

# Protéger son code sources PHP avec bcompiler

par [maxime.ohayon](#)

Date de publication : 01/06/2007

Dernière mise à jour :

Il est possible de protéger son code sources PHP des petits malins qui pourraient le récupérer car parfois on peut vendre ou donner des scripts mais on ne souhaite pas que le destinataire possède les sources. La bibliothèque Bcompiler vous aidera dans cette tâche.

I - Bcompiler c'est quoi ?

I-A - Description

I-B - Comment ça marche ?

II - Avec un exemple

II-A - Pré-requis

II-B - Transformation du code

II-B-1 - Explications

II-B-2 - Résultat

III - Finition


IV - Conclusion

## I - Bcompiler c'est quoi ?

### I-A - Description

Bcompiler est une bibliothèque php créée à l'origine :

- 1 Pour encoder un script complet dans une application PHP propriétaire
- 2 Pour encoder des classes et/ou des fonctions dans une application PHP propriétaire
- 3 Pour permettre d'utiliser des applications PHP-GTK sur des bureaux clients sans avoir besoin du fichier php.exe.
- 4 Pour rendre faisable de convertir un code PHP en C

Nous resterons que sur le premier but. Il faut activer l'extension php\_bcompiler.dll sous Windows ou télécharger, décompresser et recompiler PHP sous linux (explication ici :  <http://us2.php.net/manual/fr/ref.bcompiler.php> )

### I-B - Comment ça marche ?

En réalité Bcompiler transforme votre code en ByteCode (similaire à Java ou C#) non lisible par un utilisateur mais seulement par PHP, il s'agit d'un code intermédiaire plus abstrait que le code machine non directement exécutable. Il est contenu dans un fichier binaire un peu plus lourd qui représente un script, tout comme un fichier objet produit par un compilateur, ce ByteCode est directement interprété par PHP et il n'est pas possible de retrouver le code PHP original.

## II - Avec un exemple

### II-A - Pré-requis

Imaginons deux scripts hello.php et function.php que nous voulons protéger.

hello.php

```
<?php
include "function.php";
echo hello();
?>
```

function.php

```
<?php
function hello()
{
    return "hello";
}
?>
```

### II-B - Transformation du code

Tout d'abord il faut transformer nos deux scripts en ByteCode, nous allons passer par un script en PHP qui permettra de les transformer.

transform.php

```
<?php

$fh = fopen("hello.phb", "w");

bcompiler_write_header($fh);

bcompiler_write_file($fh, "hello.php");

bcompiler_write_footer($fh);

fclose($fh);

?>
```

#### II-B-1 - Explications

On crée par cette transformaiton le fichier hello.phb

```
$fh = fopen("hello.phb", "w");
```

On écrit dans ce fichier un en-tête de type Bcompiler pour que l'interpréteur PHP puisse comprendre qu'il s'agit de ByteCode.



### III - Finition

Maintenant que l'on a nos deux scripts hello.php et function.php il faut modifier les fichiers hello.php et function.php pour qu'ils puissent appeler leurs homologues 'ByteCodé'.

hello.php

```
<?php
include "hello.phpb" ;
?>
```

function.php

```
<?php
include "function.phpb" ;
?>
```

L'appel par include "mon\_fichier\_ByteCodé.phpb" permet d'exécuter le ByteCode, donc sur la page hello.php on verra hello.

## IV - Conclusion

Il faut éviter de faire de multiples includes de fichiers car cela ralentit considérablement l'exécution des scripts. Cependant, il existe d'autre fonction de cette extension qui permettent ce genre d'opération (<http://us2.php.net/manual/fr/ref.bcompiler.php>).

Bcompiler est une bonne façon de protéger son code et offre de nouvelle perspective pour PHP, de plus il est gratuit et offre une excellente alternative à tous les autres logiciels (payants) proposant les mêmes services.

