Fortify Software Security Center

Developer Workbook

IWA - 1

Note: This report calculates counts based on issue priority. Issue priority is initially set based on the raw scan information. However, reviewers are able to modify the original issue priority based on additional contextual information. If the issue details section is included in the report, it will indicate the issues where the original value has been changed.

Table of Contents

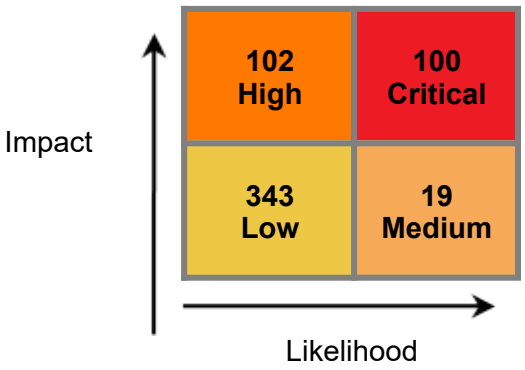
Executive Summary

This workbook is intended to provide all necessary details and information for a developer to understand and remediate the different issues discovered during the IWA - 1 project audit. The information contained in this workbook is targeted at project managers and developers.

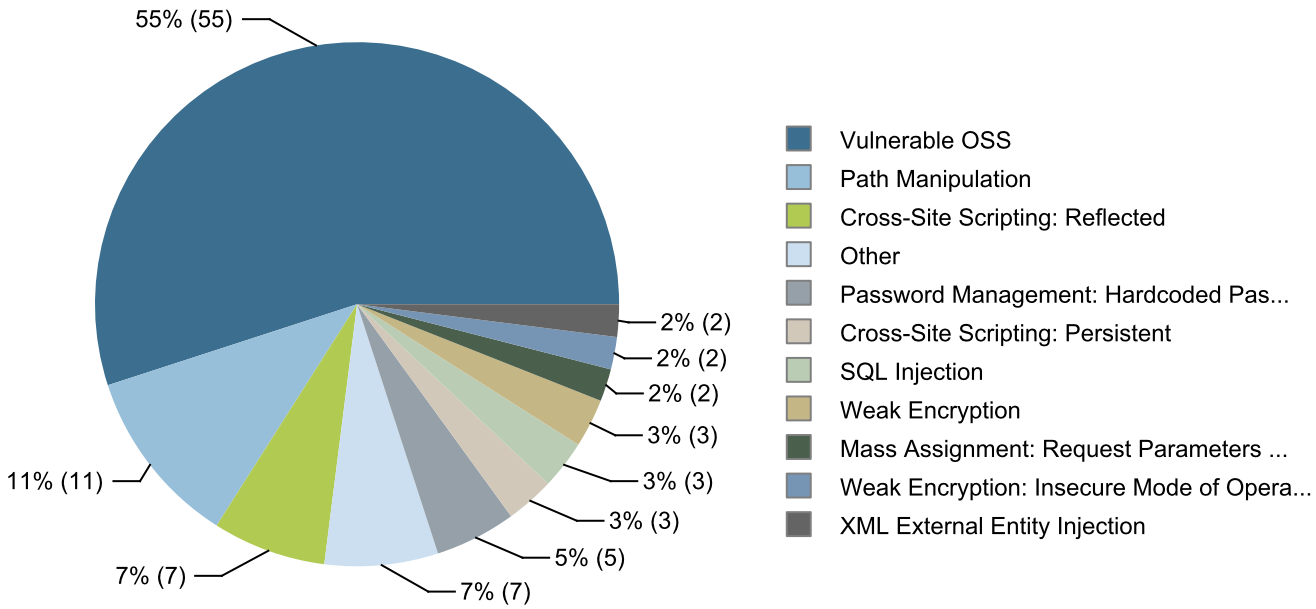
This section provides an overview of the issues uncovered during analysis.

Project Name:	IWA
Project Version:	1
SCA:	Results Present
WebInspect:	Results Present
WebInspect Agent:	Results Not Present
Other:	Results Present

Issues by Priority



Top Ten Critical Categories



** Denotes the existence of issues changed from or to this priority. The issue detail section identifies the specific issues.

Project Description

This section provides an overview of the Fortify scan engines used for this project, as well as the project meta-information.

SCA

Date of Last Analysis:	Sep 14, 2023 9:43 AM	**Engine Version:**	23.1.0.0136
Host Name:	EC2AMAZ-V0MASK3	**Certification:**	VALID
Number of Files:	240	**Lines of Code:**	9,078

Rulepack Name	Rulepack Version
Fortify Secure Coding Rules, Community, Cloud	2023.2.0.0007
Fortify Secure Coding Rules, Community, Universal	2023.2.0.0007
Fortify Secure Coding Rules, Core, Android	2023.2.0.0007
Fortify Secure Coding Rules, Core, Annotations	2023.2.0.0007
Fortify Secure Coding Rules, Core, Cloud	2023.2.0.0007
Fortify Secure Coding Rules, Core, Java	2023.2.0.0007
Fortify Secure Coding Rules, Core, JavaScript	2023.2.0.0007
Fortify Secure Coding Rules, Core, SQL	2023.2.0.0007
Fortify Secure Coding Rules, Core, Universal	2023.2.0.0007
Fortify Secure Coding Rules, Extended, Configuration	2023.2.0.0007
Fortify Secure Coding Rules, Extended, Content	2023.2.0.0007
Fortify Secure Coding Rules, Extended, Java	2023.2.0.0007
Fortify Secure Coding Rules, Extended, JavaScript	2023.2.0.0007
Fortify Secure Coding Rules, Extended, JSP	2023.2.0.0007
Fortify Secure Coding Rules, Extended, SQL	2023.2.0.0007

SONATYPE

Date of Last Analysis:	Dec 16, 2022 10:06 AM	**Engine Version:**	1.0
Host Name:	isNew,admin	**Certification:**	NOT_PRESENT
Number of Files:	0	**Lines of Code:**	0

WEBINSPECT

Date of Last Analysis:	Jun 21, 2023 4:49 AM	**Engine Version:**	22.1
Host Name:	https://iwa.onfortify.com/	**Certification:**	NOT_PRESENT
Number of Files:	0	**Lines of Code:**	0

Accessibility	External Public Network Access Required
Development Phase	Active Development
Development Strategy	Partially Outsourced

Issue Breakdown by Fortify Categories

The following table depicts a summary of all issues grouped vertically by Fortify Category. For each category, the total number of issues is shown by Fortify Priority Order, including information about the number of audited issues.

Category	Fortify Priority (audited/total)				Total Issues
	Critical	High	Medium	Low	
Access Control: Database	0	8 / 8	0	28 / 28	36 / 36
API Discovery	0	0	0	0 / 2	0 / 2
Build Misconfiguration: External Maven Dependency Repository	0	0	0	0 / 1	0 / 1
Cache Management: Headers	0	0	1 / 2	0	1 / 2
Code Correctness: Byte Array to String Conversion	0	0	0	0 / 2	0 / 2
Compliance Failure: Missing Privacy Policy	0	0	0	1 / 1	1 / 1
Cookie Security: Cookie not Sent Over SSL	0	0	1 / 1	0	1 / 1
Cookie Security: Missing SameSite Attribute	0	0	0	3 / 18	3 / 18
Cross-Frame Scripting	0	0 / 1	1 / 1	0	1 / 2
Cross-Site Request Forgery	0	0	0	0 / 7	0 / 7
Cross-Site Scripting: Persistent	3 / 3	0	0	0	3 / 3
Cross-Site Scripting: Reflected	4 / 7	0	0	0	4 / 7
Database Bad Practices: Use of Restricted Accounts	0	0	0	0 / 3	0 / 3
Dead Code: Unused Field	0	0	0	0 / 10	0 / 10
Dead Code: Unused Method	0	0	0	0 / 1	0 / 1
Header Manipulation	0	4 / 4	0	0	4 / 4
Hidden Field	0	0	0	0 / 2	0 / 2
HLI: Detected Libraries : Bootstrap v4.5.3	0	0	0	1 / 1	1 / 1
HLI: Detected Libraries : core-js vcore-js-pure@2.6.11; core-js-pure@2.6.11	0	0	0	0 / 1	0 / 1
HLI: Detected Libraries : jQuery (Fast path)	0	0	0	1 / 1	1 / 1
HLI: Detected Libraries : jQuery v3.5.1	0	0	0	1 / 1	1 / 1
HLI: Detected Libraries : Moment.js v2.29.1	0	0	0	1 / 1	1 / 1
HLI: Detected Libraries : React	0	0	0	0 / 1	0 / 1
HLI: Detected Libraries : React (Fast path)	0	0	0	0 / 1	0 / 1
HTML5: CORS Unsafe Methods Allowed	0	0	0	1 / 4	1 / 4
HTML5: Missing Content Security Policy	0 / 2	0	0	1 / 1	1 / 3
HTML5: Missing Framing Protection	0 / 1	0	0	0	0 / 1
HTML5: Overly Permissive CORS Policy	0	0	0	1 / 4	1 / 4
HTML5: Overly Permissive Message Posting Policy	0	0	0 / 2	0	0 / 2
HTTP Verb Tampering	0	0	0 / 1	0	0 / 1
Information Discovery: Session Token	0	0	0	0 / 1	0 / 1
Insecure Deployment: HTTP Request Smuggling	0 / 1	0	0	0	0 / 1
Insecure Randomness	0	0 / 5	0	0	0 / 5
Insecure Randomness: Hardcoded Seed	0	0 / 1	0	0	0 / 1
Insecure SSL: Server Identity Verification Disabled	1 / 1	0	0	0	1 / 1
Insecure Transport: HSTS not Set	0	0	0	1 / 1	1 / 1
Insecure Transport: SSLv3/TLS Renegotiation Stream Injection	0	0	1 / 1	0	1 / 1
J2EE Bad Practices: Non-Serializable Object Stored in Session	0	0	0	0 / 2	0 / 2
JSON Injection	1 / 1	4 / 4	0	0	5 / 5
JSON Web Token: Missing Protection Claims	0	0	0	0 / 2	0 / 2
Key Management: Hardcoded Encryption Key	0	0 / 1	0	0	0 / 1
Log Forging	0	0	3 / 3	7 / 7	10 / 10
Log Forging (debug)	0	0	0	24 / 24	24 / 24

Category	Fortify Priority (audited/total)				Total Issues
	Critical	High	Medium	Low	
Mass Assignment: Request Parameters Bound into Persisted Objects	1 / 2	0	0	0	1 / 2
Missing XML Validation	0	0	0	2 / 2	2 / 2
Null Dereference	0	0	0	1 / 1	1 / 1
Often Misused: Boolean.getBoolean()	0	0	0 / 1	0	0 / 1
Often Misused: File Upload	0	0	0 / 2	0	0 / 2
Open Redirect	1 / 1	0	0	0	1 / 1
Password Management: Empty Password in Configuration File	0	0 / 1	0	0	0 / 1
Password Management: Hardcoded Password	0 / 5	0 / 4	0	0	0 / 9
Password Management: Password in Comment	0	0	0	0 / 14	0 / 14
Password Management: Weak Password Policy	0	0	0	0 / 1	0 / 1
Path Manipulation	11 / 11	8 / 8	0	0	19 / 19
Poor Error Handling: Empty Catch Block	0	0	0	0 / 1	0 / 1
Poor Error Handling: Overly Broad Catch	0	0	0	0 / 2	0 / 2
Poor Error Handling: Overly Broad Throws	0	0	0	0 / 3	0 / 3
Poor Error Handling: Unhandled Exception	0	0	0	0 / 2	0 / 2
Poor Style: Value Never Read	0	0	0	0 / 7	0 / 7
Privacy Violation	0	0	0	1 / 1	1 / 1
Privacy Violation: Autocomplete	0	0	0	2 / 36	2 / 36
Race Condition: Singleton Member Field	0	0 / 1	0	0	0 / 1
Resource Injection	0	0	0	2 / 2	2 / 2
Session Management: Easy-to-Guess Session Identifier Name	0	0	0	0 / 1	0 / 1
Session Puzzling: Spring	0	0	0	7 / 14	7 / 14
Spring Boot Misconfiguration: DevTools Enabled	0	0 / 1	0	0	0 / 1
Spring Security Misconfiguration: Lack of Fallback Check	0	0	0	0 / 1	0 / 1
Spring Security Misconfiguration: Overly Permissive Firewall Policy	0	0	0 / 1	0	0 / 1
SQL Injection	3 / 3	5 / 5	0	0 / 6	8 / 14
System Information Leak	0	0	0	0 / 11	0 / 11
System Information Leak: External	0	3 / 3	0	17 / 17	20 / 20
System Information Leak: Internal	0	0	0	24 / 24	24 / 24
Trust Boundary Violation	0	0	0	52 / 52	52 / 52
Unreleased Resource: Files	0	1 / 1	0	0	1 / 1
Unreleased Resource: Streams	0	1 / 1	0	0	1 / 1
Vulnerable OSS	0 / 55	0 / 48	0 / 3	0	0 / 106
Weak Encryption	1 / 3	0	0	0	1 / 3
Weak Encryption: Insecure Mode of Operation	0 / 2	0	0	0	0 / 2
Web Server Misconfiguration: Insecure Content-Type Setting	0	0	0	0 / 1	0 / 1
Web Server Misconfiguration: OPTIONS HTTP Method	0	0	0	1 / 1	1 / 1
Web Server Misconfiguration: Server Error Message	0	0	0	0 / 8	0 / 8
Web Server Misconfiguration: Unprotected Directory	0	0	0	0 / 2	0 / 2
Web Server Misconfiguration: Unprotected File	0	0 / 2	0	0	0 / 2
Web Server Misconfiguration: Weak Authentication	0	0	0	1 / 1	1 / 1
XML Entity Expansion Injection	0	0	1 / 1	4 / 4	5 / 5
XML External Entity Injection	2 / 2	3 / 3	0	0	5 / 5

Results Outline

Access Control: Database (36 issues)

Abstract

Without proper access control, executing a SQL statement that contains a user-controlled primary key can allow an attacker to view unauthorized records.

Explanation

Database access control errors occur when:

1. Data enters a program from an untrusted source.

2. The data is used to specify the value of a primary key in a SQL query.

Example 1: The following code uses a parameterized statement, which escapes metacharacters and prevents SQL injection vulnerabilities, to construct and execute a SQL query that searches for an invoice matching the specified identifier [1]. The identifier is selected from a list of all invoices associated with the current authenticated user.

```
...
id = Integer.decode(request.getParameter("invoiceID"));
String query = "SELECT * FROM invoices WHERE id = ?";
PreparedStatement stmt = conn.prepareStatement(query);
stmt.setInt(1, id);
ResultSet results = stmt.execute();
...
```

The problem is that the developer has failed to consider all of the possible values of `id`. Although the interface generates a list of invoice identifiers that belong to the current user, an attacker might bypass this interface to request any desired invoice. Because the code in this example does not check to ensure that the user has permission to access the requested invoice, it will display any invoice, even if it does not belong to the current user.

Some think that in the mobile world, classic web application vulnerabilities, such as database access control errors, do not make sense -- why would the user attack themself? However, keep in mind that the essence of mobile platforms is applications that are downloaded from various sources and run alongside each other on the same device. The likelihood of running a piece of malware next to a banking application is high, which necessitates expanding the attack surface of mobile applications to include inter-process communication.

Example 2: The following code adapts `Example 1` to the Android platform.

```
...
        String id = this.getIntent().getExtras().getString("invoiceID");
        String query = "SELECT * FROM invoices WHERE id = ?";
        SQLiteDatabase db = this.openOrCreateDatabase("DB", MODE_PRIVATE,
null);
        Cursor c = db.rawQuery(query, new Object[]{id});
...
```

A number of modern web frameworks provide mechanisms to perform user input validation (including Struts and Spring MVC). To highlight the unvalidated sources of input, Fortify Secure Coding Rulepacks dynamically re-prioritize the issues Fortify Static Code Analyzer reports by lowering their probability of exploit and providing pointers to the supporting evidence whenever the framework validation mechanism is in use. We refer to this feature as Context-Sensitive Ranking. To further assist the Fortify user with the auditing process, the Fortify Software Security Research group makes available the Data Validation project template that groups the issues into folders based on the validation mechanism applied to their source of input.

Recommendation

Rather than relying on the presentation layer to restrict values submitted by the user, access control should be handled by the application and database layers. Under no circumstances should a user be allowed to retrieve or modify a row in the database without the appropriate permissions. Every query that accesses the database should enforce this policy, which can often be accomplished by simply including the current authenticated username as part of the query.

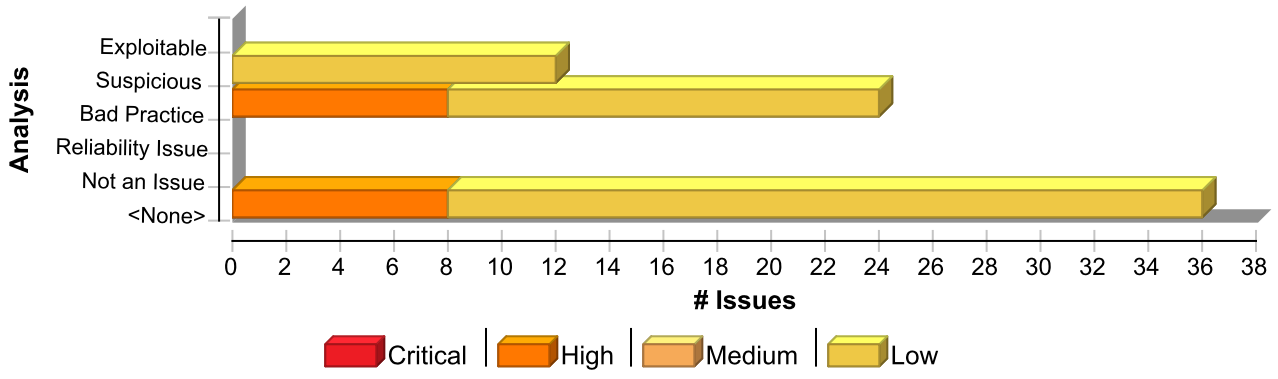
Example 3: The following code implements the same functionality as `Example 1` but imposes an additional constraint to verify that the invoice belongs to the currently authenticated user.

```
...
userName = ctx.getAuthenticatedUserName();
id = Integer.decode(request.getParameter("invoiceID"));
String query =
        "SELECT * FROM invoices WHERE id = ? AND user = ?";
PreparedStatement stmt = conn.prepareStatement(query);
stmt.setInt(1, id);
stmt.setString(2, userName);
ResultSet results = stmt.execute();
...
```

And here is an Android equivalent:

```
...
        PasswordAuthentication pa = authenticator.getPasswordAuthentication();
        String userName = pa.getUserName();
        String id = this.getIntent().getExtras().getString("invoiceID");
        String query = "SELECT * FROM invoices WHERE id = ? AND user = ?";
        SQLiteDatabase db = this.openOrCreateDatabase("DB", MODE_PRIVATE,
null);
        Cursor c = db.rawQuery(query, new Object[]{id, userName});
...
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Access Control: Database	36	0	0	36
Total	**36**	**0**	**0**	**36**

Access Control: Database — High

URL: null

src/main/java/com/microfocus/example/service/ProductService.java, line 271 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.633

Source Details

Source: createReview(0)
From: com.microfocus.example.api.controllers.ApiReviewController.createReview
File: src/main/java/com/microfocus/example/api/controllers/ApiReviewController.java
:129
URL: null

```
126  @PostMapping(value = {""}, produces = {"application/json"}, consumes =
{"application/json"})
127  @ResponseStatus(HttpStatus.CREATED)
128  public ResponseEntity<ReviewResponse> createReview(
129  @io.swagger.v3.oas.annotations.parameters.RequestBody(description = "")
@Valid @RequestBody ReviewRequest newReview) {
130  log.debug("API::Creating new review: " + newReview.toString());
131  return new ResponseEntity<>(new
ReviewResponse(productService.saveReviewFromApi(null, newReview)),
HttpStatus.OK);
132  }
```

Sink Details

Sink: org.springframework.data.repository.CrudRepository.findById()
Enclosing Method: saveReviewFromApi()
File: src/main/java/com/microfocus/example/service/ProductService.java:271
Taint Flags: PRIMARY_KEY, WEB, XSS

```
268  if (optionalProduct.isPresent()) {
269  rtmp.setProduct(optionalProduct.get());
270  }
271  Optional<User> optionalUser = userRepository.findById(review.getUserId());
272  if (optionalUser.isPresent()) {
273  rtmp.setUser(optionalUser.get());
274  }
```

src/main/java/com/microfocus/example/service/ProductService.java, line 357 (Access Control: Database)

Issue Details

Access Control: Database — High

URL: null

src/main/java/com/microfocus/example/service/ProductService.java, line 357 (Access Control: Database)

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.633

Source Details

Source: updateOrder(0)
From: com.microfocus.example.api.controllers.ApiOrderController.updateOrder
File: src/main/java/com/microfocus/example/api/controllers/ApiOrderController.java:144
URL: null

```
141  })
142  @PutMapping(value = {"/{id}"}, produces = {"application/json"}, consumes = {"application/json"})
143  public ResponseEntity<OrderResponse> updateOrder(
144  @io.swagger.v3.oas.annotations.parameters.RequestBody(description = "") @Valid @RequestBody OrderRequest newOrder,
145  @Parameter(description = "UUID of the order to be updated. Cannot be empty.", example = "c9b31f33-17a4-4fcd-927e-c14cdee32201", required = true) @PathVariable("id") UUID id) {
146  log.debug("API::Updating order with UUID: " + id);
147  return new ResponseEntity<>(new OrderResponse(productService.saveOrderFromApi(id, newOrder)), HttpStatus.OK);
```

Sink Details

Sink: org.springframework.data.repository.CrudRepository.findById()
Enclosing Method: saveOrderFromApi()
File: src/main/java/com/microfocus/example/service/ProductService.java:357
Taint Flags: PRIMARY_KEY, WEB, XSS

```
354  otmp.setOrderNum(order.getOrderNum());
355  otmp.setCart(order.getCart());
356  otmp.setAmount(order.getAmount());
357  Optional<User> optionalUser = userRepository.findById(order.getUserId());
358  if (optionalUser.isPresent()) {
359  otmp.setUser(optionalUser.get());
360  }
```

src/main/java/com/microfocus/example/service/ProductService.java, line 357 (Access Control: Database)

Issue Details

Access Control: Database — High

URL: null

src/main/java/com/microfocus/example/service/ProductService.java, line 357 (Access Control: Database)

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.633

Source Details

Source: createOrder(0)
From: com.microfocus.example.api.controllers.ApiOrderController.createOrder
File: src/main/java/com/microfocus/example/api/controllers/ApiOrderController.java:128
URL: null

```
125  @PostMapping(value = {""}, produces = {"application/json"}, consumes = {"application/json"})
126  @ResponseStatus(HttpStatus.CREATED)
127  public ResponseEntity<OrderResponse> createOrder(
128  @io.swagger.v3.oas.annotations.parameters.RequestBody(description = "") @Valid @RequestBody OrderRequest newOrder) {
129  log.debug("API::Creating new order: " + newOrder.toString());
130  return new ResponseEntity<>(new OrderResponse(productService.saveOrderFromApi(null, newOrder)), HttpStatus.OK);
131  }
```

Sink Details

Sink: org.springframework.data.repository.CrudRepository.findById()
Enclosing Method: saveOrderFromApi()
File: src/main/java/com/microfocus/example/service/ProductService.java:357
Taint Flags: PRIMARY_KEY, WEB, XSS

```
354  otmp.setOrderNum(order.getOrderNum());
355  otmp.setCart(order.getCart());
356  otmp.setAmount(order.getAmount());
357  Optional<User> optionalUser = userRepository.findById(order.getUserId());
358  if (optionalUser.isPresent()) {
359  otmp.setUser(optionalUser.get());
360  }
```

src/main/java/com/microfocus/example/service/UserService.java, line 387 (Access Control: Database)

Issue Details

Access Control: Database — High

URL: null

src/main/java/com/microfocus/example/service/UserService.java, line 387 (Access Control: Database)

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.633

Source Details

Source: createMessage(0)
From: com.microfocus.example.api.controllers.ApiMessageController.createMessage
File: src/main/java/com/microfocus/example/api/controllers/ApiMessageController.java:126
URL: null

