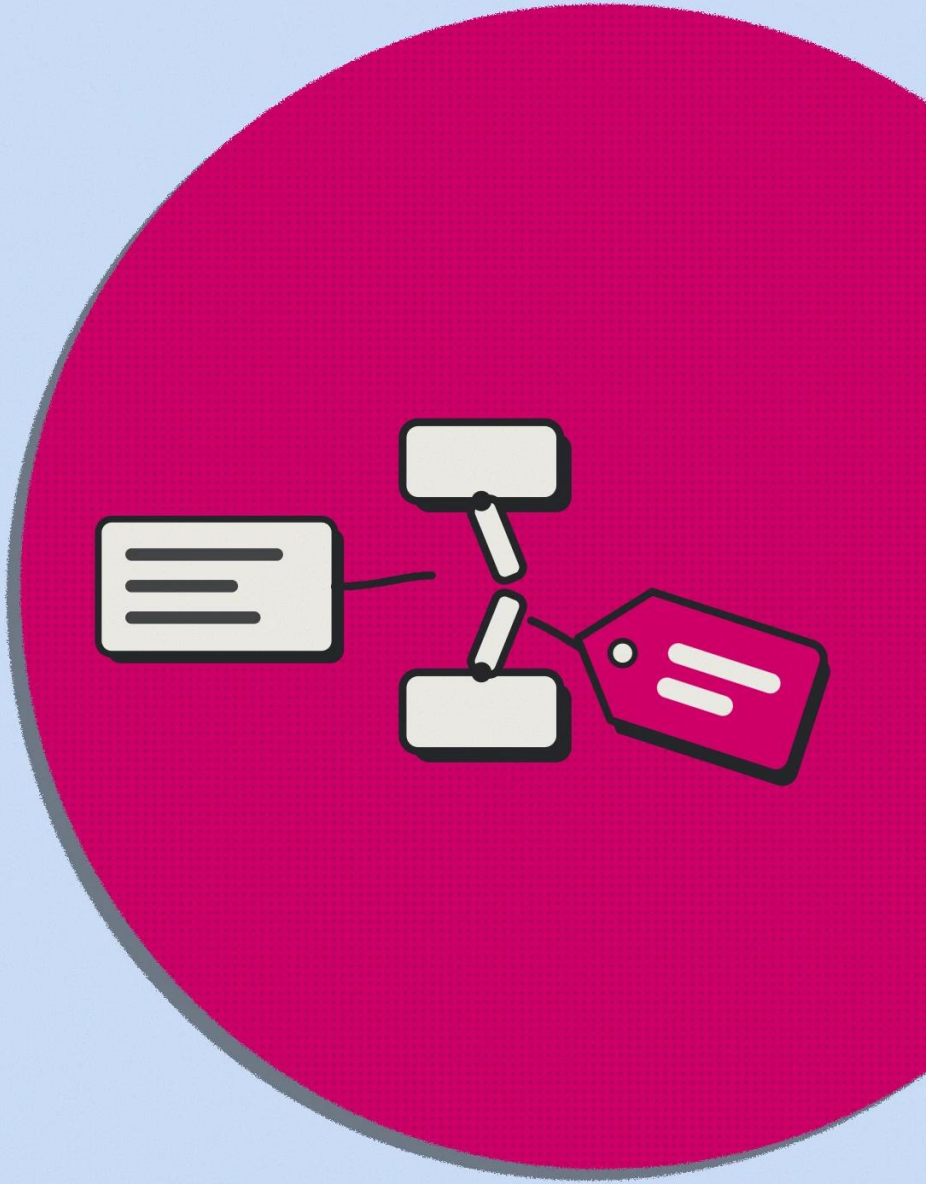
2.08

TEMPS 2 · LE TAMAGOTCHI TROUÉ · SÉANCE 2

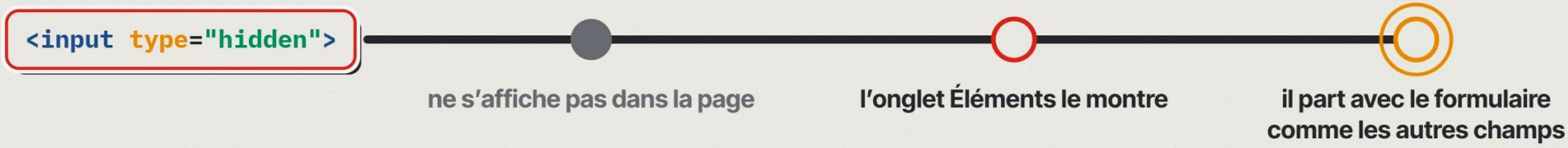
Le prix du navigateur

Thibault Six · IFOSUP Wavre · 2026-2027



? Un champ de formulaire que la page n'affiche pas : peut-on quand même changer sa valeur ?

Un champ caché ne s'affiche pas, mais il part avec le formulaire, et sa valeur se change.



CWE-472

le CWE, un catalogue des défauts de logiciel

une application qui croit qu'un paramètre ne peut pas changer alors que le visiteur le contrôle

plusieurs paniers d'achat de l'année 2000 laissaient changer le prix par un champ caché

OWASP

une fondation qui publie des guides sur la sécurité des applications web

un coût de 70.00 dans un champ caché, qu'un attaquant baisse avant d'envoyer le formulaire

Le navigateur cache le champ à l'œil, pas à qui ouvre les outils de développement. Le magasin du Tamagotchi fait la même chose.

? D'où vient chaque nombre de $total : du navigateur ou de la base ?

Le total vient du prix et de la quantité envoyés par la page : l'article en base n'est pas lu.

resources/js/pages/shop/Index.vue

```
<input
    type="hidden"
    name="price_cents"
    :value="item.price_cents"
/>
```

extrait

min="1"

LE FORMULAIRE ENVOIE

creature_id

quantity

price_cents

app/Http/Controllers/PurchaseController.php

```
public function store(Request $request, Item $item): RedirectResponse
```

chargé depuis la base par la liaison de modèle, avec son vrai prix : le contrôleur ne le lit pas

