***RAPPORT TP***

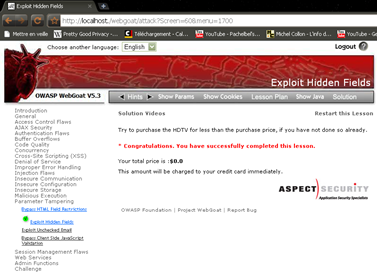
***SECURITE APPLICATIVE***

**Réalisé par : Encadré par :**

***ELFARISSI Abdessamad ELBAKKALI Hanane***

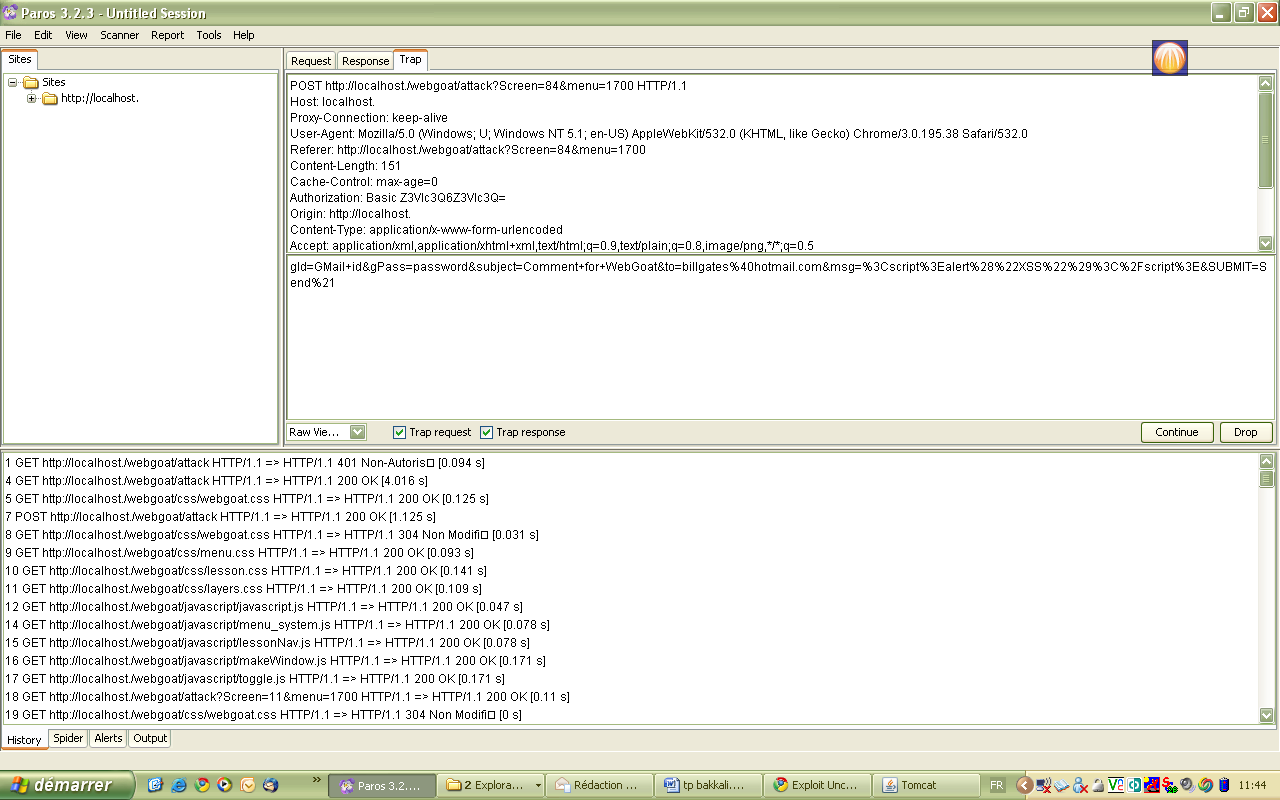
***KHALIL El mahdi***

1. **Première attaque : Changement de prix d’une commande**



Le problème c’est que la vérification se fait cotée client. Donc un client mal intentionné peut faire ce quel veut du cout qu’il possède entre ses main un outil comme paros (webScramble) qui lui permet de récupérer les requêtes http avant qu’elles soient envoyées à destination du serveur. Par l’exemple ici, il passe une commande avec un prix falsifié.

