Using vulnerability configuration editor

All Hackazon vulnerabilities are managed in configuration editor. The editor allows to control contexts, fields and vulnerabilities very flexible.

Let's review the main keywords.

Vulnerability context is an entity which describes a certain area in code (more exactly - describes fields and vulnerabilities). For example, **default** context describes the entire application, the **faq** context comprises the faq controller, and so on.

Context can contain child contexts, for example the **default** context contain all contexts in the application, including controller contexts and services (REST, GWT, AMF etc.) contexts. **faq** context can contain **index** context for the **index** action.

Each context may have a field collection. A field is a vulnerability container for a request field with certain name and certain source (query string, cookie, etc.).

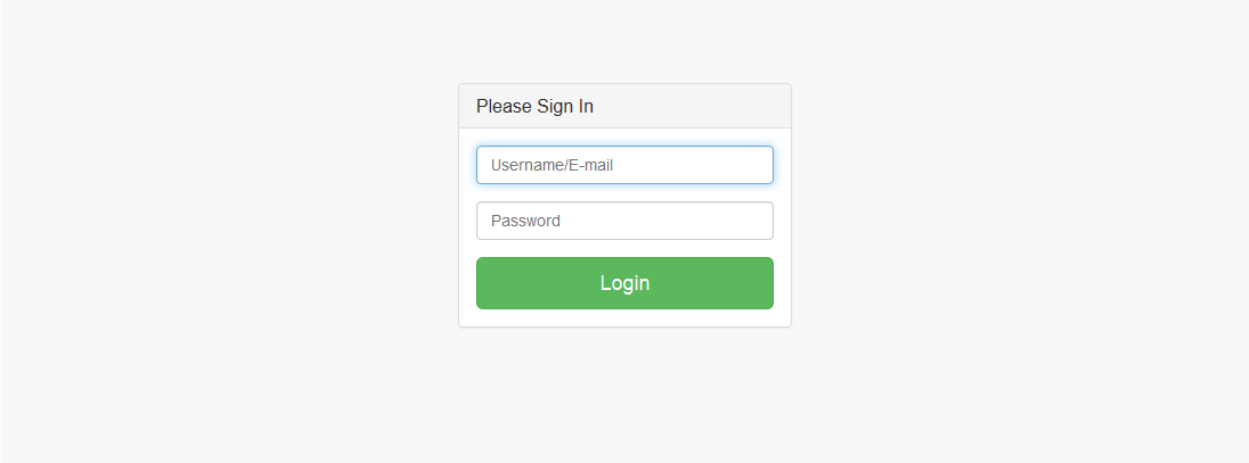
Contexts and fields are vulnerability hosts, that is they may have bound vulnerabilities.

Vulnerabilities are groupped into sets. A set contains vulnerabilities, and, possibly, child sets. Also, conditional vulnerability sets may contain conditions for matching with the request (whether or not the request is AJAX, etc.). Nesting conditional sets allows to add conditional vulnerabilities.

