Heap Based Buffer Over-Read Vulnerability in Rldns-1.3

Written by : Antonius (w1sdom)
Web : www.bluedragonsec.com

First of all, I apologize if this write-up feels a bit "robotic." As a roboticist and electronic engineer who spends most of my time building hardware, my writing style tends to be quite literal.

Vulnerability Discovery

During a fuzzing session with rldns-1.3, I discovered a **heap-based out-of-bounds read vulnerability**. I used **honggfuzz 2.6** for this session.

What is Rldns? Rldns is an open-source DNS server project that I maintain.

You can find the archives here : https://github.com/bluedragonsecurity/rldns_archives

and the source codes for latest rldns here : https://github.com/bluedragonsecurity/rldns

Currently, the latest version is rldns-1.4.

In February 2026, I finally had some spare time to fuzz the server. Through this process, I identified several vulnerabilities in version 1.3, including **null pointer dereferences** and **heap-based buffer over-reads**—which was quite alarming since I am the developer!

Fuzzing Methodology & Environment

To detect memory corruption, I relied on **AddressSanitizer (ASan)**. I used **honggfuzz** specifically to generate malformed DNS packets to trigger crashes.

I used the following `CFLAGS` and `LDFLAGS` to ensure ASan would capture the corruption details:

CFLAGS = -O1 -g -fno-omit-frame-pointer -fno-optimize-sibling-calls -D_FORTIFY_SOURCE=0 -fsanitize=address,undefined
LDFLAGS = -fsanitize=address,undefined

Full Makefile url :

https://raw.githubusercontent.com/bluedragonsecurity/rldns/refs/heads/main/Makefile_for_honggfuzz

To enable ASan logging, I created a shell script named `run.sh` to execute rldns

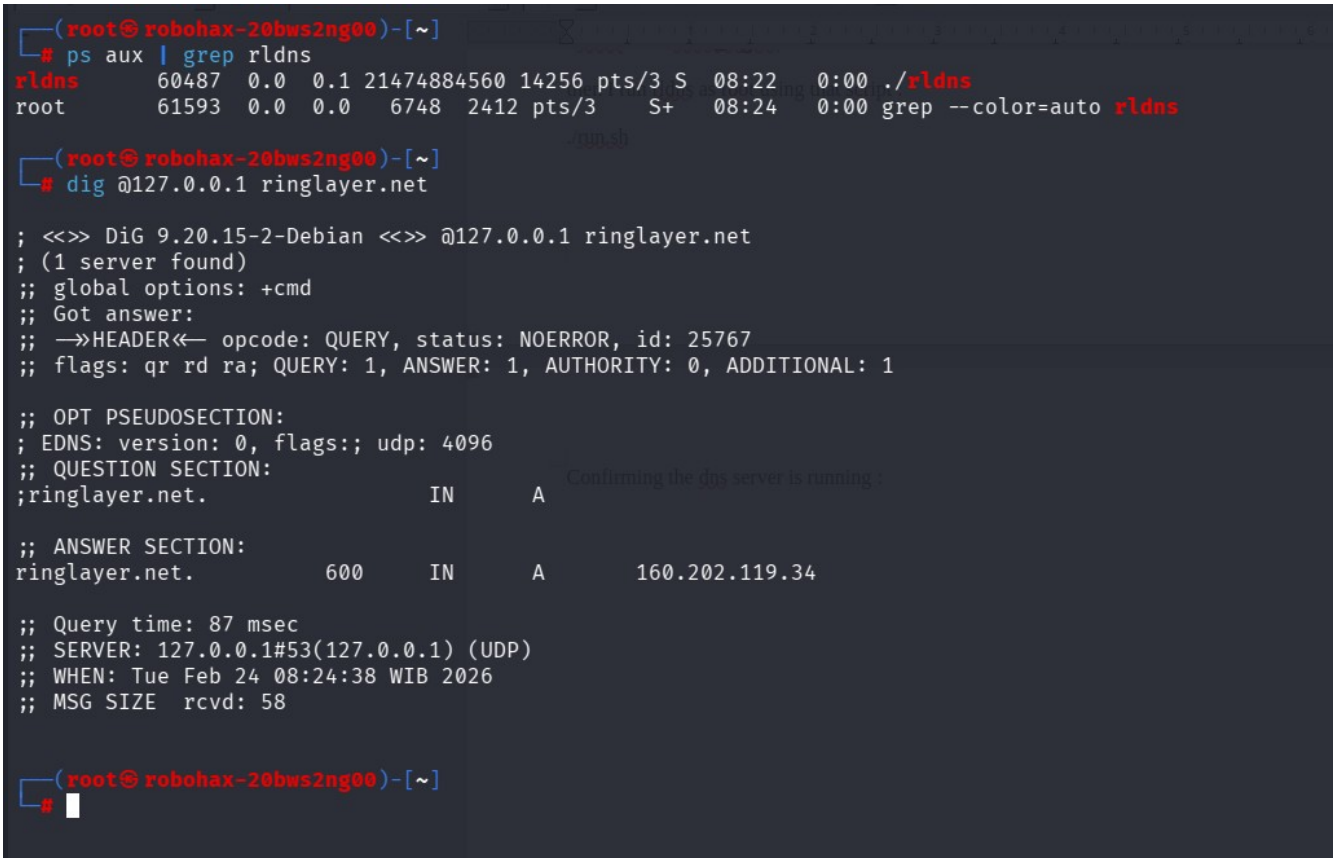
```
#!/bin/bash
pkill -9 rldns
echo "" > asan_log.txt
```

```
ASAN_PATH=$(gcc -print-file-name=libasan.so)
if [[ "$ASAN_PATH" == "libasan.so" ]]; then
   ASAN_PATH=$(ldd ./rldns | grep libasan | awk '{print $3}')
fi
export LD_PRELOAD="$ASAN_PATH"
export ASAN_OPTIONS="log_path=stderr:abort_on_error=1:detect_leaks=0"
./rldns &>> asan_log.txt &
```

