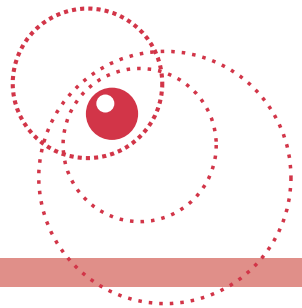


Web mining

Sécurité du Web

2 octobre 2012

Pierre Senellart

TELECOM ParisTech

- N'importe qui peut créer un site Web et s'arranger pour qu'il soit référencé par les moteurs de recherche. . . y compris des personnes malveillantes.
- Aucune raison de faire confiance *a priori* aux créateurs d'un site que l'on visite.
- Les navigateurs sont censés fournir un environnement protégé (bac à sable) dans lequel le code HTML est rendu, les scripts JavaScripts sont exécutés, etc.
- Mais cette protection n'est pas toujours parfaite.

- Principe général d'interaction entre contextes d'éxécution JavaScript : deux scripts (dans différentes fenêtres ou frames du navigateur) ne peuvent interagir que si l'URL du contexte correspondant a le même protocole, port, et nom de domaine
- En pratique, de nombreuses subtilités
- Possibilité de communiquer entre deux scripts qui collaborent avec l'API postMessage
- AJAX est restreint de manière similaire

- Pas de restriction d'origine pour :
  - le chargement d'images, vidéos, et applets
  - le chargement de scripts CSS
  - les `<iframe>` HTML
  - le chargement d'un script JavaScript avec la balise `<script>`
- Les cookies fonctionnent de manière plus libérale (la notion de domaine est étendue, pas de contrôle du port ou du protocole) pour leur écriture/lecture, mais peuvent être restreints à un chemin
- Les plugins Flash, Silverlight, Java, etc., implémentent une politique d'interaction similaire. . . mais des différences de comportement parfois dangereuses

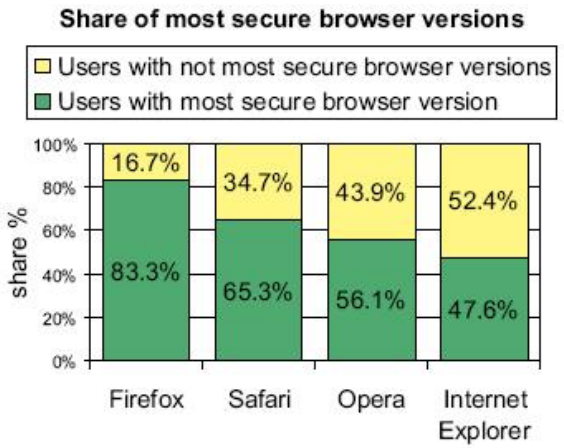
- Mauvais fonctionnement des logiciels
- Divulgation de données confidentielles (mots de passe, numéros de carte de crédit, adresses e-mail, etc.)
- Perte de données locales (vandalisme, rançonnement)
- Mise à disposition des ressources d'un ordinateur local au sein d'un botnet :
  - Stockage de données illégales
  - Relais pour d'autres formes d'attaques
  - Envoi de spams

