UNIVERSITÀ DEGLI STUDI DI BRESCIA

Sicurezza Informatica

A.A. 2020-2021
Prof. Federico Cerutti

TECHINT: Darknet honeypot

Stefano Trerotola (716197)

Introduzione

Un honeypot è un apparato informatico privo di utilità se non quella di attirare attacchi informatici e tracciarne l'esecuzione per mezzo di sonde. In questo report vengono messi in pratica approcci basati su apprendimento automatico per il riconoscimento di traffico di rete malevolo partendo dal traffico riscontrato all'interno di una honeynet.

1 Preprocessing Dataset

Il dataset proposto è composto da 81132277 elementi, ciascuno di essi rappresenta un pacchetto che ha attraversato la rete soggetto. Le informazioni associate a ciascun pacchetto sono di basso livello essendo perlopiù limitate a ciò che può essere estratto dagli header tcp, ed includono timestamp, ip e porta sorgente e destinazione, protocollo di livello 3 e 4 utilizzato, lunghezza pacchetto, flags tcp e un'informazione binaria sulla possibilità che il pacchetto sia parte di un attacco di tipo Mirai.

Valori mancanti ed attributi non informativi

Avendo notato che il protocollo di livello 3 utilizzato è sempre IP (`ethtype 0x0800`) e che il protocollo di livello quattro è sempre TCP (`proto 6`) ho scartato questi attributi. Ho in seguito rilevato che il 6.1% dei pacchetti non riportava informazioni per gli attributi `tcp_flags` e `mirai`; pertanto, ho scelto di eliminare questi pacchetti in quanto una volta privata di queste informazioni una riga risulta essenzialmente priva di significato.

Dummy variables

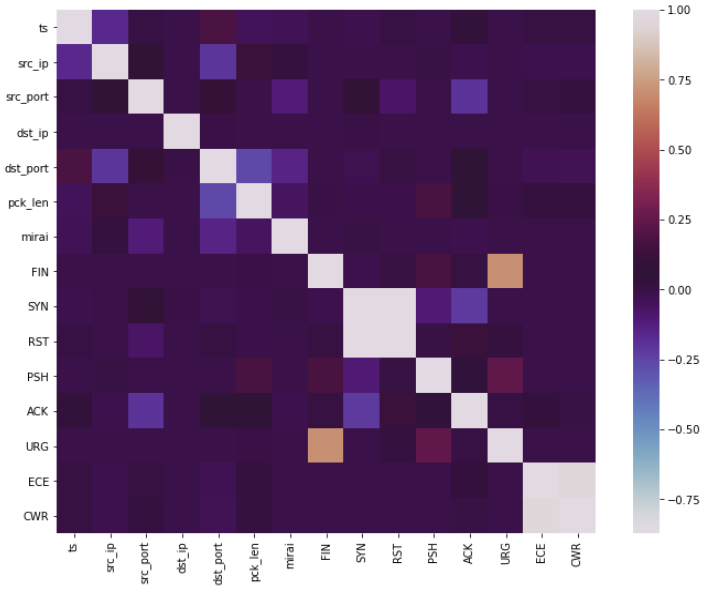
In seguito, ho spacchettato l'informazione dell'attributo `tcp_flags` secondo la sua semantica ufficiale sostituendo questo attributo con nove attributi binari rappresentanti

ciascun flag. Questo passaggio è fondamentale per l'efficacia della maggior parte degli algoritmi di apprendimento automatico, in particolare, lo è per quelli basati su reti neurali e, in generale, su moltiplicazione e somma algebrica. Delle nove classi, ho successivamente scartato NS in quanto sempre zero, e CWR in quanto sempre uguale a ECE.

Scaling e PCA

Ho applicato scalatura per mezzo di `sklearn.preprocessing.StandardScaler` agli attributi `timestamp`, `ip` e `port` (sia mittente che destinazione) al fine di migliorare le prestazioni degli algoritmi di apprendimento.

A scopo di analisi ho applicato PCA riscontrando come sia possibile proiettare il dataset su appena sei attributi anziché i quattordici scelti mantenendo il 99.45% della varianza complessiva. Se, da un lato, ciò renderebbe più snello e velocemente elaborabile il dataset, dall'altro farebbe perdere valenze semantiche necessarie ai passi successivi: nello specifico, perdere informazione diretta su timestamp, ip e porte rende impossibile associare tra loro pacchetti differenti. Ho quindi preferito non applicare PCA, e salvare il dataset così elaborato.



