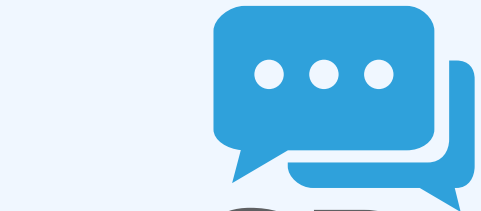
Automation with MISP Workflows

A new way to support your CTI pipelines

Sami Mokaddem

MISP Project
https://www.misp-project.org/

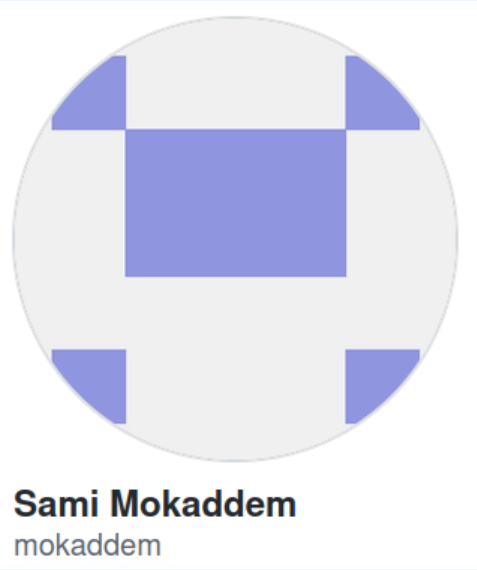
CTIS 2022



CTIS-2022

$ whoami

- Working @ circl.lu
- Part of the MISP-Project team
- Adding easter eggs for the past 4 years



What problems are we trying to tackle?

- **Prevent** default MISP behaviors to happen
- **Hook** specific actions to run callbacks
- Use-cases:
  - Prevent publication of events not passing sanity checks
  - Prevent querying thrid-party services with sensitive information
  - Send notifications in a chat rooms
  - And much much more...

What already exists in MISP?

MISP API / PyMISP

- Needs CRON Jobs in place
- Heavy for the server
- Not realtime

PubSub channels

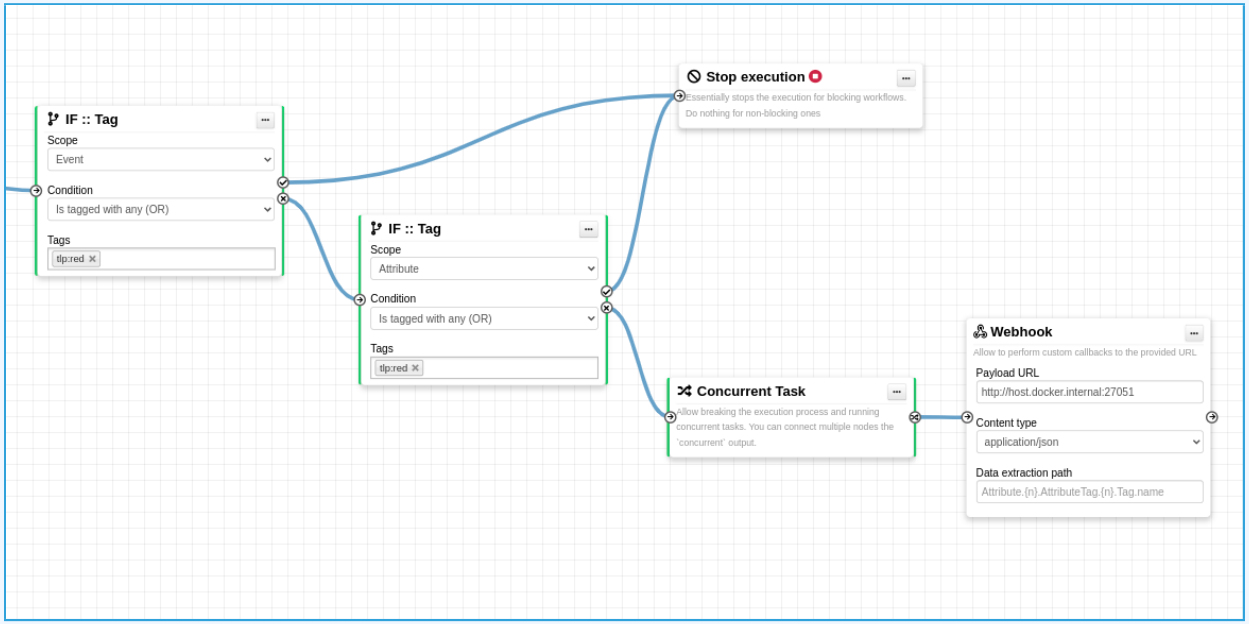
- After the actions happen: No feedback to MISP
- Tougher to put in place & to share
- Full integration amounts to develop a new tool

Simple automation made easy

- Why?
  - Everyone loves **simple automation**
  - **Visual** dataflow programming
  - Users want **more control**
- How?
  - **Drag & Drop** editor
  - Prevent actions **before they happen**
  - Flexible **Plug & Play** system
  - **Share** workflows, **debug** and **replay**

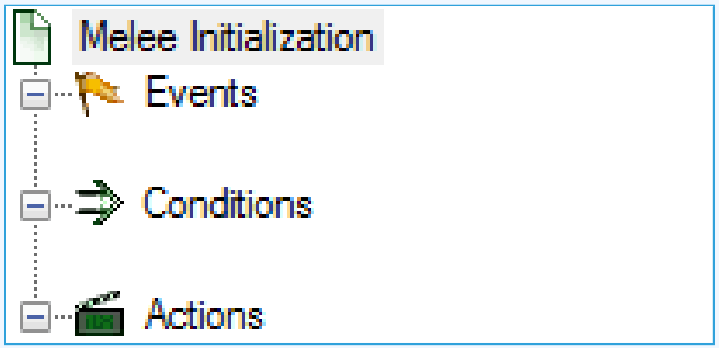
Content of the presentation

- MISP Workflows fundamentals
- Demo by examples
- Get started
- Using the system & how it can be extended



Workflow - Fundamentals

How does it work



1. An **event** happens in MISP
2. Check if all **conditions** are satisfied
3. Execute all **actions**
   - May prevent MISP to complete its original event

What kind of events?

Events

- New MISP Event
- Attribute has been saved
- New discussion post
- New user created
- Query against third-party services
- ...

In MISP Workflow terminology, supported events are called **Triggers**

What kind of conditions?

- An MISP Event is tagged with `tlp:red`
- The distribution an Attribute is a sharing group
- The creator organisation is `circl.lu`
- Or any other **generic** conditions

In MISP Workflow terminology, these are also called **Logic modules**

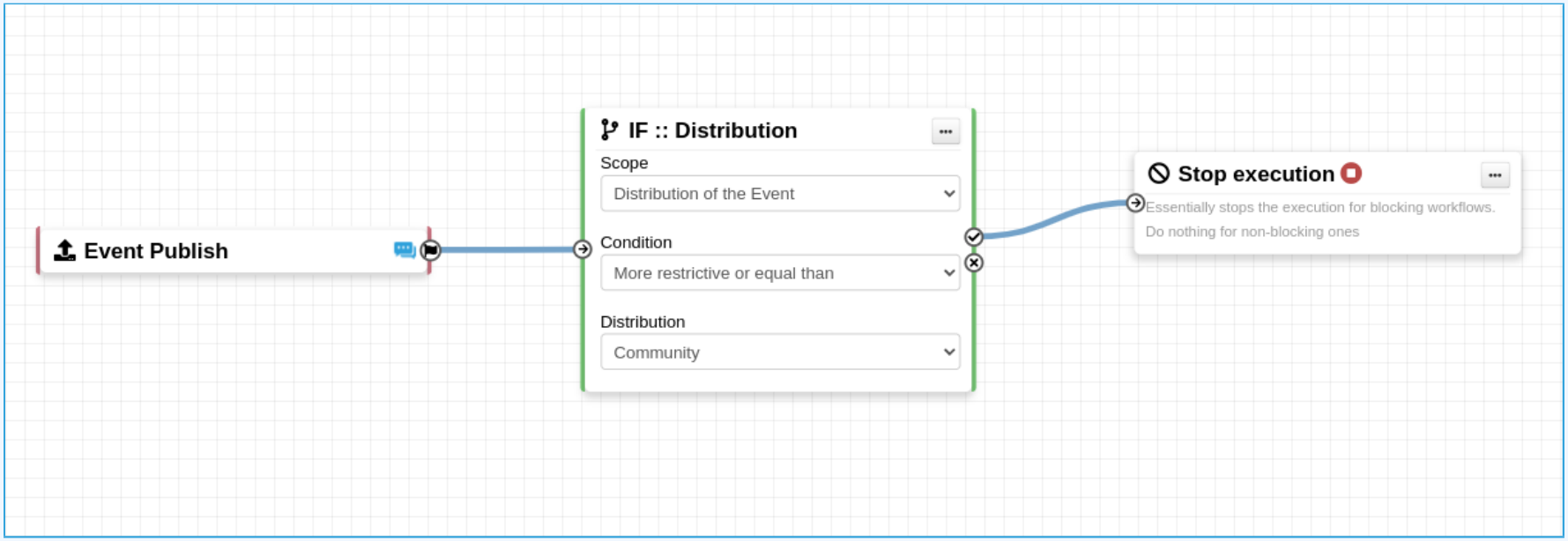
What kind of actions?

