rete: viene mandata dalla macchina sorgente a quella di destinazione con l'intento di verificare se queste possono effettivamente comunicare e questo viene fatto in un processo messaggio/risposta.

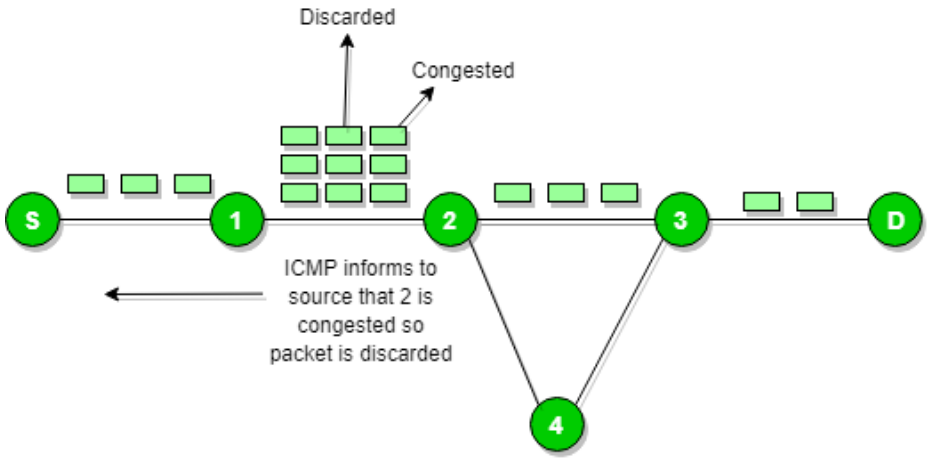

Figura 2.7: Funzionamento ICMP durante perdita di pacchetti [15]

Il pacchetto ICMPv4 mostrato in Figura 2.8 è composto dai seguenti campi:

- *Type*: descrive il tipo di messaggio in corso di invio così che il ricevente possa sapere cosa aspettarsi e come rispondere.
- *Code*: informazioni ulteriori sul messaggio di errore e sul tipo di errore.
- *Checksum*: viene utilizzato per garantire che il messaggio sia arrivato nella sua interezza, garantisce l'integrità dei dati.

- *Extended Header*: può contenere informazioni opzionali non incluse nell'intestazione standard.
- *Data/Payload*: qui vengono contenuti i dati effettivi da trasmettere, che possono essere un messaggio di rete con richiesta o risposta oppure di errore, varia in base alla situazione riscontrata.

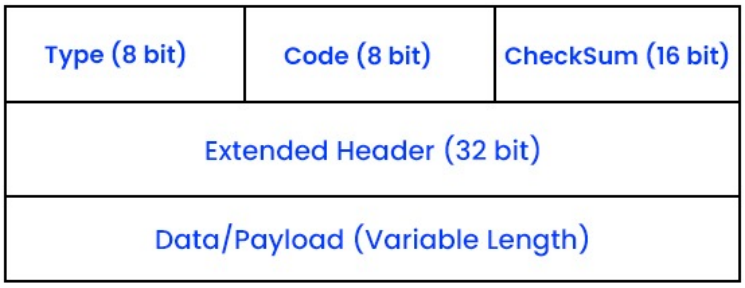

Figura 2.8: Composizione pacchetti ICMPv4 [15]

Tipologie di messaggi ICMP:

- *0 – Echo Reply*: risposta ad una richiesta Echo che può essere positiva o negativa se le macchine possono comunicare oppure no.
- *3 – Destination Unreachable*: utilizzato quando non è disponibile la destinazione, che essa sia host, rete, protocollo, porta e così via.
- *5 – Redirect Message*: reindirizza i datagram per la rete, per l'host e così via.
- *8 – Echo Request*: richiesta Echo per verificare se la comunicazione con una determinata macchina è possibile.
- *9 – Router Advertisement e 10 – Router Solicitation*: utilizzati per scoprire gli indirizzi di router operativi.
- *11 – Time Exceeded*: utilizzato quando il pacchetto supera il tempo di vita o i frammenti del pacchetto vengono riassemblati dopo il tempo di vita.
- *12 – Parameter Problem*: utilizzato quando ci sono errori di lunghezza, puntatore o pacchetto incompleto.

- *13 – Timestamp*: utilizzato per la sincronizzazione tra macchine.
- *14 – Timestamp Reply*: risposta al messaggio di Timestamp.

2.3 Covert Channel

2.3.1 Terminologia

Definiamo i termini che verranno utilizzati in questa tesi per facilitare la comprensione e la lettura della stessa. Quando parliamo di "canali segreti", "occultamento di informazioni", "steganografia" ci riferiamo sempre alla stessa cosa. La divergenza tra i termini è data dal fatto che, essendo questo un argomento che ha origini antichissime, negli anni è stato trattato in forme e modalità diverse, in epoche molto distanti [42].

- *Covert Channel (canale nascosto o segreto)*: si tratta dell'occultamento di informazioni nei protocolli di rete, attraverso il quale due peer riescono a scambiarsi in modo segreto dei dati.
- *Overt Channel (canale palese)*: un canale di comunicazione noto con il quale, attraverso traffico di rete legittimo, mittente e destinatario si scambiano informazioni.
- *Mittente nascosto (Secret Sender, SS)*: riserviamo l'aggettivo nascosto alle parti coinvolte nella comunicazione che si svolge attraverso il canale segreto. In questo caso particolare ci si riferisce a colui che invia i dati.
- *Destinatario nascosto (Secret Receiver, SR)*: è l'altra parte coinvolta nella comunicazione segreta, colui che riceve i dati o li estrapola grazie alla giusta interpretazione di alcune informazioni del traffico legittimo.
- *Mittente palese (Overt Sender, OS)*: come nel caso della comunicazione segreta, anche in quella palese vengono coinvolte due figure, e questa è colei che invia i dati nel canale palese.
- *Destinatario palese (Overt Receiver, OR)*: colui che riceve i dati dal canale palese, da sottolineare che SS e SR molto spesso non combaciano con OS e OR

e quindi questi ultimi non sono a conoscenza che la loro comunicazione viene sfruttata per altre operazioni da terze persone.

- *Warden*: colui che controlla e ha pieno accesso al canale di comunicazione e che rappresenta l'ostacolo da superare.
- *Traffico di copertura o legittimo*: la comunicazione tra OS e OR sfruttata per inserire al suo interno un canale nascosto.
- *Messaggio nascosto o segreto*: le informazioni segrete scambiate tra SS e SR attraverso il canale nascosto.
- *Steganografia*: tecnica generale di occultamento di informazioni in un contenuto.
- *Processo di incorporamento*: processo tramite il quale il mittente del canale nascosto inietta i dati segreti all'interno della trasmissione legittima.
- *Processo di estrazione*: processo svolto dal SR tramite il quale riesce ad estrapolare le informazioni segrete dal canale palese. Queste possono essere rappresentate fisicamente da bit oppure ottenute tramite la giusta interpretazione di alcune caratteristiche del flusso di dati come il tempo di inter-arrivo tra pacchetti: un intervallo più alto di una certa soglia prestabilita viene valutato come "1"e viceversa.
- *Stegokey*: chiave segreta scambiata tra SS e SR per svolgere i processi di incorporamento e estrazione dei dati nascosti nel canale segreto [35].
- *Stegobit*: il bit dei dati segreti che viene inserito nel pacchetto preso in considerazione.

2.3.2 Il problema del prigioniero

Quando si affronta il tema dei canali segreti è d'obbligo introdurre l'argomento con il problema del prigioniero, mostrato in Figura 2.9, teorizzato da Simmons nel 1983 [45], in quanto rappresenta il modello de facto di questo tipo di comunicazione. Immaginiamoci Alice e Bob, prigionieri che vogliono comunicare per escogitare la fuga, non possono però farlo comunicando liberamente perchè ogni loro parola è ascoltata dalla

guardia Wendy che, al primo segnale sospetto, li spedirebbe in isolamento. L'obiettivo è quello di comunicare tramite messaggi che sembrino innocui agli occhi di Wendy ma che nascondano al loro interno informazioni segrete. Queste informazioni saranno estratte dai messaggi apparentemente legittimi, che anche Wendy leggerà, tramite una tecnica che i due condividono.

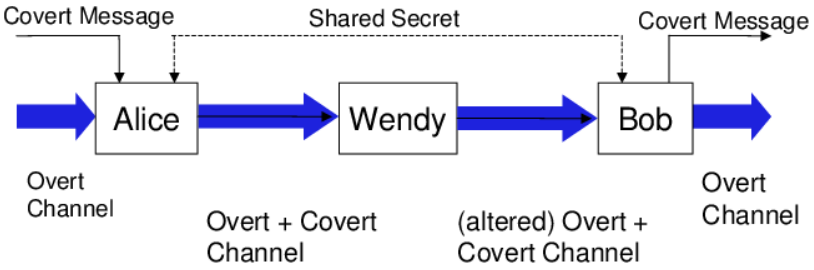

Figura 2.9: Problema del Prigioniero [14]

Nella Figura 2.10 troviamo il cifrario di Cesare, una possibile soluzione al problema che si basa sullo "shiftare" le lettere utilizzate nel messaggio di tre nell'alfabeto. La parola Hello, shiftando le sue lettere diverrebbe KHOOR:

- *H*: diventa K
- *E*: diventa H
- *L*: diventa O
- *L*: diventa O
- *O*: diventa R

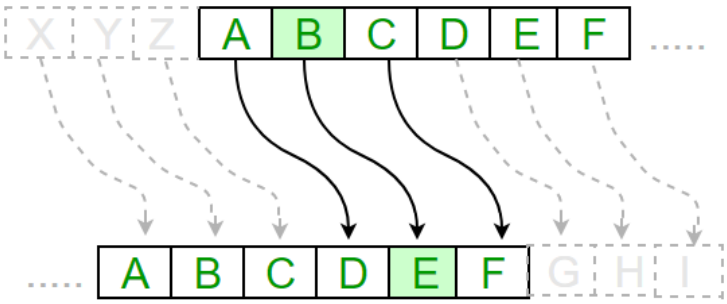

Figura 2.10: Funzionamento cifrario di Cesare [1]

Ovviamente andrebbe utilizzato un messaggio che, prima di essere shiftato, mantenga comunque un senso per Wendy che, altrimenti, potrebbe insospettirsi ugualmente. Ricordiamoci che quello che deve essere fatto non è pura criptazione, ma l'instaurazione di un Covert Channel. Una comunicazione può essere criptata nei suoi argomenti ma può comunque risultare palese che essa stia avvenendo, cosa che Bob ed Alice vogliono evitare.
Successivamente questo scenario è stato esteso alle reti di computer da Hendel nel 1996 [17]. Immaginiamo che Alice e Bob utilizzino due computer per scambiarsi messaggi: abbiamo quindi un canale di comunicazione palese che ne contiene uno nascosto. I due condividono dunque un segreto che gli permette di comunicare in maniera non palese. In questo caso Wendy sarà colei che gestisce e monitora la rete, in grado di poterla interrompere o alterare il flusso dei dati a suo piacimento nel momento in cui si accorge di qualcosa di sospetto. Alice e Bob possono anche essere la stessa persona oppure anche più persone in base alla situazione. Per quanto riguarda i possibili comportamenti di Wendy facciamo riferimento allo studio di Craver del 1998 che descrive i diversi tipi di avversari [8]:

- *Avversario passivo*: può solo spiare il canale ma non può alterare nessun messaggio.
- *Avversario attivo*: può modificare leggermente i messaggi, ma senza alterare la semantica.
- *Avversario maligno*: può alterare i messaggi senza incappare in conseguenze.

2.3.3 Le tipologie steganografiche

La steganografia viene impiegata principalmente per due scopi:

- *La protezione contro l'individuazione di dati segreti (data hiding).*
- *La protezione contro la rimozione di informazioni (document marking).*

Queste a loro volta si dividono in sottocategorie, troviamo infatti la steganografia iniettiva o generativa e la steganografia watermarking o fingerprinting. La differenza tra iniettiva e generativa è che nella prima il messaggio segreto viene "iniettato"

all'interno del traffico autentico mentre nella seconda il messaggio viene "generato" da zero cercando di rispettare caratteristiche di verosimiglianza. La steganografia digitale può, a sua volta, essere classificata in varie tecniche [22]:

- Text steganography.
- Image steganography.
- Audio steganography.
- Video steganography.
- Network steganography.

Quella che interessa a noi è la Network Steganography che verrà trattata nella prossima sezione.

2.3.4 Network stenography

Il concetto di steganografia di rete nasce da Krzysztof Szczypiorsky nell'anno 2003 [50]. Questo ramo è stato recentemente protagonista di uno sviluppo considerevole che ha portato alla nascita di molte e innovative tecniche di occultamento delle comunicazioni. Se prima infatti i principali supporti staganografici erano rappresentati da immagini e tracce audio, ora ci siamo spostati sui pacchetti e frame di dati che si spostano nella rete [4] [34]. Jun O. Seo afferma che la steganografia di rete ha tre vantaggi rispetto alla steganografia dei media digitali [22]: ha una migliore antirilevabilità, il ciclo di vita delle informazioni segrete è più breve e la larghezza di banda è più flessibile e non è limitata alla dimensione del file di copertura.

2.3.5 Le caratteristiche dei Covert Channel

I Covert Channel sono canali "coperti" che vengono adoperati per occultare informazioni all'interno del traffico legittimo, questo viene fatto per garantire uno scambio segreto di dati. Il termine nasce nel 1973 ad opera di Lampson [24]. C'è una netta differenza tra l'uso che si fa della crittazione dei dati e dei Covert Channel: se da un lato infatti abbiamo l'occultamento del contenuto dei dati (ma non della comunicazione, che è palese a tutti), dall'altro abbiamo un vero e proprio occultamento

della comunicazione. Perché occultare la comunicazione? Le motivazioni possono essere tante e variegate. Immaginate di essere spie, terroristi [28], cittadini in uno stato dittatoriale [39], virus informatici [52]. Se i vostri "avversari" dovessero anche solo venire a conoscenza dell'esistenza di una vostra comunicazione/interazione col mondo esterno, anche se criptata, questo potrebbe danneggiarvi in maniera irrimediabile. Dunque la crittazione dei dati non è sempre la soluzione migliore, serve anche nascondere la comunicazione ad occhi indiscreti. Questo però vale anche per chi si occupa di allestire e mantenere sistemi di cybersecurity [13]. Conoscere l'esistenza e le caratteristiche somatiche dei Covert Channel sta diventando sempre più fondamentale per poter garantire che i sistemi informatici siano sicuri, soprattutto in un mondo dove ciò che oggi è considerabile "protetto" domani potrebbe non esserlo più.

2.3.6 Le tipologie di canali

Possiamo andare a suddividere i Covert Channel in due principali categorie "semplificative" (secondo la proposta del 1985 [12]), ossia i canali di memorizzazione e quelli di temporizzazione. La classificazione è stata oggetto di ampie discussioni ed aggiornamenti nel corso degli anni (2007 [42], 2015 [46], 2016 [53], 2018 [54] e 2022 [47]) ad opera di Zander, Wendzel, Fechner, Herdin, Caviglione. La proposta del 1985 però si rivela perfetta per l'argomento di discussione di questa tesi in quanto ci permette di dare una suddivisione che descriva l'argomento in modo semplice ma completo e di focalizzarci sul tema cardine dei "Covert Timing Channel" che verranno descritti nelle prossime sezioni. Le tipologie dunque sono:

- *Covert Storage Channel*: modificano i bit dei pacchetti andando a memorizzare informazioni direttamente nel traffico.
- *Covert Timing Channel*: modificano la tempistica o il comportamento del flusso così che il destinatario possa decodificare informazioni segrete andando ad interpretare il traffico.

