cesanta / **mongoose** Public

Code Issues 4 Pull requests 5 Discussions Actions Projects

Commit `0d882f1`

scaprile committed 2 days ago

```
patch
```

`master` · 7.21

1 parent `1bb8579` commit `0d882f1`

13 files changed +528 -252 lines changed

Top

Filter files…

- mongoose.c
- mongoose.h
- src
  - dns.c
  - http.c
  - json.c
  - net_builtin.c
  - str.c
  - tls_aes128.c
  - tls_aes128.h
  - tls_builtin.c
  - tls_chacha20.c
  - tls_chacha20.h
- test
  - unit_test.c

13 files changed +528 -252 lines changed

Search within code

mongoose.c

Load Diff

Large diffs are not rendered by default.

mongoose.h

```
@@ -1961,7 +1961,9 @@ int mg_aes_gcm_encrypt(unsigned char *output, const unsigned char *input,
1961 1961  int mg_aes_gcm_decrypt(unsigned char *output, const unsigned char *input,
1962 1962                        size_t input_length, const unsigned char *key,
1963 1963                        const size_t key_len, const unsigned char *iv,
1964      -                     const size_t iv_len);
     1964 +                     const size_t iv_len, unsigned char *aead,
     1965 +                     size_t aead_len, const unsigned char *tag,
     1966 +                     const size_t tag_len);
1965 1967
1966 1968  #endif /* TLS_AES128_H */
1967 1969
@@ -2727,6 +2729,7 @@ PORTABLE_8439_DECL size_t mg_chacha20_poly1305_encrypt(
2727 2729  PORTABLE_8439_DECL size_t mg_chacha20_poly1305_decrypt(
2728 2730      uint8_t *restrict plain_text, const uint8_t key[RFC_8439_KEY_SIZE],
2729 2731      const uint8_t nonce[RFC_8439_NONCE_SIZE],
     2732 +    const uint8_t *restrict ad, size_t ad_size,
2730 2733      const uint8_t *restrict cipher_text, size_t cipher_text_size);
2731 2734  #if defined(__cplusplus)
2732 2735  }
```

src/dns.c

```
@@ -287,13 +287,15 @@ void mg_resolve(struct mg_connection *c, const char *url) {
287 287    }
```

```
288 288   }
289 289
    290 + // Response header length is 10 bytes
290 291   static const uint8_t mdns_answer[] = {
291 292       0, 1,          // 2 bytes - record type, A
292 293       0, 1,          // 2 bytes - address class, INET
293 294       0, 0, 0, 120,  // 4 bytes - TTL
294 295       0, 4           // 2 bytes - address length
295 296   };
296 297
    298 + // A name length is name->len + '.local' + 2 = name->len + 8
297 299   static uint8_t *build_name(struct mg_str *name, uint8_t *p) {
298 300     *p++ = (uint8_t) name->len;  // label 1
299 301     memcpy(p, name->buf, name->len), p += name->len;
        @@ -305,6 +307,7 @@ static uint8_t *build_name(struct mg_str *name, uint8_t
        *p) {
305 307
306 308   void mg_getlocaddr(struct mg_connection *, struct mg_addr *, struct mg_addr *);
307 309
    310 + // An A record length is 10 + 4 = 14 bytes
308 311   static uint8_t *build_a_record(struct mg_connection *c, uint8_t *p,
309 312                                  struct mg_addr *addr) {
310 313     memcpy(p, mdns_answer, sizeof(mdns_answer)), p += sizeof(mdns_answer);
        @@ -326,6 +329,7 @@ static uint8_t *build_a_record(struct mg_connection *c,
        uint8_t *p,
326 329     return p;
327 330   }
328 331
    332 + // A srv name length is r->srvcproto.len + '.local' + 2 = r->srvcproto.len + 8
329 333   static uint8_t *build_srv_name(uint8_t *p, struct mg_dnssd_record *r) {
330 334     *p++ = (uint8_t) r->srvcproto.len - 5;  // label 1, up to '._tcp'
331 335     memcpy(p, r->srvcproto.buf, r->srvcproto.len), p += r->srvcproto.len;
        @@ -346,6 +350,7 @@ static uint8_t *build_mysrv_name(struct mg_str *name,
        uint8_t *p,
346 350   }
347 351   #endif
348 352
    353 + // A PTR record length is 10 + name->len + 3 = name->len + 13
349 354   static uint8_t *build_ptr_record(struct mg_str *name, uint8_t *p, uint16_t o) {
350 355     uint16_t offset = mg_htons(o);
```

```
351 356       memcpy(p, mdns_answer, sizeof(mdns_answer));
    @@ -360,6 +365,7 @@ static uint8_t *build_ptr_record(struct mg_str *name,
    uint8_t *p, uint16_t o) {
360 365       return p;
361 366     }
362 367
    368 + // An SRV record length is 10 + name->len + 9 = name->len + 19
363 369     static uint8_t *build_srv_record(struct mg_str *name, uint8_t *p,
364 370                                      struct mg_dnssd_record *r, uint16_t o) {
365 371       uint16_t port = mg_htons(r->port);
    @@ -380,6 +386,7 @@ static uint8_t *build_srv_record(struct mg_str *name,
    uint8_t *p,
380 386       return p;
381 387     }
382 388
    389 + // A TXT record length is r->txt.len (txt contents) + 10
383 390     static uint8_t *build_txt_record(uint8_t *p, struct mg_dnssd_record *r) {
384 391       uint16_t len = mg_htons((uint16_t) r->txt.len);
385 392       memcpy(p, mdns_answer, sizeof(mdns_answer));
    @@ -390,6 +397,8 @@ static uint8_t *build_txt_record(uint8_t *p, struct
    mg_dnssd_record *r) {
390 397       return p;
391 398     }
392 399
    400 + // Each additional record has a 2-byte field pointing to the name label
    401 +
393 402     // RFC-6762 16: case-insensitivity --> RFC-1034, 1035
394 403
395 404     static void handle_mdns_query(struct mg_connection *c) {
    @@ -478,6 +487,10 @@ static void handle_mdns_query(struct mg_connection *c)
    {
478 487         uint8_t *o = p, *aux;
479 488         uint16_t offset;
480 489         if (respname->buf == NULL || respname->len == 0) return;
    490 +       if ((sizeof(*h) + req.r->srvcproto.len + 8 + respname->len + 13 + 2 +
    491 +            respname->len + 19 + 2 + req.r->txt.len + 10 + 2 + 14) >
    492 +           sizeof(buf))  // srv name + PTR + 2 + SRV + 2 + TXT + 2 + A
    493 +         return;
481 494         h->num_other_prs = mg_htons(3);  // 3 additional records
482 495         p = build_srv_name(p, req.r);
```

```
483  496          aux = build_ptr_record(respname, p, (uint16_t) (o - buf));
     @@ -498,12 +511,18 @@ static void handle_mdns_query(struct mg_connection *c)
     {
498  511          *p |= 0xC0, p += 2;
499  512          p = build_a_record(c, p, req.addr);
500  513        } else if (rr.atype == MG_DNS_RTYPE_TXT) {
     514  +       if ((sizeof(*h) + req.r->srvcproto.len + 8 + req.r->txt.len + 10) >
     515  +           sizeof(buf))  // srv name + TXT
     516  +         return;
501  517          p = build_srv_name(p, req.r);
502  518          p = build_txt_record(p, req.r);
503  519        } else if (rr.atype == MG_DNS_RTYPE_SRV) {  // serve SRV + A
504  520          uint8_t *o, *aux;
505  521          uint16_t offset;
506  522          if (respname->buf == NULL || respname->len == 0) return;
     523  +       if ((sizeof(*h) + req.r->srvcproto.len + 8 + respname->len + 19 + 2 +
     524  +            14) > sizeof(buf))  // srv name + SRV + 2 + A
     525  +         return;
507  526          h->num_other_prs = mg_htons(1);  // 1 additional record
508  527          p = build_srv_name(p, req.r);
509  528          o = p - 7;  // point to '.local' label (\x05local\x00)
     @@ -517,6 +536,8 @@ static void handle_mdns_query(struct mg_connection *c) {
517  536        } else {  // A requested
518  537          // RFC-6762 6: 0 Auth, 0 Additional RRs
519  538          if (respname->buf == NULL || respname->len == 0) return;
     539  +       if ((sizeof(*h) + respname->len + 8 + 14) > sizeof(buf))  // name + A
     540  +         return;
520  541          p = build_name(respname, p);
521  542          p = build_a_record(c, p, req.addr);
522  543        }
```

