

UNIVERSITÀ DEGLI STUDI DI PERUGIA  
Dipartimento di Matematica e Informatica



TESI MAGISTRALE IN INFORMATICA

Sviluppo di un Covert Channel tramite il protocollo ICMP per l'esfiltrazione di dati

*Relatore*

**Prof. Santini Francesco**

*Laureando*

**Mecarelli Marco**

---

Anno Accademico 2024-2025

---

# Indice

<b>1 Parte 1 - Introduzione</b>	<b>5</b>
<b>Introduzione</b>	<b>5</b>
1.1 Strumenti Utilizzati . . . . .	5
<b>2 Parte 2 - Covert Channel implementati</b>	<b>9</b>
<b>Covert Channel implementati</b>	<b>9</b>
2.1 Timing Covert Channel . . . . .	9
2.1.1 Covert Channel Temporale tramite pacchetti ICMP Echo . . .	9
2.2 Storage Covert Channel . . . . .	11
2.3 Struttura della comunicazione fra le entità . . . . .	13
<b>Test e Risultati</b>	<b>16</b>



## Listings

## Elenco delle figure

1	.....	6
2	Traffico relativo al comando <i>ls -s</i> .....	7
3	Architettura generale <i>icmp tunnel</i> .....	8
4	Diagramma delle entità presenti .....	14
5	Diagramma del dlusso di comunicazione fra le entita .....	15

# 1 Parte 1 - Introduzione

ICMP (Internet Control Message Protocol) è un protocollo che opera al livello di rete (livello 3 nel modello ISO/OSI). e permette la **segnalazione errori**, la **diagnostica di rete** e la **messaggistica di controllo**. Proprio per questo che viene utilizzato per il monitoraggio dello stato di una rete e per la risoluzione dei problemi che avvengono in essa. Data la sua necessità per la diagnostica di rete e la segnalazione degli errori, può essere utilizzato in modo improprio per mettere a segno degli attacchi o per studiare la rete (ricognizione della rete).

Nei seguenti capitoli verrà illustrato come può essere sfruttato per la creazione di un Covert Channel; un attacco che permette (in ambienti ritenuti sicuri) la capacità di comunicare e/o trasferire dati in maniera non autorizzata e non voluta. L'attacco opera al di fuori degli usuali meccanismi di comunicazioni e per questo risulta difficile da rilevare e/o identificare. Sia dagli amministratori che dai tipici strumenti di monitoraggio. Infine, siccome qualsiasi risorsa condivisa può essere utilizzata per la sua creazione, può esistere in qualunque sistema.

## 1.1 Strumenti Utilizzati

### Virtual Box

Il codice sviluppato è stato testato in un ambiente Linux. Per poter far ciò sono state create, per ciascun entità necessaria, una macchina virtuale contenente Ubuntu. Alla fine si sono ottenute quattro macchine virtuali: una per l'attaccante e la vittima, mentre le altre due per i proxy. Si poteva usare anche un singolo proxy ma si voleva testare anche come i dati ricavati dalla vittima, e da inoltrare all'attaccante, venissero distribuiti ai proxy connessi a essa.

### Scapy

Scapy è un framework per la manipolazione dei pacchetti scritto in Python che consente di falsificare molti tipi di pacchetti (http, tcp, ip, udp, icmp, ecc.) È in grado di creare o decodificare pacchetti di vari protocolli. Inoltre può inviarli in rete, catturarli, memorizzarli o leggerne i dati.

Svolge principalmente due funzioni: invia i pacchetti e riceve le risposte, consentendo all'utente di inviare, intercettare, analizzare e falsificare pacchetti di rete. Questa capacità consente la creazione di strumenti in grado di sondare, scansionare o attaccare le reti.

Nella libreria il metodo **send** (o simili) permetteranno di inviare un definito pacchetto. Per definire un pacchetto basterà concatenare i livelli che dovranno essere presenti, e opportunamente inizializzati; mentre per poter ascoltare il traffico di ree basterà una variabile di tipo *AsyncSniffer*.

## RITA

AAAAA

## ICMP Door

È stato studiato per comprendere come potesse effettuare il tunneling dei dati. Oltre alla struttura delle entità, che successivamente verrà ridefinita, risulta interessante come il programma richiede degli argomenti dall'utente. Tramite la libreria *argparse* richiede all'utente l'interfaccia su cui ascoltare i dati e l'indirizzo di destinazione dei pacchetti. Inoltre usa il metodo *sniff* del framework Scapy per ascoltare il flusso dei dati mentre tramite *sr* invia i pacchetti.

