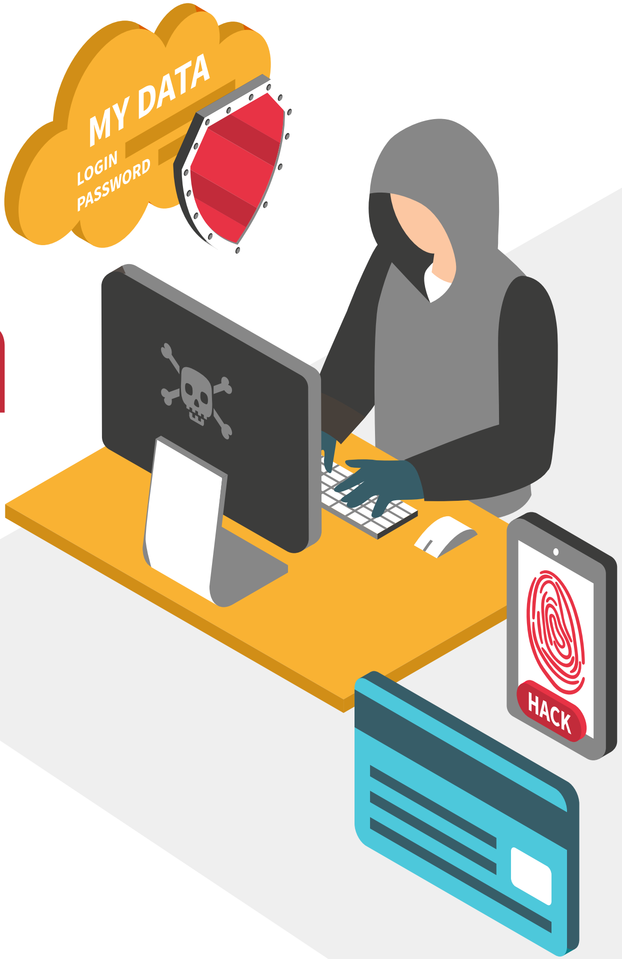
Técnicas de Percepção de Redes

Attacks through Twitter



Diana Rocha 98524
Diogo Correia 90327

7 de novembro de 2023

Why Twitter?

We chose twitter because of how easy it is to run a bot on it and also we were interested in study these anomalies through social media.
It allows for a great communication channel between a bot and a commander. The data stolen can be disguised as a day to day tweets.

Importance of the security issue/solution

- ➔ The victim’s personal data within their own device can be easily stolen or compromised.
- ➔ The bot is not easily detected, because it runs in its own Twitter account (so the victim never detects any unusual behavior and is not aware that its device is compromised).

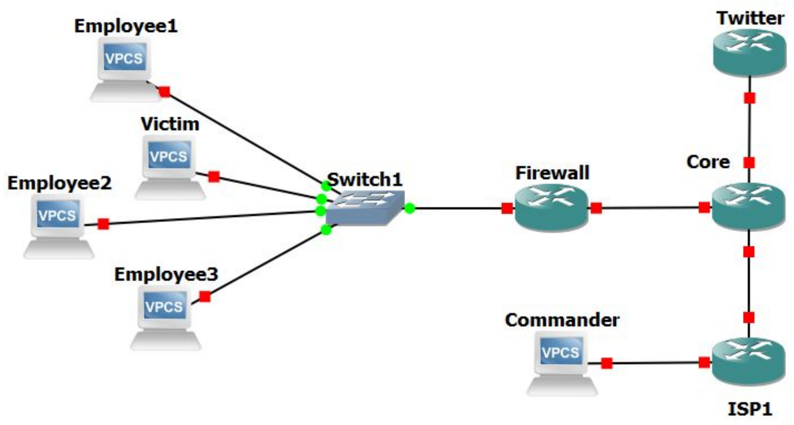
Real World Scenario

We use **Command and Control** by infecting a victim's computer with a bot that is able to run commands in the victim's operating system. On our side, we have a program that behaves as the commander that sends commands to the bot.

We use **Twitter** as the communication channel and it's *hashtag* function as a way to subscribe to topics.

The bot can infect a device via, for example, the installation of a dummy program, behaving as a *trojan*.

