Potential Vulnerabilities in the Contiki-NG stack

Out of bounds read when using Neighbor Solicitation Processing in IPv6 CIMP

Description

The function ns_input in (os/net/ipv6/uip-nd6.c) (line 173 as of cc7f61c) decodes incoming IPv6 ICMP data related to neighbor solicitation. There are many checks to ensure the length of the data (`uip_len`) does not exceed the maximum size and will not be smaller than all ICMP+IPv6 header, but there are no checks to ensure there is enough data for the neighbor solicitation header, leading to out of bounds access.

Technical Details

This function works with data in the packet buffer. If the `UIP_CONF_IPV6_CHECKS` macro is enabled, then a cast will occur on line 188 that seeks to the ICMP payload (past all IPv6 + ICMP headers) and casts to a `uip_nd6_ns` struct. This struct has a size of 20 bytes but there is no check (either in the vulnerable function or in its caller) that the ICMP payload has up to 20 additional bytes. Hence, if the ICMP payload is incomplete or truncated, there will be an out-of-bound access.

Impact

An out-of-bounds read can cause a crash and lead to a denial of service. This condition could occur on Contiki-ng deployments that support memory safety checks at the hardware level or the software level (e.g., via ASAN-style instrumentation).

Fix Recommendation

We recommend adding a check to ensure there is enough space to perform this cast: if(uip_l3_icmp_hdr_len + sizeof(uip_nd6_ns) > uip_len) The function can return (or drop) the packet if this condition is true. Here is a PR that makes this change.

Out of bounds read when inputting l2cap frame flow channel

Description

The function input_l2cap_frame_flow_channel in (os/net/mac/ble/ble-l2cap.c) (line 416 as of cc7f61c) decodes incoming packet data, and performs many copies. There are many checks to

ensure the destination is large enough for the data, but no checks are performed to ensure the source is large enough, leading to out of bounds reads.

Technical Details

This function works with data in the packet buffer. First, the payload length is read directly from the payload on line 429. Later, data is read from the packet buffer using this payload size (lines 440 and 454). This size is checked on line 432 where it is stated the payload length will not exceed the destination size (which is usually identical to the packer buffer size).

The problem is when the payload length is the same size or slightly smaller (at most 6 bytes). The memcpy operations listed above perform the copy using offsets from the start, so if the payload length is the size of the packetbuf and we offset by 6, then we have 6 bytes of out of bounds data read. Because the payload length can be provided by the attacker, this scenario could occur by crafting a malicious packet.

Impact

An out-of-bounds read can cause a crash and lead to a denial of service. This condition could occur on Contiki-ng deployments that support memory safety checks at the hardware level or the software level (e.g., via ASAN-style instrumentation).

Fix Recommendation

We recommend adding checks to ensure there is enough space in the packet buffer when taking into account the offset. On line 432, one can add a check that (payload_len > data_len - 6). Additionally, on line 448, a similar check can be added that checks that (payload_len > data_len - 4). Here is a PR that makes this change.

Out of Bounds Read in DIO Input Processing in RPL-Classic

Description

The dio_input function in rpl-icmp6.c file in rpl-classic has multiple out-of-bound reads. The buffer is indexed multiple times (with index variable i) without any validation that i is within the bounds of remaining buffer data. Up to 9 + sizeof(dio.dag_id) bytes can be read out-of-bounds at the start of the packet, with multiple more out-of-bound read possible within the function.

Interestingly, this vulnerability was fixed last year in the rpl-lite version of rpl-icmp6.c file (PR here). Why was this not fixed in rpl-classic? As we can see from this PR, rpl-classic is still undergoing active development.

Fix Recommendation

The fix in PR #2484 for rpl-lite should be ported to rpl-classic (adjusted for the size of data that is being read. Here is a PR that makes this change.