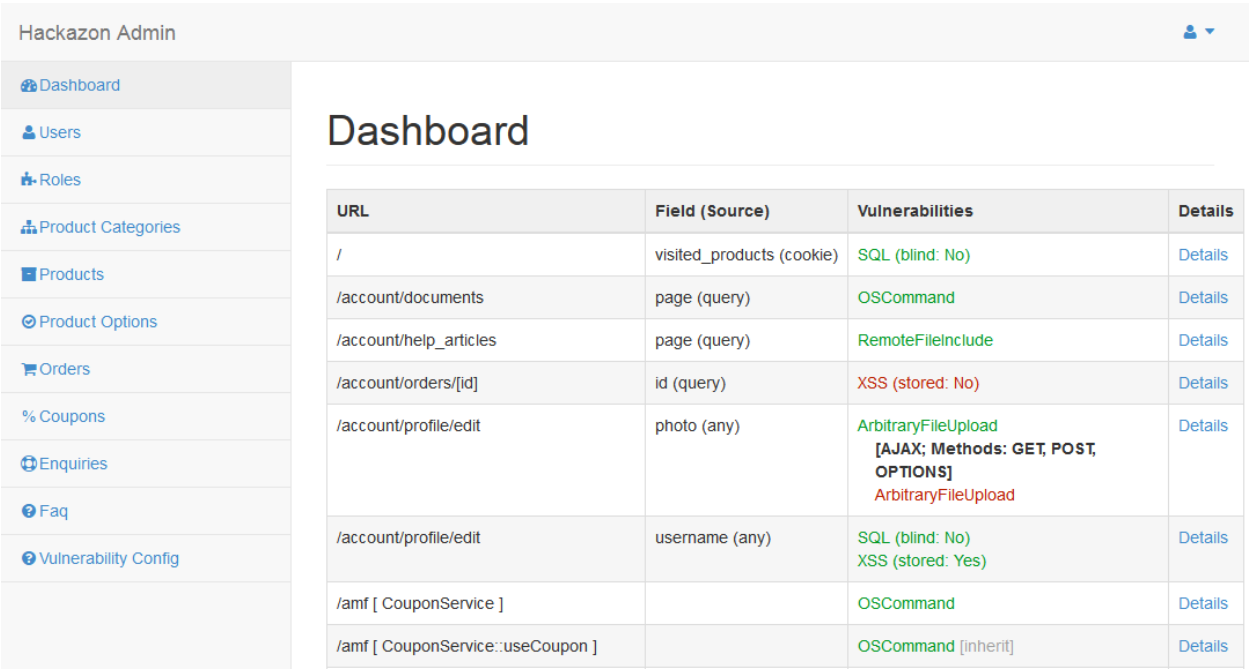
These are the basic terms, and now let's see how to operate them in the editor.

1. Open the editor.

Go to the admin panel: **`/admin`**

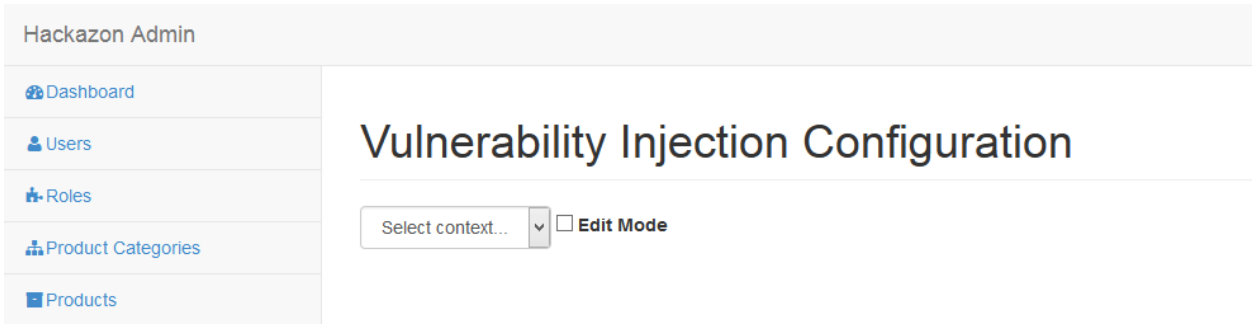


Enter your credentials and log in:



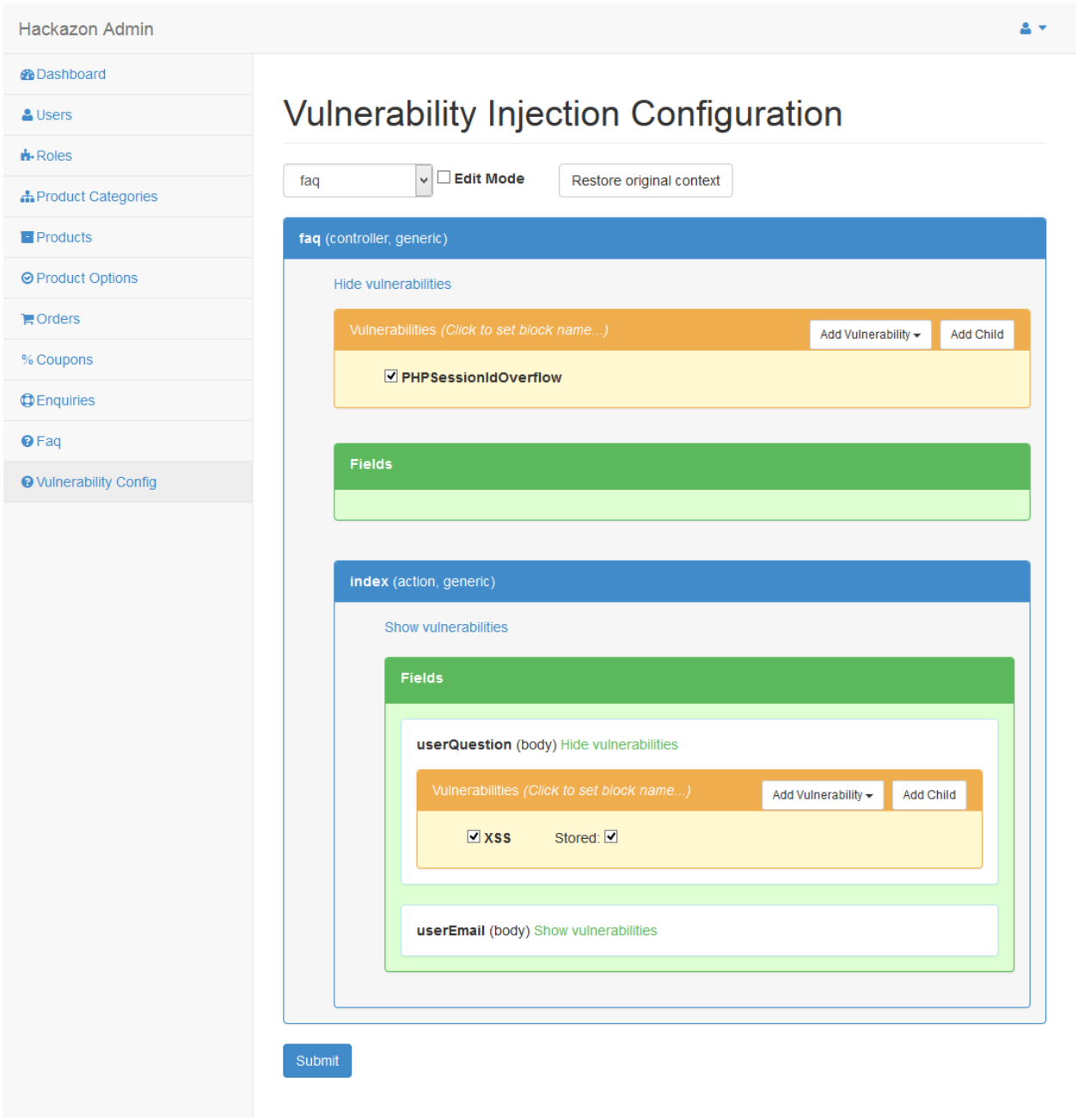
Here you see a vulnerability matrix. All enabled vulnerabilities and explicitly disabled ones.

Click “Vulnerability config” link on the left sidebar:



And select the config you want to edit.

For example **faq**.