Laravel charge l'article d'après l'adresse

```
// …
$quantity = $request->integer('quantity', 1);
$price = $request->integer('price_cents');
$total = $price * $quantity;
```

extrait

Le prix est caché, et le champ « Quantité » refuse un nombre inférieur à 1.
Mais ces deux protections vivent dans le navigateur de Mallory, et elle peut les modifier.

Remontez chaque nombre de $total jusqu'à sa source : les deux viennent de la page. L'article chargé depuis la base, lui, n'est pas lu.

? Mallory a 5 €, et le Coussin doré coûte 19,90 €. Comment le paie-t-elle quand même ?

Dans l'onglet Éléments, Mallory remplace 1990 centimes par 1 et paie le Coussin doré 0,01 €.



sa créature : Encre

onglet Éléments · Edit as HTML

```
<input type="hidden" name="price_cents" value="1990">

<input type="hidden" name="price_cents" value="1">
```

Rien ne change à l'écran : l'étiquette affiche toujours 19,90 €.

ONGLET RÉSEAU

```
POST /magasin/7/acheter
Content-Type: application/json

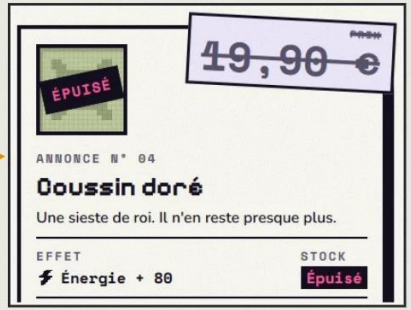
{"price_cents":"1","creature_id":"3","quantity":"1"}
```

```
Réponse : 302
Location: http://…/magasin
```

l'en-tête de Mallory

SOLDE 4,99 €

Achat effectué pour Encre.



le Coussin doré est marqué « Épuisé »

Une seule valeur change, dans la page de Mallory. Le serveur accepte la requête, débite un centime et vide le stock.

? Le champ Quantité refuse -5. Comment Mallory l'envoie-t-elle quand même ?

