MISP

INTELLIGENCE SHARING

The integrated dashboard in MISP 2.5

Build a useful view, understand its scope

CIRCL / MISP Project

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

Reference snapshot: 29 October 2026

Build a view that supports a decision

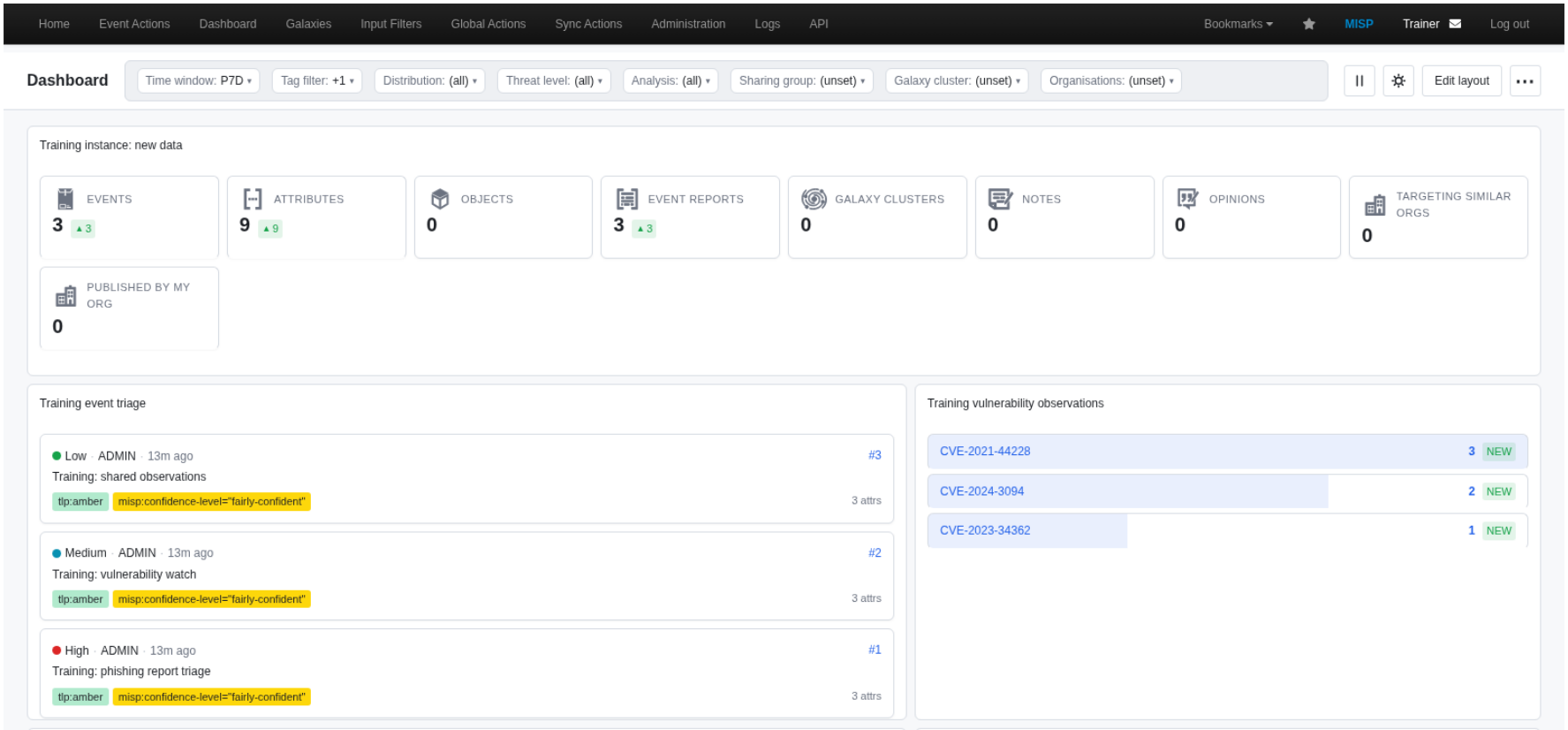
1. Choose widgets for the question you need to answer.
2. Set their filters and understand their scope.
3. Save a useful personal layout.
4. Follow interesting results back to the underlying data.

