Automated Jira Ticketing System

Prepared By: Rafsan Saadi
Date: 08/20/2019

Problem Definition

Security Vulnerabilities are event driven should be persisted in JIRA. Rest API integration and maintaining data quality and integrity is a key component of the Development Security Platform.

Demo Requirements

Create functional POC/wireframe using AWS Lambda/Python serverless application that can be used to create JIRAs once a certain monitored condition is met (e.g. Temperature obtained from weather underground RestAPI).

Suggest/implement additional features that could be considered.

Present and outline the solution key points, customer benefits, and security considerations.

Notes:

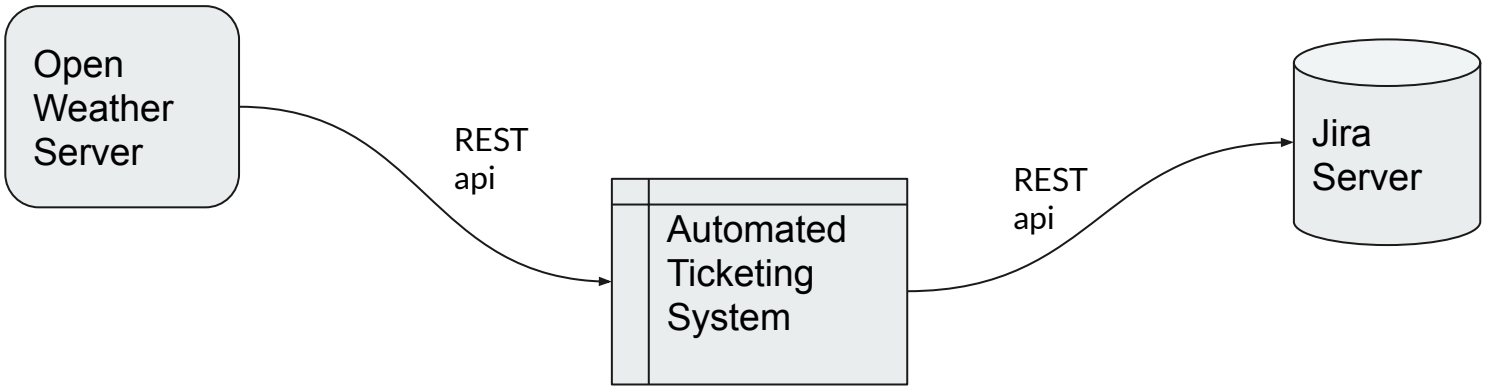
Use the free JIRA test instance or local JIRA development.

Assumptions

1. Major components are running either in the same or different(ideal case) server/network space
2. This demo project can be implemented as a Python serverless application
3. The Goal is to create a simple wireframe project to demo a working business use case
4. Monitor condition data can be collected from any REST data source

Note: No more free Rest API available for weather underground service

System Components



Design Components

1. Jira Ticketing Client
2. Open weather REST API Client
3. Monitoring and Ticketing Application

Jira Client

This class implements a JIRA client using Python ‘jira’ SDK

```
def __init__(self, **kwargs):

def create_new_issue(self, fields=None, **fieldargs):

def get_issue_by_id(self, issueID):

def get_all_issues_assigned_to_current_user(self):
```

Open Weather API Client

This class implements a Open Weather REST API client

```
def __init__(self):

def url_constructor(self, city_name=None, city_zip_code=None):

def make_weather_api_request(self, target_url):

def get_current_temperature(self, json_response):

def get_current_weather_report(self, json_response):
```

Monitoring and Ticketing Application

```
# Enable automated monitoring for temperature and create JIRA ticket as needed
i = 1
while(True):
    weather_json = myWeather.make_weather_api_request(weather_api_url)
    current_temp = myWeather.get_current_temperature(weather_json)
    print("{} temperature is {}".format(target_city, current_temp))
    issue_dict = {
        'project': {'key': "CCI"},
        'summary': "[ATTENTION] {} current temperature is changed to {}F".format(target_city, current_temp),
        'description': "Temperature for {} has changed to {}F. It requires urgent attention.".format(target_city,
                                                                                                     current_temp),
        'issuetype': {'name': "Bug"}
    }
    # Monitoring condition to create JIRA ticket automatically
    if current_temp < 65.46 or current_temp > 80:
        myJira = JiraHandler(username='rafsan.saadi', password='Pass!23')
        myJira.create_new_issue(fields=issue_dict)
    else:
        print("No JIRA ticket is created for this time")

    if i == 2:
        break
    else:
        i += 1
    print("Waiting for 30s before checking temperature again")
    time.sleep(30)
```

