Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE
PREMIER MINISTRE

S . G . D . S . N
Agence nationale de la sécurité des systèmes d'information
CERTA

Paris, le 03 janvier 2005
N° CERTA-2004-INF-001-001

Affaire suivie par :
CERTA

NOTE D'INFORMATION DU CERTA

Objet : Sécurité des applications Web et vulnérabilité de type "injection de données"

Gestion du document

Référence	CERTA-2004-INF-001-001
Titre	Sécurité des applications Web et vulnérabilité de type "injection de données"
Date de la première version	13 décembre 2004
Date de la dernière version	03 janvier 2005
Source(s)	
Pièce(s) jointe(s)	Aucune

TAB. 1 – *Gestion du document*

Une gestion de version détaillée se trouve à la fin de ce document.

1 Introduction

L'injection de données est une technique couramment employée et largement répandue pour un grand nombre d'attaques sur les applications "web" ou les scripts CGI (*Common Gateway Interface*). Le but est d'insérer des données en entrée d'une fonction, d'un programme ou bien d'un script afin de les détourner de leur fonction d'origine.

Le but de ces attaques peut être très varié :

- exécution de code arbitraire à distance ;
- SQL injection ;
- vol de cookies ;
- débordements de mémoire.

Afin d'éviter d'être vulnérable à ce type d'attaque, un certains nombre de précautions doivent être prises.

2 Descriptions des attaques

2.1 L'utilisation des méta-caractères

Les méta-caractères sont des caractères spéciaux qui, placés en entrée d'une ligne de commande, ne sont pas interprétés comme des données. Ces caractères spécifiques ont un sens différent suivant le programme qui les

interprétera (shell, CGI, SQL, ...).

En voici une liste non-exhaustive :

```
& ~ " # ' { } ( [ ] ( ) - | ` _ \ ^ @ \*/.<>,;:!$
```

L'emploi de ces caractères permet à un individu mal intentionné de construire des chaînes malicieuses qui seront interpretées par le programme qui les traite. Il est donc important de limiter leur usage et de les filtrer.

2.2 Failles de codage HTML, PHP, ASP

Il est nécessaire d'effectuer une inspection attentive du code de certaines pages HTML, des pages dynamiques (PHP ou ASP par exemple) afin d'échapper aux vulnérabilités de type injection.

Certains éléments peuvent participer à l'injection de code ou de scripts malicieux. Il convient de s'assurer de l'intégrité de ceux-ci lorsqu'ils sont retournés :

- paramètres et contenus des URLs ;
- éléments des formulaires ;
- cookies ;
- requêtes vers une base de données.

et de manière plus générale, tout élément qui provient de l'extérieur et qui n'est pas parfaitement maîtrisé par le développeur.

L'une de ces faille les plus connues est le `Cross Site Scripting` (cf. Bulletin d'information CERTA-2002-INF-001). Cette vulnérabilité est due à un défaut de filtrage des données entrées par un utilisateur permettant l'injection de code HTML, Javascript, ...

Plusieurs moyens permettent d'échapper à ce type de faille :

- le codage explicite des caractères ;
  Il est nécessaire de spécifier le type d'encodage par défaut ISO-8859-1 afin que certains caractères tels '<' ou ' >' ne soient pas interprétés comme des tags par certains navigateurs web. Ils seront alors remplacés respectivement par '&lt' et '&gt'.

```
<HTML>
<HEAD>
<META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
....
```

- l'identification des caractères spéciaux ;
- filtrage des données en entrée ;
- examen des cookies.

2.3 Les attaques de type "SQL injection"

Le langage SQL (Structured Query Language) est un langage utilisé pour manipuler les bases de données relationnelles. Il permet la manipulation (création, extraction, modification ou bien suppression) de données d'une base SQL.

La vulnérabilité de type "SQL injection" est une technique permettant d'injecter du code de type SQL dans les champs des formulaires HTML. Ces parties sont ensuite intégrées dans les requêtes SQL utilisées par le serveur WEB ce qui permet de contourner les contrôles et d'avoir accès à des informations normalement protégées.

Prenons l'exemple simple d'une page HTML de type formulaire permettant d'effectuer une requête sur une table possédant cinq champs : un ID, un nom, un prénom, son statut et son salaire. En indiquant la valeur de l'ID dans le champ correspondant du formulaire, on accède aux informations de la fiche personnelle d'un employé :

```
<HTML>
<HEAD>
```

```
<TITLE> Test SQL injection <TITLE>
</HEAD>

<BODY>
<FORM METHOD="post" action="requete.php">
<INPUT type="text" name="ID" size=20><BR>
<INPUT type="submit" name="envoyer" value="envoyer">
</FORM>
</BODY>
</HTML>
```

Les paramètres inscrits dans le formulaire seront traités par la page dynamique *requete.php* qui inclura dans son code la ligne suivante :

```
<?php
...
$id_requete=my_sqlquery("select * from table_employe WHERE id='$ID')
...
?php>
```

La requête SQL est alors préétablie dans la page PHP. Elle permet d'exécuter la requête SQL suivante vers le serveur :

```
SELECT * from table_employe WHERE id=xxx.
```

Un utilisateur mal intentionné voulant connaître les informations sur l'ensemble des salariés sans connaître aucun ID ni même leur format peut rentrer dans le champ `ID` du formulaire HTML une chaîne malicieusement construite : `' OR 1=1 --`.

Par ce biais, la requête SQL va être modifiée :

```
SELECT * FROM table_employe WHERE id='' OR 1=1 --.
```

L'utilisateur aura alors accès à la totalité des informations (salaire d'un employé sans avoir connaissance de l'ID).

