關於專題

時間: 大三、大四期間

類別: 畢業專題

專題名稱: 基於深度學習的惡意流量分類器

負責內容: 找尋相關可用方法與資料集、修改模型以符合使用需求以及開發擷取網卡流量的程式。

專題動機

透過專題指導教授所提供的題目作為媒介，並結合資策會的5G訊號指紋辨識研究，在專題實驗中了解深度學習、網際網路的各項概念，透過積極的練習增加自身的實力，期許自己能完成專題，對於分類器在資安方面的應用做出一點貢獻。

專題目標

處理加密流量的分類

基於安全與隱私的緣故，有越來越多的網路流量採取加密的方式，例如像是VPN、HTTPS、SSH 等協定，使得無法取得封包內容(payload)，而連帶使得偵測機制的準確性有所下降。因此我們把 TCP/UDP/IP 封包標頭(header)的 byte 內容看做像是灰階影像般(把各個 byte 看成是 pixel，byte 值看成是灰階值)。如此一來就可以用一般影像分類的技術(ex. CNN)來進行分類。

早期偵測

要取得 flow 的統計特徵往往需要等待檢視完該 flow 所屬的全部封包後才能得到正確的統計資訊。但遇到需要判別並阻擋惡意流量時，則無法等待看完 flow 所屬的全部封包再行判斷，因為該惡意流量的封包可能已全數傳完。因此，我們選擇只檢查每個 flow 中前幾個 packet 的前幾個 bytes 以進行早期檢測。

開發擷取網卡流量的常駐程式

開發網卡流量擷取的常駐程式，具備鑑別正常與惡意流量之功能，可鑑別如 Ransomware、Remote Access Trojan (RAT)、C&C connection 等惡意活動。

相關文獻

[1] P. M. S. Sánchez, J. M. J. Valero, A. H. Celdrán, G. Bovet, M. G. Pérez and G. M. Pérez, "A Survey on Device Behavior Fingerprinting: Data Sources, Techniques, Application Scenarios, and Datasets", in IEEE Communications Surveys & Tutorials
.簡介:介紹使用規則(rule)方式、統計方式(statistical)、知識(knowledge)方式、機器學習方式、時間序列等方式，來分析網路裝置行為特徵。除此之外，其中還整理了現今以網路為主的公開資料集來源，以協助攻擊行為與異常偵測相關的研究。

[2] Benjamin J. Radford, Bartley D. Richardson, Shawn E. Davis, "Sequence Aggregation Rules for Anomaly Detection in Computer Network Traffic", arXiv:1805.03735v2 [cs.CR] 14 May 2018.簡介:將flow視為機器間的"語言"而使用自然語言處理，並用LSTM、RNN架構來測試偵測惡意流量的表現，雖然測試結果顯示在偵測大多數惡意流量時frequency-based的架構表現較好，但這篇論文經過測試後得到一個想法是LSTM或許適合偵測一個攻擊的flow的開始，而不是整個與攻擊有關的flow。

[3] R. Hwang, M. Peng, C. Huang, P. Lin and V. Nguyen, "An Unsupervised Deep Learning Model for Early Network Traffic Anomaly Detection," in IEEE Access, vol. 8, pp. 30387-30399, 2020, doi: 10.1109/ACCESS.2020.2973023.
簡介:提出了一種有效的異常流量檢測機制，D-PACK，它由一個卷積神經網絡（ CNN ）和一個自動編碼器(AutoEncoder)所組成，用於自動分析流量模式和過濾異常流量。值得注意的是，D-PACK 僅檢查每個 flow 中前幾個 packet 的前幾個 bytes 以進行早期檢測。實驗結果表明，僅通過檢查每個 flow 中的前兩個 packet，D-PACK 仍然有接近 100% 的準確率，並且僅有 0.83% 的誤報率。該機制的特點是可以有效減少需要處理的 packet 的數量並及時阻止惡意流量。
方法:只取每個 flow 的前 n 個 packets，並將每個 packet 去除到固定長度 l。將 n*l 長度的 bytes 丟到一維的 vector 中後做為 CNN 的 input。1D-CNN 適合 sequential data 或是語言，為了不要有無關的輸入因此而使用 1D-CNN，inpu t為將 n 個長度為 l 個 bytes 的 packets 串接起來，如果不夠就補 0。