Problème

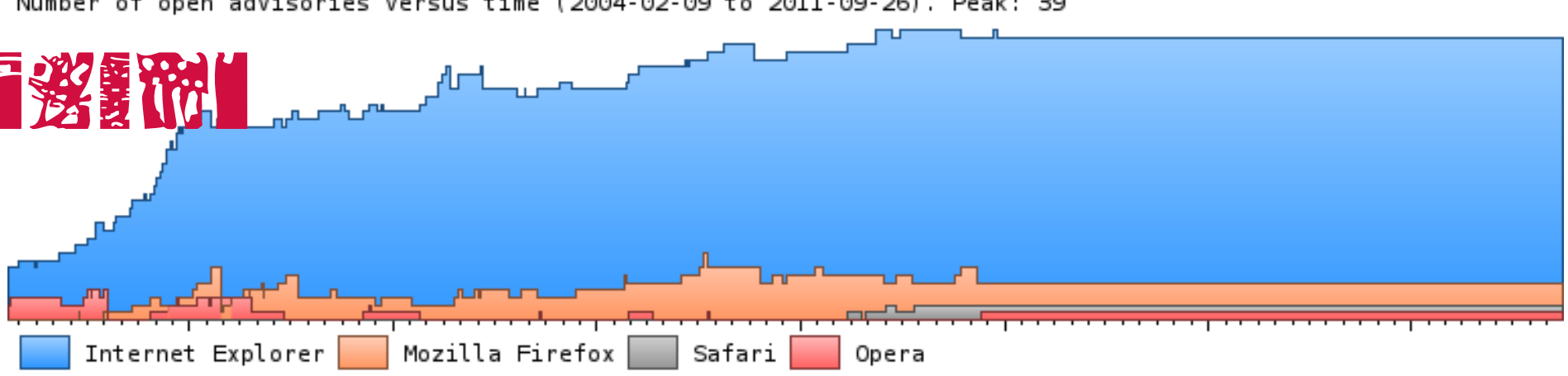
Un défaut de programmation du navigateur Web rend possible des comportements non voulus (du moins grave, un comportement un peu erratique ou un plantage, au plus grave, la prise de contrôle de l'ordinateur par un ordinateur distant).

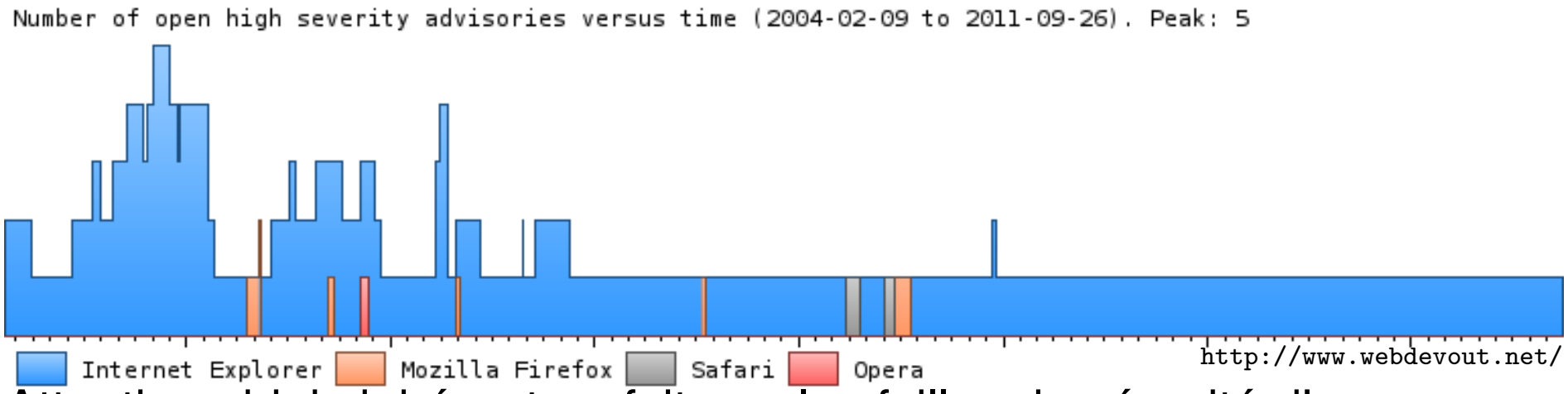
Solution

Mettre à jour son navigateur très régulièrement (par exemple via les mises à jour automatiques du système d'exploitation ou du logiciel lui-même).



http://www.techzoom.net/publications/insecurity-iceberg/index.en





Attention : biais inhérent au fait que les failles de sécurité d'un navigateur avec une faible part de marché seront moins souvent découvertes.

Une erreur de programmation d'une extension d'un navigateur Web (blocage de publicité, plug-in Java, Flash ou PDF, traduction des menus, lecteur multimédia...) rend possible des comportements non voulus.

Solution

Mettre à jour toutes les extensions concernées le plus régulièrement possible (peut devenir compliqué). Désinstaller les extensions non utilisées. S'informer des problèmes de sécurité les plus importants. Favoriser les extensions dont les mises à jours sont intégrées à celle du système d'exploitation (ou du navigateur).