1 Matrice di correlazione

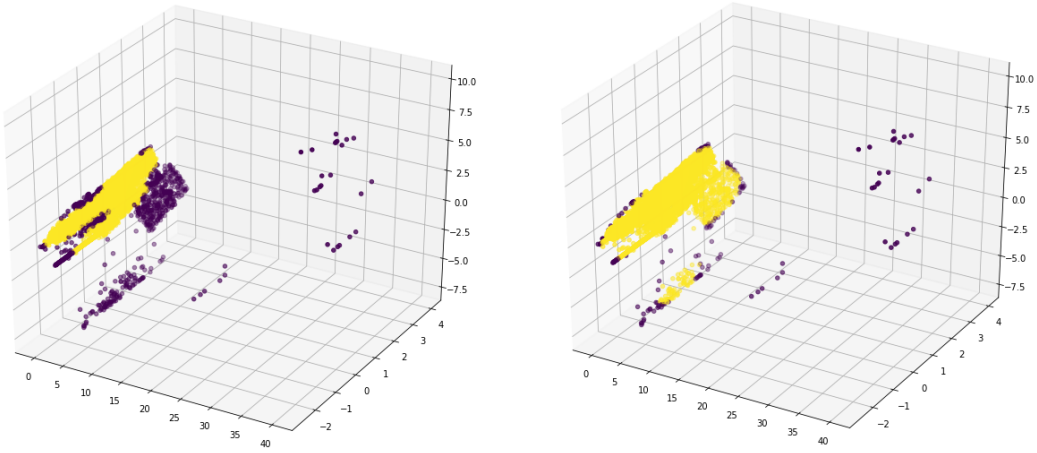
2 Analisi statica per raggruppamento

Essendo i pacchetti riportati in questo dataset tutti di origine malevola, non è possibile utilizzare alcuna tecnica di apprendimento supervisionata sui singoli pacchetti: ciò che è possibile fare, tuttavia, è ottenere una loro descrizione che possa essere utilizzata come identikit per discernere tra ulteriori pacchetti (che, in caso non aderissero a nessun identikit creato, potrebbero essere sia parte di schemi di attacco non noti, sia essere traffico non malevolo). Bisogna comunque fare i conti con la povertà di informazioni presenti in questo dataset: gli attributi riportano, come già illustrato, solo informazioni di basso livello (non è catturata la semantica applicativa di codesti pacchetti), pertanto non è difficile immaginare

come del traffico innocuo possa avere le stesse caratteristiche di del traffico malevolo. Ciò che, invece, potrebbe essere utilizzato per distinguere fra traffico malevolo e non malevolo è la persistente categorizzazione dei pacchetti appartenenti ad un flusso di traffico ad una delle categorie malevole. Ipotizzo (e, non avendo del traffico non malevolo di esempio, non ho potuto verificare l'efficacia del seguente approccio) l'utilizzo di una macchina a due livelli: il primo livello utilizza algoritmi di clustering per attribuire un'etichetta di appartenenza del singolo pacchetto ad uno degli identikit malevoli, o nessuna etichetta; il secondo livello basato su rete neurale di tipo ricorrente (o non ricorrente, basti pensare all'efficacia delle architetture di tipo Transformers) addestrata a riconoscere sequenze di etichette di appartenenza a cluster diversi, e categorizzare quindi il traffico come malevolo o meno.