- Send an email notification
- Perform enrichments
- Send a chat message on MS Teams
- Attach a local tag
- ...

In MISP Workflow terminology, these are also called **Action modules**

What is a MISP Workflow?

- Sequence of all nodes to be executed in the specified order
- Basically the whole connected graph.
- Workflows can be enabled / disabled
- Workflows are always linked to a **trigger**



Workflow execution for Event publish

An Event is about to be published

- The workflow for the `event-publish` trigger starts

Conditions are evaluated

Actions are executed

- **success**: Continue the publishing action

execute_workflow	Finished executing workflow for trigger `event-publish` (180). Outcome: success

- **failure** | `blocked`: Stop publishing and log the reason

execute_workflow	Execution stopped. Node `stop-execution` (8) from Workflow `Workflow for trigger event-publish` (180) returned the following error: Execution stopped

Blocking and non-blocking

Two types of workflows:

Blocking Workflows

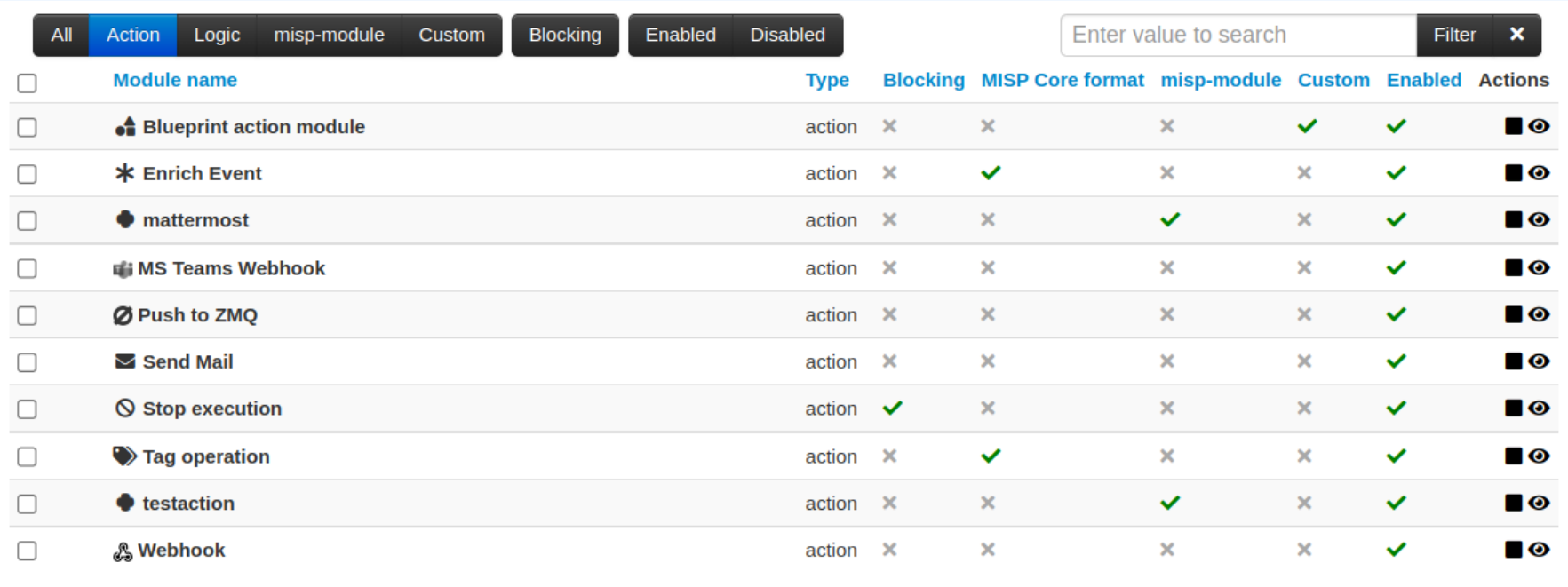
- Can prevent / block the original event to happen
- If a **blocking module** blocks the action

Regular Workflows execution outcome has no impact

- **Blocking modules** No way to prevent something that has already happened

Workflow - Action modules

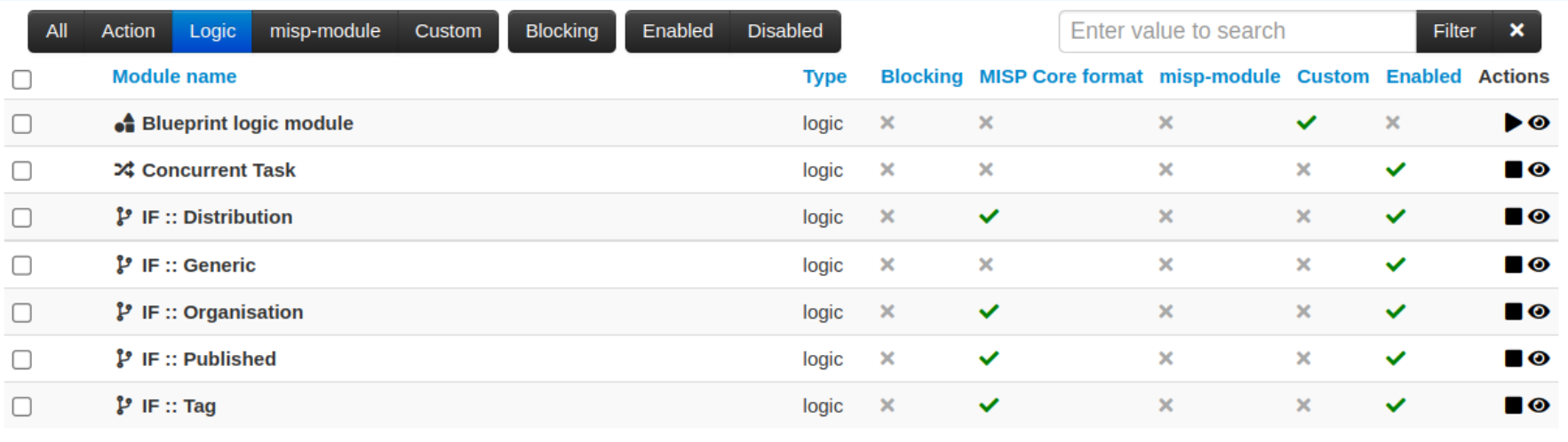
- **action** modules: Allow to executes operations or custom scripts
  - Tag operations
  - Send notifications
  - Webhooks



Module name	Type	Blocking	MISP Core format	misp-module	Custom	Enabled	Actions
Blueprint action module	action	✗	✗	✗	✓	✓	
Enrich Event	action	✗	✓	✗	✗	✓	
mattermost	action	✗	✗	✓	✗	✓	
MS Teams Webhook	action	✗	✗	✗	✗	✓	
Push to ZMQ	action	✗	✗	✗	✗	✓	
Send Mail	action	✗	✗	✗	✗	✓	
Stop execution	action	✓	✗	✗	✗	✓	
Tag operation	action	✗	✓	✗	✗	✓	
testaction	action	✗	✗	✓	✗	✓	
Webhook	action	✗	✗	✗	✗	✓	

Workflow - Logic modules

- **logic** modules: Allow to redirect the execution flow.
  - IF conditions
  - Delay execution



Sources of Workflow modules (1)

- Built-in **default** modules
  - Part of the MISP codebase
  - Get in touch if you want us to increase the selection!

