Managing record access

Karolina Przerwa
CERN

1. Use case
2. Packages
3. Solution structure
4. Development

Use case

Access to the record restricted:

CRUD operation allowed only for the owner of the record.


KEEP
OUT

Permission factory

Function deciding which user has a permission to access the record

```
7   def owner_permission_factory(record=None):
8       """Permission factory with owner access."""
9       return Permission(UserNeed(record["owner"]))
10
```

Coming to the rescue

flask-login
flask-principal

invenio-access
invenio-records-rest

Invenio-access

UserNeed and RoleNeed

Invenio-access

UserNeed and RoleNeed

```
superuser_access = ActionNeed('superuser-access')
"""Superuser access aciton which allow access to everything."""

any_user = SystemRoleNeed('any_user')
"""Any user system role.

authenticated_user = SystemRoleNeed('authenticated_user')
"""Authenticated user system role.
```

Invenio-access

```python
class Permission(_Permission):
    """Represents a set of required needs.

    Extends Flask-Principal's :py:class:`flask_principal.Permission` with
    support for loading action grants from the database including caching
    support.

    Essentially the class works as a translation layer that expands action
    needs into a list of user/roles needs. For instance, take the following
    permission:

    .. code-block:: python

        Permission(ActionNeed('my-action'))

    Once the permission is checked with an identity, the class will fetch a
    list of all users and roles that have been granted/denied access to the
    action, and expand the permission into something similar to (depending
    on the state of the database):

    .. code-block:: python

        Permission(UserNeed('1'), RoleNeed('admin'))
```

```python
Permission(UserNeed(1), RoleNeed('admin'))
```

Configuration

Configuration

```
24  RECORDS_REST_ENDPOINTS = {
25      'recid': dict(
26          pid_type='recid',
27          pid_minter='recid',
28          pid_fetcher='recid',
29          default_endpoint_prefix=True,
30          search_class=RecordsSearch,
31          indexer_class=RecordIndexer,
32          search_index='records',
33          search_type=None,
34          record_serializers={...},
38          search_serializers={...},
42          record_loaders={...},
46          list_route='/records/',
47          item_route='/records/<pid(recid):pid_value>',
48          default_media_type='application/json',
49          max_result_window=10000,
50          error_handlers=dict(),
51          create_permission_factory_imp=allow_all,
52          read_permission_factory_imp=owner_permission_factory,
53          update_permission_factory_imp=allow_all,
54          delete_permission_factory_imp=allow_all,
55          list_permission_factory_imp=allow_all
56      ),
57  }
58  """REST API for my-site."""
```

Configuration

```python
RECORDS_REST_ENDPOINTS = {
    'recid': dict(
        pid_type='recid',
        pid_minter='recid',
        pid_fetcher='recid',
        default_endpoint_prefix=True,
        search_class=RecordsSearch,
        indexer_class=RecordIndexer,
        search_index='records',
        search_type=None,
        record_serializers={...},
        search_serializers={...},
        record_loaders={...},
        list_route='/records/',
        item_route='/records/<pid(recid):pid_value>',
        default_media_type='application/json',
        max_result_window=10000,
        error_handlers=dict(),
        create_permission_factory_imp=allow_all,
        read_permission_factory_imp=owner_permission_factory,
        update_permission_factory_imp=allow_all,
        delete_permission_factory_imp=allow_all,
        list_permission_factory_imp=allow_all
    ),
}
"""REST API for my-site."""
```

```python
RECORDS_UI_ENDPOINTS = {
    'recid': {
        'pid_type': 'recid',
        'route': '/records/<pid_value>',
        'template': 'records/record.html',
        'permission_factory_imp': 'my_site.records.permissions:'
                                  'owner_permission_factory',
    },
}
"""Records UI for my-site."""
```


DENIED

Search filters

```python
from elasticsearch_dsl import Q
from flask_security import current_user
from invenio_search.api import DefaultFilter, RecordsSearch


def owner_permission_filter():
    """Search filter with permission."""
    return [Q('match', owner=current_user.get_id())]
```

Search filters

```python
from elasticsearch_dsl import Q
from flask_security import current_user
from invenio_search.api import DefaultFilter, RecordsSearch


def owner_permission_filter():
    """Search filter with permission."""
    return [Q('match', owner=current_user.get_id())]


class OwnerRecordsSearch(RecordsSearch):
    """Class providing permission search filter."""

    class Meta:
        index = 'records'
        default_filter = DefaultFilter(owner_permission_filter)
        doc_types = None
```

Search filters config

```
17  from my_site.records.permissions import owner_permission_factory
18  from my_site.records.search import OwnerRecordsSearch
```

```
26  RECORDS_REST_ENDPOINTS = {
27      'recid': dict(
28          pid_type='recid',
29          pid_minter='recid',
30          pid_fetcher='recid',
31          default_endpoint_prefix=True,
32          search_class=OwnerRecordsSearch,
33          indexer_class=RecordIndexer
```