Sebbe il programma ci introduce a una possibile struttura del Covert Channel; gli si sono trovati dei difetti. Gli svantaggi sono che i dati vengono trasemssi non solo nel campo data, del messaggio di tipologia ICMP Echo Reply, ma anche in chiaro. Inoltre il valore del campo identifier rimano invariato per tutta la sessione. Un sistema di sicurezza, se vedesse le molteplici risposte (che non combaciano con il numero di richieste) e leggesse i testi in chiaro, potrebbe identificare il canale nascosto. Di solito per ogni Echo Request corrisponde una singola Echo Reply in cui la risposta rimanda i dati ricevuti e il campo data di solito contiene frasi già preimpostate e sempre costanti (e.g. '*helloworld*').



Figura 1

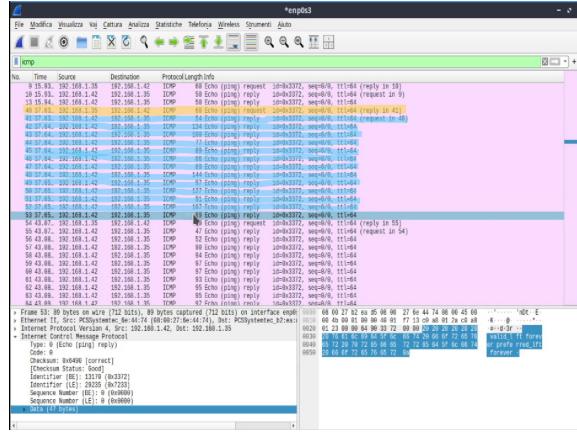


Figura 2: Traffico relativo al comando *ls -s*

### •ICMP Exfil

Analizzato per vedere come un Covert Channel temporizzato postesse funzionare. Riceve un dato, lo converte in binario e dopodichè avrà una lista di numeri binari. Per mandare il dato, invia un ping con un timeout pari a **binary\_number+leway**. Il valore leway viene utilizzato per rallentare il numero di pacchetti inviati ed avere una connessione maggiormente silenziosa. L'autore infine indica come miglioria, la crittografia dei dati per aggiungere del rumore, dell'entropia.

Ciò su cui si affida è che un osservatore, vedendo i pacchetti ICMP, li veda come vailidi; iccome non riuscirebbero a trovare alcun dato, a meno che non sappiano della tecnica utilizzata.

### •ICMP Tunnel

Strumento che permette il tunneling del traffico IP. Tramite delle richieste e risposte ICMP Echo, incapsula il traffico e lo invia al server proxy. Questi ultimo lo decapsula e lo inoltrano. I pacchetti in entrata, che sarebbero diretti alla macchina vittima, sono poi incapsulati dal proxy e inviati. L'approccio è possibile siccome RFC-792, che indica le linee guida del protocollo ICMP, permette una quantità arbitraria di dati nei pacchetti ICMP Echo (sia richiesta che risposta).

