

VISUALISE ALL THE THINGS

BUILDING DASHBOARD WIDGETS FOR MISP

CIRCL / TEAM MISP PROJECT

[HTTP://WWW.MISP-PROJECT.ORG/](http://www.misp-project.org/)
TWITTER: [@MISPPROJECT](https://twitter.com/MISPPROJECT)

MISP PROJECT



2022-09-13

Visualise all the things

VISUALISE ALL THE THINGS
BUILDING DASHBOARD WIDGETS FOR MISP

CIRCL / TEAM MISP PROJECT

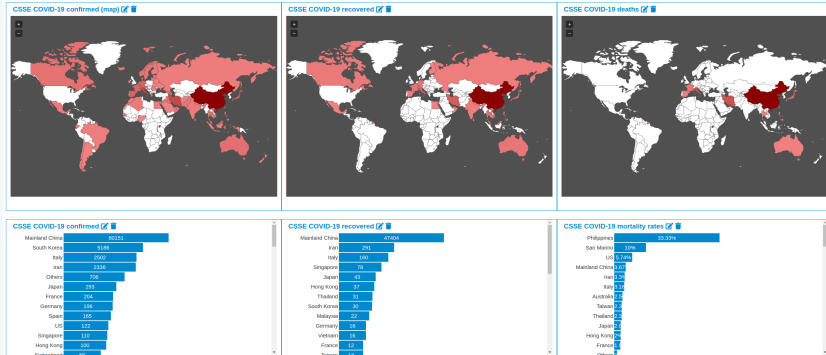
[HTTP://WWW.MISP-PROJECT.ORG/](http://www.misp-project.org/)
TWITTER: [@MISPPROJECT](https://twitter.com/MISPPROJECT)

MISP PROJECT



DASHBOARD IN MISP

- User configurable simple dashboard interface
- **Visualise, aggregate** and **track** data important to you
- Brand new feature, still undergoing reworks



2022-09-13

Visualise all the things

Dashboard in MISP

DASHBOARD IN MISP

- User configurable simple dashboard interface
- **Visualise, aggregate** and **track** data important to you
- Brand new feature, still undergoing reworks



- **Backend** for the widget, full access to all MISP internals
- **Load, convert, format** to be represented via view widgets
- **Widget metadata** - size, name, description, behaviours
- Only main function required to be implemented: **handler()**
- Optional: **checkPermissions()** for **ACL**
- Accepts **user configuration** for which a template can be provided
- Located in `/var/www/MISP/app/Lib/Dashboard/`
- Custom widgets can be placed in `/var/www/MISP/app/Lib/Dashboard/Custom/`

└─ The internals of awidget

- **Backend** for the widget, full access to all MISP internals
- **Load, convert, format** to be represented via view widgets
- **Widget metadata** - size, name, description, behaviours
- Only main function required to be implemented: **handler()**
- Optional: **checkPermissions()** for **ACL**
- Accepts **user configuration** for which a template can be provided
- Located in `/var/www/MISP/app/Lib/Dashboard/`
- Custom widgets can be placed in `/var/www/MISP/app/Lib/Dashboard/Custom/`

- View files are included by default and reusable
- Currently we have a small but growing list of views
 - ▶ BarChart
 - ▶ SimpleList
 - ▶ WorldMap
- Converts the data passed by the Widget logic to HTML
- Located in
`/var/www/MISP/view/Elements/dashboard/Widgets/`

└─ The view layer of a widget

- View files are included by default and reusable
- Currently we have a small but growing list of views
 - ▶ BarChart
 - ▶ SimpleList
 - ▶ WorldMap
- Converts the data passed by the Widget logic to HTML
- Located in
`/var/www/MISP/view/Elements/dashboard/Widgets/`

■ Widgets can additionally be tied to certain **behaviours**:

▶ Caching

- Executions of the widget logic are cached
- **Separate caches for each organisation in addition to site admins**
- Cache duration is controlled by the widget logic

▶ Refresh

- Widgets can be set to refresh after x seconds
- ▶ Both of these should be used with special care in regards to the use of **system resources**

■ Widgets can additionally be tied to certain **behaviours**:

- ▶ Caching
 - Executions of the widget logic are cached
 - **Separate caches for each organisation in addition to site admins**
 - Cache duration is controlled by the widget logic
- ▶ Refresh
 - Widgets can be set to refresh after x seconds
- ▶ Both of these should be used with special care in regards to the use of **system resources**

- Let's start with a skeleton
- Create `/var/www/MISP/app/Lib/Dashboard/Custom/WhoamiWidget.php`
- MISP will parse anything ending with `Widget.php` in this directory

