# Projet RT0706

## Table des matières

# Access control vulnerabilities

## Challenge 1 - Unprotected admin functionality

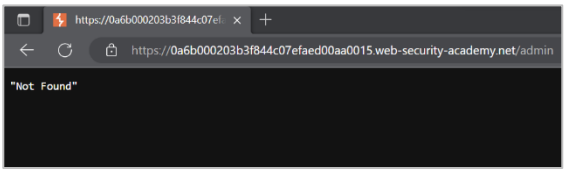
### *Contexte*

Le site a une page d'administrateurs non protégée. Le but étant de supprimer l'utilisateur Carlos. Quand on lance le challenge, une page web de vente s'affiche.

### *Vulnérabilité / problème*

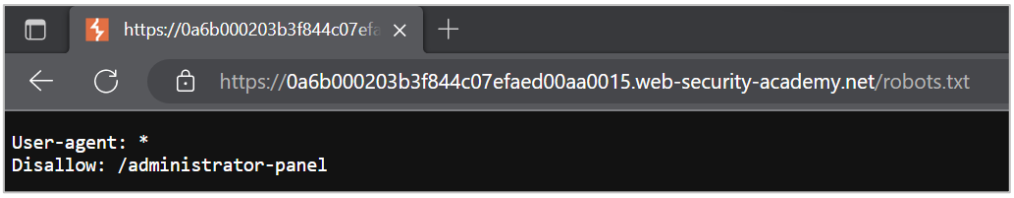
D'après la documentation, nous avons testé dans un premier temps l'escalade verticale des privilèges. L'escalade verticale des privilèges est quand les fonctions administratives ne sont pas seulement accessibles à partir de la page d'accueil d'un administrateur, mais également à partir de la page d'accueil d'un utilisateur.

### *Exploitation*




Nous avons testé d'afficher une potentielle page d'administrateur avec l'url suivante : https://0a6b000203b3f844c07efaed00aa0015.web-security-academy.net/admin
mais la page n'était pas référencée.

Nous avons donc regardé le contenu du fichier `robots.txt` . En effet, dans certains cas, l'URL d'administration peut être divulguée à d'autres emplacements comme ce dernier fichier. Nous avons rentré l'URL suivante :
https://0a6b000203b3f844c07efaed00aa0015.web-security-academy.net/robots.txt
Ce fichier nous apprend qu'une page est non indexée :




Nous avons donc essayé d'accéder à cette dernière page avec l'URL :
https://0a6b000203b3f844c07efaed00aa0015.web-security-academy.net/administrator-panel
Cette page nous donne accès à des fonctionnalités d'administrateurs, telles que supprimer l'utilisateur Carlos.


### *Conseils de corrections*

Il ne faut pas utiliser le fichier `robots.txt` pour empêcher l'affichage des informations sensibles (comme la possibilité de supprimer des utilisateurs en tant que simple utilisateur) dans les résultats

des moteurs de recherche. Il serait nécessaire d'utiliser une protection par mot de passe ou la directive `meta noindex`.

## Challenge 2 - Unprotected admin functionality with unpredictable URL

### *Contexte*

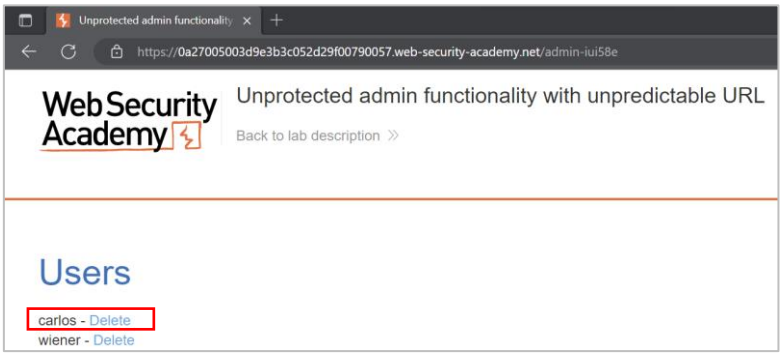
Le site a une page d'administrateurs non protégée. Le but étant de supprimer l'utilisateur Carlos. Quand on lance le challenge, une page web de vente s'affiche.

### *Vulnérabilité / problème*

Dans certains cas, les fonctionnalités sensibles ne sont pas protégées de manière robuste mais sont dissimulées en leur donnant une URL moins prévisible : ce qu'on appelle la sécurité par obscurité. Le simple fait de masquer des fonctionnalités sensibles ne fournit pas un contrôle d'accès efficace car les utilisateurs peuvent toujours découvrir l'URL obscurcie de différentes manières.

