Closed Issue created 1 year ago by **Samanta Navarro**

libfetch information leak or crash

Dear Alpine Linux team,

I have discovered a potentially security-relevant issue in libfetch. It is used in apk and I have reported the issue to FreeBSD upstream. Maybe you want to be informed about this before it is fixed there.

This is the mail I have just sent to the FreeBSD security team.

Problem:

The passive mode in FTP communication allows an out of boundary read while libfetch uses strtol to parse the relevant numbers into address bytes. It does not check if the line ends prematurely. If it does, the for-loop condition checks for *p == '\0' one byte too late because p++ was already performed.

Impact:

The connection buffer size can be controlled by a malicious FTP server because the size is increased until a newline is encountered (or no more characters are read). This also allows to move the buffer into more interesting areas within the address space, potentially parsing relevant numbers for the attacker.

Since these bytes become available to the server in form of a new TCP connection to a constructed port number or even part of the IPv6 address this is a potential information leak.

Proof of Concept:

Set up the malicious FTP server like this

```
cat > replies.txt.b64 << EOF
begin-base64 644 replies.txt
MjIwIG9rCjIzMCBvawoyNTcgIi8iCjIwMCBvawoyMDAgb2sKMjEzIDE4NDQ2NzQ0MDczNzA5NTUx
NjE2CjIxMyAwMDAwMDAwMDAwMDAwMDAwMAoyMDAgb2sKMjAwIG9rCjIyOCAxCjIyOCAxMTExMTExMTEx
MTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTEx
MTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTEx
MTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTEx
MTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTEx
MTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTEx
MTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTEx
MTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTEx
MTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTEx
MTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTEx
MTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTEx
MTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTEx
MTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTEx
MTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTEx
MTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTEx
MTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTEx
MTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTEx
MTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTEx
MTExMTExMTExMTExMTExMTExMTEx
====
EOF
b64decode -o replies.txt replies.txt.b64
nc -Nl ::1 21 < replies.txt
```

Compile libfetch and fetch with CFLAGS="-fsanitize=address -fsanitize=undefined" and start client

```
fetch 'ftp://[::1]/poc'
```

Considerations:

Since libfetch is used outside of FreeBSD as well, e.g. in Alpine Linux package keeper apk, I recommend to issue a CVE for this so these users are informed about the patch as well.

The information leak is fixed in the second ftp.c patch chunk.