then I run rldns as root using that script :

./run.sh

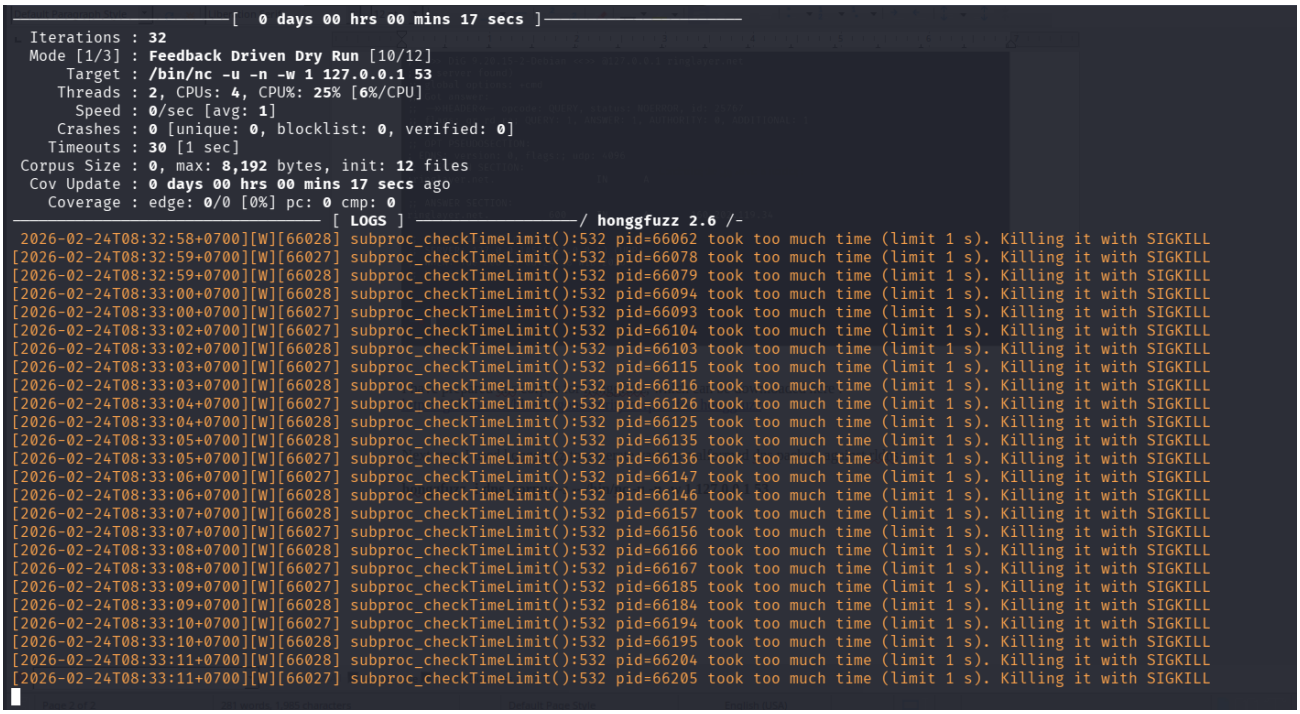
Confirming the dns server is running :