### *Exploitation*

Après avoir ouvert les outils de développement du navigateur, dans l'onglet « Sources » le fichier « `index` » nous avons remarqué du code Javascript. En regardant dans la condition « `if (isAdmin)` » nous avons remarqué la ligne « `adminPanelTag.setAttribute('href', '/admin-iui58e');` ». Cette page devrait être accessible si la condition est `isAdmin` vérifiée, c'est-à-dire si nous étions administrateur. Nous avons donc rentré l'URL qui ne devrait être visible que pour les administrateurs : https://0a27005003d9e3b3c052d29f00790057.web-security-academy.net/admin-iui58e
Ainsi, nous avons eu accès à la page d'administration permettant de supprimer les utilisateurs :




### *Conseils de corrections*

Il serait souhaitable de ne pas stocker des informations sensibles côté client, ainsi que de faire attention aux informations visibles dans les scripts. La page devrait être accessible par une authentification. Nous pouvons envisager un langage côté serveur plutôt que côté client.

## Challenge 3 - User role controlled by request parameter

### *Contexte*

Ce laboratoire dispose d'une page d'administration accessible en rajoutant « `/admin` » à la fin de l'URL du site web du challenge. Cette page identifie les administrateurs à l'aide d'un cookie falsifiable. Le but de ce défi est d'accéder au panneau d'administration et de supprimer l'utilisateur Carlos. Nous avons accès à notre propre compte (identifiant : `wiener`, mot de passe `peter`).

### *Vulnérabilité / problème*

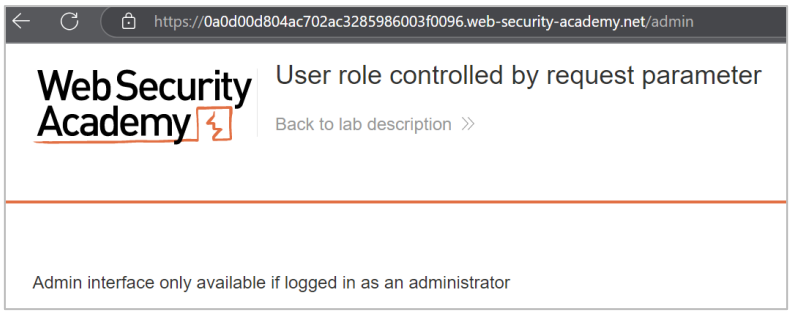
Certaines applications déterminent les droits d'accès ou le rôle de l'utilisateur lors de la connexion, puis stockent ces informations dans un emplacement contrôlable par l'utilisateur, tel qu'un champ masqué, un cookie ou un paramètre de chaîne de requête prédéfini. L'application prend des décisions de contrôle d'accès ultérieures en fonction de la valeur soumise. Cette approche est

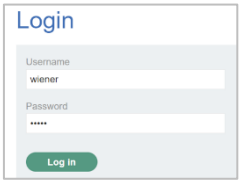
fondamentalement non sécurisée car un utilisateur peut simplement modifier la valeur et accéder à des fonctionnalités auxquelles il n'est pas autorisé, telles que les fonctions administratives.

*Exploitation*

Dans un premier temps nous avons, comme énoncé dans le sujet, essayé d'accéder à la page administrative avec l'URL : https://0a0d00d804ac702ac3285986003f0096.web-security-academy.net/admin

La page ne nous est pas accessible puisque nous ne sommes pas logués en tant qu'administrateur :



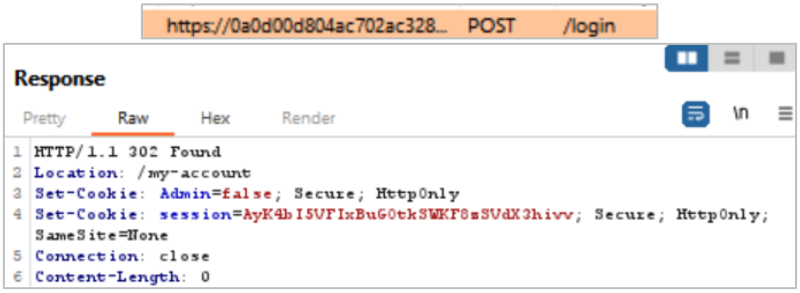




Nous avons donc essayé de nous connecter grâce au formulaire accessible depuis la page « My account ».

