# Mass Assignment: Sensitive Field Exposure Developer SOP

Modern frameworks allow developers to automatically bind HTTP request parameters from both request query and body into model objects for ease of development and increased productivity. If the binder is not correctly configured to control which HTTP request parameters are bound to which model attributes, an attacker may be able to abuse the model binding process and set any other attributes that should not be exposed to user control. This binding is possible even if the model attributes do not appear in the web forms or API contracts.

# Mass Assignment: Sensitive Field Exposure Prevention

In order to avoid mass assignment vulnerabilities it is important to control the HTTP request to model class binding process. There are different alternatives depending on the framework used:

**Architectural:** Define dedicated DTO classes to be bound to user data and containing only those attributes that should be exposed to end users. Map their attributes with the Domain Objects used along the application. These Domain Objects will contain attributes to receive the validated user data and extra attributes that should never be exposed to user control.

**Binder configuration:** Certain frameworks such as Spring, allow the developers to configure the model binder to accept or reject HTTP request parameters based on their names.

# Example

The following Spring controller method (registerUser) is accessed from a web form that asks a user to register an account by providing their first name, last name, and age:

@RequestMapping(method = RequestMethod.POST)

public String registerUser(@ModelAttribute("user") User user, BindingResult result, SessionStatus status) {

if (db.save(user).hasErrors()) {

return "CustomerForm";

} else {

status.setComplete();

return "CustomerSuccess";

}

}

# References

<http://www.hpenterprisesecurity.com/vulncat/en/vulncat/java/mass_assignment_sensitive_field_exposure.html>