A loro volta queste due macro categorie possono essere suddivise in ulteriori sub categorie man mano sempre più caratterizzanti. Per avere una visione esemplificativa della tassonomia dei Covert Storage Channel si può utilizzare la Figura 2.11, le categorie sono state determinate durante le classificazioni citate precedentemente. Per

PDU (Protocol Data Unit) si intende l'unità di informazione o pacchetto scambiata tra due macchine in un protocollo di comunicazione.

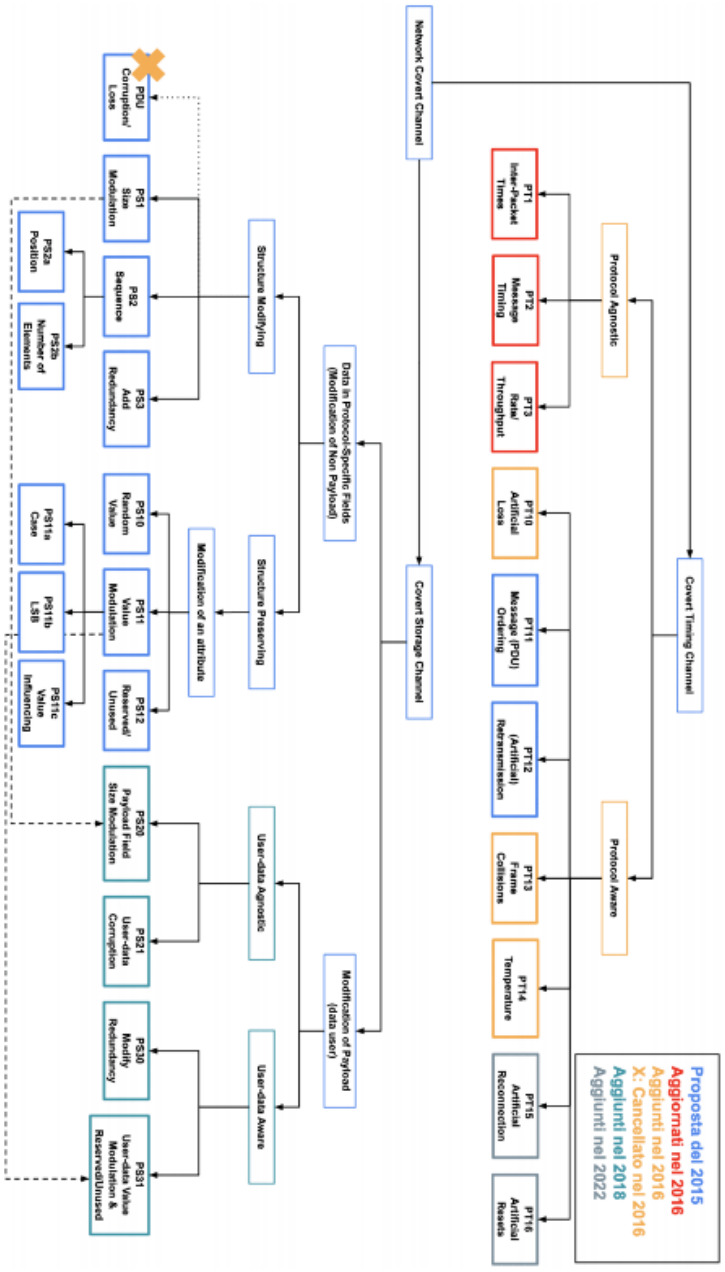

Figura 2.11: Flow chart per l'identificazione di Covert Channel [7]

La suddivisione finale delle tipologie dei Covert Storage Channel è la seguente:

- *Size Modulation*: il canale nascosto utilizza le dimensioni di un elemento di intestazione o di una PDU per codificare un messaggio nascosto.
- *Sequence*: il canale segreto altera la sequenza degli elementi header/PDU per codificare le informazioni nascoste. Questo modello si divide ulteriormente in: Position e Number of Elements.
- *Add Redundancy*: il canale nascosto crea nuovo spazio all'interno di un dato elemento di intestazione o all'interno di una PDU in cui nascondere i dati.
- *Random Value*: il canale nascosto incorpora i dati nascosti in un elemento di intestazione contenente un valore (pseudo-)casuale.
- *Value Modulation*: il canale nascosto seleziona uno degli n valori che un elemento di intestazione può contenere per codificare un messaggio nascosto. Questo schema si divide ulteriormente in: Case, Less Significant Bit (LSB) e Value Influencing.
- *Reserved/Unused*: il canale nascosto che codifica i dati nascosti in un elemento header/PDU riservato o inutilizzato.
- *Payload Field Size Modulation*: La dimensione del carico utile di un flusso viene utilizzata per codificare le informazioni nascoste (si tratta di un derivato di "Size Modulation", ma per il carico utile, poiché comporta la modifica del campo di lunghezza del carico utile di una PDU, ossia "Size Modulation").
- *User-data Corruption*: il canale di copertura esegue un inserimento (cieco) di dati nascosti nel payload di un flusso (simile a "Artificial Loss").
- *Modify Redundancy*: il canale di copertura comprime il carico utile di un flusso e lo spazio libero risultante viene utilizzato per nascondere i dati.
- *User-data Value Modulation and Reserved/Unused*: il canale di copertura esegue una modifica del carico utile di unflusso in un modo che non è riflesso da "Modify Redundancy" e che non risulta in un'interpretazione significativamente modificata dei dati, ad esempio modificando i bit meno significativi delle immagini digitali o nascondendo i dati nei bit inutilizzati/riservati del carico utile.

Stesso discorso vale per i Covert Timing Channel, questi possono dunque essere suddivisi in:

- *Inter-Packet Times*: il canale nascosto altera gli intervalli di tempo tra i messaggi di rete di un flusso (tempi di interarrivo) per codificare i dati nascosti.
- *Message Timing*: i canali nascosti sono codificati nella tempistica delle sequenze di messaggi all'interno di un flusso, ad esempio riconoscendo ogni n-esimo messaggio ricevuto o inviando comandi m volte.
- *Rate/Throughput*: il mittente del canale nascosto altera la velocità di trasmissione dei dati di un flusso da se stesso o da una terza parte al ricevitore nascosto.
- *Artificial Loss*: il canale nascosto segnala le informazioni nascoste attraverso la perdita artificiale dei messaggi trasmessi da un flusso, ad esempio tramite frame-corruption o caduta di messaggi.
- *Message Ordering*: il canale nascosto codifica i dati utilizzando un ordine sintetico dei messaggi in un flusso.
- *Retransmission*: il canale nascosto ritrasmette i messaggi precedentemente inviati o ricevuti di un flusso.
- *Frame Collision*: il mittente provoca collisioni artificiali di frame per segnalare informazioni nascoste.
- *Temperature*: il mittente influenza la temperatura hardware di un nodo terzo utilizzando il traffico di un flusso. Deve esistere una tecnica per la ricezione occulta per misurare la temperatura (indirettamente).

2.4 Timing channel

2.4.1 Le Caratteristiche dei Timing Channel

I canali temporizzati [43] si basano su un approccio che non va ad interagire direttamente con il contenuto di un flusso di dati, cosa che invece fanno i canali di archiviazione, ma che va a modificare caratteristiche del flusso stesso, come mostrato

in maniera semplificativa in Figura 2.12, come la frequenza di invio dei pacchetti ad esempio. Sta quindi a mittente e destinatario mettersi d'accordo su quale segreto utilizzare per trasmettere il messaggio nascosto all'interno della rete in chiaro. Riconoscere ed identificare canali temporizzati è molto difficoltoso, soprattutto all'interno di reti private, dove un utente mai si aspetterebbe possa crearsi una rete del genere, soprattutto perchè le metodologie di identificazione dei canali timing non sono infallibili, dato l'approccio statistico, c'è quindi sempre una probabilità che un canale timing non venga rilevato.

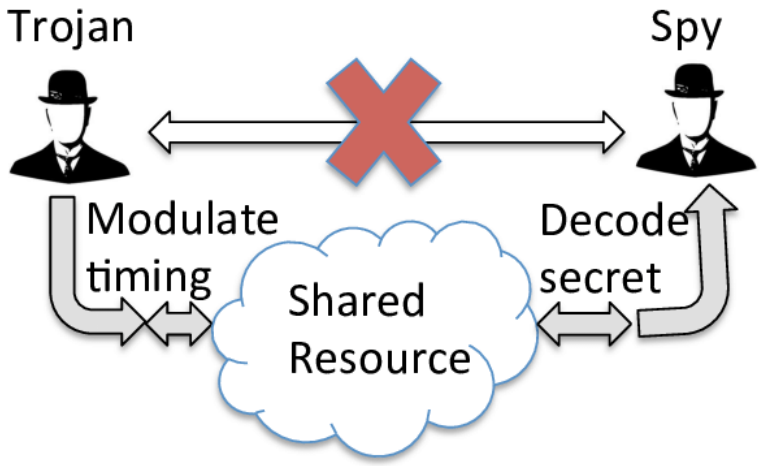

Figura 2.12: Funzionamento Timing Channel [20]

Nella Figura 2.13 vi è un esempio di Timing Channel che utilizza i tempi di inter-invio dei pacchetti per inviare messaggi nascosti. Il ricevente e il mittente si mettono d'accordo su uno specifico tempo di ritardo (o delay) e, ad ogni pacchetto inviato, chi manda può decidere di trasmetterlo con un determinato delay o subito. A questo punto il destinatario si occuperà di analizzare i tempi di invio tra un pacchetto e l'altro, quelli con un tempo che corrisponde al delay verranno interpretati come 1 mentre gli altri come 0. Oltre ai bit del messaggio vengono anche inclusi bit addizionali tra cui i bit di parità per controllare se ci sono stati errori di trasmissione e correggerli, bit di sincronizzazione (fondamentale nei timing channel) e bit di crittazione per mantenere ulteriore sicurezza. Il messaggio usato per la trasmissione è quindi suddiviso in blocchi di dati binari, che contengono al loro interno i dati, i bit

di sincronia, i bit di crittazione e quelli di parità. Una volta ottenuti i bit del messaggio, questo verrà poi decodificato da binario a carattere ed il destinatario otterrà l'informazione che il canale nascondeva.

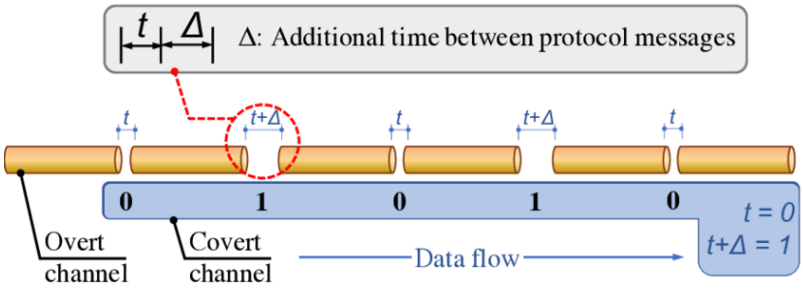

Figura 2.13: Funzionamento Timing Channel con tempi di inter-invio dei pacchetti [11]

Dato tutto quello che è stato detto finora, diventa di fondamentale importanza, non solo costruire sistemi che possano garantire la sicurezza degli utenti dall'esfiltrazione dei dati causati dai Covert Channel, ma anche studiare ed implementare canali coperti così da poterne analizzare e comprendere il funzionamento. Non solo, la ricerca sui Covert Channel non dovrebbe soltanto fossilizzarsi sui suoi usi maligni, ma anche su quelli benigni, perchè nessuna tecnologia è buona o cattiva di per sé, ma assume un determinato connotato in base agli usi che se ne fa. Difficilmente sarà possibile trovare una soluzione permanente all'uso maligno dei Covert Channel, anche perchè il settore dell'informatica è in continua evoluzione, eppure dovremmo comunque cercare di trovarne uno sbocco positivo dato che l'argomento cela possibilità a innovazioni tecnologiche interessanti [3].

Capitolo 3

La letteratura scientifica sui Covert Timing Channel

Qui sono contenuti articoli scientifici più o meno recenti sull'argomento, che ne delineano implementazioni, strategie di difesa, applicazioni in contesti particolari e mostrano quanto i Covert Timing Channel possano avere ampio respiro negli argomenti più disparati, oggi più che mai.

3.1 Implementazioni

Quì troviamo tutti quegli articoli che trattano di modalità di funzionamento o di applicazione o che concernono l'applicazione di Timing Channel.

3.1.1 Timing Channel su reti Wireless IEEE 802.11

Tra le implementazioni viene proposto un nuovo Canale Covert per le reti wireless che impiega lo standard wireless IEEE 802.11 (Wi-Fi) [51]. Quando si valuta un Canale Covert, lo si può esaminare lungo tre dimensioni differenti: rilevabilità, facilità di implementazione e tassi di trasferimento dati raggiungibili. La rilevabilità può essere valutata considerando l'osservabilità del canale ai diversi livelli di rete. Un canale osservabile solo al livello fisico più basso (livello 1) sarà più difficile da rilevare rispetto a un canale osservabile a un livello più alto come il livello di collegamento dati (livello 2). Questo è spesso inversamente correlato alla facilità di implementazione, poiché

i canali basati sul livello fisico che non sono osservabili ai livelli superiori possono richiedere cambiamenti sostanziali all'hardware o al software del livello fisico. Al contrario, le soluzioni pronte all'uso offrono un accesso limitato alle configurazioni di sistema ai livelli più bassi. Viene proposto di alterare il tempo di trasmissione di specifici frame di gestione IEEE 802.11 per fornire un Canale Covert. Questi parametri di temporizzazione sono accessibili nei prodotti commerciali presenti oggi sul mercato. Un'applicazione di questo studio potrebbe essere la distribuzione covert di materiale crittografico per una rete wireless. Il materiale crittografico potrebbe essere codificato utilizzando il Canale Covert e decodificato da un co-cospiratore a conoscenza del canale. Il co-cospiratore non ha bisogno di essere già connesso alla rete prima di utilizzare il canale poiché i frame di gestione vengono inviati in modo non criptato. Una volta ottenuto, il co-cospiratore può accedere alla rete wireless utilizzando il materiale crittografico acquisito. In un Canale Covert a bassa velocità di 20 bps, 256 bit di materiale crittografico (ad esempio una chiave AES trasformata) possono essere trasmessi a una velocità di circa quattro volte al minuto (una volta ogni 13 secondi). Quello che si crea dunque è un Covert Channel che può essere rilevato solo al livello fisico, dove l'informazione è nascosta negli intervalli di tempo tra le probe request frames o i beacon frames e dove la velocità è di 50 bps con un margine di errore del 5 per cento.