I have prepared dns corpus for honggfuzz, which can be downloaded here :
https://github.com/bluedragonsecurity/corpus_for_honggfuzz

Next step, I used honggfuzz to generate various malformed dns packets for rldns :

honggfuzz -i dns_corpus -s -- /bin/nc -u -n -w 1 127.0.0.1 53



Crash Analysis

Once rldns crashed, the ASan log reported a **READ of size 16**:

READ of size 16 at 0x7bff60fe0a7f thread T0
#0 0x7fdf62c7e4ce in strlen ../../../../src/libsanitizer/sanitizer_common/sanitizer_common_interceptors.inc:425
#1 0x564dea55f3fa in _count_octet_length src/main.c:1021
#2 0x564dea55f3fa in parse_enumerate_and_fetch_each_octet_and_just_return_a_name src/main.c:667
#3 0x564dea561d80 in rldns_main src/main.c:1943
#4 0x564dea568cb7 in main src/main.c:1882
#5 0x7fdf62029f67 in __libc_start_call_main ../sysdeps/nptl/libc_start_call_main.h:58
#6 0x7fdf6202a024 in __libc_start_main_impl ../csu/libc-start.c:360
#7 0x564dea552690 in _start (/home/robohax/Desktop/CVE/rldns Remote Heap-Based Buffer Over-read/rldns-1.3/rldns+0x2f690) (BuildId: b895cc58067deb351b9d579ecbaa4f52e8b2b4f1)

The vulnerability is triggered because the `strlen` function encounters an out-of-bounds read. Specifically, if `a_name` contains invalid bytes (like `0xff`), `strlen` may result in a segmentation fault.

The bug originates in the function `parse_enumerate_and_fetch_each_octet_and_just_return_a_name` at `src/main.c:667`. The routines responsible for building the name from a DNS query do not properly null-terminate the string. When a malformed packet ends with a non-null byte (e.g., `0xff`), the subsequent call to `strlen(a_name)` reads past the boundary of the allocated heap buffer.

Capturing the Malformed Packet

Since honggfuzz does not automatically log the specific packet that causes a crash, I created a shared object (`dns_logger.c`) to hook the `recvfrom` function and log the last received packet to `/tmp/last_packet.dns`

```
#define _GNU_SOURCE
#include <dlfcn.h>
#include <stdio.h>
#include <sys/socket.h>
#include <unistd.h>

ssize_t recvfrom(int sockfd, void *buf, size_t len, int flags,
                 struct sockaddr *src_addr, socklen_t *addrlen) {
    static ssize_t (*real_recvfrom)(int, void*, size_t, int, struct sockaddr*, socklen_t*) = NULL;
    if (!real_recvfrom) real_recvfrom = dlsym(RTLD_NEXT, "recvfrom");

    ssize_t n = real_recvfrom(sockfd, buf, len, flags, src_addr, addrlen);
    if (n > 0) {
        FILE *f = fopen("/tmp/last_packet.dns", "wb");
        if (f) {
            fwrite(buf, 1, n, f);
            fflush(f);
            fsync(fileno(f));
            fclose(f);
        }
    }
    return n;
}
```

Next, I compile it as shared object :

```
gcc -fPIC -shared -o dns_logger.so dns_logger.c -ldl
```

Since this shared object can not be preloded when ASan activated, I use the original Makefile from rldns-1.3 :

```
OBJ = obj
SRC = src
```

```
all : rldns

rldns : main.o
	gcc -Wall -fPIC -pthread -fstack-protector-all -o rldns $(OBJ)/main.o

main.o : $(SRC)/main.c $(SRC)/vars.h $(SRC)/structs.h $(SRC)/oops.h $(SRC)/headers.h
	gcc -Wall -fPIC -pthread -fstack-protector-all -c $(SRC)/main.c -o $(OBJ)/main.o

clean:
	rm -f $(OBJ)/*

install:
	rm -rf /usr/local/rldns;mkdir /usr/local/rldns
	mkdir /usr/local/rldns/zones
	cp rldns /usr/local/rldns
	cp rldns.conf /usr/local/rldns
	cp zones/* /usr/local/rldns/zones
	cp docs/rldns.1 /usr/share/man/man1
```