Our Scenario



We assume that we have:

- **Firewall** as a network security device that monitors incoming and outgoing network traffic and decides to allow or block specific traffic according to a defined set of security rules.
- The **Employees** that use Twitter will have their traffic analysed, to see their behaviour, and get their profile taken.
- The **Victim** has the bot on his device.

Our Scenario

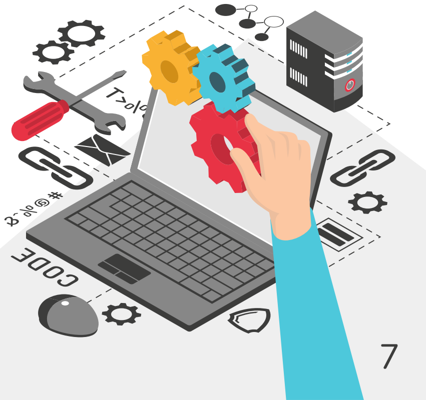
- **Scenario:** bot uses Twitter with his own account to exchange data with the commander and access to files on the infected user's device. The actions through Twitter include writing tweets and reload the page (communication checks).
- **Types of bots**:
  -> Dumb: uses a fixed time interval for communication checks and response intervals.
  -> Intermediate: introduces some randomness in its timing, the communication checks and response intervals have a random component, making the bot less predictable.
  -> Smart: introduces more randomness in its timing, the communication checks and burst intervals have a wider range of randomness, making this bot less predictable in terms of response times. It also may respond faster or slower than the "regular" bot, introducing additional unpredictability.

Behaviours

- **Good:**
  - The users go through Twitter and uses it as usual
- **Bad:**
  - The bot run isolated (user does not use Twitter nor does anything to generate traffic)
  - The bot runs while other traffic is also being generated on Twitter (bad + good)

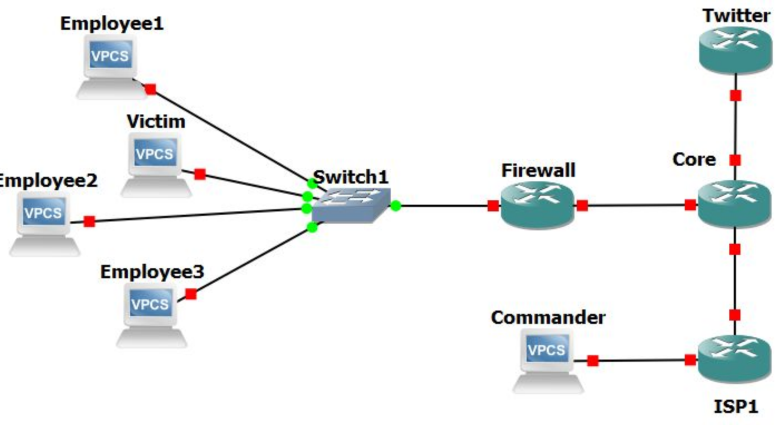
01.

Test Scenario



Data sources

- Captures of the Twitter traffic
  - 9 Captures:
    - 3 Clients
    - 3 bots isolated
    - 3 bots + good traffic



Metrics extracted

- Twitter packets
  - Number of **Upload** and **Download Packets**
  - Number of **Upload** and **Download Bytes**

Observation Window

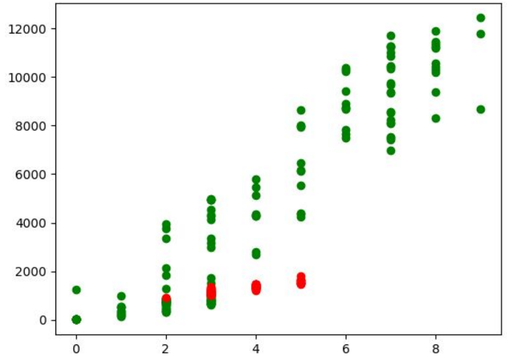
- Width of 80 sampling intervals
- Slide of 10 sampling intervals
- Sampling period of 1 second

