# CWE Detail – CWE-1321

## Description

The product receives input from an upstream component that specifies attributes that are to be initialized or updated in an object, but it does not properly control modifications of attributes of the object prototype.

## Extended Description

By adding or modifying attributes of an object prototype, it is possible to create attributes that exist on every object, or replace critical attributes with malicious ones. This can be problematic if the product depends on existence or non-existence of certain attributes, or uses pre-defined attributes of object prototype (such as hasOwnProperty, toString or valueOf). This weakness is usually exploited by using a special attribute of objects called proto, constructor or prototype. Such attributes give access to the object prototype. This weakness is often found in code that assigns object attributes based on user input, or merges or clones objects recursively.

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-2018-3721: Prototype pollution by merging objects.

**•** CVE-2019-10744: Prototype pollution by setting default values to object attributes recursively.

**•** CVE-2019-11358: Prototype pollution by merging objects recursively.

**•** CVE-2020-8203: Prototype pollution by setting object attributes based on dot-separated path.

## Related Attack Patterns (CAPEC)

* CAPEC-1
* CAPEC-180
* CAPEC-77

## Attack TTPs

**•** T1574.010: Services File Permissions Weakness (Tactics: persistence, privilege-escalation, defense-evasion)

## Modes of Introduction

**•** Architecture and Design: N/A

**•** Implementation: N/A

## Common Consequences

**•** Impact: Modify Application Data — Notes: An attacker can inject attributes that are used in other components.

**•** Impact: DoS: Crash, Exit, or Restart — Notes: An attacker can override existing attributes with ones that have incompatible type, which may lead to a crash.

## Potential Mitigations

**•** Implementation: By freezing the object prototype first (for example, Object.freeze(Object.prototype)), modification of the prototype becomes impossible. (Effectiveness: High)

**•** Architecture and Design: By blocking modifications of attributes that resolve to object prototype, such as proto or prototype, this weakness can be mitigated. (Effectiveness: High)

**•** Implementation: When handling untrusted objects, validating using a schema can be used. (Effectiveness: Limited)

**•** Implementation: By using an object without prototypes (via Object.create(null) ), adding object prototype attributes by accessing the prototype via the special attributes becomes impossible, mitigating this weakness. (Effectiveness: High)

**•** Implementation: Map can be used instead of objects in most cases. If Map methods are used instead of object attributes, it is not possible to access the object prototype or modify it. (Effectiveness: Moderate)

## Applicable Platforms

**•** JavaScript (Class: None, Prevalence: Undetermined)

## Demonstrative Examples

**•** This function does not check if the attribute resolves to the object prototype. These codes can be used to add "isAdmin: true" to the object prototype.