Cette faille résulte dans le fait que la chaîne de caractères renseignée dans le champ ID n'a pas été vérifiée. En effet, la vérification de la longueur maximale de la chaîne et de l'utilisation de certains méta-caractères permet, entre autres, d'empêcher l'usage de ceux-ci pour la modification de la requête SQL initiale.

Pour cette faille, le filtrage des méta-caractères peut se révéler insuffisant. Effectivement, une chaîne de caractères de type :

```
1 UNION ALL SELECT salaire FROM table_employe WHERE status LIKE directeur
```

ne comporte pas de méta-caractères et permet pourtant d'avoir accès à des informations normalement protégées. Il est donc nécessaire de filtrer également les verbes SQL tels que :

```
SELECT, FROM, WHERE, UNION, INSERT, INTO, DROP.
```

Les exemples d'utilisation de cette faille peuvent être très variés :

- lancement de plusieurs requêtes sur une base de données ;
- élévation de privilèges ;
- utilisation de code arbitraire à distance ;
- lecture ou modification de clés de la base de registre.

3 Solution afin d'éliminer ce type de faille

Il est nécessaire de filtrer les données d'entrée d'un script, d'un shell ou d'une fonction afin de supprimer tout ce qu'un utilisateur malicieux aurait put y insérer. Ce filtrage doit être effectué au niveau du serveur et non du client pour éviter que celui-ci ne soit désactivé (ex: javascript).

Pour réaliser ce filtrage, certains éléments peuvent être pris en considération :

- suppression des méta-caractères ;
- contrôle des paramètres passés en argument, de leur valeur, et de leur nom ;
- limitation de la longueur des données ;
- respect du type de données (chaine de caractères, date, caractères numériques).

Rappelons ici qu'il s'agit bien d'un filtrage au niveau applicatif, et que les filtrages au niveau réseau (protocole IP) ou transport (protocoles TCP, UDP, ...) ne peuvent pas protéger un serveur contre ce type d'attaque.

De plus, même si l'application des correctifs du système d'exploitation et des serveurs est essentielle, elle ne suffit pas pour se protéger contre l'injection de données.

Le relecture du code est donc une étape essentielle dans le développement des applications Web.

Il existe de nombreux logiciels spécifiques pour le développement de sites dynamiques, mais leur utilisation nécessite également des précautions particulières.

Les logiciels de développement en PHP (PHP-Nuke, phpBB, ...) fournissent des briques de base pour le développement de sites. Ces briques font régulièrement l'objet de mises à jour de sécurité qu'il convient d'appliquer.

Il existe également des logiciels propriétaires utilisant leurs propres fonctions pour la récupération des données d'un formulaire et la création de requêtes vers une base de données. Les filtrages des données récupérées ne sont pas effectués par défaut, et il appartient au développeur de les mettre en œuvre.

4 Contournement provisoire

Si un filtrage des données au niveau des applications n'est pas possible, il convient de protéger le serveur HTTP lui-même, ou de mettre en place un dispositif de sécurité extérieur au serveur.

Concernant la sécurité du serveur HTTP, citons par exemple le module *mod_security* du serveur Apache ou le service *ISA* (*Internet Security and Acceleration*) pour le serveur *IIS* de Microsoft (cf. section Documentation).

Il est également possible de filtrer des URLs au moyen d'un relais inverse (ou *reverse-proxy*). Grâce à des expressions régulières, seules les URLs spécifiquement autorisées ne sont pas rejetées.

Il est primordial de restreindre le champ des URLs autorisées au strict minimum. Il est également nécessaire de modifier ces règles au fur et à mesure de l'évolution des applications sur le serveur Web.

5 Documentation

- Avis de sécurité CERTA-2002-AVI-156 du CERTA :
  http://www.certa.ssi.gouv.fr/site/CERTA-2002-AVI-156/index.html
- Avis du CERT/CC sur l'injection de code malicieux du 03 février 2000 :
  http://www.cert.org/advisories/CA-2000-02.html
- Bulletin du CERT/CC sur le développement d'applications Web :
  http://www.cert.org/tech_tips/malicious_code_mitigation.html
- Bulletin du CERT/CC sur le filtrage des méta-caractères :
  http://www.cert.org/tech_tips/cgi_metacharacters.html
- Note d'information du CERTA sur le *Cross-Site Scripting* :
  http://www.certa.ssi.gouv.fr/site/CERTA-2002-INF-001/index.html
- Alerte du CERTA sur l'exploitation de la vulnérabilité "include PHP" du 09 septembre 2003 :
  http://www.certa.ssi.gouv.fr/site/CERTA-2003-ALE-003/index.html

- Site sur la sécurité des portails PHP :
  http://www.phpsecure.info
- How-To Linux pour une programmation sûre :
  http://tldp.org/HOWTO/Secure-Programs-HOWTO/index.html
- Un exemple de module Perl pour traiter les scripts CGI :
  http://stein.cshl.org/WWW/CGI/
- Page de documentation du serveur HTTP Apache pour la mise en place de CGI :
  http://httpd.apache.org/docs-2.0/howto/cgi.html
- Module *mod_security* pour le serveur HTTP Apache :
  http://www.modsecurity.org
- Page de documentation pour la mise en place d'un relais inverse avec le serveur HTTP Apache :
  http://www.apacheweek.com/features/reverseproxies
- Site Internet du serveur ISA de Microsoft :
  http://www.microsoft.com/isaserver

Gestion détaillée du document

13 décembre 2004 version initiale.
03 janvier 2005 Ajout des liens et du paragraphe sur les relais inverses.