bareiron – Vulnerability Report

Reported by: Soulaimane Hafid (suleif) + vmpr0be
Date: 2025-12-02

---

1. Overview

- **Project:** bareiron
- **Total Vulnerabilities:** 4
- **Reported On:** 2025-12-03
- **CVE Request:** Yes (after patch)

---

2. Vulnerability List

Vulnerability 1: Improper Input Validation Leading to Out-of-Bounds Access

Discovered by: Soulaimane Hafid (suleif) + vmpr0be
Severity: High

Impact:

- Allows assigning an 8-bit integer with an arbitrary value, leading to OOB reads.
- **Affected versions:** Starting from commit `269e8346ebfb97177f6e11ebd108149920c86fd0` .

Technical Summary:
The client can set the `hotbar` field to any user-controlled integer without validation. Since the player's inventory is ~50 elements, this results in out-of-bounds (OOB) accesses throughout the codebase.

Path: `handlePacket` → `cs_setHeldItem`

```
https://github.com/p2r3/bareiron/blob/8e4d4020d7e0797153512768621fba12f0ce68f1/src/packets.c#L809
int cs_setHeldItem (int client_fd) {
  ...
  player->hotbar = (uint8_t)readUint16(client_fd);S
  ...
}
```

`handlePlayerAction` does not validate whether `player->hotbar` is within inventory bounds. The previously corrupted index is later used to read or write memory beyond the intended arrays, allowing controlled OOB access of up to 255 bytes.

Path: `handlePacket` -> `cs_playerAction` -> `handlePlayerAction` -> `sc_setContainerSlot`

```
https://github.com/p2r3/bareiron/blob/8e4d4020d7e0797153512768621fba12f0ce68f1/src/packets.c#L459
int sc_setContainerSlot (int client_fd, int window_id, uint16_t slot, uint8_t count, uint16_t item) {
  ...i
  writeVarInt(client_fd, count);
  if (count > 0) {
    writeVarInt(client_fd, item);
  ...
}

https://github.com/p2r3/bareiron/blob/8e4d4020d7e0797153512768621fba12f0ce68f1/src/procedures.c#L1153
void handlePlayerAction (PlayerData *player, int action, short x, short y, short z) {
  ...

  if (action == 3 || action == 4) {
    sc_setContainerSlot(
      player->client_fd, 0,
      serverSlotToClientSlot(0, player->hotbar),
      player->inventory_count[player->hotbar], <=== !!!!
      player->inventory_items[player->hotbar]  <=== !!!!
    );
    return;
  }

  ...
}
```

Additional uses of `player->hotbar` across the code provide attackers with limited but meaningful manipulation primitives:

```
void bumpToolDurability (PlayerData *player) {
  ...
  if (...) {
    https://github.com/p2r3/bareiron/blob/8e4d4020d7e0797153512768621fba12f0ce68f1/src/procedures.c#L727
    player->inventory_items[player->hotbar] = 0; <=== !!! OOB nulling
    player->inventory_count[player->hotbar] = 0; <=== !!! OOB nulling
    sc_entityEvent(player->client_fd, player->client_fd, 47);
    sc_setContainerSlot(player->client_fd, 0, serverSlotToClientSlot(0, player->hotbar), 0, 0);
  }
}

uint8_t handlePlayerEating (PlayerData *player, uint8_t just_check) {
  ...
  https://github.com/p2r3/bareiron/blob/8e4d4020d7e0797153512768621fba12f0ce68f1/src/procedures.c#L985
  uint16_t *held_item = &player->inventory_items[player->hotbar];
  uint8_t *held_count = &player->inventory_count[player->hotbar];

  ...
  https://github.com/p2r3/bareiron/blob/8e4d4020d7e0797153512768621fba12f0ce68f1/src/procedures.c#L1018
  *held_count -= 1; <=== OOB decrement !!
  if (*held_count == 0) *held_item = 0;  <=== !!! OOB nulling

  ...
  https://github.com/p2r3/bareiron/blob/8e4d4020d7e0797153512768621fba12f0ce68f1/src/procedures.c#L1024
  sc_setContainerSlot(
    player->client_fd, 0,
    serverSlotToClientSlot(0, player->hotbar),
    *held_count, *held_item <=== !!!! OOB leaking
  );
```

Suggested Fix:
Validate the hotbar value before assignment to ensure it is within the valid inventory index range.