Sincerely, Samanta

```
Index: fetch.c
===================================================================
--- fetch.c	(revision 370066)
+++ fetch.c	(working copy)
@@ -421,7 +421,7 @@
        /* port */
        if (*p == ':') {
                for (n = 0, q = ++p; *q && (*q != '/'); q++) {
-                       if (*q >= '0' && *q <= '9' && n < INT_MAX / 10) {
+                       if (*q >= '0' && *q <= '9' && n < IPPORT_MAX) {
                                n = n * 10 + (*q - '0');
                        } else {
                                /* invalid port */
Index: ftp.c
===================================================================
--- ftp.c	(revision 370066)
+++ ftp.c	(working copy)
@@ -424,8 +424,14 @@
        }
        for (ln = conn->buf + 4; *ln && isspace((unsigned char)*ln); ln++)
                /* nothing */ ;
-       for (us->size = 0; *ln && isdigit((unsigned char)*ln); ln++)
+       for (us->size = 0; *ln && isdigit((unsigned char)*ln); ln++) {
+               if (us->size > OFF_MAX / 10 - (*ln - '0')) {
+                       ftp_seterr(FTP_PROTOCOL_ERROR);
+                       us->size = -1;
+                       return (-1);
+               }
                us->size = us->size * 10 + *ln - '0';
+       }
        if (*ln && !isspace((unsigned char)*ln)) {
                ftp_seterr(FTP_PROTOCOL_ERROR);
                us->size = -1;
@@ -704,8 +710,11 @@
                                goto ouch;
                        }
                        l = (e == FTP_PASSIVE_MODE ? 6 : 21);
-                       for (i = 0; *p && i < l; i++, p++)
+                       for (i = 0; *p && i < l; i++, p++) {
                                addr[i] = strtol(p, &p, 10);
+                               if (*p == '\0' && i < l - 1)
+                                       break;
+                       }
                        if (i < l) {
                                e = FTP_PROTOCOL_ERROR;
                                goto ouch;
Index: http.c
===================================================================
--- http.c	(revision 370066)
+++ http.c	(working copy)
@@ -163,11 +163,15 @@
                if (!isxdigit((unsigned char)*p))
                        return (-1);
                if (isdigit((unsigned char)*p)) {
+                       if (io->chunksize > OFF_MAX / 16 - (*p - '0'))
+                               return (-1);
                        io->chunksize = io->chunksize * 16 +
                            *p - '0';
                } else {
-                       io->chunksize = io->chunksize * 16 +
-                           10 + tolower((unsigned char)*p) - 'a';
+                       int digit = 10 + tolower((unsigned char)*p) - 'a';
+                       if (io->chunksize > OFF_MAX / 16 - digit)
+                               return (-1);
+                       io->chunksize = io->chunksize * 16 + digit;
                }
        }

@@ -908,8 +912,11 @@
 {
        off_t len;

-       for (len = 0; *p && isdigit((unsigned char)*p); ++p)
+       for (len = 0; *p && isdigit((unsigned char)*p); ++p) {
+               if (len > OFF_MAX / 10 - (*p - '0'))
```

```
+                       return (-1);
                len = len * 10 + (*p - '0');
+       }
        if (*p)
                return (-1);
        DEBUGF("content length: [%lld]\n", (long long)len);
@@ -932,17 +939,26 @@
                first = last = -1;
                ++p;
        } else {
-               for (first = 0; *p && isdigit((unsigned char)*p); ++p)
+               for (first = 0; *p && isdigit((unsigned char)*p); ++p) {
+                       if (first > OFF_MAX / 10 - (*p - '0'))
+                               return (-1);
+                       first = first * 10 + *p - '0';
+               }
                if (*p != '-')
                        return (-1);
-               for (last = 0, ++p; *p && isdigit((unsigned char)*p); ++p)
+               for (last = 0, ++p; *p && isdigit((unsigned char)*p); ++p) {
+                       if (last > OFF_MAX / 10 - (*p - '0'))
+                               return (-1);
+                       last = last * 10 + *p - '0';
+               }
        }
        if (first > last || *p != '/')
                return (-1);
-       for (len = 0, ++p; *p && isdigit((unsigned char)*p); ++p)
+       for (len = 0, ++p; *p && isdigit((unsigned char)*p); ++p) {
+               if (len > OFF_MAX / 10 - (*p - '0'))
+                       return (-1);
+               len = len * 10 + *p - '0';
+       }
        if (*p || len < last - first + 1)
                return (-1);
        if (first == -1) {
```

Drag your designs here or click to upload.

Tasks 0

No tasks are currently assigned. Use tasks to break down this issue into smaller parts.

Linked items 0

Related merge requests 1

libfetch: fix out of bounds accesses for passive FTP and HTTP implementations

!64

When this merge request is accepted, this issue will be closed automatically.

Ariadne Conill @ariadne · 1 year ago (Developer)

Do we need a CVE identifier for this?

Ariadne Conill @ariadne · 1 year ago (Developer)

Ah, you haven't requested one yet. I can do it, or I can coordinate with baptiste over at FreeBSD.

Samanta Navarro @ferivoz · 1 year ago (Author)

Hi @kaniini, this is correct. I did not request a CVE because I wanted to let FreeBSD decide if it is security-relevant from their point of view. But since you use libfetch in a slightly older version as well, I thought that it is only fair to give you the chance to evaluate this issue on your own, too.

It makes sense if you coordinate this with baptiste. Thank you in advance!

Ariadne Conill @ariadne · 1 year ago (Developer)

I have gone ahead and requested a CVE.

Ariadne Conill @ariadne · 1 year ago (Developer)

VoidLinux also uses libfetch for xbps, I have notified them as well.

Ariadne Conill @ariadne mentioned in merge request !64 (closed) 1 year ago

Ariadne Conill @ariadne · 1 year ago (Developer)

This is CVE-2021-36159.

Timo Teräs @fabled · 1 year ago (Maintainer)

I've been looking the patch here, and the MR. And while it's mostly doing the job, there's a minor omission in the port parsing, and I'd prefer to make the string parsing a separate function.