Sources of Workflow modules (2)

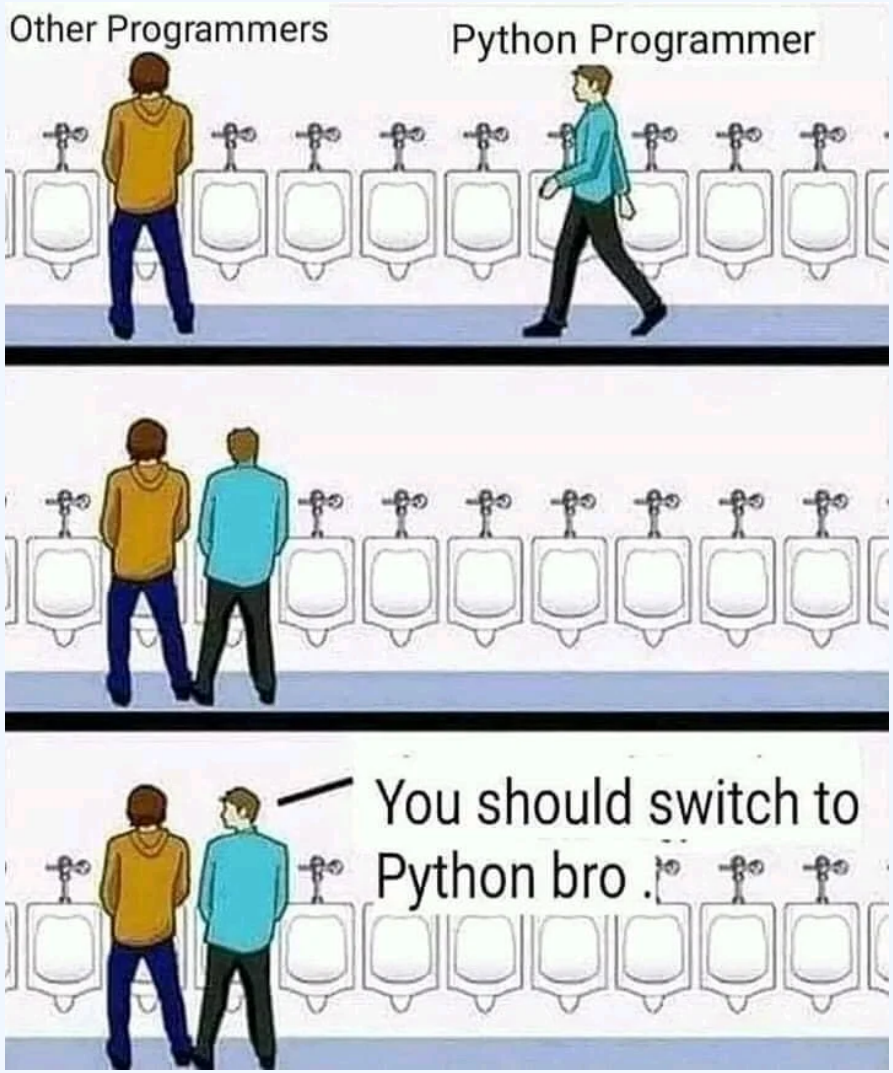
User-defined **custom** modules

- Written in PHP
- Extend existing modules
- MISP code reuse

Sources of Workflow modules (3)

Modules from the misp-module **enrichment service**

- Written in Python
- Can use any python libraries
- Plug & Play



Triggers currently available

Currently 10 triggers can be hooked. 3 being Blocking.

Triggers

List the available triggers that can be listened to by workflows.
Missing a trigger? Feel free to open a Github issue!
Documentation and concepts

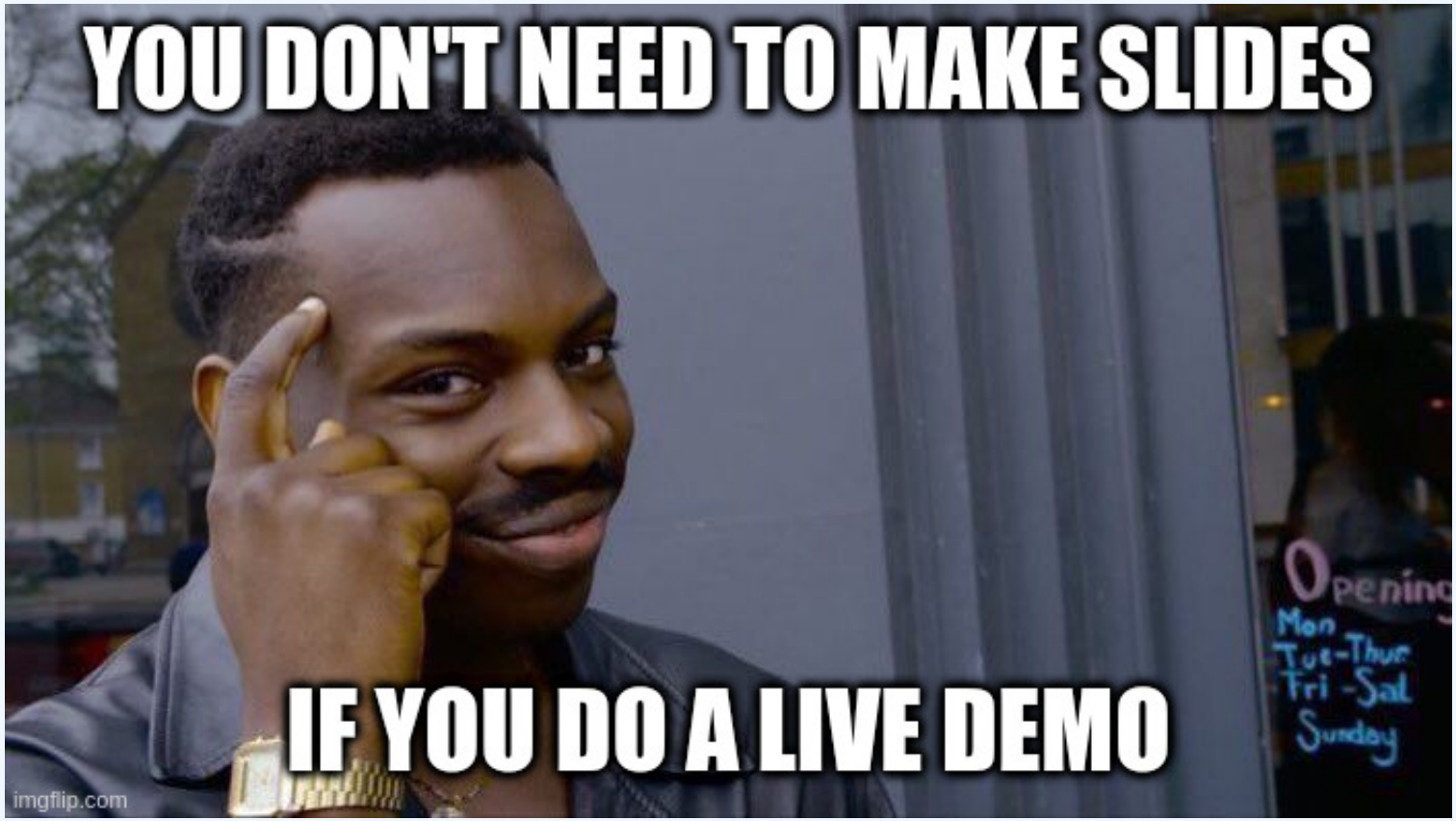
« previous | next »

All | attribute | event | object | others | post | user | Blocking | Enabled | Disabled

Trigger name	Scope	Trigger overhead	Run counter	Blocking Workflow	MISP Core format	Workflow ID	Last Update	Debug enabled	Enabled	Actions
Attribute After Save	attribute	high	83	✗	✓	160	2022-08-03 09:00:41	☐	✗	
Enrichment Before Query	others	low	1154	✓	✓	162	2022-10-17 12:35:57	☐	✓	
Event After Save	event	high	49	✗	✓	175	2022-10-14 13:32:01	☐	✓	
Event After Save New	event	low	5	✗	✓	182	2022-10-17 09:12:14	☑	✓	
Event After Save New From Pull	event	low	6	✗	✓	183	2022-10-17 09:01:36	☑	✓	
Event Publish	event	low	126	✓	✓	180	2022-10-13 10:42:53	☑	✓	
Object After Save	object	high	35	✗	✓	161	2022-08-05 07:12:52	☐	✗	
Post After Save	post	low	36	✗	✗	176	2022-07-28 13:59:51	☐	✗	
User After Save	user	low	0	✗	✗	181	2022-08-05 07:19:46	☐	✗	
User Before Save	user	low	42	✓	✗	158	2022-07-28 14:00:32	☐	✗	

Page 1 of 1, showing 1 records out of 10 total, starting on record 1, ending on 10

Demo by examples



Workflow - Getting started

Getting started with workflows (1)

