Attaque par injection xss

BouzianeYacine
Rahmani Taha

introduction

Les attaques XSS (Cross-Site Scripting) sont un type d'injection dans lequel des scripts malveillants sont injectés dans des sites Web de confiance. Les attaques XSS se produisent lorsqu'un attaquant utilise une application Web pour envoyer un code malveillant, généralement sous la forme d'un script d'un navigateur , à un utilisateur différent. Les failles qui permettent à ces attaques de réussir sont assez répandues et se produisent partout où une application web utilise l'entrée d'un utilisateur dans la sortie qu'elle génère sans la valider ou la coder.

Un attaquant peut employer XSS pour envoyer un script malveillant à un utilisateur non averti. Le navigateur de l'utilisateur n'a aucun moyen de savoir qu'il ne doit pas faire confiance a ce script et exécutera le script. Parce qu'il pense que le script provient d'une source fiable,
le script malveillant peut accéder à tous les cookies, jetons de session ou autres informations sensibles conservées par le navigateur et utilisées avec ce site.
Ces scripts peuvent même réécrire le contenu de la page HTML.

Description

Les attaques XSS se produisent lorsque:

1- Les données entrent dans une application Web via une source non fiable, le plus souvent une requête Web.
2- Les données sont incluses dans le contenu dynamique envoyé à un internaute sans être validé pour un contenu malveillant.

Le contenu malveillant envoyé au navigateur Web prend souvent la forme d'un code JavaScript, mais peut également inclure du code HTML, Flash ou tout autre type de code que le navigateur peut exécuter.
La variété des attaques basées sur XSS est presque illimitée, mais elles incluent généralement la transmission de données privées, comme des cookies ou d'autres informations de session, redirigeant la victime vers un contenu web contrôlé par l'attaquant ou effectuant d'autres opérations malveillantes sur la machine de l'utilisateur. sous l’apparence du site vulnérable.

Attaques XSS stockées et réfléchies

Les attaques XSS peuvent généralement être classées en deux catégories: stockées et réfléchies. Il existe un troisième type d'attaque XSS, beaucoup moins connu, appelé DOM Based XSS, qui est traité ici séparément.

Attaques XSS stockées

Les attaques stockées sont celles où le script injecté est stocké de manière permanente sur les serveurs cibles, comme une base de données, un forum de messages, un journal des visiteurs, un champ de commentaire, etc. La victime récupère ensuite le script malveillant sur le serveur. information. XSS stocké est également parfois appelé Persistent ou Type-I XSS.

Attaques XSS réfléchies

Les attaques réfléchies sont celles où le script injecté est affiché sur le serveur Web, comme dans un message d'erreur, un résultat de recherche ou toute autre réponse qui inclut tout ou partie de l'entrée envoyée au serveur dans le cadre de la demande. Les attaques réfléchies sont transmises aux victimes via un autre itinéraire, par exemple dans un message électronique ou sur un autre site Web. Lorsqu'un utilisateur est trompé en cliquant sur un lien malveillant, en soumettant un formulaire spécialement conçu ou même en naviguant sur un site malveillant, le code injecté se rend sur le site Web vulnérable, ce qui reflète l'attaque vers le navigateur de l'utilisateur. Le navigateur exécute alors le code car il provient d'un serveur "de confiance". XSS réfléchi est également parfois appelé XSS non persistant ou de type II.

DOM basé XSS(Document Object Model)

DOM XSS (ou comme on l'appelle dans certains textes, "type-0 XSS") est une attaque XSS dans laquelle la charge utile d'attaque est exécutée suite à la modification du DOM "environnement" dans le navigateur de la victime utilisé par le client d'origine. script, de sorte que le code côté client s'exécute de manière "inattendue». Autrement dit, la page elle-même (la réponse HTTP) ne change pas, mais le code côté client contenu dans la page s'exécute différemment en raison des modifications malveillantes qui se sont produites dans l'environnement DOM.

Exemple :

Supposons que le code suivant soit utilisé pour créer un formulaire permettant à l'utilisateur de choisir sa langue préférée. Une langue par défaut est également fournie dans la chaîne de requête, en tant que paramètre "default".

```
 ...

 Choisissez votre langue:

 <select> <script>

 document.write ("<OPTION value = 1>" + document.location.href.substring
(document.location.href.indexOf ("default =") + 8) + "</ OPTION>");

 document.write ("<OPTION value = 2> English </ OPTION>");

 </ script> </ select>
 ...
```

La page est invoquée avec une URL telle que:

```
http://www.some.site/page.html?default=French
```