Mallory efface `min="1"`, envoie -5 croquettes, et l'achat la crédite au lieu de la débiter.

AVEC MIN="1"

-5 → 

Cette valeur doit être supérieure ou égale à 1. **aucune requête ne part**

Cette vérification est faite par le navigateur, à partir de l'attribut.

ONGLET ÉLÉMENTS · EDIT AS HTML

```
<input id="quantity-2" name="quantity" type="number"> min="1"
```



```
POST /magasin/2/acheter
Content-Type: application/json

{"price_cents":"250","creature_id":"3","quantity":"-5"}
```

de 50 en stock
à 55 en stock

STOCK
55 en stock

l'en-tête de Mallory

SOLDE 17,49 €



En haut, le navigateur bloque -5 tant que l'attribut est là. En bas, l'attribut retiré, la requête part et le serveur l'accepte.

? Avec une quantité de -5, que calcule le contrôleur, ligne après ligne ?

Avec -5, le total vaut -1250 : aucun contrôle ne bloque, et Mallory gagne 12,50 €.

APRÈS LE COUSSIN DORÉ

Mallory avait 499 centimes, et Encre 30 points de nourriture.

app/Http/Controllers/PurchaseController.php

```
$total = $price * $quantity;                          → -1250
if ($item->stock < $quantity) {                       → 50 < -5 ? non, ça passe
if ($user->balance_cents < $total) {                  → 499 < -1250 ? non, ça passe
$item->update(['stock' => $item->stock - $quantity]); → 55
$user->balance_cents -= $total;                       → 1749
```

extrait

Encre
la jauge de nourriture
-70

Une jauge va de 0 à 100.

Soustraire un nombre négatif revient à ajouter : Mallory est payée pour acheter.

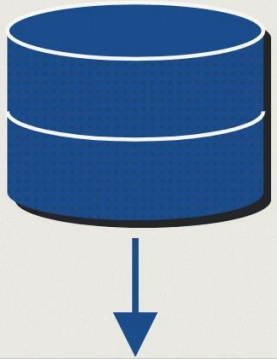
Suivez -5 de ligne en ligne : chaque contrôle compare et laisse passer. La soustraction finale ajoute 1250 centimes au solde.

? Sur votre copie, que garde la base après vos deux attaques ?

À vous, cinq minutes : refaites les deux attaques sur votre copie, puis lisez la base.

5 min À FAIRE

1. Lancez `php artisan migrate:fresh --seed`, connectez-vous comme Mallory et ouvrez « 04 Magasin ».
2. Ouvrez l'onglet Réseau. Dans l'onglet Éléments, changez la valeur du champ `price_cents` du Coussin doré en 1, puis achetez-le.
3. Tapez `-5` dans la quantité des Croquettes et cliquez sur « Acheter » : notez ce que fait le navigateur.
4. Retirez `min="1"` de ce champ dans l'onglet Éléments, puis achetez `-5` croquettes.
5. Lancez la commande Tinker et lisez les deux lignes d'achat.



```
$ php artisan tinker --execute="echo App\Models\Purchase::all(['item_id', 'quantity', 'total_cents'])->toJson();"
[{"item_id":7,"quantity":1,"total_cents":1},{"item_id":2,"quantity":-5,"total_cents":-1250}]
```

Relancez `php artisan migrate:fresh --seed` à la fin.

Cinq étapes à gauche, la base à droite : le premier achat a coûté un centime, le second a rapporté 12,50 €.

? Le prix doit-il venir de la requête ? Et la quantité, qui la choisit ?

Le prix se lit sur l'article en base, et une règle de validation borne la quantité de 1 à 10.

app/Http/Controllers/PurchaseController.php

```php
$quantity = $request->integer('quantity', 1);
$price = $request->integer('price_cents');
$total = $price * $quantity;
```

extrait

chargé depuis la base, avec son vrai prix

`Item $item`

`$request->validate()` applique des règles directement, dans la méthode

VOUS DEVEZ OBTENIR

Le total se calcule avec le prix de l'article lu dans la base.
Un `price_cents` envoyé par la requête n'est plus lu.