Problème

Un utilisateur lance une application, une applet Java sans bac à sable, un contrôle ActiveX, etc., liés depuis une page Web, ceux-ci pouvant avoir n'importe quel comportement malveillant.

Solution

Faire attention aux messages d'avertissement du navigateur ! Et réfléchir.

Problème

Un utilisateur visitant un site malveillant est conduit à cliquer à un endroit dangereux (avertissement du navigateur, zone d'un frame appartenant à un site de confiance, etc.) en masquant la zone où le clic va avoir lieu jusqu'au dernier moment.

Solution

Timing entre affichage et clic sur une action importante dans les navigateurs ; se méfier des comportements bizarres du pointeur de la souris dans un site

Via un lien (dans un e-mail, sur le Web...), un internaute est amené sur une page Web qu'il pense être celle d'un site auquel il fait confiance (banque en-ligne, commerce électronique...), et se voit demander ses identifiants de connexion. Le site n'est qu'une imitation du site officiel, et ces identifiants sont ainsi divulgués à des individus malveillants.

Solution

Toujours contrôler scrupuleusement l'URL d'un site auquel on parvient via un lien. Dans le cas où un site manipule des informations sensibles ou permet de réaliser des opérations sensibles, ne l'utiliser que sous HTTPS, en vérifiant le certificat SSL (les navigateurs récents affichent l'identité validée du propriétaire du site dans la barre d'adresse). Faire preuve de bon sens !

Problème

Sur un réseau local, ou sur un réseau WiFi non crypté (ou avec un cryptage WEP simple à casser), il est possible à un attaquant de regarder le contenu des paquets IP en clair, contenant l'ensemble de la communication entre le navigateur et le serveur Web, y compris l'ensemble des paramètres HTTP, etc. Ce problème existe aussi, mais de manière moins importante, hors du cadre d'un réseau local ou sans fil.

Solution

Ne pas utiliser HTTP pour transmettre des informations sensibles au travers d'Internet, d'autant plus dans le cadre d'un réseau local ou sans fil. HTTPS, un autre protocole permettant l'envoi crypté de messages sur le Web (et ayant d'autres fonctionnalités avancées par rapport à HTTP), doit être utilisé.

Utilisation d'une des techniques présentées dans ce cours (en particulier, XSS ou capture de paquets IP) pour récupérer l'identifiant de session d'un utilisateur (identifiant ordinairement stocké dans un Cookie), pour se faire passer pour lui.

Solution

Résoudre les autres problèmes ! Côté serveur, prévoir la possibilité de terminer la session (en PHP, avec session_destroy) dès que celle-ci n'est plus nécessaire.

Sécurité côté serveur

- Introduction
- Injection de code
- Non-validation des paramètres HTTP
- Autres attaques

Références

Sécurité côté serveur

- Introduction
- Injection de code
- Non-validation des paramètres HTTP
- Autres attaques

Références

- Web : environnement hostile
- À moins de contrôler l'accès même au serveur Web, n'importe qui peut avoir accès au site Web. . . y compris des personnes malveillantes.
- Certaines choses sont de la responsabilité de l'administrateur (ex., avoir un serveur Web mis à jour régulièrement), le reste est de la responsabilité du webmestre.

- Divulgation de données confidentielles (mots de passe, numéros de carte de crédit, adresses e-mail, etc.) des utilisateurs
- Mise en danger des utilisateurs
- Perte de données locales (vandalisme, rançonnement)
- Mise à disposition des ressources du serveur au sein d'un botnet :
  - Stockage de données illégales
  - Relais pour d'autres formes d'attaques
  - Envoi de spams

Il est possible d'imposer un certain nombre de restrictions sur les données pouvant être envoyée dans un formulaire, côté client :

- `maxlength` sur un `<input type="text">` impose une taille maximale aux champs de saisie texte
- Les `<select>`, bouton radio, case à cocher obligent à choisir une (ou plusieurs) des valeurs proposées
- Un champ caché `<input type="hidden">` ou non modifiable (`readonly`) fixe la valeur d'un paramètre
- Du code JavaScript de validation peut faire des vérifications arbitraire

Il est possible d'imposer un certain nombre de restrictions sur les données pouvant être envoyée dans un formulaire, côté client :