```
123  @PostMapping(value = {""}, produces = {"application/json"}, consumes = {"application/json"})
124  @ResponseStatus(HttpStatus.CREATED)
125  public ResponseEntity<MessageResponse> createMessage(
126  @io.swagger.v3.oas.annotations.parameters.RequestBody(description = "") @Valid @RequestBody MessageRequest newMessage) {
127  log.debug("API::Creating new message: " + newMessage.toString());
128  return new ResponseEntity<>(new MessageResponse(userService.saveMessageFromApi(null, newMessage)), HttpStatus.OK);
129  }
```

Sink Details

Sink: org.springframework.data.repository.CrudRepository.findById()
Enclosing Method: saveMessageFromApi()
File: src/main/java/com/microfocus/example/service/UserService.java:387
Taint Flags: PRIMARY_KEY, WEB, XSS

```
384
385  public Message saveMessageFromApi(UUID messageId, MessageRequest message) throws MessageNotFoundException, UserNotFoundException {
386  Message mtmp = new Message();
387  Optional<User> optionalUser = userRepository.findById(message.getUserId());
388  if (optionalUser.isPresent()) {
389  mtmp.setUser(optionalUser.get());
390  // are we creating a new message or updating an existing message?
```

src/main/java/com/microfocus/example/service/ProductService.java, line 242 (Access Control: Database)

Issue Details

Access Control: Database — High

URL: null

src/main/java/com/microfocus/example/service/ProductService.java, line 242 (Access Control: Database)

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.532

Source Details

Source: saveReview(0)
From: com.microfocus.example.web.controllers.admin.AdminReviewController.saveReview
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminReviewControl ler.java:101
URL: null

```
98   }
99
100  @PostMapping("/{id}/save")
101  public String saveReview(@Valid @ModelAttribute("adminReviewForm") AdminReviewForm adminReviewForm,
102  BindingResult bindingResult, Model model,
103  RedirectAttributes redirectAttributes,
104  Principal principal) {
```

Sink Details

Sink: org.springframework.data.repository.CrudRepository.findById()
Enclosing Method: saveReviewFromAdminReviewForm()
File: src/main/java/com/microfocus/example/service/ProductService.java:242
Taint Flags: PRIMARY_KEY, WEB, XSS

```
239  }
240
241  public Review saveReviewFromAdminReviewForm(AdminReviewForm adminReviewForm) throws ReviewNotFoundException {
242  Optional<Review> optionalReview = reviewRepository.findById(adminReviewForm.getId());
243  if (optionalReview.isPresent()) {
244  Review rtmp = optionalReview.get();
245  rtmp.setComment(adminReviewForm.getComment());
```

src/main/java/com/microfocus/example/service/ProductService.java, line 271 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Access Control: Database — High

URL: null

src/main/java/com/microfocus/example/service/ProductService.java, line 271 (Access Control: Database)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.633

Source Details

Source: updateReview(0)
From: com.microfocus.example.api.controllers.ApiReviewController.updateReview
File: src/main/java/com/microfocus/example/api/controllers/ApiReviewController.java
:145
URL: null

```
142  })
143  @PutMapping(value = {"/{id}"}, produces = {"application/json"}, consumes
= {"application/json"})
144  public ResponseEntity<ReviewResponse> updateReview(
145  @io.swagger.v3.oas.annotations.parameters.RequestBody(description = "")
@Valid @RequestBody ReviewRequest newReview,
146  @Parameter(description = "UUID of the review to be updated. Cannot be
empty.", example = "822f734a-3d13-4ebc-bff6-9c36d29866a6", required = true)
@PathVariable("id") UUID id) {
147  log.debug("API::Updating review with UUID: " + id);
148  return new ResponseEntity<>(new
ReviewResponse(productService.saveReviewFromApi(id, newReview)),
HttpStatus.OK);
```

Sink Details

Sink: org.springframework.data.repository.CrudRepository.findById()
Enclosing Method: saveReviewFromApi()
File: src/main/java/com/microfocus/example/service/ProductService.java:271
Taint Flags: PRIMARY_KEY, WEB, XSS

```
268  if (optionalProduct.isPresent()) {
269  rtmp.setProduct(optionalProduct.get());
270  }
271  Optional<User> optionalUser = userRepository.findById(review.getUserId());
272  if (optionalUser.isPresent()) {
273  rtmp.setUser(optionalUser.get());
274  }
```

src/main/java/com/microfocus/example/service/ProductService.java, line 322 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Access Control: Database — High

URL: null

src/main/java/com/microfocus/example/service/ProductService.java, line 322 (Access Control: Database)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.532

Source Details

Source: saveOrder(0)
From: com.microfocus.example.web.controllers.admin.AdminOrderController.saveOrder
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminOrderControll
er.java:101
URL: null

```
98   }
99
100  @PostMapping("/{id}/save")
101  public String saveOrder(@Valid @ModelAttribute("adminOrderForm")
AdminOrderForm adminOrderForm,
102  BindingResult bindingResult, Model model,
103  RedirectAttributes redirectAttributes,
104  Principal principal) {
```

Sink Details

Sink: org.springframework.data.repository.CrudRepository.findById()
Enclosing Method: saveOrderFromAdminOrderForm()
File: src/main/java/com/microfocus/example/service/ProductService.java:322
Taint Flags: PRIMARY_KEY, WEB, XSS

```
319  }
320
321  public Order saveOrderFromAdminOrderForm(AdminOrderForm adminOrderForm) throws
OrderNotFoundException {
322  Optional<Order> optionalOrder = orderRepository.findById(adminOrderForm.getId());
323  if (optionalOrder.isPresent()) {
324  Order otmp = optionalOrder.get();
325  otmp.setOrderNum(adminOrderForm.getOrderNum());
```

src/main/java/com/microfocus/example/service/UserService.java, line 387 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)

Access Control: Database — High

URL: null

src/main/java/com/microfocus/example/service/UserService.java, line 387 (Access Control: Database)

AA_Confidence 0.633

Source Details

Source: updateMessage(0)
From: com.microfocus.example.api.controllers.ApiMessageController.updateMessage
File: src/main/java/com/microfocus/example/api/controllers/ApiMessageController.jav
a:142
URL: null

```
139  })
140  @PutMapping(value = {"/{id}"}, produces = {"application/json"}, consumes
= {"application/json"})
141  public ResponseEntity<MessageResponse> updateMessage(
142  @io.swagger.v3.oas.annotations.parameters.RequestBody(description = "")
@Valid @RequestBody MessageRequest newMessage,
143  @Parameter(description = "UUID of the message to be updated. Cannot be
empty.", example = "6914e47d-2f0a-4deb-a712-12e7801e13e8", required = true)
@PathVariable("id") UUID id) {
144  log.debug("API::Updating message with UUID: " + id);
145  return new ResponseEntity<>(new
MessageResponse(userService.saveMessageFromApi(id, newMessage)),
HttpStatus.OK);
```

Sink Details

Sink: org.springframework.data.repository.CrudRepository.findById()
Enclosing Method: saveMessageFromApi()
File: src/main/java/com/microfocus/example/service/UserService.java:387
Taint Flags: PRIMARY_KEY, WEB, XSS

```
384
385  public Message saveMessageFromApi(UUID messageId, MessageRequest message) throws
MessageNotFoundException, UserNotFoundException {
386  Message mtmp = new Message();
387  Optional<User> optionalUser = userRepository.findById(message.getUserId());
388  if (optionalUser.isPresent()) {
389  mtmp.setUser(optionalUser.get());
390  // are we creating a new message or updating an existing message?
```

Access Control: Database — Low

Package: com.microfocus.example.repository

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 110 (Access Control: Database)

Issue Details

Access Control: Database — Low

Package: com.microfocus.example.repository

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 110 (Access Control: Database)

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.505

Source Details

Source: Read this.defaultPageSize
From: com.microfocus.example.web.controllers.ProductController.firstaid
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:96

```
93  @GetMapping("/firstaid")
94  public String firstaid(Model model, @Param("keywords") String keywords, @Param("limit") Integer limit, Principal principal) {
95  log.debug("Searching for products using keywords: " + ((keywords == null || keywords.isEmpty()) ? "none" : keywords));
96  productService.setPageSize((limit == null ? defaultPageSize : limit));
97  List<Product> products = productService.getAllActiveProducts(0, keywords);
98  model.addAttribute("keywords", keywords);
99  model.addAttribute("products", products);
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findByKeywords()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:110
Taint Flags: ARGS, ENVIRONMENT, NUMBER, PROPERTY

```
107  " OR lower(summary) LIKE '%" + query + "%'" +
108  " OR lower(description) LIKE '%" + query + "%'" +
109  " LIMIT " + limit + " OFFSET " + offset;
110  return jdbcTemplate.query(sqlQuery, new ProductMapper());
111  }
112
113  public List<Product> findByKeywordsFromProductName(String keywords) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 128 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Access Control: Database — Low

Package: com.microfocus.example.repository

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 128 (Access Control: Database)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.549

Source Details

Source: Read this.pageSize
From: com.microfocus.example.service.ProductService.getPageSize
File: src/main/java/com/microfocus/example/service/ProductService.java:77

```
74  private Integer pageSize;
75
76  public Integer getPageSize() {
77  return pageSize;
78  }
79
80  public void setPageSize(Integer pageSize) {
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findAvailableByKeywords()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:128
Taint Flags: ARGS, ENVIRONMENT, NUMBER, PROPERTY

```
125  " OR lower(description) LIKE '%" + query + "%'" +
126  " AND available = true " +
127  " LIMIT " + limit + " OFFSET " + offset;
128  return jdbcTemplate.query(sqlQuery, new ProductMapper());
129  }
130
131  public List<Product> findAvailableByKeywordsFromProductName(String keywords) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 128 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.505

Source Details

Access Control: Database — Low

Package: com.microfocus.example.repository

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 128 (Access Control: Database)

Source: Read this.defaultPageSize
From: com.microfocus.example.web.controllers.ProductController.index
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:109

```
106  @GetMapping(value = {"", "/"})
107  public String index(Model model, @Param("keywords") String keywords, @Param("limit") Integer limit, Principal principal) {
108  log.debug("Searching for products using keywords: " + ((keywords == null || keywords.isEmpty()) ? "none" : keywords));
109  productService.setPageSize((limit == null ? defaultPageSize : limit));
110  List<Product> products = productService.getAllActiveProducts(0, keywords);
111  model.addAttribute("keywords", keywords);
112  model.addAttribute("products", products);
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findAvailableByKeywords()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:128
Taint Flags: ARGS, ENVIRONMENT, NUMBER, PROPERTY

```
125  " OR lower(description) LIKE '%" + query + "%'" +
126  " AND available = true " +
127  " LIMIT " + limit + " OFFSET " + offset;
128  return jdbcTemplate.query(sqlQuery, new ProductMapper());
129  }
130
131  public List<Product> findAvailableByKeywordsFromProductName(String keywords) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 55 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.505

Source Details

Access Control: Database — Low

Package: com.microfocus.example.repository

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 55 (Access Control: Database)

Source: Read this.defaultPageSize
From: com.microfocus.example.web.controllers.ProductController.firstaid
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:96

```
93  @GetMapping("/firstaid")
94  public String firstaid(Model model, @Param("keywords") String keywords, @Param("limit") Integer limit, Principal principal) {
95  log.debug("Searching for products using keywords: " + ((keywords == null || keywords.isEmpty()) ? "none" : keywords));
96  productService.setPageSize((limit == null ? defaultPageSize : limit));
97  List<Product> products = productService.getAllActiveProducts(0, keywords);
98  model.addAttribute("keywords", keywords);
99  model.addAttribute("products", products);
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findAll()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:55
Taint Flags: ARGS, ENVIRONMENT, NUMBER, PROPERTY

```
52  public List<Product> findAll(int offset, int limit) {
53  String sqlQuery = "select * from products" +
54  " LIMIT " + limit + " OFFSET " + offset;
55  return jdbcTemplate.query(sqlQuery, new ProductMapper());
56  }
57
58  public List<Product> findAvailable(int offset, int limit) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 62 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.505

Source Details

Source: Read this.defaultPageSize
From: com.microfocus.example.web.controllers.ProductController.firstaid
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:96

Access Control: Database — Low

Package: com.microfocus.example.repository

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 62 (Access Control: Database)

```
93  @GetMapping("/firstaid")
94  public String firstaid(Model model, @Param("keywords") String keywords,
@Param("limit") Integer limit, Principal principal) {
95  log.debug("Searching for products using keywords: " + ((keywords == null
|| keywords.isEmpty()) ? "none" : keywords));
96  productService.setPageSize((limit == null ? defaultPageSize : limit));
97  List<Product> products = productService.getAllActiveProducts(0, keywords);
98  model.addAttribute("keywords", keywords);
99  model.addAttribute("products", products);
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findAvailable()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:62
Taint Flags: ARGS, ENVIRONMENT, NUMBER, PROPERTY

```
59  String sqlQuery = "select * from products" +
60  " where available = true " +
61  " LIMIT " + limit + " OFFSET " + offset;
62  return jdbcTemplate.query(sqlQuery, new ProductMapper());
63  }
64
65  public Optional<Product> findById(UUID id) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 62 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.549

Source Details

Source: Read this.pageSize
From: com.microfocus.example.service.ProductService.getPageSize
File: src/main/java/com/microfocus/example/service/ProductService.java:77

Access Control: Database — Low

Package: com.microfocus.example.repository

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 62 (Access Control: Database)

```
74  private Integer pageSize;
75
76  public Integer getPageSize() {
77  return pageSize;
78  }
79
80  public void setPageSize(Integer pageSize) {
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findAvailable()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:62
Taint Flags: ARGS, ENVIRONMENT, NUMBER, PROPERTY

```
59  String sqlQuery = "select * from products" +
60  " where available = true " +
61  " LIMIT " + limit + " OFFSET " + offset;
62  return jdbcTemplate.query(sqlQuery, new ProductMapper());
63  }
64
65  public Optional<Product> findById(UUID id) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 55 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.549

Source Details

Source: Read this.pageSize
From: com.microfocus.example.service.ProductService.getPageSize
File: src/main/java/com/microfocus/example/service/ProductService.java:77

Access Control: Database — Low

Package: com.microfocus.example.repository

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 55 (Access Control: Database)

```
74  private Integer pageSize;
75
76  public Integer getPageSize() {
77  return pageSize;
78  }
79
80  public void setPageSize(Integer pageSize) {
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findAll()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:55
Taint Flags: ARGS, ENVIRONMENT, NUMBER, PROPERTY

```
52  public List<Product> findAll(int offset, int limit) {
53  String sqlQuery = "select * from products" +
54  " LIMIT " + limit + " OFFSET " + offset;
55  return jdbcTemplate.query(sqlQuery, new ProductMapper());
56  }
57
58  public List<Product> findAvailable(int offset, int limit) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 128 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.505

Source Details

Source: Read this.defaultPageSize
From: com.microfocus.example.web.controllers.ProductController.firstaid
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:96

Access Control: Database — Low

Package: com.microfocus.example.repository

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 128 (Access Control: Database)

```
93  @GetMapping("/firstaid")
94  public String firstaid(Model model, @Param("keywords") String keywords,
@Param("limit") Integer limit, Principal principal) {
95  log.debug("Searching for products using keywords: " + ((keywords == null
|| keywords.isEmpty()) ? "none" : keywords));
96  productService.setPageSize((limit == null ? defaultPageSize : limit));
97  List<Product> products = productService.getAllActiveProducts(0, keywords);
98  model.addAttribute("keywords", keywords);
99  model.addAttribute("products", products);
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findAvailableByKeywords()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:128
Taint Flags: ARGS, ENVIRONMENT, NUMBER, PROPERTY

```
125  " OR lower(description) LIKE '%" + query + "%'" +
126  " AND available = true " +
127  " LIMIT " + limit + " OFFSET " + offset;
128  return jdbcTemplate.query(sqlQuery, new ProductMapper());
129  }
130
131  public List<Product> findAvailableByKeywordsFromProductName(String keywords) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 62 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.505

Source Details

Source: Read this.defaultPageSize
From: com.microfocus.example.web.controllers.ProductController.index
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:109

Access Control: Database — Low

Package: com.microfocus.example.repository

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 62 (Access Control: Database)

```
106  @GetMapping(value = {"", "/"})
107  public String index(Model model, @Param("keywords") String keywords,
@Param("limit") Integer limit, Principal principal) {
108  log.debug("Searching for products using keywords: " + ((keywords == null
|| keywords.isEmpty()) ? "none" : keywords));
109  productService.setPageSize((limit == null ? defaultPageSize : limit));
110  List<Product> products = productService.getAllActiveProducts(0,
keywords);
111  model.addAttribute("keywords", keywords);
112  model.addAttribute("products", products);
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findAvailable()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:62
Taint Flags: ARGS, ENVIRONMENT, NUMBER, PROPERTY

```
59  String sqlQuery = "select * from products" +
60  " where available = true " +
61  " LIMIT " + limit + " OFFSET " + offset;
62  return jdbcTemplate.query(sqlQuery, new ProductMapper());
63  }
64
65  public Optional<Product> findById(UUID id) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 55 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.505

Source Details

Source: Read this.defaultPageSize
From: com.microfocus.example.web.controllers.ProductController.index
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:109

Access Control: Database — Low

Package: com.microfocus.example.repository

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 55 (Access Control: Database)

```
106  @GetMapping(value = {"", "/"})
107  public String index(Model model, @Param("keywords") String keywords,
@Param("limit") Integer limit, Principal principal) {
108  log.debug("Searching for products using keywords: " + ((keywords == null
|| keywords.isEmpty()) ? "none" : keywords));
109  productService.setPageSize((limit == null ? defaultPageSize : limit));
110  List<Product> products = productService.getAllActiveProducts(0,
keywords);
111  model.addAttribute("keywords", keywords);
112  model.addAttribute("products", products);
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findAll()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:55
Taint Flags: ARGS, ENVIRONMENT, NUMBER, PROPERTY

```
52  public List<Product> findAll(int offset, int limit) {
53  String sqlQuery = "select * from products" +
54  " LIMIT " + limit + " OFFSET " + offset;
55  return jdbcTemplate.query(sqlQuery, new ProductMapper());
56  }
57
58  public List<Product> findAvailable(int offset, int limit) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 110 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.549

Source Details

Source: Read this.pageSize
From: com.microfocus.example.service.ProductService.getPageSize
File: src/main/java/com/microfocus/example/service/ProductService.java:77

Access Control: Database — Low

Package: com.microfocus.example.repository

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 110 (Access Control: Database)

```
74  private Integer pageSize;
75
76  public Integer getPageSize() {
77  return pageSize;
78  }
79
80  public void setPageSize(Integer pageSize) {
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findByKeywords()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:110
Taint Flags: ARGS, ENVIRONMENT, NUMBER, PROPERTY

```
107  " OR lower(summary) LIKE '%" + query + "%'" +
108  " OR lower(description) LIKE '%" + query + "%'" +
109  " LIMIT " + limit + " OFFSET " + offset;
110  return jdbcTemplate.query(sqlQuery, new ProductMapper());
111  }
112
113  public List<Product> findByKeywordsFromProductName(String keywords) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 110 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.505

Source Details

Source: Read this.defaultPageSize
From: com.microfocus.example.web.controllers.ProductController.index
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:109

Access Control: Database — Low

Package: com.microfocus.example.repository

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 110 (Access Control: Database)

```
106  @GetMapping(value = {"", "/"})
107  public String index(Model model, @Param("keywords") String keywords,
@Param("limit") Integer limit, Principal principal) {
108  log.debug("Searching for products using keywords: " + ((keywords == null
|| keywords.isEmpty()) ? "none" : keywords));
109  productService.setPageSize((limit == null ? defaultPageSize : limit));
110  List<Product> products = productService.getAllActiveProducts(0,
keywords);
111  model.addAttribute("keywords", keywords);
112  model.addAttribute("products", products);
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findByKeywords()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:110
Taint Flags: ARGS, ENVIRONMENT, NUMBER, PROPERTY

```
107  " OR lower(summary) LIKE '%" + query + "%'" +
108  " OR lower(description) LIKE '%" + query + "%'" +
109  " LIMIT " + limit + " OFFSET " + offset;
110  return jdbcTemplate.query(sqlQuery, new ProductMapper());
111  }
112
113  public List<Product> findByKeywordsFromProductName(String keywords) {
```

URL: null

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 110 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.608

Source Details

Source: index(2)
From: com.microfocus.example.web.controllers.ProductController.index
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:107
URL: null

Access Control: Database — Low

URL: null

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 110 (Access Control: Database)

```
104  }
105
106  @GetMapping(value = {"", "/"})
107  public String index(Model model, @Param("keywords") String keywords,
@Param("limit") Integer limit, Principal principal) {
108  log.debug("Searching for products using keywords: " + ((keywords == null
|| keywords.isEmpty()) ? "none" : keywords));
109  productService.setPageSize((limit == null ? defaultPageSize : limit));
110  List<Product> products = productService.getAllActiveProducts(0,
keywords);
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findByKeywords()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:110
Taint Flags: NUMBER, WEB

```
107  " OR lower(summary) LIKE '%" + query + "%'" +
108  " OR lower(description) LIKE '%" + query + "%'" +
109  " LIMIT " + limit + " OFFSET " + offset;
110  return jdbcTemplate.query(sqlQuery, new ProductMapper());
111  }
112
113  public List<Product> findByKeywordsFromProductName(String keywords) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 55 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.607

Source Details

Source: getReviewsByKeywords(3)
From: com.microfocus.example.api.controllers.ApiReviewController.getReviewsByKeywords
File: src/main/java/com/microfocus/example/api/controllers/ApiReviewController.java:99
URL: null

Access Control: Database — Low

URL: null

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 55 (Access Control: Database)

```
96  @Parameter(description = "UUID of the product to find reviews for.", example = "eec467c8-5de9-4c7c-8541-7b31614d31a0") @RequestParam("pid") Optional<UUID> pid,
97  @Parameter(description = "Keyword(s) search for reviews to be found.") @RequestParam("keywords") Optional<String> keywords,
98  @Parameter(description = "Offset of the starting record. 0 indicates the first record.") @RequestParam("offset") Optional<Integer> offset,
99  @Parameter(description = "Maximum records to return. The maximum value allowed is 50.") @RequestParam("limit") Optional<Integer> limit) {
100  log.debug("API::Retrieving reviews by keyword(s)" + (pid.map(value -> " for product id:" + value).orElse("")));
101  if (limit.isPresent()) {
102  productService.setPageSize(limit.get());
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findAll()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:55
Taint Flags: NUMBER, WEB

```
52  public List<Product> findAll(int offset, int limit) {
53  String sqlQuery = "select * from products" +
54  " LIMIT " + limit + " OFFSET " + offset;
55  return jdbcTemplate.query(sqlQuery, new ProductMapper());
56  }
57
58  public List<Product> findAvailable(int offset, int limit) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 55 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.608

Source Details

Access Control: Database — Low

URL: null

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 55 (Access Control: Database)

Source: firstaid(2)
From: com.microfocus.example.web.controllers.ProductController.firstaid
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:94
URL: null

```
91   }
92
93   @GetMapping("/firstaid")
94   public String firstaid(Model model, @Param("keywords") String keywords, @Param("limit") Integer limit, Principal principal) {
95   log.debug("Searching for products using keywords: " + ((keywords == null || keywords.isEmpty()) ? "none" : keywords));
96   productService.setPageSize((limit == null ? defaultPageSize : limit));
97   List<Product> products = productService.getAllActiveProducts(0, keywords);
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findAll()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:55
Taint Flags: NUMBER, WEB

```
52  public List<Product> findAll(int offset, int limit) {
53  String sqlQuery = "select * from products" +
54  " LIMIT " + limit + " OFFSET " + offset;
55  return jdbcTemplate.query(sqlQuery, new ProductMapper());
56  }
57
58  public List<Product> findAvailable(int offset, int limit) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 55 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.642

Source Details

Access Control: Database — Low

URL: null

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 55 (Access Control: Database)

Source: getProductsByKeywords(1)
From: com.microfocus.example.api.controllers.ApiProductController.getProductsByKeywords
File: src/main/java/com/microfocus/example/api/controllers/ApiProductController.java:76
URL: null

```
73  @GetMapping(value = {""}, produces = {"application/json"})
74  public ResponseEntity<List<ProductResponse>> getProductsByKeywords(
75  @Parameter(description = "Keyword(s) search for products to be found.") @RequestParam("keywords") Optional<String> keywords,
76  @Parameter(description = "Offset of the starting record. 0 indicates the first record.") @RequestParam("offset") Optional<Integer> offset,
77  @Parameter(description = "Maximum records to return. The maximum value allowed is 50.") @RequestParam("limit") Optional<Integer> limit) {
78  log.debug("API::Retrieving products by keyword(s)");
79  if (limit.isPresent()) {
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findAll()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:55
Taint Flags: NUMBER, WEB

