Dnsmasq Bug Report

Overview

Software: dnsmasq (https://thekelleys.org.uk/dnsmasq/doc.html)
Affected Version: TBD
Vulnerabilities: N/A
Estimated Risk: Low
CVE: N/A
Researchers: Max Julian Hofmann

Summary

The Advanced Research Team of CrowdStrike Intelligence discovered multiple memory safety bugs in dnsmasq. However, in the current code base, these bugs are not externally reachable but only by directly passing certain command line arguments or configuration options to dnsmasq.

Technical Details

#1: Heap-Overwrite via add_rev6()

The option **`--rev-server=<addr>/<prefix>,<ipaddr>`** triggers a bug for certain arguments due to a missing size check on the given prefix for IPv6 addresses.

As shown in the following source code excerpt from **`option.c`**, dnsmasq invokes **`add_rev6()`** if the command line option **`--rev-server`** was specified in combination with an IPv6 address (identified by using **`inet_pton()`**). The prefix is separated and converted from the passed argument using **`split_chr()`** and **`atoi_check()`**, and is then passed to **`add_rev6()`** via the second argument as an integer.

```
static int one_opt(int option, char *arg, char *errstr, char *gen_err, int
command_line, int servers_only)
{
[...]
      switch (option)
      {
            case LOPT_REV_SERV: /* --rev-server */
                  {
[...]
                  int size;
[...]
```

```
                if (!(string = split_chr(arg, '/')) || !atoi_check(string,
            &size))
                      ret_err(gen_err);
[...]
                else if (inet_pton(AF_INET6, arg, &addr6)
                {
                      serv = add_rev6(&addr6, size);
[...]
}
```

The function **atoi_check()** (implemented in **option.c:678**) does not check the range of the given number but only checks whether its argument contains numeric symbols only.

As can be seen in the following excerpt from **option.c**, the function **add_rev6()** receives this prefix via the argument **msize** where it is used to format the given IPv6 address as a string.

```
static struct server *add_rev6(struct in6_addr *addr, int msize)
{
[...]
      char *p = serv->domain = opt_malloc(73);
[...]
  for (i = msize-1; i >= 0; i -= 4)
    {
      int dig = ((unsigned char *)addr)[i>>3];
      p += sprintf(p, "%.1x.", (i>>2) & 1 ? dig & 15 : dig >> 4);
    }
[...]
}
```

For large values of the prefix, the repeated call to **sprintf()** will overflow the heap-allocated buffer pointed to by **p**.

Proof-of-Concept

```
$ dnsmasq --rev-server=1::2/1234,3::4 --test
malloc(): invalid size (unsorted)
[1]    110013 abort (core dumped)  dnsmasq --rev-server=1::2/1234,3::4 --test
```

#2: free() of non-malloced address via multiple options

The function **canonicalise_opt()** returns an empty static string if the passed argument - a string to be stored in a heap-allocated memory - is empty. This empty static string will be passed to **free()** later if the memory is no longer used. There are several ways to trigger this bug:

- Using **--srv-host=<name>,<target>,<port>,<priority>,<weight>** with an invalid port and empty name or an empty target. Proof-Of-Concept:
  - **$ dnsmasq --srv-host=,a,b --test** (empty name is about to be freed)
  - **$ dnsmasq --srv-host=n,,23,42,w --test** (empty target is about to be freed)
- Using **--cname=<alias>,<target>[,<ttl>]** with an empty alias and empty target and a trailing comma. Example: **$ dnsmasq --cname=,,23, --test**. This will trigger the logic to parse the number not as a TTL but as a target, and the trailing comma makes the function read in another entry to overwrite the current values. Before they are overwritten the function is trying to free them.
- Using **--ptr-record=<name>,<target>** with empty name and malformed target. Example: **$ dnsmasq --ptr-record=,$(echo -n "\xff") --test**
- Using **--naptr-record=<name>,<naptr>** with empty name. Example: **$ dnsmasq --naptr-record=",,x,,," --test**

#3: free() of arbitrary address via dhcp-match option

As shown in the following source code excerpt from **option.c** the function **parse_dhcp_opt()** is invoked if **--dhcp-match=set:<tag>,<optspec>** is specified.

```
static int one_opt(int option, char *arg, char *errstr, char *gen_err, int
command_line, int servers_only)
{
[...]
      switch (option)
      {
            case 'O':           /* --dhcp-option */
            case LOPT_FORCE:    /* --dhcp-option-force */
            case LOPT_OPTS:
            case LOPT_MATCH:    /* --dhcp-match */
                  return parse_dhcp_opt(errstr, arg,
                        option == LOPT_FORCE ? DHOPT_FORCE :
                        (option == LOPT_MATCH ? DHOPT_MATCH :
                        (option == LOPT_OPTS ? DHOPT_BANK : 0)));
[...]
      }
[...]
}
```

