IdentifyFlow

Edoardo Maione
Matr: **489532**
edoardo.maione@gmail.com
Corso di **Gestione di Reti**
Dipartimento di Informatica
Università di Pisa

Introduzione: Flusso(di pacchetti nelle Reti di Calcolatori)

Un flusso, nelle reti di calcolatori, non è altro che una sequenza di pacchetti che hanno le stesse caratteristiche: stesso mittente, stesso destinatario, stesso contenuto/protocollo a livello applicativo, che utilizzano le stesse risorse. Il concetto di flusso è legato al concetto di “protocollo connection oriented”, quindi usato per stabilire una connessione continua, generalmente usata per trasportare la stessa tipologia di dati, o per lo meno uno stream congruo di informazioni.

Identify Flow

Questo componente, per l’appunto, non è un applicazione completa, ma è un modulo scritto in Python, che offre la funzionalità di identificare i flussi di pacchetti, in base alle informazioni raccolte dagli header degli stessi pacchetti. Questa funzione è resa possibile grazie alla generazione di una chiave univoca per ogni flusso, creata sulla base delle informazioni “chiave” degli header dei vari livelli(Link-Rete-Trasporto) per identificare un flusso:

Header livello Link:
→ Protocollo di livello Rete: IPv4/IPv6/ICMP/...
Header livello Rete(se IPv4 o IPv6):
→ Indirizzo di Sorgente e Destinatario
→ Protocollo di livello Trasporto: TCP/UDP
oppure
(se IPv6) → Etichetta di Flusso
Header livello Trasporto(se TCP o UDP):
→ Porta Sorgente e Porta destinazione.

Qualora non fossero presenti queste informazioni, e il pacchetto utilizzasse protocolli diversi, la chiave viene composta da una stringa che fa in recap del contenuto del pacchetto(Protocolli usati, tipo di messaggi che trasportano, …).

Main Point: FlowCollector

FlowCollector è la classe che implementa tutte le funzionalità richieste: creazione degli identificatori di flussi da un pacchetto, identificazione dei pacchetti nei relativi flussi di appartenenza, e varie query sulla struttura(Cercare un flusso in base a: pacchetto, indirizzo, protocollo, grandezza, …).
La classe è implementata con un dizionario al suo interno, <key, val>, dove key è una tupla(oggetto immutabile, quindi valido allo scopo di chiave), strutturata come descritto in precedenza, e val è il numero dei pacchetti che compongono il flusso fino a quel momento.
Ho scelto di implementare le chiavi del dizionario come tuple, in modo tale da semplificare la ricerca di flussi in base ai capi della chiave, come per esempio indirizzo del mittente/destinatario o protocolli usati.

Metodi

→ Costruttore (**__init__**(verbose))
Semplice costruttore dove specificare il modo di operare dell’istanza della classe, con verbose=True/False.

→ **getFlowKey**(pkt)
Ricava la chiave(tupla) con il meccanismo sopra descritto per identificare univocamente il flusso.
Esempio di chiave:
 # (2048, '131.114.33.20', '10.101.16.182', 17, 53, 45292)

→ **add**(pkt)
La funzione principale è quella di aggiungere il pacchetto passato, ad uno dei flussi già esistenti e ritornando la stringa vuota “”, se presente; altrimenti ne crea uno nuovo e restituendo la stringa così formattata:

 # New Flow: +key+
 # First Packet -> +pkt.summary()+

Il comportamento di questo metodo cambia se il flag del costruttore, verbose, è stato settato a True: per ogni pacchetto aggiunto alla struttura, e quindi collocato nel proprio flusso, viene restituita la stringa che contiene il riassunto del pacchetto(src,dst, protocolli usati, …).
Riporto di seguito la formattazione:

 # "Flow: +key+ : new packet -> +pkt.summary()+"

→ **getFlows**(key || pkt || ipv4 || ipv6 || proto)
Ricerca uno o più flussi all’interno della struttura, dato UNO degli argomenti a scelta, restituendo una stringa contenente i flussi trovati. Il metodo è implementato con un if-else-if a cascata, nel quale viene usato solo un argomento, in ordine:

key → pkt → IPv4 → IPv6 → Proto

Per “key” e “pkt”, utilizza semplicemente una ricerca, con chiave data, nel dizionario interno.
Per “IPv4”, “IPv6”, e “Proto”, ricerca l’indirizzo/protocollo dato, all’interno dei campi delle keys(tuple) del dizionario, aggiungendo alla stringa restituita, la chiave del flusso trovato.