```
52  public List<Product> findAll(int offset, int limit) {
53  String sqlQuery = "select * from products" +
54  " LIMIT " + limit + " OFFSET " + offset;
55  return jdbcTemplate.query(sqlQuery, new ProductMapper());
56  }
57
58  public List<Product> findAvailable(int offset, int limit) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 110 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.607

Source Details

Access Control: Database — Low

URL: null

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 110 (Access Control: Database)

Source: getReviewsByKeywords(3)
From: com.microfocus.example.api.controllers.ApiReviewController.getReviewsByKeywords
File: src/main/java/com/microfocus/example/api/controllers/ApiReviewController.java:99
URL: null

```
96  @Parameter(description = "UUID of the product to find reviews for.", example = "eec467c8-5de9-4c7c-8541-7b31614d31a0") @RequestParam("pid") Optional<UUID> pid,
97  @Parameter(description = "Keyword(s) search for reviews to be found.") @RequestParam("keywords") Optional<String> keywords,
98  @Parameter(description = "Offset of the starting record. 0 indicates the first record.") @RequestParam("offset") Optional<Integer> offset,
99  @Parameter(description = "Maximum records to return. The maximum value allowed is 50.") @RequestParam("limit") Optional<Integer> limit) {
100  log.debug("API::Retrieving reviews by keyword(s)" + (pid.map(value -> " for product id:" + value).orElse("")));
101  if (limit.isPresent()) {
102  productService.setPageSize(limit.get());
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findByKeywords()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:110
Taint Flags: NUMBER, WEB

```
107  " OR lower(summary) LIKE '%" + query + "%'" +
108  " OR lower(description) LIKE '%" + query + "%'" +
109  " LIMIT " + limit + " OFFSET " + offset;
110  return jdbcTemplate.query(sqlQuery, new ProductMapper());
111  }
112
113  public List<Product> findByKeywordsFromProductName(String keywords) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 110 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.608

Access Control: Database — Low

URL: null

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 110 (Access Control: Database)

Source Details

Source: firstaid(2)
From: com.microfocus.example.web.controllers.ProductController.firstaid
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:94
URL: null

```
91  }
92
93  @GetMapping("/firstaid")
94  public String firstaid(Model model, @Param("keywords") String keywords, @Param("limit") Integer limit, Principal principal) {
95  log.debug("Searching for products using keywords: " + ((keywords == null || keywords.isEmpty()) ? "none" : keywords));
96  productService.setPageSize((limit == null ? defaultPageSize : limit));
97  List<Product> products = productService.getAllActiveProducts(0, keywords);
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findByKeywords()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:110
Taint Flags: NUMBER, WEB

```
107  " OR lower(summary) LIKE '%" + query + "%'" +
108  " OR lower(description) LIKE '%" + query + "%'" +
109  " LIMIT " + limit + " OFFSET " + offset;
110  return jdbcTemplate.query(sqlQuery, new ProductMapper());
111  }
112
113  public List<Product> findByKeywordsFromProductName(String keywords) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 55 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.608

Source Details

Access Control: Database — Low

URL: null

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 55 (Access Control: Database)

Source: index(2)
From: com.microfocus.example.web.controllers.ProductController.index
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:107
URL: null

```
104  }
105
106  @GetMapping(value = {"", "/"})
107  public String index(Model model, @Param("keywords") String keywords, @Param("limit") Integer limit, Principal principal) {
108  log.debug("Searching for products using keywords: " + ((keywords == null || keywords.isEmpty()) ? "none" : keywords));
109  productService.setPageSize((limit == null ? defaultPageSize : limit));
110  List<Product> products = productService.getAllActiveProducts(0, keywords);
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findAll()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:55
Taint Flags: NUMBER, WEB

```
52  public List<Product> findAll(int offset, int limit) {
53  String sqlQuery = "select * from products" +
54  " LIMIT " + limit + " OFFSET " + offset;
55  return jdbcTemplate.query(sqlQuery, new ProductMapper());
56  }
57
58  public List<Product> findAvailable(int offset, int limit) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 62 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.607

Source Details

Access Control: Database — Low

URL: null

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 62 (Access Control: Database)

Source: getProductsByKeywords(2)
From: com.microfocus.example.api.controllers.ApiProductController.getProductsByKeywords
File: src/main/java/com/microfocus/example/api/controllers/ApiProductController.java:77
URL: null

```
74  public ResponseEntity<List<ProductResponse>> getProductsByKeywords(
75  @Parameter(description = "Keyword(s) search for products to be found.") @RequestParam("keywords") Optional<String> keywords,
76  @Parameter(description = "Offset of the starting record. 0 indicates the first record.") @RequestParam("offset") Optional<Integer> offset,
77  @Parameter(description = "Maximum records to return. The maximum value allowed is 50.") @RequestParam("limit") Optional<Integer> limit) {
78  log.debug("API::Retrieving products by keyword(s)");
79  if (limit.isPresent()) {
80  productService.setPageSize(limit.orElse(productService.getPageSize()));
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findAvailable()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:62
Taint Flags: NUMBER, WEB

```
59  String sqlQuery = "select * from products" +
60  " where available = true " +
61  " LIMIT " + limit + " OFFSET " + offset;
62  return jdbcTemplate.query(sqlQuery, new ProductMapper());
63  }
64
65  public Optional<Product> findById(UUID id) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 128 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.607

Source Details

Access Control: Database — Low

URL: null

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 128 (Access Control: Database)

Source: getProductsByKeywords(2)
From: com.microfocus.example.api.controllers.ApiProductController.getProductsByKeywords
File: src/main/java/com/microfocus/example/api/controllers/ApiProductController.java:77
URL: null

```
74  public ResponseEntity<List<ProductResponse>> getProductsByKeywords(
75  @Parameter(description = "Keyword(s) search for products to be found.") @RequestParam("keywords") Optional<String> keywords,
76  @Parameter(description = "Offset of the starting record. 0 indicates the first record.") @RequestParam("offset") Optional<Integer> offset,
77  @Parameter(description = "Maximum records to return. The maximum value allowed is 50.") @RequestParam("limit") Optional<Integer> limit) {
78  log.debug("API::Retrieving products by keyword(s)");
79  if (limit.isPresent()) {
80  productService.setPageSize(limit.orElse(productService.getPageSize()));
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findAvailableByKeywords()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:128
Taint Flags: NUMBER, WEB

```
125  " OR lower(description) LIKE '%" + query + "%'" +
126  " AND available = true " +
127  " LIMIT " + limit + " OFFSET " + offset;
128  return jdbcTemplate.query(sqlQuery, new ProductMapper());
129  }
130
131  public List<Product> findAvailableByKeywordsFromProductName(String keywords) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 62 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.608

Source Details

Access Control: Database — Low

URL: null

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 62 (Access Control: Database)

Source: index(2)
From: com.microfocus.example.web.controllers.ProductController.index
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:107
URL: null

```
104  }
105
106  @GetMapping(value = {"", "/"})
107  public String index(Model model, @Param("keywords") String keywords, @Param("limit") Integer limit, Principal principal) {
108  log.debug("Searching for products using keywords: " + ((keywords == null || keywords.isEmpty()) ? "none" : keywords));
109  productService.setPageSize((limit == null ? defaultPageSize : limit));
110  List<Product> products = productService.getAllActiveProducts(0, keywords);
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findAvailable()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:62
Taint Flags: NUMBER, WEB

```
59  String sqlQuery = "select * from products" +
60  " where available = true " +
61  " LIMIT " + limit + " OFFSET " + offset;
62  return jdbcTemplate.query(sqlQuery, new ProductMapper());
63  }
64
65  public Optional<Product> findById(UUID id) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 128 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.607

Source Details

Access Control: Database — Low

URL: null

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 128 (Access Control: Database)

Source: getReviewsByKeywords(3)
From: com.microfocus.example.api.controllers.ApiReviewController.getReviewsByKeywords
File: src/main/java/com/microfocus/example/api/controllers/ApiReviewController.java:99
URL: null

```
96  @Parameter(description = "UUID of the product to find reviews for.", example = "eec467c8-5de9-4c7c-8541-7b31614d31a0") @RequestParam("pid") Optional<UUID> pid,
97  @Parameter(description = "Keyword(s) search for reviews to be found.") @RequestParam("keywords") Optional<String> keywords,
98  @Parameter(description = "Offset of the starting record. 0 indicates the first record.") @RequestParam("offset") Optional<Integer> offset,
99  @Parameter(description = "Maximum records to return. The maximum value allowed is 50.") @RequestParam("limit") Optional<Integer> limit) {
100  log.debug("API::Retrieving reviews by keyword(s)" + (pid.map(value -> " for product id:" + value).orElse("")));
101  if (limit.isPresent()) {
102  productService.setPageSize(limit.get());
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findAvailableByKeywords()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:128
Taint Flags: NUMBER, WEB

```
125  " OR lower(description) LIKE '%" + query + "%'" +
126  " AND available = true " +
127  " LIMIT " + limit + " OFFSET " + offset;
128  return jdbcTemplate.query(sqlQuery, new ProductMapper());
129  }
130
131  public List<Product> findAvailableByKeywordsFromProductName(String keywords) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 62 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.608

Access Control: Database — Low

URL: null

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 62 (Access Control: Database)

Source Details

Source: firstaid(2)
From: com.microfocus.example.web.controllers.ProductController.firstaid
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:94
URL: null

```
91   }
92
93   @GetMapping("/firstaid")
94   public String firstaid(Model model, @Param("keywords") String keywords, @Param("limit") Integer limit, Principal principal) {
95   log.debug("Searching for products using keywords: " + ((keywords == null || keywords.isEmpty()) ? "none" : keywords));
96   productService.setPageSize((limit == null ? defaultPageSize : limit));
97   List<Product> products = productService.getAllActiveProducts(0, keywords);
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findAvailable()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:62
Taint Flags: NUMBER, WEB

```
59   String sqlQuery = "select * from products" +
60   " where available = true " +
61   " LIMIT " + limit + " OFFSET " + offset;
62   return jdbcTemplate.query(sqlQuery, new ProductMapper());
63   }
64
65   public Optional<Product> findById(UUID id) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 55 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.607

Source Details

Access Control: Database — Low

URL: null

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 55 (Access Control: Database)

Source: getProductsByKeywords(2)
From: com.microfocus.example.api.controllers.ApiProductController.getProductsByKeywords
File: src/main/java/com/microfocus/example/api/controllers/ApiProductController.java:77
URL: null

```
74  public ResponseEntity<List<ProductResponse>> getProductsByKeywords(
75  @Parameter(description = "Keyword(s) search for products to be found.") @RequestParam("keywords") Optional<String> keywords,
76  @Parameter(description = "Offset of the starting record. 0 indicates the first record.") @RequestParam("offset") Optional<Integer> offset,
77  @Parameter(description = "Maximum records to return. The maximum value allowed is 50.") @RequestParam("limit") Optional<Integer> limit) {
78  log.debug("API::Retrieving products by keyword(s)");
79  if (limit.isPresent()) {
80  productService.setPageSize(limit.orElse(productService.getPageSize()));
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findAll()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:55
Taint Flags: NUMBER, WEB

```
52  public List<Product> findAll(int offset, int limit) {
53  String sqlQuery = "select * from products" +
54  " LIMIT " + limit + " OFFSET " + offset;
55  return jdbcTemplate.query(sqlQuery, new ProductMapper());
56  }
57
58  public List<Product> findAvailable(int offset, int limit) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 62 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.607

Source Details

Access Control: Database — Low

URL: null

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 62 (Access Control: Database)

Source: getReviewsByKeywords(3)
From: com.microfocus.example.api.controllers.ApiReviewController.getReviewsByKeywords
File: src/main/java/com/microfocus/example/api/controllers/ApiReviewController.java:99
URL: null

```
96  @Parameter(description = "UUID of the product to find reviews for.", example = "eec467c8-5de9-4c7c-8541-7b31614d31a0") @RequestParam("pid") Optional<UUID> pid,
97  @Parameter(description = "Keyword(s) search for reviews to be found.") @RequestParam("keywords") Optional<String> keywords,
98  @Parameter(description = "Offset of the starting record. 0 indicates the first record.") @RequestParam("offset") Optional<Integer> offset,
99  @Parameter(description = "Maximum records to return. The maximum value allowed is 50.") @RequestParam("limit") Optional<Integer> limit) {
100  log.debug("API::Retrieving reviews by keyword(s)" + (pid.map(value -> " for product id:" + value).orElse("")));
101  if (limit.isPresent()) {
102  productService.setPageSize(limit.get());
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findAvailable()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:62
Taint Flags: NUMBER, WEB

```
59  String sqlQuery = "select * from products" +
60  " where available = true " +
61  " LIMIT " + limit + " OFFSET " + offset;
62  return jdbcTemplate.query(sqlQuery, new ProductMapper());
63  }
64
65  public Optional<Product> findById(UUID id) {
```

src/main/java/com/microfocus/example/service/UserService.java, line 346 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.503

Access Control: Database — Low

URL: null

src/main/java/com/microfocus/example/service/UserService.java, line 346 (Access Control: Database)

Source Details

Source: deleteRole(0)
From: com.microfocus.example.api.controllers.ApiRoleController.deleteRole
File: src/main/java/com/microfocus/example/api/controllers/ApiRoleController.java:148
URL: null

```
145  })
146  @DeleteMapping (value = {"/{id}"})
147  public ResponseEntity<ApiStatusResponse> deleteRole(
148  @Parameter(description = "UUID of the role to be updated. Cannot be empty.", example = "6bdd6188-d659-4390-8d37-8f090d2ed69a", required = true) @PathVariable("id") Integer id) {
149  log.debug("API::Deleting role with UUID: " + id);
150  roleService.deleteRoleById(id);
151  ApiStatusResponse apiStatusResponse = new ApiStatusResponse
```

Sink Details

Sink: org.springframework.data.repository.CrudRepository.deleteById()
Enclosing Method: deleteRoleById()
File: src/main/java/com/microfocus/example/service/UserService.java:346
Taint Flags: NUMBER, WEB

```
343  }
344
345  public void deleteRoleById(Integer id) {
346  roleRepository.deleteById(id);
347  }
348
349  //
```

src/main/java/com/microfocus/example/service/UserService.java, line 338 (Access Control: Database)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.503

Source Details

Access Control: Database — Low

URL: null

src/main/java/com/microfocus/example/service/UserService.java, line 338 (Access Control: Database)

Source: findRoleById(0)
From: com.microfocus.example.api.controllers.ApiRoleController.findRoleById
File: src/main/java/com/microfocus/example/api/controllers/ApiRoleController.java:95
URL: null

```
92  })
93  @GetMapping(value = {"/{id}"}, produces = {"application/json"})
94  public ResponseEntity<Authority> findRoleById(
95  @Parameter(description = "UUID of the role to be found. Cannot be empty.", example = "6bdd6188-d659-4390-8d37-8f090d2ed69a", required = true) @PathVariable("id") Integer id) {
96  log.debug("API::Retrieving role with UUID: " + id);
97  if (!roleService.roleExistsById(id))
98  throw new RoleNotFoundException("Role with UUID: " + id.toString() + " does not exist.");
```

Sink Details

Sink: org.springframework.data.repository.CrudRepository.findById()
Enclosing Method: findRoleById()
File: src/main/java/com/microfocus/example/service/UserService.java:338
Taint Flags: NUMBER, WEB

```
335  }
336
337  public Optional<Authority> findRoleById(Integer id) {
338  return roleRepository.findById(id);
339  }
340
341  public Authority saveRole(Authority role) {
```

Build Misconfiguration: External Maven Dependency Repository (1 issue)

Abstract

This maven build script relies on external sources, which could allow an attacker to insert malicious code into the final product or to take control of the build machine.

Explanation

Several tools exist within the Java development world to aid in dependency management: both Apache Ant and Apache Maven build systems include functionality specifically designed to help manage dependencies and Apache Ivy is developed explicitly as a dependency manager. Although there are differences in their behavior, these tools share the common functionality that they automatically download external dependencies specified in the build process at build time. This makes it much easier for developer B to build software in the same manner as developer A. Developers just store dependency information in the build file, which means that each developer and build engineer has a consistent way to obtain dependencies, compile the code, and deploy without the dependency management hassles involved in manual dependency management. The following examples illustrate how Ivy, Ant, and Maven can be used to manage external dependencies as part of a build process.

Under Maven, instead of listing explicit URLs from which to retrieve the dependencies, developers specify the dependency names and versions and Maven relies on its underlying configuration to identify the server(s) from which to retrieve the dependencies. For commonly used components this saves the developer from having to researching dependency locations.

Example 1: The following excerpt from a Maven pom.xml file shows how a developer can specify multiple external dependencies using their name and version:

```
<dependencies>
  <dependency>
    <groupId>commons-logging</groupId>
    <artifactId>commons-logging</artifactId>
    <version>1.1</version>
  </dependency>
  <dependency>
    <groupId>javax.jms</groupId>
    <artifactId>jms</artifactId>
    <version>1.1</version>
  </dependency>
  ...
</dependencies>
```

Two distinct types of attack scenarios affect these systems: An attacker could either compromise the server hosting the dependency or compromise the DNS server the build machine uses to redirect requests for hostname of the server hosting the dependency to a machine controlled by the attacker. Both scenarios result in the attacker gaining the ability to inject a malicious version of a dependency into a build running on an otherwise uncompromised machine.

Regardless of the attack vector used to deliver the Trojan dependency, these scenarios share the common element that the build system blindly accepts the malicious binary and includes it in the build. Because the build system has no recourse for rejecting the malicious binary and existing security mechanisms, such as code review, typically focus on internally-developed code rather than external dependencies, this type of attack has a strong potential to go unnoticed as it spreads through the development environment and potentially into production.

Although there is some risk of a compromised dependency being introduced into a manual build process, by the tendency of automated build systems to retrieve the dependency from an external source each time the build system is run in a new environment greatly increases the window of opportunity for an attacker. An attacker need only compromise the dependency server or the DNS server during one of the many times the dependency is retrieved in order to compromise the machine on which the build is occurring.

Recommendation

The simplest solution is to refrain from adopting automated dependency management systems altogether. Managing dependencies manually eliminates the potential for unexpected behavior caused by the build system. Obviously, an attacker could still mount one of the attacks described previously to coincide with the manual retrieval of a dependency, but limiting the frequency with which the dependency must be retrieved significantly reduces the window of opportunity for an attacker. Finally, this solution forces the development organization to rely on what is ostensibly an antiquated build system. A system based on manual dependency management is often more difficult to use and maintain, and might be unacceptable in some software development environments.

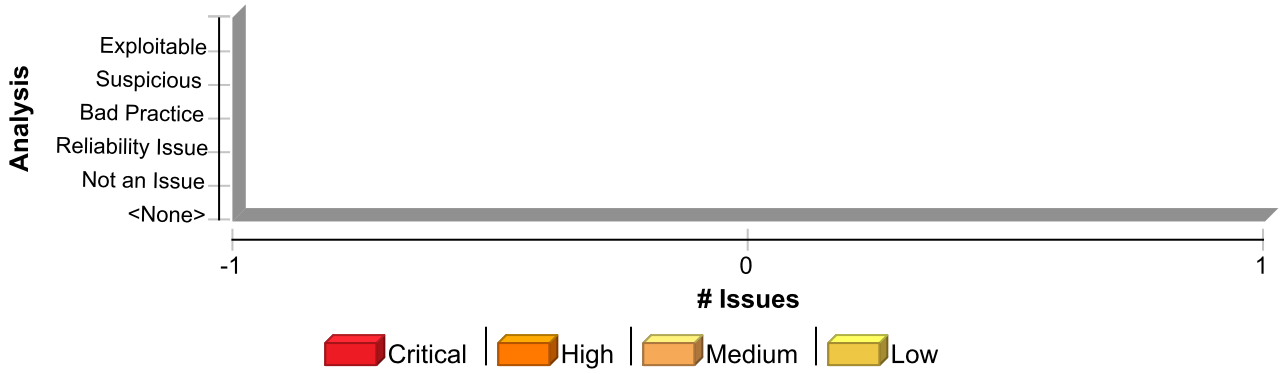
The second solution is a hybrid of the traditional manual dependency management approach and the fully automated solution that is popular today. The biggest advantage of the manual build process is the decreased window of attack, which can be achieved in a semi-automated system by replicating external dependency servers internally. Any build system that requires an external dependency can then point to the internal server using a hard-coded internal IP address to bypass the risk of DNS-based attacks. As new dependencies are added and new versions released, they can be downloaded once and included on the internal repository. This solution reduces the attack opportunities and allows the organization leverage existing internal network security infrastructure.

To implement this solution using Maven, a project should have the IP address for an internal repository hard coded the pom.xml. Specifying the IP address in the pom.xml ensures the internal repository will be used by the corresponding build, but is tied to a specific project. Alternatively, the IP address can be specified in settings.xml, which makes the configuration easier to share across multiple projects.

Example 2: The following Maven pom.xml demonstrates the use of an explicit internal IP address (the entries can also be used in settings.xml):

```
<project>
  ...
  <repositories>
    <repository>
      <releases>
        <enabled>true</enabled>
        <updatePolicy>always</updatePolicy>
        <checksumPolicy>warn</checksumPolicy>
      </releases>
      <snapshots>
        <enabled>true</enabled>
        <updatePolicy>never</updatePolicy>
        <checksumPolicy>fail</checksumPolicy>
      </snapshots>
      <id>central</id>
      <name>Internal Repository</name>
      <url>http://172.16.1.13/maven2</url>
      <layout>default</layout>
    </repository>
  </repositories>
  <pluginRepositories>
    ...
  </pluginRepositories>
  ...
</project>
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Build Misconfiguration: External Maven Dependency Repository	1	0	0	1
Total	**1**	**0**	**0**	**1**

Build Misconfiguration: External Maven Dependency Repository — Low

Package: <none>

pom.xml, line 2 (Build Misconfiguration: External Maven Dependency Repository)

Issue Details

Kingdom: Environment
Scan Engine: SCA (Configuration)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: //project/repositories
File: pom.xml:2

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://
maven.apache.org/xsd/maven-4.0.0.xsd">
3
4   <modelVersion>4.0.0</modelVersion>
5
6   <parent>
7   <groupId>org.springframework.boot</groupId>
```

Code Correctness: Byte Array to String Conversion (2 issues)

Abstract

Converting a byte array into a `String` may lead to data loss.

Explanation

When data from a byte array is converted into a `String`, it is unspecified what will happen to any data that is outside of the applicable character set. This can lead to data being lost, or a decrease in the level of security when binary data is needed to ensure proper security measures are followed.

Example 1: The following code converts data into a String in order to create a hash.

```
...
FileInputStream fis = new FileInputStream(myFile);
byte[] byteArr = byte[BUFSIZE];
...
int count = fis.read(byteArr);
...
String fileString = new String(byteArr);
String fileSHA256Hex = DigestUtils.sha256Hex(fileString);
// use fileSHA256Hex to validate file
...
```

Assuming the size of the file is less than `BUFSIZE`, this works fine as long as the information in `myFile` is encoded the same as the default character set, however if it's using a different encoding, or is a binary file, it will lose information. This in turn will cause the resulting SHA hash to be less reliable, and could mean it's far easier to cause collisions, especially if any data outside of the default character set is represented by the same value, such as a question mark.

Recommendation

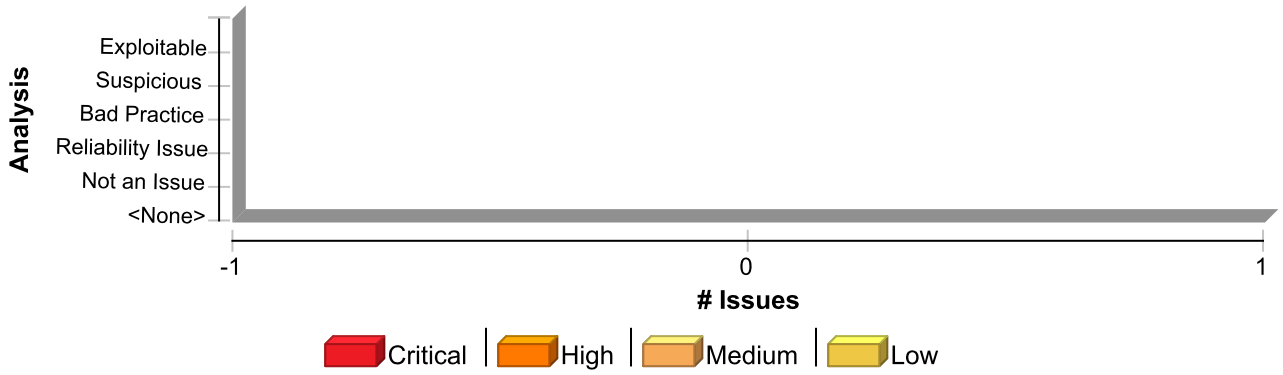
Generally speaking, a byte array potentially containing noncharacter data should never be converted into a `String` object as it may break functionality, but in some cases this can cause much larger security concerns. In a lot of cases there is no need to actually convert a byte array into a String, but if there is a specific reason to be able to create a `String` object from binary data, it must first be encoded in a way such that it will fit into the default character set.

Example 2: The following uses a different variant of the API in `Example 1` to prevent any validation problems.