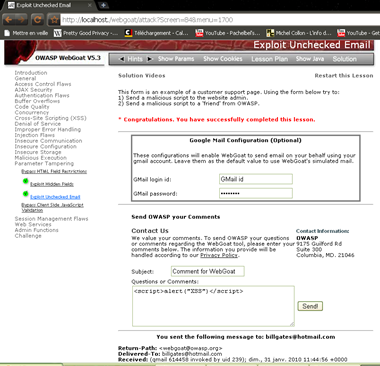
1. **Deuxième attaque : Envoie de code malicieux via un mail.**



Le problème ici c’est que y’a aucun contrôle sur le contenu du mail qui peut contenir des scripts (JavaScript, flash, VB Script,…) malicieux qui peuvent comporter des bout de code dangereux pour contrôler à distance le serveur à l’insu de son utilisateur (webmaster).

On a envoyé une alerte par JavaScript qui est la suivante dans le champ mail : <script>alert("c’est piraté")</script>

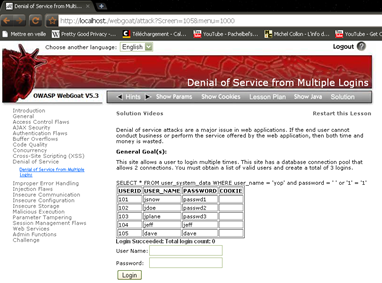
La seconde partie à été de modifier le destinataire, en interceptant le message via Paros et en changeant le paramètre ‘to’ de [webgoatadmin@webgot.org](mailto:webgoatadmin@webgot.org) vers [billgates@hotmail.com](mailto:billgates@hotmail.com) (exploiter la confiance de client vers l’administrateur du site pour faire passer des code malicieux dans le mail).



**III) Troisiéme attaque : injection SQL (causé un déni de service par conncté multiples comptes à la fois).**

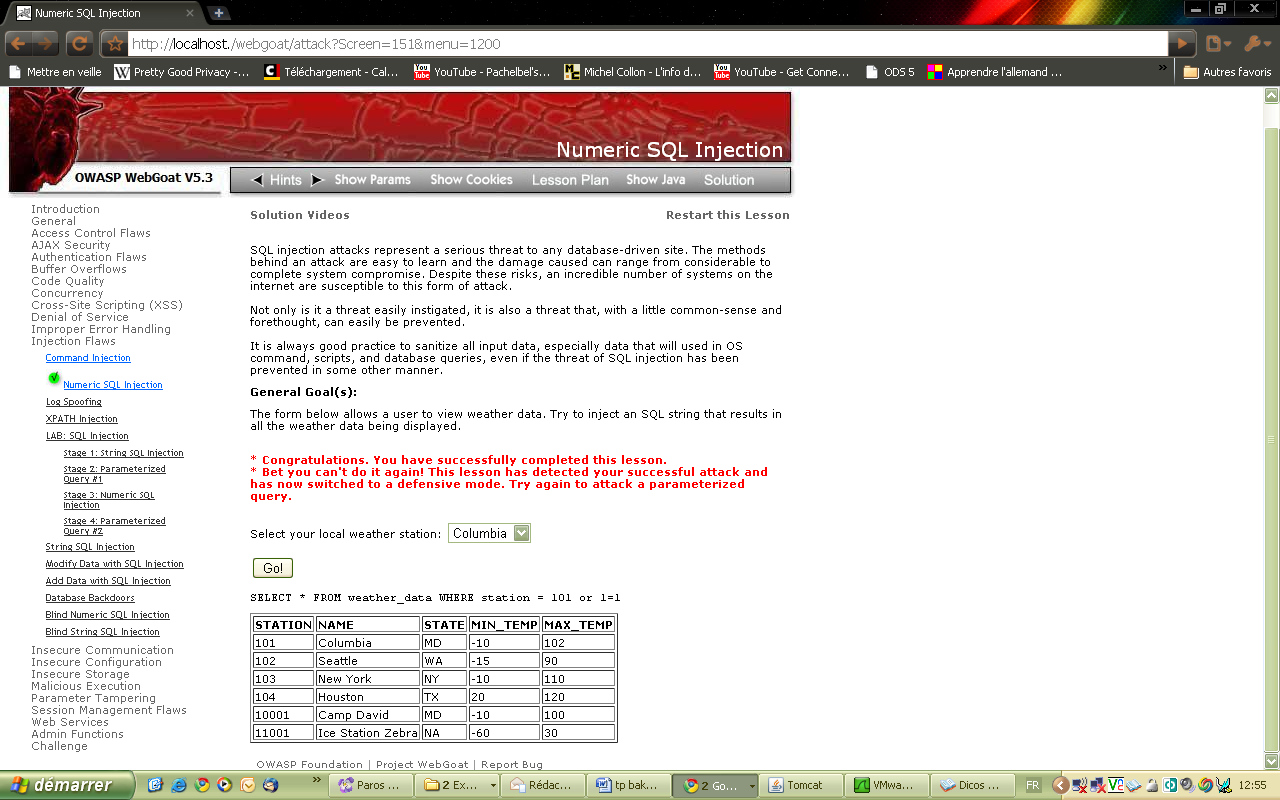
SELECT \* FROM user\_system\_data WHERE user\_name = 'yop' and password = ' ' or '1' = '1'

