



GitHub:

https://github.com/cirodevita/DNS_Steganography

DNS Steganography



Progetto “Cyber Security”

Prof. A. Castiglione

Ciro Giuseppe De Vita
0120000191



A.A. 2020-2021

Indice

01

Introduzione

05

Ambiente di sviluppo

02

Stego-Sistema proposto

06

Test e valutazioni

03

Client Side

07

Conclusioni

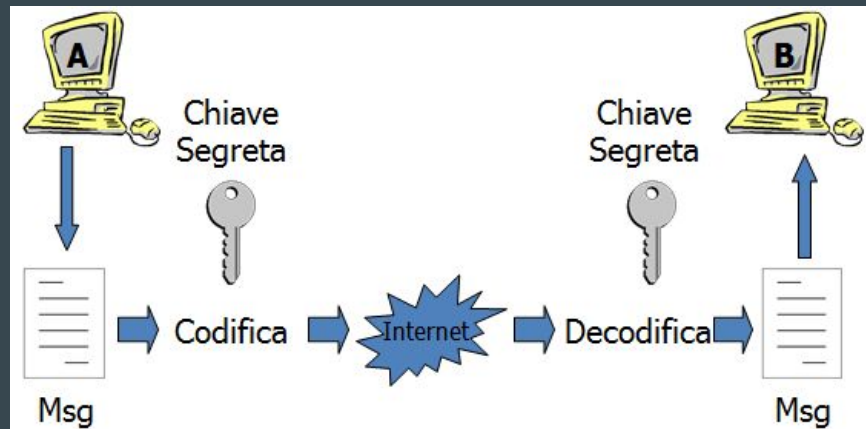
04

Server Side

Introduzione

Crittografia

- La tecnica più comune per trasferire informazioni sicure è la **crittografia**, che utilizza algoritmi per mascherare ciò che si vuole comunicare
- Sebbene ciò renda le informazioni illeggibili da parte di terzi, non nasconde il fatto che la comunicazione sia in corso



Covert Channel

- I **covert channel** sono utilizzati per tentare di nascondere l'esistenza di un canale di comunicazione
- L'applicazione originale dei covert channel era quella di risolvere il **problema dei prigionieri**, in cui due parti vogliono comunicare ma la comunicazione è mediata da un guardiano, denominato **Warden**, che è in grado di leggere i messaggi e determinare se sono autorizzati
- Dunque, le due parti devono escogitare un modo per nascondere la loro conversazione segreta in un modo che non sembri sospetto

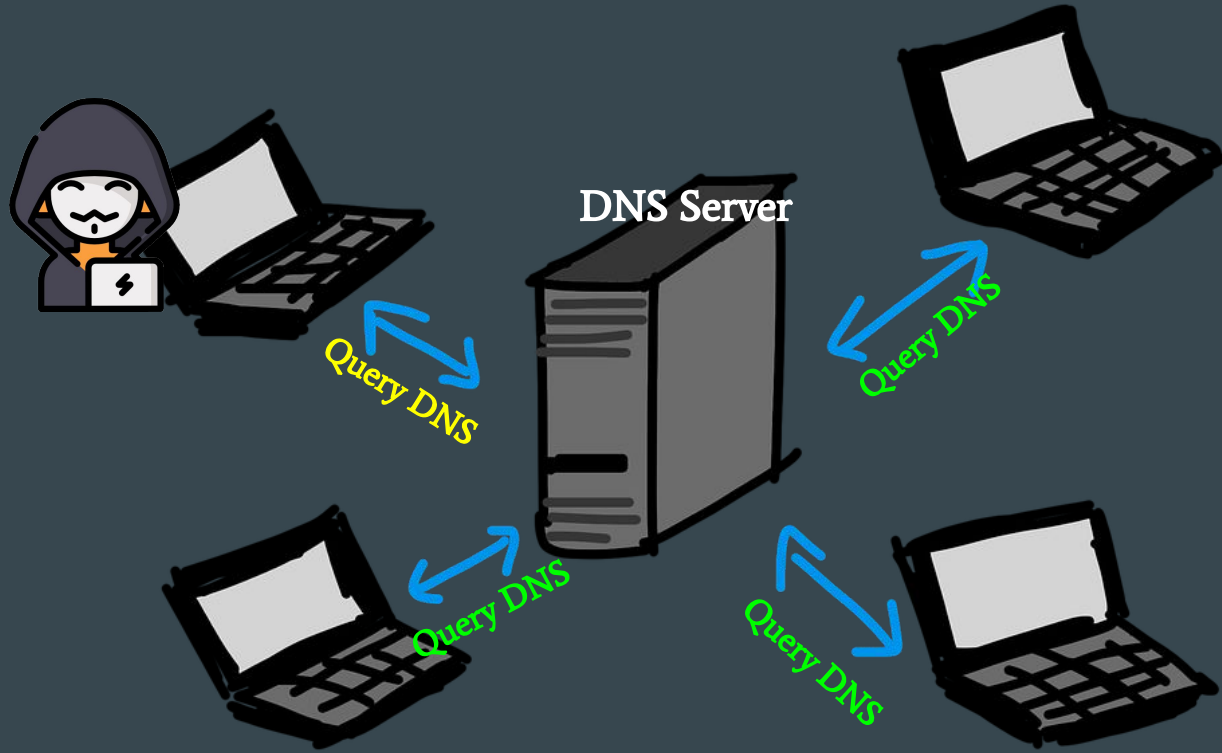
Covert Channel

- Con l'enorme quantità di traffico su Internet, i protocolli di rete sono diventati un **veicolo** comune per i covert channel, che in genere nascondono le informazioni negli **header fields** dei pacchetti
- Protocolli:
 - HTTP
 - TCP/IP
 - DNS

Steganografia

- A differenza della crittografia, la **steganografia** non cifra le informazioni, ma le nasconde solamente. Potremmo dire che la steganografia è l'arte di **nascondere** le informazioni all'interno di altri dati
- Per esempio, le informazioni possono essere nascoste all'interno di:
 - immagini
 - file sonori
 - video
 - network

Idea

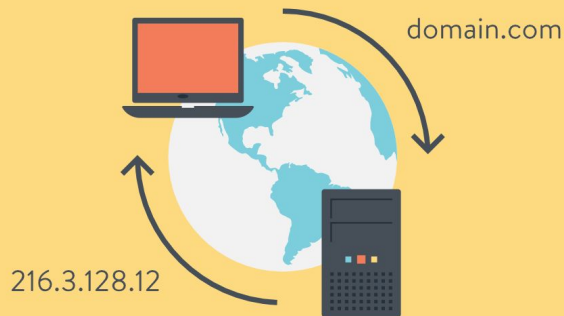


Protocollo DNS

- Uno dei protocolli più importanti per il funzionamento di Internet è il **DNS**. Il **Domain Name System** (DNS) è di vitale importanza per qualsiasi risorsa connessa a Internet o alla rete privata
- Lo usiamo ogni volta che apriamo pagine **Internet** o controlliamo le nostre **e-mail** e sarà molto difficile immaginare l'architettura della moderna rete di comunicazione senza DNS
- È un buon candidato per **nascondere informazioni** al suo interno, considerando anche il fatto che le query DNS sono **poco filtrate** a causa della grande necessità di accesso a Internet

Protocollo DNS

- Il DNS viene utilizzato per tradurre il nome mnemonico del server nel suo indirizzo IP corrispondente
- Il DNS si basa sul protocollo **UDP**, il che significa che è **connectionless** e ha una bassa affidabilità



Stego-Sistema proposto

Analisi steganografica

- Dalle specifiche del protocollo DNS si può notare che una query DNS può contenere anche **risposte**, infatti il protocollo non vieta l'inserimento di una risposta, ma lo consiglia solamente (**should**)
- Il punto precedente è stato confermata con test effettuati inviando query standard e preparate a un server **DNS di Google** (IP: **8.8.8.8**), e seguendole in **Wireshark**, un programma per il monitoraggio del traffico di rete

Analisi steganografica

Da tale analisi sono state tratte le seguenti conclusioni:

- il server DNS che elabora le query ha ignorato le query distorte, ovvero le query contenenti header fields formattati in modo non standard
- anche le query DNS (messaggi con un flag QR impostato su 0) possono avere **risposte; tale query non viene trattata come distorta**