Implementation Details

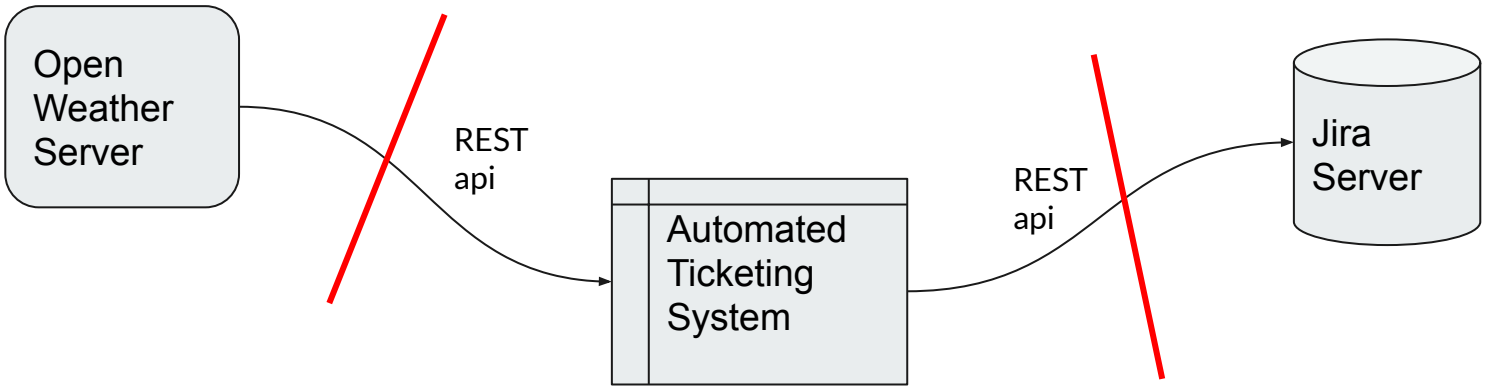
Let’s see the code for more details…

Reference Link:

Open Weather: *https://openweathermap.org/current*

Jira SDK: *https://jira.readthedocs.io/en/latest/index.html*

System Security Data Flow



Security Considerations

Primary Security Consideration

1. Collecting weather data from the Open Weather server
2. Submitting Open Weather Data as part of the Jira Ticket creation in JIRA
3. Implementation and data handling of the Automated Jira Ticketing client

Data From Open Weather

1. Data should be collected using the secure HTTP connection
2. Preferably use POST instead of GET
3. Data should be sanitized/normalized before use it in any other application

Data Submitted to JIRA

1. Data should be submitted to JIRA using the secure HTTP connection
2. Preferably use POST instead of GET
3. Ticket data details should be sanitized/normalized and submitted to Jira

Automated JIRA Ticketing Client

1. All the connection/communication should be made over the secure HTTP connection
2. Should normalize/sanitize any data that collected from any outside data source (Open Weather)
3. Locally stored data should be stored securely using PRIVATE data member
4. Should provide getter()/setter() method to manipulate private data
5. No user credentials and Keys should be hardcoded in the application
6. Error message and system log message should not giveaway user secret data
7. A server like data validation should be performed on the client application(i.e. Username/Password Policy/requirements, required data, special data escaping, and so on)

Issues of the Ticketing Client Implementation

1. User credentials and API Tokens are hardcoded in the Program
2. User credentials are stored in PUBLIC data field instead of the PRIVATE data field
3. No getter()/setter() function defined to manipulate private data field
4. Used BasicAuth to authenticate to JIRA
5. There is no client-side validation implemented
    a. There is no user credentials policy check implemented
    b. Data are not sanitized/normalized before submitting for JIRA ticket creation
6. The client is relying on the JIRA server for all data validation
    a. User credentials and Jira ticket data validation
    b. The actual content of submitted data to create ticket details

Customer Benefit

1. Can be integrated into CI pipeline as part of the SSDLC
2. System security vulnerabilities are being identified and tracked without human intervention
3. It helps to reduce business operating cost to monitor security vulnerabilities in the System
4. Fixing security vulnerabilities task can be included and tracked as part of the regular software development process
5. Can provide standardized system security best practice for a wide variety of projects/products

Thank You !