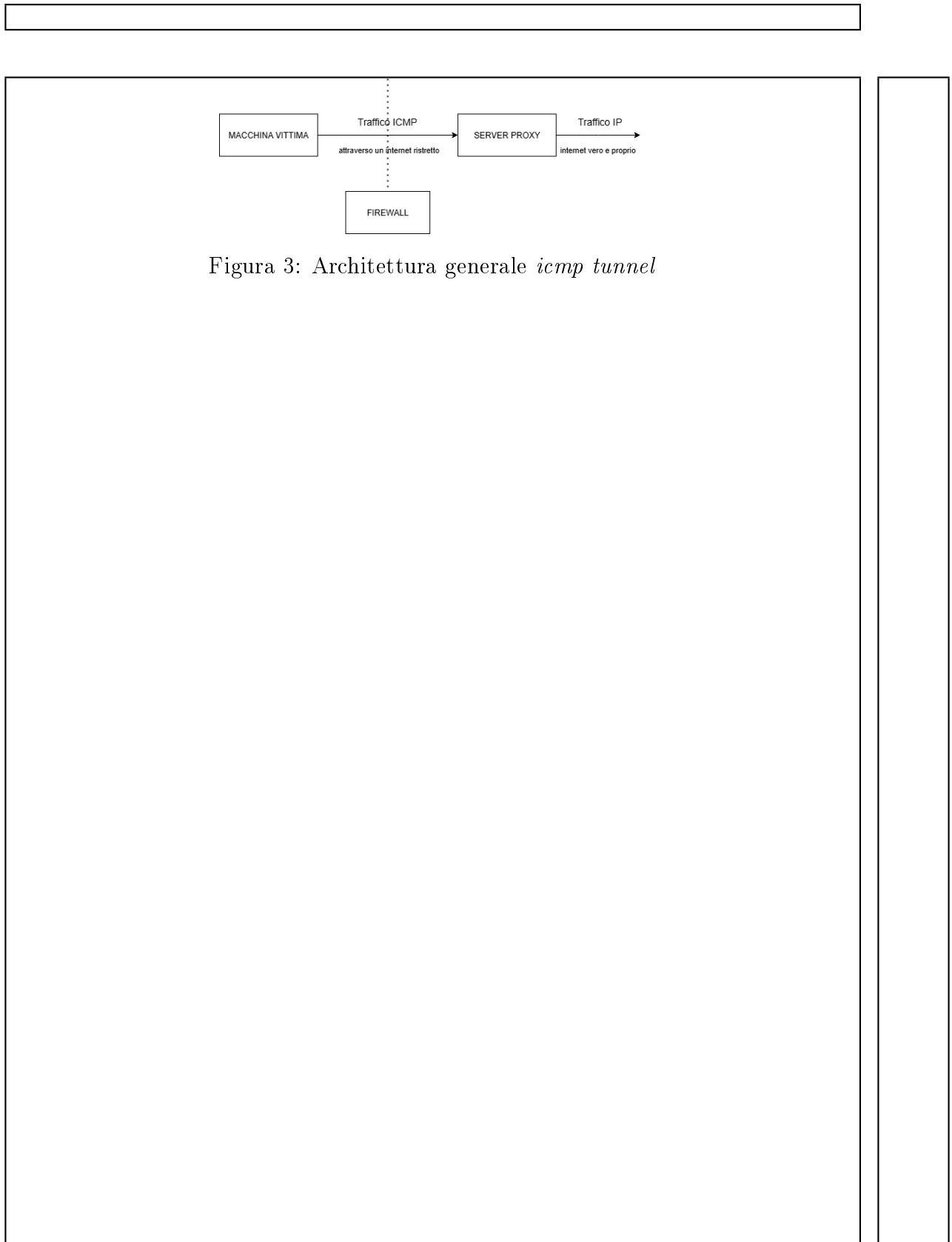


Figura 3: Architettura generale *icmp tunnel*

## 2 Parte 2 - Covert Channel implementati

Le principali categorie di Covert Channel sono:

- Timing Covert Channel (Temporizzazione):

Sfruttano gli intervalli di tempo o l'ordine degli eventi per codificare informazioni (e.g. ritardi fra i pacchetti di rete,...).

- Storage Covert Channel (Archiviazione):

Implicano la scrittura di dati su un'area di memoria condivisa accedibile da entrambi i processi. I veicoli saranno tutte quelle risorse che consentono la scrittura (diretta o indiretta) da parte di un processo e la lettura (diretta o indiretta) da parte di un altro. Un processo scrive su una risorsa condivisa, mentre un altro processo legge da essa.

### 2.1 Timing Covert Channel

#### 2.1.1 Covert Channel Temporale tramite pacchetti ICMP Echo

In questo caso si definisce un Covert Channel Temporale inviando pacchetti ICMP (di tipo Echo) in intervalli specifici. Tramite la differenza fra il tempo precedente e il tempo di ricezione del nuovo pacchetto; l'attaccante riuscirà a codificare un bit (o coppie di bit) mentre la vittima riuscirà a decodificare il bit (o la coppia di bit) associato all'intervallo.

La formula per calcolare i tempi associati alle codifiche dei bit è la seguente:

$$\text{tempo\_base} + \text{index} * (2 * \text{distanza\_tempi}) \quad (1)$$

In essa il **tempo di base** indica il minimo dei secondi da aspettare per ogni tempo possibile. L'indice indicherà l'indice associato alla codifica e siccome si trattano bit, corrisponderà all'intero rappresentato dalla coppia di bit (eg 101=5). Invece la distanza è la differenza minima di tempo che tutti i tempi calcolati dovranno avere.

Siccome RITA riesce ad individuare gli schemi temporali calcolati, si è definito un range così da poter mandare tempi randomici e non costanti. Ora la codifica di una coppia di bit, non è più definita da un singolo, costante intervallo di tempo, ma da

un range di intervalli temporali. Quindi se si volesse mandare il codice *01*, non si aspetteranno più 7 secondi, ma un tempo randomico che appartiene al range range [5, 9].

Inoltre si è pensato di introdurre anche un tempo (in minuti) che definirà quanto aspettare una volta che si è mandato un byte. Ciò potrà essere usato se lo scambio di informazioni risultasse prolungato; infatti un numero elevato e continuo di pacchetti potrebbe risultare anomalo.