2.4.160 Epic summer release

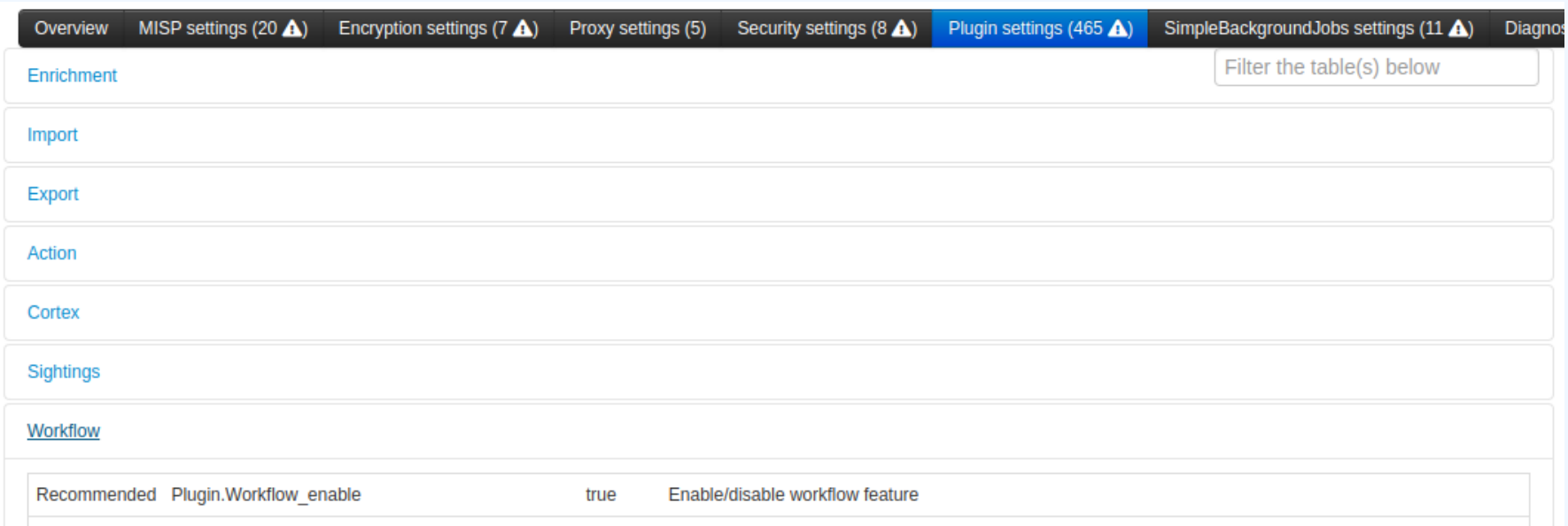
 **iglocska** released this 08 Aug 2022 v2.4.160 71d4e2c

1. Update your MISP server
2. Update all your sub-modules

Getting started with workflows (2)

Review MISP settings:

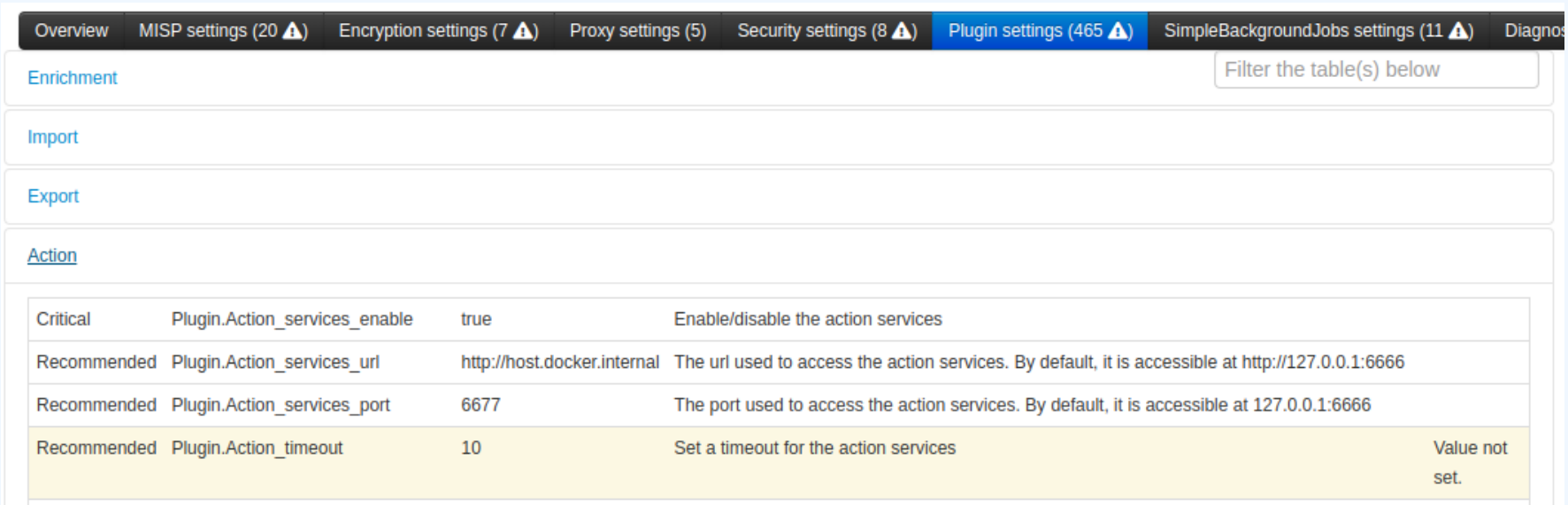
1. Make sure **MISP.background_jobs** is turned on
2. Make sure workers are **up-and-running** and healthy
3. Turn the setting **Plugin.Workflow_enable** on



Getting started with workflows (3)

[optional] Wanna enjoy misp-module ?

1. Turn the setting **Plugin.Action_services_enable** on



Getting started with workflows (4)

1. Go to the list of modules
   - `Administration > Workflows > List Modules`
   - or `/workflows/moduleIndex`
2. Make sure **default** modules are loaded
3. [optional:misp-module] Make sure **misp-module** modules are loaded

Getting started with workflows (4)

Everything is ready?

Let's see how to build a workflow!

Creating a workflow with the editor

- 1. Go to the list of triggers `Administration > Workflows`
- 2. Enable the trigger
- 3. Edit the `trigger` you want to create a workflow for
- 4. Drag an `action` module from the side panel to the canvas
- 5. Drag another `action` module or a logic module from the side panel to the canvas
- 6. From the `trigger` output, drag an arrow into the `action`'s input (left side)
- 7. Continue linking modules with the input/output system until all wanted modules are connected
- 8. Find an action that would execute the desired trigger
- 9. Execute the action and observe the effect!
- 10. Optionally, enable debug mode to see realtime execution
- 10.1. Even more text to make the slide even more unreadable
- 10.2. And even more boring

