

Sécurité du Navigateur



Yannick Chevalier
Université de Toulouse
CSA M1, Security



BUT DU COURS

SÉCURITÉ SUR INTERNET

- ▶ la sécurité côté serveur ne sera pas abordée
- ▶ la sécurité des communications repose sur TLS
- ▶ ce cours : sécurité du client

PLUS GÉNÉRALEMENT

- ▶ exemple d'analyse détaillée d'un composant
- ▶ avantage : tout le monde connaît les navigateurs

PLAN

[0]

ARCHITECTURE DU CLIENT

PARTITIONNEMENT SPATIAL

ATTAQUES LIÉES AU BROWSER

QUELQUES SOLUTIONS POSSIBLES

SÉPARATION SPATIALE (PRINCIPE)

PRINCIPE D'UN BROWSER

- ▶ Il affiche un document à partir duquel il est possible de faire des requêtes vers des sites Webs
- ▶ Ce document contient du html, avec des requêtes formées à partir des balises `a`
- ▶ Il contient aussi des scripts, *i.e.*, des programmes qui peuvent générer des requêtes avec ou sans l'intervention de l'utilisateur

SÉPARATION SPATIALE (PRATIQUE)

SANDBOXING

Chaque fenêtre d'un browser est isolée de son environnement :

- ▶ Politique de **même origine** (SOP) : un script sur une page dans un sous-répertoire de `http://www1.example.com:80/` ne peut effectuer des requêtes que vers les sous-répertoires du site `http://www1.example.com:80/`
- ▶ Partitionnement spatial : les browsers assurent que les scripts hébergés par différentes pages :
 1. ne peuvent pas avoir de variables communes
 2. ne partagent pas de codes de fonctions (mais copies de code autorisées)

CONTENU D'UNE PAGE WEB

BASE

- ▶ Une page Web est représentée par l'arbre des balises HTML qu'elle contient (DOM)
- ▶ Une balise spéciale, `script`, permet d'inclure en plus n'importe quel programme écrit en un langage reconnu par le browser
- ▶ Langage universellement reconnu : JavaScript (JS)

MODIFICATION D'UNE PAGE WEB

- ▶ Un script contenu dans une page peut modifier cette page
- ▶ Il peut ajouter, modifier, ou supprimer des balises de n'importe quel type
- ▶ Le browser réagit à ces modifications en temps réel

COMMUNICATIONS À PARTIR D'UNE PAGE WEB (1/2)

ÉLÉMENTS HTML

- ▶ Certains éléments HTML effectuent naturellement des requêtes GET (`img`, `script`, `a` si l'utilisateur clique)
- ▶ Ces requêtes sont définies par une url qui est recherchée
- ▶ L'usage est d'encoder des éventuelles informations à passer à un serveur dans cette url
- ▶ Les scripts peuvent aussi envoyer des requêtes POST (e.g., soumission d'un formulaire)

COMMUNICATIONS À PARTIR D'UNE PAGE WEB (2/2)

REQUÊTES À PARTIR DE JAVASCRIPT

- ▶ (universel) On crée un objet XMLHttpRequest
- ▶ On indique :
 - ▶ quelle méthode HTTP on désire utiliser
 - ▶ l'url à contacter
 - ▶ éventuellement le contenu de la requête
 - ▶ la fonction à appeler lors de la réception de la réponse (inutile si la requête est synchrone)
- ▶ Et voilà !

PLAN

[0]

ARCHITECTURE DU CLIENT

PARTITIONNEMENT SPATIAL

ATTAQUES LIÉES AU BROWSER

QUELQUES SOLUTIONS POSSIBLES

VIOLATIONS DU PARTITIONNEMENT SPATIAL (1/5)

BIBLIOTHÈQUES

- ▶ Il serait très gênant de demander à l'utilisateur de recharger une bibliothèque très répandue (e.g., JQuery) chaque fois qu'il va sur un nouveau site
- ▶ Solution :
 - ▶ Utilisation du cache du browser
 - ▶ Utilisation d'un **Content Distribution Network** pour distribuer la version de la bibliothèque qu'on utilise
- ▶ Pour cela, on a besoin d'autoriser le chargement de scripts sur d'autres sites Web

IMPORTANT

L'importation de scripts n'est jamais soumise à la politique de même origine

VIOLATIONS DU PARTITIONNEMENT SPATIAL (2/5)

EXTENSIONS

- ▶ Les extensions des browsers peuvent définir des pages virtuelles
- ▶ Ces pages sont isolées des pages affichées (pour le code, pas pour les éléments)
- ▶ Requêtes Javascript et événements possibles entre ces pages et celles affichées

COOKIES/LOCALSTORAGE

Stockage d'informations dans le browser

- ▶ Pas de différences pour la sécurité
- ▶ Définitions de domaine différente pour les cookies du reste
- ▶ Le code servi par un domaine ne peut utiliser que des informations définies par ce domaine

VIOLATIONS DU PARTITIONNEMENT SPATIAL (3/5)

LA SÉCURITÉ, C'EST EMBÊTANT !

