Traccia 1:

Descrizione del codice e della configurazione:

Il codice del progetto è realizzato tramite il linguaggio di programmazione C, il codice può essere diviso in tre sezioni a livello concettuale:  
- file di configurazione Docker : Dockerfile, ControllerDockerfile e docker-compose.yml

- file del programma C&C denominato controller per semplicità: controller.c utils.c common.h controller.h utils.h

- file programma bot: bot.c bot.h

**Comunicazione**

Controller e bot comunicano tramite scambio di messaggi http, l’ip del controller è statico e deciso a priori.  
  
Nel progetto ho deciso di usare la tecnologia di containerizzazione fornita da Docker, questa porta vari vantaggi tra cui la possibiltà di poter far girare instanze multiple del programma bot ognuna con un suo indirizzo IP univoco.  
  
A livello implementativo, viene usato un file docker-compose.yml per gestire vari aspetti di configurazione e comunicazione dei due programmi, ognuno dei quali ha la sua immagine Docker specificata tramite file Dockerfile e ControllerDockerfile rispettivamente per bot e controller.

Nel file docker-compose.yml viene definita una custom network bridge: “frontend” a cui appartengono bot e controller, alla rete frontend viene assegnata una subnet: 172.20.0.0/24 così da poter assegnare indirizzi ip univoci a tutti I bot e al controller, al quale però viene impostato un indirizzo ip statico data la sua natura di server.   
L’ip del bot viene assegnato automaticamente dal daemon docker rispettando il vincolo di appartenenza alla subnet prima citata, la porta di connessione del bot invece è randomica, viene selezionata una porta libera del sistema dalla quale il bot può ricevere comandi dal C&C, ciò viene fatto tramite la funzione definita .

Per permettere la comunicazione tramite http si fa uso della libreria : “libcurl” , lato client per l’invio di richieste http e della libreria “microhttpd” lato server per ricevere richieste http ed inviare risposte. Sia bot che controller usano entrambe queste librerie allo stesso tempo per scopi differenti:

controller usa libcurl per inviare comandi ai suoi bot e microhttpd per ricevere notifiche sull’esecuzione dei comandi inviati. Bot invece usa libcurl per fare la prima connessione al controller e quindi essere registrato nella botnet e microhttpd per rimanere costantemente in modalità di ricezione di comandi dal controller, una volta ricevuto un comando e portato a termine bot invia una notifica al controller tramite libcurl per segnalare l’avvenuto eseguimento del comando.