AYSOME Security Information and Event Management (SIEM) system.

**Table of Contents**

[​ First steps 2](#__RefHeading___Toc109_2420582686)

[​ The project vision and description 2](#__RefHeading___Toc111_2420582686)

[​ Architecture 2](#__RefHeading___Toc113_2420582686)

[​ Setup 3](#__RefHeading___Toc115_2420582686)

[​ Processes 3](#__RefHeading___Toc117_2420582686)

[​ UCR\_CORE\_WORKFLOW 3](#__RefHeading___Toc119_2420582686)

[​ UCR\_TESTING\_WORKFLOW 4](#__RefHeading___Toc142_2102129628)

[​ Code 4](#__RefHeading___Toc121_2420582686)

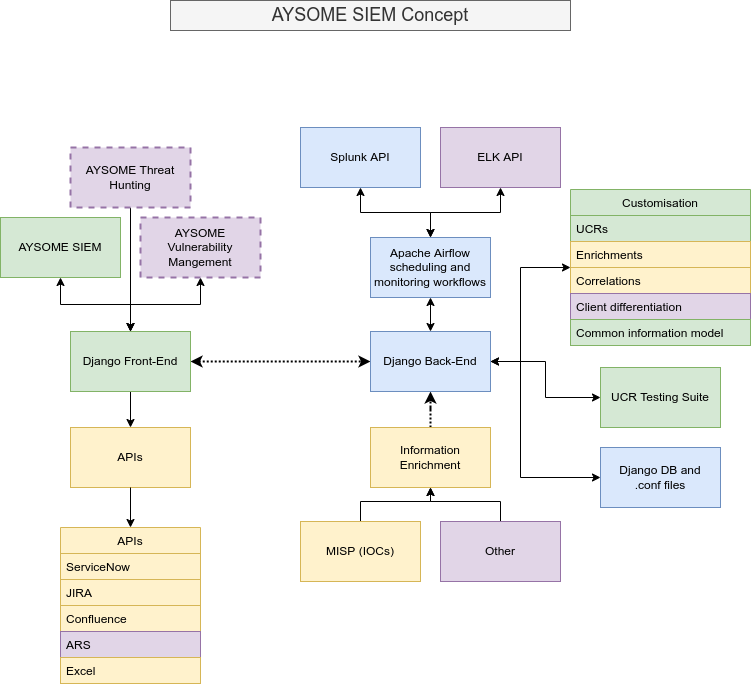
# First steps

## The project vision and description

AYSOME SIEM system is an open source project which aims to become a comprehensive IT security solution for log monitoring and incident handling. It is build upon other open source products and can be used freely. The project is managed by AYSOME IT Security GmbH from Düsseldorf, Germany. Feel free to add functionality and raise questions on GitHub or send us an Email.

The solution comprises several modules, which can be used separately. Following is the current state of the vision for this solution.

## Architecture

Currently only the blue part of the solution is implemented. It is based on:

* Django Back-End, where the data about groups, use case rules (UCRs), enrichments, correlations and alarms can be found.
* Splunk API configuration files
* Apache Airflow server, which is responsible for scheduling and monitoring workflows
* .conf files, which are used for a quick and simple importing or migrating new UCR, group, enrichment, correlation data into the database
* UCR Testing Suite

Further development will provide even more functionality. The plan is to implement blocks in the following sequence: green → yellow → purple blocks, as it can be seen on the diagram.

## Setup

Minimal independent installation is based upon following packages. More about installation process of every package can be found online.

* Django Back-End (django)  
  https://www.djangoproject.com/download/
* Apache Airflow  
  https://airflow.apache.org/docs/apache-airflow/stable/installation/index.html
* Splunk (splunk-sdk)  
  https://dev.splunk.com/enterprise/docs/devtools/python/sdk-python/gettingstartedpython/installsdkpython/

After installation a setup process can be started. Django has a few changes in the native settings.py file. Currently most important are INSTALLED\_APPS and TIME\_ZONE variables, which have to be set accorlingly.

Apache Airflow works out of the box. The only change in the airflow.cfg file is a path variable to the dag folder: dags\_folder = .../aysome.siem/siem/backend/dags. Please provide a full path to the applications dags folder. Otherwise the Airflow engine will not import dag configurations and would not schedule or monitor any workflows.

Splunk installation has to have Splunk Management port 8089 open and available to the application. Additionally a configuration file can be found under siem/backend/api/api\_settings.py, where different variables are defined to successfully initiate a connection. The value for SPLUNK\_API\_TOKEN variable can be found in Splunk administration panel under Navigation Menu→ Account Settings → Generate User API Access Token.

# Processes

To understand the concept of the application, one should clearly understand the processes and workflows behind it. Currently main processes are managed by Apache Airflow engine, which is responsible for monitoring, scheduling and error handling of the underlying code.

The workflows in Apache Airflow can be found in DAGs: “A DAG (Directed Acyclic Graph) is the core concept of Airflow, collecting Tasks together, organized with dependencies and relationships to say how they should run.”

DAG files can be found under .../siem/backend/dags folder.

Currently following workflows are defined:

* UCR\_CORE\_WORKFLOW
* UCR\_TESTING\_WORKFLOW

## UCR\_CORE\_WORKFLOW

UCR\_CORE\_WORKFLOW is a core workflow, which is responsible for searching and creating alarms. It has four stages:

* Stage one is called with search\_for\_notables.expand(ucr\_info = call\_for\_current\_ucrs()). The inner function call\_for\_current\_ucrs() returns a dictionary of UCRS with search queries, earliest and latest time points for the current run. The outer function search\_for\_notables() make an API calls and returns results, based on search queries from inner function. Stage one passes a list of a\_ids for each notable to the next stage.
* Stage two and stage tree are called with enrich\_current\_notables() and correlate\_current\_notables() functions, which call for every available enrichment and correlation and try for find additional information for every a\_id from the last stage.
* Stage four is called with create\_alarms\_from\_notables(), where alarms are created and passed to the Django DB.

## UCR\_TESTING\_WORKFLOW

UCR\_TESTING\_WORKFLOW is a testing workflow for use case rules. The main objective of the workflow is to test the whole process of the UCR\_CORE\_WORKFLOW. Therefore a testing notable event is created an send to the API to imitate a finding. If the UCR\_CORE\_WORKFLOW works as expected, then the finding will be found and processed as expected.

# Code

AYSOME SIEM application is written in Python. Therefore it is easy accessible and simple to follow. Nevertheless a few key functions and classes will be described here to deepen the understanding of the key concepts of the application.