Il dato da inviare (di qualunque tipo esso sia) dovrà essere convertito in una lista di byte, dalla quale estrarremo i singoli bit da mandare.

Dopodichè avverrà la comunicazione vera e propria. Si manderà un pacchetto per indicare al destinatario l'inizio della comunicazione e impostare così il tempo di inizio. Se non si determina questo tempo, il destinatario non riuscirà a calcolare l'intervallo di tempo fra il tempo di arrivo del pacchetto corrente e di quello del pacchetto precedente. Di conseguenza non si riuscirà a decodificare il primo bit. Successivamente si itererà l'intera lista contenente i singoli bit e si aspetterà il tempo associato.

<b>Num bit</b>	<b>IPv4</b>	<b>IPv6</b>
1	8:33	14:46
2	7:20	12:08
4	12:00	22:20

Tabella 1: Tempi di esecuzione del Timing Channel

L'unica differenza presente nella versione (del Timing Channel) che utilizza il protocollo IPV6, è la struttura del pacchetto. Si aggiungerà un livello *Ether* in cui si specifica l'indirizzo MAC di destinazione e l'indirizzo MAC sorgente. Inoltre nel livello *IPv6* bisognerà definire lo ScopeId dell'indirizzo di destinazione; ciò verrà fatto specificando nel campo non solo l'indirizzo IP di destinazione, ma anche l'interfaccia per poter raggiungerlo.

## 2.2 Storage Covert Channel

Verranno utilizzate le tipologie di messaggi presenti nel protocollo ICMP, per codificare dei dati nei campi presenti. Nelle tabelle sono stati indicati quali campi verranno sfruttati, per ogni tipologia di messaggio ICMP, oltre alla quantità di byte codificabili.

Tipologia	Byte trasmessi	Campi Sfruttati	Uso
Destination Unreachable	3-8	unused, header+64 bits	Una destinazione (rete, porta,...) risulta non disponibile
Source Quench	3-8	unused, header+64 bits	Un gateway notifica il buffer di memoria per i pacchetti pieno (indica la congestione nella rete).
Redirect Message	3-4	header+64 bits	Suggerisce un reindirizzamento del pacchetto verso un percorso migliore.
Time Exceeded	3-8	unused, header+64 bits	Il gateway notifica che il TTL del pacchetto ricevuto risulta zero.
Parameter Problem	4-8	pointer, unused, header+64 bits	Il gateway rileva dei problemi nei campi dell'intestazione
Echo Request	2	identifier, data	Usato per inviare un dato a un destinatario e ricevere una risposta indietro.

--	--	--	--

Echo Reply	2	identifier, data	Replica i dati ricevuti nella richiesta rimandandoli al mittente
Timestamp Request	5	identifier,timestamp, data	Usato per inviare una serie di timestamp a un destinatario e ricevere una risposta indietro.
Timestamp Reply	5	identifier,timestamp, data	Replica i timestamp ricevuti nella richiesta rimandandoli al mittente
Information Request	2	identifier	Permette di scoprire se l'host si trova nella stessa rete di chi ha risposto. Nel mandare il pacchetto lascia il campo <i>destinazione</i> vuoto.
Information Reply	2	identifier	Risponde alla richiesta con tutti i campi compilati correttamente. In particolare quello relativo al proprio indirizzo IP.

Tabella 3: Tipologie di messaggi ICMPv4

--

Tipologia	Codici	Campi Sfruttati	Uso
Destination Unreachable	4-8	unused, invoking packet	Una destinazione (rete, porta,...) risulta non disponibile
Packet Too Big	8	mtu, invoking packet	Un router notifica l'impossibilità nell'inoltrare un pacchetto (indica la congestione nella rete)
Time Exceeded	4-8	unused, invoking packet	Il gateway notifica che il TTL del pacchetto ricevuto risulta zero.
Parameter Problem	8	pointer, invoking packet	Il gateway rileva dei problemi nei campi dell'intestazione
Echo Request	2	identifier, data	Usato per inviare un dato a un destinatario e ricevere una risposta indietro.
Echo Reply	2	identifier, data	Replica i dati ricevuti nella richiesta rimandandoli al mittente

Tabella 5: Tipologie di messaggi ICMPv6

### 2.3 Struttura della comunicazione fra le entità

La struttura del programma è definita nel seguente modo: sono presenti tre entità, che verranno definite come **attaccante**, **proxy** e **vittima**. Il caso standard sarà quello rappresentato in [Fig.4]; tuttavia sarà possibile, per l'attaccante, comunicare direttamente con la vittima. In questo caso l'entità proxy non verrà utilizzata ed

alcune funzioni presenti in essa verranno eseguite dall'attaccante (e.g connettersi alla vittima).