Does the following work equally good?

```
diff --git a/libfetch/common.c b/libfetch/common.c
index bcba889..03cf2a0 100644
--- a/libfetch/common.c
+++ b/libfetch/common.c
@@ -1210,3 +1210,27 @@ fetchIO_write(fetchIO *f, const void *buf, size_t len)
                return EBADF;
        return (*f->io_write)(f->io_cookie, buf, len);
 }
+
+uintmax_t
+parse_unsigned(const char *str, const char **endptr, int radix, uintmax_t max)
+{
+       uintmax_t val = 0, maxx = max / radix, d;
+       const char *p;
+
+       for (p = str; *p && isxdigit((unsigned char)*p); p++) {
+               unsigned char ch = (unsigned char)*p;
+               if (isdigit(ch))
+                       d = ch - '0';
+               else    d = tolower(ch - 'a');
+               if (d > radix || val > maxx) goto err;
+               val *= radix;
+               if (val > max-d) goto err;
+               val += d;
+       }
+       if (p == str || val > max) goto err;
+       *endptr = p;
+       return val;
+err:
+       *endptr = "\xff";
+       return 0;
+}
diff --git a/libfetch/common.h b/libfetch/common.h
index dd5c14c..f2d7ef2 100644
--- a/libfetch/common.h
+++ b/libfetch/common.h
@@ -38,6 +38,8 @@
 #define FTP_DEFAULT_PROXY_PORT 21
 #define HTTP_DEFAULT_PROXY_PORT        3128

+#include <sys/types.h>
+#include <limits.h>
 #include "openssl-compat.h"

 #if defined(__GNUC__) && __GNUC__ >= 3
```

```
@@ -53,6 +55,14 @@
 #define HAVE_SA_LEN
 #endif

+#ifndef IPPORT_MAX
+# define IPPORT_MAX 65535
+#endif
+
+#ifndef OFF_MAX
+# define OFF_MAX (((((off_t)1 << (sizeof(off_t) * CHAR_BIT - 2)) - 1) << 1) + 1)
+#endif
+
 /* Connection */
 typedef struct fetchconn conn_t;

@@ -125,6 +135,7 @@ fetchIO            *http_request(struct url *, const char *,
 fetchIO                *ftp_request(struct url *, const char *, const char *,
                     struct url_stat *, struct url *, const char *);

+uintmax_t      parse_unsigned(const char *p, const char **endptr, int radix, uintmax_t max);

 /*
  * Check whether a particular flag is set
diff --git a/libfetch/fetch.c b/libfetch/fetch.c
index a0d4dbd..2bb076e 100644
--- a/libfetch/fetch.c
+++ b/libfetch/fetch.c
@@ -473,15 +473,12 @@ find_user:

        /* port */
        if (*p == ':') {
-               for (q = ++p; *q && (*q != '/'); q++)
-                       if (isdigit((unsigned char)*q))
-                               u->port = u->port * 10 + (*q - '0');
-                       else {
-                               /* invalid port */
-                               url_seterr(URL_BAD_PORT);
-                               goto ouch;
-                       }
-               p = q;
+               u->port = parse_unsigned(p + 1, &p, 10, IPPORT_MAX);
+               if (*p && *p != '/') {
+                       /* invalid port */
+                       url_seterr(URL_BAD_PORT);
+                       goto ouch;
+               }
        }

        /* document */
diff --git a/libfetch/ftp.c b/libfetch/ftp.c
index 8f9f04f..7ca29d7 100644
--- a/libfetch/ftp.c
+++ b/libfetch/ftp.c
@@ -471,8 +471,7 @@ ftp_stat(conn_t *conn, const char *file, struct url_stat *us)
        }
        for (ln = conn->buf + 4; *ln && isspace((unsigned char)*ln); ln++)
                /* nothing */ ;
-       for (us->size = 0; *ln && isdigit((unsigned char)*ln); ln++)
-               us->size = us->size * 10 + *ln - '0';
+       us->size = parse_unsigned(ln, (const char **) &ln, 10, OFF_MAX);
        if (*ln && !isspace((unsigned char)*ln)) {
                ftp_seterr(FTP_PROTOCOL_ERROR);
                us->size = -1;
@@ -700,7 +699,7 @@ retry_mode:

        if (pasv) {
                unsigned char addr[64];
-               char *ln, *p;
+               const char *ln, *p;
                unsigned int i;
                int port;

@@ -737,10 +736,14 @@ retry_mode:
                        for (p = ln + 3; *p && !isdigit((unsigned char)*p); p++)
                                /* nothing */ ;
                        if (!*p) goto protocol_error;
-                       l = (e == FTP_PASSIVE_MODE ? 6 : 21);
-                       for (i = 0; *p && i < l; i++, p++)
-                               addr[i] = strtol(p, &p, 10);
-                       if (i < l) goto protocol_error;
+                       l = (e == FTP_PASSIVE_MODE ? 6 : 21) - 1;
+                       for (i = 0; *p && i < l; i++, p++) {
+                               while (isspace((unsigned char)*p)) p++;
