Flag Type Plugins

Flag Types

Flag types are used to give developers a way to allow teams to submit flags which do not conform to a hardcoded string or a regex-able value.

The approach is very similar to Challenges with a base Flag/Key class and a global dictionary specifying all the Flag/Key types:

```
class BaseFlag(object):
    name = None
    templates = {}

    @staticmethod
    def compare(self, saved, provided):
        return True


class CTFdStaticFlag(BaseFlag):
    name = "static"
    templates = {  # Nunjucks templates used for key editing & viewing
        "create": "/plugins/flags/assets/static/create.html",
        "update": "/plugins/flags/assets/static/edit.html",
    }

    @staticmethod
    def compare(chal_key_obj, provided):
        saved = chal_key_obj.content
        data = chal_key_obj.data

        if len(saved) != len(provided):
            return False
        result = 0
```

```
        if data == "case_insensitive":
            for x, y in zip(saved.lower(), provided.lower()):
                result |= ord(x) ^ ord(y)
        else:
            for x, y in zip(saved, provided):
                result |= ord(x) ^ ord(y)
        return result == 0


class CTFdRegexFlag(BaseFlag):
    name = "regex"
    templates = {  # Nunjucks templates used for key editing & viewing
        "create": "/plugins/flags/assets/regex/create.html",
        "update": "/plugins/flags/assets/regex/edit.html",
    }

    @staticmethod
    def compare(chal_key_obj, provided):
        saved = chal_key_obj.content
        data = chal_key_obj.data

        if data == "case_insensitive":
            res = re.match(saved, provided, re.IGNORECASE)
        else:
            res = re.match(saved, provided)

        return res and res.group() == provided


FLAG_CLASSES = {"static": CTFdStaticFlag, "regex": CTFdRegexFlag}


def get_flag_class(class_id):
    cls = FLAG_CLASSES.get(class_id)
    if cls is None:
        raise KeyError
    return cls
```

When a challenge solution is submitted, the challenge plugin itself is responsible for:

1. Loading the appropriate Key class using the `get_flag_class()` function.
2. Properly calling the static `compare()` method defined by each Flag class.

3. Returning the correctness boolean and the message displayed to the user.

This is properly implemented by the following code copied from the default standard challenge:

```
@staticmethod
def attempt(challenge, request):
    data = request.form or request.get_json()
    submission = data['submission'].strip()
    flags = Flags.query.filter_by(challenge_id=challenge.id).all()
    for flag in flags:
        if get_flag_class(flag.type).compare(flag, submission):
            return True, 'Correct'
    return False, 'Incorrect'
```

Previous
« Challenge Type Plugins

Next
Imports »

Was this page helpful? 👍 👎

Share your feedback