- `maxlength` sur un `<input type="text">` impose une taille maximale aux champs de saisie texte
- Les `<select>`, bouton radio, case à cocher obligent à choisir une (ou plusieurs) des valeurs proposées
- Un champ caché `<input type="hidden">` ou non modifiable (`readonly`) fixe la valeur d'un paramètre
- Du code JavaScript de validation peut faire des vérifications arbitraire

Mais tout ceci n'est vrai que si le client Web joue le jeu. Très facile à contourner (désactivation de JavaScript, modification de la page).
Ne jamais faire confiance aux données envoyées par un client Web !
Toujours (re)faire les validations côté serveur.

Problème

L'utilisateur peut entrer, à l'intérieur d'un paramètre HTTP destiné à être affiché, du code HTML (et donc également des indications de style CSS, du code JavaScript...). Il modifie ainsi le code de la page HTML produite. Si ce paramètre est stocké pour être réaffiché (ex. commentaires de blog), ce code influe sur l'apparence du site pour d'autres utilisateurs.

Exemple

Supposons que le paramètre HTTP `login` contienne :

```
<div style='color: red'>
```

dans le code PHP :

```
echo "Bonjour ".$_REQUEST["login"]."!";
```

Solution

Utiliser les fonctions de protection des caractères spéciaux HTML (en PHP, htmlspecialchars).

Problème

Cas particulier de l'attaque précédente : insertion de code JavaScript dans une page HTML, qui sera réaffiché par d'autres utilisateurs ; le code JavaScript « vole » les informations saisies par l'utilisateur pour les transmettre à un autre site.

Solution

Comme avant, utiliser htmlspecialchars, en particulier quand un texte saisi par un utilisateur est destiné à être affiché par un autre.

Problème

Un site tiers charge (p. ex., dans un `<iframe>`) une page du site attaqué. Le navigateur accède à cette page en étant authentifié comme l'utilisateur original, ce qui permet au site attaquant d'accomplir une action à la place de l'utilisateur (p. ex., passer une commande) ou de déduire des informations sur l'utilisateur

Solution

- Nécessiter une requête POST pour tout comportement ayant un effet de bord
- empêcher l'inclusion d'une page dans une autre (avec l'en-tête HTTP `X-Frame-Options`)
- Faire en sorte que l'accès à cette page soit contrôlé par des tokens uniques non devinables

Un utilisateur peut modifier une requête SQL en mettant des guillemets simples dans une variable à partir de laquelle sera construite la requête.

Exemple

Supposons que `$passwd` contienne « `' OR 1=1 --` » dans le code :

```
$result=mysql_query("SELECT * FROM T WHERE passwd='$passwd'");
```

Solution

Utiliser les fonctions de protection des caractères spéciaux SQL (p. ex., `mysql_real_escape_string` en PHP). Mieux, ne jamais construire de requête de cette façon et utiliser les requêtes préparées.

Problème

L'utilisateur peut modifier les programmes externes appelés côté serveur (par exemple, en PHP, appelés à l'aide des fonctions `system`, `exec`, `passthru`...)

Exemple

Supposons que `$rep` contienne « `&& cat /etc/password` » dans le code PHP :

```
passthru("ls $rep");
```

Solution

Éviter l'appel à des programmes externes depuis le serveur Web. Vérifier soigneusement le contenu des lignes de commande. En PHP, utiliser escapeshellcmd ou escapeshellarg pour protéger les caractères spéciaux pour le shell.

Sécurité côté serveur

- Introduction
- Injection de code
- Non-validation des paramètres HTTP
- Autres attaques

Références

Problème

Un utilisateur peut, lors de l'accès à des fichiers sur le serveur (par exemple, en PHP, avec `fopen`, `readfile`...), en utilisant '/', '..', accéder à des fichiers auquel il n'est pas censé avoir accès.

Exemple

Supposons que `$fichier` contienne « `../../../../../etc/passwd` » dans le code PHP :

```
readfile($fichier);
```

Solution

Vérifier que l'argument des fonctions accédant à des fichiers ne pointe pas vers des fichiers auxquels on ne souhaite pas donner accès (par ex., vérifier qu'il n'y a pas de '/' à l'intérieur).