Protocollo DNS - Answer

- **Name:** contiene il nome dell'oggetto, della zona o del dominio che identifica la query
- **Type:** contiene il tipo di record. Il tipo più popolare è il record **A**, ovvero una query per ottenere l'indirizzo IPv4 del dominio specificato nel campo Nome. Rispettivamente, **AAAA** è una query per ottenere l'indirizzo IPv6
- **Class:** definisce la classe di una query e di solito ha il valore "I" che è **IN** (Internet)
- **TTL:** serve a dire al server ricorsivo o al resolver locale per quanto tempo deve mantenere tale record nella sua cache. Più lungo è il TTL, più a lungo il resolver conserva le informazioni nella sua cache. Minore è il TTL, minore è il tempo in cui il resolver conserva tali informazioni nella sua cache
- **RData:** contiene byte di dati. Ad esempio, per un record di tipo A (una query base per un DNS), sono richiesti quattro byte contenenti un indirizzo IPv4

Campo utilizzato: Answer -> TTL

- Usato per inserire e nascondere informazioni di **start session**
- Composto da **16 bit**
 - Utilizzati solamente 12
 - I restanti sono stati generati casualmente

Protocollo DNS - Header

- **ID**: può essere visto come una chiave di autenticazione per ogni richiesta DNS e dovrebbe essere abbastanza casuale da assicurarsi che ogni query abbia un ID DNS univoco
- **QDCount**: specifica il numero di answers
- **ANCount**: specifica il numero di queries mandate

Campo utilizzato: DNS ID

- visto come una chiave di autenticazione
- usato per mandare un **messaggio segreto**
- Composto da **16 bit**

Client side

Metodologia

L'algoritmo sviluppato per poter nascondere messaggi in query DNS può essere visto come composto da due fasi ben distinte:

- **Start Session**
- **Data Exchange**

Metodologia

- Nella prima fase si vuole indicare al server DNS che si sta per trasmettere un messaggio segreto
- Mentre nella seconda fase c'è il vero invio del messaggio segreto nascosto in diverse query DNS

Passi

I passi dell'algoritmo sono i seguenti:

1. Cifratura del messaggio da inviare
2. Calcolo lunghezza del messaggio
3. Incapsulazione del messaggio di start session all'interno del campo TTL
4. Divisione del messaggio in chunks da massimo 16 bytes ciascuno
5. Per ciascun chunk invio di 16 query DNS in cui sono incapsulati i singoli caratteri del messaggio (16 bytes equivalgono a 16 caratteri) nel campo DNS ID

Passi

I passi dell'algoritmo sono i seguenti:

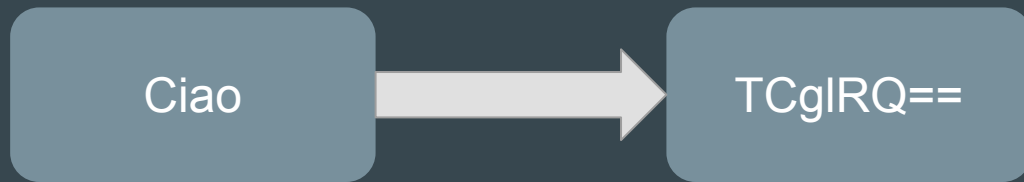
1. Cifratura del messaggio da inviare
2. Calcolo lunghezza del messaggio
3. Incapsulazione del messaggio di start session all'interno del campo TTL
4. Divisione del messaggio in chunks da massimo 16 bytes ciascuno
5. Per ciascun chunk invio di 16 query DNS in cui sono incapsulati i singoli caratteri del messaggio (16 bytes equivalgono a 16 caratteri) nel campo DNS ID

Cifratura del messaggio da inviare

- Il messaggio da inviare non è inviato in “*plain text*” ma è prima **cifrato** e poi inviato
- Ciò è fatto per migliorare ulteriormente la **sicurezza**
- Anche se un intercettatore riesce a capire il meccanismo di incapsulazione dei messaggi, non vedrebbe il messaggio in chiaro. Potrà decrittare il messaggio solo se avrà la **chiave di decifratura**

Cifratura del messaggio da inviare - XOR

- **XOR** è uno dei cifrari a flusso più semplici
- La crittografia viene eseguita facendo uno **XOR** tra la chiave segreta ed il messaggio
- Genera sempre lo stesso risultato (rappresentato in **base64**)



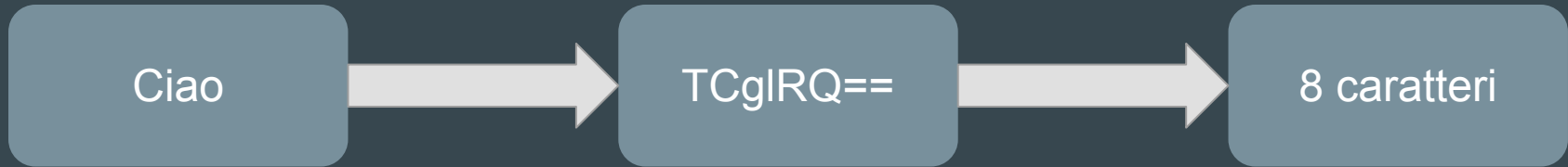
Passi

I passi dell'algoritmo sono i seguenti:

1. Cifratura del messaggio da inviare
2. **Calcolo lunghezza del messaggio**
3. Incapsulazione del messaggio di start session all'interno del campo TTL
4. Divisione del messaggio in chunks da massimo 16 bytes ciascuno
5. Per ciascun chunk invio di 16 query DNS in cui sono incapsulati i singoli caratteri del messaggio (16 bytes equivalgono a 16 caratteri) nel campo DNS ID

Calcolo lunghezza del messaggio

- Calcolo della lunghezza del messaggio cifrato da inviare



Passi

I passi dell'algoritmo sono i seguenti:

1. Cifratura del messaggio da inviare
2. Calcolo lunghezza del messaggio
3. Incapsulazione del messaggio di start session all'interno del campo TTL
4. Divisione del messaggio in chunks da massimo 16 bytes ciascuno
5. Per ciascun chunk invio di 16 query DNS in cui sono incapsulati i singoli caratteri del messaggio (16 bytes equivalgono a 16 caratteri) nel campo DNS ID

Start Session

- Incapsulazione del messaggio di start session all'interno del campo TTL
- Il campo TTL è composto da 16 bit e le informazioni sono state nascoste nel seguente modo:
 - nei **primi 4 bit pari** è stato inserito un **pattern**, che servirà al mio server DNS per controllare se prendere in considerazione la query
 - nei **primi 8 bit dispari** è stata inserita la lunghezza del messaggio espressa in binario, che servirà per indicare quante query DNS col campo DNS ID modificato aspettarsi
 - gli ultimi **4 bit pari**, invece, sono stati generati casualmente

Start Session

P L P L P L P L R L R L R L R L

16 bit

P = bit pattern (**primi 4 bit pari**)

L = bit lunghezza messaggio (**primi 8 bit dispari**)

R = bit random (**ultimi 4 bit pari**)

