CVE-2011-3192

The official version of this document resides at [http://httpd.apache.org/security/CVE-2011-3192.txt] - this document is for drafting and discussion of the workarounds and side effects upon clients.

```
          Apache HTTPD Security ADVISORY
          ==============================

Title:       Range header DoS vulnerability Apache HTTPD prior to 2.2.20.

CVE:         CVE-2011-3192
Last Change: 20110831 1800Z
Date:        20110824 1600Z
Product:     Apache HTTPD Web Server
Versions:    Apache 2.0 - all versions prior to 2.2.20 and prior to 2.0.65
             Apache 1.3 is NOT vulnerable.

Draft changes since update 3
============================
Note PR #51748.

Changes since update 2
======================
2.2.20 has a fix, 2.2.21 an improved one. Version 1.3 is not vulnerable.
Further regex/rule improvements.  Explained DoS.  Added wiki link.
Highlight fact that LimitRequestFieldSize workaround was insufficient.

Changes since update 1
======================
In addition to the 'Range' header - the 'Request-Range' header is equally
affected. Furthermore various vendor updates, improved regexes (speed and
accommodating a different and new attack pattern).

Description:
============

A denial of service vulnerability has been found in the way the multiple
overlapping ranges are handled by the Apache HTTPD server prior to version
2.2.20:

     http://seclists.org/fulldisclosure/2011/Aug/175

An attack tool is circulating in the wild. Active use of this tool has
been observed.

The attack can be done remotely and with a modest number of requests can
cause very significant memory and CPU usage on the server.

The default Apache httpd installations version 2.0 prior to 2.0.65 and
version 2.2 prior to 2.2.20 are vulnerable.

Apache 2.2.20 does fix this issue; however with a number of side effects
(see release notes). Version 2.2.21 corrects a protocol defect in 2.2.20
(PR 51748 https://issues.apache.org/bugzilla/show_bug.cgi?id=51748 ),
and also introduces the MaxRanges directive.

Version 2.0.65 has not been released, but will include this fix, and is
anticipated in September.

Apache 1.3
==========

Apache 1.3 is NOT vulnerable. However as explained in the background section
in more detail - this attack does cause a significant and possibly unexpected
load. You are advised to review your configuration in that light.

Type of Attack
==============

This vulnerability concerns a 'Denial of Service' attack. This means that
```

a remote attacker, under the right circumstances, is able to slow your
service or server down to a crawl or exhausting memory available to serve
requests, leaving it unable to serve legitimate clients in a timely manner.

There are no indications that this leads to a remote exploit; where a
third party can compromise your security and gain foothold of the server
itself. The result of this vulnerability is purely one of denying service
by grinding your server down to a halt and refusing additional connections
to the server.

Background and the 2007 report
==============================

There are two aspects to this vulnerability. One is new, is Apache specific;
and resolved with this server side fix. The other issue is fundamentally a
protocol design issue dating back to 2007:

      http://seclists.org/bugtraq/2007/Jan/83

The contemporary interpretation of the HTTP protocol (currently) requires a
server to return multiple (overlapping) ranges; in the order requested. This
means that one can request a very large range (e.g. from byte 0- to the end)
100's of times in a single request.

Being able to do so is an issue for (probably all) webservers and currently
subject of an IETF discussion to change the protocol:

      http://trac.tools.ietf.org/wg/httpbis/trac/ticket/311

This advisory details a problem with how Apache httpd and its so called
internal 'bucket brigades' deal with serving such "valid" request. The
problem is that currently such requests internally explode into 100's of
large fetches, all of which are kept in memory in an inefficient way. This
is being addressed in two ways. By making things more efficient. And by
weeding out or simplifying requests deemed too unwieldy.

FIX
====

This vulnerability has been fixed in release 2.2.20 and further corrected
in 2.2.21. You are advised to upgrade to version 2.2.21 (or newer) or the
legacy 2.0.65 release, once this is published (anticipated in September).

If you cannot upgrade, or cannot wait to upgrade - you can apply the
appropriate source code patch and recompile a recent existing version;

  http://www.apache.org/dist/httpd/patches/apply_to_2.2.14/ (for 2.2.9 - .14)
  http://www.apache.org/dist/httpd/patches/apply_to_2.2.19/ (for 2.2.15 - .19)
  http://www.apache.org/dist/httpd/patches/apply_to_2.0.64/ (for 2.0.55 - .64)

If you cannot upgrade and/or cannot apply above patches in a timely manner
then you should consider to apply one or more of the mitigation suggested below.

CAVEATS
=======

Note that this fix 1) will return a "200 OK" in cases where a 206 respond would
be larger and 2) changes the behavior of chunked responses. This may affect
certain clients. See the above background section and IETF reference for
more detail and the various discussions around fixing this in the protocol.