Covert Channel Proposto

L'informazione può essere codificata sia nel valore del tempo di arrivo stesso che nella differenza tra i tempi di arrivo successivi. Forniamo il seguente esempio del primo caso. Scegliamo otto diversi intervalli di tempo per rappresentare log2 8 = 3 bit alla volta. Scegliamo gli intervalli di tempo e la mappatura del codice gray mostrata in Tabella 3.1. Per inviare la sequenza di tre byte A1 B2 C3, divideremmo la sequenza in gruppi di 3 bit (101 000 011 011 001 011 000 011) e invieremmo richieste probe ai seguenti intervalli: 35 ms, 5 ms, 15 ms, 15 ms, 10 ms, 15 ms, 5 ms e 15 ms. Il tempo totale per la trasmissione è di 115 ms per un data rate effettivo di 208,7 bps.
Potremmo anche codificare i dati nelle transizioni tra due diversi intervalli di tempo. In questo caso, avremmo bisogno di 8 possibili transizioni per rappresentare 3 bit per transizione. Questo potrebbe essere ottenuto utilizzando 3 diversi intervalli di tempo poiché ci sono 32=932=9, diverse permutazioni di due scelte. Questo può

Symbol (Time interval)	**Data bits**
5 ms	000
10 ms	001
15 ms	011
20 ms	010
25 ms	110
30 ms	111
35 ms	101
40 ms	100

Tabella 3.1: Tempi di arrivo [51]

essere visto nella nostra mappatura rivisitata in Tabella 3.2. Ora, per inviare la sequenza di tre byte A1 B2 C3, invieremmo richieste probe ai seguenti intervalli: 15 ms, 5 ms, 5 ms, 5 ms, 5 ms, 15 ms, 5 ms, 15 ms, 5 ms, 10 ms, 5 ms, 15 ms, 5 ms, 5 ms, 5 ms e 15 ms. Il tempo totale per la trasmissione è di 135 ms per un data rate effettivo di 177,8 bps. Se utilizzassimo lo stesso numero di intervalli di tempo come in Tabella 3.1, raddoppieremmo il numero di simboli e quindi potremmo aggiungere un bit extra a ciascun simbolo. Mentre questo aumenterebbe il numero di bit per simbolo, aumenterebbe anche il tempo medio del simbolo. Tuttavia, questo approccio differenziale ha il potenziale di migliorare le prestazioni di rilevabilità.

Symbol (Time interval change)	**Data bits**
5 ms to 5 ms	000
5 ms to 10 ms	001
5 ms to 15 ms	011
10 ms to 15 ms	010
10 ms to 10 ms	110
10 ms to 5 ms	111
15 ms to 5 ms	101
15 ms to 10 ms	100

Tabella 3.2: Transizioni intervalli di tempo [51]

Operazione di Trasmissione e Ricezione

- *Trasmissione*: al trasmettitore, una mappatura tra i dati di input e l'intervallo di tempo è completata secondo una tabella di look-up simile a quella mostrata in Tabella 3.1 o Tabella 3.2. La configurazione dell'intervallo di trasmissione dei beacon o delle richieste probe viene quindi regolata di conseguenza. Per i router wireless tipici, questa impostazione può essere scelta in incrementi di 1 µs tra 1 µs e 1023 µs per i frame beacon.

- *Ricezione*: al ricevitore, il tempo di arrivo tra i successivi frame beacon o probe deve essere misurato. Questo può essere fatto utilizzando uno script che esegue un programma come tcpdump in cui il tempo di arrivo del frame è fornito e può essere confrontato con il tempo di arrivo del frame precedente per l'SSID in questione.

Prestazioni

Per esplorare le prestazioni del Canale Covert, osserviamo da vicino il primo esempio della sezione precedente. Se abbiamo un flusso di dati casuale, possiamo aspettarci che ogni simbolo (intervallo di tempo) sia rappresentato in modo equo. Ci vorranno, in media, 5760 ms (o 5.76 secondi) per inviare una chiave AES di 256 bit. Tuttavia, questo non fornisce alcuna protezione contro i frame di richiesta probe persi o ritardati, come mostrato in Figura 3.1. Se si verifica un errore di bit e un frame viene perso, due simboli di intervallo di tempo successivi appariranno come un singolo intervallo di tempo esteso. Ad esempio, se perdiamo il secondo frame di richiesta nel nostro esempio, registreremmo un intervallo di 40 ms invece di un intervallo di 35 ms seguito da un intervallo di 5 ms. In fase di decodifica, interpreteremmo questo come la sequenza di bit 111 invece della sequenza di bit 101 000. Per evitare che ciò accada, trasmettiamo ciascun intervallo di tempo più volte per formare un singolo simbolo. Questo ha l'effetto di ridurre il data rate effettivo. Nel nostro esempio iniziale, se trasmettiamo ciascun intervallo 10 volte, abbiamo aumentato il tempo totale di trasmissione di un fattore di 10 e ridotto il data rate effettivo a 20.87 bps. Inoltre, se la risoluzione dell'intervallo di tempo è troppo piccola tra i livelli, sarà difficile differenziare tra diversi simboli. I tempi di trasmissione effettivi dei frame non sono deterministici a causa dello schema di accesso al mezzo basato su contesa

utilizzato dallo standard IEEE 802.11 (Carrier Sense Multiple Access With Collision Avoidance, comunemente noto come CSMA/CA). Con l'aumentare della risoluzione per accomodare la variazione nei tempi di trasmissione dei frame, l'intervallo di tempo medio aumenta e, di conseguenza, la durata media del simbolo aumenta. Tornando ancora una volta all'esempio, se dobbiamo fornire 10 ms tra i valori degli intervalli di tempo, allora dovremo utilizzare intervalli di tempo di 10 ms, 20 ms, 30 ms, e così via fino a 80 ms. Questo risulta in un intervallo di tempo medio di 45 ms che effettivamente raddoppia il tempo medio di trasmissione e dimezza il data rate effettivo. Basandoci su queste osservazioni, le prestazioni sono una funzione sia del numero di volte in cui un intervallo di tempo deve essere trasmesso che della dimensione del passo minimo raggiungibile. Mentre il primo dipenderà dalla qualità del canale, il secondo dipenderà dai vincoli hardware e dal carico di traffico.

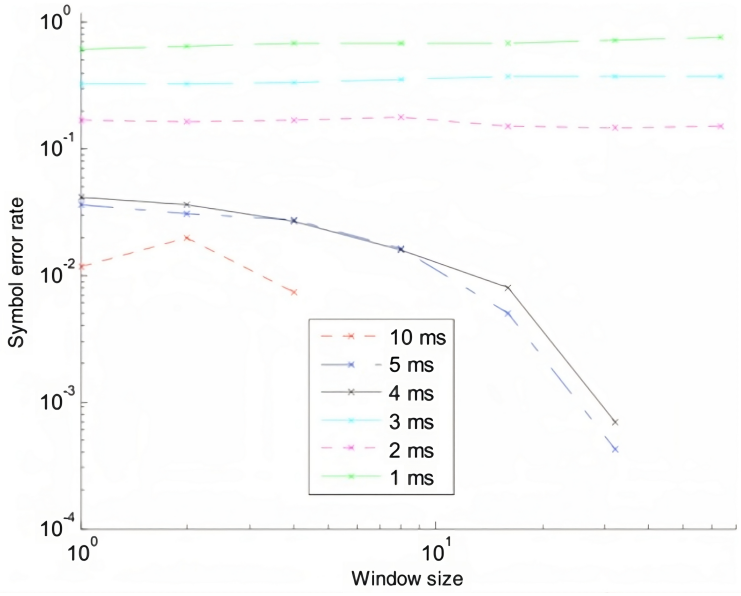

Figura 3.1: Rate degli errori dei simboli in funzione della window size per diversi intervalli [51]

Fuga di Informazioni

- Il frame di richiesta probe non contiene né un campo di intervallo né un campo timestamp, pertanto la potenziale fuga di informazioni è limitata all'identificazione del Canale Covert basata sulle misurazioni effettive dei tempi di arrivo dei frame

di richiesta probe al ricevitore a livello fisico (livello 1), dunque la variante di richiesta probe del canale è rilevabile solo al livello 1.

- Nel caso del frame beacon, l'intervallo del beacon e un timestamp sono inclusi come parte del payload nel frame. Per evitare che questo intervallo sia facilmente disponibile senza effettuare misurazioni effettive dei tempi di arrivo, i pacchetti utilizzati per creare il canale devono essere modificati prima della trasmissione. Il timestamp deve essere coerente con l'intervallo di tempo falsificato inserito nel campo di intervallo del beacon e il campo di intervallo del beacon deve essere impostato per essere l'intervallo di tempo medio (arrotondato al ms più vicino) per il Canale Covert implementato. Se questi campi non sono falsificati, il canale sarebbe rilevabile sia al livello 1 che al livello 2.

Risultati

Gli esperimenti sono stati condotti in un ambiente di un piccolo ufficio e i dati raccolti mostrati nella Tabella 3.3. Due laptop con adattatori wireless USB sono stati utilizzati negli esperimenti e sono stati posizionati a 28 piedi di distanza. Un laptop ha inviato frame beacon utilizzando Scapy e uno script Python ha permesso di impostare l'intervallo su base pacchetto per pacchetto. L'adattatore del secondo laptop è stato posizionato in modalità monitor per consentire la misurazione dei tempi di arrivo dei frame utilizzando l'applicazione tcpdump su Linux. L'adattatore wireless trasmittente era un Alpha 802.11 b/g/n Long Range USB adapter modello AWUS036NEH e non sono state fatte alterazioni all'implementazione off-the-shelf di nessuno dei due adattatori wireless. Per scopi di test, il trasmettitore ha ciclicamente trasmesso attraverso i simboli in Tabella 3.1 dieci volte separate per un totale di 80 simboli e 240 bit. Gli intervalli di tempo dei frame beacon ricevuti sono stati arrotondati all'intervallo di passo più vicino e un algoritmo K-nearest neighbors è stato eseguito per determinare il simbolo. In caso di parità, è stato scelto il simbolo precedente. I risultati sia per il tasso di successo della trasmissione dei simboli che per il data rate effettivo sono forniti in funzione della dimensione della finestra K-nearest neighbor e dei valori di ripetizione dell'intervallo di tempo (frame beacon successivi trasmessi per simbolo). I data rate superiori a 50 bps possono essere visti con tassi di errore inferiori al 6,5 per cento. Va notato che in questa implementazione specifica

del Canale Covert, un periodo di transizione dell'ordine di 15 ms era presente ogni volta che l'intervallo della richiesta probe veniva cambiato. Questo periodo di transizione era probabilmente un artefatto dell'hardware e del software specifico utilizzato nell'implementazione e aveva l'effetto di gonfiare il primo intervallo di tempo di un nuovo simbolo. Questo è stato affrontato dall'algoritmo K-nearest neighbor, ma ha l'effetto di ridurre il data rate raggiungibile poiché il primo intervallo di tempo in un nuovo simbolo sarà ora dell'ordine di 15 ms. In futuro, ulteriori ottimizzazioni del software e diverse configurazioni hardware del dispositivo di trasmissione potrebbero ridurre il periodo di transizione.

Scheme	Data rate (bps)	Error rate			Difficulties
		Window size 1	Window size 3	Window size 5	
Proposed w/2 frames per symbol	57.133	6.25% 1.43%	X	X	Low: Requires changes in configuration only
Proposed w/3 frames per symbol	39.271	0%	0%	X	Low: Requires changes in configuration only
Proposed w/5 frames per symbol	24.462	4.86% 1.17%	5.21% 0.90%	5.79% 0.91%	Low: Requires changes in configuration only
Ternary Channel [TAH15]	1620	<1%			High: Results presented for simulation only
Binary Channel [TAH15]	1023	<1%			High: Results presented for simulation only
Covert-DCF [RAD13]	1890	<1%			Medium: Requires writing to adapter specific register
Covert-DCF [RAD13]	1367	<1%			Medium: Requires writing to adapter specific register

Tabella 3.3: Dati del Covert Channel in un piccolo ufficio [51]

3.1.2 Autenticazione tramite Timing Channel

L'autenticazione è una parte fondamentale per mantenere la sicurezza dell'intera rete. Da quando Lamport ha proposto il sistema di autenticazione tramite password nel 1973, sono stati discussi vari tipi di schemi di autenticazione, come le firme digitali, le smart card, le password dinamiche e, recentemente, le caratteristiche biometriche e così via. Questi metodi sono frequentemente utilizzati nella vita quotidiana per confermare l'identità degli utenti. Allo stesso tempo, le tecniche avanzate di crittografia vengono applicate per proteggere le informazioni di identità crittografandole e trasmettendo il testo cifrato su Internet. Il sistema di autenticazione Kerberos, ampiamente utilizzato e proposto da Neuman come servizio di autenticazione per reti di computer, utilizza una serie di messaggi crittografati per dimostrare l'identità dell'utente. Zhang ha proposto uno schema di verifica batch basato sull'identità per le firme digitali, che è un metodo di autenticazione efficiente utilizzato nelle Reti Ad Hoc Veicolari. L'idea chiave di questi schemi si basa su segreti condivisi, ossia la parte che viene autenticata e l'autenticatore condividono gli stessi segreti. Tutti i segreti condivisi vengono trasmessi nella rete aperta dove un terzo malintenzionato può intercettare tutti i pacchetti di dati inviati e ricevuti. Pertanto, chi attacca può violare l'algoritmo di autenticazione monitorando un numero sufficiente di pacchetti di comunicazione; rubare e frodare l'identità intercettando la comunicazione e fingendo di essere l'utente/server autorizzato, come nel caso di attacchi di phishing e man-in-the-middle. I ricercatori propongono molte nuove idee per autenticare l'identità senza trasmettere pacchetti nella rete aperta, come il metodo di autenticazione proposto da Rick basato sulle latenze della digitazione, e il metodo di autenticazione proposto da Gamboa basato sulle informazioni comportamentali delle interazioni uomo-computer. I Canali Covert hanno sempre avuto come uso principale quello dell'esfiltrazione di informazioni. Nell'opinione comune, il TCC (Covert Timing Channel) è una minaccia nelle reti sensibili alle informazioni; ma nella seguente ricerca, viene applicato per trasmettere il messaggio segreto per l'autenticazione dell'identità. Viene proposto un nuovo metodo [58] di autenticazione dell'identità basato sui Timing Covert Channel, denominato autenticazione basata su TCC, in cui gli intervalli dei pacchetti vengono sfruttati per indicare i tag segreti. Il metodo proposto coinvolge due canali di comunicazione: il canale overt per il processo di comunicazione comune e il Canale Covert per il processo di autenticazione dell'identità. Il tempo di invio dei pacchetti

trasmessi nel canale overt viene manipolato dalla parte autenticata per indicare le informazioni di identità trasmesse nel Canale Covert, dove avviene la reale autenticazione e validazione dell'utente. È stata progettata una piattaforma FTP per verificare il metodo proposto, composta dal client e dal server: nel client, gli intervalli dei pacchetti sono controllati per presentare il tag di autenticazione, il lungo ritardo significa "1" e il breve ritardo significa "0"; nel server, viene registrato il tempo di arrivo di tutti i pacchetti per decodificare il tag di autenticazione. Gli esperimenti mostrano che gli intervalli e le soglie sono i fattori chiave per l'accuratezza e l'efficienza dell'autenticazione. Questo metodo si dimostra un modo più sicuro per autenticare, e potrebbe essere implementato in varie applicazioni di rete.

3.1.3 Timing Channel IP ad alta velocità

Nonostante il potenziale dei Canali Covert basati sulla temporizzazione IP, la loro implementazione risulta più difficile a causa dei rigorosi requisiti di tempistica e dell'instabilità della rete. Ad esempio, molti canali di temporizzazione IP esistenti, si basano sulla modulazione dei ritardi tra pacchetti, il che porta a problemi di bassa capacità del canale, alta percentuale di errori e bassa copertura. Anche se i canali esistenti hanno una bassa velocità di trasmissione e una cattiva copertura, gli amministratori di rete non dovrebbero supporre che la minaccia dei Canali Covert sia trascurabile. Per attirare l'attenzione dei ricercatori sulla necessità di tecniche di mitigazione più complesse, viene qui proposto [18] un nuovo framework per i Canali Covert ad alta velocità basato su temporizzazione IP nei protocolli TCP e UDP. In questo design, consideriamo un Canale Covert multi-percorso, dove diversi simboli in un messaggio covert prendono percorsi diversi. Adottando questo metodo, eliminiamo i ritardi tra pacchetti e miglioriamo la velocità di trasmissione. Inoltre, il traffico appare legittimo senza ritardi artificiali. Durante l'implementazione dei questo Canale Covert, la sfida maggiore è combinare correttamente le stringhe di codici provenienti da percorsi diversi sul lato ricevente. Durante questo processo ci sono molti rischi, come la perdita di pacchetti in UDP e il riordinamento dei pacchetti. Per evitare lo spostamento delle stringhe di dati durante la perdita di pacchetti, viene introdotta una tecnica di delimitazione per separare i pacchetti.

