Philippe CHAUVELIN - François-Guillaume RIBREAU

CSII 3e année

Les attaques XSS

E.P.S.I - NANTES

114 rue des Hauts Pavés

BP 41903

44019 NANTES CEDEX 01

Sommaire

Introduction

« Au printemps dernier, les dirigeants de Sony présentaient leurs excuses à 75 millions d'utilisateurs pour ne pas avoir suffisamment protégé leurs comptes. ».

Sony est une multinationale japonaise spécialisée dans le domaine du multimédia. Celle-ci à mis plus d'une semaine avant de remarquer qu'elle se faisait voler les données personnelles de près de 75 millions de joueurs abonnés à son réseau Playstation Network. Une erreur qui lui coutera environ 171 millions de dollars. Symantec, créateur de la suite de sécurité Norton, estime les pertes infligées par la cybercriminalité à près de 271 milliards d'euros, et le nombre de cybercrime quotidien à un million.

L'expansion de l'informatique conduit de plus en plus de personnes à développer leurs propres programmes, que cela soit en entreprise ou pour une utilisation personnelle. Cette expansion est une porte ouverte au piratage, de plus en plus courante en raison du nombre de développeurs qui augmente chaque année. Les exploits, faisant partie de nos jours des logiciels malveillants les plus utilisés par les pirates, sont de plus en plus virulents. Ce thème est très intéressant car il nous permettra de nous pencher sur l'une des failles les plus courantes du moment. **Nous allons donc dans ce rapport aborder les différents vecteurs d'attaque et de protection des applications Web relatives aux failles XSS**.

Le Cross Site Scripting, ou XSS, est la faille la plus présente sur le web, et d'assez loin. Elle est désignée par quantité de noms, parmi lesquels "faille des livres d'or", tout simplement car ceux-ci ont permis une généralisation de celles-ci. La faille de type XSS se caractérise par une injection possible de code arbitraire dans une application web côté client. Autrement dit, une possibilité d'éxécution d'une variable mal contrôlée par le site. L'attaque par XSS à pour but la récupération d'informations personnelles sur cette dernière afin de permettre l'usurpation d'identité (phishing) ou alors la propagation d'un virus comme

par exemple le virus Samy qui a contaminé plus de un million de profils MySpace.

Nous aborderons dans une première partie le fonctionnement détaillé d'une faille XSS et plus particulièrement la phase de détection de celle-ci. Nous réaliserons qu'il existe deux sous-catégories d'attaque XSS. La première sous-catégorie, la moins dangereuse est le XSS réfléchi. Elle permet, via l'injection de code Javascript dans une url menant à un site non protégé, de contrôler le contenu de celui-ci. Elle est considérée comme moins dangereuse par la communauté des développeurs étant donné qu'il est presque nécessaire d'utiliser l'ingénierie sociale pour assurer son fonctionnement. D'un autre coté, on distinguera le XSS stocké, ou permanent, qui est à l'origine d'un certain nombre d'attaque de grand compte (MySpace, Twitter, etc.). Contrairement au XSS réfléchi, le code injecté peut être stocké sur le serveur et par conséquent toucher un grand nombre de personnes (Exemple : injection de code malveillant dans un forum non protégé). La compréhension de ces différents éléments nous permettra d'aborder les solutions envisageables afin de protéger ses applications Web tel que les protections à appliquer lors des insertions en base de données, la protection des cookies disposant d'informations sur l'utilisateur ou encore les différentes solutions proposées par les langages et frameworks.

« Les entreprises investissent d'avantage dans leurs machines à café que dans la sécurité informatique ».

Le premier ver XSS a été créé en 2005 sur le réseau social mondial MySpace. Sans gravité et nommé «Samy » il avait pour but d'ajouter à la liste d'amis de la victime un contact appelé Samy, pour ensuite se répliquer. Au final Samy a contaminé près de 1 000 000 de profils. Cependant, ce dernier n'a pas été le seul ver à se propager à si grande échelle, il y a eu également le ver « Yammaner » (crée en 2006). Celui-ci a contaminé un grand nombre de serveur web-mail, notamment Yahoo. S'appuyant sur le même principe que Samy, Yammaner avait pour but de voler la liste de contact de la victime. Pour cela cette dernière recevait un email qui une fois ouvert s'envoyait lui-même à tous les contacts de la victime.
Nous verrons dans une seconde partie les différentes phases de l'attaque de MySpace. Cette attaque étant l'une des plus grande en terme d'utilisateurs touchés, il y sera abordé la manière dont la faille a été découverte, le procédé de fabrication du code source visant exploiter celle-ci, la méthode de propagation et pour finir la solution qui été trouvé pour

contourner l'attaque.