→ **biggestFlow**( )
Restituisce una stringa con la chiave del flusso più grande, e il numero dei pacchetti che lo compongono. La stringa è così formattata:
 # "Biggest Flow: +key+ whith -> +max+ Packets"

→ **show**( )
Stampa a schermo tutti i flussi identificati, con la seguente formattazione:
*# ***Flow Key: +key+*
 # Packets count: +num+
*# ***Flow Key: +key+*
 # Packets count: +num+
 # ...

Installazione e utilizzo

Per poter utilizzare il componente, è necessario avere installato, oltre che Python2.7 , anche il package python “scapy”, il quale contiene le funzioni utilizzate per catturare i pacchetti, ispezionarli e leggere il contenuto degli header. Per download e info su scapy, seguire questo link:
(https://thepacketgeek.com/tag/scapy/), dove viene riportato tutto ciò da sapere a riguardo.
In fine, copiare la cartella del package FlowTools all’interno del proprio progetto, dichiarare i vari import del caso:

"import scapy"
"from scapy.all import * "
"from FlowTools.flowCollector import FlowCollector"

ed eseguire con i diritti “privilegiati” da SuperUser/Amministratore.

Test

Per testare questo componente, eseguire il file “***prova.py***” (con l’opzione -v se si vuole testare il comportamento verbose), il quale effettua la cattura di 100 pacchetti, li inserisce nella struttura del componente, ed effettua vari test su tutti i metodi della classe FlowCollector, prendendo in base random campi adeguati(Indirizzi IPv4 e IPv6, key, pacchetti, ecc ...) dei pacchetti della cattura, come argomenti da passare ai metodi.

Esempio di Esecuzione del test *“prova.py -v”*:

*Nota: per motivi di spazio ho tagliato alcune righe di output

```
<Sniffed: TCP:82 UDP:14 ICMP:0 Other:4>
/------------------------/
New Flow: ('Ether / IPv6 / ICMPv6ND_NS / ICMPv6 Neighbor Discovery Option - Source Link-Layer Address 9c:97:26:24:09:26',)
     First Packet -> Ether / IPv6 / ICMPv6ND_NS / ICMPv6 Neighbor Discovery Option - Source Link-Layer Address 9c:97:26:24:09:26
New Flow: ('Ether / IPv6 / ICMPv6 Neighbor Discovery - Neighbor Advertisement (tgt: fe80::a62b:982c:f3b3:2ec1)',)
     First Packet -> Ether / IPv6 / ICMPv6 Neighbor Discovery - Neighbor Advertisement (tgt: fe80::a62b:982c:f3b3:2ec1)
New Flow: (2048, '54.230.202.199', '192.168.1.82', 6, 443, 41162)
     First Packet -> Ether / IP / TCP 192.168.1.82:41162 > 54.230.202.199:https A
New Flow: (2048, '192.168.1.82', '54.230.202.199', 6, 41162, 443)
     First Packet -> Ether / IP / TCP 54.230.202.199:https > 192.168.1.82:41162 A      // qui si testa il comportamento del metodo add( ) verbose
New Flow: (2048, '54.230.202.88', '192.168.1.82', 6, 443, 43900)
     First Packet -> Ether / IP / TCP 192.168.1.82:43900 > 54.230.202.88:https A
New Flow: (2048, '192.168.1.82', '54.230.202.88', 6, 43900, 443)
     First Packet -> Ether / IP / TCP 54.230.202.88:https > 192.168.1.82:43900 A
New Flow: ('Ether / EAPOL EAPOL-Key / Raw',)
     First Packet -> Ether / EAPOL EAPOL-Key / Raw
Flow: ('Ether / EAPOL EAPOL-Key / Raw',): new packet -> Ether / EAPOL EAPOL-Key / Raw
Flow: (2048, '54.230.202.88', '192.168.1.82', 6, 443, 43900): new packet -> Ether / IP / TCP 192.168.1.82:43900 > 54.230.202.88:https PA / Raw
Flow: (2048, '54.230.202.88', '192.168.1.82', 6, 443, 43900): new packet -> Ether / IP / TCP 192.168.1.82:43900 > 54.230.202.88:https FA
Flow: (2048, '192.168.1.82', '54.230.202.88', 6, 43900, 443): new packet -> Ether / IP / TCP 54.230.202.88:https > 192.168.1.82:43900 FA
Flow: (2048, '54.230.202.88', '192.168.1.82', 6, 443, 43900): new packet -> Ether / IP / TCP 192.168.1.82:43900 > 54.230.202.88:https A
Flow: (2048, '192.168.1.82', '54.230.202.88', 6, 43900, 443): new packet -> Ether / IP / TCP 54.230.202.88:https > 192.168.1.82:43900 A
Flow: (2048, '54.230.202.199', '192.168.1.82', 6, 443, 41162): new packet -> Ether / IP / TCP 192.168.1.82:41162 > 54.230.202.199:https PA / Raw
Flow: (2048, '54.230.202.199', '192.168.1.82', 6, 443, 41162): new packet -> Ether / IP / TCP 192.168.1.82:41162 > 54.230.202.199:https FA
Flow: (2048, '192.168.1.82', '54.230.202.199', 6, 41162, 443): new packet -> Ether / IP / TCP 54.230.202.199:https > 192.168.1.82:41162 FA
Flow: (2048, '54.230.202.199', '192.168.1.82', 6, 443, 41162): new packet -> Ether / IP / TCP 192.168.1.82:41162 > 54.230.202.199:https A
Flow: (2048, '192.168.1.82', '54.230.202.199', 6, 41162, 443): new packet -> Ether / IP / TCP 54.230.202.199:https > 192.168.1.82:41162 A
New Flow: (2048, '192.168.1.254', '192.168.1.82', 17, 53, 56149)
     First Packet -> Ether / IP / UDP / DNS Qry "www.html.it."
New Flow: (2048, '192.168.1.82', '192.168.1.254', 17, 56149, 53)
     First Packet -> Ether / IP / UDP / DNS Ans "192.124.249.168"
New Flow: (2048, '192.124.249.168', '192.168.1.82', 6, 80, 57204)
     First Packet -> Ether / IP / TCP 192.168.1.82:57204 > 192.124.249.168:http S
New Flow: (2048, '192.168.1.82', '192.124.249.168', 6, 57204, 80)
     First Packet -> Ether / IP / TCP 192.124.249.168:http > 192.168.1.82:57204 SA
Flow: (2048, '192.124.249.168', '192.168.1.82', 6, 80, 57204): new packet -> Ether / IP / TCP 192.168.1.82:57204 > 192.124.249.168:http A
Flow: (2048, '192.124.249.168', '192.168.1.82', 6, 80, 57204): new packet -> Ether / IP / TCP 192.168.1.82:57204 > 192.124.249.168:http PA / Raw