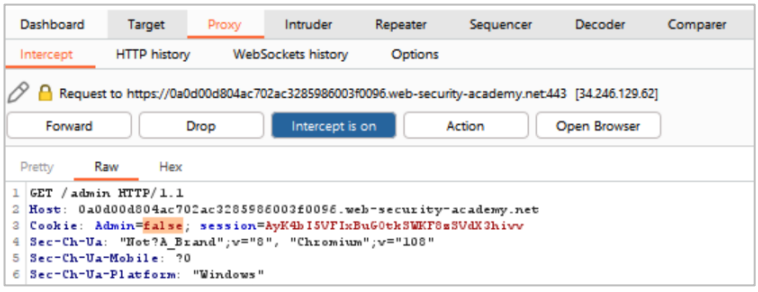
Une fois connectés l'URL est devenue : https://0a0d00d804ac702ac3285986003f0096.web-security-academy.net/my-account?id=wiener

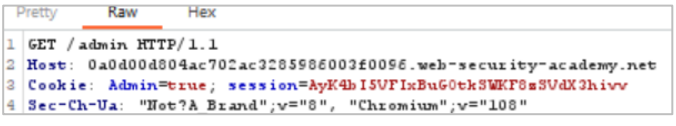
Nous avons essayé de modifier la valeur du champ « `id` » avec « `admin` / `administrator` » mais sans grand succès.

Nous avons donc utilisé Burp Suite Community pour intercepter les échanges et voir s'il était possible de modifier des valeurs afin de nous connecter en tant qu'administrateur. Une fois reconnectés dans le navigateur web de Burp, nous avons lancé l'intercepteur et loader la page `/admin`. Nous avons remarqué que la réponse de la requête avait un cookie avec comme paramètre `Admin=false`.

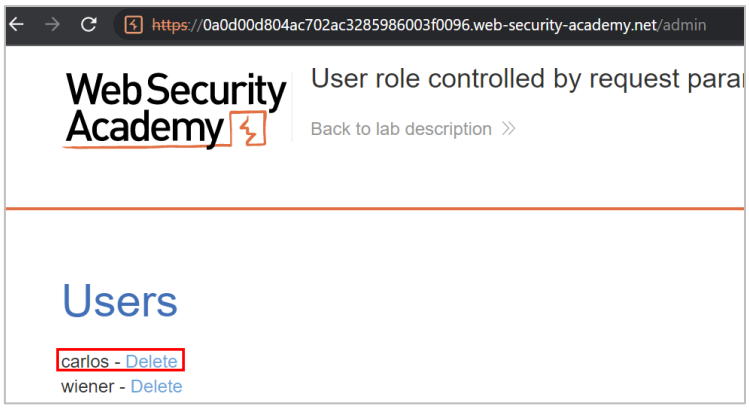



Donc nous avons voulu changer la réponse de cette requête en changeant la valeur du paramètre Admin à vrai. Pour cela, nous nous sommes délogués et relogués. Avant de passer l'intercepteur à OFF, nous avons changé la valeur du cookie présent dans la réponse.




Cela nous a permis d'afficher la page d'admin avec l'URL donné précédemment.




*Conseils de corrections*

Il serait souhaitable de vérifier la valeur, côté serveur, de la session et non la valeur du paramètre Admin qui est trop facilement modifiable.

## Challenge 4 - URL-based access control can be circumvented

*Contexte*

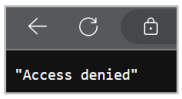
D'après l'énoncé, le contrôle d'accès basé sur les URL peut être contourné. Ce site Web dispose d'une page d'administration accessible sans authentification : la page `/admin`. Cependant, un système frontal a été configuré pour bloquer l'accès externe à ce chemin. Toutefois, l'application principale repose sur une infrastructure qui prend en charge le header X-Original-URL.

Le but étant de supprimer l'utilisateur Carlos.

*Vulnérabilité / problème*

Certaines applications appliquent des contrôles d'accès au niveau de la couche de plate-forme en limitant l'accès à des URL et des méthodes HTTP spécifiques en fonction du rôle de l'utilisateur. Si un site Web utilise des contrôles frontaux rigoureux pour restreindre l'accès en fonction de l'URL, mais que l'application permet de remplacer l'URL via un en-tête de requête, il peut être possible de contourner les contrôles d'accès à l'aide d'une requête : `X-Original-URL`

