

Remote Heap-Based Buffer Underflow

Vulnerability at BuptLab dns relay server

Written by : Antonius (w1sdom)
Web : www.bluedragonsec.com
Github : <https://github.com/bluedragonsecurity>
Vulnerability discovered by : Antonius
Date of Discovery : March 6 2026

On March 6 2026 I discovered a buffer underflow vulnerability at BuptLab dns relay server.

BuptLab dns relay server is a dns relay server developed by Agicy from Beijing University. Here is the repository of BuptLab dns relay server : https://github.com/agicy/buptLab-dns_relay_server

The vulnerability occurs when the dns relay server processed a 5 bytes of truncated dns packet. Here is the the packet that triggers the vulnerability at BuptLab dns relay server :

```
# xxd poc_dns_packet.bin
00000000: a77e 0014 a6
# wc -c poc_dns_packet.bin
5 poc_dns_packet.bin
```

The packet doesn't have a completed dns header, it's a truncated dns packet.

Source Code Analysis

The asan log indicates a write of 1 byte outside the boundary of buffer :

```
==22332==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x7b96a5d245ef at pc
0x55d5fd9b3604 bp 0x7ffcbf8a090 sp 0x7ffcbf8a088
```

```
WRITE of size 1 at 0x7b96a5d245ef thread T0
```

```
#0 0x55d5fd9b3603 in get_name_from_name_field src/network/dns_utility.c:41
```

```
#1 0x55d5fd9bd123 in logger_dns_message src/module/logger.c:153
```

```
#2 0x55d5fd9af98e in main src/dns_relay.c:263
```

```
#3 0x7f76a6a29f67 in __libc_start_call_main ../sysdeps/nptl/libc_start_call_main.h:58
```

```
#4 0x7f76a6a2a024 in __libc_start_main_impl ../csu/libc-start.c:360
```

```
#5 0x55d5fd9af3c0 in _start
```

```
(/home/robohax/Desktop/fuzz/dns/buptLab-dns_relay_server/dns_relay+0x1f3c0) (BuildId:
c5699f255c9870f6ad7559482666aea647197291)
```

More specifically it's a write of 1 byte before the allocated buffer region

0x7b96a5d245ef is located 1 bytes before 1-byte region [0x7b96a5d245f0,0x7b96a5d245f1) allocated by thread T0 here:

```
#0 0x7f76a771a0ab in malloc ../../../../src/libsanitizer/asan/asan_malloc_linux.cpp:67
```

```
#1 0x55d5fd9b3281 in get_name_from_name_field src/network/dns_utility.c:31
```

```
#2 0x55d5fd9bd123 in logger_dns_message src/module/logger.c:153
```

```
#3 0x55d5fd9af98e in main src/dns_relay.c:263
```

```
#4 0x7f76a6a29f67 in __libc_start_call_main ../sysdeps/nptl/libc_start_call_main.h:58
```

Based on the error message only we can determine that there is a buffer underflow vulnerability that triggered because of the truncated dns packet.

```
minal Help ← → Q buptLab-dns_relay_server ne
C dns_utility.c C logger.c C dns_relay.c X
src > C dns_relay.c > main(int, char *[])
206 int main(int argc, char *argv[]) {
209     logger_write(LOG_LEVEL_INFO, "Socket bind %s successfully.", DNS_PORT);
240
241     struct sockaddr_in dns_server_address;
242     memset(&dns_server_address, 0, sizeof(dns_server_address));
243     dns_server_address.sin_family = AF_INET;
244     dns_server_address.sin_port = htons(DNS_PORT);
245     dns_server_address.sin_addr.s_addr = inet_addr(options.isp_dns_server_ip);
246
247     struct sockaddr_in client_addr;
248     socklen_t client_addr_len = sizeof(client_addr);
249
250     while (1) {
251         ssize_t rcv_len = recvfrom(sockfd, buf, BUF_SIZE, 0, (struct sockaddr *)&client_addr, &client_addr_len);
252         if (rcv_len < 0) {
253             assert(rcv_len == -1);
254             if (errno != EINTR)
255                 logger_write(LOG_LEVEL_WARNING, "Failed when receiving!");
256             continue;
257         }
258         logger_write(LOG_LEVEL_INFO, "Received %zd byte(s) from client %s:%d.", rcv_len, inet_ntoa(client_addr.sin_addr),
259
260         logger_hex(LOG_LEVEL_DEBUG, (uint8_t *)buf, rcv_len);
261         dns_message_t *message = parse_dns_message(buf);
262
263         logger_dns_message(LOG_LEVEL_DEBUG, message);
264         distribute_frame(message, &dns_server_address, &client_addr);
265
266         dns_message_destroy(message);
267         message = NULL;
268     }
269
270     return 0;
271 }
272
```

Flameshot
Hello, I'm here! Click icon in the tray to take a screenshot. Right button to see more options.

in dns_relay.c line 263, we can see the bug triggered by logger_dns_message(LOG_LEVEL_DEBUG, message) function call.