- *Framework*: vengono incorporati messaggi covert nella tempistica degli eventi

e trasmesse le stringhe di codice come traffico legittimo. I codici covert sono classificati e inviati a diversi percorsi in ordine, dove ogni percorso corrisponde a un indirizzo IP o una porta (o un processo). Il framework consente non solo punti finali del percorso regolabili, ma anche un numero regolabile di percorsi. A seconda dell'algoritmo di codifica, possiamo considerare di codificare diversi bit in un pacchetto. In questo caso, ogni percorso trasporterà una parola di codice. Ad esempio, se implementiamo un Canale Covert a 4 percorsi, qualsiasi pacchetto trasmesso attraverso il primo percorso simboleggia "00", qualsiasi pacchetto trasmesso attraverso il secondo percorso indica "01", e le parole di codice "10" o "11" sono inviate rispettivamente attraverso il terzo e il quarto percorso. Possiamo ulteriormente espandere il nostro Canale Covert multi-percorso utilizzando diversi metodi di codifica, che aggiungeranno più robustezza e randomizzazione.

- *Codifica e Trasmissione*: analizziamo un caso semplice considerando un Canale Covert a due percorsi, dove un pacchetto ricevuto attraverso un percorso indica un bit "1" nel messaggio covert e un pacchetto ricevuto attraverso l'altro percorso indica un bit "0" nel messaggio covert. Per inviare "0", un pacchetto è inviato al Percorso0 e per inviare "1", un pacchetto è inviato al Percorso1. Non c'è un intervallo di tempo specifico tra l'invio dei pacchetti, il che aggiunge più randomizzazione al canale.

- *Decodifica*: dal lato del ricevitore, se un pacchetto è ricevuto dal Percorso0, indica "0" e se un pacchetto è ricevuto dal Percorso1, è decodificato come "1". Poi il ricevitore unisce le informazioni dal Percorso0 e dal Percorso1 e ordina i pacchetti in base al loro tempo di arrivo.

3.1.4 Android

Dal 2010, gli smartphone, che contengono un gran numero di informazioni personali private, come contatti, foto, e così via, facilitano notevolmente la vita quotidiana delle persone. Tuttavia, grazie ai numerosi sensori utilizzati negli smartphone, è possibile costruire facilmente un Canale Covert, rendendo quindi gli smartphone una grande minaccia per la privacy. Il lavoro presentato in [44] introduce la modellazione e la classificazione del Canale Covert, concentrandosi poi sul principio di costruzione del

Canale Covert tipico su Android e sul metodo di rilevamento mirato ai Canali Covert su Android. Inoltre, questo articolo cerca di fornire una direzione generale e un riferimento utile per lo studio dei Canali Covert su Android.

Covert Channel ed Android

- *Soundcomber*: viene proposto nel 2011 ed è un sistema capace di estrarre informazioni utili dal sensore audio del telefono utilizzando permessi minimi [38]. Soundcomber può essere avviato automaticamente dal sistema operativo Android. Quando un utente effettua una chiamata, Soundcomber registra la conversazione telefonica in background, senza alcun avviso all'utente. Quando un utente chiama una hotline di servizio, come una banca, è molto probabile che senta messaggi di benvenuto come "Grazie per aver chiamato xxx banca". Utilizzando la tecnologia di riconoscimento vocale, Soundcomber può ottenere il breve messaggio di testo, confrontarlo con un database di hotline di servizio memorizzato internamente, quindi acquisire il numero di telefono chiamato senza accedere alla lista delle chiamate tramite il permesso di lettura dei dati di contatto. Una volta acquisito con successo il numero di telefono chiamato, Soundcomber utilizza l'algoritmo di Goertzel per riconoscere i toni DTMF nella elaborazione del segnale digitale. Successivamente, mediante un'analisi contestuale con profili mirati, Soundcomber può ottenere dati sensibili in modo intelligente, come numeri di carta di credito e PIN. Grazie alla sua forte capacità di nascondersi e alla combinazione con la tecnologia di riconoscimento vocale attuale, Soundcomber può ottenere comunicazioni dell'utente con precisione. La registrazione può essere disabilitata vietando la registrazione di sistema, ma a volte la registrazione di sistema è necessaria per gli utenti.

- *TouchLogger*: un sistema che utilizza la tastiera dello smartphone per dedurre il tasto numerico premuto [6]. Quando l'utente digita diversi tasti sullo smartphone, il telefono vibra in modo diverso e produce diversi dati di accelerazione sugli assi x, y e z. TouchLogger utilizza principalmente il sensore di accelerazione per ottenere lo spostamento della posizione e le informazioni di accelerazione generate dalla vibrazione del telefono quando l'utente inserisce numeri. La posizione del dito premuto può essere dedotta in base a come il telefono è po-

sizionato durante la digitazione, combinata con la posizione del tasto numerico sulla tastiera virtuale e l'abitudine di input degli utenti. Mediante la tecnologia di data mining, è possibile dedurre ciò che l'utente sta digitando.

- *WCTC*: regola la frequenza della CPU del sistema operativo Android [26]. Il traffico di rete del dispositivo Android è strettamente correlato al carico della CPU che esegue l'applicazione mobile, quindi i dati possono essere inviati come segue: collegare il dispositivo Android allo strumento adb shell, quindi impostare la frequenza della CPU al valore massimo, 100 pacchetti inviati significano 1 binario, pausa per un secondo; reimpostare la frequenza della CPU al valore minimo e 100 pacchetti inviati significano 0 binario, pausa per un secondo. Ripetere il processo. Il server calcola la differenza di tempo tra 100 pacchetti consecutivi uno per uno, ne fa la media, confronta il valore medio con una soglia predefinita: il valore medio dell'intervallo di pacchetti sopra la soglia rappresenta 1 binario, e il valore sotto la soglia rappresenta 0 binario, in questo modo l'informazione del mittente può essere decodificata. L'accuratezza della decodifica ottenuta dagli esperimenti sul protocollo TFTP era del 95,5 per cento, il che ha pienamente dimostrato la fattibilità della capacità di trasmissione di informazioni covert di WCTC. Tuttavia, la rilevazione di WCTC è relativamente facile poiché modifica solo l'intervallo di tempo tra i pacchetti. Per distruggere WCTC, è necessario modificare la velocità dei pacchetti invadendo l'intera rete.

- *Ultrasuoni*: il suono può essere ricevuto dallo stesso dispositivo tramite il microfono o altri dispositivi. Il suono creato da vibrazioni a breve termine può bypassare molti meccanismi di rilevamento di sicurezza per comunicare.

- *Vibrazioni*: utilizzando un motore vibrante su un dispositivo mobile per codificare il segnale letto dall'accelerometro dal malware, è possibile ottenere un effetto significativo sui dati accelerati, che vengono poi ricevuti e decodificati da un'altra applicazione.

- *Consumo batteria*: è possibile costruire un Canale Covert monitorando il consumo energetico del dispositivo mobile. Modificando l'uso della batteria in modo specifico, il malware può trasmettere dati codificati in base ai cambiamenti nei livelli di consumo energetico, che possono essere letti da un'altra applicazione

con accesso ai dati della batteria.

- *Sensore magnetico*: i sensori magnetici integrati nei dispositivi mobili possono essere utilizzati per creare Canali Covert. Variando il campo magnetico nelle vicinanze del dispositivo, il malware può inviare segnali che vengono rilevati e interpretati dal sensore magnetico, permettendo così la trasmissione di informazioni.

- *Canale vocale*: un altro metodo per la trasmissione di dati covert è l'uso del canale vocale delle reti cellulari. Il malware può modulare il segnale audio durante le chiamate vocali per incorporare dati nascosti, che possono essere estratti e decodificati dal ricevente con le tecniche appropriate.

Contromisure ed Android

- *Domain Isolation*: proposto nel 2011 tramite app TrustDroid [48]. TrustDroid assegna un livello di fiducia a un'applicazione prima dell'installazione. Durante l'installazione, le applicazioni devono superare il controllo del certificato del proprio produttore. Per quanto riguarda la comunicazione di rete, TrustDroid aggiorna la strategia del firewall per prevenire connessioni da reti non affidabili, isolando quindi la comunicazione tramite un proxy di rete. Durante l'esecuzione del sistema operativo Android, TrustDroid monitora tutte le comunicazioni delle applicazioni e l'accesso ai database pubblici condivisi, al file system e alla rete, impedendo lo scambio di dati tra diversi domini e comunicazioni tra applicazioni. Rispetto alle soluzioni già esistenti, TrustDroid realizza un isolamento a livello di middleware, kernel e rete dello stack software Android.

- *Privilege Escalation Attack*: il sistema operativo Android impedisce alle applicazioni di divulgare dati limitando i loro permessi e isolandole tra loro. Tuttavia, le applicazioni possono fare affidamento su altre applicazioni per ottenere indirettamente permessi per eseguire operazioni non autorizzate, ossia per elevare i propri permessi. L'idea generale è che se l'App 1 non ha il permesso P ma l'App 2 sì, l'App 1 può acquisire il permesso di P con l'aiuto dell'App 2. Nel 2011 è stato proposto un framework chiamato XMandroid per rilevare l'escalation dei privilegi nelle applicazioni Android [49]. XMandroid monitora la comunicazione tra tutti i componenti alla base del sistema. Per più applicazioni che

vengono eseguite nel sistema per la prima volta, la relazione viene stabilita utilizzando la teoria dei grafi. Successivamente, tutti i percorsi di trasmissione dei permessi vengono cercati tramite backtracking e ricerca in profondità, e spetta a XMandroid determinare se c'è un attacco di escalation dei privilegi in base alle regole di sicurezza definite nella politica del sistema. Quando queste applicazioni vengono eseguite nuovamente, DecisionMaker in XMandroid troverà efficacemente la soluzione precedente. Se sì, eseguilo direttamente; in caso contrario, riconsideralo. Per le regole di sicurezza delle applicazioni generali, sono vietate le autorizzazioni per accedere sia ai contatti che a Internet, nonché le autorizzazioni per chiamare, registrare e connettersi a Internet. Quest'ultima può essere utilizzata per rilevare Soundcomber.

- *Driver Permission Detection*: a causa della natura open source del sistema Android, i produttori di telefoni Android hanno effettuato sviluppi personalizzati per soddisfare le proprie esigenze. Tuttavia, oltre il 60 per cento delle vulnerabilità delle applicazioni provengono dagli sviluppi personalizzati dei produttori. ADDICTED è un'applicazione che correla le operazioni del dispositivo sensibile alla sicurezza con il loro file Linux, quindi li confronta con i file corrispondenti dal sistema Android open source [55]. In base all'abbassamento dell'autorizzazione del file del driver, ad esempio, da solo lettura a lettura e scrittura, ADDICTED può sapere se esiste una minaccia alla sicurezza. I problemi reali scoperti da ADDICTED su Galaxy SII, Galaxy ACE e Galaxy GRAND hanno dimostrato la validità di questo approccio.

- *Artificial Intelligent Detection*: è stato proposto un metodo di rilevazione tramite intelligenza artificiale nel 2017 basato sul consumo energetico del telefono cellulare per rilevare i canali nascosti [56]. Vengono proposti due metodi di rilevazione: Rilevazione Basata su Regressione (RBD) e Rilevazione Basata su Classificazione (CBD).

Sebbene siano stati trovati alcuni metodi di rilevamento efficaci, sono ancora troppo specifici. XMandroid e il metodo di intelligenza artificiale di Caviglione possono rilevare solo il canale nascosto nelle applicazioni colluse. Pertanto, manca ancora una sistematica e completa metodologia di rilevamento.

3.1.5 Influenza della rete su Timing Channel

La tesi in [33] discute sull'influenza che ha una rete nel mondo reale sulla qualità di un Canale Covert Temporizzato. Per un Canale Temporizzato Covert, tutti i messaggi vengono decodificati in base al tempo di arrivo dei pacchetti sul lato ricevente. Un ritardo inter-pacchetto entro un certo intervallo indica un bit alto, e altri intervalli di arrivo indicano un bit basso. Nell'ambiente di rete reale, il mittente e il destinatario possono perdere sincronizzazione, rendendo il Canale Temporizzato Covert inutile. Pertanto, la sincronizzazione tra mittente e destinatario è un problema chiave per l'accuratezza della decodifica. Molti problemi come jitter, ritardo di coda e ritardo di elaborazione possono causare la perdita di sincronizzazione tra il mittente e il destinatario. In altre parole, le condizioni di rete possono influenzare la qualità della trasmissione del Canale Temporizzato Covert in un ambiente di rete reale. Vengono quindi introdotte metodologie di valutazione e verifica della qualità della rete per l'uso di Covert Channel. Quando la rete è occupata, la perdita di pacchetti e il ritardo si verificano frequentemente, e molte delle informazioni nascoste trasmesse attraverso il canale nascosto risultano errate. Quando la rete è libera, la perdita di pacchetti e il ritardo si verificano occasionalmente, e la maggior parte delle informazioni nascoste possono essere trasmesse con successo tra il mittente e il destinatario. Quando la condizione della rete è occupata, il tasso di perdita dei pacchetti è dell'11.38 per cento e il tasso di errore di bit è del 29.4 per cento. Quasi un quarto dei bit nascosti sono trasmessi in modo errato; quando la condizione della rete è libera, il tasso di perdita dei pacchetti è dello 0.4 per cento e il tasso di errore di bit è del 2.8 per cento, una qualità di trasmissione accettabile per il mittente e il destinatario.