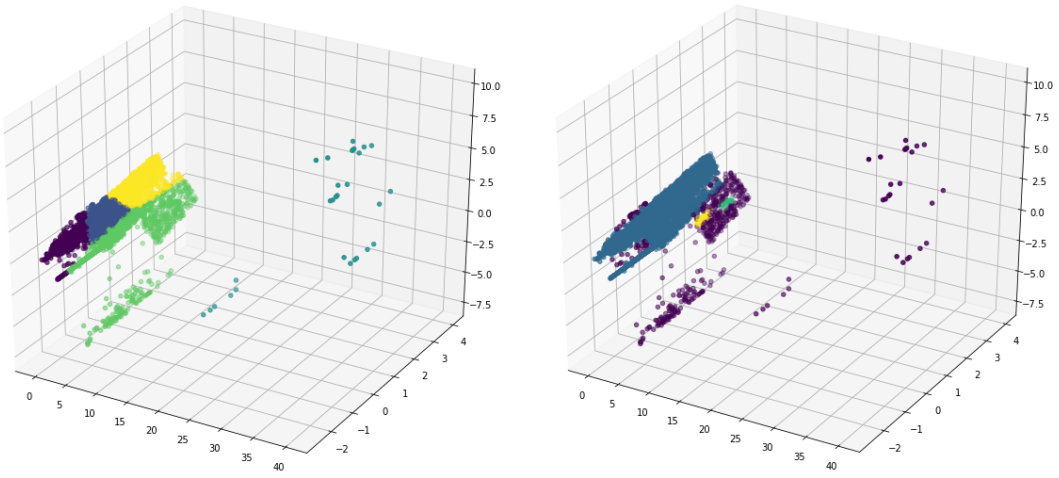
Mi sono, tuttavia, limitato ad esplorare la possibilità di creare cluster seguendo gli approcci di K-Means, DBSCAN, LOF, OSVM e gerarchico. Data la complessità computazionale di questi algoritmi, e la necessità di visualizzare graficamente i risultati per una valutazione efficace, ho eseguito test su una frazione di circa lo 0,03% del dataset preprocessato e proiettato su tre dimensioni tramite PCA.

A seguito di valutazioni grafiche supportate dalla metrica silhouette ritengo che l'a3lgoritmo più efficace, almeno nelle proiezioni 3D, risulti essere quello di tipo gerarchico. OSVM e LOF hanno evidenti difficoltà nel raggruppare i punti, permettendo a punti esterni di invadere un gruppo denso e compatto, inoltre, questi algoritmi non possono produrre più di due raggruppamenti.



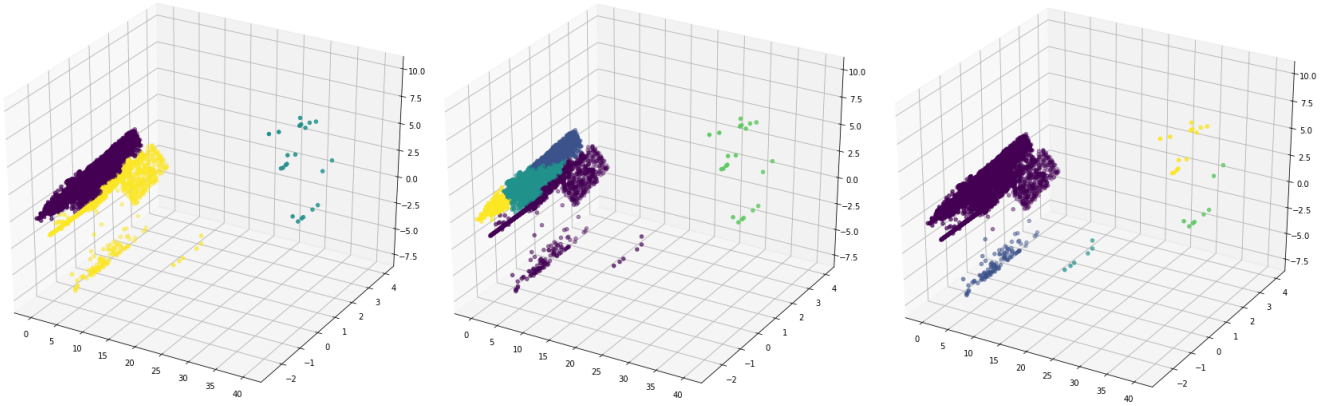
2 LOF (sinistra) e OSVM (destra) hanno difficoltà nel trovare i confini tra i gruppi

K-Means e DBSCAN sono in grado di creare più di due raggruppamenti, ma entrambi soffrono di bias: DBSCAN è molto sensibile ai parametri, e facilmente produce gruppi di dimensioni molto sbilanciate; K-Means, invece, produce gruppi più bilanciati, ma è caratterizzato dalla creazione di confini lineari tra i cluster che difficilmente combaciano coi confini reali.