D'après ce qui a été écrit, il n'est pas possible

- ▶ de faire référence à une ressource hébergée par un autre domaine
- ▶ d'envoyer des informations personnelles sur l'utilisateur à un autre site (*e.g.*, `doubleclick.com`)

CROSS-ORIGIN RESOURCE SHARING (CORS)

- ▶ Les scripts situés sur la page servie par le domaine A peuvent effectuer des requêtes vers le domaine B
- ▶ Pour avoir une réponse, le domaine B doit être configuré pour accepter explicitement les requêtes venant du domaine A (ou de tout domaine)

IMPORTANT

Politique de contrôle d'accès à compléter éventuellement au sein domaine B avec des informations personnelles de l'utilisateur

VIOLATIONS DU PARTITIONNEMENT SPATIAL (4/5)

RAPPEL

- ▶ Les scripts peuvent être servis par n'importe quel site
- ▶ Donc on peut effectuer une requête vers n'importe quel site en utilisant une balise `script` avec une url forgée correctement

JSON/JSONP (UTILISATION LÉGITIME)

- ▶ JSON : codage d'objets JavaScript simples en du texte
- ▶ JSONP : enrobage du texte pour fournir un code JS valide (appel d'une fonction)
- ▶ Utilisation :
 - ▶ insertion d'une balise `script` dans la page avec une adresse codant les informations qu'on désire envoyer
 - ▶ on laisse le navigateur **évaluer** le code JS reçu en réponse de la requête

VIOLATION DU PARTITIONNEMENT SPATIAL (5/5)

FRAME/IFRAME

- ▶ Élément html permettant d'inclure un autre site
- ▶ Les DOM sont séparés correctement en fonction de la SOP
- ▶ <http://qanimate.com/same-origin-policy-in-nutshell/>

VIOLATION DE PARTITION

- ▶ Gestion complexe de l'affichage (à quels éléments un pixel est relié)
- ▶ Transparence
- ▶ En pratique, une action de l'utilisateur peut toucher plusieurs éléments, y compris en dehors et à l'intérieur d'une (i)frame

COOKIES

DÉFINITION DU DOMAINE

- ▶ Par défaut, un cookie est lié à un domaine et un chemin dans ce domaine (bleu ignoré) :

http://www.example.com/dir1/dir2/toto.html
protocole domaine chemin fichier

- ▶ Par défaut, le protocole utilisé n'est pas pris en compte
- ▶ Secure Cookie : en plus, il faut utiliser le protocole HTTP
- ▶ Il est possible de spécifier un autre domaine pour le cookie (third-party cookies), mais bloquables par l'utilisateur

IMPORTANT

La politique de gestion des cookies est établie sur des bases différentes de celle des autres ressources, ce qui est une source potentielle de conflit

UTILISATION D'UN COOKIE

POSITIONNEMENT DU DOMAINE

Une page à l'adresse `a.b/c/d/` peut créer des cookies :

- ▶ pour un suffixe du domaine (en enlevant une partie `a`) de son adresse
- ▶ pour un préfixe de son répertoire (en enlevant un sous-chemin `d`)
- ▶ il faut au moins 2 parties dans le domaine (`example` et `org`), plus quelques restrictions (dépendantes du browser) pour éviter les gags...

UTILISATION

- ▶ Un cookie est stocké dans le browser
- ▶ Chaque fois qu'une requête est faite vers un sous-domaine et/ou un sous-répertoire d'un cookie, le cookie est ajouté dans le header HTTP de la requête
- ▶ Conséquence (header HTTP) : taille limité à 4ko
- ▶ Ajout automatique, quelque soit la page faisant la requête

LOCALSTORE ET SESSIONSTORE

QUELQUES CARACTÉRISTIQUES EN VRAC

- ▶ Lié à une page Web (\neq lié aux requêtes vers. . .) en fonction de SOP (géré par le navigateur)
- ▶ Pas de limitation de taille
- ▶ Programmatiquement, le code JS sur une page Web a accès à 2 variables **localStorage** et **sessionStorage**, et en fait ce qu'il veut.