src/http.c

```
     @@ -237,6 +237,8 @@ static const char *skiptorn(const char *s, const char
     *end, struct mg_str *v) {
237  237    static bool mg_http_parse_headers(const char *s, const char *end,
238  238                                      struct mg_http_header *h, size_t max_hdrs)
     {
239  239      size_t i, n;
     240  +   int cl_count = 0, te_count = 0, auth_count = 0;
```

```
+   int conn_count = 0, cookie_count = 0;
    for (i = 0; i < max_hdrs; i++) {
      struct mg_str k = {NULL, 0}, v = {NULL, 0};
      if (s >= end) return false;
@@ -252,6 +254,13 @@ static bool mg_http_parse_headers(const char *s,
const char *end,
      while (v.len > 0 && (v.buf[v.len - 1] == ' ' || v.buf[v.len - 1] ==
'\t')) {
        v.len--;  // Trim spaces
      }
+     // detect duplicated headers -> discard
+     if (((mg_strcasecmp(k, mg_str("Content-Length")) == 0) && (++cl_count >
1)) ||
+       ((mg_strcasecmp(k, mg_str("Transfer-Encoding")) == 0) && (++te_count >
1)) ||
+       ((mg_strcasecmp(k, mg_str("Authorization")) == 0) && (++auth_count >
1)) ||
+       ((mg_strcasecmp(k, mg_str("Cookie")) == 0) && (++cookie_count > 1)) ||
+       ((mg_strcasecmp(k, mg_str("Connection")) == 0) && (++conn_count > 1)))
+       return false;
      // MG_INFO(("--HH [%.*s] [%.*s]", (int) k.len, k.buf, (int) v.len,
v.buf));
      h[i].name = k, h[i].value = v;  // Success. Assign values
    }
@@ -286,6 +295,8 @@ int mg_http_parse(const char *s, size_t len, struct
mg_http_message *hm) {
    // If we're given a version, check that it is HTTP/x.x
    version_prefix_valid =
        hm->proto.len > 5 && (mg_ncasecmp(hm->proto.buf, "HTTP/", 5) == 0);
+   if (!is_response && !version_prefix_valid)
+     return -1; // no version detected in request
    if (!is_response && hm->proto.len > 0 &&
        (!version_prefix_valid || hm->proto.len != 8 ||
         (hm->proto.buf[5] < '0' || hm->proto.buf[5] > '9') ||
@@ -1035,7 +1046,11 @@ static void http_cb(struct mg_connection *c, int
ev, void *ev_data) {
          hm.message.len = c->recv.len - ofs;  // and closes now, deliver MSG
          hm.body.len = hm.message.len - (size_t) (hm.body.buf -
hm.message.buf);
        }
```

```
1038      -       if ((te = mg_http_get_header(&hm, "Transfer-Encoding")) != NULL) {
      1049  +       bool is_http_1_0 =
      1050  +           hm.proto.len > 8 && mg_ncasecmp(hm.proto.buf, "HTTP/1.0", 8) == 0;
      1051  +       // HTTP/1.0 does not use "Transfer-Encoding: chunked"
      1052  +       if (!is_http_1_0 &&
      1053  +           (te = mg_http_get_header(&hm, "Transfer-Encoding")) != NULL) {
1039  1054            if (mg_strcasecmp(*te, mg_str("chunked")) == 0) {
1040  1055              is_chunked = true;
1041  1056            } else {
```