Problème

Un attaquant peut produire un comportement inattendu côté serveur en exploitant une faille de raisonnement qui suppose qu'un bloc d'instructions sera exécuté en une seule fois, sans être en concurrence avec d'autres instructions.

Exemple

Un script PHP récupère le plus grand entier stocké dans une table MySQL, l'augmente de un, sauvegarde un fichier avec pour nom cet entier, et ajoute une ligne correspondante dans une table MySQL. Il y aura concurrence si deux scripts s'exécutent simultanément, et que les deux consultent la table pour connaître le plus grand entier avant d'avoir ajouté une ligne dans celle-ci.

Solution

Bien réfléchir aux cas de concurrence critique. Utiliser les transactions du SGBD (p. ex., InnoDB dans MySQL 5), utiliser des verrous sur les fichiers.

Problème

Attaquer un site Web (ou un autre service sur Internet) en exigeant du serveur plus que ce qu'il ne peut servir (très grand nombre de connexions, calculs coûteux. . .)

Solution

Essentiellement de la responsabilité de l'administrateur du site, mais le webmestre peut prévenir certaines attaques en 1) évitant les fichiers trop lourds à télécharger 2) évitant les calculs coûteux inutiles côté serveur.

Les mots de passe peu sûrs (noms propres ou mots du dictionnaire sans ou avec petites variations, très courts) peuvent être cassés par force brute, en explorant un à un une liste de mots de passe possible ; c'est d'autant moins coûteux si on arrive à se procurer une liste de mots de passe cryptée.

Solution

Imposer aux utilisateurs (et à soi-même !) des mots de passe sûrs. Ne pas divulguer un fichier de mots de passe, même cryptés. Surveiller les accès à un site Web visant à essayer une liste de mots de passe, et les bloquer.

Problème

Probablement la plus utilisée des attaques, à la base de la majorité des attaques : exploiter une faille non pas dans un quelconque logiciel, mais dans le raisonnement humain ! Pousser un utilisateur honnête à donner des informations confidentielles, etc.

Solution

Garder en tout un esprit critique, faire preuve de bon sens, et ne pas se laisser abuser par une méconnaissance technique !

Afin de prévenir des attaques, ou d'en limiter la conséquence :

- Utiliser des algorithmes de cryptographie reconnus, pas du bricolage.
- Ne jamais stocker de mots de passe dans une base de données, mais uniquement un hash cryptographique non réversible.
- Ne jamais stocker de numéros de cartes de crédit ou autres informations sensibles dans une base de données.
- Faire en sorte que le serveur Web (logiciel) ait des droits d'accès limités à l'ordinateur sur lequel il tourne.
- Ne pas faire une confiance aveugle à du code tiers.
- Réfléchir et se mettre dans la peau d'un attaquant

Sécurité côté client

Sécurité côté serveur

Références

- *The Art of Deception*, Kevin Mitnick, pour se sensibiliser aux dangers de l'ingénierie sociale
- *Hacking, the Next Generation*, Nitesh Dhanjani, Billy Rios, Brett Hardi (O'Reilly) pour des exemples concrets d'attaques modernes
- Firebug pour Firefox, ou autres extensions de navigateur similaires, pour étudier et analyser les sites côté client
- Sites de challenges de hacking comme `http://www.hackthissite.org/` pour se mettre dans la peau d'un attaquant

Article 323-1

Le fait d'accéder ou de se maintenir, frauduleusement, dans tout ou partie d'un système de traitement automatisé de données est puni de deux ans d'emprisonnement et de 30000 euros d'amende.
Lorsqu'il en est résulté soit la suppression ou la modification de données contenues dans le système, soit une altération du fonctionnement de ce système, la peine est de trois ans d'emprisonnement et de 45000 euros d'amende.

Article 323-2

Le fait d'entraver ou de fausser le fonctionnement d'un système de traitement automatisé de données est puni de cinq ans d'emprisonnement et de 75000 euros d'amende.

Article 323-3

Le fait d'introduire frauduleusement des données dans un système de traitement automatisé ou de supprimer ou de modifier frauduleusement les données qu'il contient est puni de cinq ans d'emprisonnement et de 75000 euros d'amende.