[4] J. Yang and H. Lim, "Deep Learning Approach for Detecting Malicious Activities Over Encrypted Secure Channels," in IEEE Access, vol. 9, pp. 39229-39244, 2021, doi: 10.1109/ACCESS.2021.3064561.
簡介:因為 SSL 的 conversation data 由公鑰系統加密，並且使用自己的 SSL record 數據單元，因此，這邊提出了一種新的惡意 SSL 流量檢測方法。從捕獲的 IP packets 中重新組裝 SSL record，並使用深度學習方法提取 SSL record 的特徵。SSL record 序列使用長短期記憶自動編碼器(LSTM Autoencoders)進行編碼，然後為每個 SSL flow 生成編碼特徵圖(feature map)。這些特徵圖被輸入到基於卷積神經網絡(CNN)的分類器，以偵測 SSL flow 是否為惡意流量。

- A. PROPOSED SSL RECORD PARSER

  僅需要能辨識 SSL record chunks 並且把這些 chunks 整合在一起變成原本的 message。

  當 parsing agent 讀到一個新的封包的時候會執行以下步驟:

  1. 將相同 flow 的 tcp segments 聚在一起: 藉由將相同 source/destination IP 地址與 port number 的封包放在一起。另外因為 SSL 在 TCP 上層，因此會丟棄不是 TCP/IP 的封包。
  2. 將沒有 SSL records 的 segments 丟棄: 有一些 TCP 封包沒有包含任何 SSL records 的部分，像是封包的 payload 大小為零的時候，或是 TCP keep-alive 跟 zero-window probe 的封包。這些封包提供 TCP 協定保存與監控連線的功能，但對 SSL record 的重組沒有幫助。keep-alive 與zero-window probe 的封包的辨識都很簡單，兩者都有 1 byte 的 payload，並且 keep-alive 的 SEQ 比 expected SEQ(也就是 SEQ 跟相同 flow direction 最後看到的 segment 的長度的總和)少 1 byte，而 zero-window probe 的封包的 SEQ 則是與 expected SEQ 相同。
  3. 處理 out-of-ordered 的 TCP segments。

- B. SSL RECORD SEQUENCE CLASSIFIER

  偵測流程由以下三個階段組成：

  1. 預處理: 首先檢測從給定的 record sequence 的前 n 個 records，這些 records 都被去除到 b bytes 的長度。如果 record 的大小小於 b 則補 0。由於在 handshake 之後傳遞 payload 是被加密的，因此當遇到 handshake 之後的 record 就全替換成 0。最後 detector 藉由將每個 byte 值除 255 來 normalizes feature map。另外如果 session 是少於 n 個 records 的話則補上 b 大小的 zero vectors 來確保所有 records 的長度是相同的。
  2. 特徵提取: 因為高維數據上模式識別的成功或失敗在很大程度上取決於找到豐富而輕量的特徵表示法，因此 detector 採用 LSTM 作為嵌入的 autoencoder 這步驟是相當重要的。考慮到 LSTM 是專門用來抓取連續特徵的，因此使用由兩層 encoding 與 decoding 的 stacked LSTM。在將 encoder train 完後就可以用 encoder 產生的 vecter 來做分類。
  3. 分類: 將 encoder 產生的 vector 拼接起來產生一個類似圖像的 input。最後用僅有 3 層卷積層與 4 層全連接層的 CNN 分類器是為了避免 overfitting 的問題。

[5] S. Rezaei and X. Liu, "Deep Learning for Encrypted Traffic Classification: An Overview," in IEEE Communications Magazine, vol. 57, no. 5, pp. 76-81, May 2019, doi: 10.1109/MCOM.2019.1800819.