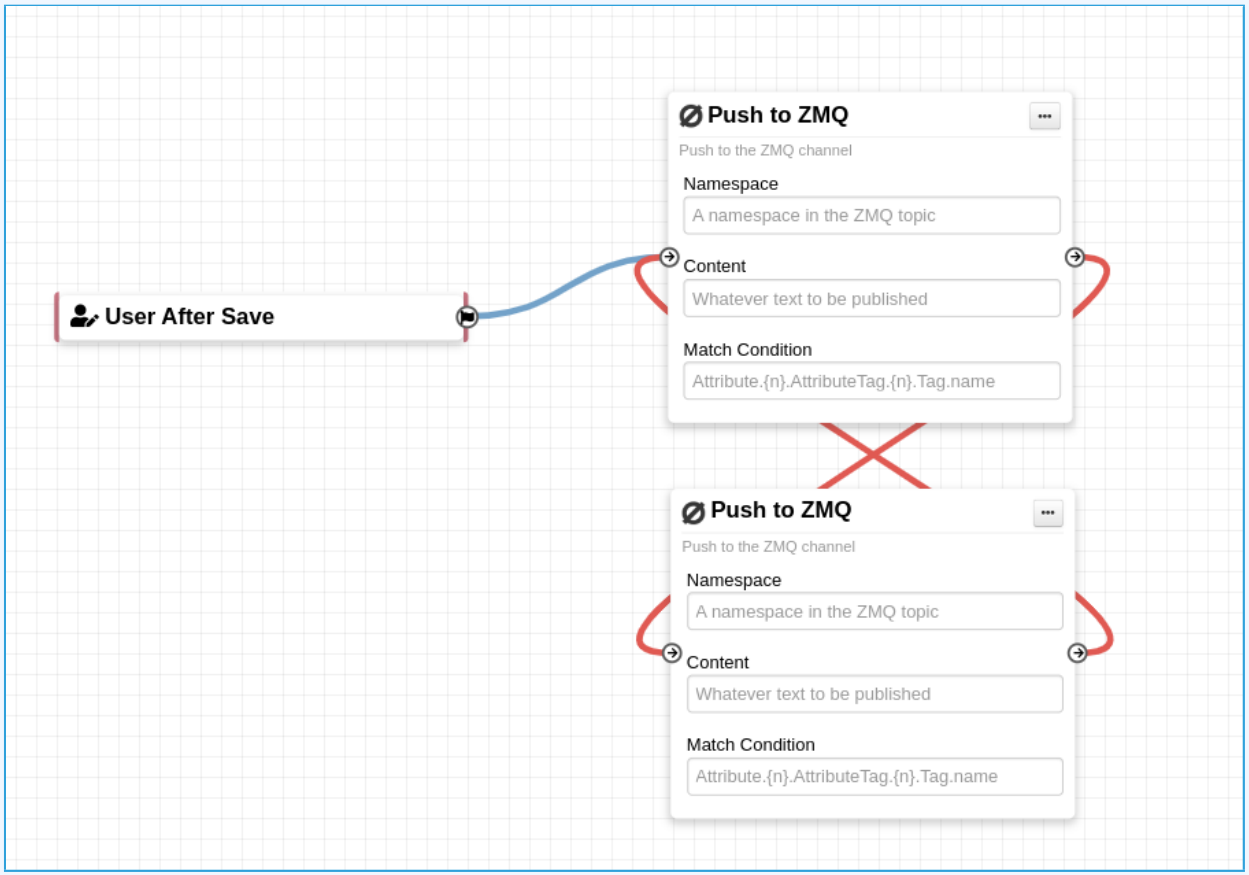
Creating a workflow with the editor

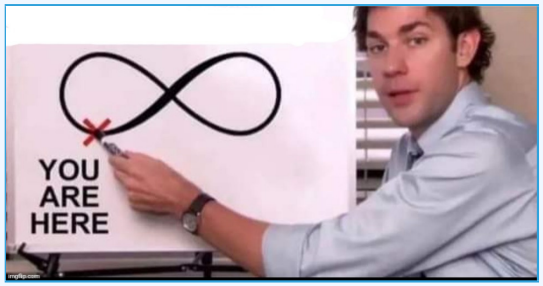
- 1. Go to the list of triggers `Administration > Workflows`
- 2. Enable t
- 3. Edit the w for
- 4. Drag an
to the canva
- 5. Drag an e
from the sid
- 6. From th
the `action`
- 7. Continu t system
until all wan
- 8. Find an trigger
- 9. Execute
- 10. Option e execution
- 10.1. Even unreadable
- 10.2. And even more boring

Considerations when working with workflows

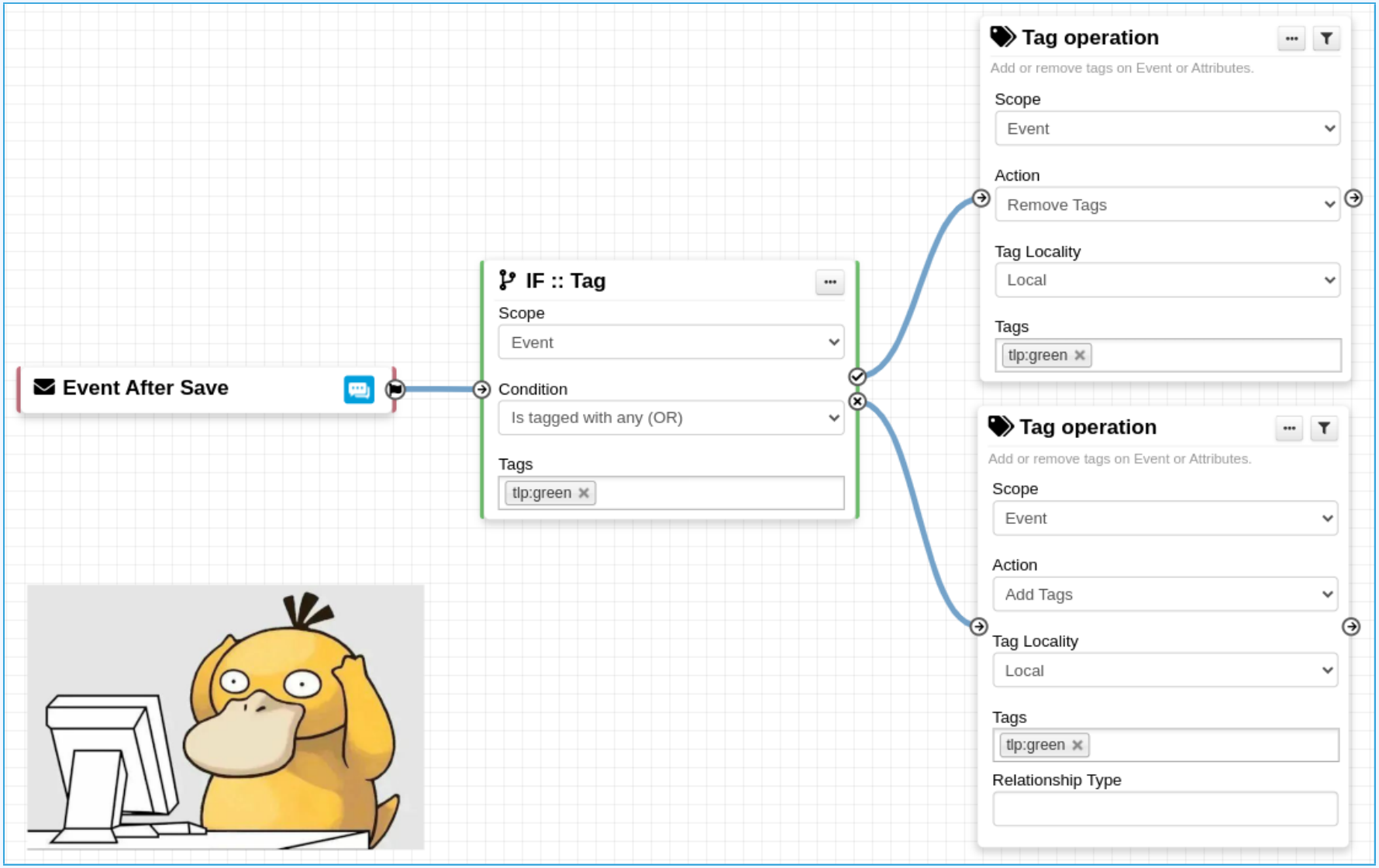
Working with the editor - Operations not allowed

