w.c.s. - Development #36128

Possible trou de sécurité quand une source json est convertie en source jsonp pour un ItemField en autocomplete

14 septembre 2019 22:09 - Benjamin Dauvergne

Statut:	Fermé	**Début:**	14 septembre 2019
Priorité:	Bas	**Echéance:**	
Assigné à:		**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Non	**Planning:**	Non

Description

J'ai réfléchi à ça en lisant le code des conditions live/soumission avec dépendance pour #35903 (ça n'a pas de rapport c'est juste pour dire comment j'en suis arrivé là).

Le code pour la soumission et l'évaluation live dépend de ça :

```
    def get_transient_formdata(self, magictoken=Ellipsis):
        if magictoken is Ellipsis:
            magictoken = get_request().form.get('magictoken')
        # create a fake FormData with current submission data
        formdata = FormData()
        formdata._formdef = self.formdef
        formdata.user = get_request().user
        formdata.data = get_session().get_by_magictoken(magictoken, {}) <-- ICI
```

ceci fonctionne parce que les données en session sont modifiées là pour la soumission :

```
 826             form_data = session.get_by_magictoken(magictoken, {})
 827             with get_publisher().substitutions.temporary_feed(
 828                     transient_formdata, force_mode='lazy'):
 829                 data = self.formdef.get_data(form) <-- obtention des données du formulaire de
puis les données soumises
 830             form_data.update(data) <-- ICI
 831
 832             session.add_magictoken(magictoken, form_data) <-- puis là
```

et là pour l'évaluation live :

```
1093         formdata = self.get_transient_formdata()
1094         get_publisher().substitutions.feed(formdata)
1095         displayed_fields = []
1096         with get_publisher().substitutions.temporary_feed(formdata, force_mode='lazy'):
1097             form = self.create_form(
1098                     page=page,
1099                     displayed_fields=displayed_fields,
1100                     transient_formdata=formdata)
1101         formdata.data.update(self.formdef.get_data(form)) <-- ICI
1102         return FormStatusPage.live_process_fields(form, formdata, displayed_fields)
```

la seule différence c'est que pour la soumission on fait ça avant le get_transient et pour le live on le fait après

dans FormDef.get_data() on appelle FormDef.get_field_data(field, widget), qui fait ça :

```
 684     def get_field_data(self, field, widget):
 685         d = {}
 686         d[field.id] = widget.parse() <-- ICI
 687         if d.get(field.id) is not None and field.convert_value_from_str:
 688             d[field.id] = field.convert_value_from_str(d[field.id])
 689         if d.get(field.id) is not None and field.store_display_value:
 690             display_value = field.store_display_value(d, field.id)
```

```
691             if display_value is not None:
692                 d['%s_display' % field.id] = display_value
693             elif d.has_key('%s_display' % field.id):
694                 del d['%s_display' % field.id]
695         if d.get(field.id) is not None and field.store_structured_value:
696             structured_value = field.store_structured_value(d, field.id)
697             if structured_value is not None:
698                 d['%s_structured' % field.id] = structured_value
699             elif '%s_structured' % field.id in d:
700                 del d['%s_structured' % field.id]
701         if getattr(widget, 'cleanup', None):
702             widget.cleanup()
703         return d
```

la ligne importante est la ligne 686, ça appelle la validation du widget, dans le cas d'un ItemField classique c'est un SingleSelectWidget qui valide expressément ce qui est soumis par rapport à la valeur actuelle des options mais dans le cas d'un ItemField en autocomplete c'est un JsonpSingleSelectWidget qui lui ne valide rien du tout par rapport à sa source de donnée :

```
1898 class JsonpSingleSelectWidget(Widget):
....
1928
1929     def parse(self, request=None):
1930         if request and request.form.get(self.name) and request.form.get(self.name + '_display
'):
1931             # store text value associated to the jsonp value
1932             if not get_session().jsonp_display_values:
1933                 get_session().jsonp_display_values = {}
1934             value = request.form.get(self.name)
1935             display_value = request.form.get(self.name + '_display')
1936             get_session().jsonp_display_values['%s_%s' % (self.url, value)] = display_value
1937
1938         return Widget.parse(self, request=request)
```

on extrait juste la valeur "_display" pour la mettre en session puis plouf on appelle Widget.parse() qui prend la valeur qui arrive sans rien valider.

Si on imagine une source JSON privée, convertible (parce que paramètre "q") et convertie en JSON pour le besoin du formulaire il me semble que rien n'empêchera la soumission; cela provoquera quelques effets de bord comme l'absence de display_value/structured_value mais si la valeur (l'id) est réutilisé plus loin sans validation (genre ma source JSON c'est la liste des élèves d'une école, c'est limité à l'école de la directrice qui soumet le formulaire, et bien si plus loin on accède à une donnée dépendant de l'identifiant de l'élève de fait elle aura accès à tous les élèves de la base).

Si le bug est avéré (il faudrait que je ponde un test pour ça bien sûr) une correction me semblerait dans le cas d'une source JSON convertie en JSONP de faire en sorte que JsonpSingleSelectWidget valide toujours que cette valeur fait bien partie de la source de donnée (on devrait lui fournir en plus de l'URL la source de donnée réelle pour qu'il puisse vérifier via data_source.get_structured_value(id)).

Pour les sources JSONP pures on peut considérer qu'elles sont toujours publiques et donc c'est moins grave (c'est juste gênant parce qu'à priori on peut soumettre une rue qui n'existe pas dans la ville sélectionnée par exemple, mais comme on aura pas de display_value je suppose que ça affichera juste un truc vide, de toute façon le JSONP pure c'est une mauvaise idée).

Historique

#1 - 17 janvier 2024 20:18 - Frédéric Péters

- Statut changé de Nouveau à Fermé

> (...) mais si la valeur (l'id) est réutilisé plus loin sans validation (...)

Via #50288 le set_value() (utilisé par get_field_data() depuis #51207) est spécialisé pour l'ItemField et vérifie désormais la donnée reçue (sauf pour le cas "pur jsonp" où ça n'est pas possible).