Chapitre

1 Approche théorique

1.1 Définition

Le XSS est une attaque visant les sites web qui affichent dynamiquement du contenu côté client et ce sans effectuer de contrôle des informations saisies par les utilisateurs. Les exploits Cross-Site Scripting consistent à forcer un site web à afficher du code HTML ou des scripts non autorisés et saisis par les utilisateurs. Le XSS est donc une attaque menée dans le but d'exécuter un code Javascript directement sur le poste de la victime. Cette technique est particulièrement utilisée par les pirates pour voler la session d'un utilisateur connecté. Si le pirate arrive à identifier un site web vulnérable, il a donc la possibilité d'insérer un script au sein d'une URL et de l'envoyé à une victime. Si cette dernière est connectée sur ce site web vulnérable, elle enverra immédiatement, et à son insu, son cookie de session au pirate. L'attaquant pourra alors utiliser ce cookie et voler la session de la victime pour mener des actions frauduleuses.

1.2 Recherche de faille

Il n'y a pas de classification officielle établie entre les différents types d'attaque XSS. Une attaque XSS peut généralement être catégorisé en deux catégories : les non-persistante (ou réfléchi) et les persistantes (ou stockée). Il existe une troisième catégorie d'attaque XSS, beaucoup moins connue, appelée XSS basée sur le DOM.

1.3 XSS réfléchi

1.4 XSS stocké

Contenu Section

1.5 Vulnérabilité basé sur le DOM

1.5.1 Définition

Les attaques XSS basées sur le DOM (aussi appelé dans d'autres textes les ẋSS de type-0́) sont possibles grâce à la modification de l'environnement DOM dans le navigateur de la victime afin de faire exécuter d'une manière non-prévue le script côté client. La page elle même ne change pas mais le code côté client contenues dans la page s'exécutera différemment en raison des modifications malveillantes qui ont eu lieu dans l'environnement DOM.

En contraste avec d'autres attaques XSS (stockées ou réfléchie), où l'origine de l'attaque est placé dans la page de réponse (grâce à une faille côté serveur).

Pour qu'une attaque de type XSS based DOM puisse avoir lieu, le script client de la page cible doit utiliser des données provenant de document.location ou document.URL ou document.referrer (ou n'importe quel autre objet que l'attaquant pourrait influencer) d'une manière non sécurisée.

Le type connu en tant que type 1, ou vulnérabilité réfléchie résulte de l'utilisation de données fournies par l'utilisateur dans un script quelconque, sans les modifier. Typiquement, une simulation en ligne ou une page de statistiques. Ainsi, si ces données ne sont pas modifiées, on peut ajouter du s̈cript dans le scriptq̈ui sera lui-même éxécuté.

Ceci dit, en modifiant les données qui doivent être traitées, le résultat du XSS ne va modifier que la page que va afficher l'utilisateur. Cela peut paraître bénin, mais ça l'est beaucoup moins quand l'attaquant utilise le Social Engineering et diffuse des pages piégées de cette façon. Ce genre de vulnérabilités est souvent utilisé pour lancer des campagnes de spam afin de ternir l'image d'un site (redirections, modifications d'apparence) ou de voler des informations (phishing). Le type 2, ou vulnérabilité persistente ou du second ordre, permet des exploitations plus en profondeur. C'est la faille des livres d'or, présente dans les forums, les formulaires d'inscription. La différence essentielle est que les données entrées sont stockées dans des bases de données et sont traitées quand un utilisateur les demande. Par conséquent, on peut affecter n'importe qui sollicitera un certain sujet dans un forum ou la liste des pseudos enregistrés, etc.. Cette faille peut permettre des éxécutions

côté client ou côté serveur selon les cas et peut permettre tout type d'exploitation, de la récupération de cookies à l'éxécution de scripts malveillants. On a vu des XSS intégrés à des bases de données institutionnels rendant inaccessibles des dizaines de sites dépendants de ces contenus. Enfin, le type 0, connu comme le DOM-based ou local cross scripting site est un problème directement sur le script côté client (en général, le javscript) de la page (variables passées en URL qui sont réutilisées dans le javascript, etc..). Cette vulnérabilité est soit exploitée à nouveau par Social Engineering, soit par liens interposés dans lesquels on injecte du code qui sera ensuite éxécuté côté client. Celui-ci est finalement très sensible au type I et est très répandu et facilement repérable, notamment par des scanners automatisés.

