[UNIX] Apache HTTPd Arbitrary Long HTTP Headers DoS

Source: http://www.derkeiler.com/Mailing−Lists/Securiteam/2004−07/0001.html

---

From: SecuriTeam (*support_at_securiteam.com*)
Date: 07/01/04

```
To: list@securiteam.com
Date: 1 Jul 2004 17:15:54 +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/mailinglist.html

− − − − − − − − −

Apache HTTPd Arbitrary Long HTTP Headers DoS
−−−

```
    if (last_field != NULL) {
        if ((len > 0) && ((*field == '\t') || *field == ' ')) {
..
fold_buf = (char *)apr_palloc(r->pool, alloc_len);
```

If the header starts with a non−linear whitespace character (space or tab), Apache will allocate as much memory for it as needed. Then, using arbitrarily long headers one can create a situation in which the server gets low on memory, either causing problems in serving pages to clients or to completely crash Apache.

The vulnerability is also applicable on 64 bit systems under the conditions that the size of an integer is 4 bytes and the size of a long integer is 8 bytes. The code can be reached by line 743: ap_escape_html(r->pool, last_field). The last field can be arbitrarily long. Taking a look at the code yields:

```
int i, j;

for (i = 0, j = 0; s[i] != '\0'; i++)
      if (s[i] == '<' || s[i] == '>')
         j += 3;
      else if (s[i] == '&')
         j += 4;

   if (j == 0)
      return apr_pstrmemdup(p, s, i);

   x = apr_palloc(p, i + j + 1);
```

The sum i+j+1 can almost be arbitrary due to integer addition between signed integers. On Linux x86_64 it was confirmed that sending about 820MB of data overflows
(i+j+1) which leads to a crash in memcpy, but with good heap layout more can be done.

Patch Availability:
An unofficial patch by an Apache developer is given below:

```
Index: server/protocol.c
− ===================================================================
RCS file: /home/cvspublic/httpd−2.0/server/protocol.c,v
retrieving revision 1.148
diff −u −r1.148 protocol.c
−−− server/protocol.c 22 Apr 2004 22:38:03 −0000 1.148
+++ server/protocol.c 13 Jun 2004 19:47:36 −0000
@@ −716,6 +716,23 @@
              * continuations that span many many lines.
              */
             apr_size_t fold_len = last_len + len + 1; /* trailing
null
*/
+
```

```
+ if ((fold_len − 1) > r−>server−>limit_req_fieldsize) {
+ r−>status = HTTP_BAD_REQUEST;
+ /* report what we have accumulated so far before the
+ * overflow (last_field) as the field with the problem
+ */
+ apr_table_setn(r−>notes, "error−notes",
+ apr_pstrcat(r−>pool,
+ "Size of a request header
field "
+ "after folding "
+ "exceeds server limit.<br
/>\n"
+ "<pre>\n",
+ ap_escape_html(r−>pool,
last_field),
+ "</pre>\n", NULL));
+ return;
+ }
+
            if (fold_len > alloc_len) {
                char *fold_buf;
                alloc_len += alloc_len;
```

ADDITIONAL INFORMATION

The information has been provided by <mailto:guninski@guninski.com>
Georgi Guninski.

==========

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.