UNIVERSITÀ DI PISA

GESTIONE DI RETE

Marzo 2015

Rilevazione di uno o più attacchi
dato un file di flussi

Ambra Buscemi
Corso B
Matricola 475947

Funzionalità del programma

Il codice realizzato si propone di risolvere il problema della rilevazione di uno o più attacchi dati “n” file di flussi in input.
Nello specifico, per ogni file in input, si vuole mostrare un report di ogni secondo in cui si è rilevato un attacco; il report sarà costituito dai seguenti campi:

- l' IP che ha subito l'attacco;
- il numero di porta;
- il protocollo;
- il numero di flussi;
- il numero di pacchetti;
- il numero di byte;
- e il “secondo” in cui si è verificato l'attacco.

Ogni report verrà mostrato sia su shell, che sul file Report_Attacks_File.txt.
La scelta di salvare i risultati anche su un file, è stata fatta nell'ipotesi di facilitare la consultazione dei risultati in un secondo momento, e di raggruppare ulteriori risultati dentro un unico file; dato che per l'appunto i risultati delle successive esecuzioni del programma, su ulteriori file di flussi in input, verranno registrati (in append) sullo stesso file di output.
Un attacco non viene determinato in base solamente all'IP dell"host colpito, ma si va più nel dettaglio distinguendo anche attacchi diversi sullo stesso host in base alla tripla <IP, numero di porta e protocollo>; inoltre si è impostata la soglia minima dei numeri di flussi che determinano un attacco a 32 mila (questa soglia viene determinata nel file infoFlow.java dalla variabile min_limit_flows_attack).

Tuttavia il codice proposto detiene le seguenti limitazioni sui file di flussi dati in input:

1. i dati dei flussi devono avere esattamente il seguente formato

   IPV4_SRC_ADDR|IPV4_DST_ADDR|IPV4_NEXT_HOP|INPUT_SNMP|OUTPUT_SNMP|
   IN_PKTS|IN_BYTES|FIRST_SWITCHED|LAST_SWITCHED|L4_SRC_PORT|L4_DST_PORT|
   TCP_FLAGS|PROTOCOL|SRC_TOS|SRC_AS|DST_AS|IPV4_SRC_MASK|IPV4_DST_MASK

2. un flusso deve essere descritto in una sola riga;
3. un flusso non può avere un “campo” del suo formato vuoto;
4. la prima riga del file non deve contenere i dati di un flusso, ma preferibilmente la sua struttura (quella descritta nel punto 1)
5. tutte le linee successive del file devono contenere dei flussi, al più possono essere vuote.

Un file di input ben formattato deve avere le limitazioni richieste sopra.

Struttura del progetto

Per la realizzazione del tool si è deciso di utilizzare il linguaggio java.
Di seguito viene elencata la struttura del modulo java:

Main_Detects_Attacks.java: è il fulcro di tutto il modulo, contiene gli algortimi per il parsing dei file di flussi, della successiva analisi e della realizzazione dei report. Utilizza le classi contenute nei due file elencati sotto.

InfoKeyFlow.java: il file contiene la classe InfoKeyFlow, con il relativo costruttore, i metodi per leggere le relative variabili d'istanza e l'override di 2 metodi “hashCode” ed equals.

InfoFlow.java: il file contiene la classe InfoFlow, con il relativo costruttore, e i metodi per leggere le relative variabili d'istanza e manipolare alcune di esse.

Logica del progetto

La logica del modulo java è la seguente:

- viene aperto e/o generato il file di output Report_Attacks_File.txt;
- per ogni file passato da linea di comando:
  - viene aperto il file di flusso;
  - viene generata una tabella hash che ha come chiavi oggetti di tipo “InfoKeyFlow” e come valori oggetti di tipo “InfoFlow”;
  - viene fatto il parsing del file linea per linea;
    - per ogni linea si analizzano e prelevano i valori significativi per la determinazione di un attacco;
    - con i suddetti valori si creano 2 oggetti, uno di tipo “InfoKeyFlow” e l'altro di tipo “InfoFlow”, questi vengono inseriti come chiave e valore nella tabella hash; tuttavia se la chiave è già presente nella tabella l'unica operazione che verrà effettuata è quella di applicare il medoto “addOneFlow” all'oggetto di tipo InfloFlow;
    - se durante l'analisi del file il campo “LAST_SWITCHED” di una linea determina il passaggio di un secondo, si genera un report sia su shell che sul file di output e si ripulisce la tabella hash;
  - alla fine del file si genera il report “dell'ultimo secondo” sia su shell che sul file di output e si ripulisce la tabella hash;
  - se non sono stati rilevati attacchi nel file, viene riportato questo dato sia su shell che sul file di output
  - infine viene chiuso il file di flusso;
- e viene chiuso il file di output.

NOTA: il report non genera nulla se non ha rilevato un attacco.

File Di Test

Il progetto è stato generato con eclipse, per il test si consiglia di inserire nella “run configuration” i seguenti file di input:

1-Attacco_Base.flows
2-Attacco_1.flows
3-Attacco_2.flows
4-Attacco_3.flows
5-Attacco_4.flows

Di seguito viene elencato cosa vogliono mettere in luce i suddetti file di input:

1- Attacco_Base.flows:
attacco fornito dal prof. Luca Deri
(i successivi sono una manipolazione di questo file)
2- Attacco_1.flows:
file senza attacco
3- Attacco_2.flows:
file con 2 attacchi nello stesso secondo
4- Attacco_3.flows:
file con 3 attacchi, uno al tempo t, l'altro al tempo t+1 sec e l'altro al tempo t+4 sec
5- Attacco_4.flows:
file con 2 attacchi stesso IP ma porte diverse