+                               addr[i] = parse_unsigned(p, &p, 10, UCHAR_MAX);
+                               if (*p != ',') goto protocol_error;
+                       }
+                       addr[i] = parse_unsigned(p, &p, 10, UCHAR_MAX);
+                       if (*p && *p != ')') goto protocol_error;
                        break;
                case FTP_EPASSIVE_MODE:
                        for (p = ln + 3; *p && *p != '('; p++)
diff --git a/libfetch/http.c b/libfetch/http.c
index 59d6292..da46456 100644
--- a/libfetch/http.c
+++ b/libfetch/http.c
@@ -134,29 +134,19 @@ struct httpio
 static int
 http_new_chunk(struct httpio *io)
 {
-       char *p;
+       const char *p;

        if (fetch_getln(io->conn) == -1)
-               return (-1);
+               return -1;

-       if (io->conn->buflen < 2 || !isxdigit((unsigned char)*io->conn->buf))
-               return (-1);
+       if (io->conn->buflen < 2)
+               return -1;

-       for (p = io->conn->buf; *p && !isspace((unsigned char)*p); ++p) {
-               if (*p == ';')
-                       break;
-               if (!isxdigit((unsigned char)*p))
-                       return (-1);
-               if (isdigit((unsigned char)*p)) {
-                       io->chunksize = io->chunksize * 16 +
-                           *p - '0';
-               } else {
-                       io->chunksize = io->chunksize * 16 +
-                           10 + tolower((unsigned char)*p) - 'a';
-               }
-       }
+       io->chunksize = parse_unsigned(io->conn->buf, &p, 16, SIZE_MAX);
+       if (*p && *p != ';' && !isspace(*p))
+               return -1;

-       return (io->chunksize);
+       return io->chunksize;
 }

 /*
@@ -501,22 +491,6 @@ http_parse_mtime(const char *p, time_t *mtime)
        return (0);
 }

-/*
- * Parse a content-length header
- */
-static int
-http_parse_length(const char *p, off_t *length)
-{
-       off_t len;
-
```

```
-       for (len = 0; *p && isdigit((unsigned char)*p); ++p)
-               len = len * 10 + (*p - '0');
-       if (*p)
-               return (-1);
-       *length = len;
-       return (0);
-}
-
 /*
  * Parse a content-range header
  */
@@ -532,17 +506,14 @@ http_parse_range(const char *p, off_t *offset, off_t *length, off_t *size)
                first = last = -1;
                ++p;
        } else {
-               for (first = 0; *p && isdigit((unsigned char)*p); ++p)
-                       first = first * 10 + *p - '0';
+               first = parse_unsigned(p, &p, 10, OFF_MAX);
                if (*p != '-')
                        return (-1);
-               for (last = 0, ++p; *p && isdigit((unsigned char)*p); ++p)
-                       last = last * 10 + *p - '0';
+               last = parse_unsigned(p+1, &p, 10, OFF_MAX);
        }
        if (first > last || *p != '/')
                return (-1);
-       for (len = 0, ++p; *p && isdigit((unsigned char)*p); ++p)
-               len = len * 10 + *p - '0';
+       len = parse_unsigned(p+1, &p, 10, OFF_MAX);
        if (*p || len < last - first + 1)
                return (-1);
        if (first == -1)
@@ -850,7 +821,7 @@ http_request(struct url *URL, const char *op, struct url_stat *us,
        int e, i, n;
        off_t offset, clength, length, size;
        time_t mtime;
-       const char *p;
+       const char *p, *q;
        fetchIO *f;
        hdr_t h;
        char hbuf[URL_HOSTLEN + 7], *host;
@@ -1050,13 +1021,16 @@ http_request(struct url *URL, const char *op, struct url_stat *us,
                                keep_alive = (strcasecmp(p, "keep-alive") == 0);
                                break;
                        case hdr_content_length:
-                               http_parse_length(p, &clength);
+                               clength = parse_unsigned(p, &q, 10, OFF_MAX);
+                               if (*q) goto protocol_error;
                                break;
                        case hdr_content_range:
-                               http_parse_range(p, &offset, &length, &size);
+                               if (http_parse_range(p, &offset, &length, &size) < 0)
+                                       goto protocol_error;
                                break;
                        case hdr_last_modified:
-                               http_parse_mtime(p, &mtime);
+                               if (http_parse_mtime(p, &mtime) < 0)
+                                       goto protocol_error;
```