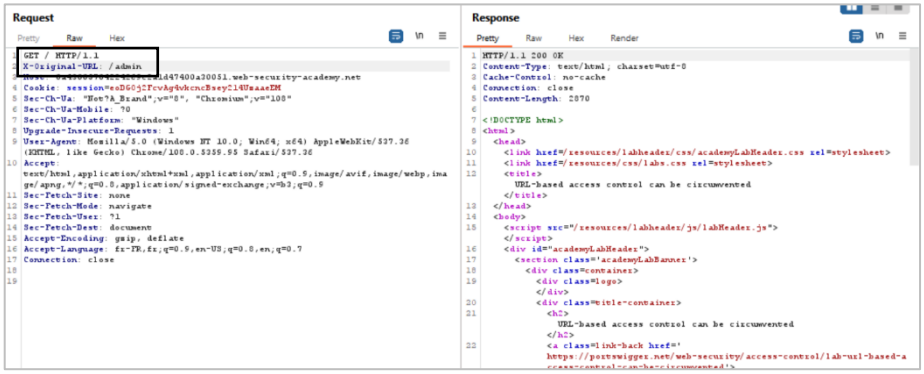
*Exploitation*

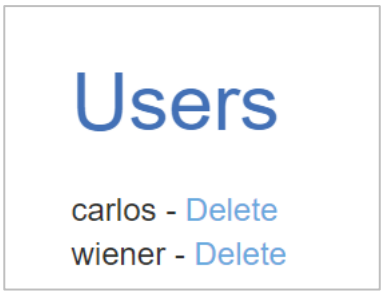



Dans un premier temps nous avons testé l'url : https://0a43005704224269c2a1d47400a30051.web-security-academy.net/admin mais l'accès a été refusé.

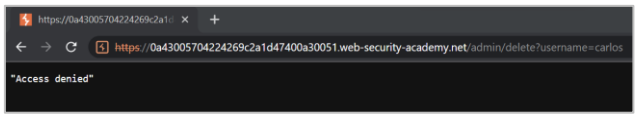
Grâce à l'indice du titre du challenge, nous avons décidé d'intercepter les requêtes avec Burp Community afin d'observer les URL dans les champs échangés. Quand nous avons intercepté la requête pour afficher la page `/admin`, voici ce que nous avons eu :


Nous avons donc testé de réécrire l'URL avec le champ `X-Original-URL` et de mettre seulement « / » sur la ligne GET. Nous avons remarqué que cela fonctionnait. En effet, après avoir ouvert la page HTML renvoyée (partie de droite) dans le navigateur nous avons bien la page souhaitée qui s'affichait.

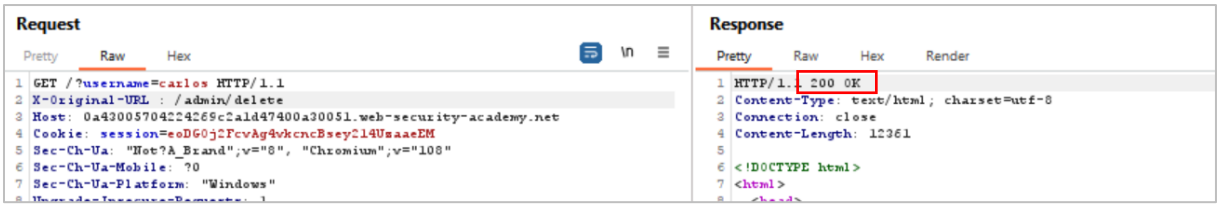


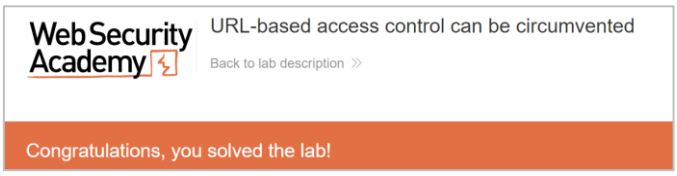

Cependant, quand nous avons souhaité valider le challenge en cliquant sur « Carlos – Delete » une erreur est apparue.




Nous avons donc, à l'aide de l'URL, modifié la requête que nous envoyions. En effet, nous avons remarqué qu'un sous dossier « `/delete` » était présent et qu'il fallait récupérer « `username=carlos`». Cela nous a permis de comprendre, après quelques essais, qu'il fallait mettre après le GET « `?=username=carlos` » et dans le champ `X-Original-URL : /admin/delete`




Cela nous a permis de supprimer l'utilisateur Carlos et donc de valider le challenge.




*Conseils de corrections*

Certains hébergeurs permettent de désactiver le service de réécriture des URL comme l'IIS7. Dans les outils d'administration du serveur (Windows) se rendre dans l'interface de configuration. Il faut sélectionner le ou les sites où nous souhaitons désactiver la réécriture et modifier leur fichier web.config. Il suffit de commenter la ligne autorisant la réécriture.

