Security

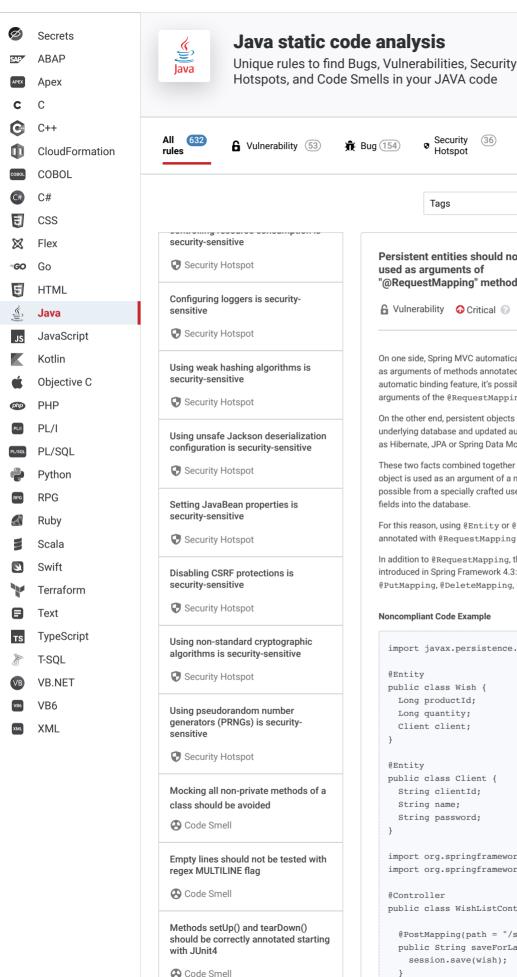
Hotspot

(36)



Products >

Quick 42



Search by name. Tags Persistent entities should not be used as arguments of Analyze your code "@RequestMapping" methods cwe spring owasp On one side, Spring MVC automatically bind request parameters to beans declared as arguments of methods annotated with @RequestMapping. Because of this automatic binding feature, it's possible to feed some unexpected fields on the $arguments \ of \ the \ {\tt @RequestMapping} \ annotated \ methods.$ On the other end, persistent objects (@Entity or @Document) are linked to the underlying database and updated automatically by a persistence framework, such as Hibernate, JPA or Spring Data MongoDB. These two facts combined together can lead to malicious attack: if a persistent object is used as an argument of a method annotated with @RequestMapping, it's possible from a specially crafted user input, to change the content of unexpected fields into the database. For this reason, using @Entity or @Document objects as arguments of methods annotated with @RequestMapping should be avoided. In addition to @RequestMapping, this rule also considers the annotations introduced in Spring Framework 4.3: @GetMapping, @PostMapping, @PutMapping, @DeleteMapping, @PatchMapping. **Noncompliant Code Example** import javax.persistence.Entity; @Entity public class Wish { Long productId:

Long quantity;

Client client:

public class Client {

String clientId; String name;

String password;

import org.springframework.stereotype.Controller;

public class WishListController {

session.save(wish);

@PostMapping(path = "/saveForLater")

public String saveForLater(Wish wish) {

 $import\ org. spring framework. web. bind. annotation. Request \texttt{Mappin}$

@RequestMapping(path = "/saveForLater", method = RequestMe

@Entity

@Controller

⊗ Code

Smell

(389)

Class members annotated with

Java static code analysis: Persistent entities should not be used as arguments of "@RequestMapping" methods

"@VisibleForTesting" should not be accessed from production code

Code Smell

"String#replace" should be preferred to "String#replaceAll"

A Code Smell

Derived exceptions should not hide their parents' catch blocks

Code Smell

String offset-based methods should be preferred for finding substrings from offsets

Code Smell

```
public String saveForLater(Wish wish) {
    session.save(wish);
}
```

Compliant Solution

```
public class WishDTO {
 Long product Id:
  Long quantity;
 Long clientId;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMappin
@Controller
public class PurchaseOrderController {
  @PostMapping(path = "/saveForLater")
  public String saveForLater(WishDTO wish) {
    Wish persistentWish = new Wish();
    // do the mapping between "wish" and "persistentWish"
    [...]
    session.save(persistentWish);
  @RequestMapping(path = "/saveForLater", method = RequestMe
  public String saveForLater(WishDTO wish) {
    Wish persistentWish = new Wish();
    // do the mapping between "wish" and "persistentWish" \,
    [...]
    session.save(persistentWish);
```

Exceptions

No issue is reported when the parameter is annotated with <code>@PathVariable</code> from Spring Framework, since the lookup will be done via id, the object cannot be forged on client side.

See

- OWASP Top 10 2021 Category A8 Software and Data Integrity Failures
- OWASP Top 10 2017 Category A5 Broken Access Control
- MITRE, CWE-915 Improperly Controlled Modification of Dynamically-Determined Object Attributes
- Two Security Vulnerabilities in the Spring Framework's MVC by Ryan Berg and Dinis Cruz

Available In:

sonarlint ⊕ | sonarcloud 💩 | sonarqube

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.

Privacy Policy