src/json.c

```
@@ -60,15 +60,28 @@ static double mg_atod(const char *p, int len, int
*numlen) {
60  60
61  61      // Exponential
62  62      if (i < len && (p[i] == 'e' || p[i] == 'E')) {
63        -     int j, exp = 0, minus = 0;
      63  +     int exp = 0, minus = 0;
64  64        i++;
65  65        if (i < len && p[i] == '-') minus = 1, i++;
66  66        if (i < len && p[i] == '+') i++;
67  67        while (i < len && p[i] >= '0' && p[i] <= '9' && exp < 308)
68  68          exp = exp * 10 + (p[i++] - '0');
69        -     if (minus) exp = -exp;
70        -     for (j = 0; j < exp; j++) d *= 10.0;
71        -     for (j = 0; j < -exp; j++) d /= 10.0;
      69  +     // use fast exponentiation
      70  +     // https://en.wikipedia.org/wiki/Exponentiation_by_squaring
      71  +     if (exp != 0) {
      72  +       double x = 10, y = 1;
      73  +       if (exp > 308) exp = 308;
      74  +       if (minus) x = 0.1;
      75  +       while (exp > 1) {
      76  +         if (exp & 1) {
      77  +           y *= x;
      78  +           --exp;
      79  +         }
      80  +         x *= x;
      81  +         exp >>= 1;
```

```
      82 +       }
      83 +       d *= x * y;
      84 +     }
72    85       }
73    86
74    87       if (numlen != NULL) *numlen = i;
```

src/net_builtin.c

```
@@ -551,9 +551,13 @@ static struct mg_connection *getpeer(struct mg_mgr
*mgr, struct pkt *pkt,
551   551           !(c->loc.is_ip6 ^ (pkt->ip6 != NULL)))  // IP or IPv6 to same dest
552   552         break;
553   553       if (!c->is_udp && pkt->tcp && c->loc.port == pkt->tcp->dport &&
554         -         !(c->loc.is_ip6 ^ (pkt->ip6 != NULL)) &&
555         -         lsn == (bool) c->is_listening &&
556         -         (lsn || c->rem.port == pkt->tcp->sport))
      554 +         ((lsn && c->is_listening && !(c->loc.is_ip6 ^ (pkt->ip6 != NULL))) ||
      555 +          (!lsn && !c->is_listening && c->rem.port == pkt->tcp->sport &&
      556 +           ((!c->loc.is_ip6 && c->rem.addr.ip4 == pkt->ip->src)
      557 + #if MG_ENABLE_IPV6
      558 +            || (c->loc.is_ip6 && MG_IP6MATCH(c->rem.addr.ip6, pkt->ip6->src))
      559 + #endif
      560 +                ))) // validate addr for established (not listening) conns
557   561         break;
558   562     }
559   563     return c;
@@ -1497,14 +1501,15 @@ static void backlog_poll(struct mg_mgr *mgr) {
1497  1501   }
1498  1502
1499  1503   // process options (MSS)
1500        - static void handle_opt(struct connstate *s, struct tcp *tcp, bool ip6) {
      1504 + static bool handle_opt(struct connstate *s, struct tcp *tcp, bool ip6) {
1501  1505     uint8_t *opts = (uint8_t *) (tcp + 1);
1502  1506     int len = 4 * ((int) (tcp->off >> 4) - ((int) sizeof(*tcp) / 4));
1503  1507     s->dmss = ip6 ? 1220 : 536;  // assume default, RFC-9293 3.7.1
1504  1508     while (len > 0) {            // RFC-9293 3.1 3.2
1505  1509       uint8_t kind = opts[0], optlen = 1;
1506  1510       if (kind != 1) {         // No-Operation
```

```
1507  1511          if (kind == 0) break;  // End of Option List
      1512  +       if (len < 2 || opts[1] == 0) return false; // Malformed options
1508  1513          optlen = opts[1];
1509  1514          if (kind == 2 && optlen == 4)  // set received MSS
1510  1515            s->dmss = (uint16_t) (((uint16_t) opts[2] << 8) + opts[3]);
      @@ -1513,6 +1518,7 @@ static void handle_opt(struct connstate *s, struct
      tcp *tcp, bool ip6) {
1513  1518        opts += optlen;
1514  1519        len -= optlen;
1515  1520      }
      1521  +   return true;
1516  1522    }
1517  1523
1518  1524    static void rx_tcp(struct mg_tcpip_if *ifp, struct pkt *pkt) {
      @@ -1527,7 +1533,7 @@ static void rx_tcp(struct mg_tcpip_if *ifp, struct
      pkt *pkt) {
1527  1533      // - check clients (Group 1) and established connections (Group 3)
1528  1534      if (c != NULL && c->is_connecting && pkt->tcp->flags == (TH_SYN | TH_ACK))
      {
1529  1535        // client got a server connection accept
1530        -     handle_opt(s, pkt->tcp, pkt->ip6 != NULL);  // process options (MSS)
      1536  +     if (!handle_opt(s, pkt->tcp, pkt->ip6 != NULL)) return;  // process
      options (MSS)
1531  1537        s->seq = mg_ntohl(pkt->tcp->ack), s->ack = mg_ntohl(pkt->tcp->seq) + 1;
1532  1538        tx_tcp_ctrlresp(ifp, pkt, TH_ACK, pkt->tcp->ack);
1533  1539        c->is_connecting = 0;  // Client connected
      @@ -1538,8 +1544,9 @@ static void rx_tcp(struct mg_tcpip_if *ifp, struct
      pkt *pkt) {
1538  1544      } else if (c != NULL && c->is_connecting && pkt->tcp->flags != TH_ACK) {
1539  1545        mg_error(c, "connection refused");
1540  1546      } else if (c != NULL && pkt->tcp->flags & TH_RST) {
1541        -     // TODO(): validate RST is within window (and optional with proper ACK)
1542        -     mg_error(c, "peer RST");  // RFC-1122 4.2.2.13
      1547  +     uint32_t seqno = mg_ntohl(pkt->tcp->seq);
      1548  +     if (seqno >= s->ack && seqno < (s->ack + MG_TCPIP_WIN))  // RFC-9293
      3.5.3
      1549  +       mg_error(c, "peer RST");  // RFC-1122 4.2.2.13
1543  1550      } else if (c != NULL) {
1544  1551        // process segment
1545  1552        s->tmiss = 0;                         // Reset missed keep-alive counter
```

```
@@ -1561,7 +1568,7 @@ static void rx_tcp(struct mg_tcpip_if *ifp, struct pkt *pkt) {
1561 1568         int key;
1562 1569         uint32_t isn;
1563 1570         if (pkt->tcp->sport != 0) {
1564      -         handle_opt(&cs, pkt->tcp, pkt->ip6 != NULL);  // process options (MSS)
     1571 +         if (!handle_opt(&cs, pkt->tcp, pkt->ip6 != NULL)) return;  // process options (MSS)
1565 1572           key = backlog_insert(c, pkt->tcp->sport,
1566 1573                                cs.dmss);  // backlog options (MSS)
1567 1574           if (key < 0) return;  // no room in backlog, discard SYN, client retries
```