## Business-logic-vulnerabilities

### Challenge 1 - Excessive trust in client-side controls

*Contexte*

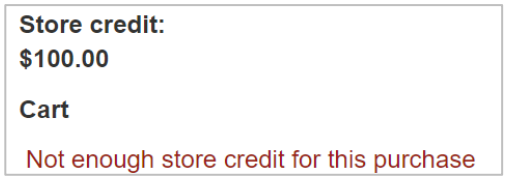
Cet atelier ne valide pas correctement les entrées des utilisateurs. Il faut donc exploiter une faille dans le flux d'achat pour acheter des articles à un prix non conventionnel. Pour résoudre ce challenge, il faut acheter l'article suivant : Lightweight l33t leather jacket. Comme pour le challenge précédent, nous avons comme accès notre propre compte (identifiant : `wiener`, mot de passe `peter`).

### *Vulnérabilité / problème*

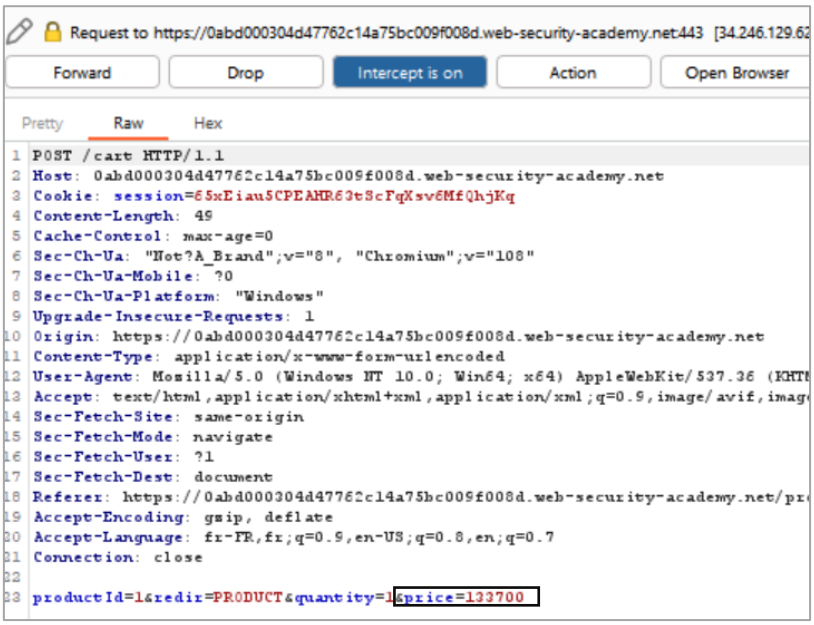
Des failles dans la logique peuvent permettre aux attaquants de contourner les règles. Par exemple, l'utilisateur peut être en mesure d'effectuer une transaction sans passer par le flux de travail d'achat prévu. Cela signifie que lorsqu'un attaquant s'écarte du comportement attendu de l'utilisateur, l'application ne prend pas les mesures appropriées pour empêcher cela et, par la suite, ne parvient pas à gérer la situation en toute sécurité.

### *Exploitation*

Pour ce challenge, nous avons utilisé Burp. Nous avons lancé le navigateur avec l'URL du challenge. Dans un premier temps, nous nous sommes connectés avec les logins fournis. Nous avons remarqué que nous avions 100€ de crédit. Nous avons ensuite rajouté dans notre panier l'article demandé dans l'énoncé. Cet article coûte 1337$, nous n'avons donc pas assez d'argent sur notre compte pour poursuivre la transaction.

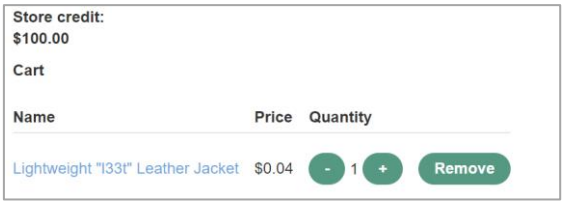



Nous avons enlevé l'article de notre panier et lancé l'intercepteur de Burp pour voir les requêtes. Nous avons rajouté de nouveau l'article en question dans notre panier et regarder la requête engendrée par cette action :

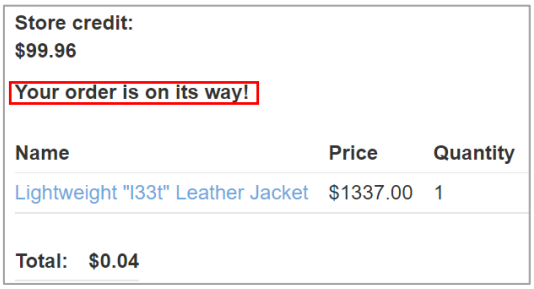



Nous avons remarqué que le prix était affiché dans la requête et qu'il était possible de le modifier par une valeur moindre.