PLAN

[0]

ARCHITECTURE DU CLIENT

PARTITIONNEMENT SPATIAL

ATTAQUES LIÉES AU BROWSER

QUELQUES SOLUTIONS POSSIBLES

UTILISATION DES ÉLÉMENTS STATIQUES

PRINCIPE

- ▶ L'affiche d'une page par le navigateur l'oblige à récupérer certains éléments (essentiellement `link/stylesheets`, `img`, les scripts sont bloquables)
- ▶ Pour récupérer ces éléments, il doit envoyer des requêtes GET contenant entre autres le domaine (SOP) de la page faisant la requête et les cookies

EXEMPLES :

- ▶ Insertion d'une image dans un mail pour savoir combien de fois il a été ouvert (sans cookies)

malicieux ou légitime ?

- ▶ Dans une page, couplé à un third-party cookie, permet de suivre les actions d'un utilisateur

Publicités, extensions Chrome/Firefox

CROSS-SITE REQUEST FORGERY (CSRF)

NON-BLOCAGE DES IMAGES + COOKIES

- ▶ À la différence des localStorage, pour lesquels cela demande un peu plus de travail, les cookies d'un domaine sont envoyés lors de chaque requête vers ce domaine.
- ▶ La page d'un domaine A peut donc faire une requête vers un domaine B en utilisant les cookies (*e.g.* identification de l'utilisateur) de ce domaine
- ▶ Si le domaine B ne vérifie pas que la requête est issue d'une page qu'il a servie, et si cette requête est valide, il va l'accepter.

CSRF

- ▶ Un site malicieux utilise le fait que l'utilisateur est potentiellement logué sur un autre site
- ▶ Il envoie des requêtes à cet autre site qui seront accompagnées des cookies de cet autre site (donc authentifiées)
- ▶ Ex : faire un virement si l'autre site est celui d'une banque, récupérer une liste de contact, etc.

LOGIN CSRF

ATTAQUE DÉPENDANT DU BROWSER UTILISÉ

PARTITIONNEMENT SPATIAL LORS DE LA VISUALISATION

- ▶ un site malicieux peut superposer une image opaque pour couvrir une iframe
- ▶ L'image contient un captcha
- ▶ L'iframe contient la page de login d'un site (Yahoo !, Google, etc.)

LOGIN CSRF

- ▶ Le captcha demande à l'utilisateur de rentrer le login et le mot de passe de l'attaquant
- ▶ Une fois l'utilisateur loggué, l'attaquant récupère sur son compte les données de navigation de l'utilisateur

INTERCEPTION DE DONNÉES

PROBLÈME DE PARTITIONNEMENT SPATIAL

- ▶ Couverture d'une iframe par un élément presque transparent
- ▶ Les événements utilisateurs sont aussi envoyés à cet élément
- ▶ Permet au site hôte de récupérer les données envoyées au site hébergé dans l'iframe

PEU DE SOLUTIONS

- ▶ Beaucoup de sites utilisent de manière critique le recouvrement par des éléments transparents
- ▶ Assurer le partitionnement spatial des pixels serait très gênant (popup html, listes descendantes)
- ▶ Pas de solutions prévues

JSONP

- ▶ L'utilisation de JSONP nécessite une bonne collaboration entre les 2 domaines
- ▶ Mais à cause de l'évaluation, le domaine B obtient tous les pouvoirs (parce qu'il peut renvoyer ce qu'il veut à la requête, ce sera évalué) sur la page affichée et ses données
- ▶ Autrement dit, il faut avoir une confiance absolue dans le domaine B pour lui faire des requêtes *via* JSONP

PLAN

[0]

ARCHITECTURE DU CLIENT

PARTITIONNEMENT SPATIAL

ATTAQUES LIÉES AU BROWSER

QUELQUES SOLUTIONS POSSIBLES

IFRAMES

- ▶ Les iframes sont des sources majeures d'insécurité à cause de l'absence de politique de contrôle d'accès efficace sur les événements dans le browser
- ▶ Seule solution pour sécuriser un site : interdire au browser de l'afficher dans un iframe
- ▶ Mais cela interdit aussi les mashup, donc rend le site moins utilisable

CONFIGURATION DE APACHE (HTTPD.CONF)

Header always append X-Frame-Options SAMEORIGIN

Header always append X-Frame-Options DENY

- ▶ Dans le premier cas, les iframes sont autorisées sur le même site
- ▶ Dans le second, elles sont toujours refusées
- ▶ Header des réponses à un GET, honoré par tous les browsers récents