Flow: (2048, '192.168.1.82', '192.124.249.168', 6, 57204, 80): new packet -> Ether / IP / TCP 192.124.249.168:http > 192.168.1.82:57204 A
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . Continua fino al raggiungimento di 100 pacchetti . . . . . . . . . . . . . . .
/------------------------/
***Flow Key: (2048, '192.168.1.254', '192.168.1.82', 17, 53, 42817)
        Packets count: 1
***Flow Key: (2048, '192.168.1.82', '192.168.1.254', 17, 44240, 53)
        Packets count: 1
***Flow Key: (2048, '46.37.29.204', '192.168.1.82', 6, 80, 37032)
        Packets count: 1
***Flow Key: (2048, '192.168.1.82', '192.124.249.168', 6, 57214, 80)
        Packets count: 1
***Flow Key: (2048, '46.37.29.204', '192.168.1.82', 6, 80, 37034)
        Packets count: 1                                                    // test del metodo show( )
***Flow Key: (2048, '192.168.1.82', '192.124.249.168', 6, 57210, 80)
        Packets count: 1
***Flow Key: (2048, '192.168.1.82', '54.230.202.199', 6, 41162, 443)
        Packets count: 3
***Flow Key: (2048, '46.37.29.204', '192.168.1.82', 6, 80, 37036)
        Packets count: 1
***Flow Key: ('Ether / EAPOL EAPOL-Key / Raw',)
        Packets count: 2
***Flow Key: (2048, '192.124.249.168', '192.168.1.82', 6, 80, 57204)
        Packets count: 15
***Flow Key: (2048, '192.168.1.82', '192.124.249.168', 6, 57208, 80)
        Packets count: 1
***Flow Key: ('Ether / IPv6 / ICMPv6ND_NS / ICMPv6 Neighbor Discovery Option - Source Link-Layer Address 9c:97:26:24:09:26',)
        Packets count: 1
***Flow Key: (2048, '192.168.1.254', '192.168.1.82', 17, 53, 54173)
        Packets count: 1
***Flow Key: (2048, '192.124.249.168', '192.168.1.82', 6, 80, 57208)
        Packets count: 3
***Flow Key: ('Ether / IPv6 / ICMPv6 Neighbor Discovery - Neighbor Advertisement (tgt: fe80::a62b:982c:f3b3:2ec1)',)
        Packets count: 1
***Flow Key: (2048, '216.58.205.170', '192.168.1.82', 6, 80, 59980)
        Packets count: 3
***Flow Key: (2048, '172.217.17.99', '192.168.1.82', 6, 80, 48902)
        Packets count: 3
***Flow Key: (2048, '178.250.0.74', '192.168.1.82', 6, 80, 37718)
        Packets count: 1
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . Continua elencando tutti i flussi rilevati . . . . . . . . . . . . .

/--------------------------------/
Test getFlows with 192.168.1.82
Flow Found:
     *** (2048, '192.168.1.254', '192.168.1.82', 17, 53, 42817)
     *** (2048, '192.168.1.82', '192.168.1.254', 17, 44240, 53)
     *** (2048, '46.37.29.204', '192.168.1.82', 6, 80, 37032)
     *** (2048, '192.168.1.82', '192.124.249.168', 6, 57214, 80)
     *** (2048, '46.37.29.204', '192.168.1.82', 6, 80, 37034)
     *** (2048, '192.124.249.168', '192.168.1.82', 6, 80, 57210)
. . . . . . . . . . . . . . . . . . . . . . . . . . . .continua elencando tutti i flussi che partono/arrivano da/a 192.168.1.82 . . . . . . . . . . . .
/------------------------/
Test getFlows with fe80::9e97:26ff:fe24:926
Flow Found:
/------------------------/
Test getFlows with (2048, '192.168.1.82', '192.168.1.254', 17, 42662, 53)
Flow Found:
     *** (2048, '192.168.1.82', '192.168.1.254', 17, 42662, 53)
/------------------------/
Test getFlows with Ether / IP / TCP 192.124.249.168:http > 192.168.1.82:57214 SA
Flow Found:
     *** (2048, '192.168.1.82', '192.124.249.168', 6, 57214, 80)
/------------------------/
Test getFlows with UDP
None
/------------------------/
Test Biggest Flow
Biggest Flow: (2048, '192.124.249.168', '192.168.1.82', 6, 80, 57212) with -> 15 Packets
/------------------------/
```