src/str.c

```
@@ -87,7 +87,9 @@ bool mg_match(struct mg_str s, struct mg_str p, struct mg_str *caps) {
87  87       } else if (i < p.len && (p.buf[i] == '*' || p.buf[i] == '#')) {
88  88         if (caps && !caps->buf) caps->len = 0, caps->buf = &s.buf[j];  // Init cap
89  89         ni = i++, nj = j + 1;
90      -     } else if (nj > 0 && nj <= s.len && ((ni < p.len && p.buf[ni] == '#') || s.buf[j] != '/')) {
    90 +     } else if (nj > 0 && nj <= s.len &&
    91 +                ((ni < p.len && p.buf[ni] == '#') ||
    92 +                 (j < s.len && s.buf[j] != '/'))) {
91  93         i = ni, j = nj;
92  94         if (caps && caps->buf == NULL && caps->len == 0) {
93  95           caps--, caps->len = 0;  // Restart previous cap
@@ -168,8 +170,7 @@ bool mg_str_to_num(struct mg_str str, int base, void *val, size_t val_len) {
168 170           i++, ndigits++;
169 171         }
170 172         break;
171      -     default:
172      -       return false;
    173 +     default: return false;
173 174     }
174 175     if (ndigits == 0) return false;
```

```
175  176      if (i != str.len) return false;
```

src/tls_aes128.c

Load Diff

Large diffs are not rendered by default.

src/tls_aes128.h

```
     @@ -59,7 +59,9 @@ int mg_aes_gcm_encrypt(unsigned char *output, const
     unsigned char *input,
59   59   int mg_aes_gcm_decrypt(unsigned char *output, const unsigned char *input,
60   60                          size_t input_length, const unsigned char *key,
61   61                          const size_t key_len, const unsigned char *iv,
62        -                      const size_t iv_len);
     62   +                      const size_t iv_len, unsigned char *aead,
     63   +                      size_t aead_len, const unsigned char *tag,
     64   +                      const size_t tag_len);
63   65
64   66   #endif /* TLS_AES128_H */
65   67
```