1.5.2 exemple

Il n'est pas rare de trouver une application HTML qui contient du code JavaScript qui *parse* l'adresse URL (en accédant à document.location ou document.URL) et réalise des actions côté client en se basant dessus. L'exemple ci-dessous est un exemple d'un tel scénario.
Supposons que le code suivant soit utilisé pour créer un formulaire afin de laisser l'utilisateur choisir sa langue préférée. La langue par défaut est également fourni dans la chaîne de requête (paramètre d̈efault).

Listing 1.1 – Code initial

```
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 chargée avec l'url suivante :
http://www.sitevulnerable.com/page.html?default=French
Une attaque XSS basée sur le DOM sur cette page peut être accompli si la victime ouvre l'URL suivante :
http://www.sitevulnerable.com/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>
sur www.sitevulnerable.com. Le serveur répondra par la page contenant le code ci-dessus Javascript. Le navigateur créera un objet DOM dans la page (pour le select) en se basant sur le document.location qui sera la chaine de caractère : http://www.sitevulnerable.com/page.html?d
Le code original Javascript dans la page n'est pas prévu pour que le paramètre d̈efaultc̈ontienne des balises HTML. De ce fait, dans notre cas, il va simplement écrire notre code HTML dans la page (le DOM) lors de son exécution. Le navigateur va alors afficher la page et exécuter le script de l'attaquant :