Features to extract

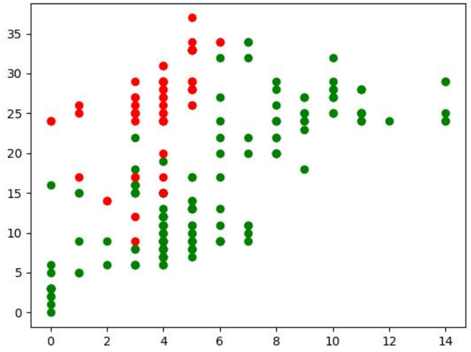
- For each metric defined
  - ➔ Mean, median, standard deviation
  - ➔ Percentiles (75%, 80%, 90% e 98%)
- Silence vs Activity:
  - ➔ Size of both periods
  - ➔ Mean and standard deviation of both periods

Plot - Features

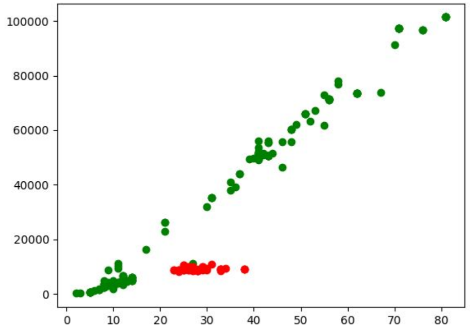
Mean up packets vs std deviation up packets



Mean up bytes vs std deviation up bytes

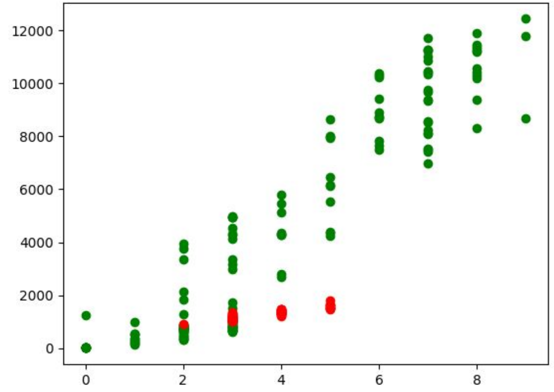


Std deviation down bytes vs 98 percentile down bytes

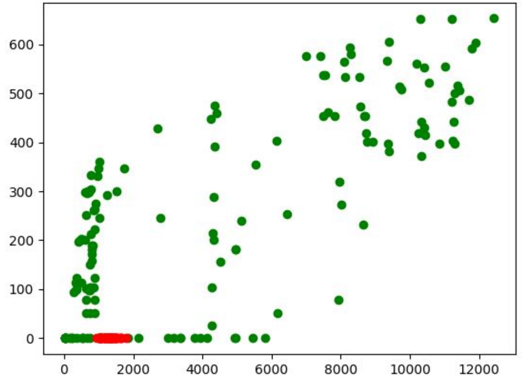


Plot - Silence and Activity Features

Size silence up packets vs std deviation silence up packets



Size activity up packets vs std deviation activity up packets



Training and Test Features

- One train set with all features from 2 good users
- One test set with all features from the 1 good user not used
- One test set with the bot features:
  -> bot's isolated
  -> all bot's isolated together
  -> bot's + good traffic
  -> all bot's + good traffic together

Anomaly Detection

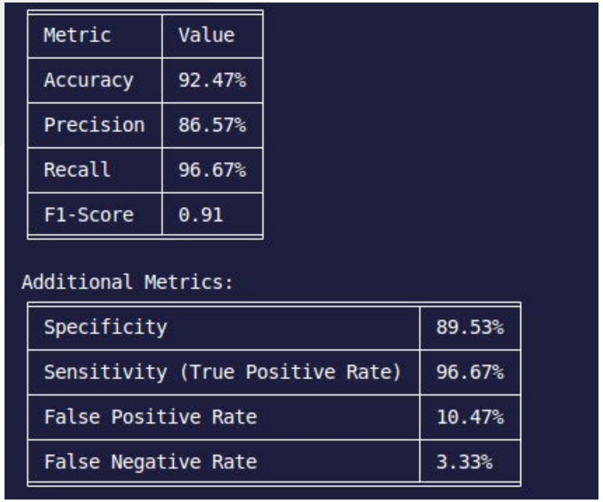
- **Clustering:**
  - **K-means** without PCA features
- **Statistical Analyses:**
  - **Centroids distances** with PCA features
  - **Centroids distances** without PCA features
- **Anomaly Detection:**
  - **Isolation Forest** with PCA features
  - **Isolation Forest** without PCA features
- **Machine Learning:**
  - **One Class Support Vector Machine** with PCA features (Linear, RBF and Poly Kernels)
  - **One Class Support Vector Machine** without PCA features (Linear, RBF and Poly Kernels)