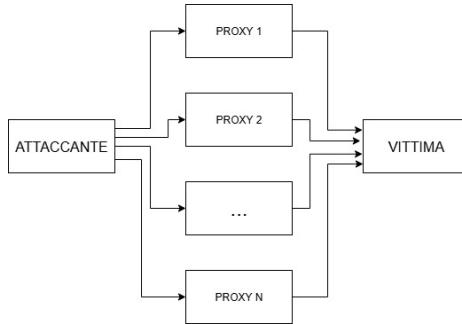


Figura 4: Diagramma delle entità presenti

Come avviene la comunicazione fra le entità è definito dal diagramma in [Fig.5]. Le entità inizializzano i parametri necessari, in particolare l'attaccante leggerà un file di configurazione in cui si indicano determinate cose. In particolare si indicano i proxy se si vorrà utilizzare per l'attacco, l'indirizzo IP della macchina vittima oltre alla tipologia di attacco che si vuole effettuare. Dopodichè l'attaccante si connetterà ad ogni proxy specificato nel file per testare se la connessione è possibile. La connessione viene ritenuta possibile se il proxy e l'attaccante riescono a comunicare fra di loro e se anche il proxy e la vittima riescono a comunicare fra di loro. Identificati i proxy sfruttabili, si inserirà un comando e, dopo averlo inviato ad uno dei proxy, si aspetteranno che i proxy ritornino i dati relativi all'operazione. Per l'invio del comando si utilizzerà un singolo proxy; quindi a tutti gli altri verrà notificato di aspettare direttamente i dati. Infine aspetta che tutti gli altri traccanti e poi la analizza, per poterli riordinare.

Il proxy invece, al momento dell'inizializzazione, imposta un server su cui l'attaccante si connerterà. Successivamente cercherà di connettersi alla vittima inviandogli un messaggio ICMP e poi aggiorna l'attaccante se la connessione è stata stabilita o no. Dopodichè rimarrà in attesa del comando che l'attaccante potrà inviare. Se lo riceve provvederà ad inoltrarlo altrimenti rimane in attesa dei dati dalla vittima. Una volta ricevuti tutti i dati dalla vittima, procederà ad inoltrarli all'attaccante. Infine rimarrà in attesa di aggiornamenti da parte dell'attaccante, il quale può decidere se inviare un altro comando o terminare la comunicazione.

La vittima, durante la sua inizializzazione, specifica il numero minimo di proxy ne-

cessari per la comunicazione. Quindi si metterà in ascolto dei proxy che vogliono connettersi; quando si connette ad un proxy, gli manderà un messaggio di conferma. Se entro un certo tempo il numero non viene raggiunto, si può scegliere se continuare ad aspettare altri proxy o avanzare con l'attacco. Dopodichè la vittima rimane in attesa del comando dell'attaccante e una volta arrivato, lo esegue e ricava i dati restituiti in caso dall'operazione. Infine, per esfiltrare i dati, invierà a ciascuno dei proxy connessi una porzione delle informazioni ricavate. Terminato di farlo, si metterà in ascolto del prossimo comando dell'attaccante, che determinerà la prossima operazione o la terminazione della comunicazione.

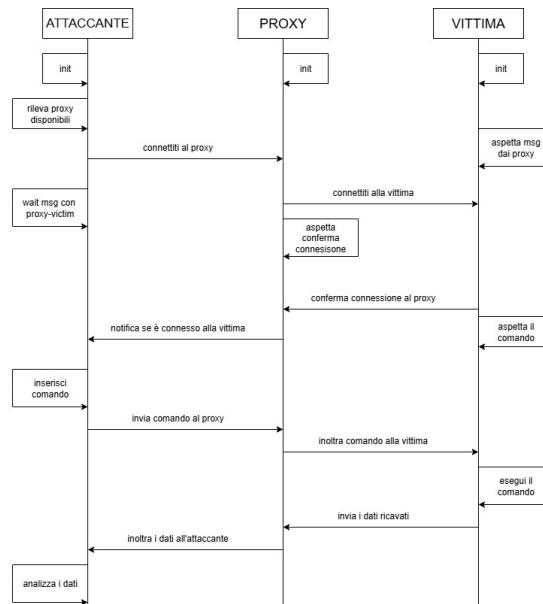
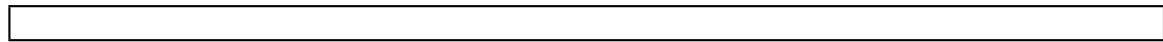


Figura 5: Diagramma del dlusso di comunicazione fra le entita



## Parte 3 - Test e Risultati

