Immagine che contiene testo, schermata, software, schermo

Descrizione generata automaticamente

Questo è il codice per eseguire **l’udp flood**.

Si noti come **socket.socket** crea un nuovo socket (nel nostro caso chiamato s) inserendo dei parametri; il primo AF\_INET specifica che vogliamo un socket che utilizzi IPv4, il secondo SOCK\_DGRAM specifica che vogliamo una connessione *UDP*.

Il pacchetto da 1024 bytes (1 KB) viene generato randomicamente; l’utente inserirà in input l’indirizzo IP target, la porta target e il numero di volte che si vuole inviare il pacchetto. Tante volte quante il numero specificato (**try** con un ciclo **for**) tale pacchetto verrà inviato all’IP e porta specificati con **sendto** applicato sull’oggetto socket s creato precedentemente, stampando dei messaggi che indicano il successo nell’invio di un pacchetto volta per volta e il numero totale di pacchetti inviati alla fine.

In caso di errori l’**except** stamperà un messaggio di errore; infine (**finally**) il socket verrà chiuso.

A seguire il risultato dell’esecuzione del codice su macchina virtuale Kali Linux; utilizziamo come indirizzo IP target quello dell’host locale (**local host, 127.0.0.1**) inviando i pacchetti a noi stessi per verificare il corretto funzionamento del programma.

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

Immagine che contiene testo, elettronica, schermata, software

Descrizione generata automaticamente

Infine avviamo **wireshark** sull’interfaccia di **loopback** (local host) per analizzare il flusso di pacchetti inviati. Ci sono 5 pacchetti *UDP* (come inserito in input nel programma python), a ognuno dei quali corrisponde l’invio di un messaggio ICMP di 590 bytes poiché stiamo inviando dei pacchetti verso una porta di destinazione dove non c'è nessun applicativo in ascolto; in particolare *l’Internet Control Message Protocol (ICMP)* è un [protocollo](https://it.wikipedia.org/wiki/Protocollo_di_rete) di servizio per [reti](https://it.wikipedia.org/wiki/Rete_informatica) a [pacchetto](https://it.wikipedia.org/wiki/Commutazione_di_pacchetto) che si occupa di trasmettere informazioni riguardanti malfunzionamenti, informazioni di controllo o messaggi tra i vari componenti di una [rete di calcolatori](https://it.wikipedia.org/wiki/Rete_di_calcolatori)). Il contenuto (**payload**) del pacchetto è effettivamente 1024 bytes, ma vediamo che la dimensione totale di ogni pacchetto è di 1066 bytes perché vengono aggiunti i vari **headers**. Osserviamo che wireshark ci indica l’IP sorgente e l’IP destinazione (in questo caso gli stessi, ossia l’indirizzo del local host 127.0.0.1), il tipo di protocollo, la porta di arrivo (5678, come inserito in input nel programma python) e la porta di invio (57280, automaticamente assegnata dal sistema) insieme alla dimensione totale e del payload per ogni pacchetto inviato. In basso è possibile trovare maggiori dettagli sui singoli pacchetti, analizzando varie informazioni sui diversi livelli del protocollo *ISO/OSI* incapsulati nel pacchetto (per esempio si possono leggere le dimensioni degli headers dei protocolli *IP* di retee *UDP* di trasporto).

Qualche considerazione finale sul codice. Il pacchetto viene inviato un numero limitato di volte e solo su una specifica porta per scopi didattici, mentre se si volessero davvero causare problemi con il **flood** potremmo inserire un ciclo **while** sempre vero che continua l’invio dei pacchetti all’infinito finché non fermato, inviando inoltre i pacchetti contemporaneamente ad un numero maggiore di porte sovraccaricando enormemente la macchina target.