# Mass Assignment: Insecure Binder Development Mitigation SOP

To ease development and increase productivity, most modern frameworks allow an object to be automatically instantiated and populated with the HTTP request parameters whose names match an attribute of the class to be bound. Automatic instantiation and population of objects speeds up development, but can lead to serious problems if implemented without caution. Any attribute in the bound classes, or nested classes, will be automatically bound to the HTTP request parameters. Therefore, malicious users will be able to assign a value to any attribute in bound or nested classes, even if they are not exposed to the client through web forms or API contracts.

## Defense Against Mass Assignment: Insecure Binder

It is a best practice to control which attributes will be bound to the model object so that even if attackers figure out other non-exposed attributes of the model or nested classes, they will not be able to bind arbitrary values from HTTP request parameters.

## Example

Let’s say we have a controller class called ExampleClaimController, with a controller method whose purpose is to update a claim’s claim date and suspense date, but nothing else.

// Request Mappings

private static final String UPDATE\_CLAIM\_DATES\_REQUEST\_MAPPING = "/updateClaimDates";

// Model Attributes

public static final String CLAIM = "claim";

@ResponseBody

@Authorize(policy = SystemPolicy.EDIT\_CLAIM\_DETAILS)

@RequestMapping(value = UPDATE\_CLAIM\_DATES\_REQUEST\_MAPPING, method = RequestMethod.POST)

public AjaxResponse<ClaimSummary> claimDatesUpdate(@ModelAttribute(CLAIM) ClaimSummary partialClaim,

                                 @RequestParam String claimID) {

      // pass claim summary data

}

## Explanation

The corresponding web form that retrieves input from the user will have the fields claimID, claimDateDt, and suspenseDate.

The controller method uses Spring to automatically bind the claimDateDt and suspenseDate fields from the body of the HTTP request to the ClaimSummary object: partialClaim. This object is annotated with the @ModelAttribute annotation (this annotation is optional; request data can still be bound to a POJO that is not annotated). The @RequestParam annotation will automatically bind a claimID request parameter to the claimID String.

If no changes are made to this code, it's possible that a malicious user could change/update more than the claim date and suspense date by manually setting other fields in the body of the request. Suppose veteranPersonId is an additional field that gets passed in the request body from the client maliciously along with the other expected fields. Since the ClaimSummary object has a veteranPersonId field, Spring will automatically bind veteranPersonId to the ClaimSummary object. In the controller method, if the code is not careful in how it handles the partialClaim object, this veteranPersonId could possibly be saved/updated with the other expected fields.

## Recommendation

It is security best practice to tell Spring specifically what fields to automatically bind to. This is done through a custom method annotated with @InitBinder that provides a “whitelist” of fields that are allowed to be bound for a particular object.

The code below demonstrates a very basic binder configuration.

@InitBinder

public void initBinder(WebDataBinder binder, WebRequest webRequest) {

binder.setAllowedFields(“claimDateDt”, “suspenseDate”);

}

The two fields which are expected from the client are passed to the setAllowedFields method of the data binder. Now, any other fields that are set in the body of HTTP requests will not be bound to the ClaimSummary object.

If a client sent these attributes to the controller:

“claimID” : “1”

“claimDateDt” : “05/02/2015”

“suspenseDate” : “06/02/2015”

“veteranPersonId” : “2”

In the controller method:

RequestParam claimId=1;

The partialClaim object would have these values if allowedFields is NOT set:

claimDateDt=05/02/2015

suspenseDate=06/02/2015

veteranPersonId=2

The partialClaim object would have these values if allowedFields is set:

claimDateDt=05/02/2015

suspenseDate=06/02/2015

veteranPersonId=null

As you can see, veteranPersonId will remain a null value and not be set if allowedFields is properly configured and set. This ensures this value will not be inadvertently used/set during processing in the controller method.

**Awards Specific Mitigation**

**Location:** webapp/src/main/java/gov/va/vba/award/mvc/initbinders/AwardsInitBinders.java

Mitigation can be achieved through the use of the custom AwardsInitBinders class. AwardsInitBinders.java provides protection by explicitly allowing or disallowing which parameters are to be bound on the specified uri request. Within AwardsInitBinders, InitBinder methods are added for the request mapping to protect against any unwanted parameters.

## References

1. [OWASP, Mass Assignment](mailto:https://www.owasp.org/index.php/CRV2_FrameworkSpecIssuesSpring)
2. [Spring, Spring MVC Known Vulnerabilities and Issues](mailto:http://support.springsource.com/security/spring-mvc)
3. [INPUT-1: Validate inputs, Oracle](http://www.oracle.com/technetwork/java/seccodeguide-139067.html%235)