```
...
FileInputStream fis = new FileInputStream(myFile);
byte[] byteArr = byte[BUFSIZE];
...
int count = fis.read(byteArr);
...
byte[] fileSHA256 = DigestUtils.sha256(byteArr);
// use fileSHA256 to validate file, comparing hash byte-by-byte.
...
```

In this case, it is straightforward to rectify, since this API has overloaded variants including one that accepts a byte array, and this could be simplified even further by using another overloaded variant of `DigestUtils.sha256()` that accepts a `FileInputStream` object as its argument. Other scenarios may need careful consideration as to whether it's possible that the byte array could contain data outside of the character set, and further refactoring may be required.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Code Correctness: Byte Array to String Conversion	2	0	0	2
Total	**2**	**0**	**0**	**2**

Code Correctness: Byte Array to String Conversion — **Low**

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/EncryptedPasswordUtils.java, line 55 (Code Correctness: Byte Array to String Conversion)

Issue Details

Code Correctness: Byte Array to String Conversion — Low

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/EncryptedPasswordUtils.java, line 55 (Code Correctness: Byte Array to String Conversion)

Kingdom: Code Quality
Scan Engine: SCA (Semantic)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: String()
Enclosing Method: encryptPassword()
File: src/main/java/com/microfocus/example/utils/EncryptedPasswordUtils.java:55

```
52  return null;
53  }
54
55  return new String(encrypted);
56  }
57
58  public static boolean matches(String password1, String password2) {
```

src/main/java/com/microfocus/example/utils/EncryptedPasswordUtils.java, line 65 (Code Correctness: Byte Array to String Conversion)

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Semantic)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: String()
Enclosing Method: matches()
File: src/main/java/com/microfocus/example/utils/EncryptedPasswordUtils.java:65

```
62  Cipher desCipher = Cipher.getInstance("DES");
63  desCipher.init(Cipher.ENCRYPT_MODE, keySpec);
64  encrypted = desCipher.doFinal(password1.getBytes());
65  encPassword1 = new String(encrypted);
66  } catch (NoSuchAlgorithmException | NoSuchPaddingException | InvalidKeyException |
IllegalBlockSizeException | BadPaddingException e) {
67  // TODO Auto-generated catch block
68  e.printStackTrace();
```

Cross-Site Request Forgery (7 issues)

Abstract

HTTP requests must contain a user-specific secret in order to prevent an attacker from making unauthorized requests.

Explanation

A cross-site request forgery (CSRF) vulnerability occurs when: 1. A Web application uses session cookies.

2. The application acts on an HTTP request without verifying that the request was made with the user's consent.

A nonce is a cryptographic random value that is sent with a message to prevent replay attacks. If the request does not contain a nonce that proves its provenance, the code that handles the request is vulnerable to a CSRF attack (unless it does not change the state of the application). This means a Web application that uses session cookies has to take special precautions in order to ensure that an attacker can't trick users into submitting bogus requests. Imagine a Web application that allows administrators to create new accounts as follows:

```
  RequestBuilder rb = new RequestBuilder(RequestBuilder.POST, "/new_user");
  body = addToPost(body, new_username);
  body = addToPost(body, new_passwd);
  rb.sendRequest(body, new NewAccountCallback(callback));
```

An attacker might set up a malicious Web site that contains the following code.

```
  RequestBuilder rb = new RequestBuilder(RequestBuilder.POST, "http://
www.example.com/new_user");
  body = addToPost(body, "attacker";
  body = addToPost(body, "haha");
  rb.sendRequest(body, new NewAccountCallback(callback));
```

If an administrator for `example.com` visits the malicious page while she has an active session on the site, she will unwittingly create an account for the attacker. This is a CSRF attack. It is possible because the application does not have a way to determine the provenance of the request. Any request could be a legitimate action chosen by the user or a faked action set up by an attacker. The attacker does not get to see the Web page that the bogus request generates, so the attack technique is only useful for requests that alter the state of the application.

Applications that pass the session identifier in the URL rather than as a cookie do not have CSRF problems because there is no way for the attacker to access the session identifier and include it as part of the bogus request.

CSRF is entry number five on the 2007 OWASP Top 10 list.

Recommendation

Applications that use session cookies must include some piece of information in every form post that the back-end code can use to validate the provenance of the request. One way to do that is to include a random request identifier or nonce, as follows:

```
RequestBuilder rb = new RequestBuilder(RequestBuilder.POST, "/new_user");
body = addToPost(body, new_username);
body = addToPost(body, new_passwd);
body = addToPost(body, request_id);
rb.sendRequest(body, new NewAccountCallback(callback));
```

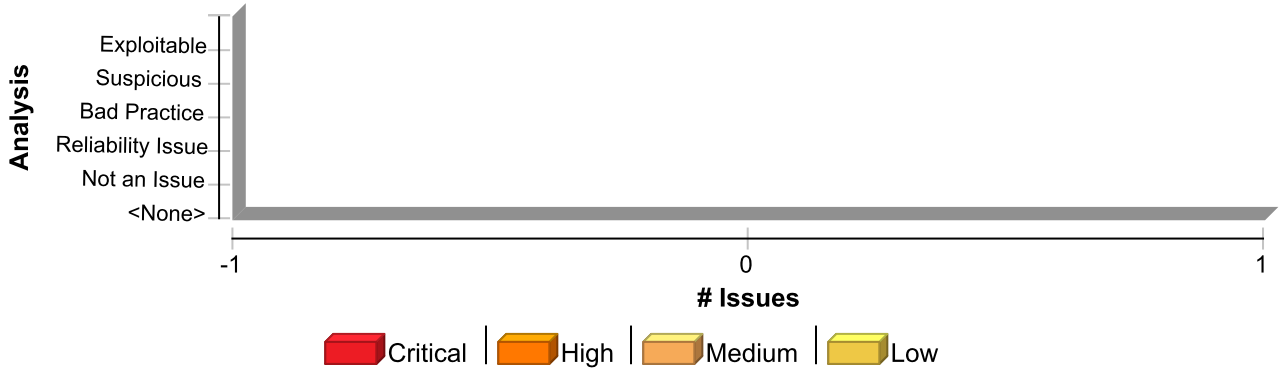
Then the back-end logic can validate the request identifier before processing the rest of the form data. When possible, the request identifier should be unique to each server request rather than shared across every request for a particular session. As with session identifiers, the harder it is for an attacker to guess the request identifier, the harder it is to conduct a successful CSRF attack. The token should not be easily guessed and it should be protected in the same way that session tokens are protected, such as using SSLv3.

Additional mitigation techniques include:

Framework protection: Most modern web application frameworks embed CSRF protection and they will automatically include and verify CSRF tokens. **Use a Challenge-Response control:** Forcing the customer to respond to a challenge sent by the server is a strong defense against CSRF. Some of the challenges that can be used for this purpose are: CAPTCHAs, password re-authentication and one-time tokens. **Check HTTP Referer/Origin headers:** An attacker won't be able to spoof these headers while performing a CSRF attack. This makes these headers a useful method to prevent CSRF attacks. **Double-submit Session Cookie:** Sending the session ID Cookie as a hidden form value in addition to the actual session ID Cookie is a good protection against CSRF attacks. The server will check both values and make sure they are identical before processing the rest of the form data. If an attacker submits a form in behalf of a user, he won't be able to modify the session ID cookie value as per the same-origin-policy. **Limit Session Lifetime:** When accessing protected resources using a CSRF attack, the attack will only be valid as long as the session ID sent as part of the attack is still valid on the server. Limiting the Session lifetime will reduce the probability of a successful attack.

The techniques described here can be defeated with XSS attacks. Effective CSRF mitigation includes XSS mitigation techniques.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Cross-Site Request Forgery	7	0	0	7
Total	**7**	**0**	**0**	**7**

Cross-Site Request Forgery — Low

Package: com.microfocus.example.config

src/main/java/com/microfocus/example/config/WebSecurityConfiguration.java, line 142 (Cross-Site Request Forgery)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: FunctionCall: disable
Enclosing Method: configure()
File: src/main/java/com/microfocus/example/config/WebSecurityConfiguration.java:142

```
139  protected void configure(HttpSecurity httpSecurity) throws Exception {
140  if (activeProfile.contains("dev")) {
141  log.info("Running development profile");
142  httpSecurity.csrf().disable();
143  httpSecurity.headers().frameOptions().disable();
144  httpSecurity.cors().disable();
145  httpSecurity.headers().xssProtection();
```

src/main/java/com/microfocus/example/config/WebSecurityConfiguration.java, line 126 (Cross-Site Request Forgery)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: FunctionCall: disable
Enclosing Method: configure()
File: src/main/java/com/microfocus/example/config/WebSecurityConfiguration.java:126

Cross-Site Request Forgery — Low

Package: com.microfocus.example.config

src/main/java/com/microfocus/example/config/WebSecurityConfiguration.java, line 126 (Cross-Site Request Forgery)

```
123   .and().sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)
124   //.and().httpBasic().authenticationEntryPoint(basicAuthenticationEntryPoint)
125   .and().exceptionHandling().accessDeniedHandler(apiAccessDeniedHandler)
126   .and().csrf().disable();
127
128   httpSecurity.addFilterBefore(authenticationJwtTokenFilter(),
UsernamePasswordAuthenticationFilter.class);
129
```

Package: src.main.resources.static.js.components

src/main/resources/static/js/components/NewProducts.js, line 38 (Cross-Site Request Forgery)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: FunctionPointerCall: get
Enclosing Method: _getProducts()
File: src/main/resources/static/js/components/NewProducts.js:38

```
35   }
36
37   async function _getProducts(limit) {
38   return await $.get(`/api/v3/products?limit=${limit}`).then();
39   }
40
41 };
```

src/main/resources/static/js/components/ProductReviews.js, line 37 (Cross-Site Request Forgery)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: FunctionPointerCall: get
Enclosing Method: _getProductReviews()
File: src/main/resources/static/js/components/ProductReviews.js:37

Cross-Site Request Forgery — Low

Package: src.main.resources.static.js.components

src/main/resources/static/js/components/ProductReviews.js, line 37 (Cross-Site Request Forgery)

```
34  }
35
36  async function _getProductReviews(pid, limit) {
37  return await $.get(`/api/v3/reviews?pid=${pid}&limit=${limit}`).then();
38  }
39
40 };
```

src/main/resources/static/js/components/CartSummary.js, line 22 (Cross-Site Request Forgery)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: FunctionPointerCall: get
Enclosing Method: lambda()
File: src/main/resources/static/js/components/CartSummary.js:22

```
19  cartIsEmpty = false;
20  let subtotal = 0.0;
21  $.each(cart, function (i, product) {
22  $.get("/api/v3/products/" + product.pid)
23  .then(response => {
24  let price = response.price;
25  if (response.onSale) {
```

src/main/resources/static/js/components/Cart.js, line 22 (Cross-Site Request Forgery)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: FunctionPointerCall: get
Enclosing Method: lambda()
File: src/main/resources/static/js/components/Cart.js:22

Cross-Site Request Forgery — Low

Package: src.main.resources.static.js.components

src/main/resources/static/js/components/Cart.js, line 22 (Cross-Site Request Forgery)

```
19  if (size > 0) {
20  cartIsEmpty = false;
21  $.each(cart, function (i, product) {
22  $.get("/api/v3/products/" + product.pid)
23  .then(response => {
24  let price = response.price;
25  if (response.onSale) {
```

src/main/resources/static/js/components/CheckUsername.js, line 29 (Cross-Site Request Forgery)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction	Not Predicted

Sink Details

Sink: FunctionPointerCall: get
Enclosing Method: _usernameTaken()
File: src/main/resources/static/js/components/CheckUsername.js:29

```
26
27  async function _usernameTaken(username) {
28  try {
29  return await $.get(`/api/v3/site/username-already-exists/${username}`);
30  } catch (error) {
31  console.log('Error', error.message)
32  return false;
```

Cross-Site Scripting: Persistent (3 issues)

Abstract

Sending unvalidated data to a web browser can result in the browser executing malicious code.

Explanation

Cross-site scripting (XSS) vulnerabilities occur when:

1. Data enters a web application through an untrusted source. In the case of persistent (also known as stored) XSS, the untrusted source is typically a database or other back-end data store, while in the case of reflected XSS it is typically a web request.

2. The data is included in dynamic content that is sent to a web user without validation.

The malicious content sent to the web browser often takes the form of a JavaScript segment, but can also include HTML, Flash or any other type of code that the browser executes. The variety of attacks based on XSS is almost limitless, but they commonly include transmitting private data such as cookies or other session information to the attacker, redirecting the victim to web content controlled by the attacker, or performing other malicious operations on the user's machine under the guise of the vulnerable site.

Example 1: The following JSP code segment queries a database for an employee with a given ID and prints the corresponding employee's name.

```
<%...
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("select * from emp where id="+eid);
if (rs != null) {
   rs.next();
   String name = rs.getString("name");
}
%>

Employee Name: <%= name %>
```

This code functions correctly when the values of `name` are well-behaved, but it does nothing to prevent exploits if they are not. This code can appear less dangerous because the value of `name` is read from a database, whose contents are apparently managed by the application. However, if the value of `name` originates from user-supplied data, then the database can be a conduit for malicious content. Without proper input validation on all data stored in the database, an attacker may execute malicious commands in the user's web browser. This type of exploit, known as Persistent (or Stored) XSS, is particularly insidious because the indirection caused by the data store makes it difficult to identify the threat and increases the possibility that the attack might affect multiple users. XSS got its start in this form with web sites that offered a "guestbook" to visitors. Attackers would include JavaScript in their guestbook entries, and all subsequent visitors to the guestbook page would execute the malicious code.

Example 2: The following JSP code segment reads an employee ID, `eid`, from an HTTP request and displays it to the user.

```
<% String eid = request.getParameter("eid"); %>
...
Employee ID: <%= eid %>
```

As in `Example 1`, this code operates correctly if `eid` contains only standard alphanumeric text. If `eid` has a value that includes metacharacters or source code, then the code is executed by the web browser as it displays the HTTP response.

Initially this might not appear to be much of a vulnerability. After all, why would someone enter a URL that causes malicious code to run on their own computer? The real danger is that an attacker will create the malicious URL, then use email or social engineering tricks to lure victims into visiting a link to the URL. When victims click the link, they unwittingly reflect the malicious content through the vulnerable web

application back to their own computers. This mechanism of exploiting vulnerable web applications is known as Reflected XSS.

Some think that in the mobile environment, classic web application vulnerabilities, such as cross-site scripting, do not make sense -- why would the user attack themself? However, keep in mind that the essence of mobile platforms is applications that are downloaded from various sources and run alongside each other on the same device. The likelihood of running a piece of malware next to a banking application is high, which necessitates expanding the attack surface of mobile applications to include inter-process communication.

Example 3: The following code enables JavaScript in Android's WebView (by default, JavaScript is disabled) and loads a page based on the value received from an Android intent.

```
...
        WebView webview = (WebView) findViewById(R.id.webview);
        webview.getSettings().setJavaScriptEnabled(true);
        String url = this.getIntent().getExtras().getString("url");
        webview.loadUrl(url);
...
```

If the value of `url` starts with `javascript:`, JavaScript code that follows executes within the context of the web page inside WebView.

As the examples demonstrate, XSS vulnerabilities are caused by code that includes unvalidated data in an HTTP response. There are three vectors by which an XSS attack can reach a victim:

- As in `Example 1`, the application stores dangerous data in a database or other trusted data store. The dangerous data is subsequently read back into the application and included in dynamic content. Persistent XSS exploits occur when an attacker injects dangerous content into a data store that is later read and included in dynamic content. From an attacker's perspective, the optimal place to inject malicious content is in an area that is displayed to either many users or particularly interesting users. Interesting users typically have elevated privileges in the application or interact with sensitive data that is valuable to the attacker. If one of these users executes malicious content, the attacker may be able to perform privileged operations on behalf of the user or gain access to sensitive data belonging to the user.

- As in `Example 2`, data is read directly from the HTTP request and reflected back in the HTTP response. Reflected XSS exploits occur when an attacker causes a user to supply dangerous content to a vulnerable web application, which is then reflected back to the user and executed by the web browser. The most common mechanism for delivering malicious content is to include it as a parameter in a URL that is posted publicly or emailed directly to victims. URLs constructed in this manner constitute the core of many phishing schemes, whereby an attacker convinces victims to visit a URL that refers to a vulnerable site. After the site reflects the attacker's content back to the user, the content is executed and proceeds to transfer private information, such as cookies that might include session information, from the user's machine to the attacker or perform other nefarious activities.

- As in `Example 3`, a source outside the application stores dangerous data in a database or other data store, and the dangerous data is subsequently read back into the application as trusted data and included in dynamic content.

A number of modern web frameworks provide mechanisms to perform user input validation (including Struts and Spring MVC). To highlight the unvalidated sources of input, Fortify Secure Coding Rulepacks dynamically re-prioritize the issues Fortify Static Code Analyzer reports by lowering their probability of exploit and providing pointers to the supporting evidence whenever the framework validation mechanism is in use. We refer to this feature as Context-Sensitive Ranking. To further assist the Fortify user with the

auditing process, the Fortify Software Security Research group makes available the Data Validation project template that groups the issues into folders based on the validation mechanism applied to their source of input.

Recommendation

The solution to prevent XSS is to ensure that validation occurs in the required places and that relevant properties are set to prevent vulnerabilities.

Because XSS vulnerabilities occur when an application includes malicious data in its output, one logical approach is to validate data immediately before it leaves the application. However, because web applications often have complex and intricate code for generating dynamic content, this method is prone to errors of omission (missing validation). An effective way to mitigate this risk is to also perform input validation for XSS.

Web applications must validate all input to prevent other vulnerabilities, such as SQL injection, so augmenting an application's existing input validation mechanism to include checks for XSS is generally relatively easy. Despite its value, input validation for XSS does not take the place of rigorous output validation. An application might accept input through a shared data store or other trusted source, and that data store might accept input from a source that does not perform adequate input validation. Therefore, the application cannot implicitly rely on the safety of this or any other data. This means that the best way to prevent XSS vulnerabilities is to validate everything that enters the application and leaves the application destined for the user.

The most secure approach to validation for XSS is to create an allow list of safe characters that can appear in HTTP content and accept input composed exclusively of characters in the approved set. For example, a valid username might only include alphanumeric characters or a phone number might only include digits 0-9. However, this solution is often infeasible in web applications because many characters that have special meaning to the browser must be considered valid input after they are encoded, such as a web design bulletin board that must accept HTML fragments from its users.

A more flexible, but less secure approach is to implement a deny list, which selectively rejects or escapes potentially dangerous characters before using the input. To form such a list, you first need to understand the set of characters that hold special meaning for web browsers. Although the HTML standard defines which characters have special meaning, many web browsers try to correct common mistakes in HTML and might treat other characters as special in certain contexts. This is why we do not recommend the use of deny lists as a means to prevent XSS. The CERT(R) Coordination Center at the Software Engineering Institute at Carnegie Mellon University provides the following details about special characters in various contexts [1]:

In the content of a block-level element (in the middle of a paragraph of text):

- "<" is special because it introduces a tag.

- "&" is special because it introduces a character entity.

- ">" is special because some browsers treat it as special, on the assumption that the author of the page intended to include an opening "<", but omitted it in error.

The following principles apply to attribute values:

- In attribute values enclosed in double quotes, the double quotes are special because they mark the end of the attribute value.

- In attribute values enclosed in single quotes, the single quotes are special because they mark the end of the attribute value.

- In attribute values without any quotes, white-space characters, such as space and tab, are special.

- "&" is special when used with certain attributes, because it introduces a character entity.

In URLs, for example, a search engine might provide a link within the results page that the user can click to re-run the search. This can be implemented by encoding the search query inside the URL, which introduces additional special characters:

- Space, tab, and new line are special because they mark the end of the URL.

- "&" is special because it either introduces a character entity or separates CGI parameters.

- Non-ASCII characters (that is, everything greater than 127 in the ISO-8859-1 encoding) are not allowed in URLs, so they are considered to be special in this context.

- The "%" symbol must be filtered from input anywhere parameters encoded with HTTP escape sequences are decoded by server-side code. For example, "%" must be filtered if input such as "%68%65%6C%6C%6F" becomes "hello" when it appears on the web page.

Within the body of a <SCRIPT> </SCRIPT>:

- Semicolons, parentheses, curly braces, and new line characters must be filtered out in situations where text could be inserted directly into a pre-existing script tag.

Server-side scripts:

- Server-side scripts that convert any exclamation characters (!) in input to double-quote characters (") on output might require additional filtering.

Other possibilities:

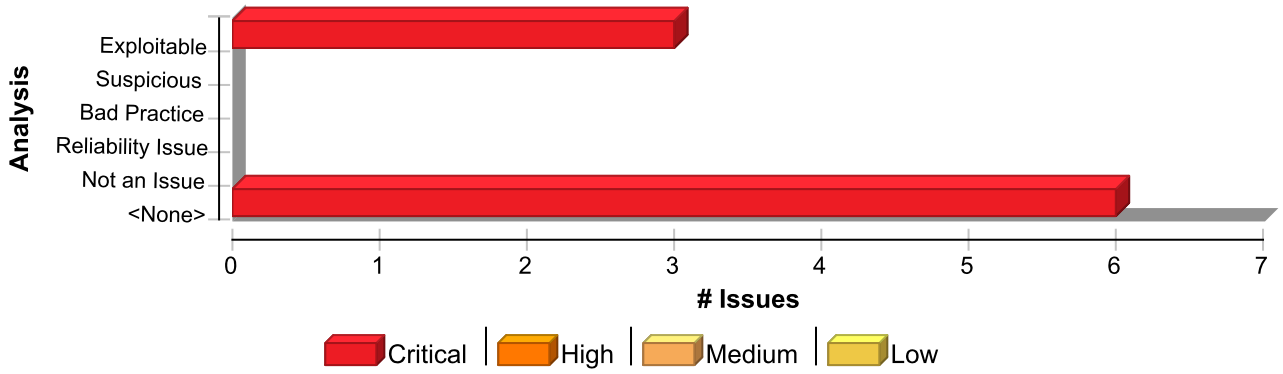
- If an attacker submits a request in UTF-7, the special character '<' appears as '+ADw-' and might bypass filtering. If the output is included in a page that does not explicitly specify an encoding format, then some browsers try to intelligently identify the encoding based on the content (in this case, UTF-7).

After you identify the correct points in an application to perform validation for XSS attacks and what special characters the validation should consider, the next challenge is to identify how your validation handles special characters. If special characters are not considered valid input to the application, then you can reject any input that contains special characters as invalid. A second option is to remove special characters with filtering. However, filtering has the side effect of changing any visual representation of the filtered content and might be unacceptable in circumstances where the integrity of the input must be preserved for display.

If input containing special characters must be accepted and displayed accurately, validation must encode any special characters to remove their significance. A complete list of ISO 8859-1 encoded values for special characters is provided as part of the official HTML specification [2].

Many application servers attempt to limit an application's exposure to cross-site scripting vulnerabilities by providing implementations for the functions responsible for setting certain specific HTTP response content that perform validation for the characters essential to a cross-site scripting attack. Do not rely on the server running your application to make it secure. For any developed application, there are no guarantees about which application servers it will run on during its lifetime. As standards and known exploits evolve, there are no guarantees that application servers will continue to stay in sync.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Cross-Site Scripting: Persistent	3	0	0	3
Total	**3**	**0**	**0**	**3**

Cross-Site Scripting: Persistent — Critical

Package: com.microfocus.example.api.controllers

src/main/java/com/microfocus/example/api/controllers/ApiOrderController.java, line 105 (Cross-Site Scripting: Persistent)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

JiraBugLink	
Analysis	Exploitable
AA_Training	Include
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.595

Audit Comments

admin: Mon Jun 19 2023 10:19:21 GMT-0000 (UTC)
This is a problem that needs fix

Source Details

Source: org.springframework.data.jpa.repository.JpaRepository.findAll()
From: com.microfocus.example.service.ProductService.getAllOrders
File: src/main/java/com/microfocus/example/service/ProductService.java:308