Vulnerability 2: Buffer Overflow via Unvalidated User-Controlled Length

Discovered by: Soulaimane Hafid (suleif)
Severity: Medium
Impact:

- Allows the client to trigger a buffer overflow, overwriting memory past `recv_buffer`
- Results in a server crash (DoS)
- **Affected versions:** Starting from commit `4a90979e2f4e16dd5802b88935f75d31c930a701`

Technical Summary:
`handlePacket` does not check whether the user-controlled packet length fits within the `recv_buffer`. This results in a global variable buffer overflow inside `recv_all`, enabling overwriting of adjacent data and causing a server crash.

```
int main () {
    ...
    https://github.com/p2r3/bareiron/blob/8e4d4020d7e0797153512768621fba12f0ce68f1/src/main.c#L695
    int length = readVarInt(client_fd);
    ...
    https://github.com/p2r3/bareiron/blob/8e4d4020d7e0797153512768621fba12f0ce68f1/src/main.c#L714
    handlePacket(client_fd, length - sizeVarInt(packet_id), packet_id, state);
    ...

void handlePacket (int client_fd, int length, int packet_id, int state) {
    ...
    case 0x1B:
      if (state == STATE_PLAY) {
        https://github.com/p2r3/bareiron/blob/8e4d4020d7e0797153512768621fba12f0ce68f1/src/main.c#L195
        recv_all(client_fd, recv_buffer, length, false); <==== !!!! length is not checked
    ...
    default:
      https://github.com/p2r3/bareiron/blob/8e4d4020d7e0797153512768621fba12f0ce68f1/src/main.c#L469
      recv_all(client_fd, recv_buffer, length, false); <===== !!! same here
      break;
    ...
```

Suggested Fix:
Check that `length` does not exceed the `recv_buffer` capacity before calling `recv_all`.

Vulnerability 3: Improper Input Validation on Entity ID Leading to Out-of-Bounds Memory Access

Discovered by: vmpr0be
Severity: Medium
Impact:

- Arbitrary 5-byte read within binary memory bounds
- Possible constrained data corruption
- **Affected versions:** Starting from commit `ba86dfd927b5e2432be797e12095642dc4091fe1`

Technical Summary:
An unvalidated entity ID from the client is used as an array index (negated) into mob_data, resulting in OOB memory access and potential disclosure of memory values via coordinates sent back to the client.

Path: `handlePacket` -> `cs_interact` -> `interactEntity`

```
int cs_interact (int client_fd) {
  https://github.com/p2r3/bareiron/blob/ba86dfd927b5e2432be797e12095642dc4091fe1/src/packets.c#L1167
  int entity_id = readVarInt(client_fd);
  uint8_t type = readByte(client_fd);
  ...
  https://github.com/p2r3/bareiron/blob/ba86dfd927b5e2432be797e12095642dc4091fe1/src/packets.c#L1182C7-L1182C11
  if (type == 0) { // Interact
    interactEntity(entity_id, client_fd); <== !!! entity_id isn't checked
  }
  ...

void interactEntity (int entity_id, int interactor_id) {
  ...
  https://github.com/p2r3/bareiron/blob/ba86dfd927b5e2432be797e12095642dc4091fe1/src/procedures.c#L1405
  MobData *mob = &mob_data[-entity_id - 2]; <=== OOB access using unchecked value

  switch (mob->type) {
    case 106: // Sheep
      ...

      https://github.com/p2r3/bareiron/blob/ba86dfd927b5e2432be797e12095642dc4091fe1/src/procedures.c#L1420C7-L1420C26
      playPickupAnimation(player, I_white_wool, mob->x, mob->y, mob->z); <=== !!!!! x, y and z are sent to client
```

Suggested Fix:
Validate `entity_id` to ensure it is negative and falls within the bounds of the `mob_data` array.

Vulnerability 4: Arbitrary Write via Unchecked Entity ID in Container Interaction

Discovered by: vmpr0be
Severity: Critical
Impact:

- Full arbitrary memory write
- Using crafted values, attacker can overwrite `storage_ptr` inside `player->craft_items`.
- **Affected versions:** Starting from commit `ba86dfd927b5e2432be797e12095642dc4091fe1`

Technical Summary:
A user-controlled slot index is used to compute pointers into either inventory or chest storage. Because the slot is not validated, an attacker may overwrite arbitrary memory, including `player->craft_items`, enabling manipulation of `storage_ptr` and achieving arbitrary writes.

Path: `handlePacket` -> `cs_clickContainer` -> `interactEntity`

```
int cs_clickContainer (int client_fd) {
  ...
  int window_id = readVarInt(client_fd);
  ...

  uint8_t slot, count, craft = false;
  uint16_t item;
  int tmp;

  uint16_t *p_item;
  uint8_t *p_count;

  ...

  for (int i = 0; i < changes_count; i ++) {

    slot = clientSlotToServerSlot(window_id, readUint16(client_fd)); <== user-controlled value

    ...
    #ifdef ALLOW_CHESTS
    if (window_id == 2 && slot > 40) {
      p_item = (uint16_t *)(storage_ptr + (slot - 41) * 3);
      p_count = storage_ptr + (slot - 41) * 3 + 2;
    } else
    #endif
    {
      p_item = &player->inventory_items[slot];
      p_count = &player->inventory_count[slot];
    }

    ...

    if (count > 0 && apply_changes) {
      *p_item = item; <===== overwrite arbitrary element
      *p_count = count;
      ...
    }
    ...
```

Suggested Fix:
Validate slot and ensure `player->craft_items` cannot be modified when a `storage_ptr` is present.