Two different dashboard applications

Surface	Integrated dashboard	Standalone misp-dashboard
Location	Inside MISP: `/dashboards`	Separate web application.
Model	Personal widgets, layouts and templates.	ZeroMQ activity stream and its own processing.
This training	Current standard course.	Historical course in the archive.

Build your personal view

A real MISP 2.5 dashboard



MISP 2.5.50, default theme, site-admin training account. Synthetic events; top of the board.

What determines your initial layout?

1. Your saved personal dashboard, if one exists.
2. Otherwise, an accessible default template.
3. Otherwise, an empty board.

An empty board can be intentional

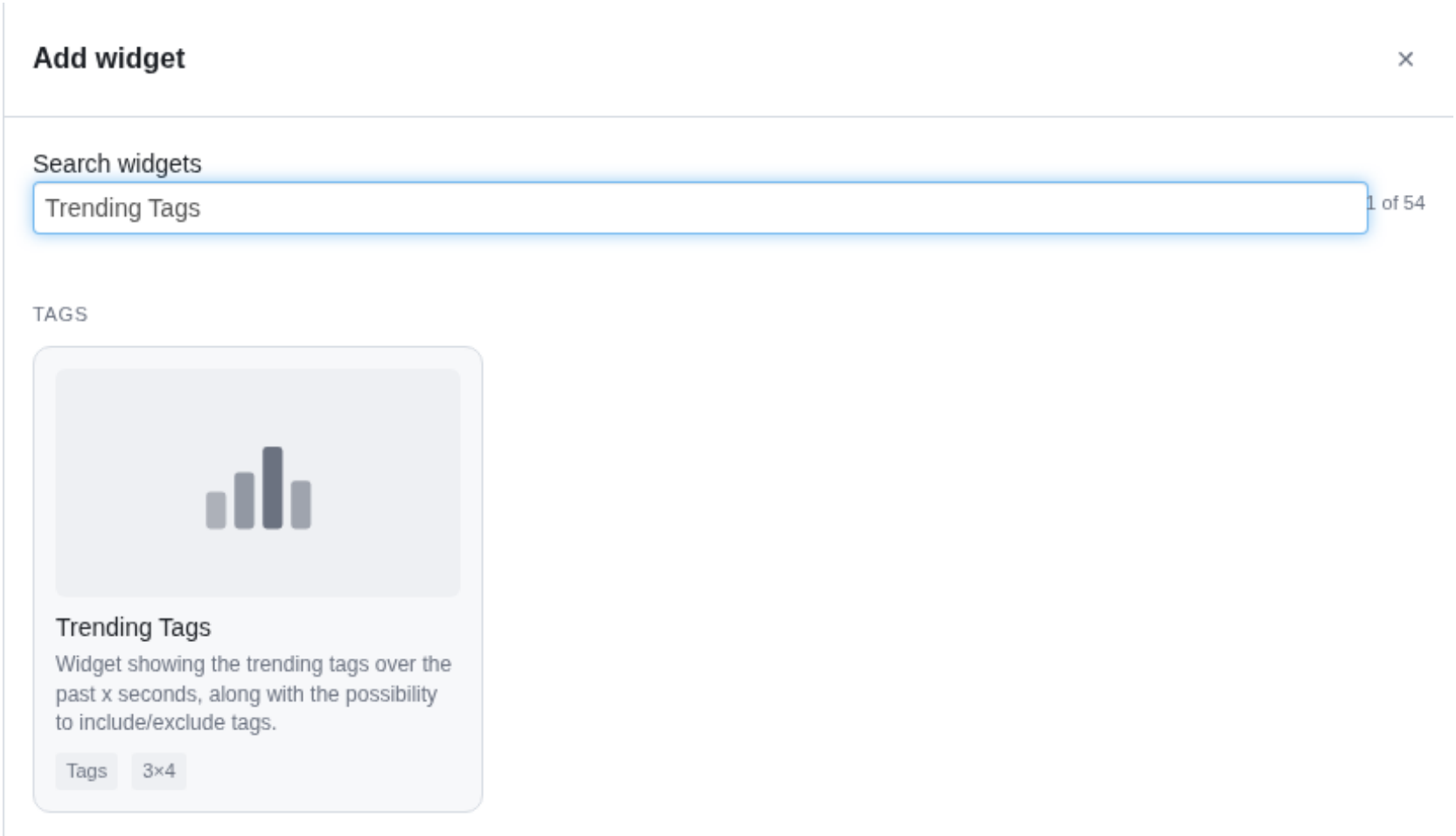
An explicitly saved empty layout is not replaced by a default on every visit.