```
305  return orderRepository.findByNumber(number);
306  }
307
308  public List<Order> getAllOrders() { return orderRepository.findAll(); }
309
310  public List<Order> getAllOrders(Integer offset, String keywords) {
311  if (keywords != null && !keywords.isEmpty()) {
```

Cross-Site Scripting: Persistent — Critical

Package: com.microfocus.example.api.controllers

src/main/java/com/microfocus/example/api/controllers/ApiOrderController.java, line 105 (Cross-Site Scripting: Persistent)

Sink Details

Sink: org.springframework.http.ResponseEntity.BodyBuilder.body()
Enclosing Method: getOrdersByKeywords()
File: src/main/java/com/microfocus/example/api/controllers/ApiOrderController.java:105
Taint Flags: DATABASE, PRIMARY_KEY, XSS

```
102  return ResponseEntity.ok().body(
103  productService.getAllOrders().stream()
104  .map(OrderResponse::new)
105  .collect(Collectors.toList()));
106  } else {
107  String k = (keywords.orElse(""));
108  Integer o = (offset.orElse(0));
```

src/main/java/com/microfocus/example/api/controllers/ApiRoleController.java, line 78 (Cross-Site Scripting: Persistent)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

JiraBugLink	
Analysis	Exploitable
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.516

Source Details

Source: org.springframework.data.jpa.repository.JpaRepository.findAll()
From: com.microfocus.example.service.UserService.getAllRoles
File: src/main/java/com/microfocus/example/service/UserService.java:326

```
323  //
324
325  public List<Authority> getAllRoles() {
326  return roleRepository.findAll();
327  }
328
329  //public List<Authority> getUserRoles(Integer userId) {
```

Sink Details

Sink: org.springframework.http.ResponseEntity.BodyBuilder.body()
Enclosing Method: getRolesByKeywords()
File: src/main/java/com/microfocus/example/api/controllers/ApiRoleController.java:78
Taint Flags: DATABASE, XSS

Cross-Site Scripting: Persistent — Critical

Package: com.microfocus.example.api.controllers

src/main/java/com/microfocus/example/api/controllers/ApiRoleController.java, line 78 (Cross-Site Scripting: Persistent)

```
75  log.debug("API::Retrieving roles by keyword(s)");
76  // TODO: implement keywords, offset and limit
77  if (keywords.equals(Optional.empty())) {
78  return ResponseEntity.ok().body(roleService.getAllRoles());
79  } else {
80  return new ResponseEntity<>(roleService.getAllRoles(), HttpStatus.OK);
81  }
```

src/main/java/com/microfocus/example/api/controllers/ApiRoleController.java, line 78 (Cross-Site Scripting: Persistent)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

JiraBugLink	
Analysis	Exploitable
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.516

Source Details

Source: org.springframework.data.jpa.repository.JpaRepository.findAll()
From: com.microfocus.example.service.UserService.getAllRoles
File: src/main/java/com/microfocus/example/service/UserService.java:326

```
323  //
324
325  public List<Authority> getAllRoles() {
326  return roleRepository.findAll();
327  }
328
329  //public List<Authority> getUserRoles(Integer userId) {
```

Sink Details

Sink: org.springframework.http.ResponseEntity.BodyBuilder.body()
Enclosing Method: getRolesByKeywords()
File: src/main/java/com/microfocus/example/api/controllers/ApiRoleController.java:78
Taint Flags: DATABASE, XSS

Cross-Site Scripting: Persistent — Critical

Package: com.microfocus.example.api.controllers

src/main/java/com/microfocus/example/api/controllers/ApiRoleController.java, line 78 (Cross-Site Scripting: Persistent)

```
75  log.debug("API::Retrieving roles by keyword(s)");
76  // TODO: implement keywords, offset and limit
77  if (keywords.equals(Optional.empty())) {
78  return ResponseEntity.ok().body(roleService.getAllRoles());
79  } else {
80  return new ResponseEntity<>(roleService.getAllRoles(), HttpStatus.OK);
81  }
```

Cross-Site Scripting: Reflected (7 issues)

Abstract

Sending unvalidated data to a web browser can result in the browser executing malicious code.

Explanation

Cross-site scripting (XSS) vulnerabilities occur when:

1. Data enters a web application through an untrusted source. In the case of reflected XSS, the untrusted source is typically a web request, while in the case of persisted (also known as stored) XSS it is typically a database or other back-end data store.

2. The data is included in dynamic content that is sent to a web user without validation.

The malicious content sent to the web browser often takes the form of a JavaScript segment, but can also include HTML, Flash or any other type of code that the browser executes. The variety of attacks based on XSS is almost limitless, but they commonly include transmitting private data such as cookies or other session information to the attacker, redirecting the victim to web content controlled by the attacker, or performing other malicious operations on the user's machine under the guise of the vulnerable site.

Example 1: The following JSP code segment reads an employee ID, `eid`, from an HTTP request and displays it to the user.

```
<% String eid = request.getParameter("eid"); %>
...
Employee ID: <%= eid %>
```

The code in this example operates correctly if `eid` contains only standard alphanumeric text. If `eid` has a value that includes metacharacters or source code, then the code is executed by the web browser as it displays the HTTP response.

Initially this might not appear to be much of a vulnerability. After all, why would someone enter a URL that causes malicious code to run on their own computer? The real danger is that an attacker will create the malicious URL, then use email or social engineering tricks to lure victims into visiting a link to the URL. When victims click the link, they unwittingly reflect the malicious content through the vulnerable web application back to their own computers. This mechanism of exploiting vulnerable web applications is known as Reflected XSS.

Example 2: The following JSP code segment queries a database for an employee with a given ID and prints the corresponding employee's name.

```
<%...
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("select * from emp where id="+eid);
if (rs != null) {
   rs.next();
   String name = rs.getString("name");
}
%>

Employee Name: <%= name %>
```

As in `Example 1`, this code functions correctly when the values of `name` are well-behaved, but it does nothing to prevent exploits if they are not. Again, this code can appear less dangerous because the value of `name` is read from a database, whose contents are apparently managed by the application. However, if the value of `name` originates from user-supplied data, then the database can be a conduit for malicious content. Without proper input validation on all data stored in the database, an attacker may execute malicious commands in the user's web browser. This type of exploit, known as Persistent (or Stored) XSS, is particularly insidious because the indirection caused by the data store makes it difficult to identify the threat and increases the possibility that the attack might affect multiple users. XSS got its start in this form

with web sites that offered a "guestbook" to visitors. Attackers would include JavaScript in their guestbook entries, and all subsequent visitors to the guestbook page would execute the malicious code.

Some think that in the mobile environment, classic web application vulnerabilities, such as cross-site scripting, do not make sense -- why would the user attack themself? However, keep in mind that the essence of mobile platforms is applications that are downloaded from various sources and run alongside each other on the same device. The likelihood of running a piece of malware next to a banking application is high, which necessitates expanding the attack surface of mobile applications to include inter-process communication.

Example 3: The following code enables JavaScript in Android's WebView (by default, JavaScript is disabled) and loads a page based on the value received from an Android intent.

```
...
        WebView webview = (WebView) findViewById(R.id.webview);
        webview.getSettings().setJavaScriptEnabled(true);
        String url = this.getIntent().getExtras().getString("url");
        webview.loadUrl(url);
...
```

If the value of `url` starts with `javascript:`, JavaScript code that follows executes within the context of the web page inside WebView.

As the examples demonstrate, XSS vulnerabilities are caused by code that includes unvalidated data in an HTTP response. There are three vectors by which an XSS attack can reach a victim:

- As in `Example 1`, data is read directly from the HTTP request and reflected back in the HTTP response. Reflected XSS exploits occur when an attacker causes a user to supply dangerous content to a vulnerable web application, which is then reflected back to the user and executed by the web browser. The most common mechanism for delivering malicious content is to include it as a parameter in a URL that is posted publicly or emailed directly to victims. URLs constructed in this manner constitute the core of many phishing schemes, whereby an attacker convinces victims to visit a URL that refers to a vulnerable site. After the site reflects the attacker's content back to the user, the content is executed and proceeds to transfer private information, such as cookies that might include session information, from the user's machine to the attacker or perform other nefarious activities.

- As in `Example 2`, the application stores dangerous data in a database or other trusted data store. The dangerous data is subsequently read back into the application and included in dynamic content. Persistent XSS exploits occur when an attacker injects dangerous content into a data store that is later read and included in dynamic content. From an attacker's perspective, the optimal place to inject malicious content is in an area that is displayed to either many users or particularly interesting users. Interesting users typically have elevated privileges in the application or interact with sensitive data that is valuable to the attacker. If one of these users executes malicious content, the attacker may be able to perform privileged operations on behalf of the user or gain access to sensitive data belonging to the user.

- As in `Example 3`, a source outside the application stores dangerous data in a database or other data store, and the dangerous data is subsequently read back into the application as trusted data and included in dynamic content.

A number of modern web frameworks provide mechanisms to perform user input validation (including Struts and Spring MVC). To highlight the unvalidated sources of input, Fortify Secure Coding Rulepacks dynamically re-prioritize the issues Fortify Static Code Analyzer reports by lowering their probability of exploit and providing pointers to the supporting evidence whenever the framework validation mechanism is in use. We refer to this feature as Context-Sensitive Ranking. To further assist the Fortify user with the

auditing process, the Fortify Software Security Research group makes available the Data Validation project template that groups the issues into folders based on the validation mechanism applied to their source of input.

Recommendation

The solution to prevent XSS is to ensure that validation occurs in the required places and that relevant properties are set to prevent vulnerabilities.

Because XSS vulnerabilities occur when an application includes malicious data in its output, one logical approach is to validate data immediately before it leaves the application. However, because web applications often have complex and intricate code for generating dynamic content, this method is prone to errors of omission (missing validation). An effective way to mitigate this risk is to also perform input validation for XSS.

Web applications must validate all input to prevent other vulnerabilities, such as SQL injection, so augmenting an application's existing input validation mechanism to include checks for XSS is generally relatively easy. Despite its value, input validation for XSS does not take the place of rigorous output validation. An application might accept input through a shared data store or other trusted source, and that data store might accept input from a source that does not perform adequate input validation. Therefore, the application cannot implicitly rely on the safety of this or any other data. This means that the best way to prevent XSS vulnerabilities is to validate everything that enters the application and leaves the application destined for the user.

The most secure approach to validation for XSS is to create an allow list of safe characters that can appear in HTTP content and accept input composed exclusively of characters in the approved set. For example, a valid username might only include alphanumeric characters or a phone number might only include digits 0-9. However, this solution is often infeasible in web applications because many characters that have special meaning to the browser must be considered valid input after they are encoded, such as a web design bulletin board that must accept HTML fragments from its users.

A more flexible, but less secure approach is to implement a deny list, which selectively rejects or escapes potentially dangerous characters before using the input. To form such a list, you first need to understand the set of characters that hold special meaning for web browsers. Although the HTML standard defines which characters have special meaning, many web browsers try to correct common mistakes in HTML and might treat other characters as special in certain contexts. This is why we do not recommend the use of deny lists as a means to prevent XSS. The CERT(R) Coordination Center at the Software Engineering Institute at Carnegie Mellon University provides the following details about special characters in various contexts [1]:

In the content of a block-level element (in the middle of a paragraph of text):

- "<" is special because it introduces a tag.

- "&" is special because it introduces a character entity.

- ">" is special because some browsers treat it as special, on the assumption that the author of the page intended to include an opening "<", but omitted it in error.

The following principles apply to attribute values:

- In attribute values enclosed in double quotes, the double quotes are special because they mark the end of the attribute value.

- In attribute values enclosed in single quotes, the single quotes are special because they mark the end of the attribute value.

- In attribute values without any quotes, white-space characters, such as space and tab, are special.

- "&" is special when used with certain attributes, because it introduces a character entity.

In URLs, for example, a search engine might provide a link within the results page that the user can click to re-run the search. This can be implemented by encoding the search query inside the URL, which introduces additional special characters:

- Space, tab, and new line are special because they mark the end of the URL.

- "&" is special because it either introduces a character entity or separates CGI parameters.

- Non-ASCII characters (that is, everything greater than 127 in the ISO-8859-1 encoding) are not allowed in URLs, so they are considered to be special in this context.

- The "%" symbol must be filtered from input anywhere parameters encoded with HTTP escape sequences are decoded by server-side code. For example, "%" must be filtered if input such as "%68%65%6C%6C%6F" becomes "hello" when it appears on the web page.

Within the body of a <SCRIPT> </SCRIPT>:

- Semicolons, parentheses, curly braces, and new line characters must be filtered out in situations where text could be inserted directly into a pre-existing script tag.

Server-side scripts:

- Server-side scripts that convert any exclamation characters (!) in input to double-quote characters (") on output might require additional filtering.

Other possibilities:

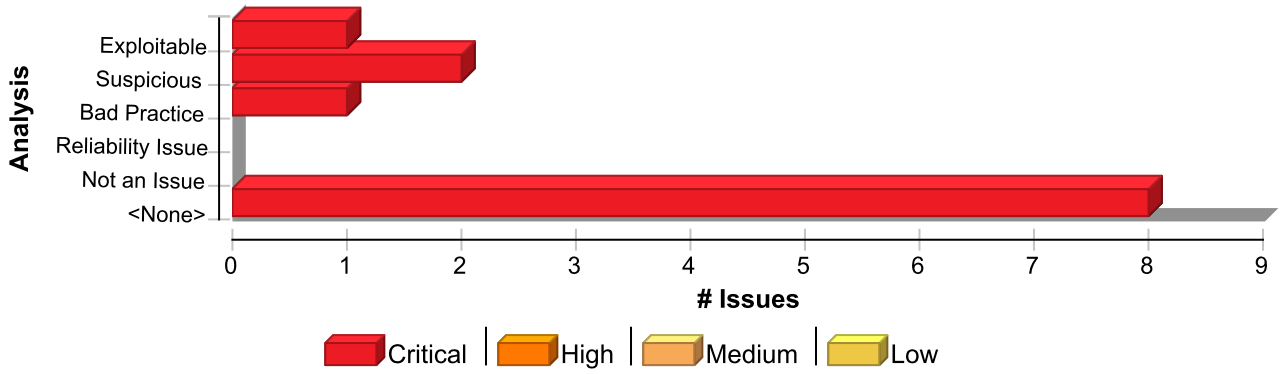
- If an attacker submits a request in UTF-7, the special character '<' appears as '+ADw-' and might bypass filtering. If the output is included in a page that does not explicitly specify an encoding format, then some browsers try to intelligently identify the encoding based on the content (in this case, UTF-7).

After you identify the correct points in an application to perform validation for XSS attacks and what special characters the validation should consider, the next challenge is to identify how your validation handles special characters. If special characters are not considered valid input to the application, then you can reject any input that contains special characters as invalid. A second option is to remove special characters with filtering. However, filtering has the side effect of changing any visual representation of the filtered content and might be unacceptable in circumstances where the integrity of the input must be preserved for display.

If input containing special characters must be accepted and displayed accurately, validation must encode any special characters to remove their significance. A complete list of ISO 8859-1 encoded values for special characters is provided as part of the official HTML specification [2].

Many application servers attempt to limit an application's exposure to cross-site scripting vulnerabilities by providing implementations for the functions responsible for setting certain specific HTTP response content that perform validation for the characters essential to a cross-site scripting attack. Do not rely on the server running your application to make it secure. For any developed application, there are no guarantees about which application servers it will run on during its lifetime. As standards and known exploits evolve, there are no guarantees that application servers will continue to stay in sync.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Cross-Site Scripting: Reflected	4	3	0	7
Total	**4**	**3**	**0**	**7**

Cross-Site Scripting: Reflected — Critical

URL: https://iwa.onfortify.com:443/products

https://iwa.onfortify.com:443/products (Cross-Site Scripting: Reflected)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: WEBINSPECT (Dynamic Analysis)

Source Details

URL: https://iwa.onfortify.com:443/products
HTTP Method: GET
Request Parameters:

keywords:12345%3c%73%43%72%49%70%54%3e%61%6c%65%72%74%28%31%37%33%34%36%29%3c%

49%70%54%3e
Vulnerable Parameter: keywords
Attack Payload: keywords:
12345%3c%73%43%72%49%70%54%3e%61%6c%65%72%74%28%31%37%33%
34%36%29%3c%2f%73%43%72%49%70%54%3e

Request

Cross-Site Scripting: Reflected — Critical

URL: https://iwa.onfortify.com:443/products

https://iwa.onfortify.com:443/products (Cross-Site Scripting: Reflected)

```
GET /products?
keywords=12345%3c%73%43%72%49%70%54%3e%61%6c%65%72%74%28%31%37%33%34%36%29%3c%
HTTP/1.1
Referer: https://iwa.onfortify.com/
Accept: */*
Pragma: no-cache
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Windows NT 6.2; Win64; x64; rv:83.0) Gecko/20100101
Firefox/83.0
Host: iwa.onfortify.com
Connection: Keep-Alive
X-RequestManager-Memo: stid="93";stmi="0";sc="1";rid="b130696a";
Cookie: ;JSESSIONID=111988E98493DDB00481E4011E815ED0
```

URL: https://iwa.onfortify.com:443/products/

https://iwa.onfortify.com:443/products/ (Cross-Site Scripting: Reflected)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: WEBINSPECT (Dynamic Analysis)

Source Details

URL: https://iwa.onfortify.com:443/products/
HTTP Method: GET
Request Parameters:

keywords:12345%3c%73%43%72%49%70%54%3e%61%6c%65%72%74%28%35%35%38%34%31%29%3c%

49%70%54%3e
Vulnerable Parameter: keywords
Attack Payload: keywords:
12345%3c%73%43%72%49%70%54%3e%61%6c%65%72%74%28%35%35%38%
34%31%29%3c%2f%73%43%72%49%70%54%3e

Request

Cross-Site Scripting: Reflected — Critical

URL: https://iwa.onfortify.com:443/products/

https://iwa.onfortify.com:443/products/ (Cross-Site Scripting: Reflected)

```
GET /products/?
keywords=12345%3c%73%43%72%49%70%54%3e%61%6c%65%72%74%28%35%35%38%34%31%29%3c%
HTTP/1.1
Referer: https://iwa.onfortify.com/products/
Accept: */*
Pragma: no-cache
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Windows NT 6.2; Win64; x64; rv:83.0) Gecko/20100101
Firefox/83.0
Host: iwa.onfortify.com
Connection: Keep-Alive
X-RequestManager-Memo: stid="101";stmi="0";sc="1";rid="9d20bb64";
Cookie: ;JSESSIONID=C6C2F29198441C948AB6FF89B0280BB4
```

URL: https://iwa.onfortify.com:443/products/xss

https://iwa.onfortify.com:443/products/xss (Cross-Site Scripting: Reflected)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: WEBINSPECT (Dynamic Analysis)

Source Details

URL: https://iwa.onfortify.com:443/products/xss
HTTP Method: GET
Request Parameters:

keywords:12345%3c%73%43%72%49%70%54%20%74%59%70%45%3d%74%45%78%54%2f%76%42%73%4

54%3e%4d%73%67%42%6f%78%28%38%31%37%35%36%29%3c%2f%73%43%72%49%70%54%3e
Vulnerable Parameter: keywords
Attack Payload: keywords:
12345%3c%73%43%72%49%70%54%20%74%59%70%45%3d%74%45%78%54%

2f%76%42%73%43%72%49%70%54%3e%4d%73%67%42%6f%78%28%38%31%37%35%36%29%3c%2f%73

54%3e

Request

Cross-Site Scripting: Reflected — Critical

URL: https://iwa.onfortify.com:443/products/xss

https://iwa.onfortify.com:443/products/xss (Cross-Site Scripting: Reflected)

```
GET /products/xss?
keywords=12345%3c%73%43%72%49%70%54%20%74%59%70%45%3d%74%45%78%54%2f%76%42%73%
HTTP/1.1
Referer: https://iwa.onfortify.com/products/firstaid
Accept: */*
Pragma: no-cache
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Windows NT 6.2; Win64; x64; rv:83.0) Gecko/20100101
Firefox/83.0
Host: iwa.onfortify.com
Connection: Keep-Alive
X-RequestManager-Memo: stid="99";stmi="0";sc="1";rid="70577d8c";
Cookie: ;JSESSIONID=342EA0DB588D436C3ABE7D19EC69A7BB
```

URL: null

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 90 (Cross-Site Scripting: Reflected)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

JiraBugLink	
Analysis	Exploitable
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.522

Audit Comments

admin: Thu Jun 15 2023 11:45:32 GMT-0000 (UTC)
we need to fix it

Source Details

Source: getKeywordsContent(0)
From: com.microfocus.example.web.controllers.ProductController.getKeywordsContent
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:86
URL: null

Cross-Site Scripting: Reflected — Critical

URL: null

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 90 (Cross-Site Scripting: Reflected)

```
83
84  @GetMapping("/xss")
85  @ResponseBody
86  public ResponseEntity<String> getKeywordsContent(@Param("keywords")
String keywords) {
87
88  String retContent = "Product search using: " + keywords;
89
```

Sink Details

Sink: org.springframework.http.ResponseEntity.BodyBuilder.body()
Enclosing Method: getKeywordsContent()
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:90
Taint Flags: WEB, XSS

```
87
88  String retContent = "Product search using: " + keywords;
89
90  return ResponseEntity.ok().body(retContent);
91  }
92
93  @GetMapping("/firstaid")
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 550 (Cross-Site Scripting: Reflected)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.549

Source Details

Source: serveFile(0)
From: com.microfocus.example.web.controllers.UserController.serveFile
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:546
URL: null

Cross-Site Scripting: Reflected — Critical

URL: null

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 550 (Cross-Site Scripting: Reflected)

```
543
544  @GetMapping("/files/{filename:.+}")
545  @ResponseBody
546  public ResponseEntity<Resource> serveFile(@PathVariable String filename)
{
547
548  Resource file = storageService.loadAsResource(filename);
549  return ResponseEntity.ok().header(HttpHeaders.CONTENT_DISPOSITION,
```

Sink Details

Sink: org.springframework.http.ResponseEntity.BodyBuilder.body()
Enclosing Method: serveFile()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:550
Taint Flags: WEB, XSS

```
547
548  Resource file = storageService.loadAsResource(filename);
549  return ResponseEntity.ok().header(HttpHeaders.CONTENT_DISPOSITION,
550  "attachment; filename=\"" + file.getFilename() + "\"").body(file);
551  }
552
553  @PostMapping("/files/upload")
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 704 (Cross-Site Scripting: Reflected)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.584

Source Details

Source: serveUnverifiedFile(0)
From: com.microfocus.example.web.controllers.UserController.serveUnverifiedFile
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:696
URL: null

Cross-Site Scripting: Reflected — Critical

URL: null

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 704 (Cross-Site Scripting: Reflected)

```
693   }
694
695   @GetMapping("/files/download/unverified")
696   public ResponseEntity<?> serveUnverifiedFile(@Param("file") String file)
{
697
698   if (Objects.isNull(file) || file.isEmpty()) {
699   return ResponseEntity.badRequest().build();
```

Sink Details

Sink: org.springframework.http.ResponseEntity.BodyBuilder.body()
Enclosing Method: serveUnverifiedFile()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:704
Taint Flags: WEB, XSS

```
701
702   Resource rfile = storageService.loadAsResource(file, true);
703   return ResponseEntity.ok().header(HttpHeaders.CONTENT_DISPOSITION,
704   "attachment; filename=\"" + rfile.getFilename() + "\"").body(rfile);
705   }
706
707 }
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 622 (Cross-Site Scripting: Reflected)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.549

Source Details

Source: serveXMLFile(0)
From: com.microfocus.example.web.controllers.UserController.serveXMLFile
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:618
URL: null

Cross-Site Scripting: Reflected — Critical

URL: null

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 622 (Cross-Site Scripting: Reflected)

```
615
616   @GetMapping("/files/xml/{filename:.+}")
617   @ResponseBody
618   public ResponseEntity<Resource> serveXMLFile(@PathVariable String
filename) {
619
620   Resource file = storageService.loadAsResource(filename);
621   return ResponseEntity.ok().header(HttpHeaders.CONTENT_DISPOSITION,
```

Sink Details

Sink: org.springframework.http.ResponseEntity.BodyBuilder.body()
Enclosing Method: serveXMLFile()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:622
Taint Flags: WEB, XSS

```
619
620  Resource file = storageService.loadAsResource(filename);
621  return ResponseEntity.ok().header(HttpHeaders.CONTENT_DISPOSITION,
622  "attachment; filename=\"" + file.getFilename() + "\"").body(file);
623  }
624
625  @PostMapping("/files/upload-xml")
```

Database Bad Practices: Use of Restricted Accounts (3 issues)

Abstract

An attempt was made to use one of the following accounts to connect to the database: admin, administrator, guest, root, or sa.

