

H To-do App.

Haitham Alhaji

Std. ID 1801886

* **Pages:**
  + Home
  + Login
  + Register
  + Admin. Panel (only for admin account)
  + MyToDo
* **Features**:
  + Session Mana.
  + Database MySQL (Storing ToDo list)
  + User types (isolation permissions)
  + Securing from SQL Injection (with escape methodology)
  + Securing from XSS (with form validation)
  + Securing GLOBAL Variables with Environment Variables
  + Securing Access Control
  + Cross-Site Request Forgery (CSRF)
  + Apache Virtual Host
  + Open Policy Agent
* **Technologies**:
  + NodeJS with Express Server
  + MySQL Database
  + My own CentOS 7 Server with Apache Server
  + HTML5 With CSS Enhancement (SASS)
  + Handlebars (code Segmentation)
* Links
  + Website (live demo): <http://todo.haitham.xyz/>
    - Admin. Info. (Email: [g@y.com](mailto:g@y.com), password: g@y.com)
  + Source code: <https://github.com/HaithamAlhaji/WebHToDo>

**Securing from SQL Injection (with escape methodology)**

Direct variable in SQL Query will make attacker implement his injection directly, therefore, there is no doubt to use escape method to clear your variable from any kinds of injection. For example:

1. Home / Login (POST)



1. Home / Register (POST)



1. Home / AddTask (POST)



1. Home / Search for task (POST)



1. Home / List tasks (POST)



**Securing from XSS (with form validation)**

Although Inputs are a compulsory channel that is letting the user communicate with our app., Strict entering inputs are mandatory. Therefore, form validation is a technique which plays a vital role to secure that has mentioned before. For example:

1. Login (HTML)
2. Login (JS)
3. Regsiter (HTML)
4. Regsiter (JS)
5. AddTask (HTML)
6. AddTask (JS)
7. Search (HTML)
8. Search (JS)

**Missing Function Level Access Control**

Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified for access rights on server, attackers can forge requests in order to access functionality without proper authorization. Most web applications don’t display links and buttons to unauthorized functions, but this “presentation layer access control” doesn't actually provide protection. You must also implement checks in the controller or business logic.

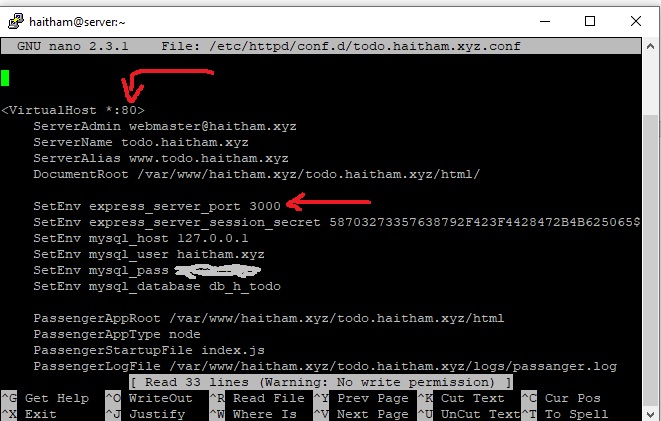
1. /mytodo (JS)
2. /mytodo (HTML)
3. /admin (JS)
4. /admin (HTML)

**Cross-Site Request Forgery (CSRF)**

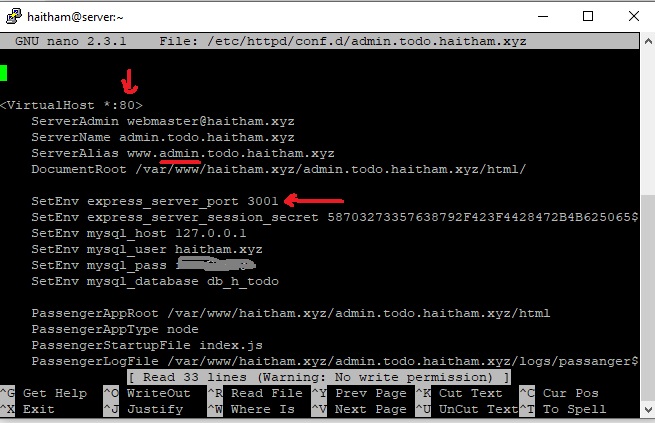
A CSRF attack forces a logged-on victim’s browser to send a forged HTTP request, including the victim’s session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim’s browser to generate requests that the vulnerable application processes as legitimate requests from the victim. As browsers automatically send credentials like session cookies with HTTP requests to the server where cookies were received from, attackers can create malicious web pages which generate forged requests that are indistinguishable from legitimate ones.

