RELAZIONE ESERCIZIO SULLE BACKDOOR

Consegna U1-S3-L4

In primis il programma importa i tre moduli: *socket*, *platform* e *os*.

Il modulo *socket* in Python contiene al suo interno la funzione *socket()* che se invocata ritorna un oggetto di tipo *socket* i cui metodi implementano le varie chiamate di sistema. I tipi di parametro in Python sono di più alto livello rispetto al linguaggio C.

Il secondo modulo importato dal programma è *platform*, che permette di leggere le caratteristiche hardware e la versione di python. Ad esempio la funzione *platform.platform ()* ritorna una stringa che aiuta a identificare il sistema nella maniera più dettagliata possibile. Inoltre il suo output sarà di tipo human readable.

Il metodo *platform.machine()* invece ritorna il tipo di architettura della macchina.

Il modulo *os* permette di utilizzare funzionalità dipendenti dal sistema operativo come ad esempio la lettura di un file con la funzione *open()* o di gestire i path con la il modulo *os.path*.

Dopo aver creato due variabili per memorizzare l'indirizzo IP e la porta di rete dove mettere in ascolto il programma, si istanzia un oggetto di tipo *socket* e gli si specifica tramite i parametri che ci interessa esaminare una connessione di tipo IPv4 tramite il protocollo TCP/IP. I parametri sono rispettivamente *AF_INET* e *SOCK_STREAM*. Se avessimo voluto esaminare una connessione UDP avremmo potuto usare il parametro *SOCK_DGRAM*.

Poi si effettua il binding del server con il metodo *bind()* per indicare al server su quale porta e IP restare in ascolto e gli si passano come parametri le due variabili che contengono tali valori: *SRV_ADDR* e *SRV_PORT*.

Tramite il metodo *listen()* si avvia l'ascolto della connessione da parte del server e passando *1* come argomento si stabilisce un tetto massimo di una connessione per volta.

Il metodo *accept()* ritorna due valori: *connection* identifica la comunicazione tra server e client, mentre *address* conterrà l'IP del client. Tale metodo serve per accettare la connessione del client.

Dopo aver stampato a schermo l'avviso per l'avvenuta connessione, si utilizza un ciclo *while* per ripetere una serie di istruzioni. *try* ed *except* servono per gestire le eccezioni durante l'esecuzione del codice. Quando si verifica un'eccezione all'interno

del *try* si va a gestire tramite le istruzioni contenute in *except*. Se *except* non contiene nulla va inserito nella parte finale del codice e gestirà ogni tipo di eccezione che si presenta.

Tramite il metodo *connection.recv()* si iniziano a ricevere dati dal client e viene impostata l'ampiezza del buffer a 1024 byte. I dati verranno salvati dentro "data". *except* si premurerà di far eseguire le istruzioni successive in caso di eccezione lanciata dal programma.

Viene implementato un controllo tramite *if - elif* su "data".

La funzione *decode()* tramite il parametro "utf-8" serve per tradurre ciò che arriva dal client codificato in sequenza di byte, in stringa, secondo la codifica UTF-8.

Se data.decode("utf-8") risulta 1, viene salvato l'output dei metodi *platform.platform()* e *platform.machine()* già discussi in precedenza dentro *tosend* aggiungendo un piccolo spazio a separazione dei due.

Tramite il metodo *sendall*() viene inviato al client connesso al server il contenuto di *tosend* che a sua volta viene codificato dal metodo *encode()* da stringa con codifica UTF-8 a sequenza di byte.

Se *data.decode()* dovesse risultare *2* viene salvato dentro *data* il messaggio ricevuto dal server tramite il metodo *recv()*.

Con il *try* gestisce le eccezioni del codice. Il metodo *listdir()* ritorna una lista contenente i nomi delle directory date da *path* in ordine arbitrario e non include le cartelle speciali "." ".." anche se presenti. *path* può essere di tipo oggetto *path* o se fosse di tipo *bytes* i nomi dei file in ritorno sarebbero di tipo *bytes*. Altrimenti sarebbero di tipo stringa. In questo caso, dentro *filelist* viene salvato il ritorno del metodo *listdir()* al quale viene passato come argomento il contenuto di *data* convertito da sequenza di byte a stringa.

Viene creata la variabile *tosend* inizializzata come stringa vuota. Tramite *ciclo for,* viene salvato di volta in volta all'interno di *tosend* il contenuto di *filelist* concatenando le varie stringhe e separandole con virgola. Se venisse lanciata un eccezione dal codice in *try*, *except* la gestirebbe salvando la stringa "Wrong path" all'interno di *tosend*. Infine all'interno dell'*elif* verrebbe eseguito il metodo *sendall()* per inviare all'altra macchina il contenuto di *tosend* codificandolo da stringa a sequenza di byte.

All'interno dell'ultimo *elif*, che viene eseguito solo se il contenuto di *data* ricodificato in stringa da sequenza di byte è uguale a *0,* è invocato il metodo *close()* che si

premurerà di interrompere la connessione tra le due macchine e tramite il metodo *accept()* riprenderà ad accettare una nuova connessione da un’altra macchina.