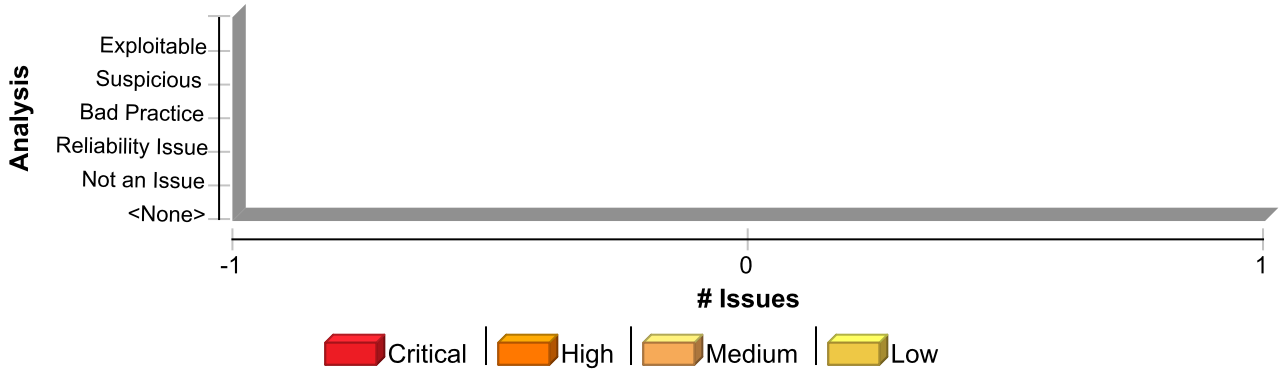
Explanation

Windows Azure SQL Database supports only SQL Server Authentication. Windows Authentication (integrated security) is not supported. Users must provide credentials (login and password) every time they connect to Windows Azure SQL Database. Per Microsoft Windows Azure SQL Database General Guidelines and Limitations, the following account names are not available: admin, administrator, guest, root, sa.

Recommendation

Do not use the following accounts in your database scripts: admin, administrator, guest, root, or sa.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Database Bad Practices: Use of Restricted Accounts	3	0	0	3
Total	**3**	**0**	**0**	**3**

Database Bad Practices: Use of Restricted Accounts — Low

Package: src.main.resources

src/main/resources/data.sql, line 9 (Database Bad Practices: Use of Restricted Accounts)

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Database Bad Practices: Use of Restricted Accounts — Low

Package: src.main.resources

src/main/resources/data.sql, line 9 (Database Bad Practices: Use of Restricted Accounts)

Sink: StringLiteral
Enclosing Method: _plsql_block1data_sql()
File: src/main/resources/data.sql:9

```
6 values ('ROLE_API', 'dfc1d81b-4a7e-4248-80f7-8445ee5cb68e');
7 INSERT INTO users (id, username, password, first_name, last_name, email, phone, address,
city, state, zip, country, date_created, enabled)
8 VALUES ('e18c8bcc-935d-444d-a194-3a32a3b35a49', 'admin',
'$2a$10$YFhTnHpCL.Z0Ev0j1CbEUub7sIWmN7Qd5RmnU8g5ekuoapV7Zdx32',
9  'Admin', 'User', 'admin@localhost.com', '+44808123456', '', '', '', '', 'United Kingdom',
CURDATE(), 1);
10 INSERT INTO users (id, username, password, first_name, last_name, email, phone, address,
city, state, zip, country, date_created, enabled)
11 VALUES ('32e7db01-86bc-4687-9ecb-d79b265ac14f', 'user1',
'$2a$10$YFhTnHpCL.Z0Ev0j1CbEUub7sIWmN7Qd5RmnU8g5ekuoapV7Zdx32',
12  'Sam', 'Shopper', 'user1@localhost.com', '+44808123456', '1 Somewhere Street', 'London',
'Greater London', 'SW1', 'United Kingdom', CURDATE(), 1);
```

src/main/resources/data.sql, line 9 (Database Bad Practices: Use of Restricted Accounts)

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction	Not Predicted

Sink Details

Sink: StringLiteral
Enclosing Method: _plsql_block1data_sql()
File: src/main/resources/data.sql:9

```
6 values ('ROLE_API', 'dfc1d81b-4a7e-4248-80f7-8445ee5cb68e');
7 INSERT INTO users (id, username, password, first_name, last_name, email, phone, address,
city, state, zip, country, date_created, enabled)
8 VALUES ('e18c8bcc-935d-444d-a194-3a32a3b35a49', 'admin',
'$2a$10$YFhTnHpCL.Z0Ev0j1CbEUub7sIWmN7Qd5RmnU8g5ekuoapV7Zdx32',
9  'Admin', 'User', 'admin@localhost.com', '+44808123456', '', '', '', '', 'United Kingdom',
CURDATE(), 1);
10 INSERT INTO users (id, username, password, first_name, last_name, email, phone, address,
city, state, zip, country, date_created, enabled)
11 VALUES ('32e7db01-86bc-4687-9ecb-d79b265ac14f', 'user1',
'$2a$10$YFhTnHpCL.Z0Ev0j1CbEUub7sIWmN7Qd5RmnU8g5ekuoapV7Zdx32',
12  'Sam', 'Shopper', 'user1@localhost.com', '+44808123456', '1 Somewhere Street', 'London',
'Greater London', 'SW1', 'United Kingdom', CURDATE(), 1);
```

src/main/resources/data.sql, line 9 (Database Bad Practices: Use of Restricted Accounts)

Issue Details

Database Bad Practices: Use of Restricted Accounts — Low

Package: src.main.resources

src/main/resources/data.sql, line 9 (Database Bad Practices: Use of Restricted Accounts)

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction	Not Predicted

Sink Details

Sink: StringLiteral
Enclosing Method: _plsql_block1data_sql()
File: src/main/resources/data.sql:9

```
6 values ('ROLE_API', 'dfc1d81b-4a7e-4248-80f7-8445ee5cb68e');
7 INSERT INTO users (id, username, password, first_name, last_name, email, phone, address,
city, state, zip, country, date_created, enabled)
8 VALUES ('e18c8bcc-935d-444d-a194-3a32a3b35a49', 'admin',
'$2a$10$YFhTnHpCL.Z0Ev0j1CbEUub7sIWmN7Qd5RmnU8g5ekuoapV7Zdx32',
9  'Admin', 'User', 'admin@localhost.com', '+44808123456', '', '', '', '', 'United Kingdom',
CURDATE(), 1);
10 INSERT INTO users (id, username, password, first_name, last_name, email, phone, address,
city, state, zip, country, date_created, enabled)
11 VALUES ('32e7db01-86bc-4687-9ecb-d79b265ac14f', 'user1',
'$2a$10$YFhTnHpCL.Z0Ev0j1CbEUub7sIWmN7Qd5RmnU8g5ekuoapV7Zdx32',
12  'Sam', 'Shopper', 'user1@localhost.com', '+44808123456', '1 Somewhere Street', 'London',
'Greater London', 'SW1', 'United Kingdom', CURDATE(), 1);
```

Dead Code: Unused Field (10 issues)

Abstract

This field is never used.

Explanation

This field is never accessed, except perhaps by dead code. Dead code is defined as code that is never directly or indirectly executed by a public method. It is likely that the field is simply vestigial, but it is also possible that the unused field points out a bug.

Example 1: The field named `glue` is not used in the following class. The author of the class has accidentally put quotes around the field name, transforming it into a string constant.

```
public class Dead {

  String glue;

  public String getGlue() {
    return "glue";
  }

}
```

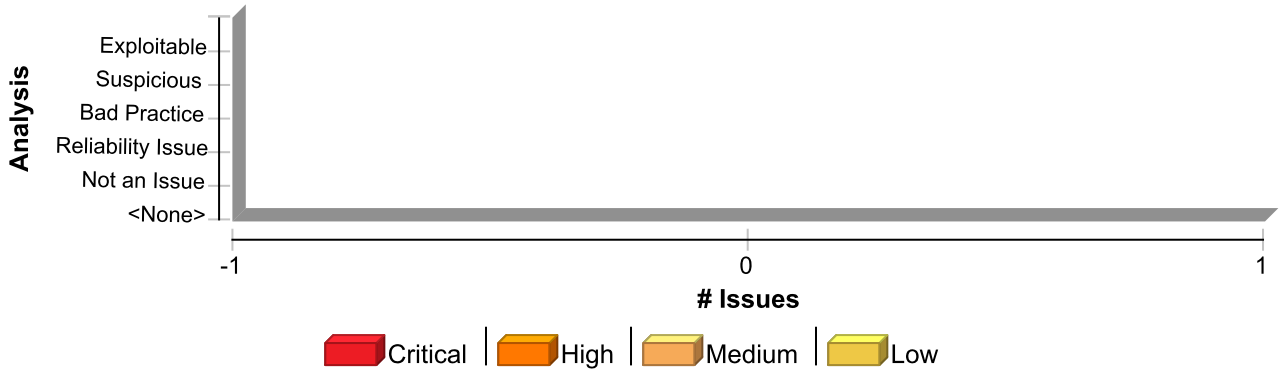
Example 2: The field named `glue` is used in the following class, but only from a method that is never called.

```
public class Dead {

  String glue;

  private String getGlue() {
    return glue;
  }

}
```

Recommendation

In general, you should repair or remove dead code. To repair dead code, execute the dead code directly or indirectly through a public method. Dead code causes additional complexity and maintenance burden without contributing to the functionality of the program.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Dead Code: Unused Field	10	0	0	10
Total	**10**	**0**	**0**	**10**

Dead Code: Unused Field — Low

Package: com.microfocus.example.config

src/main/java/com/microfocus/example/config/WebSecurityConfiguration.java, line 78 (Dead Code: Unused Field)

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: Field: sessionRegistry
File: src/main/java/com/microfocus/example/config/WebSecurityConfiguration.java:78

```
75  }
76
77  @Autowired
78  private SessionRegistry sessionRegistry;
79
80  @Value("${spring.profiles.active:Unknown}")
81  private String activeProfile;
```

src/main/java/com/microfocus/example/config/WebSecurityConfiguration.java, line 64 (Dead Code: Unused Field)

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Dead Code: Unused Field — Low

Package: com.microfocus.example.config

src/main/java/com/microfocus/example/config/WebSecurityConfiguration.java, line 64 (Dead Code: Unused Field)

Sink: Field: basicAuthenticationEntryPoint
File: src/main/java/com/microfocus/example/config/WebSecurityConfiguration.java:64

```
61  private CustomUserDetailsService userDetailsService;
62
63  @Autowired
64  private BasicAuthenticationEntryPointCustom basicAuthenticationEntryPoint;
65
66  @Autowired
67  private ApiAccessDeniedHandler apiAccessDeniedHandler;
```

Package: com.microfocus.example.service

src/main/java/com/microfocus/example/service/UserService.java, line 78 (Dead Code: Unused Field)

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: Field: activeProfile
File: src/main/java/com/microfocus/example/service/UserService.java:78

```
75  private OrderRepository orderRepository;
76
77  @Value("${spring.profiles.active:Unknown}")
78  private String activeProfile;
79
80  @Value("${app.data.page-size:25}")
81  private Integer pageSize;
```

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/EmailUtils.java, line 37 (Dead Code: Unused Field)

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Dead Code: Unused Field — Low

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/EmailUtils.java, line 37 (Dead Code: Unused Field)

Sink: Field: emailServer
File: src/main/java/com/microfocus/example/utils/EmailUtils.java:37

```
34  private final Logger log = LoggerFactory.getLogger(getClass());
35
36  @Value("${spring.mail.host}")
37  private String emailServer;
38
39  private static String EMAIL_SERVER;
40
```

src/main/java/com/microfocus/example/utils/EmailUtils.java, line 48 (Dead Code: Unused Field)

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction	Not Predicted

Sink Details

Sink: Field: emailPort
File: src/main/java/com/microfocus/example/utils/EmailUtils.java:48

```
45  }
46
47  @Value("${spring.mail.port}")
48  private int emailPort;
49
50  private static int EMAIL_PORT;
51
```

src/main/java/com/microfocus/example/utils/EmailUtils.java, line 68 (Dead Code: Unused Field)

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction	Not Predicted

Sink Details

Sink: Field: emailPassword
File: src/main/java/com/microfocus/example/utils/EmailUtils.java:68

Dead Code: Unused Field — Low

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/EmailUtils.java, line 68 (Dead Code: Unused Field)

```
65  }
66
67  @Value("${spring.mail.password}")
68  private String emailPassword;
69
70  private static String EMAIL_PASSWORD;
71
```

src/main/java/com/microfocus/example/utils/EmailUtils.java, line 58 (Dead Code: Unused Field)

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction	Not Predicted

Sink Details

Sink: Field: emailUsername
File: src/main/java/com/microfocus/example/utils/EmailUtils.java:58

```
55  }
56
57  @Value("${spring.mail.username}")
58  private String emailUsername;
59
60  private static String EMAIL_USERNAME;
61
```

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 130 (Dead Code: Unused Field)

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction	Not Predicted

Sink Details

Sink: Field: sessionRegistry
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:130

Dead Code: Unused Field — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 130 (Dead Code: Unused Field)

```
127  EmailSenderService emailSenderService;
128
129  @Autowired
130  private SessionRegistry sessionRegistry;
131
132  @Autowired
133  LocaleConfiguration localeConfiguration;
```

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java, line 89 (Dead Code: Unused Field)

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction	Not Predicted

Sink Details

Sink: Field: storageService
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java:89

```
86  private UserService userService;
87
88  @Autowired
89  private StorageService storageService;
90
91  private String thRCECMD = "";
92
```

Package: com.microfocus.example.web.validation

src/main/java/com/microfocus/example/web/validation/PasswordConstraintValidator.java, line 56 (Dead Code: Unused Field)

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction	Not Predicted

Sink Details

Sink: Field: dictionaryRule
File: src/main/java/com/microfocus/example/web/validation/PasswordConstraintValidator.java:56

Dead Code: Unused Field — Low

Package: com.microfocus.example.web.validation

src/main/java/com/microfocus/example/web/validation/PasswordConstraintValidator.java, line 56 (Dead Code: Unused Field)

```
53
54  private static final Logger log =
LoggerFactory.getLogger(PasswordConstraintValidator.class);
55
56  private DictionaryRule dictionaryRule;
57
58 /*
59  @Value("${app.invalidPasswordList}")
```

Dead Code: Unused Method (1 issue)

Abstract

This method is not reachable from any method outside the class.

Explanation

This method is never called or is only called from other dead code.

Example 1: In the following class, the method `doWork()` can never be called.

```
public class Dead {
  private void doWork() {
    System.out.println("doing work");
  }
  public static void main(String[] args) {
    System.out.println("running Dead");
  }
}
```

Example 2: In the following class, two private methods call each other, but since neither one is ever invoked from anywhere else, they are both dead code.

```
public class DoubleDead {
  private void doTweedledee() {
    doTweedledumb();
  }
  private void doTweedledumb() {
    doTweedledee();
  }
  public static void main(String[] args) {
    System.out.println("running DoubleDead");
  }
}
```

(In this case it is a good thing that the methods are dead: invoking either one would cause an infinite loop.)

Recommendation

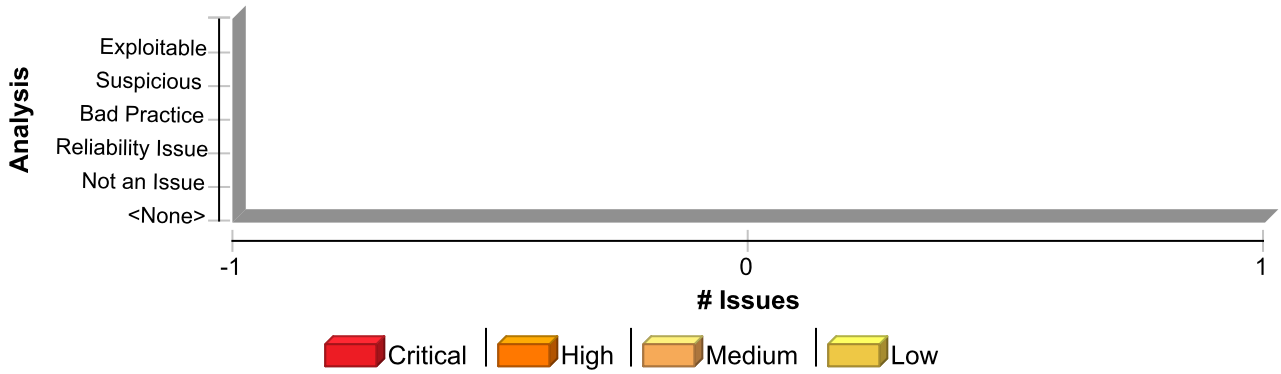
A dead method may indicate a bug in dispatch code.

Example 3: If method is flagged as dead named `getWitch()` in a class that also contains the following dispatch method, it may be because of a copy-and-paste error. The 'w' case should return `getWitch()` not `getMummy()`.

```
public ScaryThing getScaryThing(char st) {
  switch(st) {
    case 'm':
      return getMummy();
    case 'w':
      return getMummy();
    default:
      return getBlob();
  }
}
```

In general, you should repair or remove dead code. To repair dead code, execute the dead code directly or indirectly through a public method. Dead code causes additional complexity and maintenance burden without contributing to the functionality of the program.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Dead Code: Unused Method	1	0	0	1
Total	**1**	**0**	**0**	**1**

Dead Code: Unused Method — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/CustomAuthenticationSuccessHandler.java, line 147 (Dead Code: Unused Method)

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Dead Code: Unused Method — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/CustomAuthenticationSuccessHandler.java, line 147 (Dead Code: Unused Method)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: Function: requestAndRegisterVerification
Enclosing Method: requestAndRegisterVerification()
File: src/main/java/com/microfocus/example/config/handlers/CustomAuthenticationSuccessHandler.java:147

```
144  session.removeAttribute(WebAttributes.AUTHENTICATION_EXCEPTION);
145  }
146
147  private boolean requestAndRegisterVerification(UUID userId) {
148  try {
149  int otp = verificationService.generateOTP(userId.toString());
150  log.debug("Generated OTP '" + String.valueOf(otp) + "' for user id: " +
userId.toString());
```

HTML5: Missing Content Security Policy (3 issues)

Abstract

Content Security Policy (CSP) is not configured.

Explanation

Content Security Policy (CSP) is a declarative security header that enables developers to dictate which domains the site is allowed to load content from or initiate connections to when rendered in the web browser. It provides an additional layer of security from critical vulnerabilities such as cross-site scripting, clickjacking, cross-origin access and the like, on top of input validation and checking an allow list in code.

Spring Security and other frameworks do not add Content Security Policy headers by default. The web application author must declare the security policy/policies to enforce or monitor for the protected resources to benefit from this additional layer of security.

Recommendation

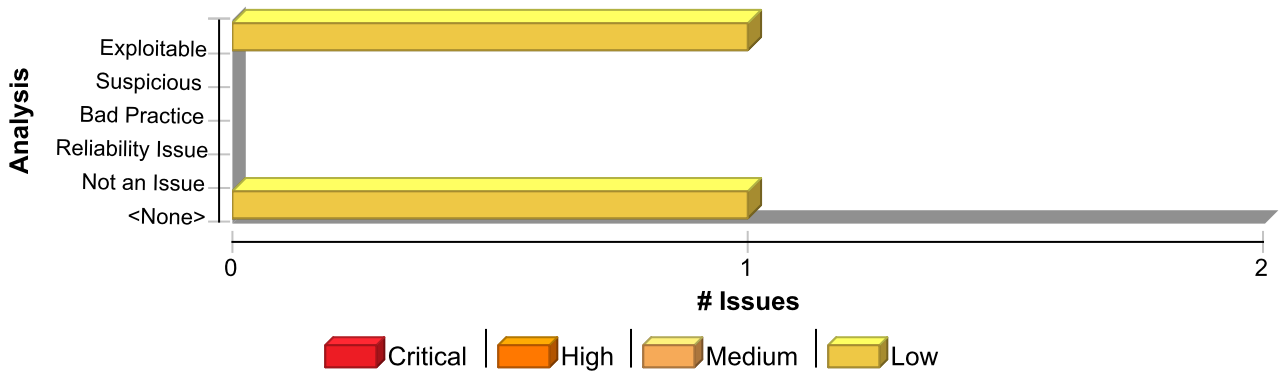
Configure a Content Security Policy to mitigate possible injection vulnerabilities.

Example: The following code sets a Content Security Policy in a Spring Security protected application:

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    ...
    String policy = getCSPolicy();
    http.headers().contentSecurityPolicy(policy);
    ...
}
```

Content Security Policy is not intended to solve all content injection vulnerabilities. Instead, you can leverage CSP to help reduce the harm caused by content injection attacks. Use regular defensive coding,above, current such as input validation and output encoding.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
HTML5: Missing Content Security Policy	2	1	0	3
Total	**2**	**1**	**0**	**3**

HTML5: Missing Content Security Policy — Critical

Package: com.microfocus.example.config

src/main/java/com/microfocus/example/config/WebSecurityConfiguration.java, line 99 (HTML5: Missing Content Security Policy)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: Function: configure
Enclosing Method: configure()
File: src/main/java/com/microfocus/example/config/WebSecurityConfiguration.java:99

```
96  public class ApiConfigurationAdapter extends WebSecurityConfigurerAdapter {
97
98  @Override
99  protected void configure(HttpSecurity httpSecurity) throws Exception {
100
101  /*http.cors().and().csrf().disable()
102  .exceptionHandling().authenticationEntryPoint(unauthorizedHandler).and()
```

src/main/java/com/microfocus/example/config/WebSecurityConfiguration.java, line 139 (HTML5: Missing Content Security Policy)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: Function: configure
Enclosing Method: configure()
File: src/main/java/com/microfocus/example/config/WebSecurityConfiguration.java:139

```
136  public class UserConfigurationAdapter extends WebSecurityConfigurerAdapter {
137
138  @Override
139  protected void configure(HttpSecurity httpSecurity) throws Exception {
140  if (activeProfile.contains("dev")) {
141  log.info("Running development profile");
142  httpSecurity.csrf().disable();
```

HTML5: Missing Content Security Policy — Low

URL: https://iwa.onfortify.com:443/

https://iwa.onfortify.com:443/ (HTML5: Missing Content Security Policy)

Issue Details

Kingdom: Encapsulation
Scan Engine: WEBINSPECT (Dynamic Analysis)

Audit Details

JiraBugLink

Analysis Exploitable

Source Details

URL: https://iwa.onfortify.com:443/
HTTP Method: GET

Request

```
GET / HTTP/1.1
Host: iwa.onfortify.com
User-Agent: Mozilla/5.0 (Windows NT 6.2; Win64; x64; rv:83.0) Gecko/20100101
Firefox/83.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
X-RequestManager-Memo:
stid="25";stmi="0";Category="EventMacro.StartMacro";MacroName="iwa1.webmacro";

Pragma: no-cache
Cookie:
```

HTML5: Missing Framing Protection (1 issue)

Abstract

The application does not restrict browsers from letting third-party sites render its content.

Explanation

Allowing your website to be added to a frame can be a security issue. For example, it may lead to clickjacking vulnerabilities or allow undesired cross-frame communications.

By default, frameworks such as Spring Security include the `X-Frame-Options` header to instruct the browser whether the application should be framed. Disabling or not setting this header can lead to cross-frame related vulnerabilities.

Example 1: The following code configures a Spring Security protected application to disable the `X-Frame-Options` header:

```
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        ...
        http.headers().frameOptions().disable();
        ...
    }
```

Recommendation

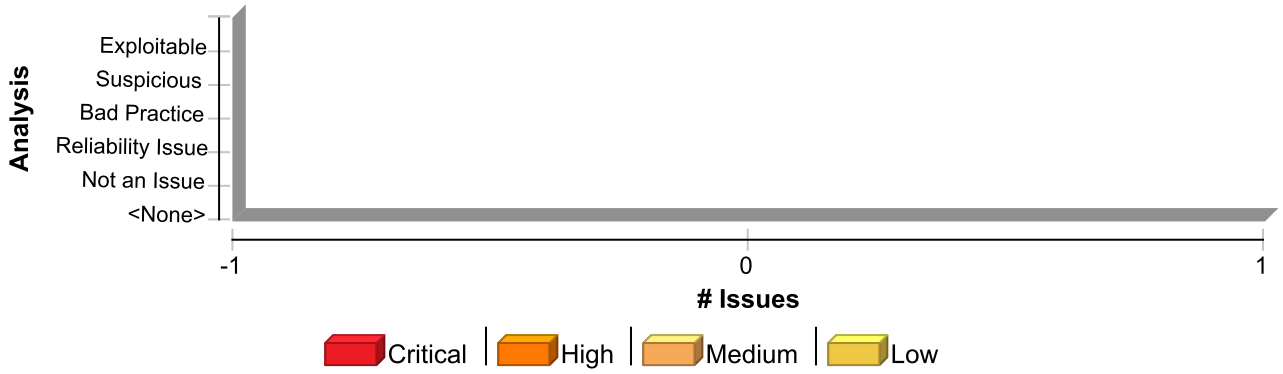
Set the `X-Frame-Options` header as an additional layer of protection. If your application never needs to be framed, set its value to `DENY`, otherwise if it needs to be framed by a different application or page from the same origin, set its value to `SAMEORIGIN`.

By default, Spring Security disables rendering within an `iframe`. You can customize `X-Frame-Options` header to fit your application's requirements.

Example 2: The following code configures a Spring Security protected application to use a `SAMEORIGIN` policy:

```
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        ...
        http.headers().frameOptions().sameOrigin();
        ...
    }
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
HTML5: Missing Framing Protection	1	0	0	1
Total	**1**	**0**	**0**	**1**

HTML5: Missing Framing Protection — Critical

Package: com.microfocus.example.config

src/main/java/com/microfocus/example/config/WebSecurityConfiguration.java, line 143 (HTML5: Missing Framing Protection)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: FunctionCall: disable
Enclosing Method: configure()
File: src/main/java/com/microfocus/example/config/WebSecurityConfiguration.java:143

```
140  if (activeProfile.contains("dev")) {
141  log.info("Running development profile");
142  httpSecurity.csrf().disable();
143  httpSecurity.headers().frameOptions().disable();
144  httpSecurity.cors().disable();
145  httpSecurity.headers().xssProtection();
146  }
```

Header Manipulation (4 issues)

Abstract

Including unvalidated data in an HTTP response header can enable cache-poisoning, cross-site scripting, cross-user defacement, page hijacking, cookie manipulation or open redirect.