Execution loop are not authorized





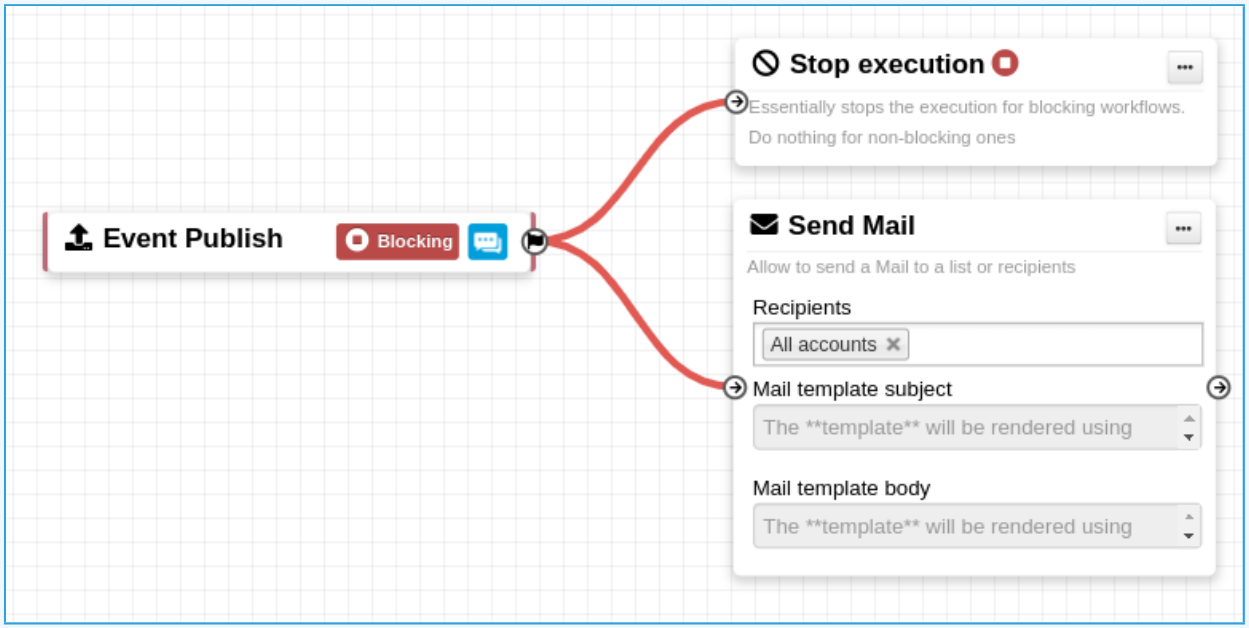
Recursive workflows



⚠Recursion: If an action re-run the workflow

Working with the editor - Operations not allowed

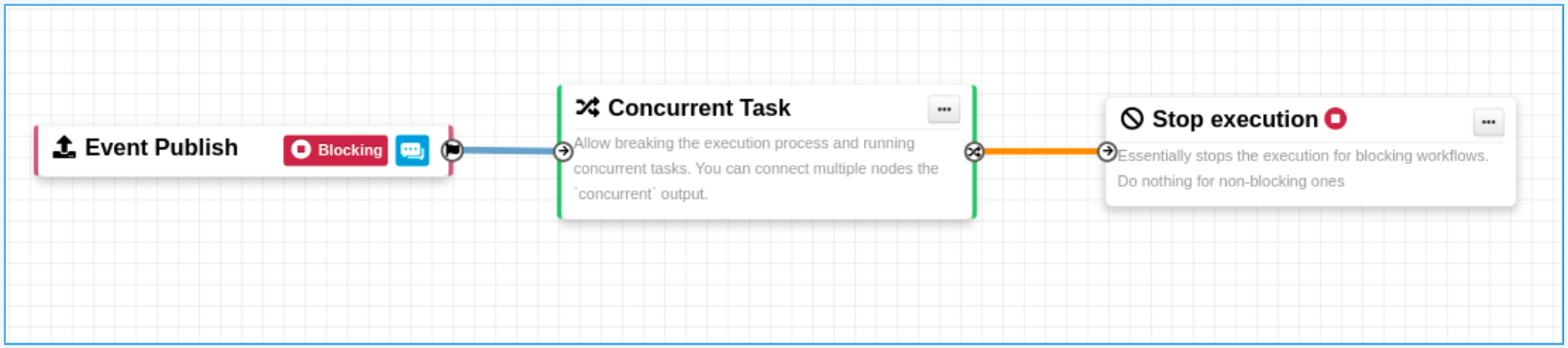
Multiple connections from the same output



- Execution order not guaranted
- Confusing for users

Cases showing a warning:

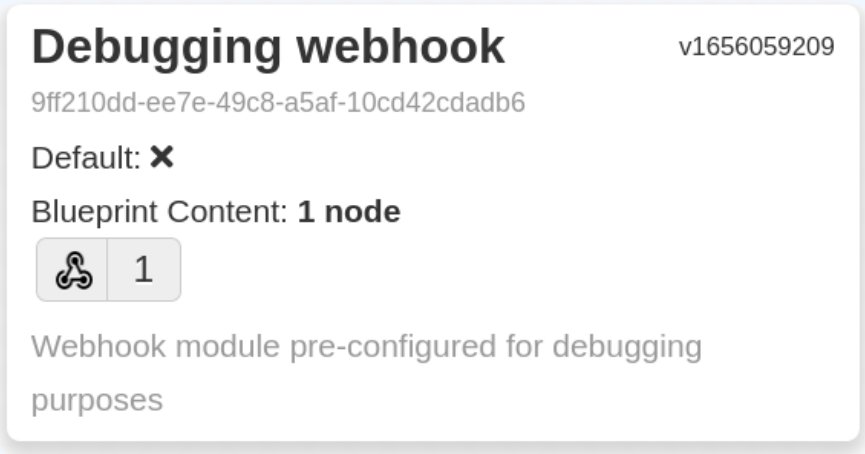
- **Blocking** modules in a **non-blocking** workflow
- **Blocking** modules after a **concurrent tasks** module



Advanced usage

Workflow blueprints

1. Blueprints allow to **re-use parts** of a workflow in another one
2. Blueprints can be saved, exported and **shared**



Blueprints sources:

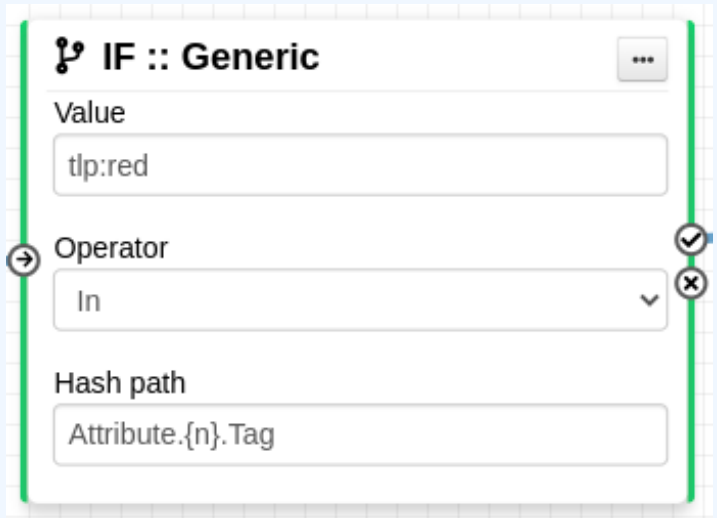
1. Created or imported by users
2. From the `MISP/misp-workflow-blueprints` repository[1]

[1] https://github.com/MISP/misp-workflow-blueprints

Hash path filtering

Filtering and checking conditions using hash path expression.

```php
$path_expression = '{n}[name=fred].id';
$users = [
    {'id': 123, 'name': 'fred', 'surname': 'bloggs'},
    {'id': 245, 'name': 'fred', 'surname': 'smith'},
    {'id': 356, 'name': 'joe', 'surname': 'smith'},
];
$ids = Hash::extract($users, $path_expression);
// => $ids will be [123, 245]
```

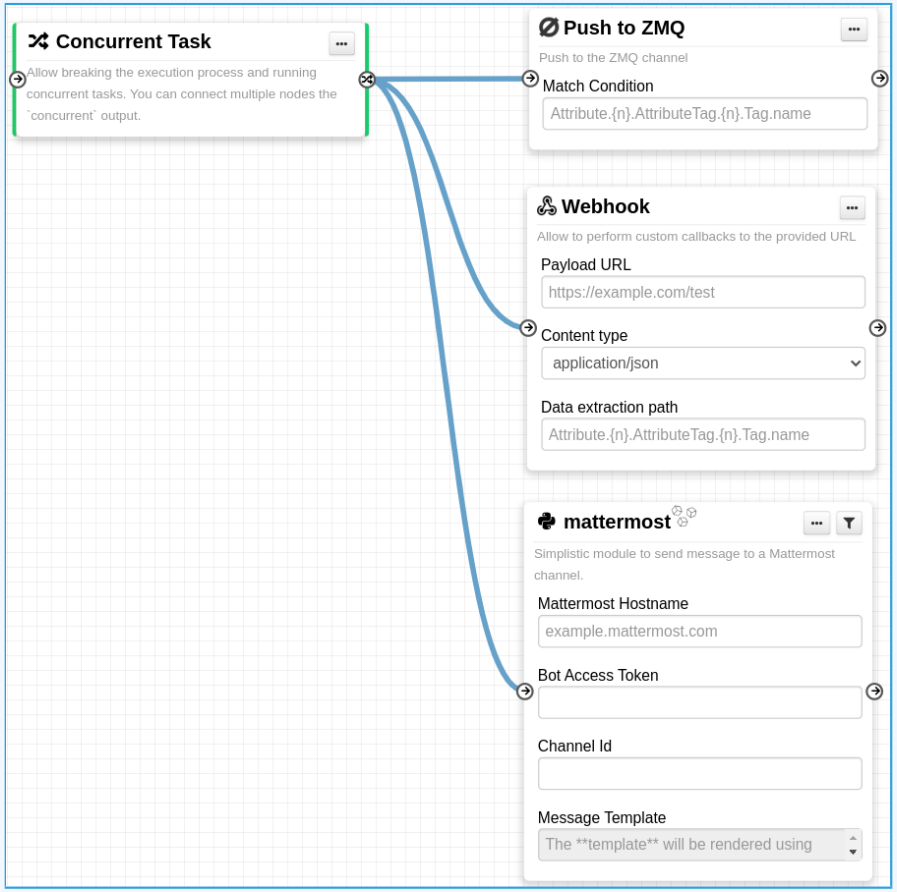


Data format in Workflows

- In most cases, the format is compliant with the **MISP Core format**
- But data has **additional properties**
  - Attributes are **always encapsulated** in the Event or Object
  - Additional key **`_AttributeFlattened`**
  - Additional key **`_allTags`**
  - Additional key **`inherited`** for Tags