3 K-Means (sinistra) e DBSCAN (destra) possono creare più di due raggruppamenti, ma soffrono di evidenti bias

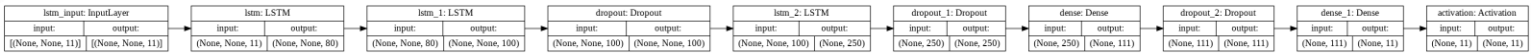
Le tecniche gerarchiche, quindi, sono le più efficaci in quanto riescono a combinare i vantaggi di DBSCAN e K-Means: gruppi di dimensioni meno sbilanciate e confini non lineari.



4 Sinistra: tre gruppi con link "ward". Centro: cinque gruppi con link "ward". Destra: cinque gruppi con link "average". Link "ward" costruisce i raggruppamenti riducendo la varianza tra i vari cluster, pertanto, è più simile a K-Means che non gli approcci con link singolo, medio o completo.

3 Analisi dinamica con LSTM

L'analisi dinamica è sicuramente più promettente ed ha il vantaggio di poter essere eseguita con le stesse tecniche che si utilizzerebbero in ambiente supervisionato pur non avendo delle classi obbiettivo su cui calcolare metriche (accuratezza, precisione, recupero...). Ciò che ho fatto è stato addestrare una rete neurale multilivello basata su LSTM a riconoscere le sequenze di pacchetti malevoli presenti in questo dataset. Pur non avendo delle controparti sequenze di pacchetti non malevoli, è ragionevole aspettarsi che il modello creato sia in grado di prevedere correttamente le sequenze di traffico malevole sulle quali

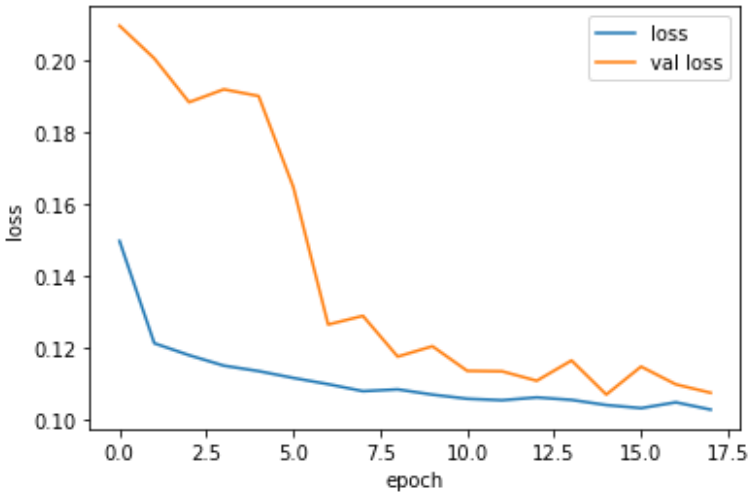


è stato addestrato, e che sia incapace di prevedere pattern di traffico non malevolo (dunque, abbia loss elevata al difuori da una honeynet).

La stessa architettura di rete può essere addestrata a riconoscere un particolare tipo di pattern di attacco in base a come viene costruito il training set per essa. Ho prodotto un training (e test) set contenente attacchi originanti da un unico indirizzo ip e destinati a qualunque indirizzo ip. Diverse condizioni porterebbero a diversi training set, e dunque a diversi riconoscitori. Ad esempio, si potrebbe costruire un secondo training set più selettivo contenente solo gli attacchi provenienti da stesso ip (eventualmente, stessa porta) e portati ad ip-porte destinazioni crescenti, in sequenza: un training set così costituito sarebbe naturalmente prono alla descrizione di port scan.

La rete proposta viene addestrate su sequenze di 17-50 pacchetti di cui viene mantenuta informazione su destinazione, lunghezza pacchetto, mirai e flags tcp (lo scopo è prevedere il pacchetto successivo).

La rete si addestra rapidamente (in genere bastano cinque epoche) e raggiunge valori di loss attorno a 0.1-0.13, dove gran parte dell'errore è solitamente concentrata nella predizione della destinazione.



3 Storia d'addestramento su training set generato a partire da 100000 elementi (valutato sul suo 10%)

Utilizzo

Dataset privato. Codice, descrizione e risultati presso: [CyberSecLabBS/Honeynet (github.com)]