Start Session

1 0 0 0 1 0 1 1 0 1 1 0 0 0 1 0

16 bit

P = 1 0 1 1 -> pattern

L = 0 0 0 1 1 0 0 0 = 24 -> lunghezza

R = 0 1 0 1 -> random

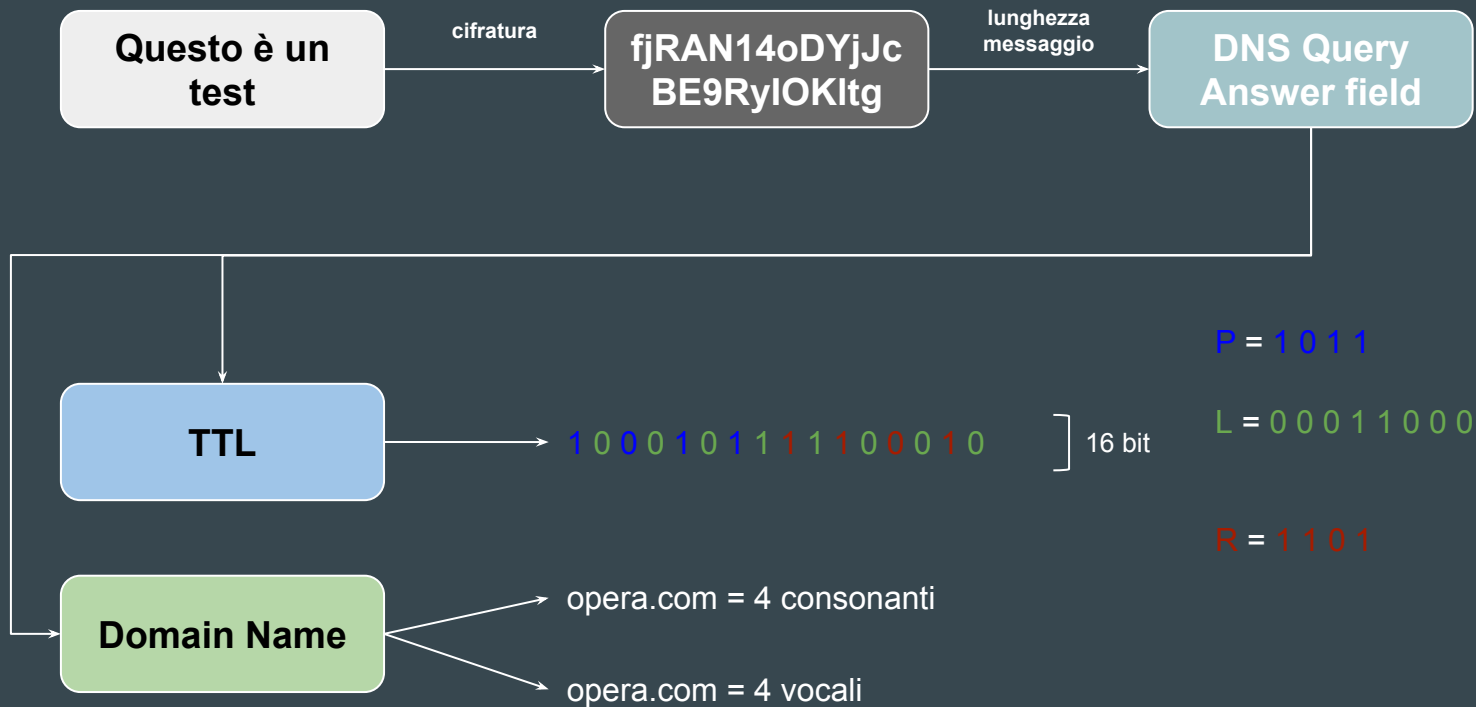
Start Session

- Inoltre, sono stati posti **ulteriori due vincoli** sul domain name della query in cui inserire il messaggio di start session:
 - **il domain name deve avere un numero di consonanti pari**
 - **il domain name deve contenere almeno 4 vocali**

Start Session



Start Session - Esempio



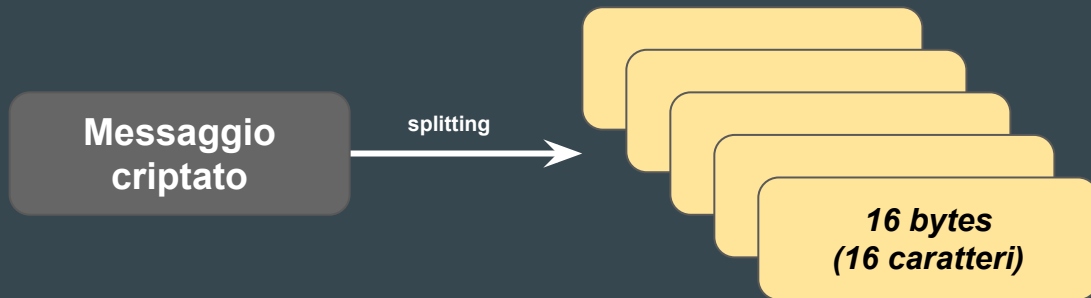
Passi

I passi dell'algoritmo sono i seguenti:

1. Cifratura del messaggio da inviare
2. Calcolo lunghezza del messaggio
3. Incapsulazione del messaggio di start session all'interno del campo TTL
4. Divisione del messaggio in chunks da massimo 16 bytes ciascuno
5. Per ciascun chunk invio di 16 query DNS in cui sono incapsulati i singoli caratteri del messaggio (16 bytes equivalgono a 16 caratteri) nel campo DNS ID

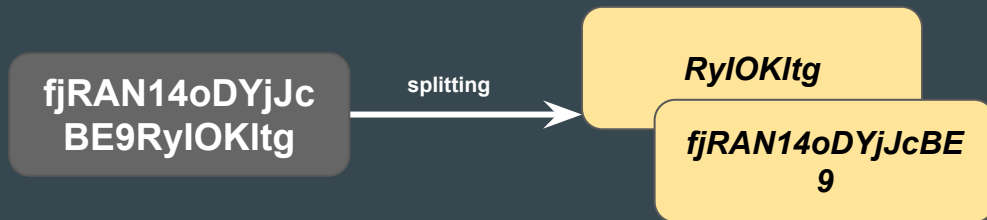
Divisione in chunks

- Il messaggio criptato è diviso in chunks, ognuno composto da 16 bytes (ovvero 16 caratteri) ciascuno, e per ciascun chunk sono effettuate 16 queries DNS



Divisione in chunks

- Il messaggio criptato è diviso in chunks, ognuno composto da 16 bytes (ovvero 16 caratteri) ciascuno, e per ciascun chunk sono effettuate 16 queries DNS



Passi

I passi dell'algoritmo sono i seguenti:

1. Cifratura del messaggio da inviare
2. Calcolo lunghezza del messaggio
3. Incapsulazione del messaggio di start session all'interno del campo TTL
4. Divisione del messaggio in chunks da massimo 16 bytes ciascuno
5. Per ciascun chunk invio di 16 query DNS in cui sono incapsulati i singoli caratteri del messaggio (16 bytes equivalgono a 16 caratteri) nel campo DNS ID

Data exchange

In ciascuna query è incapsulato un singolo carattere che compone il messaggio. L'incapsulazione avviene nel seguente modo:

- nei **primi 8 bit pari** è stato inserito il carattere ASCII corrispondente ed espresso in binario. Ad esempio la lettera '**a**' corrisponde a **97** ed espressa in binario corrisponde a '**01100001**'
- nei **primi 4 bit dispari** è stato inserito un valore di **sequencing**, ovvero un numero che identifica la posizione del carattere all'interno del chunk. Ciò è stato utile al server DNS per ricostruire il messaggio segreto inviato, poichè, essendo il DNS un protocollo UDP, non è detto che i pacchetti arrivino sequenzialmente

Data exchange

In ciascuna query è incapsulato un singolo carattere che compone il messaggio. L'incapsulazione avviene nel seguente modo:

- infine, negli **ultimi 4 bit dispari** è stato inserito il pattern, che serve per indicare al server DNS di prendere in considerazione la query. Da notare che il pattern è lo stesso di quello usato nello step precedente (fase di **start session**)

Data exchange

C S C S C S C S C P C P C P C P

16 bit

C = bit carattere (**primi 8 bit pari**)

S = bit sequencing (**primi 4 bit dispari**)

P = bit pattern (**ultimi 4 bit dispari**)

Data exchange

0 0 1 0 1 0 0 0 1 1 0 1 1 0 1

16 bit

C = 0 1 1 0 0 1 1 0 = 102 = f (ASCII)