Une attaque XSS DOM basée sur cette page peut être effectuée en envoyant l'URL suivante à une victime:

```
http://www.some.site/page.html?default= <script> alert (document.cookie) </
script>
```

Lorsque la victime clique sur ce lien, le navigateur envoie une requête pour:

```
/page.html?default=<script>alert(document.cookie) </ script>
```

à www.some.site. Le serveur répond avec la page contenant le code Javascript ci-dessus. Le navigateur crée un objet DOM pour la page, dans lequel l'objet document.location contient la chaîne:

```
http://www.some.site/page.html?default= <script> alert (document.cookie) </
script>
```

Le code Javascript d'origine dans la page ne s'attend pas à ce que le paramètre par défaut contienne le balisage HTML, et en tant que tel, il l'échoue simplement dans la page (DOM) à l'exécution. Le navigateur restitue ensuite la page résultante et exécute le script de l'attaquant:

```
alerte (document.cookie)
```

Notez que la réponse HTTP envoyée par le serveur ne contient pas la charge utile de l'attaquant. Cette charge utile se manifeste dans le script côté client lors de l'exécution, lorsqu'un script défectueux accède à la variable DOM document.location et suppose qu'il n'est pas malveillant.

Outils de test et techniques :

Minded Security a fait des recherches significatives sur XSS basé sur DOM. Ils travaillent sur deux projets pour aider avec XSS DOM:

1. L'outil DOMinator - Un outil commercial basé sur le navigateur Firefox avec un moteur Javascript Spidermonkey modifié qui aide les testeurs à identifier et à vérifier les failles XSS du DOM.
2. Le DOM XSS Wiki - Le début d'une base de connaissances pour définir les sources d'entrées et de sorties contrôlées par l'attaquant qui pourraient potentiellement introduire des problèmes XSS basés sur DOM. C'est très immature à partir du 17/11/2011. S'il vous plaît contribuer à ce wiki si vous connaissez des puits plus dangereux et / ou des alternatives sûres !!
3. DOM Snitch - Une extension expérimentale de Chrome qui permet aux développeurs et aux testeurs d'identifier les pratiques non sécurisées couramment rencontrées dans le code côté client. De Google

Conséquences de l'attaque XSS

La conséquence d'une attaque XSS est la même, qu'elle soit stockée ou réfléchie (ou DOM XSS). La différence réside dans la façon dont les scripts arrive sur le serveur. . XSS peut entraîner une variété de problèmes pour l'utilisateur qui varient en gravité d'un ennui jusqu'au vol des informations confidentielles .

Les attaques XSS les plus graves impliquent la divulgation du cookie de session de l'utilisateur, permettant à un attaquant de détourner la session de l'utilisateur et de prendre le contrôle du compte. D'autres attaques incluent la divulgation des fichiers de l'utilisateur , l'installation des programmes, redirection de l'utilisateur vers une autre page ou un autre site, ou modifient la présentation du contenu. Une vulnérabilité XSS permettant à un attaquant de modifier un article de presse qui pourrait affecter le cours des actions d'une entreprise ou diminuer la confiance des consommateurs. Une vulnérabilité XSS sur un site pharmaceutique pourrait permettre à un attaquant de modifier les informations de dosage entraînant un surdosage.

Comment s'en protéger?

Les mesures de sécurité que nous allons voir seront implémentés au niveau du serveur et non pas sur le client. Donc, si le serveur est sécurisé contre les attaques XSS, ses clients n'auront pas à affronter des ennuis comme ce qui a été expliqué dans le paragraphe précédent.

Au niveau du code PHP :

Comme pour les injections SQL, l'attaque XSS est dues à des entrées provenant de l'extérieur (donc non fiables). La solution consiste donc à filtrer les entrées de l'utilisateur en appliquant les fonctions comme **addslashes()** ou **strip_tags()** qui supprime toutes les balises contenues dans la chaine entrée. On peut aussi formater les mots-clés HTML à l'aide de la fonction **htmlentities()** ou **htmlspacialchars()**. Cependant, une chaine de caractères faisant office d'un script XSS est généralement longue, il faut alors n'autoriser qu'une certaine longueur maximale pour les entrées en la vérifiant à l'aide de la fonction **strlen()** ou en la tronquant systématiquement à l'aide la fonction **substr()**.

Au niveau de la configuration du serveur :

Comme pour l'injection SQL, on peut activer la directive **magic_quotes_gpc** dans le fichier **php.ini** pour échapper automatiquement tous les caractères spéciaux (notamment les simples et doubles quotes) figurants dans les chaines provenant le l'extérieur. Bien que ce n'est pas suffisant, mais cette solution apportera un peu d'aide quand même.