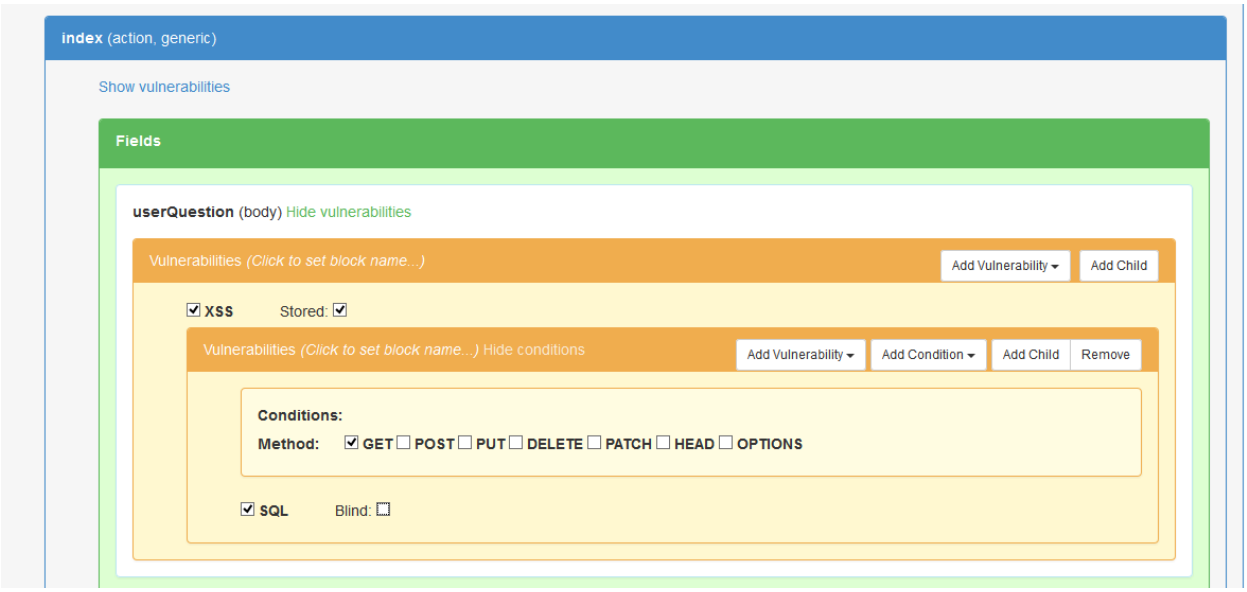
Here we see the **faq** context and its elements:

1. Contexts own vulnerability set.
2. Field set.
3. And child context **index**.

Empty vulnerability blocks are hidden to save some screen space. You can see it on userEmail field.
To add a vulnerability simply press Add Vulnerability button and select the one you want.

Please keep in mind that although you can select any vulnerability in the list, the majority is special case vulns, e.g. they are used only in some places in code and in other contexts won't mean much. Nevertheless such vulns as SQL and XSS are widely used and can be bound to many fields and contexts.

It's also possible to add conditional children to vuln sets. Just click the **Add child** button and set up vulnerabilities and conditions you want. For example, you want to add a SQL injection on **userQuestion** field in **faq/index** context only when GET request method is used. Just add a child to vulnerability block, add a method condition to it, and set up a SQL injection:

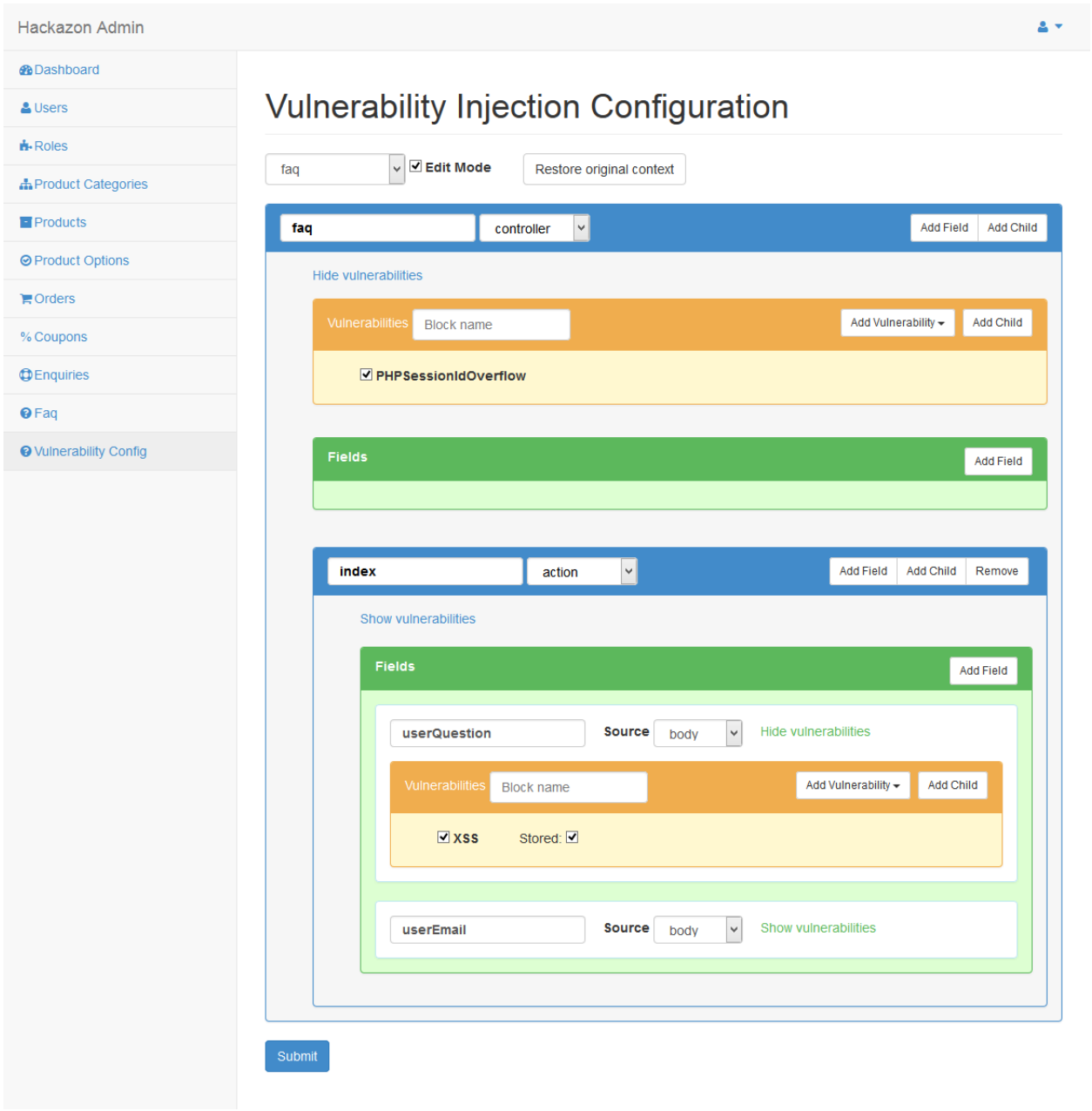


Now the vulnerability subsystem will match the first valid set of vulnerabilities and use it. In our case the first set of vulnerabilities is without conditions, so subsystem searches for a first matching child. And if the request method is GET our child vulnerability set matches, and SQL injection becomes enabled.

In most cases we only want to add/remove or enable/disable vulnerabilities to already defined contexts and fields. But sometimes that is not enough. In such cases the **Edit mode** helps us.

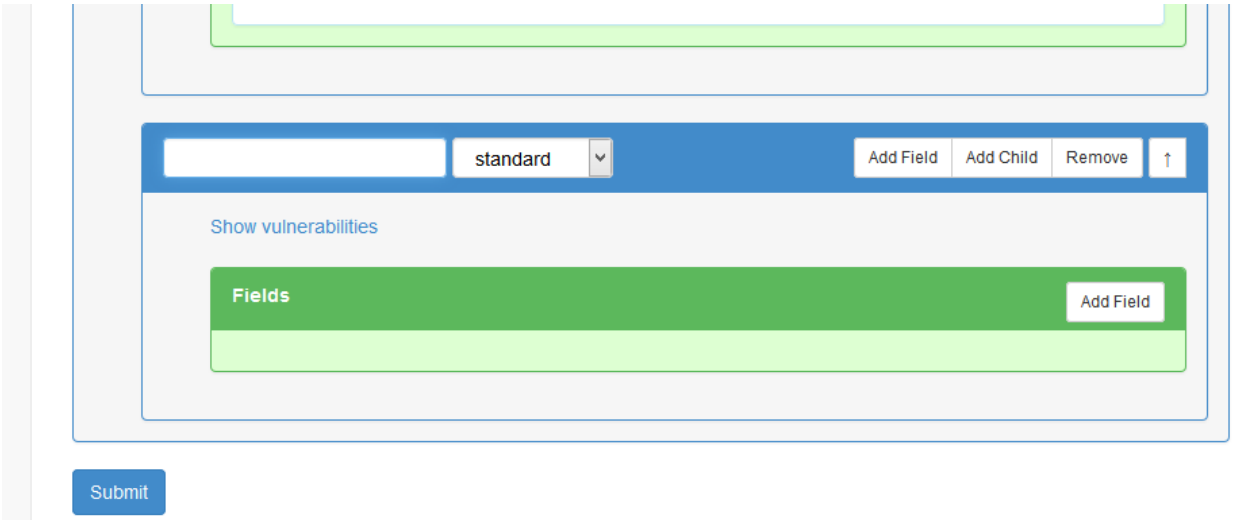
2. Edit mode

To switch to edit mode enable the **Edit mode** checkbox. You'll see the similar picture:

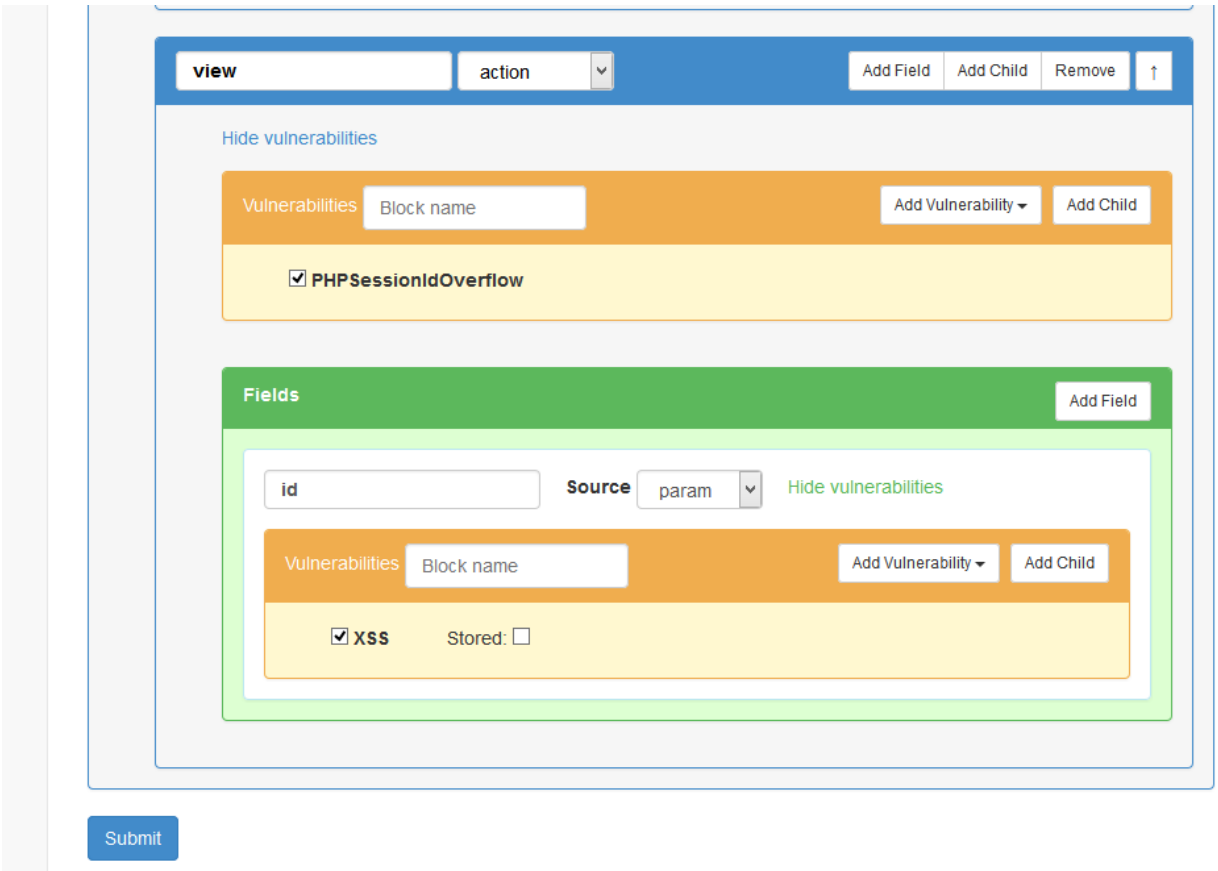


The edit mode allows you to modify not only vulnerabilities, but also the structure of the config.
That is you can add/remove/modify contexts and fields, reorder child items. These features are mainly needed when you extend the controllers. That's why they're disabled by default, so you can't spoil the stucture.