Nous avons arrêté l'intercepteur sur Burp. Une fois retournés dans notre onglet panier, nous avons remarqué que le prix retenu était bien celui que nous avions renvoyé.



Nous avons pu valider le challenge en procédant à la transaction puisque nous avions ainsi assez de crédit.




*Conseils de corrections*

Le prix ne devrait pas être fixé/donné par la requête mais être stocké et géré sur le serveur. En effet, la requête devrait seulement récupérer l'ID de l'article pour qu'une requête SQL puisse retrouver le prix associé dans la base de données. Ainsi, il ne serait pas possible de modifier le prix.

## Challenge 2 - High-level logic vulnerability

*Contexte*

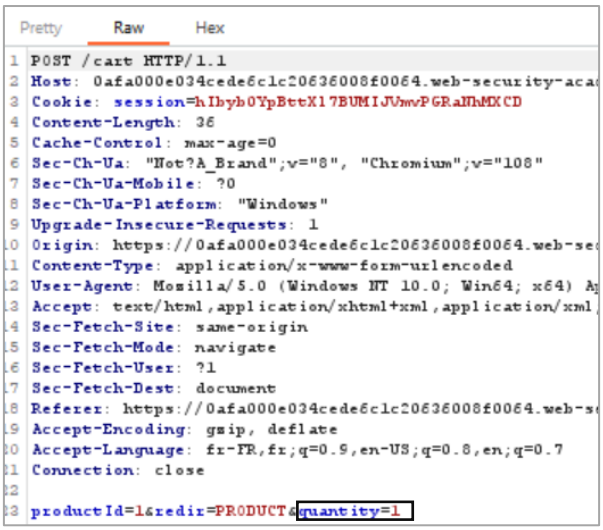
Cet atelier ne valide pas correctement les entrées des utilisateurs. Il faut donc exploiter une faille dans le flux d'achat pour acheter des articles à un prix non conventionnel. Pour résoudre ce challenge, il faut acheter l'article suivant : Lightweight l33t leather jacket. Comme pour le challenge précédent nous avons comme accès notre propre compte ( identifiant : `wiener`, mot de passe `peter`).

*Vulnérabilité / problème*

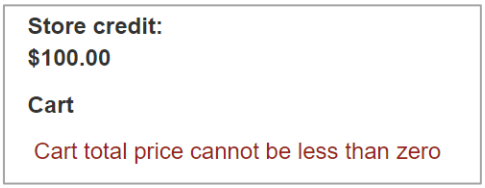
Ce challenge a une faille semblable au challenge précédent. En effet, nous avons à faire à une faille de Flaw Logique également qui permet aux attaquants de contourner les règles. Par exemple, l'utilisateur peut être en mesure d'effectuer une transaction sans passer par le flux de travail d'achat prévu.

*Exploitation*

Pour ce challenge, nous avons également utilisé Burp. Nous avons lancé le navigateur avec l'URL du challenge. Dans un premier temps, nous nous sommes connectés avec les logins fournis. Nous avons remarqué que nous avions 100€ de crédit. Avant de rajouter à notre panier l'article demandé, nous avons lancé l'intercepteur de Burp. Nous n'avons plus accès au prix comme au challenge précédent. Cependant nous avons accès à la quantité.

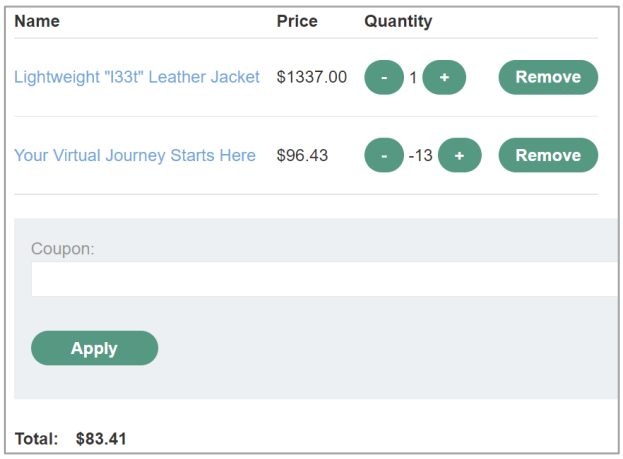



Une idée est de passer cette quantité en négatif. Nous l'avons donc passée à -4. Nous remarquons que dans notre panier, le prix de la transaction est de -4011$. Nous avons essayé de passer la transaction pour voir si l'argent nous serait « remboursé ». Mais une sécurité vérifie que la commande ne soit pas négative.