Ici ‘1’ =’1’ est toujours une condition vrai, ce qui donne que sans mot de passe on peut accéder au contenu de la teble user\_system\_data pour l’utilisateur qui nous intéresse. Cette vulnérabilité est due au fait quel n’est y’a aucun contrôle sur les requêtes passé (Si elle contient des caractères spécieux, ou respecte une format bien donné par exemple).

.

**VI) 4éme injection Injection flaws : Injection SQL numérique**

Accéder à des informations, préalablement non accébles (toujours probléme de non contrôle des requetes passé par l’utilisateur).

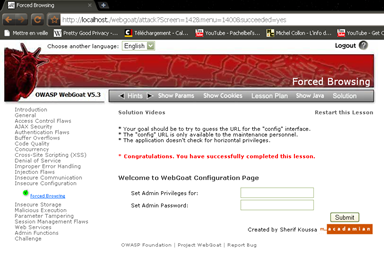


**V) 5éme attaque: Configuration non sécurisée (FORCED BROWSING):**

Le but de cette attaque est de découvrir des url réservés à des personnes privilégiées.

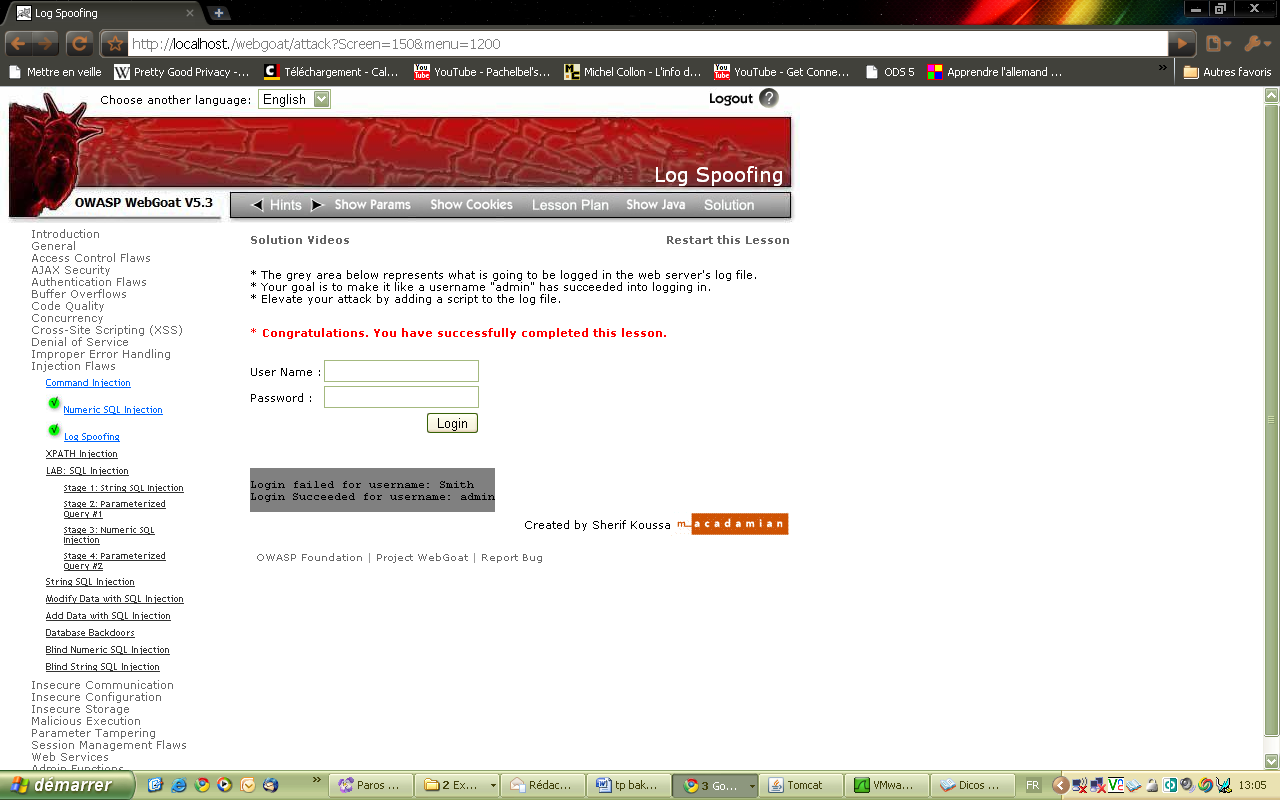
La vulnérabilité qui sera exploitée, c’est qu’un développeur d’une application web, utilise des URL connus et communs pour données des accès privilégiés.

Ici nous avons essayé de deviner l’url du webmaster : après avoir remplacé le chemin de l’URL /attack par /configuration, on a essayé /admin, et ça a donné lieu à la page suivante.



**VI) 6éme attaque : Log spoofing :**

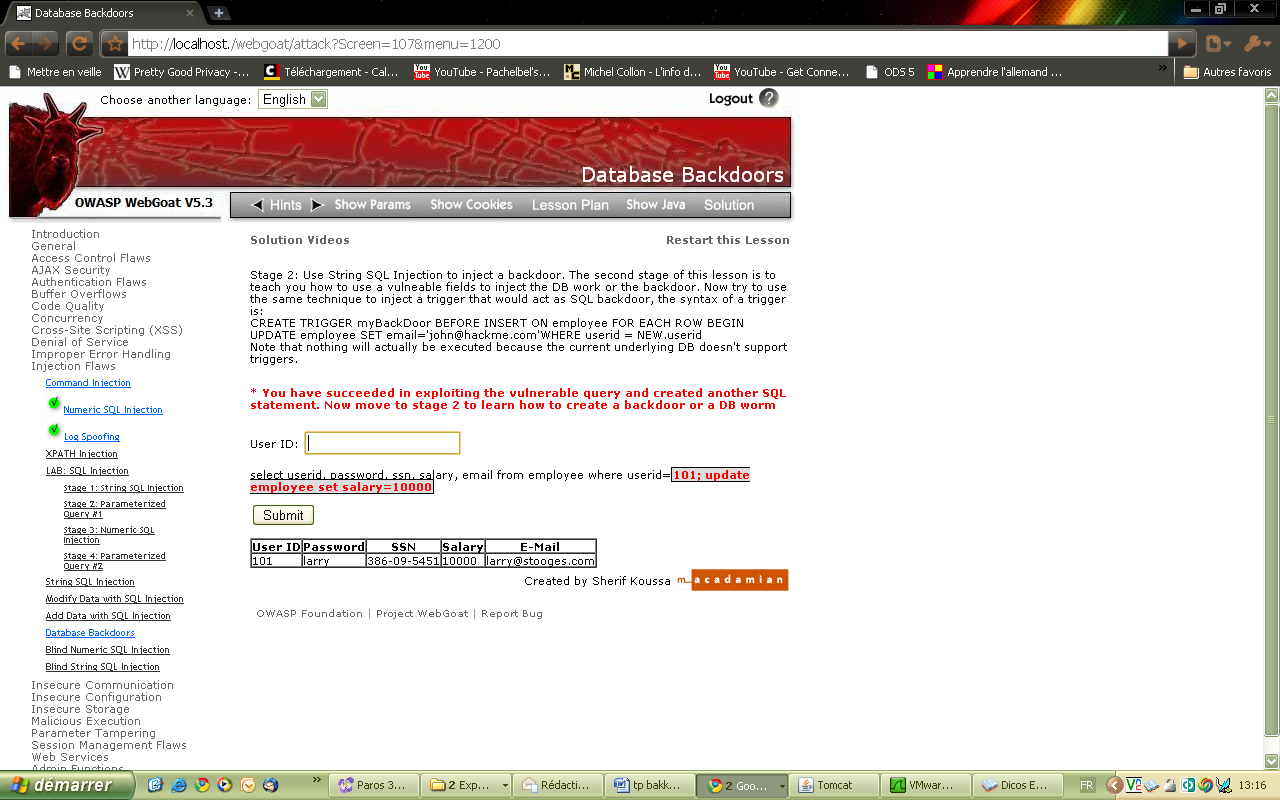
Cette attaque a pour but de toucher au fichier log qui regroupe les différentes connexions qui ont eu lieu. Cette attaque peut être profitable à un hacker qui a envie d’effacer ses traces.



