Visualise all the things

Building dashboard widgets for MISP

CIRCL / Team MISP Project

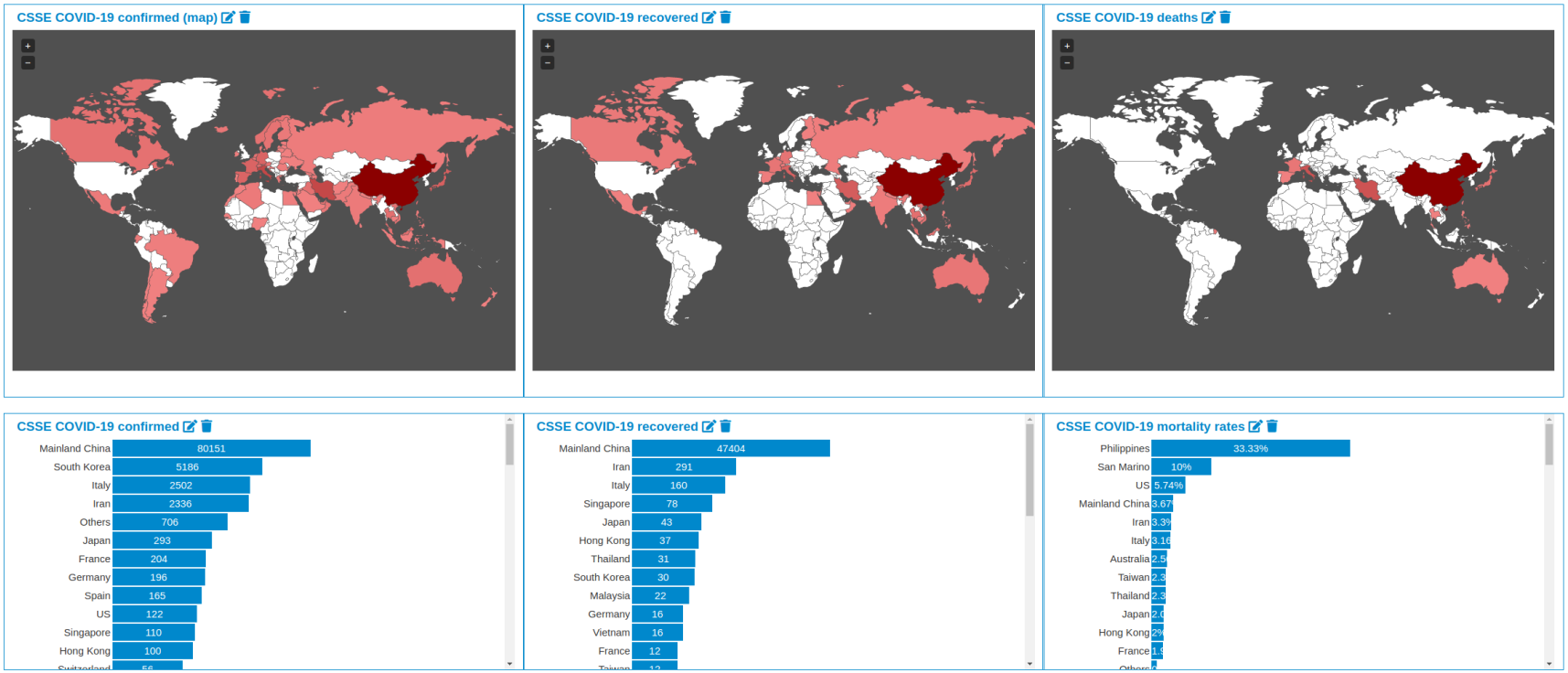
http://www.misp-project.org/
Twitter: @MISPProject

13th ENISA-EC3 Workshop

MISP
Threat Sharing

Dashboard in MISP

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



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/

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/

Widget behaviours

- 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**

Exercise module: simple Whoami

- 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

```php
<?php
class MispWhoamiWidget
{
  public $title = 'Whoami';
  public $render = 'SimpleList';
  public $width = 2;
  public $height = 2;
  public $params = array();
  public $description = 'Shows information about the
      currently logged in user.';
  public $cacheLifetime = false;
  public $autoRefreshDelay = 3;

  public function handler($user, $options = array())
  {
    $data = array();
    return $data;
  }
}
```

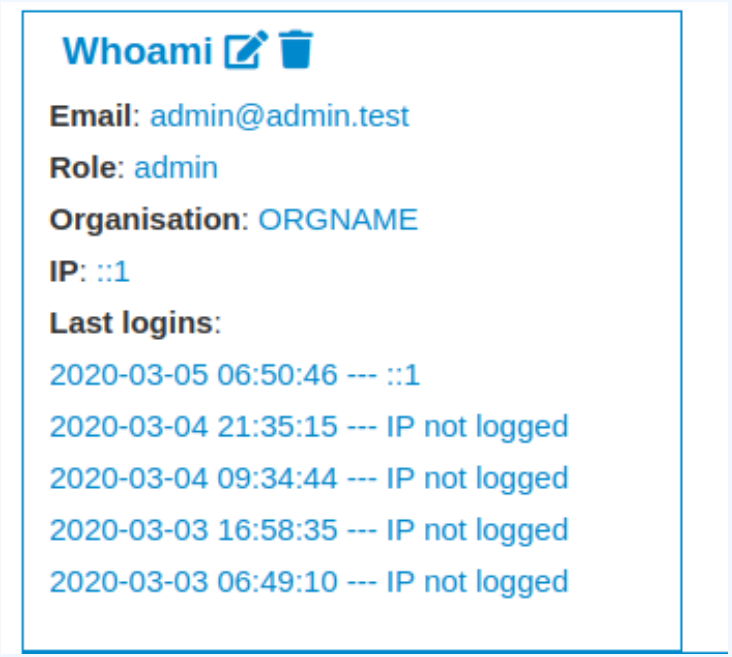
Meta information

- **$title**: The name of the widget
- **$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 }
```

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