Ariadne Conill @ariadne · 1 year ago

Developer

Yes, I think that patch is better.

Timo Teräs @fabled · 1 year ago

Maintainer

I pushed the `goto` merging as separate commit already, and updated the patch in the earlier comment on top of that. Notably in addition to the original patch, it fixes the following:

- http_parse_{range,mtime} return code is now checked
- The suggested range check

```
+               if (len > OFF_MAX / 10 - (*p - '0'))
+                       return (-1);
```

should be

```
+               if (len > (OFF_MAX - *p + '0') / 10)
+                       return (-1);
```

or similar to allow for the maximum value. I think there might be failure to parse OFF_MAX otherwise.

- The passive mode reply checking should be more rigorous now. Hopefully it did not break anything.

I'll commit, backport and will do stable releases during near days if this looks good. I'll try to test the FTP side of things still.

Samanta Navarro @ferivoz · 1 year ago

Author

You are right, the OFF_MAX check in my patch does not work properly. @kaniini, can you inform the other libfetch users/maintainers with whom you are in contact that my patch is not fully functional?

I have checked the suggested patch as well. The "maxx-d" issue in parse_unsigned has been already fixed, so nothing left I could comment on.

Thank you @fabled for writing a more comprehensive patch. I have kept mine small to highlight the parts which trigger the issues. Mixing the information leak and undefined behavior in one patch already felt like too much.

Please register or sign in to reply

Timo Teräs @fabled · 1 year ago

Maintainer

Would the following be suitable commit message?

```
libfetch: fix range checking for http/ftp protocol parsing

Various parsing of numeric strings were not having adequate range
checking causing information leak or potential crash.

CVE-2021-36159

Co-authored-by: Ariadne Conill <ariadne@dereferenced.org>
Reported-by: Samanta Navarro <ferivoz@riseup.net>
```

Ariadne Conill @ariadne · 1 year ago

Developer

Yes looks good to me!

Ariadne Conill @ariadne · 1 year ago

Developer

(I think we should remove the FTP support since I don't think there are any Alpine mirrors using it, but we can do that later.)

Samanta Navarro @ferivoz · 1 year ago

Author

If FTP is not needed I recommend that as well. It is easy to switch to FTP even if HTTP is initially used due to redirects supporting ftp://.

I have avoided this in my PoC to keep it straight to the point without having another netcat instance pretending to be an HTTP server.

Please register or sign in to reply

Timo Teräs @fabled closed via commit ca1d975e 1 year ago

Timo Teräs @fabled mentioned in commit 36048e8f 1 year ago

Timo Teräs @fabled mentioned in commit 9d967bb6 1 year ago

Ariadne Conill @ariadne made the issue visible to everyone 1 year ago

Ariadne Conill @ariadne · 1 year ago Developer
Declassifying this.

Martin Falatic @MartinFalatic · 1 year ago
I see this was released into 3.14 - when would we expect to see this critical fix in 3.12 and other affected (Docker image) releases? The only mitigation appears to be to use the vulnerable version of `apk` to upgrade itself, when one really shouldn't be using the vulnerable version of `apk` at all.

Note: The only other place that seemed appropriate for this question was on github's docker-alpine repo. However, I'm not sure if that's the official repo for this critical issue (it hasn't garnered much attention there.)

Ariadne Conill @ariadne · 1 year ago Developer
It was released into 3.12 as part of apk-tools 2.10.7.

The apk team does not manage Alpine releases, there will be a new set of releases for 3.11/3.12/3.13 soon including this fix.

Martin Falatic @MartinFalatic · 1 year ago
Who manages Alpine releases, and is there a more appropriate repo or forum for questions like this (e.g., "when will existing fix X be in release line Y?")

Edit: I'm also trying to get a gauge of how soon "soon" is - if there's a more detailed discussion/timeline elsewhere it'd be good to be able to reference it for the purpose of internal tracking here.

Edited by Martin Falatic 1 year ago

Please register or sign in to reply