Recompile to test and run the rldns with LD_PRELOAD:

make clean && make

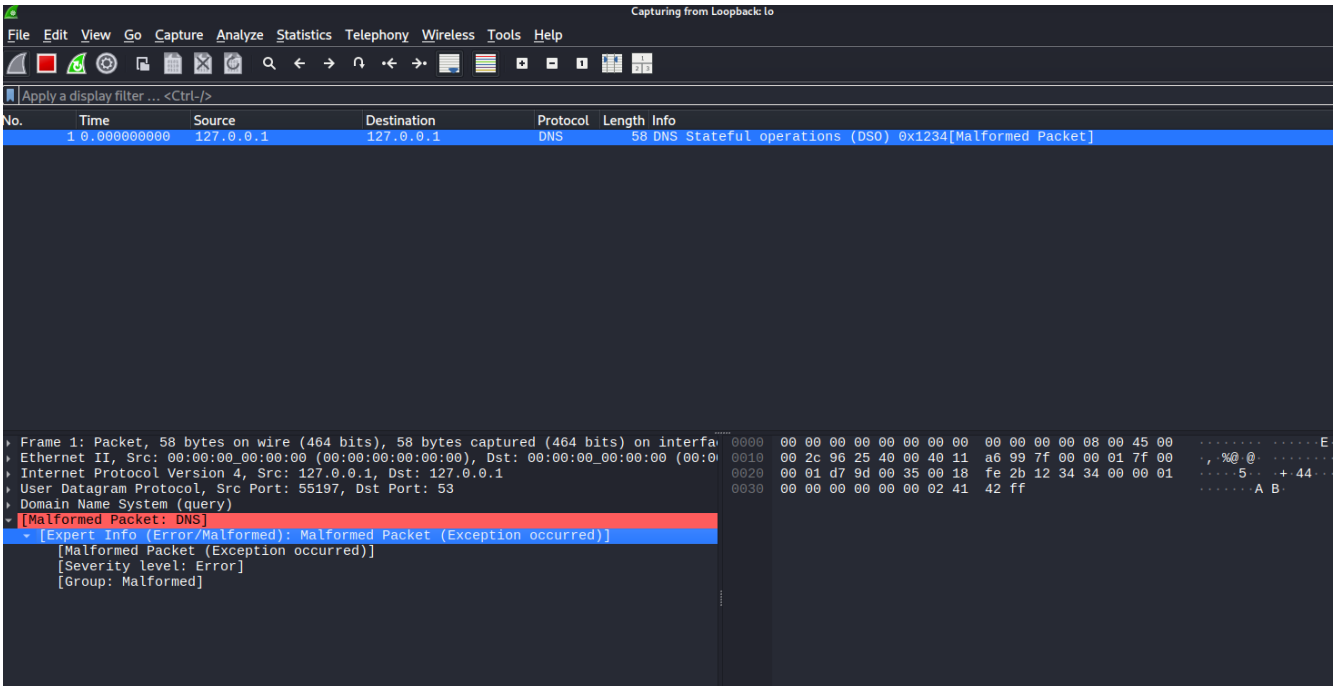
cp dns_logger.so /tmp/dns_logger.so
LD_PRELOAD=/tmp/dns_logger.so ./rldns

Next, I execute honggfuzz :

honggfuzz -i dns_corpus -s -- /bin/nc -u -n -w 1 127.0.0.1 53

Once rldns-1.3 crashes, I got the last dns packet from honggfuzz at **/tmp/last_packet.dns**

Using xxd, we know that it’s a malformed dns packet, since it contains only 16 bytes and ended with 0xff.



The POC

The PoC that crashes rldns-1.3 is available here:

https://github.com/bluedragonsecurity/rldns-1.3-heap-out-of-bounds-vulnerability-fixed-in-rldns-1.4/

I expect to be busy over the next three years developing intelligence/hacking devices and hunting for 0-days. If you discover any vulnerabilities in rldns, please report them to me at: **bluedragonsec2023@gmail.com**.