Let's check the logger_dns_message function at modules/logger.c :

```
Help  ← →  buptLab-dns_relay_server
C dns_utility.c  C logger.c  X  C dns_relay.c
src > module > C logger.c > ...
117 }
118
119 void logger_dns_message(const log_level level, const dns_message_t *dns_message) {
120     char *buffer = malloc(1 << 20);
121     assert(buffer);
122     char *ptr = buffer;
123     sprintf(ptr, "\n\tid=0x%04" PRIx16 "\n", dns_message->header->id);
124     ptr = ptr + strlen(ptr);
125     sprintf(ptr, "\tqr=%d, opcode=%d, aa=%d, tc=%d, rd=%d, ra=%d, z=%d, ad=%d, cd=%d, rcode=%d \n",
126             dns_message->header->flag.flags.qr,
127             dns_message->header->flag.flags.opcode,
128             dns_message->header->flag.flags.aa,
129             dns_message->header->flag.flags.tc,
130             dns_message->header->flag.flags.rd,
131             dns_message->header->flag.flags.ra,
132             dns_message->header->flag.flags.z,
133             dns_message->header->flag.flags.ad,
134             dns_message->header->flag.flags.cd,
135             dns_message->header->flag.flags.rcode);
136     ptr = ptr + strlen(ptr);
137     sprintf(ptr, "\tqdcount=%" PRIu16 " , ancount=%" PRIu16 " , nscount=%" PRIu16 " , arcount=%" PRIu16 "\n",
138             dns_message->header->qdcount,
139             dns_message->header->ancount,
140             dns_message->header->nscount,
141             dns_message->header->arcount);
142     ptr = ptr + strlen(ptr);
143
144     forward_list_node_t *list_ptr;
145
146     sprintf(ptr, "\tquestions:\n");
147     ptr = ptr + strlen(ptr);
148
149     list_ptr = dns_message->questions;
150     for (size_t i = 0; i < dns_message->header->qdcount; ++i) {
151         assert(list_ptr);
152         question_t *question = (question_t *) (list_ptr->value);
153         char *name = get_name_from_name_field(question->qname);
154         sprintf(ptr, "\t\tqname=%s, qtype=%" PRIu16 " , qclass=%" PRIu16 "\n",
155                 name,
156                 question->qtype,
157                 question->qclass);
158     }
159 }
```

in line 153 the vulnerability triggered by a call to `get_name_from_name_field` :

```
char *name = get_name_from_name_field(question->qname);
```

```
C dns_utility.c X C logger.c C dns_relay.c
src > network > C dns_utility.c > dns_message_destroy(dns_message_t*)
8 name_field_t *name_field_create(const char *const name, size_t length) {
26     return p;
27 }
28
29 char *get_name_from_name_field(const name_field_t *const name_field) {
30     assert(name_field->length > 1);
31     char *result = malloc(name_field->length - 1);
32     char *base = result;
33     assert(result);
34     const uint8_t *ptr = name_field->name;
35     while (*ptr) {
36         memcpy(base, ptr + 1, *ptr);
37         base += *ptr;
38         *base++ = '.';
39         ptr += (*ptr + 1);
40     }
41     *--base = '\0';
42     return result;
43 }
44
45 static size_t name_length_default = 512;
46
47 dns_message_t *parse_dns_message(const char *const base) {
48     const char *ptr = base;
49     dns_message_t *p = malloc(sizeof(dns_message_t));
50     assert(p);
51
52     p->header = malloc(sizeof(dns_header_t));
53     assert(p->header);
54     memcpy(p->header, ptr, sizeof(dns_header_t));
```

The vulnerability occurs since this loop never occurs :

```
while (*ptr) {
    memcpy(base, ptr + 1, *ptr);
    base += *ptr;
    *base++ = '.';
    ptr += (*ptr + 1);
}
```

The loop won't run since name_field->name[0] == 0

```
*--base = '\0';
```

This code will shift the pointer position 1 step backward and fill the last character with null terminated string.

Since the name_field->name is Null, the base[-1] will be overwritten by null terminated string, hence a buffer underflow occurs !

POC

https://github.com/bluedragonsecurity/buptLab-dns_relay_server_remote_heap_based_buffer_underflow/blob/main/poc.c

References

<https://medium.com/@w1sdom/remote-heap-based-buffer-underflow-vulnerability-at-buptlab-dns-relay-server-bac6505070a9>

https://github.com/bluedragonsecurity/buptLab-dns_relay_server_remote_heap_based_buffer_underflow