HTTP HEAD method trick in php scripts (Tested on php version 5.3.5)

Adam Iwaniuk

March 3, 2011

This article is about behavior of php scripts opened by http HEAD method. A lot of coders assume that their scripts won't be interrupted and will run to the end (especially for short scripts). When HEAD method is used, php script stops on the first output, what can provide some security holes.

```
php-5.3.5\main\SAPI.c line 315:
if (SG(request_info).request_method &&
    !strcmp(SG(request_info).request_method, "HEAD")) {
        SG(request_info).headers_only = 1;
```

```
php-5.3.5\main\output.c line 699
(fucntion php_ub_body_write which is executed when
output data arrives):
if (SG(request_info).headers_only) {
    if(SG(headers_sent)) {
        return 0;
    }
    php_header(TSRMLS_C);
    zend_bailout(); // <--- this will stop script
}
```

Simple guestbook based on files script as seen in real world:

```
<?php
$line='Nick: '.htmlspecialchars($_POST['nick']).'<br>
Text: '.htmlspecialchars($_POST['text']).'<hr>';
$f=fopen("book.txt","r");
$data=fread($f,filesize("book.txt"));
fclose($f);
$f=fopen("book.txt","w");
$data=$line.$data;
echo $data;
fwrite($f,$data);
fclose($f);
?>
```

When someone will open that script using HEAD method script will stop on `"echo $data;"`, so book.txt will be empty.

Another simple example:

```
<?php
session_start();
$_SESSION['admin']=1;
if (!isset($_POST['pass']) || $_POST['pass']!='somepassword')
{
    echo '<b>Wrong or empty password.</b><br>';
    $_SESSION['admin_level']=0;
}
?>
```

In this example, after HEAD method script will stop on:

```
"echo '<b>Wrong or empty password.</b><br>';"
```

Variable `$_SESSION['admin_level']` will stay set to 1. In a great deal of servers `output_buffering` is set to 4096, so first output will flush when 4096 bytes will arrive to the buffer. In this case vulnerable script should look like this:

```
<?php
session_start();
echo 'A long string contains about 4090 characters';
$_SESSION['admin']=1;
if (!isset($_POST['pass']) || $_POST['pass']!='somepassword')
{
    echo '<b>Wrong or empty password.</b><br>';
    $_SESSION['admin_level']=0;
}
?>
```

This kind of vulnerabilities are not common, but a lot of scripts may have interesting behavior when we stop running them in the middle.