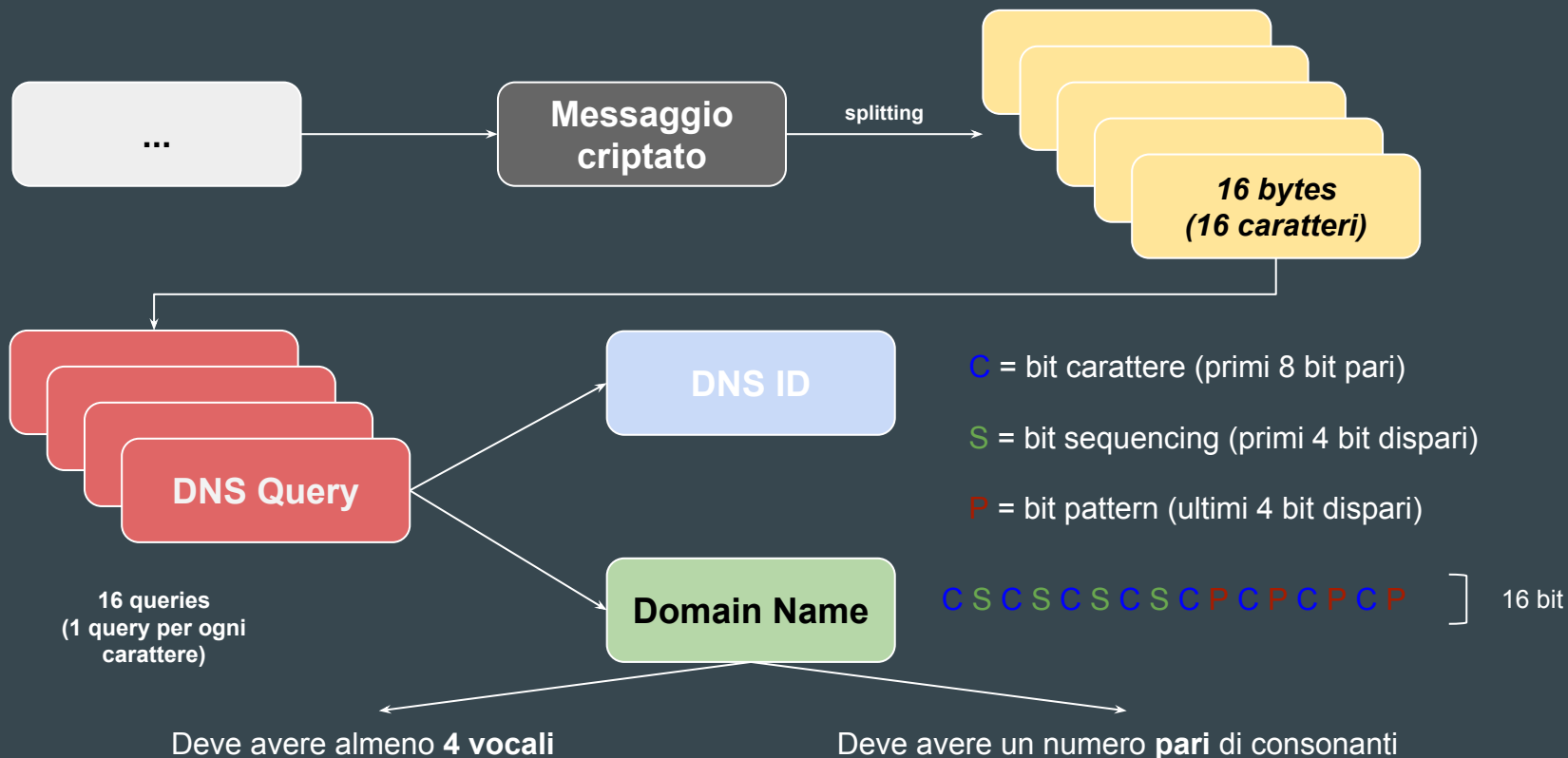
S = 0 0 0 0 = 0 (primo carattere del chunk)

P = 1 1 0 1

Data exchange

- Inoltre, sono stati posti **ulteriori due vincoli** sul domain name della query in cui inserire il messaggio di start session:
 - **il domain name deve avere un numero di consonanti pari**
 - **il domain name deve contenere almeno 4 vocali**

Data exchange



Server side

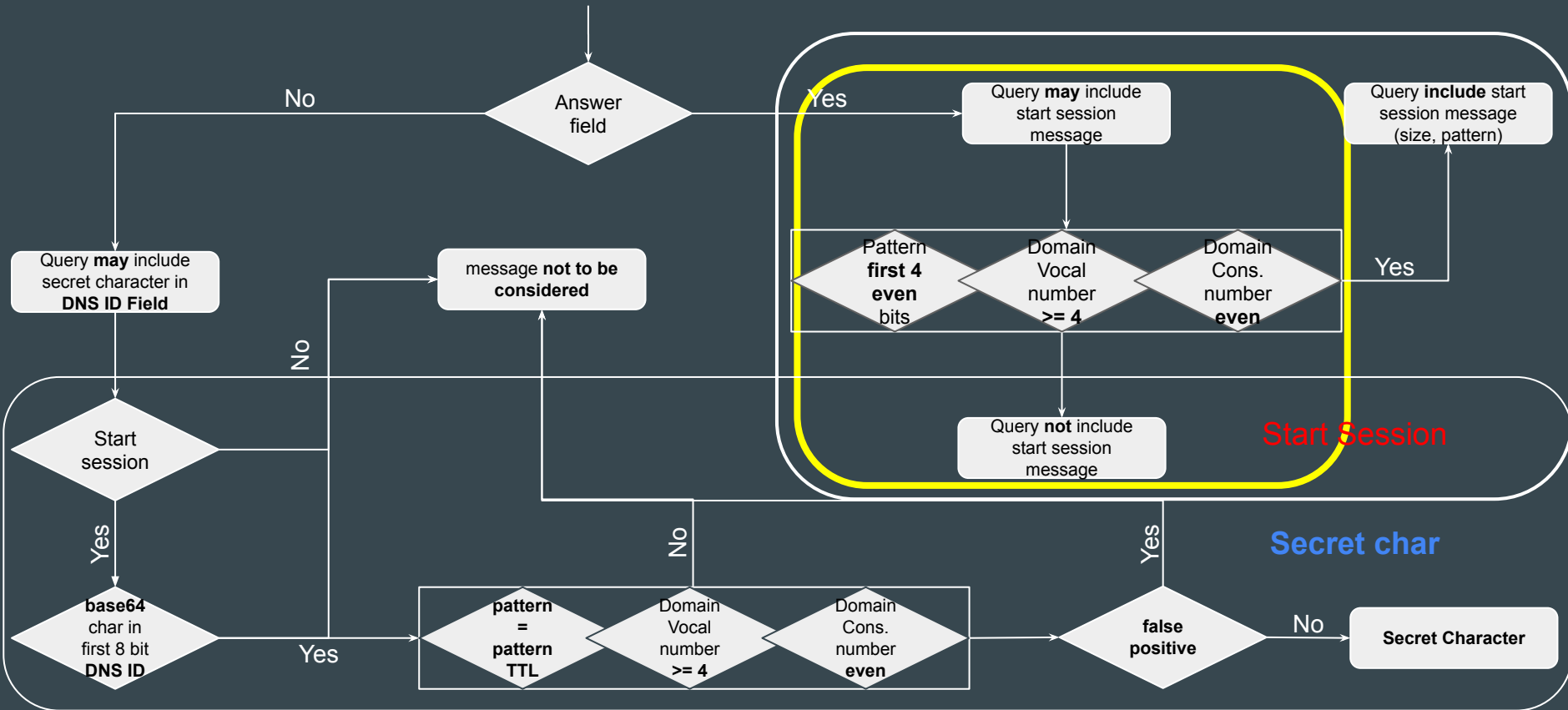
Server side

- Il server DNS è sempre in ascolto e ogni volta che riceve una query DNS effettua i seguenti controlli:
 - controlla se la query DNS contiene **anche** una risposta (**answer field**)
 - in caso affermativo, quella query potrebbe contenere il messaggio di start session
 - altrimenti, quella query potrebbe contenere un carattere segreto nascosto nel campo DNS ID

Start session - passo 1

- Nel caso una query DNS contenga anche una risposta, allora, si effettuano i seguenti passaggi:
 - lettura campo **TTL** per ottenere il pattern di identificazione
 - si converte il campo TTL, che è un numero, in binario e si estraggono i **primi 4 bit pari**
 - verifica che il **domain name** abbia **almeno 4 vocali** e un **numero pari di consonanti**