3.1.6 Comunicazione subacquea

A causa dell'apertura e dell'inaffidabilità dei canali di comunicazione acustica sottomarina, possono essere facilmente intercettati e persino soggetti a vari attacchi. Pertanto, garantire la segretezza e la sicurezza della comunicazione è diventato un argomento di ricerca molto discusso nella comunicazione acustica sottomarina. In generale, la comunicazione acustica sottomarina coperta è classificata in tre categorie: bassa probabilità di rilevamento (LPD), bassa probabilità di riconoscimento (LPR) e bassa probabilità di intercettazione (LPI). Nella comunicazione acustica sottomarina

coperta tradizionale (UAC), un metodo per realizzare la comunicazione coperta è la tecnologia di diffusione dello spettro. L'approccio ciclico (CA), l'algoritmo ciclico ponderato nuovo (WeCAN) e l'algoritmo ciclico di correlazione periodica nuovo (PeCAN) sono utilizzati per sintetizzare le forme d'onda di diffusione. Con queste forme d'onda di diffusione, la tecnologia di diffusione diretta (DSSS) è utilizzata per ridurre la densità spettrale di potenza dei segnali di comunicazione, rendendo così i segnali di comunicazione più nascosti. Per utilizzare WeCAN, è essenziale conoscere in anticipo la risposta all'impulso del canale, ma questo è impossibile. Per risolvere questo problema, una sequenza disponibile è proposta in una varietà di ambienti, dove valori compresi tra 0 e 1 sostituiscono il valore binario 1 o 0 nelle metriche di ponderazione precedenti. Sono utilizzate anche sequenze pseudo-casuali e sequenze caotiche come codici di diffusione per ridurre il rapporto segnale-rumore (SNR) dei segnali di comunicazione. Questi due metodi possono migliorare la segretezza riducendo l'SNR dei segnali di comunicazione. Tuttavia, i segnali di comunicazione possono ancora essere rilevati tramite rilevamento dell'inviluppo o rilevamento dell'energia, e la riduzione dell'SNR aumenterà il tasso di errore dei bit (BER) e limiterà la distanza di comunicazione. Nella UAC tradizionale coperta, un altro metodo per rendere i segnali di comunicazione più nascosti è la tecnologia di modulazione multi-portante. L'approccio multibanda con equalizzazione congiunta e de-spreading è utilizzato per migliorare l'affidabilità della comunicazione a bassi SNR. In [59] viene proposto un UAC (underwater acoustic communication) bionico. Poiché i richiami dei mammiferi marini sono generalmente classificati come rumore oceanico e filtrati in UAC, possono essere utilizzati per realizzare una trasmissione nascosta di informazioni nell'ambiente sottomarino. Poiché le caratteristiche tempo-frequenza dei fischi sono piuttosto buone, modulate le informazioni utilizzando la struttura tempo-frequenza (TFS), viene proposto un metodo UAC bionico basato sul fischio dei delfini con ritardo di tempo (BCUAC-TD). Nel metodo proposto, la TFS è divisa in diversi segmenti uguali per rappresentare segnali simbolici differenti. Il ritardo di tempo viene utilizzato per modulare le informazioni su ciascun segmento. I segnali di comunicazione costruiti con il metodo UAC bionico proposto sono simili ai fischi originali e possiedono una buona copertura. Le prestazioni del schema BCUAC-TD proposto sono verificate sotto il canale modulato e il canale sperimentale.

3.1.7 Luce e corrente elettrica

Recentemente, i "prodotti intelligenti", come le lampadine controllate via wireless, sono apparsi sul mercato promettendo pianificazioni automatiche e una gestione degli edifici più semplice. Desiderosi di migliorare l'efficienza delle loro operazioni, le aziende e i gestori di edifici hanno iniziato ad adottare queste tecnologie IoT. Tuttavia, queste lampadine intelligenti potrebbero essere abusate ed utilizzate per inviare informazioni in modo covert su lunghe distanze tramite modulazioni impercettibili della luminosità della luce. Purtroppo, man mano che i dispositivi IoT diventano più diffusi, la loro sicurezza rimane minima e gli attaccanti continuano a sfruttarli in numero sempre maggiore. In questo articolo [32], vengono mostrate tre strategie di Canale Covert per lampadine intelligenti e analizzate la loro throughput, stealthiness e generalità. Un altro vettore di attacco incorpora sensori di corrente collegati al circuito elettrico su cui risiede la lampadina intelligente. Assumiamo che il circuito che controlla la lampadina sia collegato a un pannello elettrico esterno a cui un attaccante ha accesso. L'attaccante può quindi osservare le modulazioni della luminosità della luce tramite i sensori di corrente sul circuito. Poiché la maggior parte delle lampadine intelligenti è progettata per consumare poca energia (e quindi utilizzare una corrente minima), la lampadina deve subire cambiamenti di luminosità maggiori per creare fluttuazioni di corrente visibili. Pertanto, questo canale assume che la lampadina intelligente risieda in una stanza non occupata, come una sala server o un ripostiglio. Il valore di corrente misurato quando la lampadina è accesa e spenta è assegnato rispettivamente a un livello logico alto e basso. Viene utilizzata la codifica manchester per aumentare la stealthiness e mitigare i problemi di temporizzazione. Queste tecniche concernono la modulazione della luminosità della luce, del colore della luce e del consumo energetico della luce:

- *Modulazione della Luminosità della Luce*: questo attacco varia la luminosità della luce bianca in una misura impercettibile all'occhio umano ma rilevabile da un sensore di colore. I valori leggermente più brillanti e più dim sono usati per rappresentare livelli logici binari. Un attaccante può utilizzare un dispositivo compromesso per connettersi alla lampadina intelligente e variare rapidamente la luminosità tra questi valori per trasmettere un messaggio in binario. Un sensore di colore puntato sulla lampadina può discernere la differenza tra i valori, permettendogli di decodificare il messaggio.

- *Modulazione del Colore della Luce*: un altro metodo per esfiltrare dati consiste nel cambiare i valori RGB dei LED colorati all'interno di una lampadina intelligente. È possibile variare la tonalità tra diversi valori in modo tale che le variazioni siano impercettibili. A differenza dell'attacco di modulazione della luminosità, una lampadina RGB ha la capacità di variare fino a tre canali di colore per implementare l'attacco. Nella nostra implementazione, utilizziamo sia i canali rosso che blu della nostra lampadina intelligente, mantenendo il verde alla massima luminosità. Questo aiuta non solo a mascherare l'attacco ma anche a ridurre le interferenze, poiché i rilevatori rosso e blu in un fotosensore a basso costo si sovrappongono con il canale verde. Infatti, utilizzando i canali rosso e blu del nostro sensore, siamo in grado di decodificare quattro diversi livelli per colore. Quindi, ogni canale può inviare due bit per periodo di clock, per un totale di quattro bit contemporaneamente.

- *Canale Covert Basato sul Consumo di Energia*: un altro vettore di attacco incorpora sensori di corrente collegati al circuito elettrico su cui risiede la lampadina intelligente. Assumiamo che il circuito che controlla la lampadina sia collegato a un pannello elettrico esterno a cui un attaccante ha accesso. L'attaccante può quindi osservare le modulazioni della luminosità della luce tramite i sensori di corrente sul circuito. Poiché la maggior parte delle lampadine intelligenti è progettata per consumare poca energia (e quindi utilizzare una corrente minima), la lampadina deve subire cambiamenti di luminosità maggiori per creare fluttuazioni di corrente visibili. Pertanto, questo canale assume che la lampadina intelligente risieda in una stanza non occupata, come una sala server o un ripostiglio. Il valore di corrente misurato quando la lampadina è accesa e spenta è assegnato rispettivamente a un livello logico alto e basso.

3.2 Contromisure

In rilevamento troviamo tutti quegli articoli che trattano di modalità di rilevamento di Timing Channel all'interno di una determinata rete.

3.2.1 Delay randomico

I canali temporizzati sono piuttosto difficili da rilevare poiché i tempi di invio dei pacchetti sono una variabile casuale dipendente dal carico della rete. Inoltre, a differenza dei canali di memorizzazione, la possibilità di costruire canali temporizzati rimane anche con i metodi tipici di sicurezza della rete, ad esempio il tunneling e la crittografia del traffico. L'eliminazione dei canali temporizzati potrebbe inoltre portare a una riduzione inaccettabile dell'efficienza del canale di comunicazione (inviando pacchetti con ritardi inter-pacchetto uguali). Si considera quindi un modo per limitare la capacità del Canale Temporizzato basato sulla modulazione degli intervalli inter-pacchetto introducendo ritardi casuali prima di inviare i pacchetti [2]. La capacità residua del Canale Temporizzato binario viene valutata con l'introduzione di un metodo di "controazione", e vengono forniti suggerimenti per scegliere un valore del parametro che garantisca sicurezza mantenendo però l'efficienza della rete.

Covert Channel e Contromisura

Il mittente e il destinatario scelgono gli intervalli $t_0 = 2d_{\max}$ e $t_1 = 6d_{\max}$ che forniscono un'alta velocità di trasmissione dati e un adeguato livello di accuratezza nel canale nascosto con il carico attuale della rete. L'intervallo medio tra i pacchetti è $t = 4d_{\max}$.

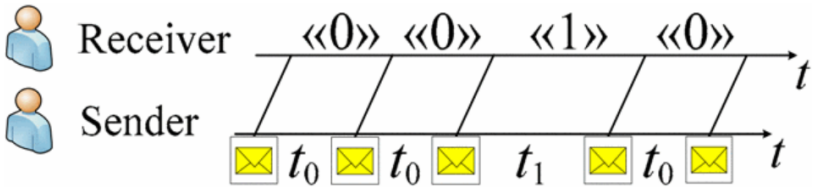

Figura 3.2: Funzionamento Covert Channel [2]

La contromisura proposta consiste nell'introduzione di ritardi casuali aggiuntivi uniformemente distribuiti nell'intervallo (0; t) dove (t < t). L'invio di ogni pacchetto viene ritardato dal tempo d selezionato casualmente dall'intervallo (0; t). Valori di ritardo distribuiti uniformemente non permetterebbero a un insider di fare supposizioni sul valore più probabile del ritardo. Vengono usati i bps come unità di misura.

Se la lunghezza media del pacchetto IP è L_{av} bit, allora la capacità residua del canale di comunicazione nelle condizioni del metodo di contromisura proposto è:

$$_{\text{res}} = \frac{L_{\text{av}}}{L_{\text{av}} + \frac{t}{2}} = 2\frac{L_{\text{av}}}{2L_{\text{av}} + t}$$

Capacità del Canale Nascosto ad Alto Carico di Rete

La capacità del canale nascosto con rumore può essere determinata come:

$$C = \max_{X} \frac{I(X,Y)}{E(T)}$$

dove X,Y sono due variabili casuali che descrivono rispettivamente le caratteristiche del canale di ingresso e di uscita. L'informazione mutua è calcolata con la formula:

$$I(X,Y) = H(Y) - H(Y|X)$$

dove l'entropia della variabile casuale Y è:

$$H(Y) = -\sum_{y \in 0,1} p_{\text{out}}(y) \log_2 p_{\text{out}}(y)$$

e l'entropia condizionale della variabile casuale Y e la variabile casuale X è:

$$H(Y|X) = -\sum_{x \in 0,1} p_{\text{in}}(x) \sum_{y \in 0,1} p_{\text{out}}(y|x) \log_2 p_{\text{out}}(y|x)$$

I simboli trasmessi attraverso il canale nascosto proposto sono equamente probabili, cioè $p_{\text{in}}(0) = p_{\text{in}}(1) = 1/2$. Il mittente invia un pacchetto con un intervallo tra i pacchetti t_0 se vuole trasmettere il bit "0" e con l'intervallo t_1 se vuole trasmettere il bit "1". Si suppone che il canale nascosto operi nel contesto di un alto carico sulla rete, ad esempio durante le ore di punta. Questo significa che il tempo di trasmissione di un pacchetto nella rete non è costante e si assume che gli intervalli tra i pacchetti sul lato del ricevitore siano, rispettivamente per "0" e "1", nell'intervallo $(0;4\text{d}_{\text{max}}) e (4d_{\text{max}}; 8d_{\text{max}})$.

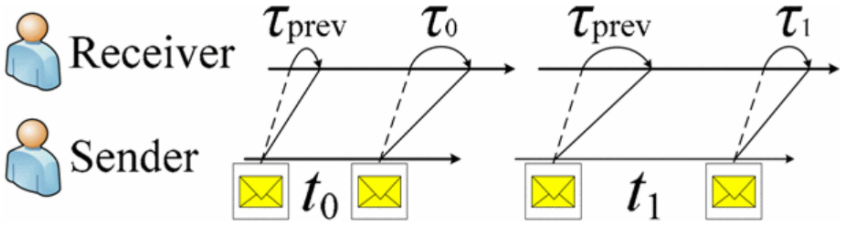

Figura 3.3: Introduzione di delay randomici [2]

Raccomandazioni Pratiche

La limitazione dei canali nascosti tramite ritardi casuali causa un incremento del numero di errori e gli intervalli di tempo t_0 e t_1 possono essere aumentati per mantenere un alto livello di accuratezza. La capacità di un canale nascosto senza errori è pari a $2/(t_0 + t_1)$ bps, tuttavia, non è sempre superiore alla capacità di un canale nascosto con errori. Per costruire canali nascosti senza errori, i valori degli intervalli di tempo t_0 e t_1 saranno selezionati come segue:

$$t_0 = 2d_{\max}, t_1 = t_0 + 2t + 4d_{\max}$$

La Figura 3.4 e la Figura 3.5 mostrano i diagrammi delle capacità dei canali nascosti con errori e senza errori nei casi di alto carico di rete rispettivamente per $t < 2d_{\max}$ e $2d_{\max} < t < 4d_{\max}$. Le linee blu indicano la capacità del canale nascosto con errori, le linee rosse si riferiscono alla capacità del canale nascosto senza errori. Come si può vedere in Figura 3.4, quando $t < 2d_{\max}$, è svantaggioso per un trasgressore costruire un canale nascosto senza errori. Come si può vedere in Figura 3.5, quando $2d_{\max} < t < 4d_{\max}$, con piccoli valori di t, la capacità del canale nascosto con errori è maggiore, con grandi valori di t, la capacità dei canali nascosti senza errori è maggiore.$_a$ è la capacità consentita di un canale nascosto tale che il funzionamento di un canale nascosto con una capacità inferiore non sia considerato pericoloso. Nel caso in cui sia necessario limitare la capacità di un canale nascosto senza errori al valore $_a$, il parametro di contromisura dovrebbe essere:

$$t = \frac{1 - 4d_{\max a}}{a}$$

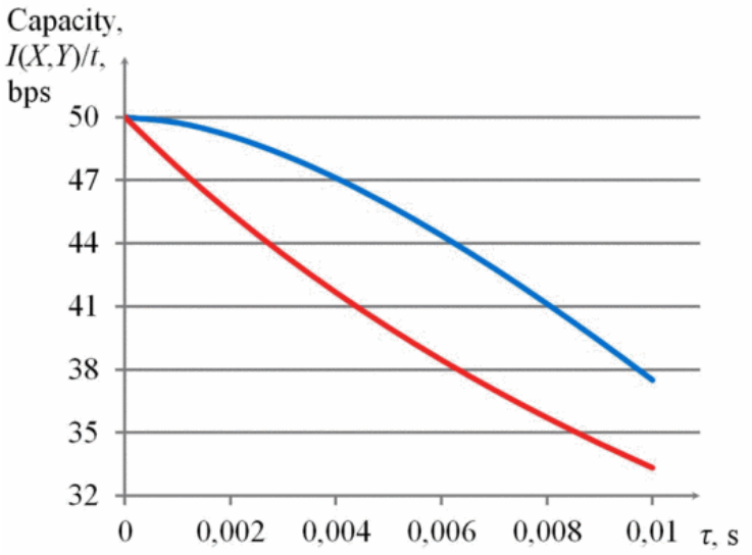

Figura 3.4: Il rapporto tra I(X,Y)/t e t con dmax=0.005sec. e t<2dmax [57]

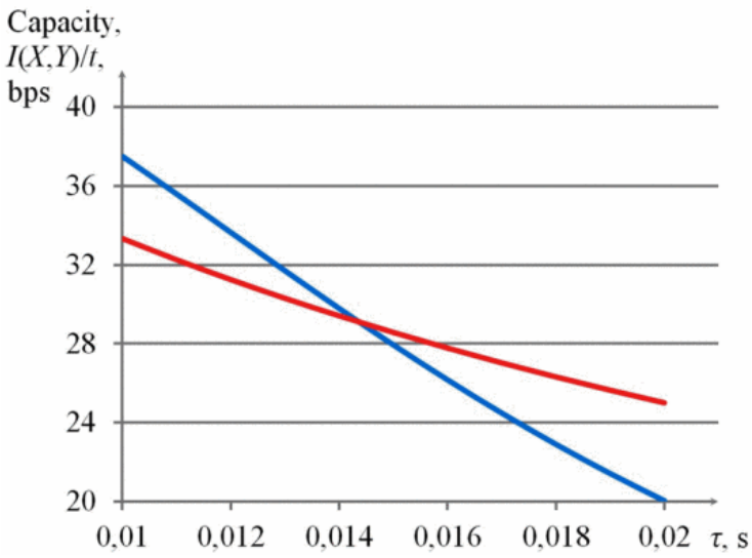

Figura 3.5: Il rapporto tra I(X,Y)/t e t con dmax=0.005sec. e 2dmax<<4dmax [57]

I canali con una capacità inferiore a 1 bps sono considerati accettabili nella maggior parte dei sistemi, ma i canali nascosti con una capacità superiore a 100 bps dovrebbero essere eliminati. Pertanto, al fine di ridurre la sua capacità a 100 bps nel contesto di un elevato carico sulla rete, il parametro di contromisura t dovrebbe essere pari a 5

mps.