Layout edits and widget settings save differently

Task	Save behavior
Add, move, resize or remove tiles	Enter *Edit layout*; use *Save changes* or *Discard*.
Configure a widget	Its panel's *Save* persists configuration separately.
Apply a shared filter	Updates supporting widgets' configuration.

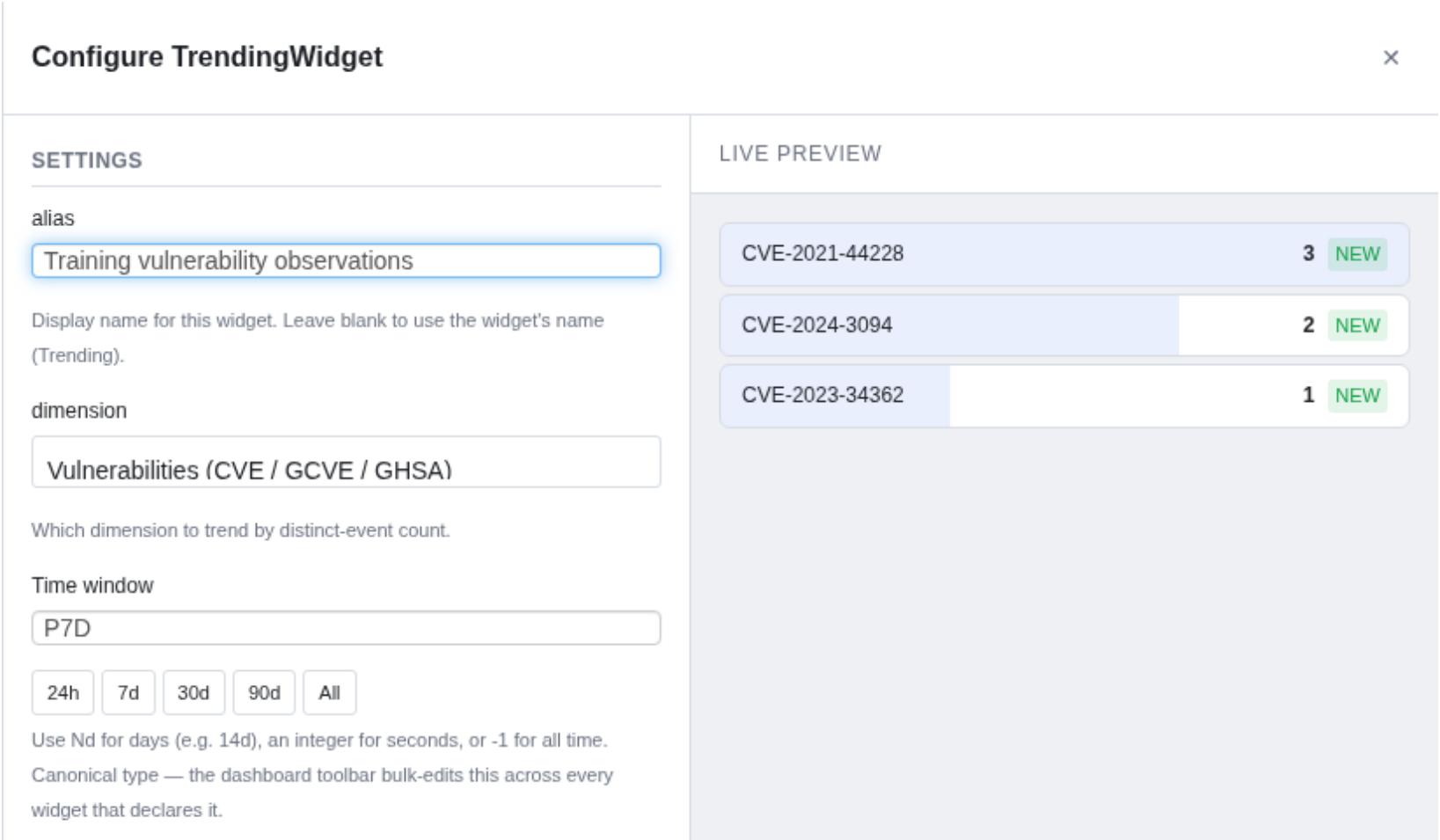
Discarding layout edits is not a general undo for saved widget settings.