簡介:Port-based、packet 檢測和經典的機器學習方法過去被廣泛使用，但由於互聯網(IoT)流量的急劇變化，特別是加密流量的增加，導致它們的準確性有所下降。 隨著深度學習方法的普及，近期的研究使用這些用於流量分類的方法並提高了精準度。 這裡面介紹了基於深度學習的流量分類的 framework。 介紹了常用的深度學習方法及其在流量分類任務中的應用。

系統設計與架構

A.資料集選用

我們主要使用的資料集為 USTC-TFC2016，其中包含了下表所列的 10 種惡意流量封包。除此之外，從 Malware Capture Facility Project 中蒐集包含以下種類的 Pcap File 來增加資料集的多樣性。

malware_types	malware_types
Cridex – Trojan/Worm	Geodo – Trojan
Htbot – Backdoor Trojan	Miuref – Trojan
Neris – Trojan	Nsis-ay – Trojan
Shifu – Trojan	Tinba – Trojan
Virut – Trojan/Viruses	Zeus – Trojan

B.分類器架構

我們擷取每條 flow 的前 8 個封包的前 80 個 bytes，並使用將其排列成大小為 (8, 80) 的向量後(不足之處補 0)，使用2維卷積網路層與2維最大池化層來辨認潛在的模式。由於 flow 的封包是有時序性的，因此我們加入了長短期記憶(Long Short-Term Memory, LSTM)模型做嘗試，同時減少訓練所需的參數量。除了最後一層因為要做多類別分類的輸出因此使用 softmax 來當做激勵函數以外，其餘每層都使用整流線性單位(Rectified Linear Unit, ReLU)。

以下為我們模型架構的流程與參數:

Conv2d(1, 32, (2, 4), 1) – MaxPool2d((2, 3)) – BN – Conv2d((32, 64), (2, 4), 1) – MaxPool2D((2, 3)) –

BN – LSTM(280, 100) – FC(6400, 100) – FC(100, 11)

Conv2d(input_channel, output_channel, kernel_size, stride)

MaxPool2d(kernel_size)

LSTM(input_size, hidden_size)

Linear(in_features, out_features)

BN: Batch Normalization

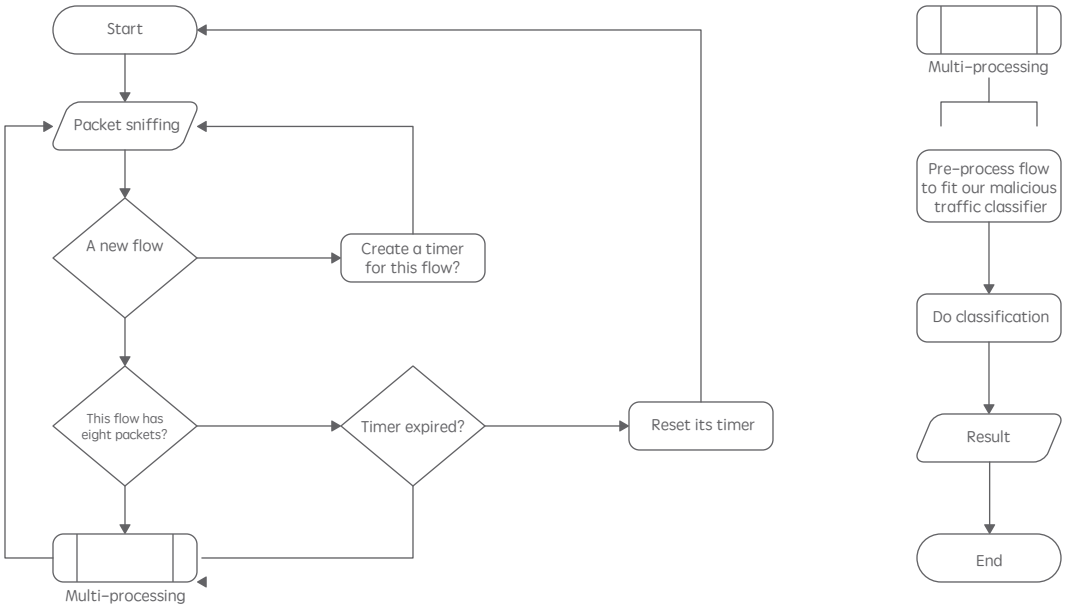
Parameters Experiments

Model	Architecture	Accuracy
Model-1a	Conv2D-BN-Conv2D-BN LSTM-Dropout(0.4)-FC-Dropout(0.2)-FC	95%
Model-1b	Conv2D-BN-Conv2D-BN LSTM-Dropout(0.4)-FC-Dropout(0.2)-FC	96.8393%
Model-2a	Conv2D-BN-Conv2D-BN LSTM-FC-FC	97.3082%
Model-2b	Conv2D-MaxPool2d-BN-Conv2D-MaxPool2d-BN-LSTM-Dropout(0.2)-FC-Dropout(0.2)-FC	97.3121%
Model-2c	Conv2D-MaxPool2d-BN-Conv2D-MaxPool2d-BN-LSTM-FC-FC	97.4801%