Logic module: Concurrent Task

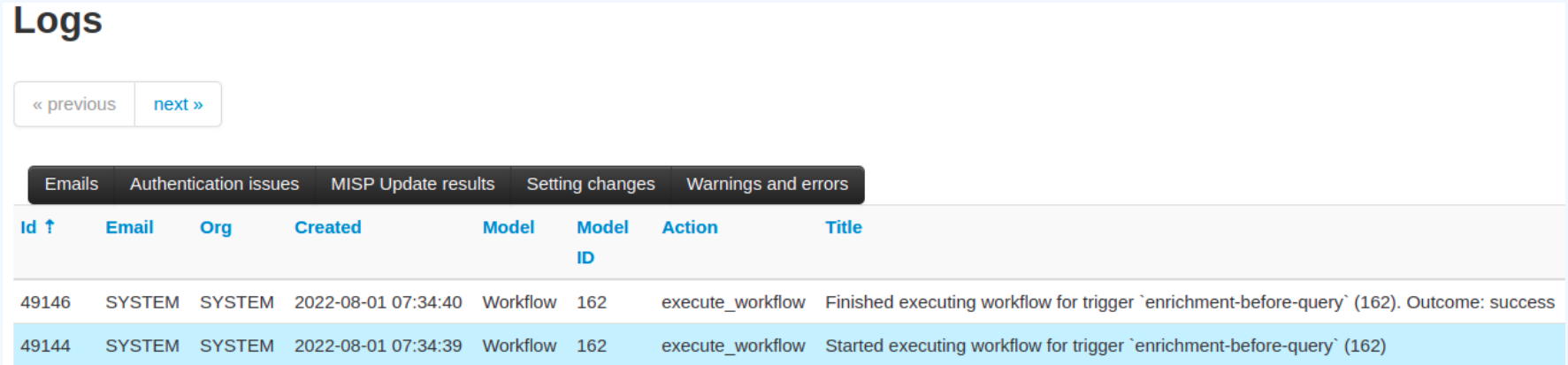
- Logic module allowing **multiple output** connections
- **Postpone the execution** for remaining modules
- Blocking modules **cannot cancel** ongoing operations 



Debugging

Debugging Workflows: Log Entries

- Workflow execution is logged in the application logs:
  - `/admin/logs/index`
- Or stored on disk in the following file:
  - `/app/tmp/logs/workflow-execution.log`



Logs

« previous | next »

Emails | Authentication issues | MISP Update results | Setting changes | Warnings and errors

Id ↑	Email	Org	Created	Model	Model ID	Action	Title
49146	SYSTEM	SYSTEM	2022-08-01 07:34:40	Workflow	162	execute_workflow	Finished executing workflow for trigger `enrichment-before-query` (162). Outcome: success
49144	SYSTEM	SYSTEM	2022-08-01 07:34:39	Workflow	162	execute_workflow	Started executing workflow for trigger `enrichment-before-query` (162)

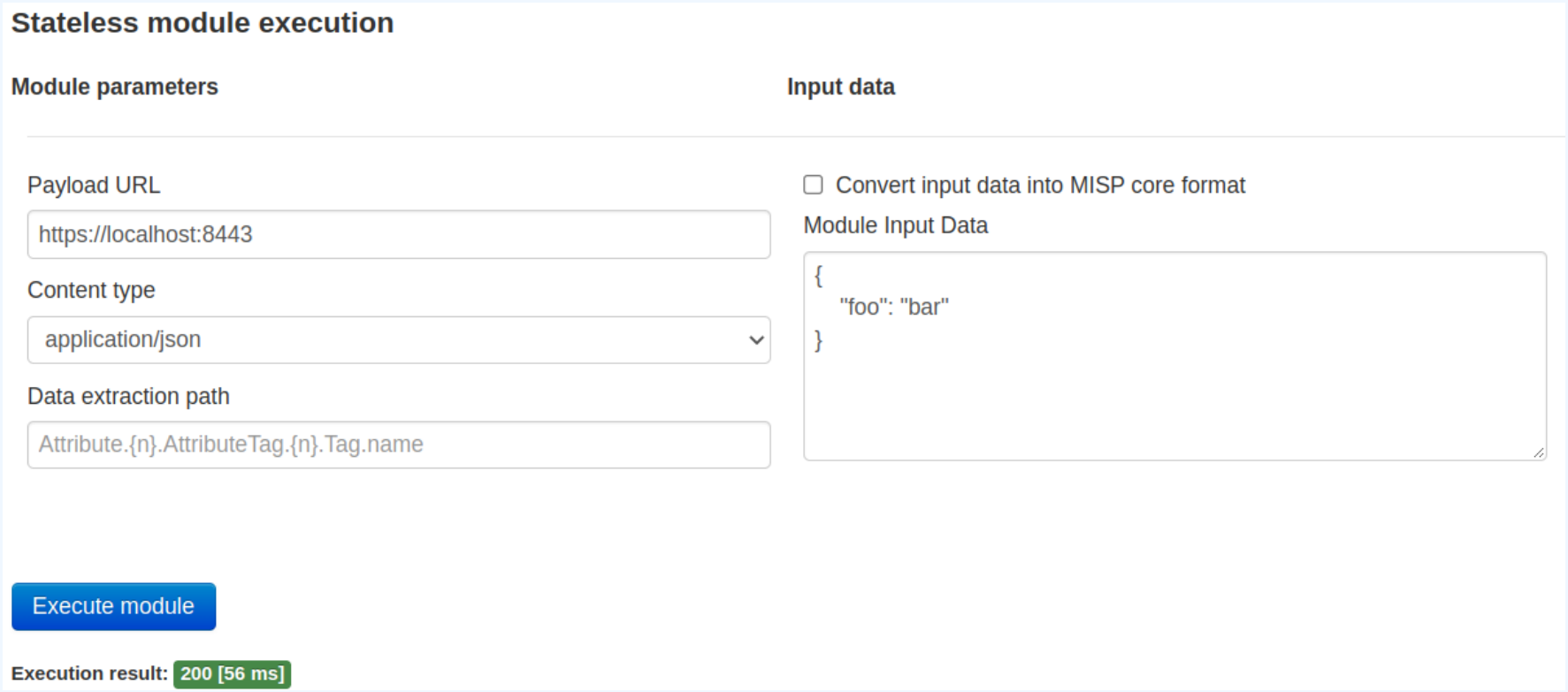
Debugging Workflows: Debug mode

- The 

can be turned on for each workflows
- Each nodes will send data to the provided URL
  - Configure the setting: `Plugin.Workflow_debug_url`
- Result can be visualized in
  - **offline**: `tools/misp-workflows/webhook-listener.py`
  - **online**: `requestbin.com` or similar websites