The function **parse_dhcp_opt()** creates a **dhcp_opt** structure. The following source code excerpt from **dnsmasq.h** shows the definition of this structure:

```
struct dhcp_opt {
  int opt, len, flags;
```

```
  union {
    int encap;
    unsigned int wildcard_mask;
    unsigned char *vendor_class;
  } u;
[...]
};
```

If **--dhcp-match=vendor:<tag>,<...>** is specified, the **flags** field of the structure is set to store **DHOPT_VENDOR** to indicate that a vendor string was passed, as the following source code excerpt from **option.c** shows:

```
static int parse_dhcp_opt(char *errstr, char *arg, int flags)
{
    struct dhcp_opt *new = opt_malloc(sizeof(struct dhcp_opt));
[...]
    while (arg) {
        comma = split(arg);
[...]
        else if (strstr(arg, "vendor:") == arg) {
            new->u.vendor_class = (unsigned char *)opt_string_alloc(arg+7);
            new->flags |= DHOPT_VENDOR;
      }
[...]
}
```

Additionally, the function will allocate heap memory to store the **tag** specified as vendor and assign the result to **new->u.vendor_class**. According to **new->flags** this memory will be freed by this function later on, as it indicates that the memory was allocated on the heap.

We have observed that two commas will let the function **parse_dhcp_opt()** break out of the loop to consume **set:<tag>** options. Example:

--dhcp-match=vendor:crowdstrike,,<remainder>.

The remainder is then processed later in **parse_dhcp_opt()**:

```
static int parse_dhcp_opt(char *errstr, char *arg, int flags)
[...]
    if (comma)
    {
        /* characterise the value */
        char c;
```

```
        int found_dig = 0, found_colon = 0;
        is_addr = is_addr6 = is_hex = is_dec = is_string = 1;
        addrs = digs = 1;
        dots = 0;
        for (cp = comma; (c = *cp); cp++) {
            if (c == ',') {
                addrs++;
                is_dec = is_hex = 0;
            } else if (c == ':') {
                digs++;
                is_dec = is_addr = 0;
                found_colon = 1;
            }
        }
[...]
        if (found_dig && (opt_len & OT_TIME) && strlen(comma) > 0) {
[...]
        } else if (is_hex && digs > 1) {
            new->len = digs;
            new->val = opt_malloc(new->len);
            parse_hex(comma, new->val, digs, (flags & DHOPT_MATCH) ?
&new->u.wildcard_mask : NULL, NULL);
            new->flags |= DHOPT_HEX;
         }
[...]
    return 1;
on_error:
  dhcp_opt_free(new);
  return 0;
}
```

As can be seen in the source code excerpt of **`parse_dhcp_opt()`**, the function consumes the remaining string by parsing it character-wise. For each '**`:`**' the counter **`digs`** gets incremented. If the trailing string consists of hexadecimal characters only (including '**`*`**') the function **`parse_hex()`** will be invoked.

The following source code excerpt of the **`parse_hex()`** function from util.c shows that a mask related to the parsing result is stored in **`new->u.wildcard_mask`**, which was passed as the pointer argument **`wildcard_mask`**.

```
int parse_hex(char *in, unsigned char *out, int maxlen, unsigned int
*wildcard_mask, int *mac_type)
{
```

```
  int done = 0, mask = 0, i = 0;
[...]
  while (!done && (maxlen == -1 || i < maxlen)) {
[...]
    if (r != in) {
      if (*r == '-' && i == 0 && mac_type) {
[...]
      } else {
        *r = 0;
        if (strcmp(in, "*") == 0) {
          mask = (mask << 1) | 1;
          i++;
        } else {
          int j, bytes = (1 + (r - in)) / 2;
          for (j = 0; j < bytes; j++) {
[...]
            mask = mask << 1;
[...]
          }
        }
      }
    }
    in = r + 1;
  }

  if (wildcard_mask)
    *wildcard_mask = mask;

  return i;
}
```

This mask stores the information (as a bitmap) at which position the argument contains a hexadecimal value or a '*****'. Therefore every bit of **`wildcard_mask`** can be controlled. As seen above at the definition of **`struct dhcp_opt`**, the fields **`wildcard_mask`** and **`vendor_class`** are part of a union. As result, we are able to control the address of **`vendor_class`**, which is expected to be located in the heap.