```
alert (document.cookie)
```

Notons que la réponse HTTP envoyée par le serveur ne contient pas la charge utile (payload) de l'attaquant. Cette charge utile se manifeste d'elle-même lors de l'exécution des scripts côté client. Ce type d'attaque est repérable par exemple dans le cas où un script défectueux accède à la variable document.location et suppose son contenu malveillant.

1.6 Solutions envisageables

Contenu Section

1.6.1 XSS basé sur le DOM

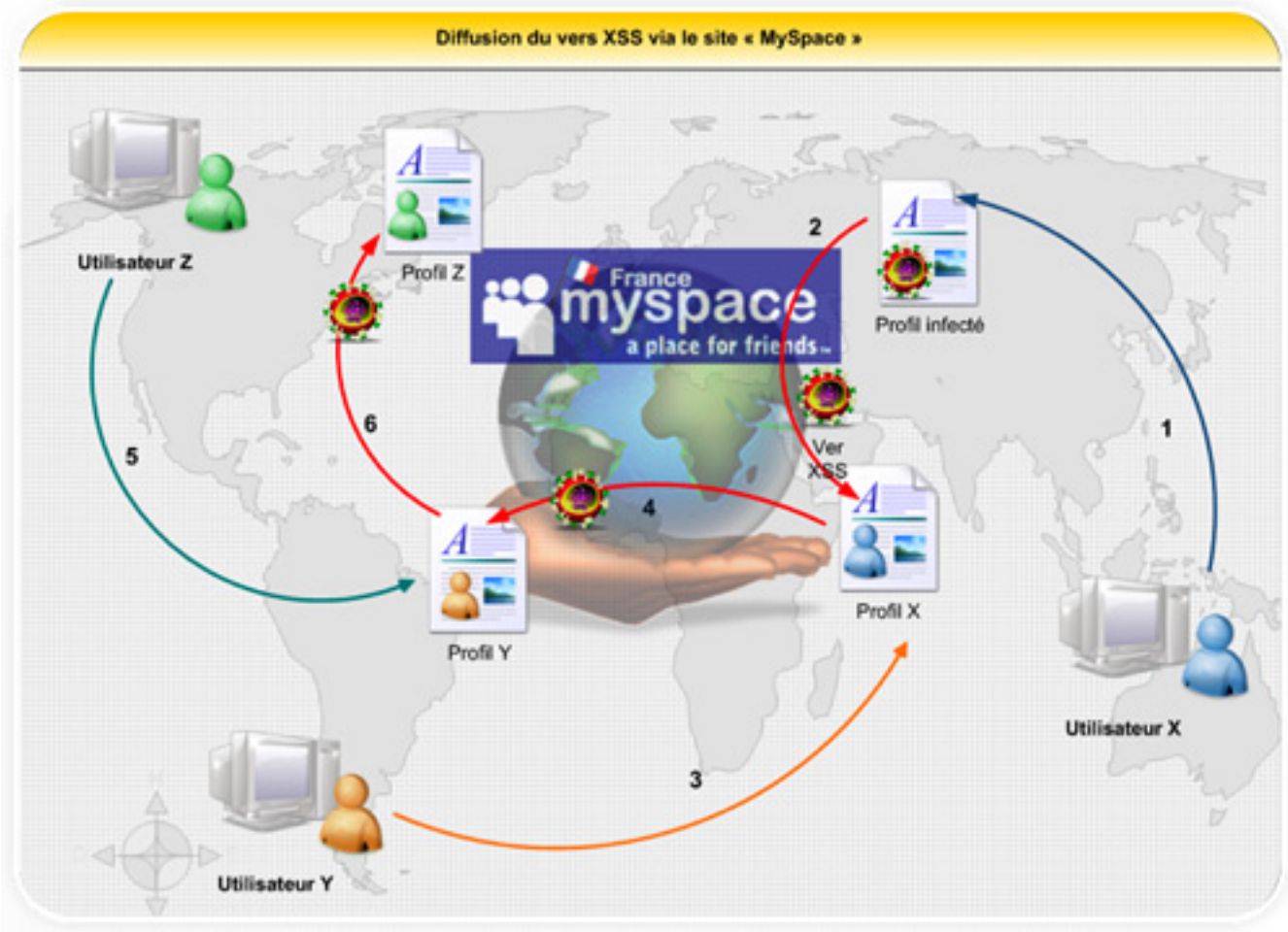
Comme nous l'avons dit précédement, lors d'une attaque XSS basée sur le DOM la charge utile n'était pas contenue dans la réponse HTTP du serveur mais à néanmoins été envoyée au serveur lors de la requête HTTP. On peut donc penser que cette attaque pourrait être détectée côté serveur. En réalité, il est possible d'éviter cette détection par le serveur et de passer par d'autres variables que document.location.

Chapitre

2 Approche pratique

2.1 Cas MySpace

Le premier ver XSS a été créé en 2005 sur le réseau social mondial MySpace. Sans gravité et nommé «Samy » il avait pour but d'ajouter à la liste d'amis de la victime un contact appelé Samy, pour ensuite se répliquer. Au final Samy a contaminé près de 1 000 000 de profils.



Comment le pirate a-t-il découvert la faille ? Il a tout d'abord remarqué que l'insertion de code HTML au sein de son profil était possible. En théorie si l'injection de code HTML

est possible, il est également possible d'injecter du code Javascript, c'est à ce niveau que le pirate s'est heurté à un problème. En effet le site Myspace dispose d'une forte restriction en ce qui concerne le terme « javascript » (Ex : <script language= ''text/javascript") par conséquent l'insertion de ce dernier semblait très délicate. Il a donc trouvé une solution pour contourner ce problème, qui consistait à séparer ce terme en deux parties par retour chariot, ce qui lui a permis ensuite d'exploiter cette faille au maximum. Ce code JS envoyait discrètement et à l'insu de la victime des requêtes POST et GET via AJAX, la seule condition était que la victime visite le profil de l'utilisateur Samy. Le plus compliqué dans la mise en place de cette faille était la génération de requête http automatique sans la moindre intervention. La victime ne se rendait pas compte que lors de sa visite du profil Samy le navigateur lançait des requêtes http, simplement parce que la validation se faisait automatiquement (très dur à réaliser en JS). Samy n'a pas été le seul ver à se propager à si grande échelle, il y a eu également le ver « Yammaner » (crée en 2006). Celui-ci a contaminé un grand nombre de serveur web-mail, notamment Yahoo. S'appuyant sur le même principe que Samy, Yammaner avait pour but de voler la liste de contact de la victime. Pour cela cette dernière recevait un email qui une fois ouvert s'envoyait lui-même à tous les contacts de la victime.

2.2 Charge utile(Payload)

2.3 Propagation

2.4 Conclusion

Conclusion

Contenu Conclusion

Bibliographie

[1] Monsieur Barbu. *La Belle thèse*. PhD thesis, Université, 2002.

[2] Fernand Dupont. *Les choux farcis*. Un gros éditeur, 2004.

[3] Nestor Dupont. *réparer son vaisseau*. l'Alliance, 2009.

[4] Patrick Durand and René Durand. Les tomates tueuses. *Le beau journal*, page 24, jan 2007.

[5] Monsieur Mauvais. Mon roman inachevé. il est chouette mon roman, feb 2000.

[6] Les Zéros. Le site du zéro, jun 2009. www.siteduzero.com.

http ://www.webappsec.org/projects/articles/071105.shtml

Table des figures

Annexes

Table des matières