For example, you decide to add the **view** action on **faq** controller to see individual entries. You modify the code of hackazon, and then edit the corresponding config. On the root **faq** context click the **Add child** button. An empty context will be added at the bottom:



Enter the name, in our case it is **view** (it corresponds to **action_view** method in the faq controller). And select the **action** value in the select box (this influences the url generator in the vulnerability matrix). And now you can add fields, vulnerabilities and child contexts (for special cases you want to implement in the code).



Context type can be one of the following:

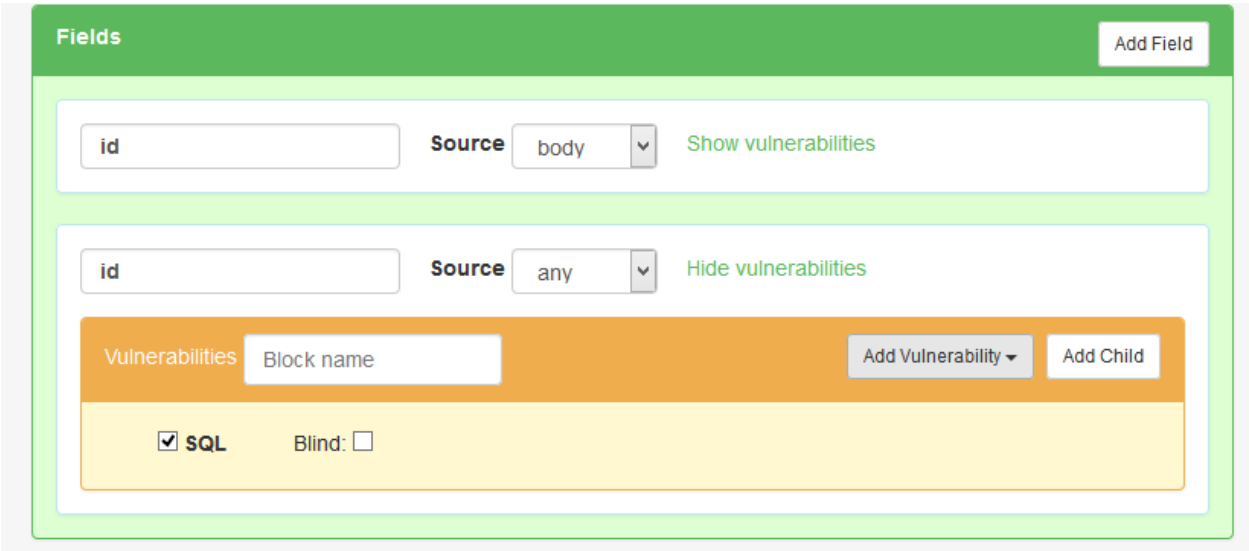
- standard (default value. does not infer special meaning)
- application (root contexts for some technology - Web, GWT, etc.)
- controller (also for web services controllers)
- action (also for web services methods)

Field sources are:

- any
- query (query string parameter from url)
- param (route parameter from a path, e.g. **id** in /product/view/**<id>**)
- body (parameter from request body, e.g. from POST request, or JSON body, etc.)
- cookie
- header

Vulnerability block name (input field at the header of the orange panel) is used to get fully-qualified path to vulnerability, when you want to use it in long-running vulnerabilities, for example stored XSS, or when you want to delay vulnerability execution by serializing vulnerable fields (like Cart service in hackazon). In most cases it's not used, but sometimes it really helps to solve the problem. If the name is absent, the element index is used as the name (so the order matters).

Also the order matters for fields. The first field that matches the requirement is used. For example, look at this picture:



Three cases possible:

- If we are looking for a field with name id from any source, wi will get the first field, and there will be no vulnerabilities.
- If we are looking for id field from body source, the result will be the same

- But if we want to get the id field from query source, wi will get the second entry, and it’s vulnerable.

Now imagine that we swap these fields. In this case we will always get the id field from source **any**, so it will always be vulnerable. Therefore the order matters. These rules also apply to conditional vulnerable blocks.

When you’ve finished editing the config just press the Submit button, and that’s it, new rules are now running.

3. Restoring the config.

If you spoil the config, or at least want to start from scratch - just click the **Restore original context** button at the top of the page. The config will be replaced by reference one.