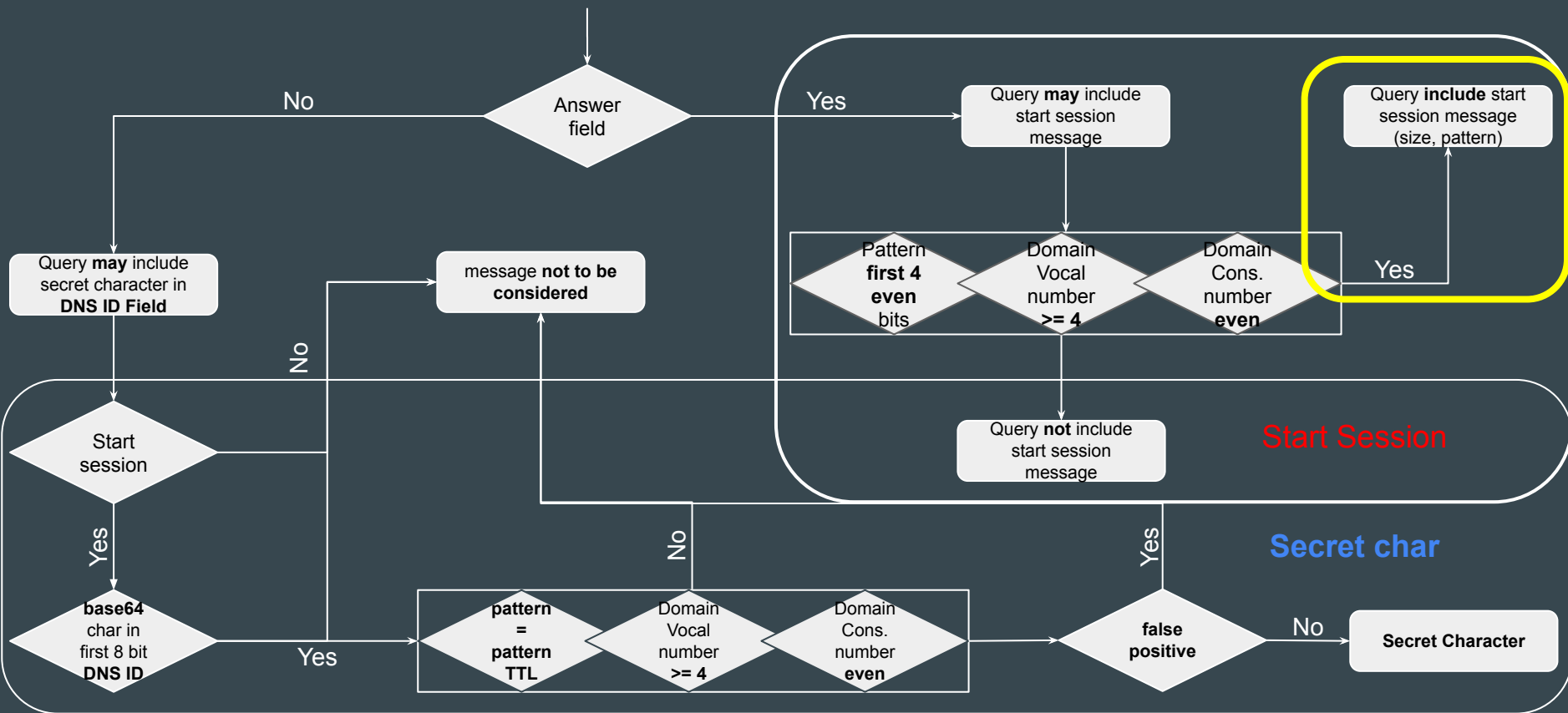
Start session - passo 1



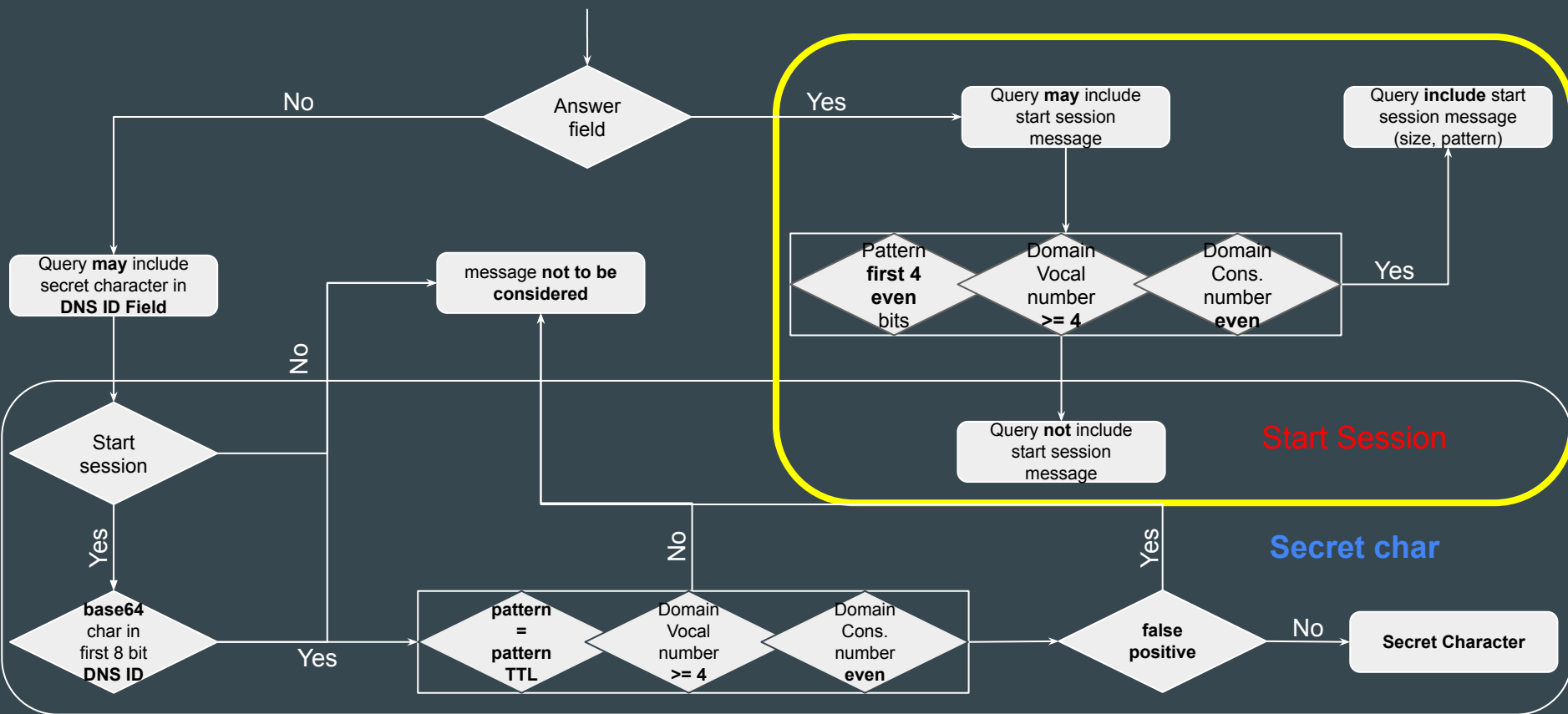
Start session - passo 2

- Nel caso siano verificate le **3 condizioni precedenti**, la query è presa in considerazione poiché sicuramente contiene il messaggio di start session e si estrae la lunghezza del messaggio, che identifica il numero di query DNS con un campo DNS ID modificato che il client invierà al server
 - Per ottenere la lunghezza si effettuano i seguenti passaggi:
 - estrazione dal campo TTL dei primi **8 bit dispari**
 - conversione da binario ad intero
- Invece, nel caso che anche solo una condizione non dovesse verificarsi, la query DNS non viene presa in considerazione perchè sicuramente non contiene alcuna informazione utile