src/tls_builtin.c

```
     @@ -555,23 +555,38 @@ static int mg_tls_recv_record(struct mg_connection
     *c) {
555  555      nonce[11] ^= (uint8_t) ((seq) & 255U);
556  556    #if MG_ENABLE_CHACHA20
557  557      {
     558  +   uint8_t associated_data[5] = {MG_TLS_APP_DATA, 0x03, 0x03,
     559  +                                 (uint8_t) ((msgsz >> 8) & 0xff),
     560  +                                 (uint8_t) (msgsz & 0xff)};
558  561      uint8_t *dec = (uint8_t *) mg_calloc(1, msgsz);
559  562      size_t n;
560  563      if (dec == NULL) {
561  564        mg_error(c, "TLS OOM");
```

```
562  565          return -1;
563  566        }
564       -      n = mg_chacha20_poly1305_decrypt(dec, key, nonce, msg, msgsz);
     567  +      n = mg_chacha20_poly1305_decrypt(dec, key, nonce, associated_data,
     568  +                                       sizeof(associated_data), msg, msgsz);
565  569        if (n == (size_t) -1) {
     570  +       mg_free(dec);
566  571          mg_error(c, "decryption error");
567  572          return -1;
568  573        }
569  574        memmove(msg, dec, n);
570  575        mg_free(dec);
571  576      }
572  577    #else
573       -    mg_gcm_initialize();
574       -    mg_aes_gcm_decrypt(msg, msg, msgsz - 16, key, 16, nonce, sizeof(nonce));
     578  +    {
     579  +      uint8_t associated_data[5] = {MG_TLS_APP_DATA, 0x03, 0x03,
     580  +                                    (uint8_t) ((msgsz >> 8) & 0xff),
     581  +                                    (uint8_t) (msgsz & 0xff)};
     582  +      mg_gcm_initialize();
     583  +      if (mg_aes_gcm_decrypt(msg, msg, msgsz - 16, key, 16, nonce,
          sizeof(nonce),
     584  +                             associated_data, sizeof(associated_data),
     585  +                             msg + msgsz - 16, 16) != 0) {
     586  +        mg_error(c, "GCM tag verify failed");
     587  +        return -1;
     588  +      }
     589  +    }
575  590    #endif
576  591
577  592      r = msgsz - 16 - 1;
     @@ -1497,7 +1512,7 @@ static int mg_tls_verify_cert_san(const uint8_t
     *der, size_t dersz,
1497 1512          if (mg_match(mg_str(server_name), mg_str_n((char *) name.value,
          name.len),
1498 1513                       NULL))
1499 1514            return 1;  // and matches the host name
1500      -      }  // TODO(): add IPv6 comparison, more items ?
     1515  +      }  // TODO(): add IPv6 comparison, more items ?
```

```
1501  1516      }
1502  1517      return -1;
1503  1518    }
      @@ -1525,7 +1540,7 @@ static int mg_tls_verify_cert_signature(const
      struct mg_tls_cert *cert,
1525  1540                                  mg_uecc_secp256r1());
1526  1541        } else if (issuer->pubkey.len == 96) {
1527  1542          MG_VERBOSE(("ignore secp386 for now"));
1528        -       return 1;
      1543  +       return 0;
1529  1544        } else {
1530  1545          MG_ERROR(("unsupported public key length: %d", issuer->pubkey.len));
1531  1546          return 0;
      @@ -1642,6 +1657,14 @@ static int mg_tls_recv_cert(struct mg_connection
      *c, bool is_client) {
1642  1657            mg_error(c, "failed to verify hostname");
1643  1658            return -1;
1644  1659          }
      1660  +       if (ci->pubkey.len > sizeof(tls->pubkey)) {
      1661  +         mg_error(c, "invalid certificate length");
      1662  +         return -1;
      1663  +       }
      1664  +       if (ci->pubkey.len > sizeof(tls->pubkey)) {
      1665  +         mg_error(c, "peer public key too large");
      1666  +         return -1;
      1667  +       }
1645  1668          memmove(tls->pubkey, ci->pubkey.buf, ci->pubkey.len);
1646  1669          tls->pubkeysz = ci->pubkey.len;
1647  1670        } else {
      @@ -1989,7 +2012,7 @@ static int mg_rsa_parse_der_int(const uint8_t **p,
      const uint8_t *end,
1989  2012    *p = value_end;
1990  2013
1991  2014    MG_VERBOSE(("DER INT: parsed %u bytes (skipped zero=%d)", len,
1992        -             (size_t)(value_end - value_start) != (size_t) len ? 1 : 0));
      2015  +             (size_t)(value_end - value_start) != (size_t) len ? 1 : 0));
1993  2016    return 0;
1994  2017  }
1995  2018
```

```
@@ -2499,7 +2522,7 @@ long mg_tls_send(struct mg_connection *c, const
void *buf, size_t len) {
2499 2522       if (!mg_tls_encrypt(c, (const uint8_t *) buf, len, MG_TLS_APP_DATA))
2500 2523         return 0;  // returning 0 means an OOM condition (iobuf couldn't
resize),
2501 2524                    // yet this is so far recoverable, let the caller decide
2502      -   }              // else, resend outstanding encrypted data in tls->send
     2525 +   }  // else, resend outstanding encrypted data in tls->send
2503 2526     while (tls->send.len > 0 &&
2504 2527            (n = mg_io_send(c, tls->send.buf, tls->send.len)) > 0) {
2505 2528       mg_iobuf_del(&tls->send, 0, (size_t) n);
```

