Вам на тестирование попал новый шаблон для детектирования уязвимостей. Ваша задача — понять, что делает этот шаблон.

```yaml
id: airflow-api-default-login

info:
  name: Apache Airflow API - Default Logins
  author: Pavel Parkhomets
  severity: critical
  tags: api,airflow,default-login,brute-force

requests:
  - method: GET
    path:
      - "{{BaseURL}}/api/v1/dags"
    headers:
      Authorization: "Basic {{base64(username + ':' + password)}}"
      Content-Type: application/json
    payloads:
      username:
        - "airflow"
        - "admin"
      password:
        - "airflow"
        - "admin"
    attack: clusterbomb
    matchers-condition: and
    matchers:
      - type: word
        words:
          - "dag_id"
      - type: word
        words:
          - "kafka_server_socketservermetrics_successful_reauthentication_rate"
        part: body
        negative: true
    stop-at-first-match: true
```

Данный шаблон нужен для проверки того, что при аутентификации через Apache Airflow Api (с помощью Basic аутентификации) не используются стандартные имена пользователя и пароли (которые заданы в шаблоне) - всего 4 возможных комбинации. При этом ответ (в теле ответа) в случае успешной аутентификации не должен содержать некоторую строку, указывающую на метрику, связанную с kafka.

Давайте разберем данный шаблон построчно (пропуская пустые строки):

Строка 1. Здесь задается уникальный идентификатор (airflow-api-default-login) для шаблона (используется Nuclei для ссылки на него)
Строки 3-7. Это секция info с информацией о шаблоне.
Строка 4. Имя шаблона - в данном случае это "Apache Airflow API - Default Logins"
Строка 5. Имя автора - в данном случае это "Pavel Parkhomets"
Строка 6. Уровень критичности уязвимости, в данном случае "critical", что означает, что уязвимость может привести к серьезным последствиям.

Строка 7. Теги, которые помогают классифицировать шаблон. Здесь указаны теги api, airflow, default-login, и brute-force (по факту тег brute-force лишний, т.к. сложно назвать полноценным брутфорсом попытку подбора из 4 всего значений).
Строки 9-34. Это секция requests с описанием запросов, отправляемых на целевой сервер и критериев успешности запросов на основании данных из ответов. Секция requests содержит список запросов и относящимся к ним критериев; в нашем случае, запрос всего один
Строка 10. HTTP метод используемый для отправки запроса - в данном случае GET
Строки 11-12. Это секция path, в которой описываются пути, по которым будет отправляться запрос. В нашем случае это один путь - "{{BaseURL}}/api/v1/dags". Здесь {{BaseURL}} — это переменная, которая будет заменена на целевой URL во время выполнения.
Строки 13-15. Это секция headers, в которой определяются заголовки, передаваемые в HTTP-запросе.
Строка 14. Заголовок Authorization, в котором передаются реквизиты для аутентификации пользователя. В данном случае используется Basic аутентификация; при этом имя пользователя и пароль передаются практически в открытом виде (в виде base64 закодированной строки, формируемой следующим образом “username + ':' + password”).
Строка 15. Заголовок Content-Type содержит описание (тип) содержимого запроса. В нашем случае - это значение "application/json", что означает, что мы передаем данные в JSON формате.
Строки 16-22. Это секция payload с определением возможных данных для полей аутентификации.
Строки 17-19. Здесь определяются возможные данные для поля “username” - в нашем случае это два значения "airflow" и "admin".
Строки 20-22. Здесь определяются возможные данные для поля “password” - в нашем случае это два значения "airflow" и "admin".
Строка 23. Здесь указывается тип атаки. В данном случае используется метод "clusterbomb", который позволяет комбинировать все возможные значения из списков “username” (строки 17-19) и “password” (строки 20-22), создавая все возможные комбинации для тестирования (всего 4 комбинации для полей “username” и “password”).
Строка 24. Здесь указывается как комбинируются matcher-ы, если их определено несколько. В нашем случае, это “and”, что означает, что все условия (в каждом matcher) должны быть выполнены, чтобы ответ на данный запрос считался подходящим (операция логическое И).
Строки 25-33. Это секция matchers, которая содержит условия (matcher), проверяющие ответ на запрос. В нашем случае определено всего 2 условия (matcher).
Строки 26-28. Это определение первого условия (matcher), которое проверяет ответ. Параметр “type” означает, что будет проверяться наличие слов в ответе, список “words” содержит слова, которые должны быть в ответе (в нашем случае - это слово “dag_id”). Другими словами, первое условие (matcher) выполнится, если ответ на HTTP запрос будет содержать слово “dag_id”.
Строки 29-33. Это определение второго условия (matcher), которое проверяет ответ. Параметр “type” означает, что будет проверяться наличие слов в ответе, список “words” содержит слова, которые должны быть в ответе (в нашем случае - это слово “kafkaserversocketservermetricssuccessfulreauthentication_rate”), параметр “part” определяет, где будет происходить поиск (в нашем случае это значение “body”, т.е. тело ответа), параметр “negative” со значением “true” означает, что мы ожидаем отсутствия искомых слов в ответе. Другими словами, второе условие (matcher) выполнится, если ответ на HTTP запрос не будет содержать слово “kafkaserversocketservermetricssuccessfulreauthentication_rate” в теле ответа.
Строка 34. Здесь указан параметр “stop-at-first-match” со значением “true”. Это означает, что Nuclei должен прекратить поиск после первого выполнения условия (комбинации первого и второго условий с помощью операции логическое И).