3.2.2 Cache server come jammer dinamico

I canali nascosti rappresentano una grave minaccia per i sistemi di sicurezza a più livelli. Se un sistema di sicurezza a più livelli presenta canali nascosti, un utente di alto livello può far trapelare informazioni riservate a un utente di basso livello, rendendo il sistema insicuro. È difficile eliminare i canali temporali nascosti e gli studi attuali si concentrano su come individuarli o limitarli. Negli anni è stato analizzato l'uso di un singolo server per memorizzare nella cache i pacchetti di rete per limitare i Timing Channel: il server invia i pacchetti in base a una strategia di jamming. È evidente che il metodo di jamming può diminuire la capacità del Canale Covert Temporizzato, se vogliamo aumentare la capacità del canale overt, dobbiamo aumentare il tasso di servizio del server, ma questo comportamento diminuirà il rumore e quindi aumenterà la capacità del Canale Covert Temporizzato. Studiando i lavori di ricerca esistenti, troviamo che il tempo di servizio dei pacchetti è indipendente dal tasso di arrivo delle reti e dalla quantità di pacchetti memorizzati nella cache del server. In alcune situazioni, questa strategia può far perdere tempo ai pacchetti in attesa di essere inviati, e il tempo potrebbe non aiutare a diminuire la capacità del Canale Covert Temporizzato. Viene presentata dunque una strategia [57] di jamming auto-adattativa che può limitare la capacità del Canale Covert Temporizzato ma non quella del del canale overt, rendendo il sistema più efficiente rispetto agli studi precedenti. Questa strategia può modificare i timing di invio dei pacchetti rispetto alla capacità del canale overt e alla lunghezza della coda, rendendola molto flessibile anche per reti complesse.

3.2.3 Chaos Theory e Threshold Secret Sharing

Con il rapido sviluppo della tecnologia dell'informazione, la sicurezza della rete ha attirato sempre più attenzione. I Canali Temporizzati Nascosti di Rete (NCTC) sono una tipologia di Covert Channel che trasmette dati modificando i ritardi inter-pacchetti (IPD) del traffico di rete legittimo, e può eludere i meccanismi di sicurezza della rete convenzionali come i firewall. Stabilendo semplicemente un collegamento di comunicazione apparentemente innocuo tra loro, gli hacker possono far trapelare

informazioni sensibili e riservate da una rete con alti privilegi di sicurezza a una rete esterna. Pertanto, è di grande importanza studiare la tecnologia di rilevamento del Canale Temporizzato nascosto di rete per risolvere il problema della sicurezza della rete. Gli IPD (ritardi inter pacchetto) del traffico legittimo variano in base alle applicazioni di rete e ai collegamenti di comunicazione. Pertanto, è difficile rilevare gli IPD che sono stati intenzionalmente modificati per la comunicazione covert. Attualmente, i metodi di rilevamento possono essere classificati in quattro categorie: test di forma, test di regolarità, test di entropia e test statistici non parametrici. Tuttavia, alcuni di questi metodi di rilevamento sono progettati per rilevare un Canale Temporizzato Covert specifico e, quindi, non riescono a rilevare altri tipi di Canali Temporizzati Covert; alcuni metodi sono troppo sensibili ai cambiamenti del traffico di rete. Concentrandosi sulla bassa robustezza e versatilità di vari metodi di rilevamento nella ricerca attuale, l'articolo in [21] propone un nuovo metodo di rilevamento dei NCTC basato sulla chaos theory e sulla Threshold Secret Sharing. In primo luogo, utilizziamo la tecnica di ricostruzione dello spazio delle fasi della chaos theory per estendere il sistema di traffico di rete dallo spazio a bassa dimensione a quello ad alta dimensione, estraendo così alcune caratteristiche del traffico che sono abbastanza robuste. Infine, utilizziamo la strategia di ricostruzione delle informazioni segrete dalla Threshold Secret Sharing per costruire identificatori di canale unici e tenaci come base di discriminazione per distinguere i NCTC (Covert Timing Channel) dai canali legali.

3.2.4 Entropia gerarchica

Il rilevamento dei canali nascosti è una questione chiave nell'infrastruttura moderna della tecnologia dell'informazione (IT). Molte aziende, paesi e agenzie governative come il governo degli Stati Uniti e gli organi militari degli Stati Uniti, l'Agenzia per la Sicurezza Nazionale, l'Aeronautica Militare degli Stati Uniti e il Centro Nazionale per la Sicurezza Informatica sono focalizzati sull'elaborazione di tecniche migliori per rilevare l'esistenza di canali nascosti. Questo servirà come un fondamentale passo in avanti per la costruzione di un sistema di supporto decisionale che protegge l'infrastruttura IT contro tali vulnerabilità. Nelle recenti ricerche inerenti al rilevamento di Timing Channel, una varietà di tecniche statistiche e di riconoscimento dei pattern sono state impiegate: una delle tecniche più popolari è la misura dell'analisi dell'entropia. Il valore di entropia (o entropia piatta) fornisce una misura della quan-

tità di bit di dati non casuali (cioè, pattern) scambiati su uno specifico flusso di dati. Maggiore è il valore dell'entropia, minore è la probabilità che bit di dati non casuali vengano scambiati sul canale. In [29], l'obiettivo è quello di dimostrare che l'entropia gerarchica, rispetto a quella piatta, consente una rilevazione più efficiente dei Canali Temporizzati Covert. Complessivamente, questo approccio divide il flusso di tempi di arrivo in due sotto-gruppi a livelli diversi ed estrae in modo il valore di entropia più basso, che a sua volta fornisce il livello più alto di evidenza sull'esistenza del Canale Temporizzato Covert. La principale motivazione per l'adozione dell'approccio di analisi dell'entropia gerarchica è rilevare l'esistenza del Canale Temporizzato Covert indipendentemente dalla scala temporale entro cui è nascosto rispetto al flusso di dati complessivo. D'altra parte, l'entropia piatta potrebbe non fornire un indicatore accurato come l'entropia gerarchica, poiché viene applicata all'intero flusso di dati e non esplora i livelli di entropia delle diverse scale temporali del flusso di dati.

Capitolo 4

Implementazione e Rilevamento

4.1 Implementazione

Per rilevamento si vanno ad intendere sia i programmi che vanno ad implementare una tipologia particolare di Covert Timing Channel sia i tool che si occupano di più Covert Channel, utilizzando sia quelli storage che quelli timing e sfruttando vari protocolli. Nella Tabella alla fine della sezione sono presenti gli strumenti di cui tratteremo nelle sotto-sezioni successive.

4.1.1 Timeshifter

Timing channel[1] scritto in C che, sfruttando il protocollo IP, va a modificare gli intervalli di invio tra i pacchetti ed assegna ad ogni "delay" un determinato bit (0 od 1), il messaggio viene dunque mandato in binario usando questi valori e può essere decifrato dal computer che conosce questi valori. In input devono essere dati la coda di netfilter, i millisecondi di soglia ed i millisecondi di delay all'avvio del programma, viene poi richiesto l'inserimento del messaggio da inviare, che verrà poi dato in output sia sul trasmettitore che sul ricevitore. Si occupa di intercettare e gestire i pacchetti di rete, utilizza la coda "netfilter" di Linux e opera in due modalità: ricevitore e trasmettitore. In modalità ricevitore inizialmente il programma, arrivato il primo pacchetto, registra l'ora di arrivo e stampa un messaggio di avvio. Per i pacchetti successivi inizia a calcolare il tempo trascorso dal precedente e lo confronta con un

[1] Anfractuosity, timeshifter, 2015, https://github.com/anfractuosity/timeshifter.

valore di soglia, se il tempo è inferiore a questa soglia verrà trattato come 0 altrimenti come 1. Questi bit vengono aggiunti ad un buffer e tradotti utilizzando il carattere ASCII corrispondente agli 8 bit ottenuti. In modalità trasmettitore la prima volta che viene intercettato un pacchetto viene stampato un messaggio di avvio e va a leggere un carattere di input (ottenuto da file o tastiera), salvandolo in un buffer. A questo punto verranno estratti dal buffer mano a mano tutti i bit contenuti rappresentativi di quel carattere, se il bit è 1 il programma va in "sleep", attendendo un periodo specifico prima di permettere al pacchetto di essere inviato, se il bit è 0 il pacchetto viene inviato immediatamente e una volta trasmessi gli 8 bit il processo ricomincia.
Client Output:

- Receiver...
- BYTE: 01001000 (H)
- BYTE: 01100101 (e)
- BYTE: 01101100 (l)
- BYTE: 01101100 (l)
- BYTE: 01101111 (o)

Server Output:

- Transmitter...
- BYTE: 01001000 (H)
- BYTE: 01100101 (e)
- BYTE: 01101100 (l)
- BYTE: 01101100 (l)
- BYTE: 01101111 (o)

4.1.2 TimingCovertChannel

Programmato in Python e sfrutta il protocollo TCP, per funzionamento similare a quello di timeshifter ma con qualche differenza sostanziale[2]. Il ricevitore (o client) si connette al server di trasmissione tramite socket ed inizia a ricevere i dati dal sender, misura dunque i ritardi di ricezione dei pacchetti per determinare se corrispondono a 0 od 1 usando il valore "ONE" di threshold. Continua questo processo per ogni carattere che viene ricevuto fino al segnale di "EOF" che sancisce la fine del messaggio. A questo punto converte la sequenza binaria ricevuta dapprima nel numero decimale corrispondente e poi in caratteri ASCII che verranno poi stampati a schermo mostrando il messaggio decifrato. Il trasmettitore (o server) ha salvato nella stringa "covert" il messaggio da nascondere, ogni carattere viene dunque convertito in una stringa binaria di 8 bit: "covertbin". Dunque apre un socket e si mette in ascolto per eventuali connessioni e, una volta avvenute, inizia a inviare un carattere alla volta del messaggio "msg" precedentemente stabilito. Il server introduce un ritardo rispetto ad ogni carattere in base al bit contenuto in "covertbin", se è 0 è breve, se è 1 è lungo e, una volta inviati tutti i caratteri del messaggio manda un segnale di "EOF" per segnalare la fine della trasmissione.
Client Output:

- Some message...Some message...Some message...
  Some message...Some message...Some message...Some message...
- Covert message: Messaggio Nascosto EOF

Server Output:

- Covert message: Messaggio Nascosto EOF
- Covert in binary: 01001101011001010111001101110011011000
  01011001110110011101101001011011110010000001001110011000
  00010001101100011011011110111001101110100011011110010000
  00001000101010011110100011 0
- Sending characters with delays:

[2] Devonknudsen, timing-covert-channel, 2019, https://github.com/devonknudsen/Timing-Covert-Channel.

4.1.3 csc-hw1-covert-channel

Programmato in Python e sfrutta il protocollo UDP, funzionamento similare al programma sopra con qualche differenza marginale[3]. In modalità ricevitore (o client) viene creato un socket UDP di ascolto su una specifica porta. Cominciano ad arrivare i pacchetti ed il client, per ogni pacchetto ricevuto, calcola il tempo trascorso tra inizio e fine ricezione e, a questo punto, valuta questo tempo rispetto a "commondelta/2", dove "commonsdelta" è il tempo di delay. Se "receivingtime" è superiore a "commondelta/2" allora il bit corrente è interpretato come 1 altrimenti come 0. Viene così ricostruito il messaggio nascosto bit per bit e poi stampato all'utente inseme al messaggio visibile "nicemessage". In modalità trasmettitore (o server) l'invio dei dati è gestito tramite thread "sendmessage()", che consente al programma di eseguire più invii contemporanei a diversi client. Si itera dunque ogni bit contenuto in "hiddenmessage" e, se il bit è 1 si attende per un periodo di tempo "commons.delta" prima di inviare il carattere corrispondente di "nicemessage" al receiver tramite UDP, se invece è 0 il sender invia immediatamente. Infine una volta inviati i dati il programma calcola anche la velocità di trasmissione dei bit (bit/secondo) che è basata sul rapporto tra la lunghezza di "hiddenmessage" e il tempo "sendingtime" di invio. Il socket viene infine chiuso una volta terminata la comunicazione.
Client Output:

- receiver is on...
- niceMessage: -I love you!
- hiddenMessage: -I hack you!

Server Output:

- speed bit/second: 45.33340248652328

4.1.4 PcapStego

Un tool[4] scritto in Python da Caviglione e Zuppelli che permette anche di implementare Covert Channel di tipo timing, sfruttando i protocolli IPv4, IPv6, ICMPv4

[3] Abbasyazdanmehr, csc-hw1-covert-channel, 2024, https://github.com/ay-sbu/csc-hw1-covert-channel.
[4] Ocram95, $\text{pcap}_i njector, 2024, https://github.com/Ocram95/\text{pcap}_i njector$.

e ICMPv6, all'interno di file .pcap che possono essere poi usati per simulazioni all'interno di una rete. Utilizza 2 modalità principali, una interattiva che permette di scegliere il tipo di Covert Channel, il segreto da usare ed il tipo di meccanismo di iniezione, l'altra invece è la bulk che permette di automatizzare il processo e di utilizzare più meccanismi insieme. Ha come dipendenze le librerie Scapy e Pandas, tshark e nei suoi file contiene 2 database che contengono malware e payloads che possono simulare una minaccia di Covert Channel. Correntemente il tool supporta i seguenti protocolli e Covert Channel:

- *IPv4*: Type of Service (8 bit/pkt), Time To Live (1 bit/pkt), Identification Number (16 bit/pkt), Timing (1 bit/pkt)
- *IPv6*: Flow Label (20 bit/pkt), Traffic Class (8 bit/pkt), Hop Limit (1 bit/pkt), Timing (1 bit/pkt)
- *ICMPv4*: Payload (48 bit/pkt), Timing (1 bit/pkt)
- *ICMPv6*: Payload (8 bit/pkt), Timing (8 bit/pkt)

In particolare per il protocollo IPv6 vengono usate queste funzioni:

- *readattack()*: serve a fare il parsing delle informazioni contenute nel file secrets.txt, specifica quali informazioni saranno mandate tramite il canale coperto.
- *findflows()*: fa il parsing delle informazioni dal file .pcap sorgente e prepara una "mappa" di tutte le comunicazioni IPv6 disponibili. Cerca dunque quali conversazioni hanno la capacità di contenere il segreto da comunicare.
- *flowselection()*: permette di identificare in maniera univoca il flusso di traffico scelto dall'utente, utilizza l'indirizzo sorgente IPv6, quello di destinazione, la porta TCP/UDP sorgente, quella di destinazione ed il protocollo utilizzato.
- *inject()*: svolge l'iniezione nel traffico desiderato, in particolare si occupa di trovare i pacchetti appartenenti a quel flusso e modificarli in base al metodo di iniezione richiesto, lasciando inalterate le caratteristiche rimanenti. E' una funzione modulare e può essere usata per implementare nuove tipologie di Covert Channel.