Find a widget in the gallery



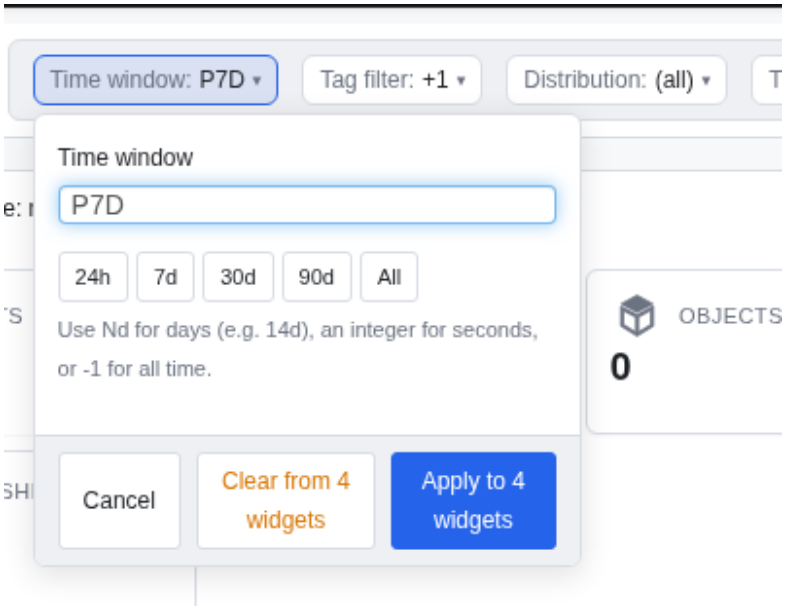
Actual gallery search for Trending Tags; the catalog is role-dependent.

Configure with typed controls and a preview



Alias, dimension and time window: detail of the actual configuration panel.

A shared filter targets supporting widgets



- Read the affected-widget count.
- A mixed value means settings differ.
- Apply changes only to widgets that declare that filter.

Make the scope visible to the next reader

- Give each widget a useful alias.
- Choose the time window and result limit deliberately.
- Check whether tags use exact labels or substring patterns.
- Use drill-down links to inspect the contributing data.

Interpret the numbers

Choose a widget for the question

Question	Useful starting point
What changed recently?	New data; event cards; recent reports or analyst data.
What recurs in the data?	Trending vulnerabilities, actors, techniques or tags.
Where is data associated?	Geolocation and country-map widgets.
Is the instance operating well?	Administrator health, resources and worker widgets.

Do not assume every metric has the same visibility

Detailed event views

Event Stream and Trending event counts use the viewer's event access.

Instance-scale aggregates

New data includes global counts and organisation-contextual metrics. Some geolocation summaries also aggregate at instance scope.

Understand what Trending counts

- Count distinct events per value, not repeated attribute rows.
- Compare with the immediately preceding equal-length window.
- Interpret minimum-count thresholds before trusting momentum.
- All-time mode has no previous window for a momentum comparison.

Training example

Public vulnerability identifiers appear in three synthetic events. A high count is not a severity score.

