Abstract

Most intrusion detection systems that detect Application Programming Interfaces (APIs) misuse or abuse use Machine Learning or Data Mining algorithms. Whenever developers use APIs, they often make mistakes that lead to bugs or system crashes, or even application security vulnerability. Many times APIs are misused to carry out malicious activities. This is called API abuse. Earlier, the API misuse detectors only focused on making a naive assumption that API usage that deviates from the most frequent API usage is misuse. However, the newer systems use application logs to predict possible security threats, including intrusion-like outer organization attacks and misuse detection. These logs are full of messages that contain warnings or even errors. However, since the number of logs is enormous, it is impossible for only humans to monitor them. So, employing natural language processing tools for text encoding and machine learning methods for automated anomaly detection can help analyze failing applications by highlighting the logs and provide insight into the problem and can help generate alerts in case of the high frequency of anomalous logs. We will use unsupervised machine learning and several machine learning models to achieve the result.

How will your solution help the industry?

Application Programming Interfaces (APIs) often impose constraints like call order or preconditions. API misuses, i.e., usages violating these constraints, may cause software crashes, data loss, and vulnerabilities. Application programming interfaces have become a crucial part of almost every business. APIs are in charge of transferring information between systems within a corporation or to external companies. For example, when one logs in to a website like Google or Facebook, an API processes the login credentials to verify they are correct. However, given the sensitive data being transferred through APIs, it's critical to secure them. Increasingly sophisticated attacks occur per annum , requiring better security controls and monitoring. Through our idea, we plan on devising a way to identify the common API misuse and abuse occurrences, i.e., Account takeovers, Credential Stuffing, Bot Content Scraping, and Fake account creation, and generate alerts against these. Moreover, companies generate billions of log records per week. In such cases,

relevant logs might get muddled in between irrelevant ones. Our idea will help surface the important logs and de-emphasize irrelevant ones.

Solution Idea:

Data Processing and feature extraction

The given access log data (training dataset) will first be parsed and broken down into the following fields/features (as applicable):

1. Timestamp - Time at which the API access request was made
2. IP Address - IP address of the client
3. Country - Country of origin of the request
4. HTTP request referrer- Address of the site from where the request originated.
5. HTTP request method- GET,POST,etc.
6. URI - Address of data requested by client.
7. Number of parameters in the query.

Some more features may be extracted based on the log format. Eg: User ID, Status code, UA String etc.

Data preparation

After extracting the wanted features from raw http server log files, we can label it using two labels: 1 to say that misuse of data as an attack and 0 for normal behaviors or no threat.Labeling should normally be done manually. In API abuse detection, it can be done automatically using a function that will look for specific patterns in each URL and decide about the type of attack.
The retrieved data can now be used to train our classifiers.

Feature Engineering

New features will be created to allow the models to draw better inferences. The new features we plan on creating are :

1. Number of requests per user (Bots tend to have a large number of requests)

2. Number of locations/countries associated with a particular user (A particular account being accessed from several countries will indicate account takeover and credential stuffing)
3. Number of unique IP addresses associated with a user
4. Average rate of requests sent by a user (Scripted bots tend to have a high request rate)
5. Number of API endpoints accessed by a user
6. Number of unique URI requests. (Bots request for lots of data and hence will have several unique URI requests)
7. Number of failed status code requests made by the user (Eg: of the form 4xx, 5xx for HTTP requests : 400-Bad Request, 502- Bad Gateway) (This will indicate an API misuse/error by a developer. Bots trying to access confidential information might also encounter these errors.)
8. Multiple user IDs having same user agent string(To indicate fake account creation)

The idea behind these features is as follows:

1. A particular user account sending requests from multiple locations will indicate possible credential stuffing and account takeover.
2. Similar reasoning will apply to the number of IP addresses as well. However, even non abuse (negative) cases can use multiple IPs. Hence, a possibly large number of IPs associated with an account could indicate credential stuffing.
3. Bot content scrapers often send a large number of requests and hence the feature for average rate of requests and number of requests per user.
4. Bots scraping the internet will typically use a large number API endpoints.
5. API Misuse cases i.e developers committing mistakes while trying to access the API will typically have a large number of failed status codes.
6. Multiple user IDs being created by the same user agent detects creation of fake accounts as, usually, multiple accounts are created by the same malicious user.

ML Models

We use classification- one of the machine learning methods. Classifiers could be implemented using both supervised and unsupervised learning algorithms. We will be implementing supervised classifiers which means that they need to be trained with labeled data before making predictions. We can use decision tree classifier, logistic regression,etc.

Feature based Classification method is a popular approach to detect bots and other malicious users. It scales well to large OSNs and gives good performance. Graph based techniques also are commonly employed to detect fake users, however feature based methods with profile attributes allow for early detection. There are many classification algorithms to iterate on and tune performance.

Instead of relying on one model, we intend to test the data on a couple of models:

1. K-Means
2. Hierarchical Clustering

Depending on the size of access log data available, we will consider manual/programmatic labelling on data.

1. Labeling will be done looking at common malicious patterns in URI

If labelling is feasible, we intend to run the following supervised learning models:

1. Support Vector Machines
2. Random Forests

Programming Language of choice : Python
Libraries: SciKit Learn, MatPlotLib, Numpy, Pandas, Seaborn

At last, we will be testing and comparing the precision from the ML model.