Le formulaire n'envoie plus `price_cents`. La page reçoit seulement `price`, le texte formaté pour l'affichage.
`price` : depuis le chapitre 2.07, le prix déjà écrit par `Money::format()`

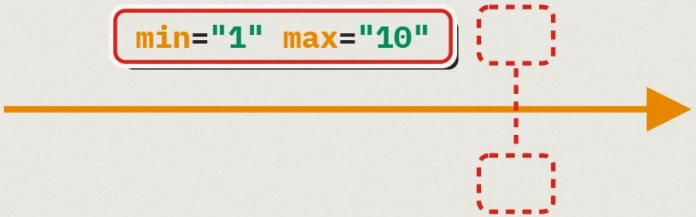
Un dernier test de la faille vérifie aussi que Mallory ne peut acheter que pour ses propres créatures.

À gauche, la ligne qui croit la page ; à droite, ce que vous devez obtenir. La quantité suit, slide suivante.

? Après la correction, la page porte min="1" et max="10". Protègent-ils quelque chose ?

Les attributs de la page aident le joueur honnête ; seule une règle du serveur protège.

DANS LA PAGE : DU CONFORT



Le champ « Quantité » porte `max="10"`, pour que le navigateur aide le joueur honnête.

Un clic droit dans l'onglet Éléments les retire.

DANS LARAVEL : LA PROTECTION



La quantité est un entier de 1 à 10. Toute autre valeur est refusée avec un message sous le champ, sans rien débiter.

Elle tourne sur le serveur, à chaque requête, quelle que soit la page qui l'a envoyée.

En haut, une barrière que Mallory retire d'un clic droit ; en bas, celle qu'elle ne touche pas, qui répond par un message sous le champ.

? Qu'est-ce qui doit être vrai à la fin ?

À vous : lisez le prix dans la base, bornez la quantité, jusqu'aux six tests verts de `Flaw08`.

À FAIRE

1. Cherchez d'abord dans `PurchaseController@store`.
2. Cherchez ensuite dans `ShopController@index`, qui envoie `price_cents` à la page, et dans `shop/Index.vue`.

Cherchez dix minutes, puis ouvrez la correction.

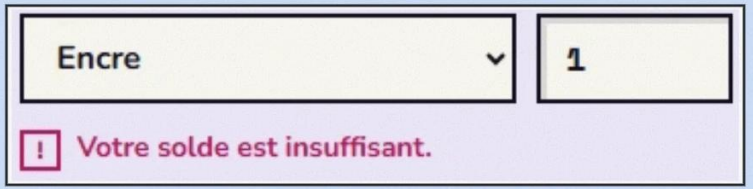
À VÉRIFIER

```
$ php artisan test --filter=Flaw08
  Tests:    6 passed (27 assertions)
```

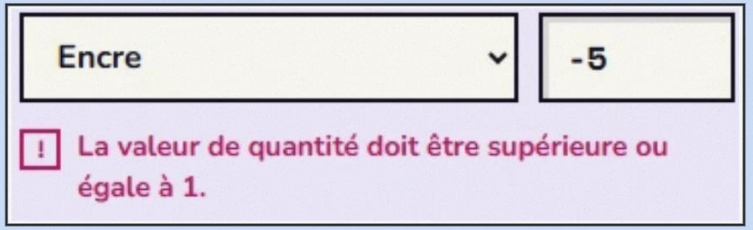
```
$ php artisan test
  Tests:    2 failed, 87 passed (365 assertions)
```

L'ATTAQUE REJOUÉE

`"price_cents":"1"` envoyé : le serveur ne lit plus cette clé, il calcule 1990 centimes



`"quantity":"-5"` envoyé : la règle la refuse



DANS VOTRE CARNET DES FAILLES

Le navigateur dit quel article et combien, jamais ce que ça coûte : le serveur relit le prix dans sa base et borne chaque valeur reçue par une règle de validation.

Deux étapes et les tests à gauche ; à droite, les deux attaques rejouées et refusées. La règle va dans votre carnet, mot pour mot.