src/tls_chacha20.c

```
@@ -82,7 +82,8 @@ static PORTABLE_8439_DECL void
poly1305_finish(poly1305_context *ctx,
82 82       defined(__AVR__)
83 83   #define __HAVE_LITTLE_ENDIAN 1
84 84   #endif
85      - // DO NOT test for LITTLE_ENDIAN, as it is defined as 1234 when including
sys/types.h in GCC
     85 + // DO NOT test for LITTLE_ENDIAN, as it is defined as 1234 when including
     86 + // sys/types.h in GCC
86 87
87 88   #ifndef TEST_SLOW_PATH
88 89   #if defined(__HAVE_LITTLE_ENDIAN)
@@ -178,7 +179,7 @@ static void core_block(const uint32_t *restrict
start,
178 179     TIMES16(__FIN)
179 180   }
180 181
181      - #define U8(x) ((uint8_t) ((x) &0xFF))
     182 + #define U8(x) ((uint8_t) ((x) & 0xFF))
182 183
183 184   #ifdef FAST_PATH
184 185   #define xor32_le(dst, src, pad)            \
@@ -503,7 +504,7 @@ static unsigned short U8TO16(const unsigned char *p)
{
503 504   /* store a 16 bit unsigned integer as two 8 bit unsigned integers in little
504 505    * endian */
```

```
505  506    static void U16TO8(unsigned char *p, unsigned short v) {
506       -    p[0] = (v) &0xff;
     507  +    p[0] = (v) & 0xff;
507  508      p[1] = (v >> 8) & 0xff;
508  509    }
509  510
          @@ -514,7 +515,7 @@ static void poly1305_init(poly1305_context *ctx,
          const unsigned char key[32]) {
514  515
515  516      /* r &= 0xffffffc0ffffffc0ffffffc0fffffff */
516  517      t0 = U8TO16(&key[0]);
517       -    st->r[0] = (t0) &0x1fff;
     518  +    st->r[0] = (t0) & 0x1fff;
518  519      t1 = U8TO16(&key[2]);
519  520      st->r[1] = ((t0 >> 13) | (t1 << 3)) & 0x1fff;
520  521      t2 = U8TO16(&key[4]);
          @@ -554,7 +555,7 @@ static void poly1305_blocks(poly1305_state_internal_t
          *st,
554  555
555  556        /* h += m[i] */
556  557        t0 = U8TO16(&m[0]);
557       -      st->h[0] += (t0) &0x1fff;
     558  +      st->h[0] += (t0) & 0x1fff;
558  559        t1 = U8TO16(&m[2]);
559  560        st->h[1] += ((t0 >> 13) | (t1 << 3)) & 0x1fff;
560  561        t2 = U8TO16(&m[4]);
          @@ -716,7 +717,7 @@ static unsigned long U8TO32(const unsigned char *p) {
716  717    /* store a 32 bit unsigned integer as four 8 bit unsigned integers in little
717  718     * endian */
718  719    static void U32TO8(unsigned char *p, unsigned long v) {
719       -    p[0] = (unsigned char) ((v) &0xff);
     720  +    p[0] = (unsigned char) ((v) & 0xff);
720  721      p[1] = (unsigned char) ((v >> 8) & 0xff);
721  722      p[2] = (unsigned char) ((v >> 16) & 0xff);
722  723      p[3] = (unsigned char) ((v >> 24) & 0xff);
          @@ -980,8 +981,8 @@ typedef unsigned uint128_t __attribute__((mode(TI)));
980  981    #define MUL128(out, x, y) out = ((uint128_t) x * y)
981  982    #define ADD(out, in) out += in
```

```
 982   983    #define ADDLO(out, in) out += in
 983         - #define SHR(in, shift) (uint64_t)(in >> (shift))
 984         - #define LO(in) (uint64_t)(in)
       984   + #define SHR(in, shift) (uint64_t) (in >> (shift))
       985   + #define LO(in) (uint64_t) (in)
 985   986
 986   987    #define POLY1305_NOINLINE __attribute__((noinline))
 987   988    #endif
       ↓↑      @@ -1010,7 +1011,7 @@ static uint64_t U8TO64(const unsigned char *p) {
1010  1011    /* store a 64 bit unsigned integer as eight 8 bit unsigned integers in little
1011  1012     * endian */
1012  1013    static void U64TO8(unsigned char *p, uint64_t v) {
1013        -   p[0] = (unsigned char) ((v) &0xff);
      1014  +   p[0] = (unsigned char) ((v) & 0xff);
1014  1015      p[1] = (unsigned char) ((v >> 8) & 0xff);
1015  1016      p[2] = (unsigned char) ((v >> 16) & 0xff);
1016  1017      p[3] = (unsigned char) ((v >> 24) & 0xff);
       ↕      @@ -1028,7 +1029,7 @@ static void poly1305_init(poly1305_context *ctx,
              const unsigned char key[32]) {
1028  1029      t0 = U8TO64(&key[0]);
1029  1030      t1 = U8TO64(&key[8]);
1030  1031
1031        -   st->r[0] = (t0) &0xffc0fffffff;
      1032  +   st->r[0] = (t0) & 0xffc0fffffff;
1032  1033      st->r[1] = ((t0 >> 44) | (t1 << 20)) & 0xfffffc0ffff;
1033  1034      st->r[2] = ((t1 >> 24)) & 0x00fffffc0f;
1034  1035
       ↓↑      @@ -1072,7 +1073,7 @@ static void
              poly1305_blocks(poly1305_state_internal_t *st,
1072  1073        t0 = U8TO64(&m[0]);
1073  1074        t1 = U8TO64(&m[8]);
1074  1075
1075        -     h0 += ((t0) &0xfffffffffff);
      1076  +     h0 += ((t0) & 0xfffffffffff);
1076  1077        h1 += (((t0 >> 44) | (t1 << 20)) & 0xfffffffffff);
1077  1078        h2 += (((t1 >> 24)) & 0x3ffffffffff) | hibit;
1078  1079
       ↓↑      @@ -1179,7 +1180,7 @@ static POLY1305_NOINLINE void
              poly1305_finish(poly1305_context *ctx,
```

```
1179  1180      t0 = st->pad[0];
1180  1181      t1 = st->pad[1];
1181  1182
1182        -   h0 += ((t0) &0xfffffffffff);
      1183  +   h0 += ((t0) & 0xfffffffffff);
1183  1184      c = (h0 >> 44);
1184  1185      h0 &= 0xfffffffffff;
1185  1186      h1 += (((t0 >> 44) | (t1 << 20)) & 0xfffffffffff) + c;
          @@ -1256,7 +1257,7 @@ static PORTABLE_8439_DECL void
          pad_if_needed(poly1305_context *ctx,
1256  1257    }
1257  1258  }
1258  1259
1259        - #define __u8(v) ((uint8_t) ((v) &0xFF))
      1260  + #define __u8(v) ((uint8_t) ((v) & 0xFF))
1260  1261
1261  1262  // TODO: make this depending on the unaligned/native read size possible
1262  1263  static PORTABLE_8439_DECL void write_64bit_int(poly1305_context *ctx,
          @@ -1329,14 +1330,26 @@ PORTABLE_8439_DECL size_t
          mg_chacha20_poly1305_encrypt(
1329  1330
1330  1331  PORTABLE_8439_DECL size_t mg_chacha20_poly1305_decrypt(
1331  1332      uint8_t *restrict plain_text, const uint8_t key[RFC_8439_KEY_SIZE],
1332        -     const uint8_t nonce[RFC_8439_NONCE_SIZE],
1333        -     const uint8_t *restrict cipher_text, size_t cipher_text_size) {
      1333  +     const uint8_t nonce[RFC_8439_NONCE_SIZE], const uint8_t *restrict ad,
      1334  +     size_t ad_size, const uint8_t *restrict cipher_text,
      1335  +     size_t cipher_text_size) {
1334  1336    // first we calculate the mac and see if it lines up, only then do we
          decrypt
1335  1337    size_t actual_size = cipher_text_size - RFC_8439_TAG_SIZE;
      1338  +   uint8_t computed_mac[RFC_8439_TAG_SIZE];
      1339  +   int diff = 0;
      1340  +   size_t i;
1336  1341    if (MG_OVERLAPPING(plain_text, actual_size, cipher_text, cipher_text_size))
          {
1337  1342      return (size_t) -1;
1338  1343    }
1339  1344
```

```
+    poly1305_calculate_mac(computed_mac, cipher_text, actual_size, key, nonce,
     ad,
+                           ad_size);
+
+    // compare tags
+    for (i = 0; i < RFC_8439_TAG_SIZE; i++)
+      diff |= computed_mac[i] ^ cipher_text[actual_size + i];
+    if (diff != 0) return (size_t) -1;
+
     chacha20_xor_stream(plain_text, cipher_text, actual_size, key, nonce, 1);
     return actual_size;
   }
```