TESTS

```python
import uuid

import pytest
from flask_principal import RoleNeed, identity_loaded
from flask_security import login_user
from invenio_accounts.models import User
from invenio_records import Record

from my_site.records.permissions import owner_permission_factory


@pytest.mark.parametrize('owner,action,is_allowed', [
    ({'owner': 1}, 'read', True),

])
def test_record_generic_access(db, users, owner, action, is_allowed):
    """Test access control for records."""

    @identity_loaded.connect
    def mock_identity_provides(sender, identity):
        """Provide additional role to the user."""
        roles = [RoleNeed('test@invenio')]
        # Gives the user additional roles, f.e. based on his groups
        identity.provides |= set(roles)

    def login_and_test(user_id):
        login_user(User.query.get(user_id))
        # Create record
        user = User.query.get(user_id)
        id = uuid.uuid4()
        record = Record.create(owner, id_=id)
        factory = owner_permission_factory(record)

        assert factory.can() if is_allowed else not factory.can()

    # test superuser access
    login_and_test(1)
    # Test owner access
    login_and_test(2)
    # Test visitor access
    login_and_test(3)
```

Create records permission

Use case: Allow creating records only for authenticated users (security reasons, also we can identify the owner of the record)

Create records permission

```python
from invenio_access import Permission, authenticated_user
```

```python
def authenticated_user_permission(record=None):
    """Return an object that evaluates if the current user is authenticated."""
    return Permission(authenticated_user)
```

Create records permission

```python
from invenio_access import Permission, authenticated_user
```

```python
def authenticated_user_permission(record=None):
    """Return an object that evaluates if the current user is authenticated."""
    return Permission(authenticated_user)
```

```python
RECORDS_REST_ENDPOINTS = {
    'recid': dict(
        pid_type='recid',
        pid_minter='recid',
        pid_fetcher='recid',
        default_endpoint_prefix=True,
        search_class=OwnerRecordsSearch,
        indexer_class=RecordIndexer,
        search_index='records',
        search_type=None,
        record_serializers={
            'application/json': ('my_site.records.serializers'
                                 ':json_v1_response'),
        },
        search_serializers={
            'application/json': ('my_site.records.serializers'
                                 ':json_v1_search'),
        },
        record_loaders={
            'application/json': ('my_site.records.loaders'
                                 ':json_v1'),
        },
        list_route='/records/',
        item_route='/records/<pid(recid):pid_value>',
        default_media_type='application/json',
        max_result_window=10000,
        error_handlers=dict(),
        create_permission_factory_imp=authenticated_user_permission,
```

Group access

Use case: Allow a managers group to access the record

Group access

```
2   from flask_principal import UserNeed, RoleNeed
3
```

```
15  def owner_manager_permission_factory(record=None):
16      """Returns permission for managers group."""
17      return Permission(UserNeed(record["owner"]), RoleNeed('managers'))
18
```

Group access filter

```python
from elasticsearch_dsl import Q
from flask_security import current_user
from invenio_search.api import DefaultFilter, RecordsSearch


def owner_manager_permission_filter():
    """Search filter with permission."""
    if current_user.has_role('managers'):
        return [Q(name_or_query='match_all')]
    else:
        return [Q('match', owner=current_user.get_id())]


class OwnerManagerRecordsSearch(RecordsSearch):
    """Class providing permission search filter."""

    class Meta:
        index = 'records'
        default_filter = DefaultFilter(owner_manager_permission_filter)
        doc_types = None
```

Group access config

```
27  RECORDS_REST_ENDPOINTS = {
28      'recid': dict(
29          pid_type='recid',
30          pid_minter='recid',
31          pid_fetcher='recid',
32          default_endpoint_prefix=True,
33          search_class=OwnerManagerRecordsSearch,
34          indexer_class=RecordIndexer,
35          search_index='records',
36          search_type=None,
37          record_serializers={...},
41          search_serializers={...},
45          record_loaders={...},
49          list_route='/records/',
50          item_route='/records/<pid(recid):pid_value>',
51          default_media_type='application/json',
52          max_result_window=10000,
53          error_handlers=dict(),
54          create_permission_factory_imp=authenticated_user_permission,
55          read_permission_factory_imp=owner_manager_permission_factory,
56          update_permission_factory_imp=owner_manager_permission_factory,
57          delete_permission_factory_imp=owner_manager_permission_factory,
58          list_permission_factory_imp=allow_all
59      ),
60  }
61  """REST API for my-site."""
```

Explicit access

Use case: Access to edit the record for specific group, read access for any authenticated user

Explicit access

Use case: Access to edit the record for specific group, read access for any authenticated user

```
{
    "_access": {
        "read": {
            "systemroles": ["campus_user"]
        },
        "update": {
            "users": [1],
            "roles": ["curators"],
        }
    }
}
```

Questions?