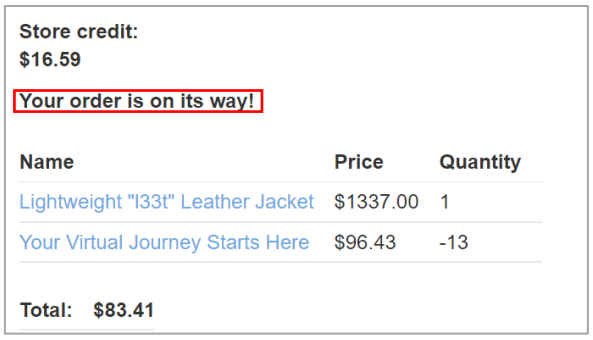



Cependant le fait qu'il y ait un nombre négatif ne semble pas poser de problème. Nous avons donc modulé nos achats afin d'avoir un total inférieur à 100$ :

- Nous avons ajouté 1X : Lightweight l33t leather jacket
- Nous avons ajoutés -13X : Your Virtual Journey Starts Here, en suivant la méthode testée juste au-dessus.




Nous avons validé la commande et nous avons ainsi pu valider le challenge.




*Conseils de corrections*

La quantité ne devrait pas être fixée/donnée par la requête mais être stockée et gérée sur le serveur. En effet, la requête devrait seulement récupérer l'ID de l'article pour qu'une requête SQL puisse retrouver le nombre d'article correspondant à cet id. La quantité du stock devrait être vérifiée et ne jamais être négative dans la base de données côté serveur. Cela implique qu'une vérification sur le serveur des quantités devrait avoir lieu. Il ne devrait pas être possible de commander un article en quantités négatives. La page web ne devrait être qu'affichage et ne pas avoir autant d'impact sur les commandes qui devraient être gérées sur le serveur.

## Challenge 3 - Inconsistent security controls

*Contexte*

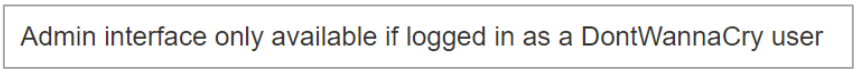
Ce challenge nous permet de travailler sur une vulnérabilité logique qui permet aux utilisateurs d'accéder à des fonctionnalités administratives qui devraient seulement être disponibles pour les employés de l'entreprise. Pour résoudre ce challenge, il faut accéder à la page d'administration et supprimez Carlos.

*Vulnérabilité / problème*

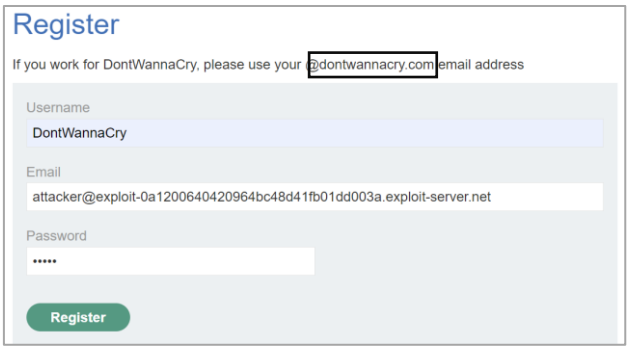
Ce challenge a une faille semblable au challenge précédent. En effet, nous avons affaire à une faille de Flaw Logique également qui permet aux attaquants de contourner les règles. Trop d'informations sont données qui permettent de comprendre et donc de contourner la sécurité mise en place sur le site.

*Exploitation*

Nous avons essayé dans un premier temps d'afficher la page :
https://0ae3005504879 6bdc4d33fee001d001c.web-security-academy.net/admin
Un message s'affiche indiquant qu'il faut être connecté en tant que l'utilisateur DontWannaCry.

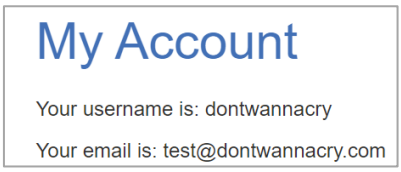



Nous avons essayé de nous créer un utilisateur avec ce nom. Pour cela, nous avons rentré le nom d'utilisateur que nous souhaitions (DontWannaCry) et utilisé l'adresse mail fournie, sur la page accessible par le bouton « Email client », pour créer notre compte. Nous ne pouvions pas directement utiliser @dontwannacry.com puisque nous n'avons pas accès à cette boîte mail et nous n'aurions pas pu valider notre compte.

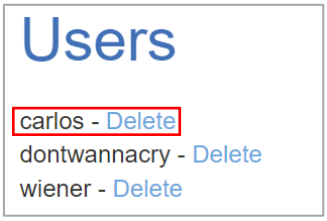



Il nous était demandé de confirmer notre compte en cliquant sur le mail.