1. /Register (JS)
2. /Register (HTML)
3. /Login (JS)
4. /Login (HTML)

**Apache Virtual Host**

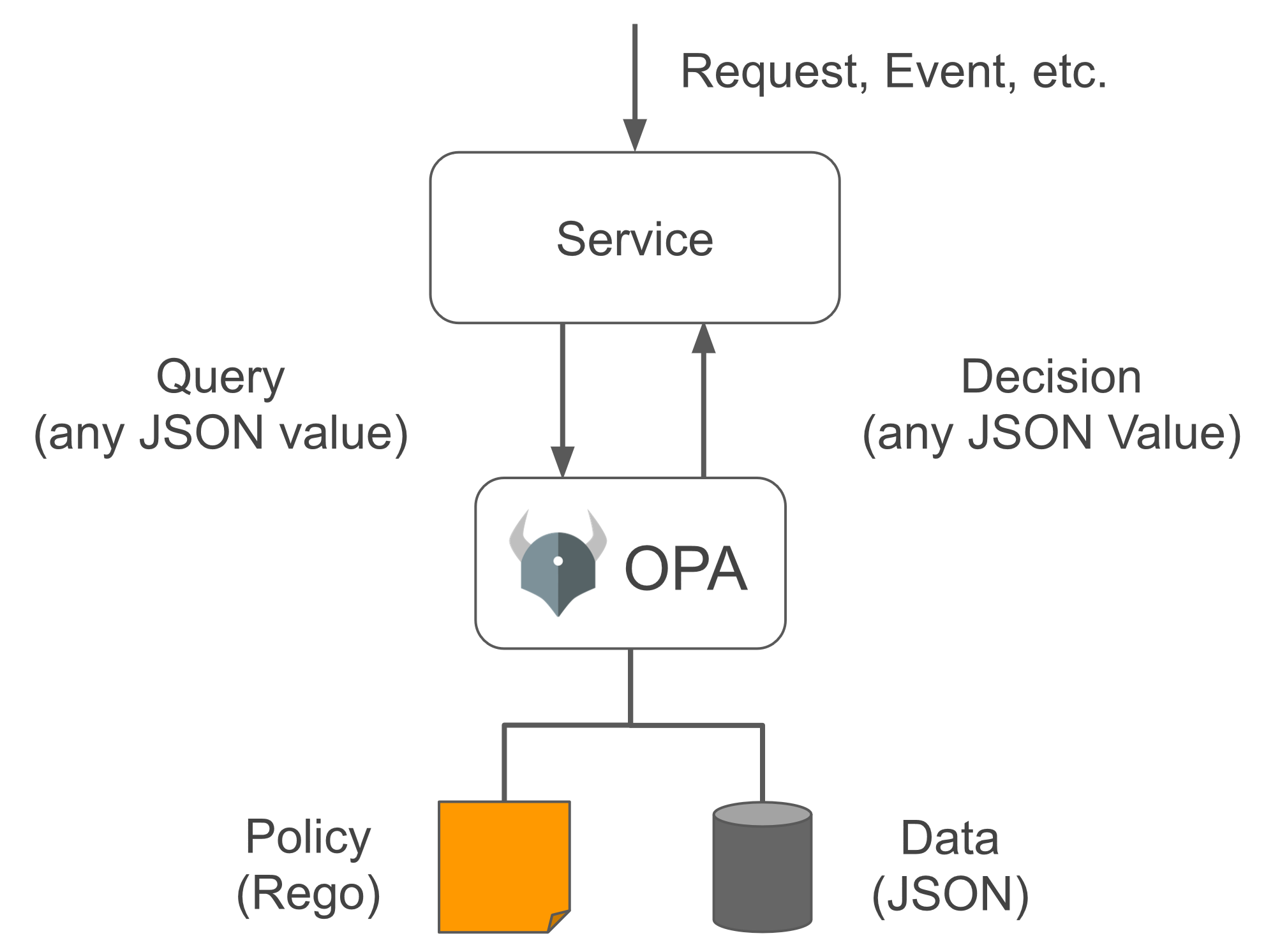
Virtual hosting is a method for hosting multiple websites on a single machine. There are two types of virtual hosting: Name-based virtual hosting and IP-based virtual hosting. IP-based virtual hosting is a technique to apply different directives based on the IP address and port a request is received on. You can assign a separate IP for each website on a single server using IP-based virtual hosting. This is mainly used to host different websites on different ports or IP addresses. ****

Admin panel, it is also has specific subdomain with special internal port.

****

**Open Policy Agent**

OPA provides a high-level declarative language that let’s you specify policy as code and simple APIs to offload policy decision-making from your software.

****

Exampleof Rego file, there are three files, first one is input.json which consists example of input data that will receive from connection. Second file, example.rego refers to rules of connection that will apply it on input.json. finally, output.json the result of evaluation. 