Explanation

Header Manipulation vulnerabilities occur when:

1. Data enters a web application through an untrusted source, most frequently an HTTP request.

2. The data is included in an HTTP response header sent to a web user without being validated.

As with many software security vulnerabilities, Header Manipulation is a means to an end, not an end in itself. At its root, the vulnerability is straightforward: an attacker passes malicious data to a vulnerable application, and the application includes the data in an HTTP response header.

One of the most common Header Manipulation attacks is HTTP Response Splitting. To mount a successful HTTP Response Splitting exploit, the application must allow input that contains CR (carriage return, also given by %0d or \r) and LF (line feed, also given by %0a or \n)characters into the header. These characters not only give attackers control of the remaining headers and body of the response the application intends to send, but also allows them to create additional responses entirely under their control.

Many of today's modern application servers will prevent the injection of malicious characters into HTTP headers. For example, recent versions of Apache Tomcat will throw an `IllegalArgumentException` if you attempt to set a header with prohibited characters. If your application server prevents setting headers with new line characters, then your application is not vulnerable to HTTP Response Splitting. However, solely filtering for new line characters can leave an application vulnerable to Cookie Manipulation or Open Redirects, so care must still be taken when setting HTTP headers with user input.

Example: The following code segment reads the name of the author of a weblog entry, `author`, from an HTTP request and sets it in a cookie header of an HTTP response.

```
String author = request.getParameter(AUTHOR_PARAM);
...
Cookie cookie = new Cookie("author", author);
     cookie.setMaxAge(cookieExpiration);
     response.addCookie(cookie);
```

Assuming a string consisting of standard alphanumeric characters, such as "Jane Smith", is submitted in the request the HTTP response including this cookie might take the following form:

```
HTTP/1.1 200 OK
...
Set-Cookie: author=Jane Smith
...
```

However, because the value of the cookie is formed of unvalidated user input the response will only maintain this form if the value submitted for `AUTHOR_PARAM` does not contain any CR and LF characters. If an attacker submits a malicious string, such as "Wiley Hacker\r\nHTTP/1.1 200 OK\r\n...", then the HTTP response would be split into two responses of the following form:

```
HTTP/1.1 200 OK
...
Set-Cookie: author=Wiley Hacker

HTTP/1.1 200 OK
...
```

Clearly, the second response is completely controlled by the attacker and can be constructed with any

header and body content desired. The ability of attacker to construct arbitrary HTTP responses permits a variety of resulting attacks, including: cross-user defacement, web and browser cache poisoning, cross-site scripting, and page hijacking.

Cross-User Defacement: An attacker will be able to make a single request to a vulnerable server that will cause the server to create two responses, the second of which may be misinterpreted as a response to a different request, possibly one made by another user sharing the same TCP connection with the server. This can be accomplished by convincing the user to submit the malicious request themselves, or remotely in situations where the attacker and the user share a common TCP connection to the server, such as a shared proxy server. In the best case, an attacker may leverage this ability to convince users that the application has been hacked, causing users to lose confidence in the security of the application. In the worst case, an attacker may provide specially crafted content designed to mimic the behavior of the application but redirect private information, such as account numbers and passwords, back to the attacker.

Cache Poisoning: The impact of a maliciously constructed response can be magnified if it is cached either by a web cache used by multiple users or even the browser cache of a single user. If a response is cached in a shared web cache, such as those commonly found in proxy servers, then all users of that cache will continue receive the malicious content until the cache entry is purged. Similarly, if the response is cached in the browser of an individual user, then that user will continue to receive the malicious content until the cache entry is purged, although only the user of the local browser instance will be affected.

Cross-Site Scripting: Once attackers have control of the responses sent by an application, they have a choice of a variety of malicious content to provide users. Cross-site scripting is common form of attack where malicious JavaScript or other code included in a response is executed in the user's browser. The variety of attacks based on XSS is almost limitless, but they commonly include transmitting private data such as cookies or other session information to the attacker, redirecting the victim to web content controlled by the attacker, or performing other malicious operations on the user's machine under the guise of the vulnerable site. The most common and dangerous attack vector against users of a vulnerable application uses JavaScript to transmit session and authentication information back to the attacker who can then take complete control of the victim's account.

Page Hijacking: In addition to using a vulnerable application to send malicious content to a user, the same root vulnerability can also be leveraged to redirect sensitive content generated by the server and intended for the user to the attacker instead. By submitting a request that results in two responses, the intended response from the server and the response generated by the attacker, an attacker may cause an intermediate node, such as a shared proxy server, to misdirect a response generated by the server for the user to the attacker. Because the request made by the attacker generates two responses, the first is interpreted as a response to the attacker's request, while the second remains in limbo. When the user makes a legitimate request through the same TCP connection, the attacker's request is already waiting and is interpreted as a response to the victim's request. The attacker then sends a second request to the server, to which the proxy server responds with the server generated request intended for the victim, thereby compromising any sensitive information in the headers or body of the response intended for the victim.

Cookie Manipulation: When combined with attacks like Cross-Site Request Forgery, attackers may change, add to, or even overwrite a legitimate user's cookies.

Open Redirect: Allowing unvalidated input to control the URL used in a redirect can aid phishing attacks.

Recommendation

The solution to prevent Header Manipulation is to ensure that input validation occurs in the required places and checks for the correct properties.

Since Header Manipulation vulnerabilities occur when an application includes malicious data in its output, one logical approach is to validate data immediately before it leaves the application. However, because web applications often have complex and intricate code for generating responses dynamically, this method is prone to errors of omission (missing validation). An effective way to mitigate this risk is to also perform input validation for Header Manipulation.

Web applications must validate all input to prevent other vulnerabilities, such as SQL injection, so augmenting an application's existing input validation mechanism to include checks for Header Manipulation is generally relatively easy. Despite its value, input validation for Header Manipulation does not take the place of rigorous output validation. An application might accept input through a shared data store or other trusted source, and that data store might accept input from a source that does not perform adequate input validation. Therefore, the application cannot implicitly rely on the safety of this or any other data. This means that the best way to prevent Header Manipulation vulnerabilities is to validate everything that enters the application or leaves the application destined for the user.

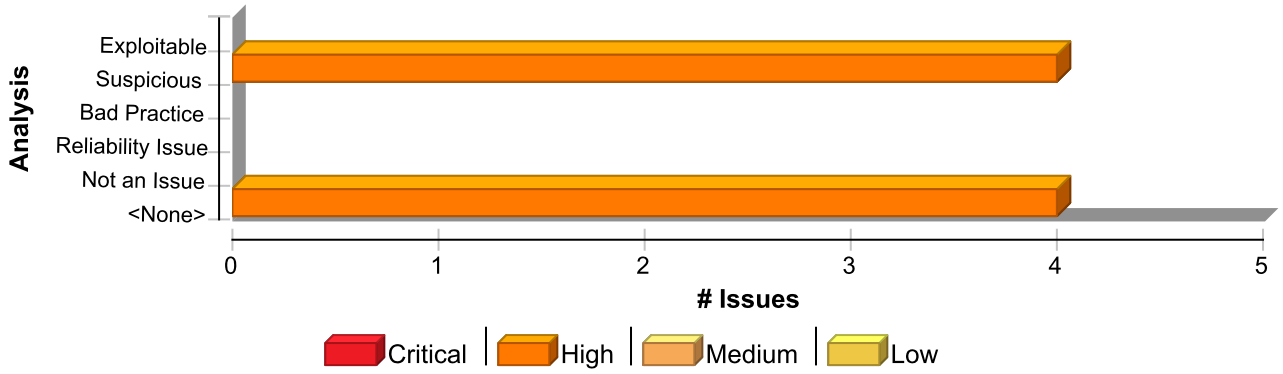
The most secure approach to validation for Header Manipulation is to create an allow list of safe characters that can appear in HTTP response headers and accept input composed exclusively of characters in the approved set. For example, a valid name might only include alphanumeric characters or an account number might only include digits 0-9.

A more flexible, but less secure approach is to implement a deny list, which selectively rejects or escapes potentially dangerous characters before using the input. To form such a list, you first need to understand the set of characters that hold special meaning in HTTP response headers. Although the CR and LF characters are at the heart of an HTTP response splitting attack, other characters, such as ':' (colon) and '=' (equal), have special meaning in response headers as well.

After you identify the correct points in an application to perform validation for Header Manipulation attacks and what special characters the validation should consider, the next challenge is to identify how your validation handles special characters. The application should reject any input destined to be included in HTTP response headers that contains special characters, particularly CR and LF, as invalid.

Many application servers attempt to limit an application's exposure to HTTP response splitting vulnerabilities by providing implementations for the functions responsible for setting HTTP headers and cookies that perform validation for the characters essential to an HTTP response splitting attack. Do not rely on the server running your application to make it secure. For any developed application, there are no guarantees about which application servers it will run on during its lifetime. As standards and known exploits evolve, there are no guarantees that application servers will continue to stay in sync.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Header Manipulation	4	0	0	4
Total	**4**	**0**	**0**	**4**

Header Manipulation — High

URL: null

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 550 (Header Manipulation)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.601

Source Details

Source: serveFile(0)
From: com.microfocus.example.web.controllers.UserController.serveFile
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:546
URL: null

```
543
544   @GetMapping("/files/{filename:.+}")
545   @ResponseBody
546   public ResponseEntity<Resource> serveFile(@PathVariable String filename)
{
547
548   Resource file = storageService.loadAsResource(filename);
549   return ResponseEntity.ok().header(HttpHeaders.CONTENT_DISPOSITION,
```

Sink Details

Header Manipulation — High

URL: null

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 550 (Header Manipulation)

Sink: org.springframework.http.ResponseEntity.HeadersBuilder.header()
Enclosing Method: serveFile()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:550
Taint Flags: WEB, XSS

```
547
548  Resource file = storageService.loadAsResource(filename);
549  return ResponseEntity.ok().header(HttpHeaders.CONTENT_DISPOSITION,
550  "attachment; filename=\"" + file.getFilename() + "\"").body(file);
551  }
552
553  @PostMapping("/files/upload")
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 704 (Header Manipulation)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.668

Source Details

Source: serveUnverifiedFile(0)
From: com.microfocus.example.web.controllers.UserController.serveUnverifiedFile
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:696
URL: null

```
693  }
694
695  @GetMapping("/files/download/unverified")
696  public ResponseEntity<?> serveUnverifiedFile(@Param("file") String file)
{
697
698  if (Objects.isNull(file) || file.isEmpty()) {
699  return ResponseEntity.badRequest().build();
```

Sink Details

Sink: org.springframework.http.ResponseEntity.HeadersBuilder.header()
Enclosing Method: serveUnverifiedFile()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:704
Taint Flags: WEB, XSS

Header Manipulation — High

URL: null

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 704 (Header Manipulation)

```
701
702  Resource rfile = storageService.loadAsResource(file, true);
703  return ResponseEntity.ok().header(HttpHeaders.CONTENT_DISPOSITION,
704  "attachment; filename=\"" + rfile.getFilename() + "\"").body(rfile);
705  }
706
707 }
```

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 170 (Header Manipulation)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.504

Source Details

Source: downloadFile(1)
From: com.microfocus.example.web.controllers.ProductController.downloadFile
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:136
URL: null

```
133
134  @GetMapping("/{id}/download/{fileName:.+}")
135  public ResponseEntity<Resource> downloadFile(@PathVariable(value = "id")
UUID productId,
136  @PathVariable String fileName, HttpServletRequest request) {
137  Resource resource;
138  File dataDir;
139  try {
```

Sink Details

Sink: org.springframework.http.ResponseEntity.HeadersBuilder.header()
Enclosing Method: downloadFile()
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:170
Taint Flags: WEB, XSS

Header Manipulation — High

URL: null

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 170 (Header Manipulation)

```
167  }
168
169  return ResponseEntity.ok().contentType(MediaType.parseMediaType(contentType))
170  .header(HttpHeaders.CONTENT_DISPOSITION, "attachment; filename=\"" +
resource.getFilename() + "\"")
171  .body(resource);
172  }
173
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 622 (Header Manipulation)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.601

Source Details

Source: serveXMLFile(0)
From: com.microfocus.example.web.controllers.UserController.serveXMLFile
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:618
URL: null

```
615
616  @GetMapping("/files/xml/{filename:.+}")
617  @ResponseBody
618  public ResponseEntity<Resource> serveXMLFile(@PathVariable String
filename) {
619
620  Resource file = storageService.loadAsResource(filename);
621  return ResponseEntity.ok().header(HttpHeaders.CONTENT_DISPOSITION,
```

Sink Details

Sink: org.springframework.http.ResponseEntity.HeadersBuilder.header()
Enclosing Method: serveXMLFile()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:622
Taint Flags: WEB, XSS

Header Manipulation — High

URL: null

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 622 (Header Manipulation)

```
619
620  Resource file = storageService.loadAsResource(filename);
621  return ResponseEntity.ok().header(HttpHeaders.CONTENT_DISPOSITION,
622  "attachment; filename=\"" + file.getFilename() + "\"").body(file);
623  }
624
625  @PostMapping("/files/upload-xml")
```

Insecure Randomness (5 issues)

Abstract

Standard pseudorandom number generators cannot withstand cryptographic attacks.

Explanation

Insecure randomness errors occur when a function that can produce predictable values is used as a source of randomness in a security-sensitive context.

Computers are deterministic machines, and as such are unable to produce true randomness. Pseudorandom Number Generators (PRNGs) approximate randomness algorithmically, starting with a seed from which subsequent values are calculated.

There are two types of PRNGs: statistical and cryptographic. Statistical PRNGs provide useful statistical properties, but their output is highly predictable and form an easy to reproduce numeric stream that is unsuitable for use in cases where security depends on generated values being unpredictable. Cryptographic PRNGs address this problem by generating output that is more difficult to predict. For a value to be cryptographically secure, it must be impossible or highly improbable for an attacker to distinguish between the generated random value and a truly random value. In general, if a PRNG algorithm is not advertised as being cryptographically secure, then it is probably a statistical PRNG and should not be used in security-sensitive contexts, where its use can lead to serious vulnerabilities such as easy-to-guess temporary passwords, predictable cryptographic keys, session hijacking, and DNS spoofing.

Example: The following code uses a statistical PRNG to create a URL for a receipt that remains active for some period of time after a purchase.

```
String GenerateReceiptURL(String baseUrl) {
    Random ranGen = new Random();
    ranGen.setSeed((new Date()).getTime());
    return (baseUrl + ranGen.nextInt(400000000) + ".html");
}
```

This code uses the `Random.nextInt()` function to generate "unique" identifiers for the receipt pages it generates. Since `Random.nextInt()` is a statistical PRNG, it is easy for an attacker to guess the strings it generates. Although the underlying design of the receipt system is also faulty, it would be more secure if it used a random number generator that did not produce predictable receipt identifiers, such as a cryptographic PRNG.

Recommendation

When unpredictability is critical, as is the case with most security-sensitive uses of randomness, use a cryptographic PRNG. Regardless of the PRNG you choose, always use a value with sufficient entropy to seed the algorithm. (Do not use values such as the current time because it offers only negligible entropy.)

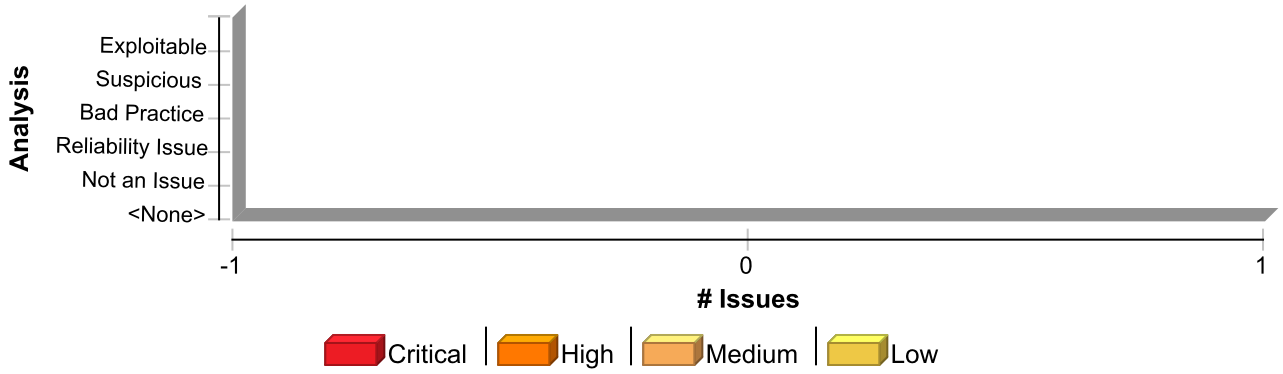
The Java language provides a cryptographic PRNG in `java.security.SecureRandom`. As is the case with other algorithm-based classes in `java.security`, `SecureRandom` provides an implementation-independent wrapper around a particular set of algorithms. When you request an instance of a `SecureRandom` object using `SecureRandom.getInstance()`, you can request a specific implementation of the algorithm. If the algorithm is available, then it is given as a `SecureRandom` object. If it is unavailable or if you do not specify a particular implementation, then you are given a `SecureRandom` implementation selected by the system.

Sun provides a single `SecureRandom` implementation with the Java distribution named `SHA1PRNG`, which Sun describes as computing:

"The SHA-1 hash over a true-random seed value concatenated with a 64-bit counter which is incremented by 1 for each operation. From the 160-bit SHA-1 output, only 64 bits are used [1]."

However, the specifics of the Sun implementation of the `SHA1PRNG` algorithm are poorly documented, and it is unclear what sources of entropy the implementation uses and therefore what amount of true randomness exists in its output. Although there is speculation on the Web about the Sun implementation, there is no evidence to contradict the claim that the algorithm is cryptographically strong and can be used safely in security-sensitive contexts.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Insecure Randomness	5	0	0	5
Total	**5**	**0**	**0**	**5**

Insecure Randomness — High

Package: com.microfocus.example.service

src/main/java/com/microfocus/example/service/UserService.java, line 177 (Insecure Randomness)

Issue Details

Insecure Randomness — High

Package: com.microfocus.example.service

src/main/java/com/microfocus/example/service/UserService.java, line 177 (Insecure Randomness)

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: randomAlphabetic()
Enclosing Method: _registerUser()
File: src/main/java/com/microfocus/example/service/UserService.java:177

```
174  utmp.setPassword(EncryptedPasswordUtils.encryptPassword(password));
175  utmp.setEmail(email);
176  utmp.setPhone(phone);
177  utmp.setVerifyCode(RandomStringUtils.randomAlphabetic(32));
178  utmp.setEnabled(false);
179  utmp.setDateCreated(new Date());
180  utmp.setAuthorities(authorities);
```

src/main/java/com/microfocus/example/service/VerificationService.java, line 61 (Insecure Randomness)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: nextInt()
Enclosing Method: generateOTP()
File: src/main/java/com/microfocus/example/service/VerificationService.java:61

```
58
59  public int generateOTP(String key){
60  Random random = new Random();
61  int otp = 100000 + random.nextInt(900000);
62  otpCache.put(key, otp);
63  return otp;
64  }
```

src/main/java/com/microfocus/example/service/ProductService.java, line 293 (Insecure Randomness)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Insecure Randomness — High

Package: com.microfocus.example.service

src/main/java/com/microfocus/example/service/ProductService.java, line 293 (Insecure Randomness)

Audit Details

AA_Prediction	Not Predicted

Sink Details

Sink: nextInt()
Enclosing Method: newOrderFromOrderForm()
File: src/main/java/com/microfocus/example/service/ProductService.java:293

```
290  Random r = new Random();
291  int low = 10;
292  int high = 100;
293  int result = r.nextInt(high-low) + low;
294  String formatted = String.format("%03d", result);
295  otmp.setOrderNum("OID-P100-"+formatted);
296  Order newOrder = orderRepository.saveAndFlush(otmp);
```

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/AdminUtils.java, line 120 (Insecure Randomness)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Audit Details

AA_Prediction	Not Predicted

Sink Details

Sink: nextInt()
Enclosing Method: genId()
File: src/main/java/com/microfocus/example/utils/AdminUtils.java:120

```
117
118  Random r=new Random();
119  r.setSeed(12345);
120  return r.nextInt();
121  }
122
123  private static String isLocked(int backupId) {
```

Package: src.main.resources.static.js.components

src/main/resources/static/js/components/SubscribeNewsletter.js, line 48 (Insecure Randomness)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Insecure Randomness — High

Package: src.main.resources.static.js.components

src/main/resources/static/js/components/SubscribeNewsletter.js, line 48 (Insecure Randomness)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: FunctionPointerCall: random
Enclosing Method: _saveEmail()
File: src/main/resources/static/js/components/SubscribeNewsletter.js:48

```
45  }
46
47  async function _saveEmail(email) {
48  let subscriberId = Math.random() * (MAX_SUBSCRIBER_ID - MIN_SUBSCRIBER_ID) +
MIN_SUBSCRIBER_ID;
49  let data = JSON.stringify(
50  {
51  id: subscriberId,
```

Insecure Randomness: Hardcoded Seed (1 issue)

Abstract

Functions that generate random or pseudorandom values, which are passed a seed, should not be called with a constant argument.

Explanation

Functions that generate random or pseudorandom values, which are passed a seed, should not be called with a constant argument. If a pseudorandom number generator (such as `Random`) is seeded with a specific value (using a function such as `Random.setSeed()`), the values returned by `Random.nextInt()` and similar methods which return or assign values are predictable for an attacker that can collect a number of PRNG outputs.

Example 1: The values produced by the `Random` object `randomGen2` are predictable from the `Random` object `randomGen1`.