Compare methods proposed by papers with ours

	Dataset	Input data(preprocessed)	Archi
[3]	USTC-TFC2016	Take the first 80 bytes of the first 2 packets in each flow. Each packet is trimmed into a fixed length of 80 bytes, starting with the header field. The data are padded with zeros if the number of packets is less than 2 or the packet size is less than 80 bytes.	CNN(Feature Extracting) + AutoEncoder(Binary classifier)
[4]	MTAN、CTU	Reassemble SSL records from captured IP packets, and take the first 256 bytes of the first 32 records in each record sequence. Each record is trimmed into a fixed length of 256 bytes, starting with the header field. The data are padded with zeros if the number of records is less than 32 or the record size is less than 256 bytes. When a record contains an encrypted region, replace its payload contents with a series of zero values.	Stacked LSTM(Feature Extracting) +CNN(Classifier)
Our	USTC-TFC2016	Take the first 80 bytes of the first 8 packets in each flow. Each packet is trimmed into a fixed length of 80 bytes, starting with the header field. The data are padded with zeros if the number of packets is less than 8 or the packet size is less than 80 bytes.	CNN(Feature Extracting) + LSTM(Classifier)

B.擷取網卡流量的常駐程式流程圖



程式開始執行並且收到封包後，首先會將封包中的 IP 位址與 PORT 號提取出來組成 KEY，然後檢查儲存 flow 的資料結構中是否包含該 KEY。若包含，則會將封包儲存至該 KEY 的 Value 中，並且檢查該 KEY 的 Value 是否累積到 8 個封包 。如果該 KEY 的 Value 累積到 8 個封包，就會去進行辨識；反之，則重置該 KEY 的 Timer。如果 flow 的資料結構中沒有包含該 KEY，則將 KEY 新增到資料結構中並且將封包儲存在該 KEY 的 Value 中，並且新增一個屬於該 KEY 的 Timer。 若某個 KEY 的 Timer timeout 就會直接將資料結構中該 KEY 所儲存的封包直接做辨識。做辨識會開一個能夠平行運算的行程來做，若偵測到惡意流量時，會產生log file，同時 log file 也會由此 process 來產生。