Le champ de remplissage du logger n’admet pas de restrictions, on y injecte alors le texte suivant : ‘Smith%0d%0aLogin Succeeded for username: admin’

Ce qui va simuler le fait que Smith n’a pas été connecté alors que l’utilisateur admin a été connecté. Un bout de code malicieux en JavaScript peut être envoyé au fichier log du site web pour une telle attaque.

**VII) 7éme DATABASE** **BACKDOORS**



1ere étape: on a rempli le champ id par : 101 ;update employee set salary = 100000 ;

On ajoute cette requête SQL : CREATE TRIGGER myBackDoor BEFORE INSERT ON employee FOR EACH ROW BEGIN UPDATE employee SET email='john@hackme.com'WHERE userid = NEW.userid

Afin de rediriger tous les mails sortant de l’admin vers celui de l’attaquant qui pourra tout espionner .

**VIII)** **SESSION MANAGEMENT FLAWS : SESSION FIXATION :**

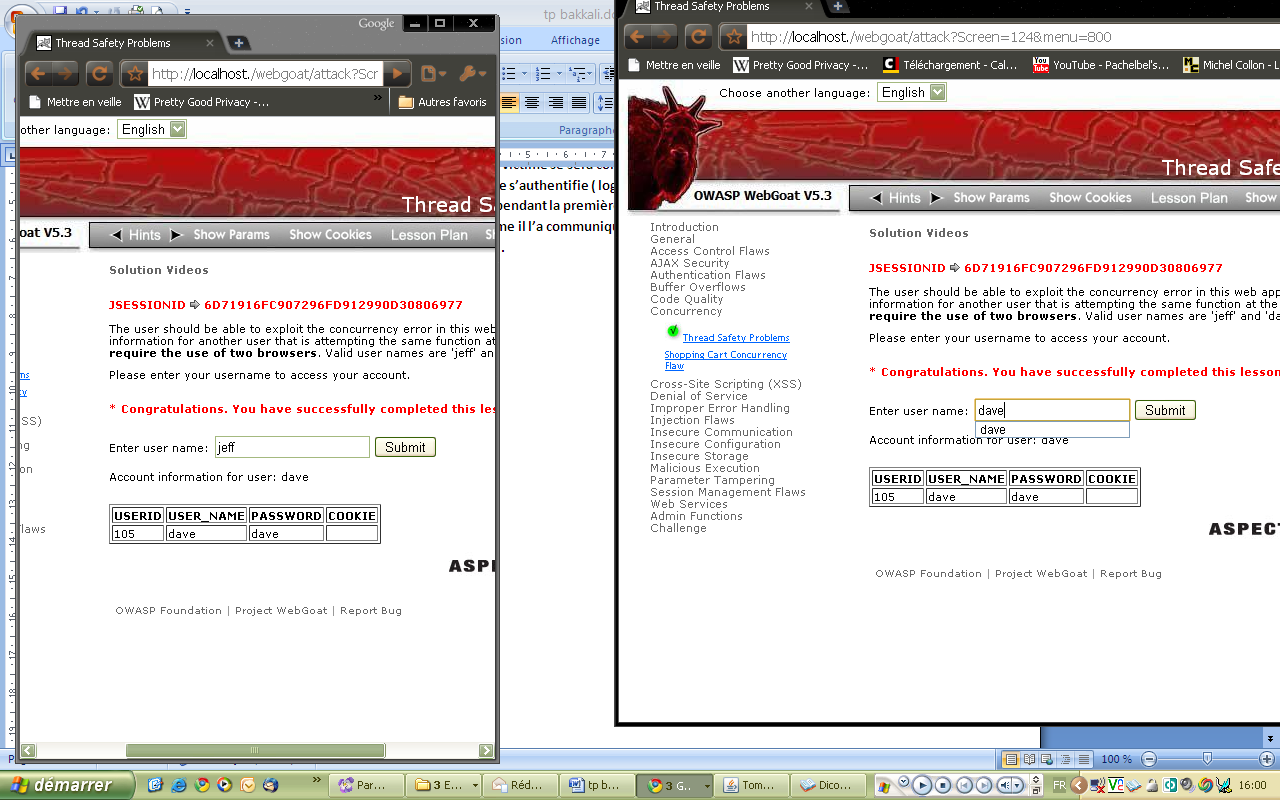
Dans les étapes 1 et 4 je suis le pirate ( hacker ) , dans les étapes 2 et 3 je suis la victime

la procédure est la suivante :

Au début, le pirate rédige un message en bonne et due forme , respectant les normes d’écriture et exempt de fautes d’orthographes , afin de se passer pour l’administrateur de l’entreprise .

Il y inclut également une url contenant un SID ( id de session , que l’on va fixer à la valeur 100 par exemple ) , une fois que la victime se sera connectée elle aura comme SID celui fixé par l’attaquant , à la 3eme étape la victime s’authentifie ( login et mot de passe ) , 4ème étape , l’attaquant connait déjà l’url qu’il va poser cependant la première SID est invalide , et il la remplace alors par la suivante : SID = 100 ( comme il l’a communiquée à la victime ) , l’attaquant peut donc se connecter au compte de la victime !! .