EXERCISE MODULE: SIMPLE WHOAMI

```
1 <?php
2 class MispWhoamiWidget
3 {
4     public $title = 'Whoami';
5     public $render = 'SimpleList';
6     public $width = 2;
7     public $height = 2;
8     public $params = array();
9     public $description = 'Shows information about the
        currently logged in user.';
10    public $cacheLifetime = false;
11    public $autoRefreshDelay = 3;
12
13    public function handler($user, $options = array())
14    {
15        $data = array();
16        return $data;
17    }
18 }
```

Visualise all the things

└ Exercise module: simple Whoami

```
1 <?php
2 class MispWhoamiWidget
3 {
4     public $title = 'Whoami';
5     public $render = 'SimpleList';
6     public $width = 2;
7     public $height = 2;
8     public $params = array();
9     public $description = 'Shows information about the
        currently logged in user.';
10    public $cacheLifetime = false;
11    public $autoRefreshDelay = 3;
12
13    public function handler($user, $options = array())
14    {
15        $data = array();
16        return $data;
17    }
18 }
```

- **\$title:** The name of the widget
- **\$description:** A description of the widget
- **\$render:** The view element to use in rendering the widget
- **\$width & \$height:** Default relative dimensions
- **\$params:** Configuration array with explanations for each key
- **\$cacheLifetime:** The lifetime of the caches in seconds (false disables it)
- **\$autoRefreshDelay:** The time in seconds between each refresh (false disables it)

└─ Meta information

- **\$title:** The name of the widget
- **\$description:** A description of the widget
- **\$render:** The view element to use in rendering the widget
- **\$width & \$height:** Default relative dimensions
- **\$params:** Configuration array with explanations for each key
- **\$cacheLifetime:** The lifetime of the caches in seconds (false disables it)
- **\$autoRefreshDelay:** The time in seconds between each refresh (false disables it)

THE HANDLER

```
1 public function handler($user, $options = array())
2 {
3     $this->Log = ClassRegistry::init('Log');
4     $entries = $this->Log->find('all', array(
5         'recursive' => -1,
6         'conditions' => array(
7             'action' => 'login', 'user_id' => $user['id']
8         ),
9         'order' => 'id desc',
10        'limit' => 5,
11        'fields' => array('created', 'ip')
12    ));
13    foreach ($entries as &$entry) {
14        $entry = $entry['Log']['created'] . ' --- ' .
15        (
16            empty($entry['Log']['ip']) ?
17            'IP not logged' :
18            $entry['Log']['ip']
19        );
20    }
21    return array(
22        array('title' => 'Email', 'value' => $user['email']),
23        array(
24            'title' => 'Role', 'value' => $user['Role']['name']
25        ),
26        array(
27            'title' => 'Organisation',
28            'value' => $user['Organisation']['name']
29        ),
30        array(
31            'title' => 'IP', 'value' => $_SERVER['REMOTE_ADDR']
32        ),
33        array('title' => 'Last logins', 'value' => $entries)
34    );
35 }
```

Visualise all the things

2022-09-13

└─ The handler

THE HANDLER

```
public function handler($user, $options = array())
{
    $this->Log = ClassRegistry::init('Log');
    $entries = $this->Log->find('all', array(
        'recursive' => -1,
        'conditions' => array(
            'action' => 'login', 'user_id' => $user['id']
        ),
        'order' => 'id desc',
        'limit' => 5,
        'fields' => array('created', 'ip')
    ));
    foreach ($entries as &$entry) {
        $entry = $entry['Log']['created'] . ' --- ' .
        (
            empty($entry['Log']['ip']) ?
            'IP not logged' :
            $entry['Log']['ip']
        );
    }
    return array(
        array('title' => 'Email', 'value' => $user['email']),
        array(
            'title' => 'Role', 'value' => $user['Role']['name']
        ),
        array(
            'title' => 'Organisation',
            'value' => $user['Organisation']['name']
        ),
        array(
            'title' => 'IP', 'value' => $_SERVER['REMOTE_ADDR']
        ),
        array('title' => 'Last logins', 'value' => $entries)
    );
}
```

Whoami

Email: admin@admin.test

Role: admin

Organisation: ORGNAME

IP: ::1

Last logins:

2020-03-05 06:50:46 --- ::1

2020-03-04 21:35:15 --- IP not logged

2020-03-04 09:34:44 --- IP not logged

2020-03-03 16:58:35 --- IP not logged

2020-03-03 06:49:10 --- IP not logged

2022-09-13

Visualise all the things

Result

RESULT

Whoami  
Email: admin@admin.test
Role: admin
Organisation: ORGNAME
IP: ::1
Last logins:
2020-03-05 06:50:46 --- ::1
2020-03-04 21:35:15 --- IP not logged
2020-03-04 09:34:44 --- IP not logged
2020-03-03 16:58:35 --- IP not logged
2020-03-03 06:49:10 --- IP not logged