Nous avons suivi les indications données dans le mail, ce qui a permis de valider notre compte. Nous nous sommes connectés avec ce nouveau compte. Une fois que nous étions connectés, le site nous a proposé de mettre à jour notre adresse électronique. Ce que nous avons fait en remplaçant notre ancien email par celui-ci : test@dontwannacry.com, puisqu'il répond au format des mails des personnes travaillant pour l'entreprise du site.




Nous avions donc maintenant le nom d'utilisateur des administrateurs ainsi que leur adresse électronique. Une nouvelle page nous était maintenant accessible : « Admin panel ». Cette page permet de supprimer les utilisateurs comme il nous était demandé dans le challenge.




*Conseils de corrections*

Dans un premier temps, il n'est pas conseillé de donner autant d'informations sur les conditions de connexion des administrateurs. Aussi, il serait nécessaire de vérifier qu'il ne soit pas possible de créer des Users avec le même nom plusieurs fois. Enfin, l'email est vérifié à la création du compte mais pas quand on la modifie, ce qui n'est pas normal.

## Challenge 4 - Insufficient workflow validation

### *Contexte*

Les indications du challenge nous informent qu'il y a une vulnérabilité sur l'achat d'articles. Pour résoudre ce challenge, il faut acheter l'article suivant : Lightweight l33t leather jacket. Nous avons accès à notre propre compte (identifiant : `wiener`, mot de passe `peter`).

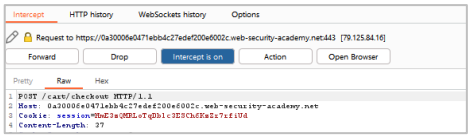
### *Vulnérabilité / problème*

Des failles dans la logique peuvent permettre aux attaquants de contourner les règles. Par exemple, l'utilisateur peut être en mesure d'effectuer une transaction sans passer par le flux de travail d'achat prévu. Cela signifie que lorsqu'un attaquant s'écarte du comportement attendu de l'utilisateur, l'application ne prend pas les mesures appropriées pour empêcher cela et, par la suite, ne parvient pas à gérer la situation en toute sécurité.

### *Exploitation*

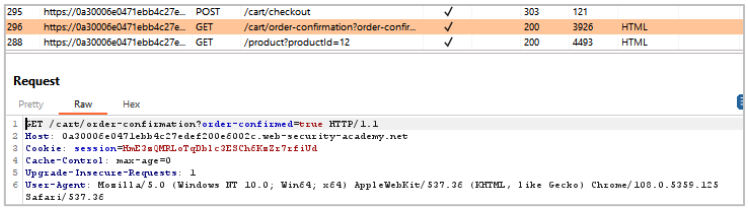
Nous avons lancé Burp Community pour intercepter les requêtes afin d'acheter la veste dans notre budget. Nous nous sommes connectés avec le compte utilisateur donné. Puis, nous avons testé les vulnérabilités vues précédemment (quantité négative, augmenter le conteur indéfiniment etc.) mais rien ne semblait fonctionner.

Alors, nous avons observé, grâce à la page « Proxy » de Burp, les requêtes échangées. Nous avons remarqué qu'une requête POST était générée dès l'ajout d'un article dans le panier et qu'une autre l'était également au moment de payer.




Nous avons essayé d'intercepter la requête de validation d'un panier contenant la veste, mais la requête ne comportait pas de champ susceptible de nous aider à l'acheter dans notre budget.

Nous avons donc supprimé notre panier et y avons rajouté un article d'un prix abordable. Nous avons finalisé son achat afin d'observer les requêtes de validation de panier. Ceci nous a permis de récupérer la requête suivante :




Nous avons envoyé cette requête de validation de commande sur la page « Repeater » de Burp. Nous avons ensuite rajouté dans notre panier la veste en cuir que nous souhaitions. Nous sommes allés sur notre panier et avons envoyé la requête qui était la requête validant notre achat antérieur (celui à petit prix). Ceci nous a permis de valider ce challenge puisque le site internet à cru que la requête de validation lui était parvenu.


### *Conseils de corrections*

Il serait préférable que le site revoie son modèle et utilise un modèle MVC. En effet, la validation du panier doit se faire côté serveur et le site reçoit seulement une notification pour afficher la nouvelle page pour informer l'utilisateur du succès de sa transaction ou non. De plus, il pourrait être préférable de rajouter à un panier un identifiant qui serait vérifier côté serveur pour être sûr qu'une requête ne soit pas réutilisée.