Start session - passo 2



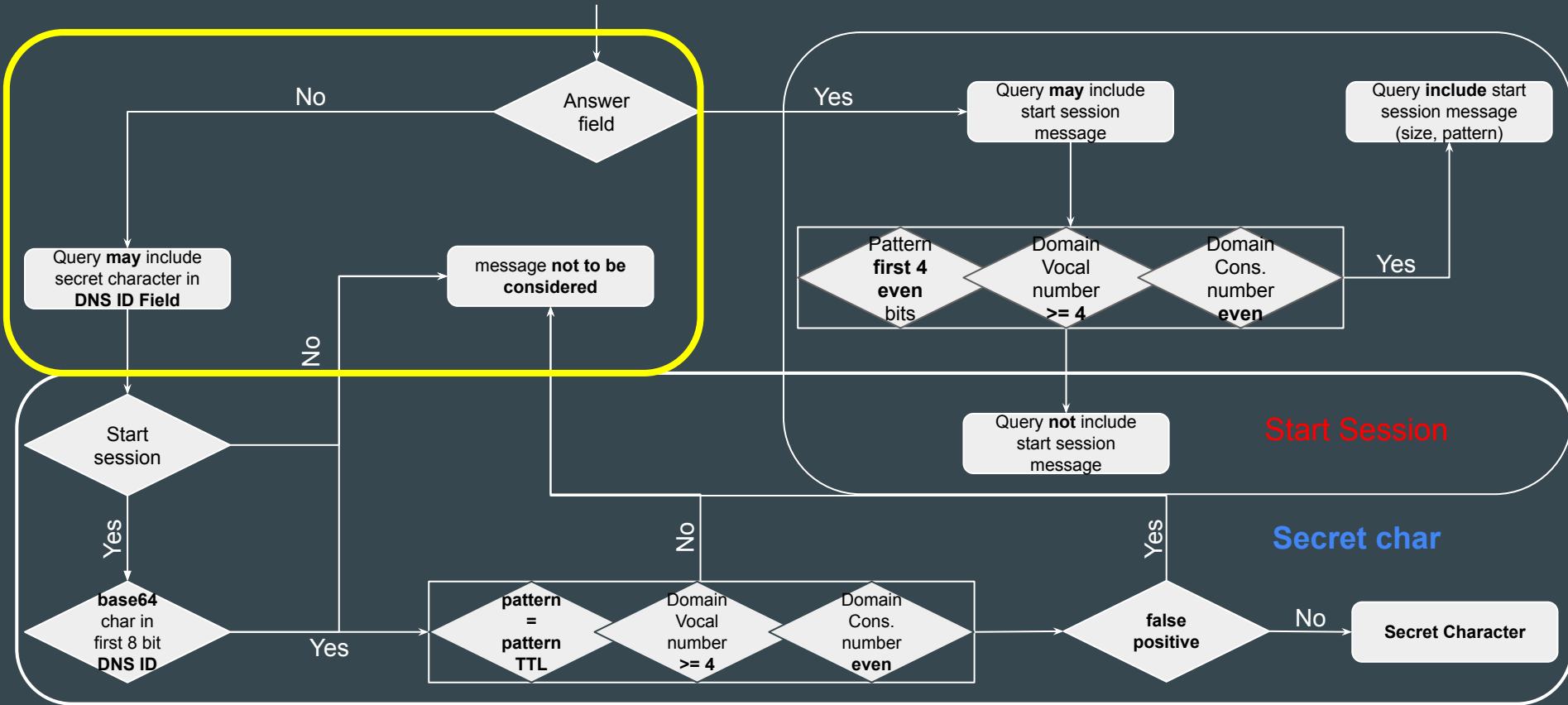
Start session



Data extraction - passo 1

- Invece, se la query DNS non contiene anche una risposta (**answer field**), allora, questa query potrebbe contenere un carattere segreto nascosto nel campo DNS ID
 - Il primo controllo da effettuare è quello di controllare se in precedenza è stata ricevuta una query contenente il messaggio di start session, poiché, se così non fosse, allora tutte le query devono essere ignorate, perchè sicuramente non contengono alcuna informazione nascosta
 - In caso affermativo, inizia la fase di estrazione del carattere dal campo DNS ID

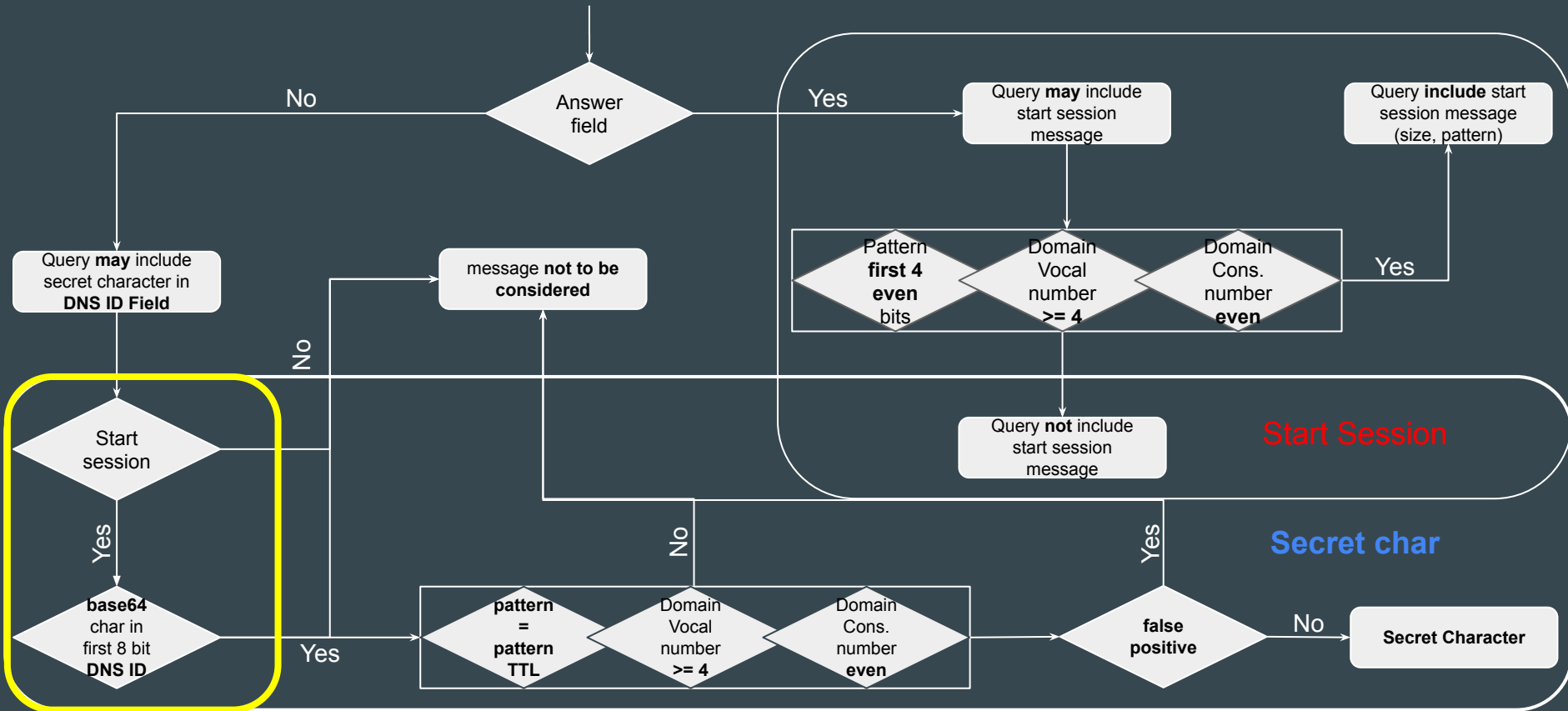
Data extraction - passo 1



Data extraction - passo 2

- La fase di data exchange è composta dai seguenti passi:
 - a. per prima cosa si estraggono i **primi 8 bit pari** dal campo **DNS ID**, ottenendo un numero binario, quest'ultimo viene convertito in intero che a sua volta viene convertito in carattere **ASCII**
 - Se il carattere appartiene all'insieme dei caratteri della codifica in **base64**, ovvero se è compreso tra **A-Z, a-z, 0-9**, oppure è **/** o **=**, allora la query viene presa in considerazione e si procede con la "*decodifica*"
 - altrimenti viene scartata perchè non può contenere alcuna informazione segreta