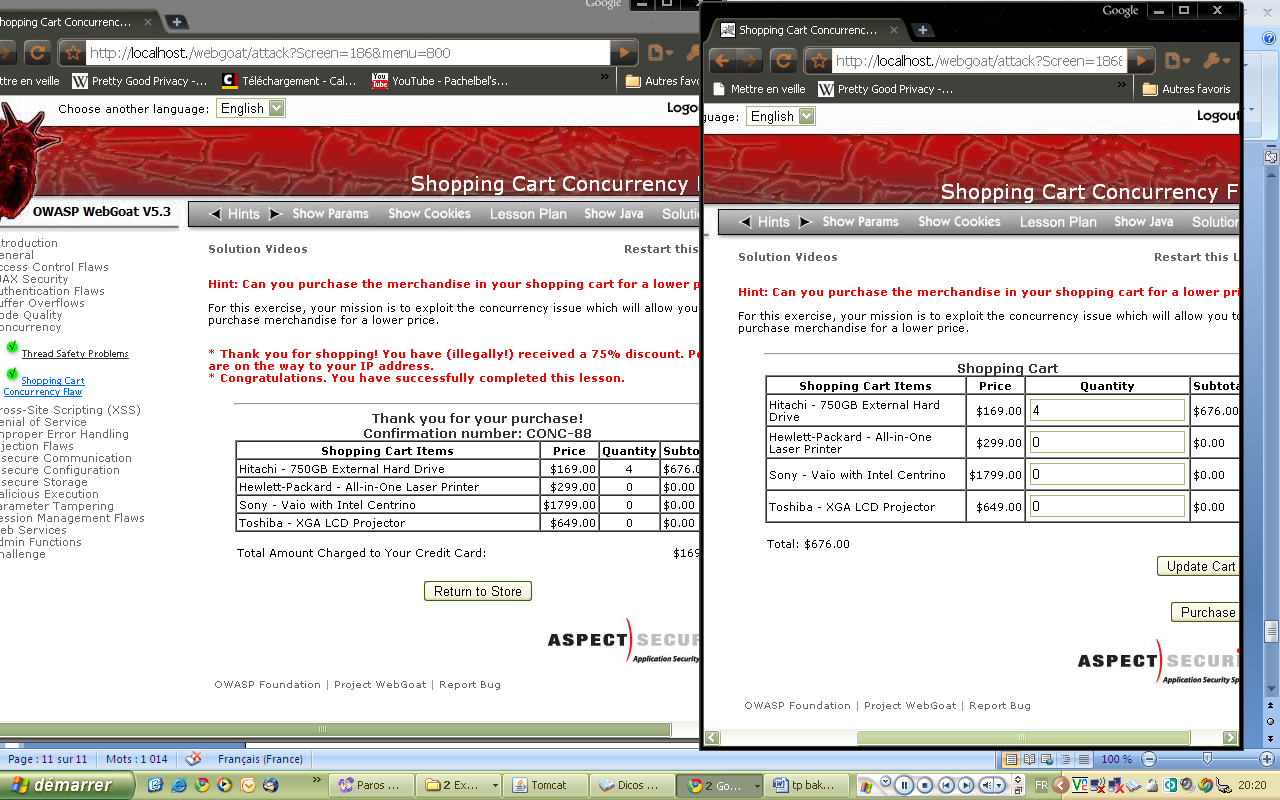
**IX) PROBLEME DE CONCURRENCE DES REQUETES ( THREAD SAFETY PROBLEMS)**

****

Le principe étant d’ouvrir deux navigateurs en même temps et essayer de se loguer avec deux utilisateurs différents, en général l’appel fait travailler les mêmes fonctionnalités systèmes et méthodes de programmation, parfois il y a des bugs dans les appels à ces méthodes qui utilisent les mêmes variables et procédures de programmation, donc les mêmes ressources.

Le résultat est le suivant : si on se logue presque simultanément dans les deux fenêtres, on accède à la requête la plus récente (enfin l’une des deux) , et un utilisateur se trouve connecté dans le compte d’un autre . !!

**X) SESSION MANAGEMENT FLAWS : Paiement par carte de crédit :**

****

On profite de la même faille, c à d la grande défaillance sécuritaire des variables face au threads multiples. on parvient à réduire de façon considérable le prix d’achat d’un produit.

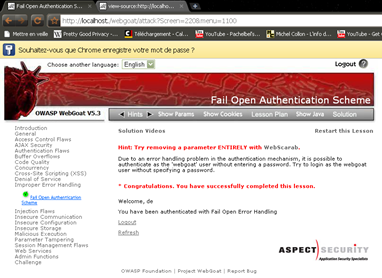
**XI) Qualité du code source :**

****

Parfois on tombe sur des informations très précieuses juste en décortiquant le code source de la page web , commentaires , champs ‘hidden’ ou autres entités html constituent des goulots susceptibles de casser la sécurité de toute une application ,gare à ces erreurs , considérées de débutant ☺ .

L’exemple ci-dessus, on a trouvé le login de l’admin ainsi que son mot de passe ( admin :adminpw)

**XII)** **Improper error handling (fail open authentification scheme) :**



Le but de cette attaque est de profité d’une défaillance de code, en particulier lors du traitement de l’exception JAVA ( POINTEUR NULL).

Lorsque le champ html password est laissé vide, la variable Java correspondant à la chaine de caractère relative est du type NULL, et donc le bloc { Catch throw correspondant à cette exception est appelé } . Or ce bloc d’erreur nous redirige vers une page ou l’on considéré authentifiés, Cette exception aurait du être traité autrement !!! d’où le nom IMPROPER ERROR HANDLING .