Refresh frequency is not data freshness

- Some widgets refresh automatically; others do not.
- Pause or resume automatic refresh from the header.
- Per-widget overrides are available where supported.
- A refresh may still return cached data.

Treat maps and coverage as evidence summaries

- Geolocation can be inferred from several kinds of data.
- A plotted origin is not proof of the attacker’s location.
- An ATT&CK tile reflects tagged information.
- Tagged coverage is not proof of detection coverage.

Separate an analyst view from an operator view

Analyst

Recent events, reports, trends and useful drill-downs.

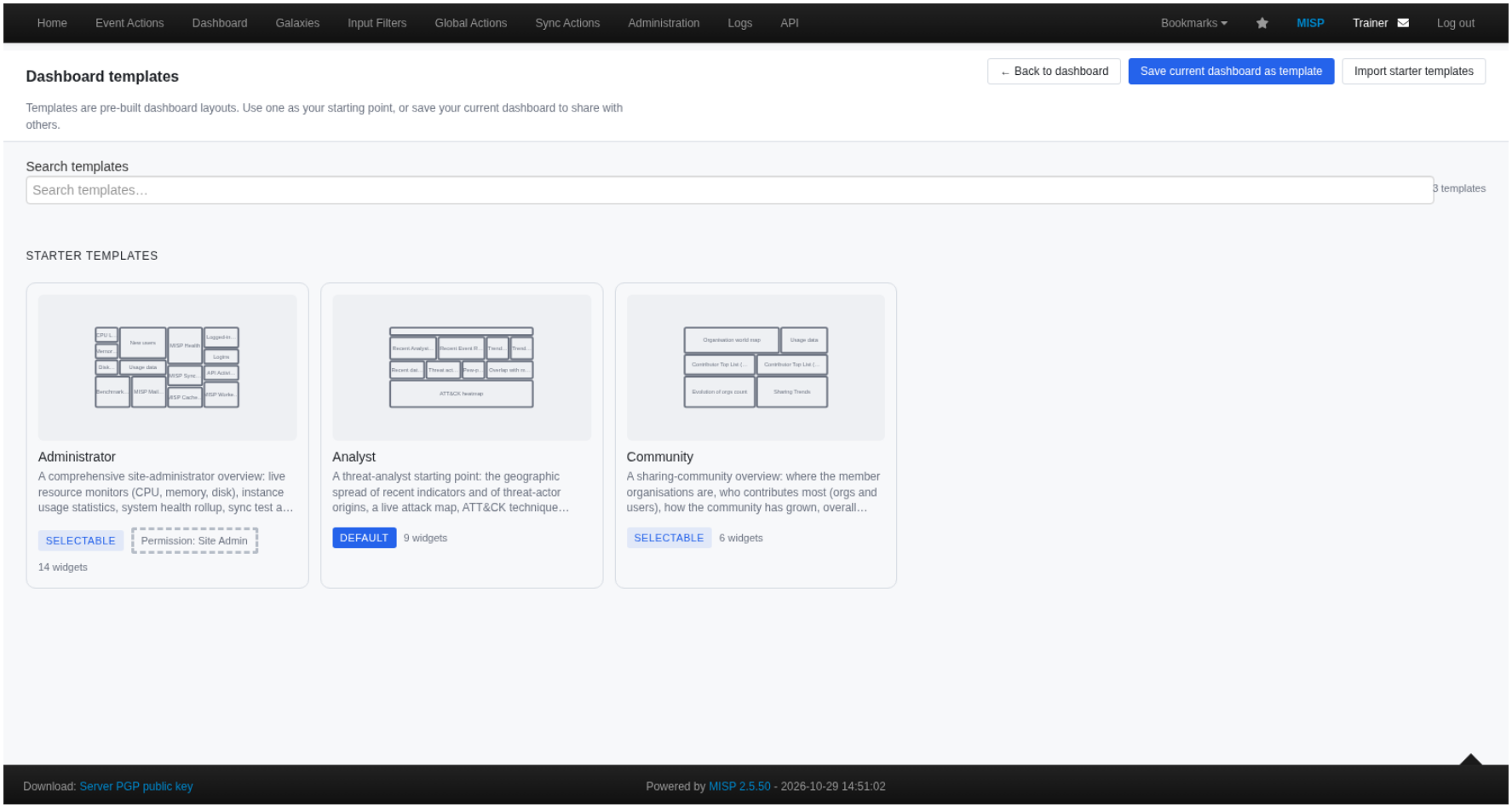
Administrator

Health, resources, queues and permission-gated actions.