src/tls_chacha20.h

```
@@ -105,6 +105,7 @@ PORTABLE_8439_DECL size_t mg_chacha20_poly1305_encrypt(
 PORTABLE_8439_DECL size_t mg_chacha20_poly1305_decrypt(
     uint8_t *restrict plain_text, const uint8_t key[RFC_8439_KEY_SIZE],
     const uint8_t nonce[RFC_8439_NONCE_SIZE],
+    const uint8_t *restrict ad, size_t ad_size,
     const uint8_t *restrict cipher_text, size_t cipher_text_size);
 #if defined(__cplusplus)
 }
```

test/unit_test.c

```
@@ -72,6 +72,14 @@ static void test_match(void) {
   ASSERT(mg_match(mg_str_n("caabxa", 6), mg_str_n("#aa#bb#", 7), NULL) == 0);
   ASSERT(mg_match(mg_str_n("a__b_c", 6), mg_str_n("a*b*c", 5), NULL) == 1);

+  {
+    char *ae_value = malloc(1);  // Exact size for ASAN
+    ae_value[0] = 'g';
+    struct mg_str input = {ae_value, 1};
+    mg_match(input, mg_str("*gzip*"), NULL);  // Check for OOB
+    free(ae_value);
+  }
+
   {
```

```
 76   84          struct mg_str caps[3];
 77   85          ASSERT(mg_match(mg_str("//a.c"), mg_str("#.c"), NULL) == true);
@@ -1796,15 +1804,15 @@ static void test_http_parse(void) {
1796 1804        }
1797 1805
1798 1806        // #2292: fail on stray \r inside the headers
1799      -      ASSERT(mg_http_parse("a є\n\n", 6, &req) == 6);
1800      -      ASSERT(mg_http_parse("a b\n\n", 5, &req) == 5);
1801      -      ASSERT(mg_http_parse("a b\na:\n\n", 8, &req) > 0);
1802      -      ASSERT(mg_http_parse("a b\na:\r\n\n", 9, &req) > 0);
1803      -      ASSERT(mg_http_parse("a b\n\ra:\r\n\n", 10, &req) == -1);
1804      -      ASSERT(mg_http_parse("a b\na:\r1\n\n", 10, &req) == -1);
1805      -      ASSERT(mg_http_parse("a b\na: \r1\n\n", 11, &req) == -1);
1806      -      ASSERT(mg_http_parse("a b\na: \rb:\n\n", 12, &req) == -1);
1807      -      ASSERT(mg_http_parse("a b\na: \nb:\n\n", 12, &req) > 0);
     1807 +      ASSERT(mg_http_parse("a є HTTP/1.0\n\n", 15, &req) == 15);
     1808 +      ASSERT(mg_http_parse("a b HTTP/1.0\n\n", 14, &req) == 14);
     1809 +      ASSERT(mg_http_parse("a b HTTP/1.0\na:\n\n", 17, &req) > 0);
     1810 +      ASSERT(mg_http_parse("a b HTTP/1.0\na:\r\n\n", 18, &req) > 0);
     1811 +      ASSERT(mg_http_parse("a b HTTP/1.0\n\ra:\r\n\n", 19, &req) == -1);
     1812 +      ASSERT(mg_http_parse("a b HTTP/1.0\na:\r1\n\n", 19, &req) == -1);
     1813 +      ASSERT(mg_http_parse("a b HTTP/1.0\na: \r1\n\n", 20, &req) == -1);
     1814 +      ASSERT(mg_http_parse("a b HTTP/1.0\na: \rb:\n\n", 21, &req) == -1);
     1815 +      ASSERT(mg_http_parse("a b HTTP/1.0\na: \nb:\n\n", 21, &req) > 0);
1808 1816
1809 1817        {
1810 1818          const char *s = "ґєт /слеш HTTP/1.0\nмісто:  київ \n\n";
@@ -1891,6 +1899,24 @@ static void test_http_parse(void) {
1891 1899          ASSERT((v = mg_http_get_header(&req, "h")) == NULL);  //
      MG_MAX_HTTP_HEADERS
1892 1900        }
1893 1901
     1902 +      {
     1903 +        // no HTTP version
     1904 +        static char *s = "a b\na:1\nb:2\nc:2\n\n";
     1905 +        ASSERT(mg_http_parse(s, strlen(s), &req) == -1);
     1906 +
     1907 +        // HTTP version
```

```
        1908 +     s = "a b HTTP/1.0\na:1\nb:2\nc:2\n\n";
        1909 +     ASSERT(mg_http_parse(s, strlen(s), &req) == strlen(s));
        1910 +
        1911 +     // Content-Length
        1912 +     s = "a b HTTP/1.0\nContent-Length:10\nb:2\nc:2\n\n";
        1913 +     ASSERT(mg_http_parse(s, strlen(s), &req) == strlen(s));
        1914 +
        1915 +     // duplicated Content-Length