- *writetocsv()*: si occupa di fare logging delle informazioni del flusso di dati che contiene la Covert Channel tra cui l'identificatore della conversazione, il meccanismo di iniezione e la lunghezza dei dati nascosti. Tutto questo viene salvato in un file CSV (chiamato flows.csv) che può essere poi utilizzato per automatizzare il processo.

Queste funzioni vengono utilizzate anche per altri protocolli opportunamente modificate per essere compatibili ma mantengono comunque generalmente le caratteristiche sopra menzionate. Nel caso in cui si voglia implementare un canale di tipo timing, questo deve essere scelto come metodo di iniezione del segreto nel momento in cui viene avviato il programma. A questo punto il funzionamento del timing channel sarà similare a quanto descritto nei programmi precedenti: verrà applicato un delay al pacchetto da inviare nel caso in cui il bit sia 1 e nel caso in cui sia 0 verrà inviato immediatamente. Garantendo che il ricevente possa decodificare questo delay allora sarà possibile comunicare in segreto e nascondere/tradurre messaggi.
Nella Tabella 4.1 sono mostrati tutti gli strumenti di cui abbiamo discusso in questa sezione.

Nome	**Licenza**	**Linguaggio**	**Descrizione**	**Rilascio**	**Ultimo Update**
Timeshifter	/	C/Shell	Timing channel in C	2015	2017
TimingCovert Channel	/	Python	Timing channel in Python	2019	2019
pcapStego	MIT	Python	Diversi tipi di Covert Channel	2021	2024
csc-hw1-covert-channel	/	Python	Timing channel in Python	2024	2024

Tabella 4.1: Strumenti utili per l'implementazione di Covert Timing Channel

4.2 Rilevamento

Per rilevamento si vanno ad intendere tutti quegli strumenti reperibili online che permettono di poter andare a rilevare Covert Channel presenti all'interno di una determinata rete. Alcuni di questi sono utilizzati principalmente come benchmark per poter testare le proprie Covert Channel da un punto di vista prettamente scientifico mentre altri si propongono come soluzioni di cybersecurity da Covert Channel. Nella Tabella a fine sezione sono presenti gli strumenti di cui tratteremo nelle sotto-sezioni successive.

4.2.1 NeFiAS

Uno strumento[5] per anomaly detection e network forensics progettato per l'uso su Network Covert Channel. Progetto nato da Steffen Wendzel che vuole essere un banco di prova per la comunità scientifica il più semplice ed accessibile. Per questa ragione è scritto in bsh e funziona su linux standard. Per il funzionamento richiede due tipi di nodi, uno master ed almeno uno slave. Il nodo master si occupa di leggere il traffico, estrarre metadati, dividere i dati e caricarli nei nodi slave, che invece hanno il compito di svolgere le operazioni. Le prestazioni dipendono principalmente dal numero di slave utilizzati. Può essere utilizzato per verificare se la tecnica di Covert Channel implementata può essere rilevata da tecniche di anomaly detection. Basato interamente su SSH ed SCP e sfrutta le funzionalità di compressione e criptazione di OpenSSH. Per il suo funzionamento è necessario impostare il NeFiAS master andando ad installare i seguenti strumenti su interfaccia CLI di linux: -bc -dialog -bash -ssh/scp -awk -sed -tr -gzip -tshark. A questo punto è necessario impostare i nodi slave di NeFiAS andando a creare un utente Linux per ciascun nodo, dando accesso ad ognuno di questi alla chiave pubblica SSH del nodo master e segnandoli come "authorized", autorizzati alla comunicazione. Ogni nodo slave andrà infine aggiunto al file di configurazione slaves.cfg del nodo master. NeFiAS è quindi stato impostato correttamente e può eseguire "jobs", ovvero funzionalità diverse in base al tipo di script implementato sui nodi slave, cosa che permette allo strumento di potersi adattare a situazione e necessità differenti. Quello che interessa a noi in particolare è lo script "eSimIATframetimerelative.sh" che, sfruttando i punteggi di "epsilon similarity" dei

[5] Cdpxe, nefias, 2017, https://github.com/cdpxe/nefias.

"gap" tra pacchetti, permette di rilevare Covert Channel di tipo timing. Lo script utilizza i tempi di inter-arrivo dei pacchetti, calcola le differenze relative tra i tempi, conta il numero dei lambda minori di epsilon e fa una proporzione tra il contatore ed il numero di tempi – 1. Questo viene fatto per analizzare la somiglianza dei tempi di invio tra i pacchetti all'interno di una finestra statica di 2001 pacchetti, eventuali deviazioni rispetto ad un modello o distribuzione attesa possono identificare la presenza di un Covert Channel. Diamo quindi in input al programma una parte del traffico analizzato e ci verranno restituiti i valori dei punteggi calcolati.

4.2.2 Wireshark/Tshark

Software per analisi di protocollo utilizzato per la soluzione di problemi di rete, l'analisi e lo sviluppo di protocolli o di software per la comunicazione e didattica. Il progetto nasce da Gerald Combs nel 1998. Oggi Wireshark[6] (Tshark[7] è la versione Cli) è lo standard de facto (e spesso de jure) in molte aziende per vedere cosa accade ad una rete a livello microscopico. Le sue principali funzionalità sono:

- Ispezione approfondita di centinaia di protocolli (in continuo aumento).
- Acquisizione del traffico online e successiva analisi offline.
- Browser dei pacchetti a tre pannelli.
- Multipiattaforma: Windows, Linux, macOS, Solaris, FreeBSD, NetBSD, etc.
- Ricca analisi VoIP.
- Lettura/Scrittura di diversi formati di file di acquisizione: tcpdump (libpcap),Pcap NG, Catapult DCT2000, etc.
- Supporto della decodifica per molti protocolli, tra cui IPsec, ISAKMP, Kerberos, etc.

Può essere utilizzato per la rilevazione dei timing channel in sinergia con altri strumenti, come ad esempio NeFiAS. La funzionalità di poter salvare porzioni di flusso

[6]The Wireshark team, TShark, Dump and analyze network traffic, 2008,https://www.wireshark.org/docs/man-pages/tshark.html.

[7]The Wireshark team, Wireshark, 1998,https://www.wireshark.org.

permette infatti di poter svolgere calcoli statistici su quell'insieme di dati e comprendere se vi è una deviazione anomala tra i tempi di inter-invio dei pacchetti (ad esempio), anomalia che aiuta a scovare eventuali timing channel.

4.2.3 Rita

Un framework[8] open source nato da John Strand per l'analisi del traffico dei network. L'idea è quella di facilitare l'hunt teaming, cioè la ricerca attiva delle minacce all'interno di una rete da parte di una squadra di sicurezza. Il framework acquisisce Zeek Logs in formato TSV e attualmente supporta le seguenti funzionalità principali:

- *Beaconing Detection*: cerca segnali di beaconing behavior all'interno e all'esterno della rete.
- *DNS Tunneling Detection*: cerca segnali di Covert Channel basati su DNS.
- *Blacklist Checking*: sfrutta blacklist per cercare domini ed host sospetti.

Rispetto alla scoperta dei timing channel utilizza i log di Zeek per l'analisi dei tempi di invio tra pacchetti per scoprire se si discostano da un determinato valore di soglia standard e, tramite ulteriori calcoli statistici, capire se effettivamente vi è in corso una comunicazione nascosta di tipo timing. Oltre a questo analizza anche la periodicità e prevedibilità dei timing, la consistenza del flusso di rete e l'entropia dei tempi di inter-arrivo dei pacchetti. Questi dati vengono analizzati perché normalmente un timing channel introduce periodicità, prevedibilità nelle comunicazioni di rete e inconsistenze e cambiamenti repentini nei timing. Una volta eseguito il programma darà in output un punteggio che permetterà all'utente di poter distinguere se quella che ha di fronte è una minaccia e, nel caso ci sia, quanto è grave. Oltre a questo genera anche degli allarmi inerenti alle minacce e dei report dettagliati che includono gli IP sorgente e destinazione, la natura delle irregolarità di timing e del flusso del traffico.

4.2.4 Zeek

Analizzatore[9] di traffico passivo ed open-source, viene usato per investigare su traffico anomalo e su attività malevole. Il progetto nasce nel 1995 da Vern Paxson, ricercatore

[8] Activecm, rita, 2016, https://github.com/activecm/rita.
[9] The Zeek team, zeek, 1995, https://github.com/zeek/zeek.

del “Lawrence Berkeley National Laboratory” ed è originariamente chiamato “Bro”. I log che raccoglie sul traffico di rete sono molto dettagliati ed espansivi, questi contengono informazioni su:

- Richieste HTTP.
- Tipi MIME.
- Risposte Server.
- Richieste DNS.
- Certificati SSL.
- Sessioni SMTP.

Ha anche altre funzioni di analisi del traffico per rilevare anomalie ed è largamente espandibile per poter effettuare funzionalità differenti. Può essere utilizzato nell'implementazione di strumentazioni per la rilevazione di timing channel in sinergia con altri programmi, come ad esempio RITA, che ne fa uso per il suo funzionamento.
Nella Tabella 4.2 sono presenti tutti gli strumenti di cui abbiamo discusso in questa sezione.

Nome	**Licenza**	**Linguaggio**	**Descrizione**	**Rilascio**	**Ultimo Update**
NeFiAS	GPL-3.0	BASH	Banco di prova per la rilevazione	2017	2022
Rita	GPL-3.0	Go/BASH	Framework per analisi del traffico	2016	2024
Wireshark	GPL-2.0	C/C++	Analisi protocolli	1998	2024
Tshark	GPL-2.0	C/C++	Analisi protocolli	1998	2024
Zeek	BSD	C++/Zeek	Analisi Rete	1995	2024

Tabella 4.2: Strumenti utili per il rilevamento dei Covert Timing Channel

Capitolo 5

Conclusione

La steganografia si conferma come uno strumento di inestimabile valore nel panorama attuale della sicurezza delle informazioni. Dalla sua antica origine fino alle sofisticate applicazioni moderne, essa ha dimostrato una straordinaria capacità di adattamento e utilità. L'era digitale ne ha esponenzialmente ampliato le possibilità di impiego, permettendo di nascondere dati all'interno di immagini, audio, video e persino nei protocolli di comunicazione di rete. Queste tecniche non solo proteggono la privacy individuale, ma giocano un ruolo cruciale nella protezione dei diritti d'autore, nelle comunicazioni militari e nel giornalismo investigativo.
Tuttavia, con l'aumento delle capacità di analisi dei dati e la crescente potenza computazionale, anche le tecniche di rilevamento della steganografia stanno diventando sempre più avanzate. Questo crea un costante gioco "del gatto e del topo" tra chi cerca di nascondere informazioni e chi cerca di scoprirle. In questo contesto, la ricerca e lo sviluppo in ambito steganografico sono essenziali per mantenere un vantaggio, innovarsi e migliorare le tecniche per superare le nuove metodologie di rilevamento. Questo implica non solo migliorare gli algoritmi di occultamento per rendere i dati nascosti ancora meno rilevabili, ma anche studiare i nuovi tipi di attacchi e analisi che potrebbero essere utilizzati per rilevarli.
In questa tesi abbiamo discusso su alcuni protocolli che permettono l'implementazione di Covert Channel: TCP/IP, ICMP, HTTP, DNS. Nel mondo reale ogni protocollo esistente può essere soggetto a Covert Channel, esistono difatti implementazioni, come il VOIP, dove i dati audio trasmessi possono essere utilizzati come vettori per informazioni nascoste, sfruttando la ridondanza del segnale audio o inserendo dati

all'interno di pacchetti di controllo o segnalazione, che sfociano in ambiti variegati e inaspettati dell'informatica.
Abbiamo inoltre trattato i Covert Channel nel loro funzionamento basilare, definendone la tassonomia, il problema dei prigioneri e i caratteri fondamentali. Ci siamo poi focalizzati in particolare sui Covert Timing Channel, una tipologia che anzichè modulare l'informazione modula il mezzo trasmissivo per oscurare un messaggio. I Covert Timing Channel rappresentano una sfida unica poiché sfruttano la temporizzazione dei pacchetti di dati piuttosto che il contenuto dei pacchetti stessi. Modificando i tempi di invio dei pacchetti, è possibile codificare informazioni in un modo che risulta estremamente difficile da rilevare con tecniche di analisi standard. Questo tipo di canale sfrutta la precisione e la prevedibilità del tempo per nascondere i dati, rendendolo un metodo altamente efficace per la steganografia in ambienti dove la sicurezza delle informazioni è critica.
La discussione si è poi spostata verso la letteratura contemporanea sui Covert Timing Channel e sulle possibili implementazioni sperimentali, in particolare, su vari studi recenti che hanno esplorato nuove tecniche per implementare Covert Timing Channel in contesti ed in sistemi operativi diversi, persino nelle comunicazioni wireless. Gli esperimenti condotti in questi studi hanno dimostrato che, con tecniche adeguate, è possibile raggiungere velocità di trasmissione sufficienti per applicazioni pratiche, pur mantenendo un basso profilo per evitare il rilevamento. Inoltre, la letteratura recente ha analizzato le contromisure possibili per contrastare questi canali nascosti. Tra le soluzioni proposte, ci sono metodi di randomizzazione dei tempi di trasmissione e l'utilizzo di algoritmi di machine learning per identificare pattern anomali nei tempi di risposta. La randomizzazione dei tempi di trasmissione rende più difficile per un potenziale attaccante prevedere e decodificare il messaggio nascosto, mentre gli algoritmi di machine learning possono essere addestrati su grandi volumi di dati per identificare comportamenti sospetti che potrebbero indicare la presenza di un Covert Channel.
Ci siamo poi spostati sulle attuali implementazioni e strumenti di rilevamento di Covert Timing Channel presenti in rete oggi. Le implementazioni hanno riguardato vere e proprie Covert Timing Channel che sfruttano i tempi di inter-invio dei pacchetti per occultare messaggi nascosti nella rete in chiaro, messaggi che solo chi conosce i delay può decifrare. Gli strumenti di rilevamento che abbiamo visto invece sono pro-

gettati per identificare e mitigare le comunicazioni nascoste che sfruttano i tempi di trasmissione dei pacchetti per trasmettere informazioni in modo furtivo. Includono strumenti basati su analisi del traffico di rete, che utilizzano tecniche di rilevamento basate su analisi statistica, che cercano deviazioni significative nei tempi di arrivo dei pacchetti rispetto a un modello di riferimento. L'efficacia di questi strumenti dipende dalla capacità di identificare con precisione le anomalie che indicano la presenza di un Covert Timing Channel senza generare troppi falsi positivi, che potrebbero sovraccaricare i sistemi di monitoraggio della rete.