Data extraction - passo 2



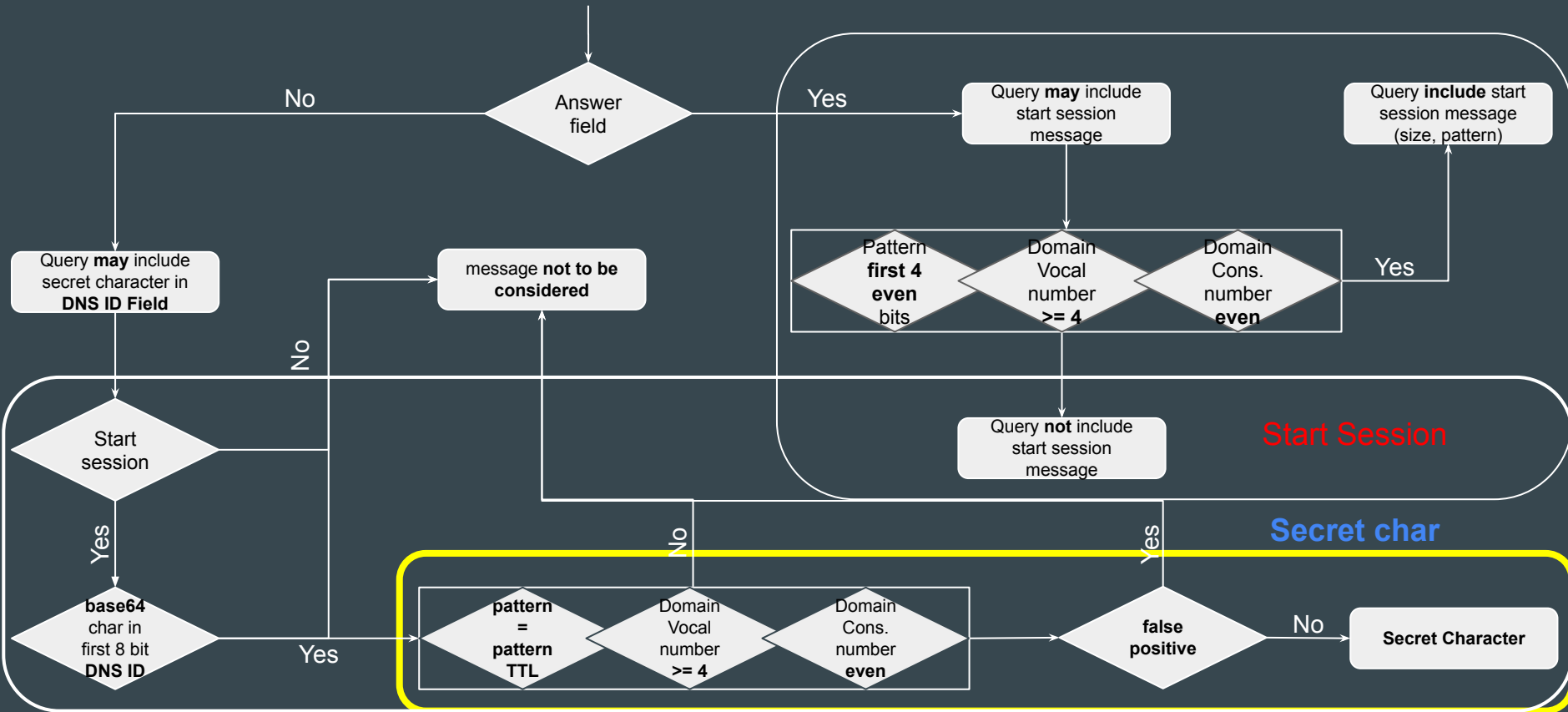
Data extraction - passo 3

- La fase di data exchange è composta dai seguenti passi:
 - b. si estraggono gli **ultimi 4 bit dispari**, sempre dal campo **DNS ID**, per ottenere il **pattern** di identificazione
 - c. si controlla il campo **domain name** che deve avere **almeno 4 vocali** e un **numero pari di consonanti**
 - Se il pattern è lo stesso di quello ottenuto nella fase di start session e sono verificate anche le condizioni sul domain name, allora si prosegue con la fase di decodifica
 - altrimenti la query non viene presa in considerazione

Data extraction - passo 4

- La fase di data exchange è composta dai seguenti passi:
 - d. infine, si va ad effettuare un ulteriore controllo per verificare se quella query è un **falso positivo**. Infatti, nonostante abbia messo un buon numero di vincoli, non è impossibile che ci possano essere delle query reali che rispettino i vincoli che ho imposto, generando, quindi, dei falsi positivi. Il controllo avviene nel seguente modo:
 - dato che da lato client non invio mai due query consecutive contenenti caratteri segreti, controllo se la query precedente è stata considerata
 - In caso positivo, allora, scarto la query corrente e proseguo
 - altrimenti prendo in considerazione la query corrente e vado ad estrarre il carattere segreto (contenuto nei primi 8 bit pari)

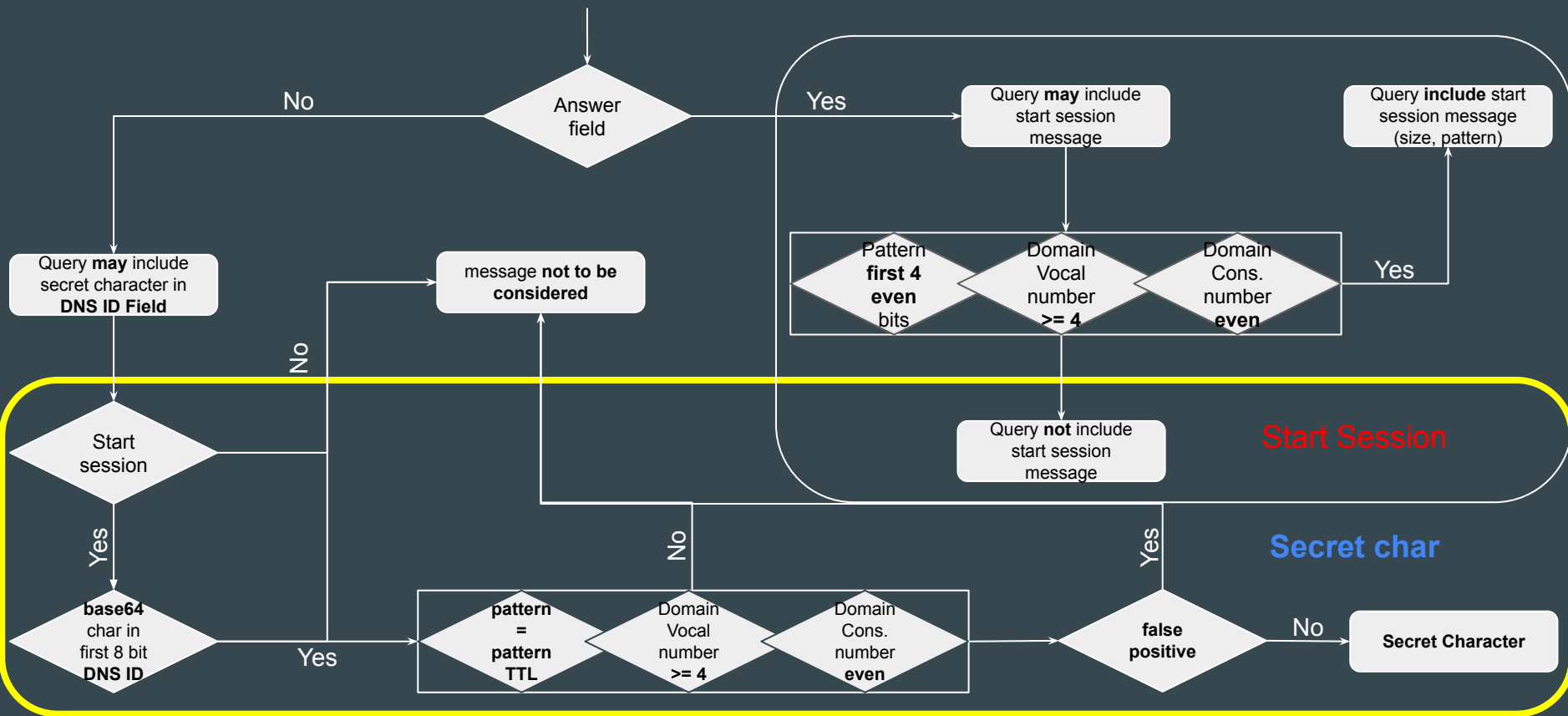
Data extraction - passo 3-4



Data extraction - passo 5

- La fase di data extraction avrà fine quando il numero di query DNS col campo DNS ID considerate avrà raggiunto la lunghezza del messaggio, ottenuta nella fase di start session
- Si costruisce il messaggio con tutti i caratteri contenuti nelle query DNS "malevole" e lo si decifra

Data extraction



Ambiente di sviluppo

Python

- Sia il **client** che il **server** sono stati scritti in **Python**, in particolare ho usato **Python 3.9**. Tale scelta è stata dovuta dal fatto che Python mette a disposizione delle librerie che permettono di intercettare e manipolare le queries DNS abbastanza facilmente
- Il server DNS è stato scritto da zero “*from scratch*” grazie all'utilizzo della libreria “**dnslib**”. Questo server DNS non fa altro che ascoltare le queries DNS che riceve da diversi client e rigirarle “*forward*” ad un vero server DNS, in questo caso ho usato un **Google DNS (8.8.8.8)**.
 - Una volta ricevuta la risposta dal server DNS reale, la inoltra al client che aveva mandato la query

DNS Server

```
from dnslib.proxy import ProxyResolver
from dnslib.server import DNSServer

if __name__ == '__main__':
    signal.signal(signal.SIGTERM, handle_sig)

    framestore = []
    port = int(os.getenv('PORT', 53))
    upstream = os.getenv('UPSTREAM', '8.8.8.8')
    resolver = Resolver(upstream)
    udp_server = DNSServer(resolver, port=port)
    tcp_server = DNSServer(resolver, port=port, tcp=True)

    udp_server.start_thread()
    tcp_server.start_thread()

    try:
        while udp_server.isAlive():
            sleep(1)
    except KeyboardInterrupt:
        pass
```