        1916 +     s = "a b HTTP/1.0\nContent-Length:10\nb:2\nContent-Length:20\n\n";
        1917 +     ASSERT(mg_http_parse(s, strlen(s), &req) == -1);
        1918 +   }
        1919 +
1894    1920       {
1895    1921         struct mg_connection c;
1896    1922         struct mg_str s,
             @@ -1916,15 +1942,15 @@ static void test_http_parse(void) {
1916    1942         struct mg_http_message hm;
1917    1943         const char *s = "a b HTTP/1.0\n\n";
1918    1944         ASSERT(mg_http_parse(s, strlen(s), &hm) == (int) strlen(s));
1919        -     s = "a b\nc:d\n\n";
        1945 +     s = "a b HTTP/1.0\nc:d\n\n";
1920    1946         ASSERT(mg_http_parse(s, strlen(s), &hm) == (int) strlen(s));
1921    1947         s = "a\nb:b\nc:c\n\n";
1922    1948         ASSERT(mg_http_parse(s, strlen(s), &hm) < 0);
1923        -     s = "a b\nc: \xc0\n\n";  // Invalid UTF in the header value: accept
        1949 +     s = "a b HTTP/1.0\nc: \xc0\n\n";  // Invalid UTF in the header value:
             accept
1924    1950         ASSERT(mg_http_parse(s, strlen(s), &hm) == (int) strlen(s));
1925    1951         ASSERT((v = mg_http_get_header(&hm, "c")) != NULL);
1926    1952         ASSERT(vcmp(*v, "\xc0"));
1927        -     s = "a b\n\xc0: 2\n\n";  // Invalid UTF in the header name: do NOT accept
        1953 +     s = "a b HTTP/1.0\n\xc0: 2\n\n";  // Invalid UTF in the header name: do
             NOT accept
1928    1954         ASSERT(mg_http_parse(s, strlen(s), &hm) == -1);
1929    1955       }
1930    1956
             @@ -3625,6 +3651,40 @@ static void test_json(void) {
3625    3651         val = mg_json_get_tok(json, "$.a");
3626    3652         ASSERT(mg_strcmp(val, expected) == 0);
```

```
3627  3653      }
      3654  +
      3655  +   // mg_json_get_num: parsing exponential
      3656  +   {
      3657  +     double d = 0.0;
      3658  +     json = mg_str(
      3659  +       "{"
      3660  +       "\"i\":1e3,"
      3661  +       "\"n\":-2.5e2,"
      3662  +       "\"p\":3E+4,"
      3663  +       "\"m\":4.25E-1,"
      3664  +       "\"z\":0e0,"
      3665  +       "\"s\":-0.0e+0,"
      3666  +       "\"a\":[6.123456789e3,-1e-9],"
      3667  +       "\"bad\":\"1e3\""
      3668  +       "}");
      3669  +     double tolerance = 1e-12;
      3670  +     ASSERT(mg_json_get_num(json, "$.i", &d) == true);
      3671  +     ASSERT(fabs(d - 1000.0) < tolerance);
      3672  +     ASSERT(mg_json_get_num(json, "$.n", &d) == true);
      3673  +     ASSERT(fabs(d - -250.0) < tolerance);
      3674  +     ASSERT(mg_json_get_num(json, "$.p", &d) == true);
      3675  +     ASSERT(fabs(d - 30000.0) < tolerance);
      3676  +     ASSERT(mg_json_get_num(json, "$.m", &d) == true);
      3677  +     ASSERT(fabs(d - 0.425) < tolerance);
      3678  +     ASSERT(mg_json_get_num(json, "$.z", &d) == true);
      3679  +     ASSERT(fabs(d - 0.0) < tolerance);
      3680  +     ASSERT(mg_json_get_num(json, "$.s", &d) == true);
      3681  +     ASSERT(fabs(d - 0.0) < tolerance);
      3682  +     ASSERT(mg_json_get_num(json, "$.a[0]", &d) == true);
      3683  +     ASSERT(fabs(d - 6123.456789) < tolerance);
      3684  +     ASSERT(mg_json_get_num(json, "$.a[1]", &d) == true);
      3685  +     ASSERT(fabs(d - -1e-9) < tolerance);
      3686  +     ASSERT(mg_json_get_num(json, "$.bad", &d) == false);
      3687  +   }
3628  3688    }
3629  3689
3630  3690    static void resp_rpc(struct mg_rpc_req *r) {
```

Comments 0

Please sign in to comment.