IFRAMES

- ▶ Les iframes sont des sources majeures d'insécurité à cause de l'absence de politique de contrôle d'accès efficace sur les événements dans le browser
- ▶ Seule solution pour sécuriser un site : interdire au browser de l'afficher dans un iframe
- ▶ Mais cela interdit aussi les mashup, donc rend le site moins utilisable

CONFIGURATION DE APACHE (HTTPD.CONF)

Header always append X-Frame-Options SAMEORIGIN

Header always append X-Frame-Options DENY

- ▶ Dans le premier cas, les iframes sont autorisées sur le même site
- ▶ Dans le second, elles sont toujours refusées
- ▶ Header des réponses à un GET, honoré par tous les browsers récents

IFRAMES

- ▶ Les iframes sont des sources majeures d'insécurité à cause de l'absence de politique de contrôle d'accès efficace sur les événements dans le browser
- ▶ Seule solution pour sécuriser un site : interdire au browser de l'afficher dans un iframe
- ▶ Mais cela interdit aussi les mashup, donc rend le site moins utilisable

CONFIGURATION DE APACHE (HTTPD.CONF)

Header always append X-Frame-Options SAMEORIGIN

Header always append X-Frame-Options DENY

- ▶ Dans le premier cas, les iframes sont autorisées sur le même site
- ▶ Dans le second, elles sont toujours refusées
- ▶ Header des réponses à un GET, honoré par tous les browsers récents

IFRAMES

- ▶ Les iframes sont des sources majeures d'insécurité à cause de l'absence de politique de contrôle d'accès efficace sur les événements dans le browser
- ▶ Seule solution pour sécuriser un site : interdire au browser de l'afficher dans un iframe
- ▶ Mais cela interdit aussi les mashup, donc rend le site moins utilisable

CONFIGURATION DE APACHE (HTTPD.CONF)

Header always append X-Frame-Options SAMEORIGIN

Header always append X-Frame-Options DENY

- ▶ Dans le premier cas, les iframes sont autorisées sur le même site
- ▶ Dans le second, elles sont toujours refusées
- ▶ Header des réponses à un GET, honoré par tous les browsers récents

BASE DES CSRF

- ▶ L'attaquant devine que l'utilisateur est loggué sur un certain site
on n'y peut rien s'il devine correctement
- ▶ L'attaquant peut construire des requêtes ayant du sens
Par contre, on peut s'arranger pour que les requêtes incluent un élément aléatoire non-devinable

EMPÊCHEMENT DES REQUÊTES CSRF

- ▶ Demander au serveur de regarder la page d'origine de la requête
utile, mais pas suffisant surtout si le site peut être dans une iframe
- ▶ Ajouter une valeur aléatoire aux requêtes lorsqu'un utilisateur est loggué
cette valeur sera présente dans toutes les requêtes issues d'une page
Web servie, il reste à la vérifier

EMPÊCHEMENT DES REQUÊTES CSRF

SPÉCIFIQUE À CHAQUE SERVEUR D'APPLICATION

IMPORTANT

Application sécurisée par défaut

EXEMPLE (RAILS), APP/CONTROLLERS/APPLICATION_CONTROLLER.RB

```
protect_from_forgery
```

Liste d'attaques

<http://guides.rubyonrails.org/security.html>

FRAGILITÉ DE SOP

RAPPELS :

- ▶ Same Origin Policy : le serveur répond en indiquant les domaines pouvant utiliser la réponse
- ▶ Idem pour l'affichage dans une frame

LA SÉCURITÉ, C'EST EMBÊTANT...

- ▶ HTML5 a ajouté SOP sur plusieurs éléments (attribut `crossorigin`)
- ▶ La sécurité repose sur la collaboration entre le requérant et le serveur
- ▶ Mais il est possible d'insérer un serveur non-coopératif entre les deux pour éliminer les headers intempestifs (www.anyorigin.com, ...), contournent ces défenses...

EXTENSIONS

RÉSUMÉ :

- ▶ une extension est une page Web qui est exécutée “en mode privilégié”
- ▶ la SOP ne s'applique pas aux extensions
- ▶ <http://www.scs.stanford.edu/~deian/pubs/heule:2015:the-most.pdf>

EXEMPLES D'ATTAQUE PAR DES EXTENSIONS

- ▶ AdBlock, qui modifie les pages Web et enlève du revenu publicitaire
- ▶ diverses extensions ajoutent une image 1×1 à chaque page Web pour tracer l'utilisateur
- ▶ CSRF après récupération des données de session

IMPORTANT

seules les extensions fournies par un Web Store sont analysées, les autres peuvent tout changer