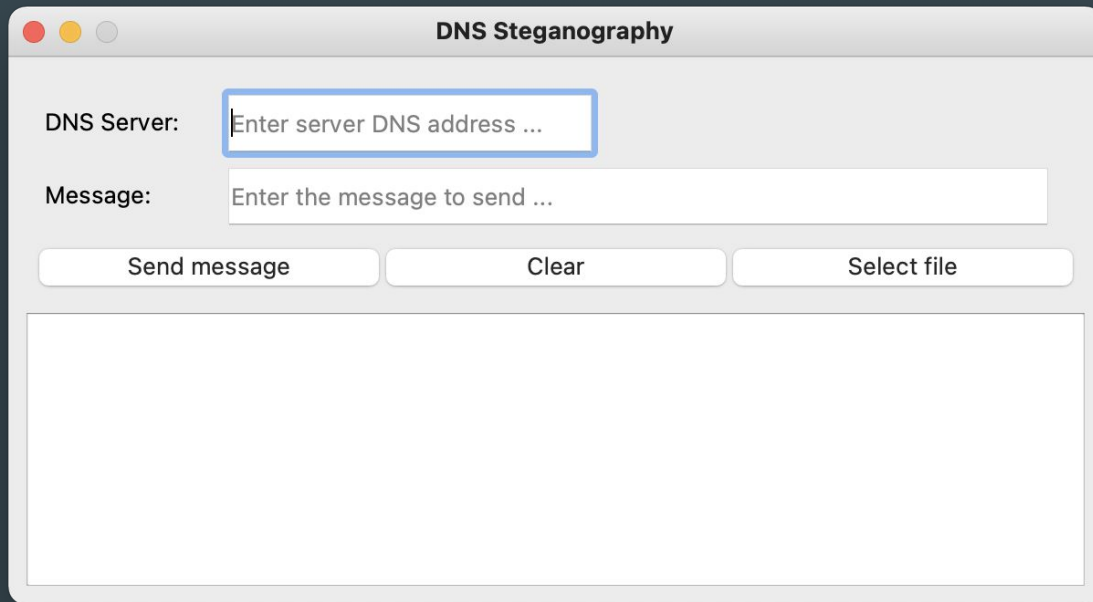
Librerie Python

- Invece, l'invio e la manipolazione delle queries DNS lato client è stato implementato grazie all'ausilio della libreria **scapy**
- Lato client ho scelto di utilizzare questa libreria e non "**dnslib**" poiché con quest'ultima mi è risultato più difficile manipolare le query DNS non essendoci una documentazione ufficiale in cui spiegano come fare, a differenza della libreria **scapy**, dove c'è molta documentazione su come creare query DNS e inviarle
- Dunque la libreria "**dnslib**" mi è stata utile per intercettare le queries e rigirarle ad un server DNS reale

Interfaccia grafica

- Lato client, è stata sviluppata anche un'interfaccia grafica per permettere una migliore esperienza all'utente e facilitare l'uso dello strumento
- L'interfaccia grafica è stata sviluppata tramite l'ausilio della libreria **PyQt5** e permette di scegliere il server DNS a cui mandare le queries e il messaggio da mandare, quest'ultimo può essere o scritto manualmente o selezionato da un file (.txt)

Interfaccia grafica



The image shows a graphical user interface for a tool named "DNS Steganography". The window has a title bar with three colored buttons (red, yellow, grey) on the left and the title "DNS Steganography" in the center. The main area contains two input fields: "DNS Server:" with a placeholder "Enter server DNS address ..." and "Message:" with a placeholder "Enter the message to send ...". Below these fields are three buttons: "Send message", "Clear", and "Select file". At the bottom of the window is a large, empty rectangular area.

DNS Steganography

DNS Server: Enter server DNS address ...

Message: Enter the message to send ...

Send message Clear Select file

Test e valutazioni

Operating System

- L'intero ambiente di sviluppo (sia client che server) è stato testato sui sistemi operativi **MacOS** e **Linux**, in particolare:
 - **MacOS Big Sur versione 11.3**
 - **Linux Lite 5.2 a 64 bit**

Test

- Tutti i test sono stati effettuati in rete locale e le queries DNS sono state monitorate tramite lo strumento **WireShark**
- I test sono stati effettuati sia con il client ed il server sulla stessa macchina (**localhost**) sia su **macchine differenti** (una ospitante il server e un'altra ospitante il client)
- I test sono stati effettuati per valutare l'overhead generato sulla rete per nascondere la comunicazione di messaggi segreti

Rumore

- Tra una query DNS contenente un'informazione segreta e un'altra sono inframezzate randomicamente delle queries DNS che non contengono alcuna informazione nascosta. Ciò è fatto solo per creare **rumore** e nascondere ancora ulteriormente le informazioni
 - Caso migliore
 - Caso medio
 - Caso peggiore

Overhead

Bytes	Caso Migliore	Caso Medio	Caso Peggior
8 Bytes	3700 Bytes	5700 Bytes	7000 Bytes
16 Bytes	6300 Bytes	10000 Bytes	12000 Bytes
32 Bytes	11200 Bytes	16300 Bytes	22000 Bytes
64 Bytes	21400 Bytes	31200 Bytes	42200 Bytes

- Nella seguente tabella è riportato l'overhead generato nella rete per poter trasmettere un messaggio segreto. Dalla tabella si può notare che per inviare un messaggio segreto di 8 bytes si è generato un traffico di 3700 bytes nel caso migliore, 5700 bytes nel caso medio e 7000 bytes nel caso peggiore
- più il messaggio da inviare è grande maggiore è l'overhead generato sulla rete
- per trasmettere un messaggio lungo 64 bytes, nel caso peggiore, in totale si è generato un traffico di 42200 bytes. Ciò, però, è un traffico insolito per queries DNS, per questo motivo si possono inviare messaggi segreti lunghi al massimo 64 caratteri (64 bytes)

Tempi

Bytes	Secondi
8 Bytes	4,5 secondi
16 Bytes	12 secondi
32 Bytes	23 secondi
64 Bytes	50 secondi

- i tempi di esecuzione dipendono dai tempi di risposta del server DNS di Google, però, in tabella sono riportati i tempi medi ottenuti da più esecuzioni del programma
- come potremmo aspettarci, maggiore è la lunghezza del messaggio da inviare, maggiore sono i tempi di esecuzione

Query DNS

Inoltre, è da specificare che il traffico intercettato da **WireShark** non desta alcun sospetto, infatti sembrano query reali. Un esempio di query generata dal programma ed intercettata da WireShark è la seguente:

```
> Frame 122: 57 bytes on wire (456 bits), 57 bytes captured (456 bits) on interface lo0, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> User Datagram Protocol, Src Port: 9819, Dst Port: 53
v Domain Name System (query)
  Transaction ID: 0xb9c3
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    v ebay.fr: type A, class IN
      Name: ebay.fr
      [Name Length: 7]
      [Label Count: 2]
      Type: A (Host Address) (1)
      Class: IN (0x0001)
```


Conclusioni

Conclusioni

- In questo progetto d'esame è stato presentato un nuovo meccanismo di "**Covert Channel over DNS**" che può essere tranquillamente usato per **nascondere informazioni segrete** da inviare ad un server DNS compromesso
- l'unico limite è quello di poter inviare messaggi di al massimo 64 bytes poichè, altrimenti, si genererebbero troppe queries DNS e ciò potrebbe destare dei sospetti e quindi essere sgamati.

Bibliografia

Bibliografia

- Michał Drzymala, Krzysztof Szczypiorski, and Marek Łukasz Urbański, “**Network Steganography in the DNS Protocol**”
- Abdulrahman H. Altalhi, Md Asri Ngadi, Syaril Nizam Omar and Zailani Mohamed Sidek, “**DNS ID Covert Channel based on Lower Bound Steganography for Normal DNS ID Distribution**”
- Christopher Hoffman, Daryl Johnson, Bo Yuan, Peter Lutz, “**A Covert Channel in TTL Field of DNS Packets**”
- Wojciech Mazurczyk, Steffen Wendzel, Ignacio Azagra Villares and Krzysztof Szczypiorski, “**On importance of steganographic cost for network steganography**”
- S.N Omar, I.Ahmedy, M.A Ngadi, “**Indirect DNS Covert Channel based on Name Reference for Minima Length Distribution**”

Domande?