A template arranges widgets; it does not grant privileges.

Reuse and verify

Choose a starting point from the template gallery



Actual gallery: imported Analyst, Administrator and Community templates.

Reuse settings without confusing them with data

- Export your configuration before replacing a layout.
- Import a configuration or use an accessible template.
- Save a useful personal view as a template when permitted.
- Review identifiers and configuration before sharing it.

Configuration export and widget-data export differ

Configuration

Widget classes, options and positions for reusing a view.

Widget data

The result produced by a widget, available as JSON or flattened CSV.

Inspect a widget through the API

```
POST /dashboards/renderWidget
```

```
{
  "widget": "TrendingWidget",
  "config": {
    "dimension": "vulnerability",
    "time_window": "P7D",
    "threshold": 5,
    "min_count": 1
  }
}
```

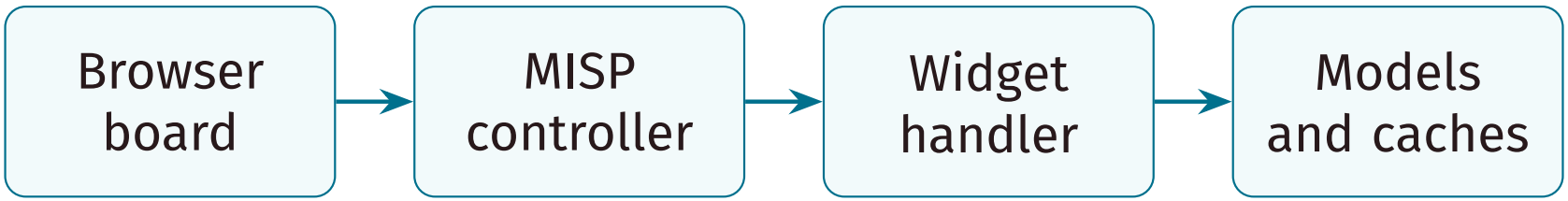
The JSON response identifies the widget, renderer, configuration and data.

What is stored in a dashboard layout?

Field	Purpose
`instance_id`	Stable identifier for one tile.
`widget`	Widget class name.
`config`	Widget-specific settings and supported filters.
`position`	Grid coordinates and dimensions: `x`, `y`, `w`, `h`.

The supplied `examples/dashboard-layout.json` is a portable widget array.

The integrated data path



Widget-specific prerequisites still apply.

Diagnose the scope before rebuilding the board

Symptom	Check first
Widget missing	Class availability, user role and permissions.
Empty or different counts	Widget definition, time window, filters and data access.
Results seem old	Refresh state, timestamps and the widget's cache.
Controls do not work	Browser console/network, JavaScript assets and server errors.

Workshop: build, inspect, restore

1. Use a disposable training account and export its current layout.
2. Import synthetic events and the five-widget sample.
3. Change a filter; inspect the affected widgets and underlying data.
4. Stage a layout removal, discard it, then verify restoration.

Finish with evidence

Export the configuration and explain the scope of each result.

What was verified for this course?

- Real MISP 2.5.50 browser screenshots with synthetic data.
- Layout import/export and widget rendering through the API.
- A staged removal and Discard in the actual browser.
- Slide and notes builds, examples and source references.

Sources and the next practical steps

- MISP 2.5 source and its shipped dashboard templates.
- This course's `README.md`: trainer guide, exercises and immutable source evidence.
- `images/misp25/README.md`: screenshot provenance.
- The standalone ZeroMQ course remains in the training archive.