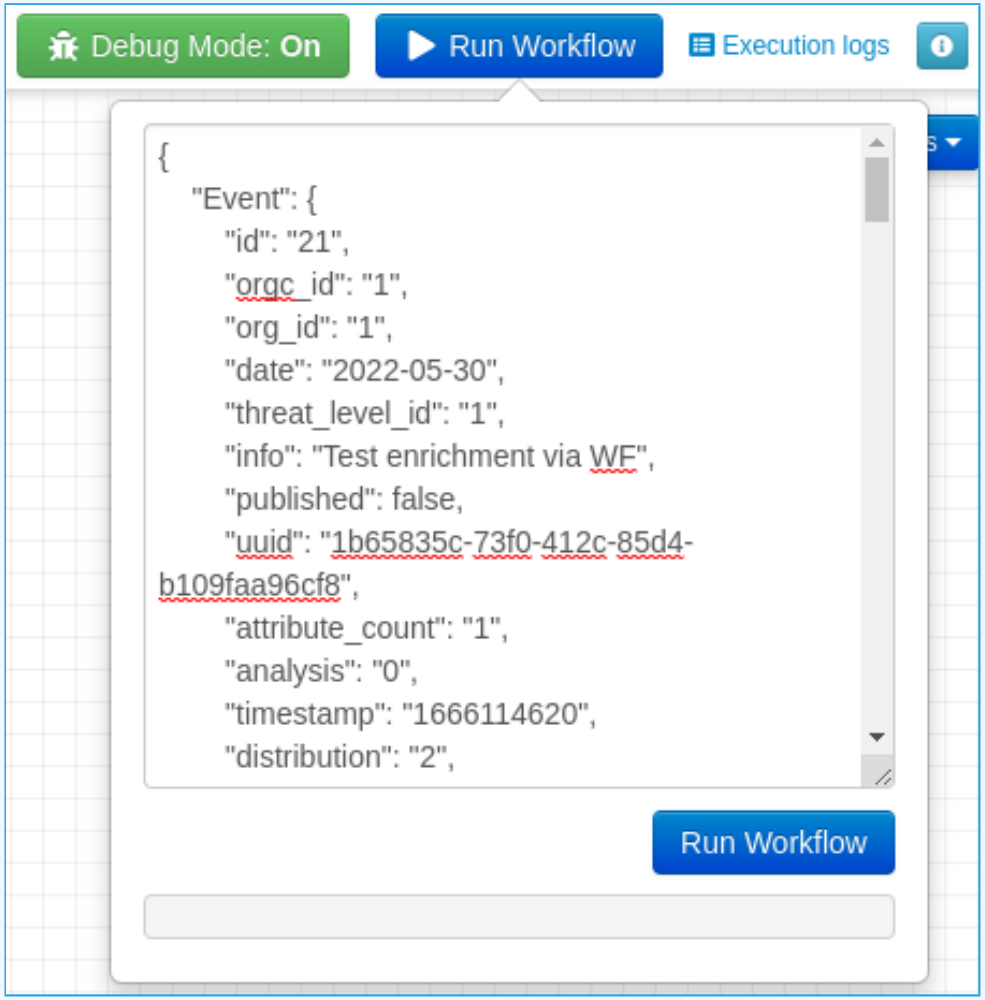
Debugging modules: Stateless execution

- Test custom modules with custom input



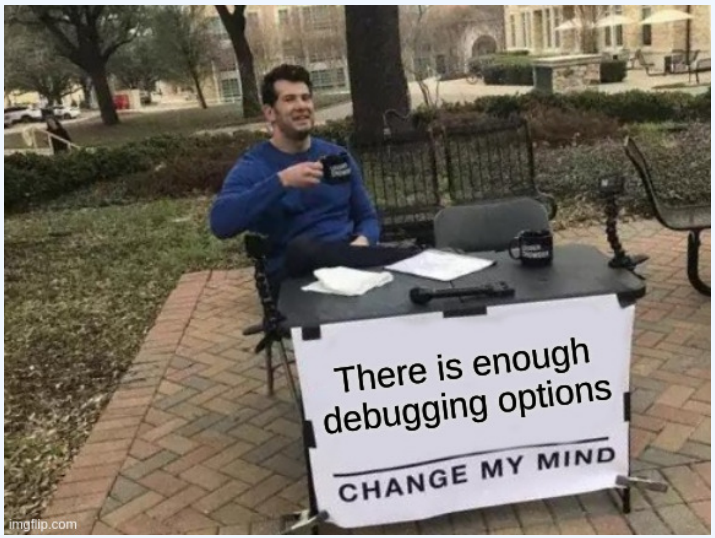
Debugging modules: Re-running workflows

- Try workflows with custom input
- Re-run workflows to ease debugging



Debugging options

- Workflow execution and outcome
- Module execution and outcome
- Live workflow debugging with module inspection
- Re-running/testing workflows with custom data
- Stateless module execution



Extending the system

Creating a new module in PHP

app > Lib > WorkflowModules > action > Module_blueprint_action_module.php > ...

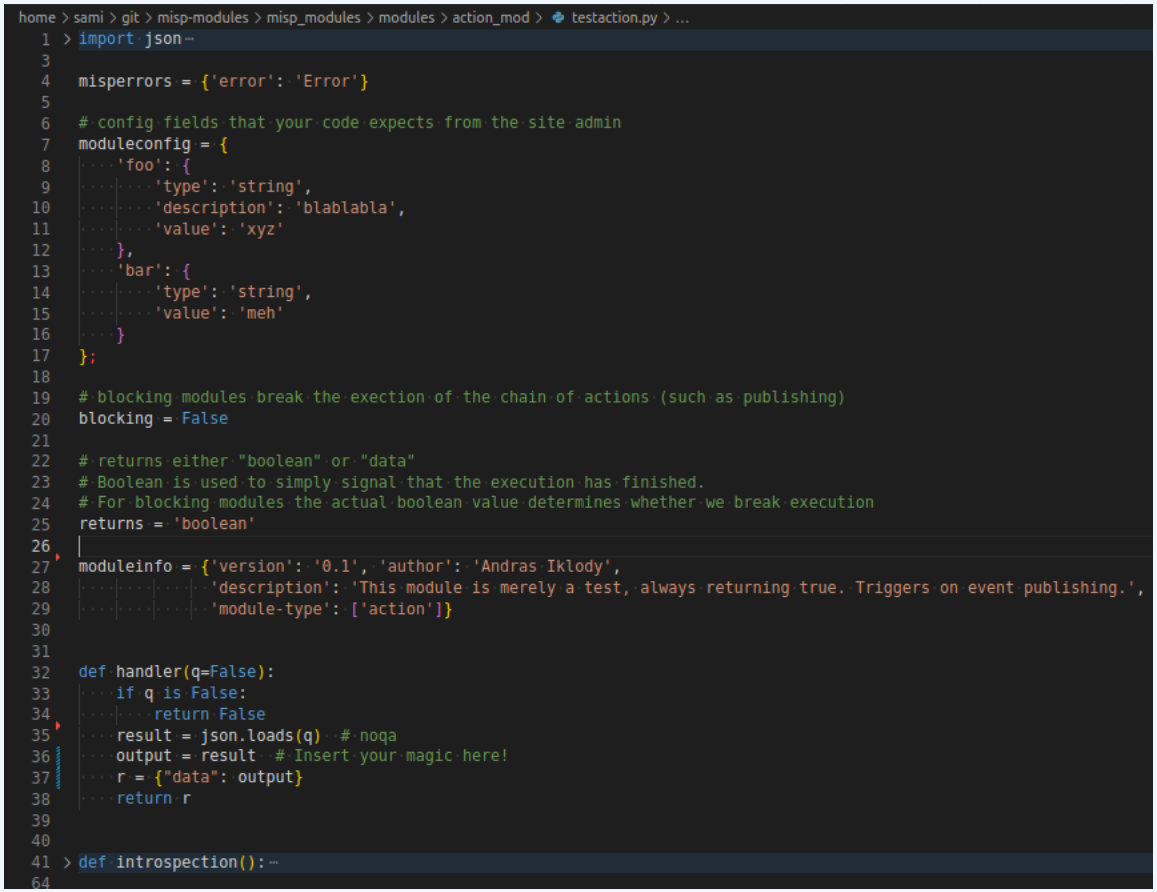
```php
<?php
include_once APP . 'Model/WorkflowModules/WorkflowBaseModule.php';

class Module_blueprint_action_module extends WorkflowBaseModule
{
    public $is_blocking = false;
    public $disabled = true;
    public $id = 'blueprint-action-module';
    public $name = 'Blueprint action module';
    public $description = 'Lorem ipsum dolor, sit amet consectetur adipisicing elit.';
    public $icon = 'shapes';
    public $inputs = 1;
    public $outputs = 1;
    public $params = [];

    public function exec(array $node, WorkflowRoamingData $roamingData, array &$errors = [])
    : bool
    {
        parent::exec($node, $roamingData, $errors);
        // If $this->is_blocking == true, returning `false` will stop the execution.
        $errors[] = __('Execution stopped');
        return false;
    }
}
```

- **app/Lib/**WorkflowModules/action/[module_name].php
- Designed to be easilty extended
  - Helper functions
  - Module configuration as variables
  - Implement runtime logic

Creating a new module in Python



- Similar to how other `misp-modules` are implemented
  - Helper functions
  - Module configuration as variables
  - Implement runtime logic

Ideas

- Chat notification a community when new user joins an instance
- Trigger on any action via log entries
- Extend existing MISP behavior: Push correlation in another system
- Sanity check to block publishing
- ...

Future works

- More modules
- More modules
- More triggers
- More documentation
- Recursion prevention system
- On-the-fly data override?

Final words

- Feature designed to quickly and cheaply support CTI pipeline
- **Beta**: Feature unlikely to change. But still..
- Waiting for feedback!