Anomaly Detection - Results & Conclusions

- PCA does not improve the results in centroids distance but improves the results in the one class SVM and isolated forest.
- Typically, we want a balance between precision and recall, especially in an anomaly detection scenario where both false positives and false negatives can have significant consequences. The F1-Score can be a particularly useful metric in this balance, as higher F1 values indicate a good balance between precision and recall.
- This way, K-Means seems to offer the most balanced approach, with the highest value of F1-score for all 3 bots isolated and even with them mixed with good traffic.
- When it comes to comparing results between all bots and expected results, K-means presents greater difficulty in detecting the intermediate one(75%), then the smart one(89%), and finally the dumb one(91%).
- This is unlike other approaches, where the results may not be as high but follow the expected pattern, with the smart bot being the most difficult to detect, followed by the intermediate bot and, finally, the easy bot.

K-means results

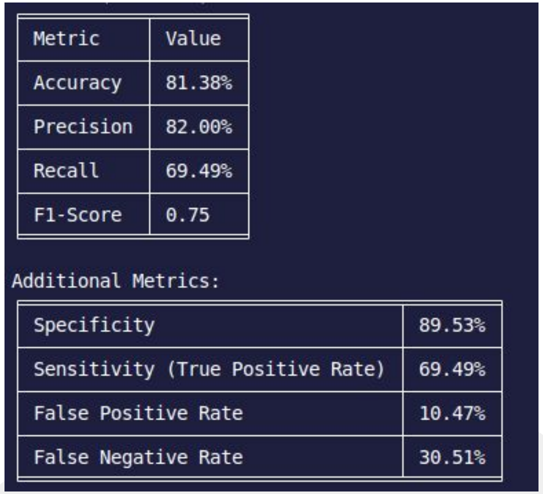
Intermediate Bot



```
Metric       Value
Accuracy     92.47%
Precision    86.57%
Recall       96.67%
F1-Score     0.91

Additional Metrics:
Specificity                          89.53%
Sensitivity (True Positive Rate)     96.67%
False Positive Rate                  10.47%
False Negative Rate                  3.33%
```

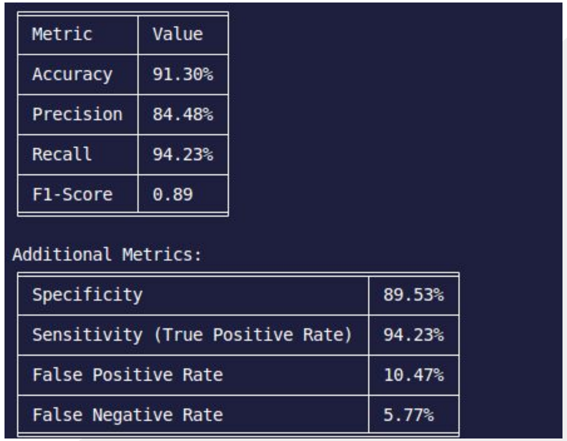
Basic Bot



```
Metric       Value
Accuracy     81.38%
Precision    82.00%
Recall       69.49%
F1-Score     0.75

Additional Metrics:
Specificity                          89.53%
Sensitivity (True Positive Rate)     69.49%
False Positive Rate                  10.47%
False Negative Rate                  30.51%
```

Smart Bot



```
Metric       Value
Accuracy     91.30%
Precision    84.48%
Recall       94.23%
F1-Score     0.89

Additional Metrics:
Specificity                          89.53%
Sensitivity (True Positive Rate)     94.23%
False Positive Rate                  10.47%
False Negative Rate                  5.77%
```

Future Work

For training and test features, we are thinking in doing more approaches:

- One train set with 75%, 50% and 25% features from each user
- One test set with the rest of the features
- One test set with the bot features we know

Do more types of methods or algorithms for anomaly detection, so we can obtain more certains about which one is ideal for our case