I possibili sviluppi futuri della steganografia sono promettenti e variegati, con numerose direzioni che potrebbero rivoluzionare il modo in cui nascondiamo e proteggiamo le informazioni. Una delle possibilità più affascinanti è l'uso dell'intelligenza artificiale e dell'apprendimento automatico, che offrono il potenziale per creare metodi steganografici sempre più sofisticati e impercettibili. Questi strumenti avanzati non solo possono essere utilizzati per analizzare e migliorare le tecniche di steganografia esistenti, ma anche per sviluppare nuovi algoritmi in grado di adattarsi a cambiamenti nel panorama delle minacce. L'AI, con la sua capacità di apprendere e riconoscere pattern complessi, può ottimizzare la creazione e la gestione di tecniche di occultamento dei dati, rendendoli sempre più difficili da rilevare per gli analisti e i sistemi di sicurezza. In particolare, l'evoluzione delle tecnologie di machine learning sta permettendo la progettazione di sistemi che possono adattarsi dinamicamente a nuovi tipi di minacce e scenari, è possibile infatti identificare e implementare algoritmi di steganografia che possono adattarsi automaticamente ai cambiamenti nel comportamento del traffico di rete o nei formati dei file, mantenendo la loro efficacia nel tempo. Questa adattabilità rappresenta un vantaggio significativo, poiché consente alle tecniche steganografiche di rimanere efficaci anche quando vengono scoperte nuove tecniche di rilevamento. Un altro sviluppo significativo potrebbe derivare dalla steganografia quantistica. Con l'avanzamento delle tecnologie quantistiche, emergono nuove possibilità per la protezione delle informazioni grazie alle proprietà uniche dei qubit, fenomeni come l'entanglement e la sovrapposizione quantistica usati per nascondere informazioni in modi completamente nuovi, teoricamente impossibili da intercettare senza perturbare il sistema stesso. Le tecniche basate su qubit offrono metodi di trasmissione delle informazioni che si discostano completamente dai metodi classici, aumentando notevolmente la difficoltà di intercettazione e decodifica dei mes-

saggi nascosti, questo potrebbe non solo migliorare la sicurezza dei dati, ma anche aprire nuove strade per la creazione di sistemi di comunicazione che siano praticamente impenetrabili. Un'altra possibilità ancora è data dall'espansione dell'Internet delle cose (IoT) e delle reti 5G, nella quale la steganografia potrebbe trovare nuove e significative applicazioni. La proliferazione di dispositivi connessi ha creato una vasta superficie di attacco, aumentando la necessità di tecniche avanzate di protezione dei dati. L'obiettivo è proteggere le comunicazioni tra dispositivi IoT, che vanno dai domestici intelligenti ai veicoli autonomi, passando per altre tecnologie emergenti integrate nella nostra vita quotidiana. Le tecniche steganografiche applicate all'IoT potrebbero garantire che le comunicazioni tra dispositivi rimangano sicure, anche in ambienti altamente esposti e vulnerabili. La crescente integrazione di questi dispositivi rende cruciale l'implementazione di tecniche di steganografia, ad esempio adattiva, che possano affrontare la vasta gamma di minacce, per gestire la diversità dei dati trasmessi tra i dispositivi IoT, assicurando che le informazioni sensibili rimangano protette anche quando i protocolli e le tecnologie utilizzati cambiano. In un mondo in cui la sicurezza e la privacy delle informazioni sono costantemente minacciate, la steganografia continuerà a giocare un ruolo cruciale, caratterizzato dall'evoluzione di tecniche sempre più avanzate per nascondere e proteggere i dati, riflettendo l'incessante progresso tecnologico e le crescenti esigenze di sicurezza. La capacità di rimanere un passo avanti rispetto alle minacce rappresenta una sfida continua, ma anche un'opportunità per innovare e migliorare continuamente le strategie di difesa della nostra privacy e delle nostre comunicazioni. Con l'aumento della capacità computazionale e l'interconnessione globale, sarà essenziale sviluppare nuove tecniche di steganografia che possano resistere alle avanzate capacità di analisi dei potenziali attaccanti, assicurando che la protezione dei dati personali e delle comunicazioni sensibili rimanga robusta ed efficace.

Bibliografia

[1] Caesar cipher in cryptography. In *GeeksForGeeks*, 2024.

[2] Konstantin Kogos Anna Belozubova, Anna Epishkina. Random delays to limit timing covert channel. In *European Intelligence and Security Informatics Conference (EISIC)*, 2016.

[3] Shishir Nagaraja Arnab Kumar Biwas, Dipak Ghosal. A survey of timing channels and countermeasures. In *ACM Computing Surveys*, pages Vol. 50, No. 1, Article 6, March 2017.

[4] Punam Bedi and Arti Dua. Network steganography using the overflow fieldof timestamp option in an ipv4 packet. In *Procedia Computer Science*, page 171:1810–1818, 2020.

[5] Blake. Dns explained. hierarchy and architecture. 2020.

[6] Le Cai and Hao Chen. Touchlogger: inferring keystrokes on touch screen from smartphone motion. In *Usenix Conference on Hot Topics in Security*, pages 9–9, 2011.

[7] Michele Ceccarelli. Un nuovo approccio alla steganografia di rete basato sugli header http. In *Università degli Studi di Perugia, rel Francesco Santini, Stefano Bistarelli*, Anno Accademico 2021-2022.

[8] Scott Craver. On public-key steganography in the presence of an active warden. In *International Workshop on Information Hiding*, page 55–368. Springer, 1998.

[9] Creativemotions. Http vs https: differenze, pro, contro e quale scegliere. 2024.

[10] Virginia 22209 Defense Advanced Research Projects Agency Information Processing Techniques Office 1400 Wilson Boulevard Arlington. Internet protocol. September 1981.

[11] Roberto Natella Domenico Cotroneo, Luigi De Simone. Timing covert channel analysis of the vxworks mils embedded hypervisor under the common criteria security certification. In *Research Gate*, 2021.

[12] Olivier Le Moigne Donald Latham and John Morris Roberts. Department of defense trusted computer system evaluationcriteria. In *Department of Defense*, page 198, 1986.

[13] Olivier Le Moigne Emanuele Jones and John Morris Roberts. Ip traceback solutions based on time to live covert channel. In *Proceedings. 2004 12th IEEE In-ternational Conference on Networks (ICON 2004)(IEEE Cat. No. 04EX955)*, pages volume 2, pages 451–457. IEEE, 2004.

[14] Fordummies. Network basics: Tcp/ip protocol suite. 2022.

[15] GeeksforGeeks. Internet control message protocol (icmp). 2024.

[16] Google. Spdy. In *The Chromium Projects*.

[17] Theodore Handel and Maxwell Sandfor. Hiding data in the osi net-work mode. In *International Workshop on Information Hiding*, page 23–38. Springer, 1996.

[18] Jianping Wang Hermine Hovhannisyan, Kejie Lu. A novel high-speed ip-timing covert channel: Design and evaluation. In *IEEE International Conference on Communications (ICC)*, 2015.

[19] California 90291 Information Sciences Institute University of Southern California 4676 Admiralty Way Marina del Rey. Transmission control protocol. September 1981.

[20] Guru Venkataramani Jie Chen. An algorithm for detecting contention-based covert timing channels on shared hardware. In *Research Gate*, 2014.

[21] Linfan Wang Zhe Wang Jinpu Xie, Yonghong Chen. A network covert timing channel detection method based on chaos theory and threshold secret sharing. In *IEEE Information Technology, Networking, Electronic and Automation Control Conference*, 2020.

[22] Sathiamoorthy Manoharan Jun Seo and Aniket Mahanti. Discussion and review of network steganography. In *2016 IEEE 14th Intl Conf on Dependable,Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence andComputing, 2nd Intl Conf on Big Data Intelligence and Computing and CyberScience and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, page 384–391. IEEE, 2016.

[23] Richard Stevens Kevin Fall. Tcp/ip illustrated. In *Addison-Wesley Professional*, 2012.

[24] Butler Lampson. A note on the confinement problem. In *Communications of the ACM*, page 16(10):613–615, 1973.

[25] Karl Matthes. Domain name system. 2020.

[26] Lori Watkins Charlie Corbett Mengjun Yue, William Robinson. Constructing timing-based covert channels in mobile networks by adjusting cpu frequency. In *Proceedings of the Third Workshop on Hardware and Architectural Support for Security and Privacy*, pages 1–8, 2014.

[27] Paul Mockapetris. Domain names - implementation and specification. In *ISI*, November 1987.

[28] BBC News. Airlines bomb plot: The e-mails. In *BBC*, page 1996. Springer, September 2009.

[29] Muhammad Anan Nidal Nasser Omar Darwish, Ala Al-Fuqaha. The role of hierarchical entropy analysis in the detection and time-scale determination of covert timing channels. In *International Wireless Communications and Mobile Computing Conference, IWCMC*, 2015.

[30] Oracle. How the tcp/ip protocols handle data communications. 2013.

[31] Huo-Chong Ling Osamah Ibrahiem Abdullaziz, Vik Tor Goh and KokSheik-Wong. Network packet payload parity based steganograph. In *2013 IEEE Conference on Sustainable Utilization and Development in Engineering and Technology (CSUDET)*, page 56–59. IEEE, 2013.

[32] Dimitris Mouris Nektarios Georgios Tsoutsos Chengmo Yang Patrick Cronin, Charles Gouert. Covert data exfiltration using light and power channels. In *IEEE International Conference on Computer Design: VLSI in Computers and Processors, (ICCD)*, 2019.

[33] Zhonggui Bao Peng Yang, Hui Zhao. Research on the influence of network condition to network covert timing channels. In *IEEE International Conference on Software Engineering and Service Sciences (ICSESS)*, 2015.

[34] Jingsong Hu Pengfei Xue, Hanlin Liu and Ronggui Hu. A multi-layer steganographic method based on audio time domain segmented and network steganograph. In *AIP Conference Proceedings*, pages volume 1967, page 020046. AIPPublishing LLC, 2018.

[35] Birgit Pfitzmann. Information hiding terminology. In *Information Hiding: First International Workshop Proceedings*, page 1996. Springer, 1996.

[36] Jon Postel. User datagram protocol. In *ISI*, 28 August 1980.

[37] Jon Postel. Internet control message protocol. In *ISI*, September 1981.

[38] Xinyu Zhou Mihir Intwala Apu Kapadia XiaoFeng Wang Ravi Schlegel, Kun Zhang. Soundcomber: A stealthy and context-aware sound trojan for smartphones. In *Network and Distributed System Security Symposium*, pages 17–33, 2011.

[39] Craig Rowlan. Covert channels in the tcp/ip protocol suite. In *MunksgaardInternational Publishers Ltd*, Copenhagen, 1997.

[40] Jeffrey Mogul Henrik Frystyk Larry Masinter Paul Leach Tim Berners-Lee Roy Fielding, Jim Gettys. Hypertext transfer protocol – http/1.1. In *Compaq, W3C, MIT, Microsoft*, June 1999.

[41] Giovanni Sacheli. Che cos'è il protocollo http? 2020.

[42] Grenville Armitage Sebastian Zander and Philip Branc. A survey of covert channels and countermeasures in computer network protocols. In *IEEECommunications Surveys Tutorials*, page 9(3):44–57, 2007.

[43] Shishir Yuanzhu Chen Shorouq Al-Eidi, Omar Darwish. Covert timing channel analysis either as cyber attacks or confidential applications. In *Machine Learning for IoT Applications and Digital Twins*, 2020.

[44] Chunhui Zhao Shunpu Li, Lejun Zhang. An overview of android covert channel. In *IEEE Advanced Information Management,Communicates,Electronic and Automation Control Conference (IMCEC)*, 2018.

[45] Gustavus Simmons. The prisoners' problem and the subliminal channel. In *Advances in Cryptology*, page 51–67. Springer, 1984.

[46] Bernhard Fechner Steffen Wendzel, Sebastian Zander and Christian Herdin. Pattern-based survey and categorization of network covert channel techniques. In *CM Computing Surveys (CSUR)*, page 47(3):1–26, 2015.

[47] Wojciech Mazurczyk Aleksandra Mileva Jana Dittmann Christian Krätzer Kevin Lamshöft-Claus Vielhauer LauraHartmann Jörg Keller et al. Steffen Wendzel, Luca Caviglione. A generic taxonomy for steganography methods. In *TechRxiv*, 2022.

[48] Alexandra Dmitrienko Stephan Heuser Ahmad-Reza Sadeghi Bhargava Shastry Sven Bugiel, Lucas Davi. Practical and lightweight domain isolation on android. In *ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, pages 51–62, 2011.

[49] Alexandra Dmitrienko Thomas Fischer Ahmad-Reza Sadeghi Sven Bugiel, Lucas Davi. Xmandroid: A new android evolution to mitigate privilege escalation attacks. 2011.

[50] Krzysztof Szczypiorski. Steganography in tcp/ip networks. In *State of the art and a proposal of a new system-hiccups. Warsaw University of Technology, Poland Institute of Telecommunications, Warsaw, Poland*, 2003.

[51] Kevin D. Fairbanks T. Owens Walker. An off-the-shelf, low detectability, low data rate, timing-based covert channel for ieee 802.11 wireless networks. In *Consumer Communications and Networking Conference, CCNC IEEE*, 2018.

[52] Steve White. Covert distributed processing with computer viruses. In *Conference on the Theory and Application of Cryptology*, page pages 616–619. Springer, 1989.

[53] Sebastian Zander Amir Houmansadr and Krzysztof Szczypiorski Wojciech Mazurczyk, Steffen Wendzel. Information hiding in communication networks:fundamentals, mechanisms, applications, and countermeasures. In *John Wiley Sons*, 2016.

[54] Steffen Wendzel Wojciech Mazurczyk and Krzysztof Cabaj. Towards deriving insights into data hiding methods using pattern-based approach. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, page 1–10, 2018.

[55] Nan Zhang Muhammad Naveed XiaoFeng Wang Xinyu Zhou, Yajin Lee. The peril of fragmentation: Security hazards in android device driver customizations. In *Security and Privacy*, pages 409–423, 2014.

[56] Nan Zhang Muhammad Naveed XiaoFeng Wang Xinyu Zhou, Yajin Lee. Seeing the unseen: Revealing mobile malware hidden communications via energy consumption and artificial intelligence. In *IEEE Transactions on Information Forensics Security*, pages 799–810, 2017.

[57] Yiqi Dai Xiong Liu, Haiwei Xue. A self adaptive jamming strategy to restrict covert timing channel. In *International Symposium on Intelligence Information Processing and Trusted Computing (IPTC)*, 2011.

[58] Ting Liu Yu Qu Yanan Sun, Xiaohong Guan. An identity authentication mechanism based on timing covert channel. In *IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2012.

[59] Huifang Chen Yuqing Jia, Lei Xie. Bionic covert underwater acoustic communication based on dolphin whistle with time-delay. In *IEEE International Conference on Communications in China (ICCC)*, 2021.

Ringraziamenti

Ringrazio di cuore il relatore di questa tesi Santini per la sua pazienza e grandissima disponibilità, per il suo aiuto fondamentale nella scrittura di questa tesi. Non sarei sicuramente riuscito a prepararla in tempo per settembre se non fosse stato per lui.
Ringrazio di cuore tutti i miei amici e compagni di corso che mi hanno accompagnato in questi anni: Daniel, Eduard, Luca, Alessandro, Chiara, Filippo, Beatrice, Jannik, Gianmarco, Leonardo, Michele et al. Senza di voi non sarei sicuramente riuscito ad arrivare fino a qui, abbiamo perdurato e lottato come i cavalieri della tavola rotonda e, alla fine, siamo riusciti a prevalere su ogni avversità, siete meravigliosi.
Ringrazio di cuore tutti i miei amici di Teatro e della palestra che, con la loro compagnia e voglia di vivere mi hanno rallegrato in tante occasioni.
Ringrazio tutti coloro che non ho ancora ringraziato e mi sono stati accanto nel tempo, gli amici di una vita, quelli appena conosciuti e quelli che conoscerò, vi voglio bene.
Ringrazio di cuore la mia famiglia, mio padre Dario e mia madre Isabella, mia sorella Alessandra, mia nonna Chiarina e mio nonno Marcello, mio cugino Marco e mia zia Donatella. Grazie per essermi stati accanto nei momenti duri e quelli felici, di avermi spronato nonostante tutto e di avermi reso quello che sono oggi, sono fiero di voi.
Ringrazio di cuore anche me stesso, per essermi fatto forza fino a raggiungere il mio obiettivo, per averci creduto fino alla fine, per aver creduto in chi mi sta accanto.
Viviamo una vita di cui essere orgogliosi.

Edoardo Tommasi