Apache2 mod_proxy_uwsgi Incorrect Request Handling

Authored by Google Security Research, Felix Wilhelm

Posted Aug 31, 2020

Apache2 suffers from an incorrect handling of large requests issue in mod_proxy_uwsgi.

tags | advisory
advisories | CVE-2020-11984
SHA-256 | a6d25204a474a382b45dc4bcc2aef5cc3b47408552e918aedeac6dce35405571

Download | Favorite | View

Related Files

Share This

Like Twee LinkedIn Reddit Digg StumbleUpon

Change Mirror Download

```
Apache2: Incorrect handling of large requests in mod_proxy_uwsgi

mod_proxy_uwsgi as included in current versions of Apache httpd incorrectly handles large
HTTP requests. The UWSGI line protocol uses uint16_t length values for both header name/values
and the overall packet size, but mod_proxy_uwsgi does not verify that these size fields do not overflow:

// modules/proxy/mod_proxy_uwsgi.c
static int uwsgi_send_headers(request_rec *r, proxy_conn_rec * conn)
{
  char *buf, *ptr;

  const apr_array_header_t *env_table;
  const apr_table_entry_t *env;

  apr_size_t headerlen = 4;
  apr_uint16_t pktsize, keylen, vallen;

  [...]

  env_table = apr_table_elts(r->subprocess_env);
  env = (apr_table_entry_t *) env_table->elts;

  for (j = 0; j < env_table->nelts; ++j) {
    headerlen += 2 + strlen(env[j].key) + 2 + strlen(env[j].val);
  }

  ptr = buf = apr_palloc(r->pool, headerlen);

  ptr += 4;

  for (j = 0; j < env_table->nelts; ++j) {
    keylen = strlen(env[j].key); ** A **
    *ptr++ = (apr_byte_t) (keylen & 0xff);
    *ptr++ = (apr_byte_t) ((keylen >> 8) & 0xff);
    memcpy(ptr, env[j].key, keylen);
    ptr += keylen;

    vallen = strlen(env[j].val); ** B **
    *ptr++ = (apr_byte_t) (vallen & 0xff);
    *ptr++ = (apr_byte_t) ((vallen >> 8) & 0xff);
    memcpy(ptr, env[j].val, vallen);
    ptr += vallen;
  }

  pktsize = headerlen - 4; ** C **

  buf[0] = 0;
  buf[1] = (apr_byte_t) (pktsize & 0xff);
  buf[2] = (apr_byte_t) ((pktsize >> 8) & 0xff);
  buf[3] = 0;

  return uwsgi_send(conn, buf, headerlen, r);
}

A malicious request can easily overflow pktsize (C) by sending
a small amount of headers with a length that is close to the LimitRequestFieldSize
default value of 8190. This can be used to trick UWSGI into parsing parts of
the serialized subprocess environment as part of the POST body.
In most configurations the security impact of this seems to be limited. However,
an attacker might be able to leak sensitive environment variables as part of the POST body and/or
strip security sensitive headers from the request.
If UWSGI is explicitly configured in persistent mode (puwsgi), this can
also be used to smuggle a second UWSGI request leading to remote code execution.
(In its standard configuration UWSGI only supports a single request per connection,
making request smuggling impossible)

RCE against a standard UWSGI config is possible if an attacker can put a controlled
name or value into subprocess_env that is longer than 0xFFFF bytes:
This would overflow the size calculation in (A) or (B) and
makes it possible to inject malicious key/value pairs into the UWSGI request. This can be turned
into code execution by setting a malicious UWSGI_FILE var
(see https://github.com/wofeiwo/webcgi-exploits/blob/master/python/uwsgi-rce-zh.md)

Using an oversized HTTP header for this attack requires a LimitRequestFieldSize
bypass and should not be possible in normal configurations.
However, mod_http2 incorrectly enforced LimitRequestFieldSize before
R1863276 (https://svn.apache.org/viewvc?view=revision&revision=1863276) so systems
without this commit can be exploited easily. Other config dependent attack vectors might exist.


Credits:
Felix Wilhelm of Google Project Zero

This bug is subject to a 90 day disclosure deadline. After 90 days elapse, the bug report
will become visible to the public. The scheduled disclosure date is 2020-07-23.
Disclosure at an earlier date is also possible if agreed upon by all parties.

Related CVE Numbers: CVE-2020-11984.



Found by: fwilhelm@google.com
```

Login or Register to add favorites

Follow us on Twitter

Subscribe to an RSS Feed

File Archive: December 2022 <

Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Top Authors In Last 30 Days

Red Hat 154 files
Ubuntu 73 files
LiquidWorm 23 files
Debian 18 files
malvuln 11 files
nu11secur1ty 11 files
Gentoo 9 files
Google Security Research 8 files
T. Weber 4 files
Julien Ahrens 4 files

File Tags

ActiveX (932)
Advisory (79,754)
Arbitrary (15,694)
BBS (2,859)
Bypass (1,619)
CGI (1,018)
Code Execution (6,926)
Conference (673)
Cracker (840)
CSRF (3,290)
DoS (22,602)
Encryption (2,349)
Exploit (50,359)
File Inclusion (4,165)
File Upload (946)
Firewall (821)
Info Disclosure (2,660)
Intrusion Detection (867)
Java (2,899)
JavaScript (821)
Kernel (6,291)
Local (14,201)
Magazine (586)
Overflow (12,419)
Perl (1,418)
PHP (5,093)
Proof of Concept (2,291)
Protocol (3,435)
Python (1,467)
Remote (30,044)
Root (3,504)
Ruby (594)
Scanner (1,631)
Security Tool (7,777)
Shell (3,103)
Shellcode (1,204)
Sniffer (886)

File Archives

December 2022
November 2022
October 2022
September 2022
August 2022
July 2022
June 2022
May 2022
April 2022
March 2022
February 2022
January 2022
Older

Systems

AIX (426)
Apple (1,926)
BSD (370)
CentOS (55)
Cisco (1,917)
Debian (6,634)
Fedora (1,690)
FreeBSD (1,242)
Gentoo (4,272)
HPUX (878)
iOS (330)
iPhone (108)
IRIX (220)
Juniper (67)
Linux (44,315)
Mac OS X (684)
Mandriva (3,105)
NetBSD (255)
OpenBSD (479)
RedHat (12,469)
Slackware (941)
Solaris (1,607)

Spoof (2,166)
SQL Injection (16,102)
TCP (2,379)
Trojan (686)
UDP (876)
Virus (662)
Vulnerability (31,136)
Web (9,365)
Whitepaper (3,729)
x86 (946)
XSS (17,494)
Other

SUSE (1,444)
Ubuntu (8,199)
UNIX (9,159)
UnixWare (185)
Windows (6,511)
Other

packet storm

Site Links

- News by Month
- News Tags
- Files by Month
- File Tags
- File Directory

About Us

- History & Purpose
- Contact Information
- Terms of Service
- Privacy Statement
- Copyright Information

Hosting By

- Rokasec

Follow us on Twitter

Subscribe to an RSS Feed