If an error occurs in the parsing process (which is easy to trigger by a malformed string as given above) the function **`dhcp_opt_free()`** is called. If **`DHOPT_VENDOR`** is set in the flags (which it is) then **`free(opt->u.vendor_class)`** is called.

Proof-of-Concept

```
$ valgrind dnsmasq
--'dhcp-match=vendor:art,,:*:aa:aa:aa:aa:aa:*:aa:*:aa:aa:aa:aa:*:aa:aa:*:aa:a
a:aa:aa:*:*:aa:*:aa:aa:aa:*:aa:aa' --test
[...]
==73092== Using Valgrind-3.17.0 and LibVEX; rerun with -h for copyright info
[...]
==73092==  Address 0x41424344 is not stack'd, malloc'd or (recently) free'd
[...]
```

#4: NULL-Dereference via dhcp-relay option

The option **--dhcp-relay=<local-addr>,<server>[,<iface>]** can be used to trigger a crash that occurs if **,<server>** is missing. The following source code excerpt from **option.c** shows how **one_opt()** handles the **dhcp-relay** option.

```
static int one_opt(int option, char *arg, char *errstr, char *gen_err, int
command_line, int servers_only)
{
[...]
      switch (option)
      {
            case LOPT_RELAY: /* --dhcp-relay */
            {
[...]
                  comma = split(arg);
[...]
                  if (inet_pton(AF_INET, arg, &new->local) &&
            inet_pton(AF_INET, comma, &new->server))
[...]
#ifdef HAVE_DHCP
                  else if (inet_pton(AF_INET6, arg, &new->local) &&
            inet_pton(AF_INET6, comma, &new->server))
[...]
}
```

The function **split()** is invoked and it will set the variable **comma** to store the **<local-addr>** part and will set the variable **arg** to point to the **<server>** part. In case **,<server>** is missing, **arg** will point to **NULL**. The function **inet_pton()** will **SIGSEGV** if **arg** is **NULL**.

Proof-of-Concept

```
$ dnsmasq --dhcp-relay=1.2.3.4 --test
```

```
[1]    115715 segmentation fault (core dumped)  dnsmasq --dhcp-relay=1.2.3.4
--test
$ dnsmasq --dhcp-relay=1234:: --test
[1]    115774 segmentation fault (core dumped)  dnsmasq --dhcp-relay=1234::
--test
```

#5 NULL-Dereference via rev-server option

The option **--rev-server=<addr>/<prefix>,<ipaddr>** can be used to trigger a crash if **,<ipaddr>** is missing. As can be seen in the following source code excerpt from **option.c** the function **one_opt()** splits the argument into the **<addr>/<prefix>** parts by using **split()** and stores **NULL** in **comma**, if **,<ipaddr>** is missing.

```
static int one_opt(int option, char *arg, char *errstr, char *gen_err, int
command_line, int servers_only)
{
[...]
      case LOPT_REV_SERV: /* --rev-server */
      {
[...]
            comma=split(arg);
[...]
            string = parse_server(comma, &serv->addr, &serv->source_addr,
      serv->interface, &serv->flags);
[...]
}
```

The function **parse_server()** will then pass **comma** as an argument to strcmp(), which leads to a NULL pointer dereference.

Proof-of-Concept

```
$ dnmasq --rev-server=10.11.12.13/24 --test
[1]    125764 segmentation fault (core dumped)  dnsmasq
```

#6 NULL-Dereference via auth-soa option

The argument **--auth-soa=<serial>[,...]** can be used to trigger a crash if an entry of the passed list (separated by commas) is empty.

The following source code excerpt from **option.c** shows the handling of the **auth-soa** argument.

```
static int one_opt(int option, char *arg, char *errstr, char *gen_err, int
command_line, int servers_only)
{
[...]
   case  LOPT_AUTHSOA: /* --auth-soa */
      comma = split(arg);
      daemon->soa_sn = (u32)atoi(arg);
      if (comma)
      {
        char *cp;
        arg = comma;
        comma = split(arg);
        daemon->hostmaster = opt_string_alloc(arg);
        for (cp = daemon->hostmaster; *cp; cp++)
          if (*cp == '@')
            *cp = '.';
[...]
}
```

In case a list like **auth-soa=<serial1>,,<serial2>** is specified, the variable **arg** points to an empty string. That empty string is then passed to opt_string_alloc(), which in this case returns NULL.The function **opt_string_alloc()**, as can be seen in the source code excerpt, will try to dereference this **NULL** stored in **cp**, where **cp** points to **daemon->hostmaster**.

Proof-of-Concept

```
$ dnsmasq --auth-soa=,,123 --test
[1]    128465 segmentation fault (core dumped)  dnsmasq --auth-soa=,,123
--test
```