Furthermore a request with a byterange beyond the end of the file used to
return 416 but now returns 200. This is a violation of a RFC2616 SHOULD.

Mitigation:
===========

There are several immediate options to mitigate this issue until a full fix
is available. Below examples handle both the 'Range' and the legacy
'Request-Range' with various levels of care.

```
Note that 'Request-Range' is a legacy name dating back to Netscape Navigator
2-3 and MSIE 3. Depending on your user community - it is likely that you
can use option '3' safely for this older 'Request-Range'.

0) Consult http://httpd.apache.org/security/CVE-2011-3192.txt for the most
   recent information (as this is the final advisory).

1) Use SetEnvIf or mod_rewrite to detect a large number of ranges and then
   either ignore the Range: header or reject the request.

   Option 1: (Apache 2.2, requires mod_setenvif and mod_headers)

          # Drop the Range header when more than 5 ranges.
          # CVE-2011-3192
          SetEnvIf Range (?:,.*?){5,5} bad-range=1
          RequestHeader unset Range env=bad-range

          # We always drop Request-Range; as this is a legacy
          # dating back to MSIE3 and Netscape 2 and 3.
          #
          RequestHeader unset Request-Range

          # optional logging.
          CustomLog logs/range-CVE-2011-3192.log common env=bad-range

   Above may not work for all configurations. In particular situations
   mod_cache and (language) modules may act before the 'unset'
   is executed upon during the 'fixup' phase.

   Option 2: (Pre 2.2, requires mod_rewrite and mod_headers)

          # Reject request when more than 5 ranges in the Range: header.
          # CVE-2011-3192
          #
          RewriteEngine on
          RewriteCond %{HTTP:range} !(^bytes=[^,]+(,[^,]+){0,4}$|^$) [NC]
          RewriteRule .* - [F]

          # We always drop Request-Range; as this is a legacy
          # dating back to MSIE3 and Netscape 2 and 3.
          #
          RequestHeader unset Request-Range

   The number 5 is arbitrary. Several 10's should not be an issue and may be
   required for sites which for example serve PDFs to very high end eReaders
   or use things such complex http based video streaming.

   WARNING These directives need to be specified in every configured
   vhost, or inherited from server context as described in:
   http://httpd.apache.org/docs/current/mod/mod_rewrite.html#vhosts

2) Use mod_headers to completely dis-allow the use of Range headers:

          RequestHeader unset Range

   Note that this may break certain clients - such as those used for
   e-Readers and progressive/http-streaming video.

   Furthermore to ignore the Netscape Navigator 2-3 and MSIE 3 specific
   legacy header - add:

          RequestHeader unset Request-Range

   Unlike the commonly used 'Range' header - dropping the 'Request-Range'
   is not likely to affect many clients.

4) Deploy a Range header count module as a temporary stopgap measure.

   A stop-gap module which is runtime-configurable can be found at:

     http://people.apache.org/~fuankg/httpd/mod_rangecnt-improved/
```

A simpler stop-gap module which requires compile-time configuration
is also available:

  http://people.apache.org/~dirkx/mod_rangecnt.c

Note
====

Earlier advisories suggested the use of LimitRequestFieldSize. This mitigation
was not fully effective and can by bypassed by splitting the attack vector up
across multiple headers. Therefore you should not rely on LimitRequestFieldSize
alone.

OS and Vendor specific information
==================================

Red Hat: Has additional RHEL specific information at:
                https://bugzilla.redhat.com/show_bug.cgi?id=732928

NetWare: Pre compiled binaries are available;
                runtime-configurable:
                http://people.apache.org/~fuankg/httpd/mod_rangecnt-improved/
                compile-time configured for 5 ranges:
                http://people.apache.org/~fuankg/httpd/mod_rangecnt/

Win32:   Pre compiled binaries are available;
                runtime-configurable:
                http://people.apache.org/~gsmith/httpd/binaries/modules/mod_rangecnt-improved/
                compile-time configured for 5 ranges:
                http://people.apache.org/~gsmith/httpd/binaries/modules/mod_rangecnt/

mod_security: Has updated their rule set; see
                http://blog.spiderlabs.com/2011/08/mitigation-of-apache-range-header-dos-attack.html

Actions:
========

Apache HTTPD users who are concerned about a DoS attack against their server
should 1) upgrade to version 2.2.21 (or 2.0.65 when it becomes available),
2) if not possible - apply the provided patches or 3) consider implementing
any of the above mitigation immediately.

When using a third party attack tool to verify vulnerability - note that most
of the versions in the wild currently check for the presence of mod_deflate;
and will (mis)report that your server is not vulnerable if this module is not
present. This vulnerability is not dependent on presence or absence of
that module.