Time measurement

- 將 packet 根據 flow 儲存所需時間

接收封包數	平均消耗時間
100	0.0181 ms
500	0.0253 ms
1000	0.0256 ms
5000	0.0315 ms

- fork 出可平行運算的 process 的時間

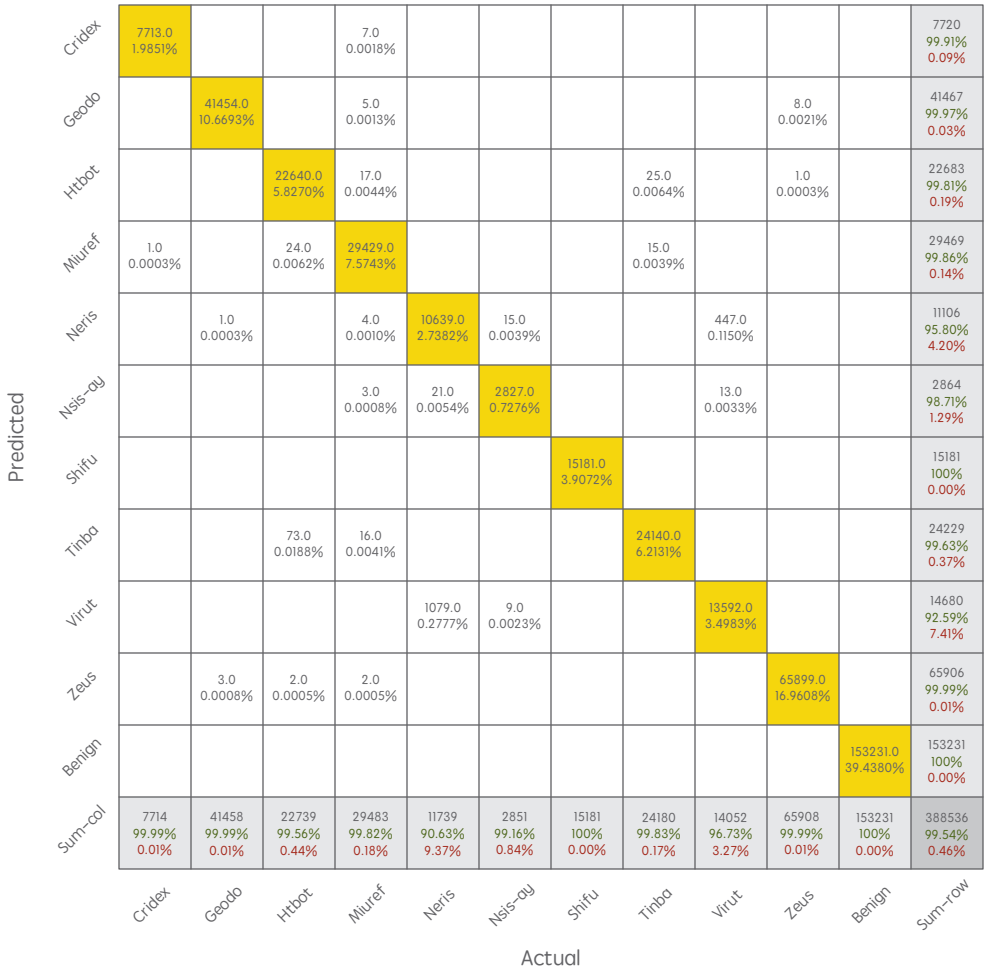
接收封包數	平均消耗時間
100	0.2263 ms
500	0.2726 ms
1000	0.3052 ms
5000	0.3133 ms

- 將封包轉成 model 的 input type model 辨認是否為惡意流量與輸出 log file 所需的時間

接收封包數	平均消耗時間
100	20.4382 ms
500	29.0516 ms
1000	23.7197 ms
5000	26.6281 ms

近期成果

Confusion Matrix



Result

Cridex	Accuracy: 99.8964% Precision: 99.9093% Recall: 99.9870%
Geodo	Accuracy: 99.9590% Precision: 99.9686% Recall: 99.9904%
Htbot	Accuracy: 99.3767% Precision: 99.8104% Recall: 99.5646%
Miuref	Accuracy: 99.6816% Precision: 99.8643% Recall: 99.8168%
Neris	Accuracy: 87.1621% Precision: 95.7951% Recall: 90.6295%
Nsis-ay	Accuracy: 97.8878% Precision: 98.7081% Recall: 99.1582%
Shifu	Accuracy: 100.0000% Precision: 100.0000% Recall: 100.0000%
Tinba	Accuracy: 99.4685% Precision: 99.6327% Recall: 99.8346%
Virut	Accuracy: 89.7754% Precision: 92.5886% Recall: 96.7264%
Zeus	Accuracy: 99.9757% Precision: 99.9894% Recall: 99.9863%
Benign	Accuracy: 100.0000% Precision: 100.0000% Recall: 100.0000%

Metrics

- $\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$
- $\text{Precision} = \frac{TP}{TP+FP}$
- $\text{Recall} = \frac{TP}{TP+FN}$

結論與未來發展

近年來在學術界已經有許多利用機器學習技術來分類網路流量與偵測惡意流量的做法，但大部分都沒有同時考慮到封包流量來源的裝置類型或個別應用協定，因此成果大都只侷限於少量且特定的資料集，且僅做精確度分析，但未對偵測結果做較深入的解讀(例如是來自什麼樣設備的攻擊，或是察覺整個攻擊的流程，包含感染及後續攻擊)。因此未來將在這些問題上做更加深入的探討，試圖解決以上問題。