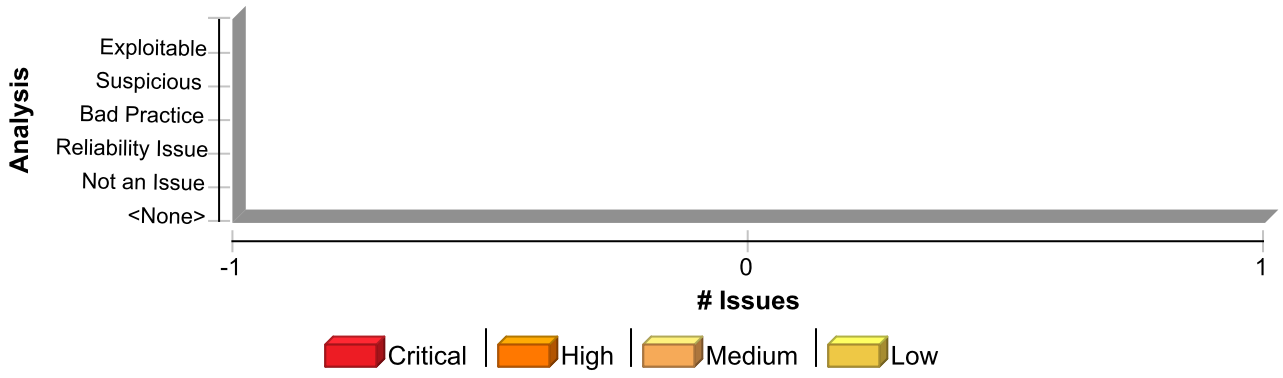
```
Random randomGen1 = new Random();
randomGen1.setSeed(12345);
int randomInt1 = randomGen1.nextInt();
byte[] bytes1 = new byte[4];
randomGen1.nextBytes(bytes1);

Random randomGen2 = new Random();
randomGen2.setSeed(12345);
int randomInt2 = randomGen2.nextInt();
byte[] bytes2 = new byte[4];
randomGen2.nextBytes(bytes2);
```

In this example, pseudorandom number generators: `randomGen1` and `randomGen2` were identically seeded, so `randomInt1 == randomInt2`, and corresponding values of arrays `bytes1[]` and `bytes2[]` are equal.

Recommendation

Use a cryptographic PRNG seeded with hardware-based sources of randomness, such as ring oscillators, disk drive timing, thermal noise, or radioactive decay. Doing so makes the sequence of data produced by `Random.nextInt()` and similar methods much harder to predict than setting the seed to a constant.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Insecure Randomness: Hardcoded Seed	1	0	0	1
Total	**1**	**0**	**0**	**1**

Insecure Randomness: Hardcoded Seed — High

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/AdminUtils.java, line 119 (Insecure Randomness: Hardcoded Seed)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: setSeed()
Enclosing Method: genId()
File: src/main/java/com/microfocus/example/utils/AdminUtils.java:119

```
116  */
117
118  Random r=new Random();
119  r.setSeed(12345);
120  return r.nextInt();
121  }
122
```

Insecure SSL: Server Identity Verification Disabled (1 issue)

Abstract

Server identity verification is disabled when making SSL connections.

Explanation

In some libraries that use SSL connections, the server certificate is not verified by default. This is equivalent to trusting all certificates.

Example 1: This application does not explicitly verify the server certificate.

```
...
Email email = new SimpleEmail();
email.setHostName("smtp.servermail.com");
email.setSmtpPort(465);
email.setAuthenticator(new DefaultAuthenticator(username, password));
email.setSSLOnConnect(true);
email.setFrom("user@gmail.com");
email.setSubject("TestMail");
email.setMsg("This is a test mail ... :-)");
email.addTo("foo@bar.com");
email.send();
...
```

When trying to connect to `smtp.mailserver.com:465`, this application would readily accept a certificate issued to "`hackedserver.com`". The application would now potentially leak sensitive user information on a broken SSL connection to the hacked server.

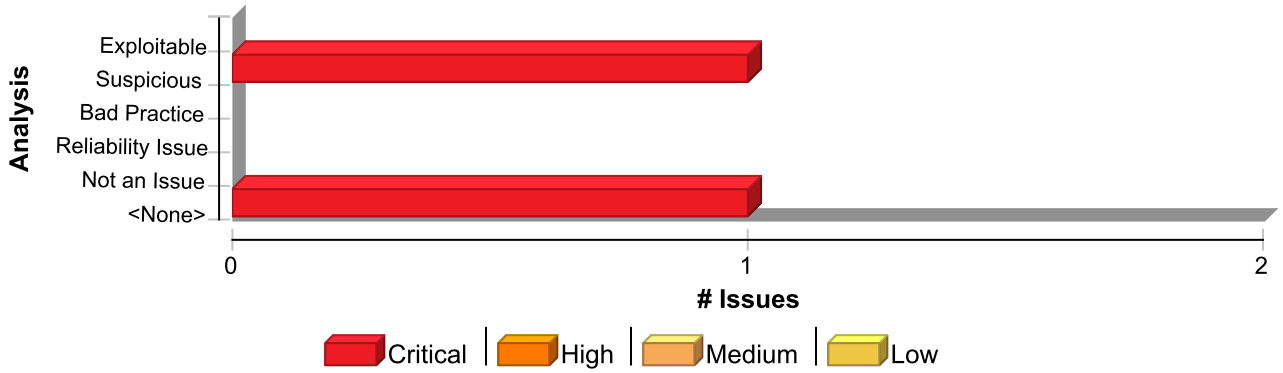
Recommendation

Do not forget server verification checks when making SSL connections. Depending on the library used, make sure to verify server identity and establish a secure SSL connection.

Example 2: This application does explicitly verify the server certificate.

```
...
Email email = new SimpleEmail();
email.setHostName("smtp.servermail.com");
email.setSmtpPort(465);
email.setAuthenticator(new DefaultAuthenticator(username, password));
email.setSSLCheckServerIdentity(true);
email.setSSLOnConnect(true);
email.setFrom("user@gmail.com");
email.setSubject("TestMail");
email.setMsg("This is a test mail ... :-)");
email.addTo("foo@bar.com");
email.send();
...
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Insecure SSL: Server Identity Verification Disabled	1	0	0	1
Total	**1**	**0**	**0**	**1**

Insecure SSL: Server Identity Verification Disabled — Critical

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/EmailUtils.java, line 95 (Insecure SSL: Server Identity Verification Disabled)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Control Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.581

Sink Details

Sink: server.send() : Mail sent to unverified server
Enclosing Method: sendEmail()
File: src/main/java/com/microfocus/example/utils/EmailUtils.java:95

```
92  server.setSubject(request.getSubject());
93  server.setMsg(request.getBody());
94  server.setBounceAddress(request.getBounce());
95  server.send();
96  }
97 }
98
```

J2EE Bad Practices: Non-Serializable Object Stored in Session (2 issues)

Abstract

Storing a non-serializable object as an `HttpSession` attribute can damage application reliability.

Explanation

A J2EE application can make use of multiple JVMs in order to improve application reliability and performance. In order to make the multiple JVMs appear as a single application to the end user, the J2EE container can replicate an `HttpSession` object across multiple JVMs so that if one JVM becomes unavailable another can step in and take its place without disrupting the flow of the application.

In order for session replication to work, the values the application stores as attributes in the session must implement the `Serializable` interface.

Example 1: The following class adds itself to the session, but because it is not serializable, the session can no longer be replicated.

```
public class DataGlob {
   String globName;
   String globValue;

   public void addToSession(HttpSession session) {
     session.setAttribute("glob", this);
   }
}
```

Recommendation

In many cases, the easiest way to fix this problem is simply to have the offending object implement the `Serializable` interface.

Example 2: The code in `Example 1` could be rewritten in the following way:

```
public class DataGlob implements java.io.Serializable {
   String globName;
   String globValue;

   public void addToSession(HttpSession session) {
     session.setAttribute("glob", this);
   }
}
```

Note that for complex objects, the transitive closure of the objects stored in the session must be serializable. If object A references object B and object A is stored in the session, then both A and B must implement `Serializable`.

While implementing the `Serializable` interface is often easy (since the interface does not force the class to define any methods), some types of objects will cause complications. Watch out for objects that hold references to external resources. For example, both streams and JNI are likely to cause complications.

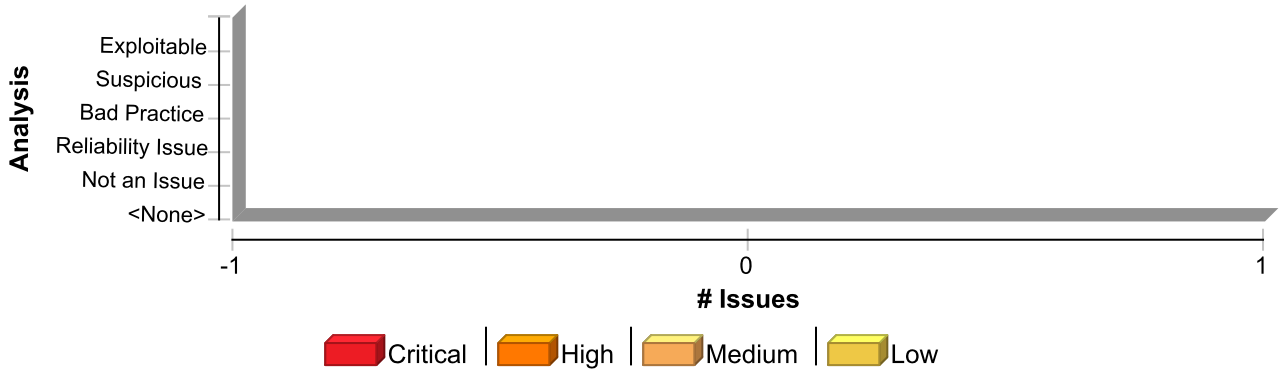
Example 3: Use type checking to require serializable objects. Instead of this:

```
public static void addToSession(HttpServletRequest req,
                     String attrib, Object obj)
{
  HttpSession sess = req.getSession(true);
  sess.setAttribute(attrib, obj);
}
```

write this:

```
public static void addToSession(HttpServletRequest req,
                     String attrib, Serializable ser) {
  HttpSession sess = req.getSession(true);
  sess.setAttribute(attrib, ser);
}
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
J2EE Bad Practices: Non-Serializable Object Stored in Session	2	0	0	2
Total	**2**	**0**	**0**	**2**

J2EE Bad Practices: Non-Serializable Object Stored in Session — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/UrlAuthenticationSuccessHandler.java, line 69 (J2EE Bad Practices: Non-Serializable Object Stored in Session)

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: FunctionCall: setAttribute
Enclosing Method: onAuthenticationSuccess()
File: src/main/java/com/microfocus/example/config/handlers/UrlAuthenticationSuccessHandler.java:69

```
66  log.debug("Generated jwtToken: " + jwtToken);
67  session.setAttribute("userId", user.getId());
68  session.setAttribute("username", user.getUsername());
69  session.setAttribute("authorities", authentication.getAuthorities());
70  session.setAttribute("jwtToken", jwtToken);
71
72  handle(request, response, authentication);
```

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/JwtUtils.java, line 111 (J2EE Bad Practices: Non-Serializable Object Stored in Session)

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Structural)

J2EE Bad Practices: Non-Serializable Object Stored in Session — Low

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/JwtUtils.java, line 111 (J2EE Bad Practices: Non-Serializable Object Stored in Session)

Audit Details

AA_Prediction	Not Predicted

Sink Details

Sink: FunctionCall: setAttribute
Enclosing Method: generateAndSetSession()
File: src/main/java/com/microfocus/example/utils/JwtUtils.java:111

```
108  log.debug("Generated jwtToken: " + jwtToken);
109  session.setAttribute("userId", user.getId());
110  session.setAttribute("username", user.getUsername());
111  session.setAttribute("authorities", authentication.getAuthorities());
112  session.setAttribute("jwtToken", jwtToken);
113  return jwtToken;
114  }
```

JSON Injection (5 issues)

Abstract

The method writes unvalidated input into JSON. This call could allow an attacker to inject arbitrary elements or attributes into the JSON entity.

Explanation

JSON injection occurs when:

1. Data enters a program from an untrusted source.

2. The data is written to a JSON stream.

Applications typically use JSON to store data or send messages. When used to store data, JSON is often treated like cached data and may potentially contain sensitive information. When used to send messages, JSON is often used in conjunction with a RESTful service and can be used to transmit sensitive information such as authentication credentials.

The semantics of JSON documents and messages can be altered if an application constructs JSON from unvalidated input. In a relatively benign case, an attacker may be able to insert extraneous elements that cause an application to throw an exception while parsing a JSON document or request. In a more serious case, such as ones that involves JSON injection, an attacker may be able to insert extraneous elements that allow for the predictable manipulation of business critical values within a JSON document or request. In some cases, JSON injection can lead to cross-site scripting or dynamic code evaluation.

Example 1: The following Java code uses Jackson to serialize user account authentication information for non-privileged users (those with a role of "default" as opposed to privileged users with a role of "admin") from user-controlled input variables `username` and `password` to the JSON file located at `~/user_info.json`:

```
...

JsonFactory jfactory = new JsonFactory();

JsonGenerator jGenerator = jfactory.createJsonGenerator(new File("~/
user_info.json"), JsonEncoding.UTF8);

jGenerator.writeStartObject();

jGenerator.writeFieldName("username");
jGenerator.writeRawValue("\"" + username + "\"");

jGenerator.writeFieldName("password");
jGenerator.writeRawValue("\"" + password + "\"");

jGenerator.writeFieldName("role");
jGenerator.writeRawValue("\"default\"");

jGenerator.writeEndObject();

jGenerator.close();
```

Yet, because the JSON serialization is performed using `JsonGenerator.writeRawValue()`, the untrusted data in `username` and `password` will not be validated to escape JSON-related special characters. This allows a user to arbitrarily insert JSON keys, possibly changing the structure of the serialized JSON. In this example, if the non-privileged user `mallory` with password `Evil123!` were to append `","role":"admin` to her username when entering it at the prompt that sets the value of the `username` variable, the resulting JSON saved to `~/user_info.json` would be:

```
{
  "username":"mallory",
  "role":"admin",
  "password":"Evil123!",
```

```
  "role":"default"
}
```

If this serialized JSON file were then deserialized to an `HashMap` object with Jackson's `JsonParser` as so:

```
JsonParser jParser = jfactory.createJsonParser(new File("~/user_info.json"));

while (jParser.nextToken() != JsonToken.END_OBJECT) {

  String fieldname = jParser.getCurrentName();

  if ("username".equals(fieldname)) {
    jParser.nextToken();
    userInfo.put(fieldname, jParser.getText());
  }

  if ("password".equals(fieldname)) {
    jParser.nextToken();
    userInfo.put(fieldname, jParser.getText());
  }

  if ("role".equals(fieldname)) {
    jParser.nextToken();
    userInfo.put(fieldname, jParser.getText());
  }

  if (userInfo.size() == 3)
    break;
}

jParser.close();
```

The resulting values for the `username`, `password`, and `role` keys in the `HashMap` object would be `mallory`, `Evil123!`, and `admin` respectively. Without further verification that the deserialized JSON values are valid, the application will incorrectly assign user `mallory` "admin" privileges.

Recommendation

When writing user supplied data to JSON, follow these guidelines:

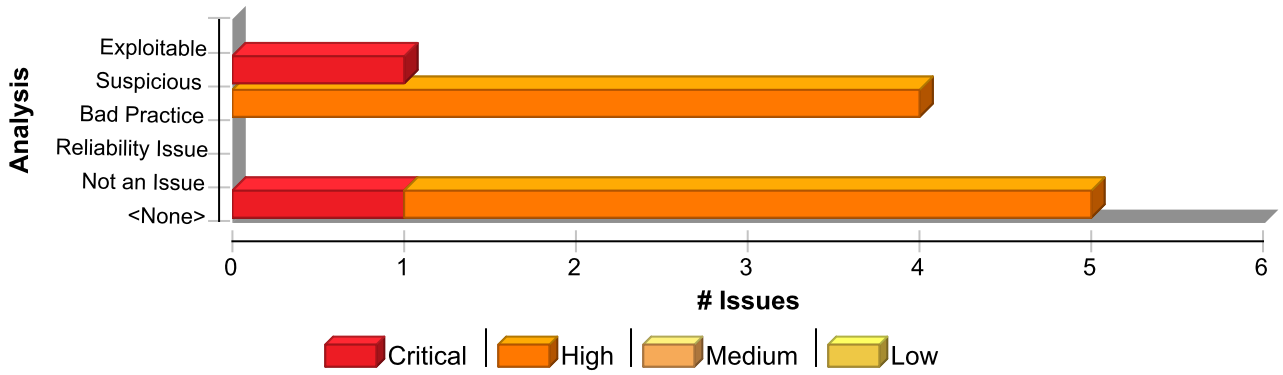
1. Do not create JSON attributes with names that are derived from user input.

2. Ensure that all serialization to JSON is performed using a safe serialization function that delimits untrusted data within single or double quotes and escapes any special characters.

Example 2: The following Java code implements the same functionality as that in `Example 1`, but instead uses `JsonGenerator.writeString()` rather than `JsonGenerator.writeRawValue()` to serialize the data, therefore ensuring that any untrusted data is properly delimited and escaped:

```
...

JsonFactory jfactory = new JsonFactory();

JsonGenerator jGenerator = jfactory.createJsonGenerator(new File("~/
user_info.json"), JsonEncoding.UTF8);

jGenerator.writeStartObject();

jGenerator.writeFieldName("username");
jGenerator.writeString(username);

jGenerator.writeFieldName("password");
jGenerator.writeString(password);

jGenerator.writeFieldName("role");
jGenerator.writeString("default");

jGenerator.writeEndObject();

jGenerator.close();
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
JSON Injection	5	0	0	5
Total	**5**	**0**	**0**	**5**

JSON Injection — Critical

URL: null

src/main/java/com/microfocus/example/utils/UserUtils.java, line 115 (JSON Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.779

Source Details

Source: subscribeUser(0)
From: com.microfocus.example.api.controllers.ApiSiteController.subscribeUser
File: src/main/java/com/microfocus/example/api/controllers/ApiSiteController.java:194
URL: null

```
191  @PostMapping(value = {"/subscribe-user"}, produces = {"application/json"}, consumes = {"application/json"})
192  @ResponseStatus(HttpStatus.OK)
193  public ResponseEntity<ApiStatusResponse> subscribeUser(
194  @io.swagger.v3.oas.annotations.parameters.RequestBody(description = "") @Valid @RequestBody SubscribeUserRequest newUser) {
195  log.debug("API::Subscribing a user to the newsletter: " + newUser.toString());
196  SubscribeUserResponse user = userService.subscribeUser(newUser);
197  ApiStatusResponse response = new ApiStatusResponse();
```

Sink Details

Sink: com.fasterxml.jackson.core.JsonGenerator.writeRawValue()
Enclosing Method: registerUser()
File: src/main/java/com/microfocus/example/utils/UserUtils.java:115
Taint Flags: WEB, XSS

```
112  jGenerator.writeFieldName("lastName");
113  jGenerator.writeRawValue("\"" + lastName + "\"");
114  jGenerator.writeFieldName("email");
115  jGenerator.writeRawValue("\"" + email + "\"");
116  jGenerator.writeFieldName("role");
117  jGenerator.writeRawValue("\"" + DEFAULT_ROLE + "\"");
118  jGenerator.writeEndObject();
```

JSON Injection — High

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/UserUtils.java, line 101 (JSON Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.584

Source Details

Source: java.io.FileReader.FileReader()
From: com.microfocus.example.utils.UserUtils.registerUser
File: src/main/java/com/microfocus/example/utils/UserUtils.java:81

```
78
79  File dataFile = new File(getFilePath(NEWSLETTER_USER_FILE));
80  if (dataFile.exists()) {
81  jsonArray = (JSONArray) jsonParser.parse(new
FileReader(getFilePath(NEWSLETTER_USER_FILE)));
82  } else {
83  dataFile.createNewFile();
84  log.debug("Created: " + getFilePath(NEWSLETTER_USER_FILE));
```

Sink Details

Sink: com.fasterxml.jackson.core.JsonGenerator.writeRawValue()
Enclosing Method: registerUser()
File: src/main/java/com/microfocus/example/utils/UserUtils.java:101
Taint Flags: FILE_SYSTEM

```
98  jGenerator.writeFieldName("lastName");
99  jGenerator.writeRawValue("\"" + (String) person.get("lastName") + "\"");
100  jGenerator.writeFieldName("email");
101  jGenerator.writeRawValue("\"" + (String) person.get("email") + "\"");
102  jGenerator.writeFieldName("role");
103  jGenerator.writeRawValue("\"" + (String) person.get("role") + "\"");
104  jGenerator.writeEndObject();
```

src/main/java/com/microfocus/example/utils/UserUtils.java, line 99 (JSON Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice

JSON Injection — High

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/UserUtils.java, line 99 (JSON Injection)

AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.584

Source Details

Source: java.io.FileReader.FileReader()
From: com.microfocus.example.utils.UserUtils.registerUser
File: src/main/java/com/microfocus/example/utils/UserUtils.java:81

```
78
79  File dataFile = new File(getFilePath(NEWSLETTER_USER_FILE));
80  if (dataFile.exists()) {
81  jsonArray = (JSONArray) jsonParser.parse(new
FileReader(getFilePath(NEWSLETTER_USER_FILE)));
82  } else {
83  dataFile.createNewFile();
84  log.debug("Created: " + getFilePath(NEWSLETTER_USER_FILE));
```

Sink Details

Sink: com.fasterxml.jackson.core.JsonGenerator.writeRawValue()
Enclosing Method: registerUser()
File: src/main/java/com/microfocus/example/utils/UserUtils.java:99
Taint Flags: FILE_SYSTEM

```
96  jGenerator.writeFieldName("firstName");
97  jGenerator.writeRawValue("\"" + (String) person.get("firstName") + "\"");
98  jGenerator.writeFieldName("lastName");
99  jGenerator.writeRawValue("\"" + (String) person.get("lastName") + "\"");
100  jGenerator.writeFieldName("email");
101  jGenerator.writeRawValue("\"" + (String) person.get("email") + "\"");
102  jGenerator.writeFieldName("role");
```

src/main/java/com/microfocus/example/utils/UserUtils.java, line 97 (JSON Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.584

Source Details

JSON Injection — High

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/UserUtils.java, line 97 (JSON Injection)

Source: java.io.FileReader.FileReader()
From: com.microfocus.example.utils.UserUtils.registerUser
File: src/main/java/com/microfocus/example/utils/UserUtils.java:81

```
78
79  File dataFile = new File(getFilePath(NEWSLETTER_USER_FILE));
80  if (dataFile.exists()) {
81  jsonArray = (JSONArray) jsonParser.parse(new
FileReader(getFilePath(NEWSLETTER_USER_FILE)));
82  } else {
83  dataFile.createNewFile();
84  log.debug("Created: " + getFilePath(NEWSLETTER_USER_FILE));
```

Sink Details

Sink: com.fasterxml.jackson.core.JsonGenerator.writeRawValue()
Enclosing Method: registerUser()
File: src/main/java/com/microfocus/example/utils/UserUtils.java:97
Taint Flags: FILE_SYSTEM

```
94  jGenerator.writeStartObject();
95  JSONObject person = (JSONObject) jsonObject;
96  jGenerator.writeFieldName("firstName");
97  jGenerator.writeRawValue("\"" + (String) person.get("firstName") + "\"");
98  jGenerator.writeFieldName("lastName");
99  jGenerator.writeRawValue("\"" + (String) person.get("lastName") + "\"");
100  jGenerator.writeFieldName("email");
```

src/main/java/com/microfocus/example/utils/UserUtils.java, line 103 (JSON Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.584

Source Details

Source: java.io.FileReader.FileReader()
From: com.microfocus.example.utils.UserUtils.registerUser
File: src/main/java/com/microfocus/example/utils/UserUtils.java:81

JSON Injection — High

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/UserUtils.java, line 103 (JSON Injection)

```
78
79  File dataFile = new File(getFilePath(NEWSLETTER_USER_FILE));
80  if (dataFile.exists()) {
81  jsonArray = (JSONArray) jsonParser.parse(new
FileReader(getFilePath(NEWSLETTER_USER_FILE)));
82  } else {
83  dataFile.createNewFile();
84  log.debug("Created: " + getFilePath(NEWSLETTER_USER_FILE));
```

Sink Details

Sink: com.fasterxml.jackson.core.JsonGenerator.writeRawValue()
Enclosing Method: registerUser()
File: src/main/java/com/microfocus/example/utils/UserUtils.java:103
Taint Flags: FILE_SYSTEM

```
100  jGenerator.writeFieldName("email");
101  jGenerator.writeRawValue("\"" + (String) person.get("email") + "\"");
102  jGenerator.writeFieldName("role");
103  jGenerator.writeRawValue("\"" + (String) person.get("role") + "\"");
104  jGenerator.writeEndObject();
105
106  }
```

Key Management: Hardcoded Encryption Key (1 issue)

Abstract

Hardcoded encryption keys can compromise security in a way that cannot be easily remedied.

Explanation

It is never a good idea to hardcode an encryption key because it allows all of the project's developers to view the encryption key, and makes fixing the problem extremely difficult. After the code is in production, a software patch is required to change the encryption key. If the account that is protected by the encryption key is compromised, the owners of the system must choose between security and availability.

Example 1: The following code uses a hardcoded encryption key:

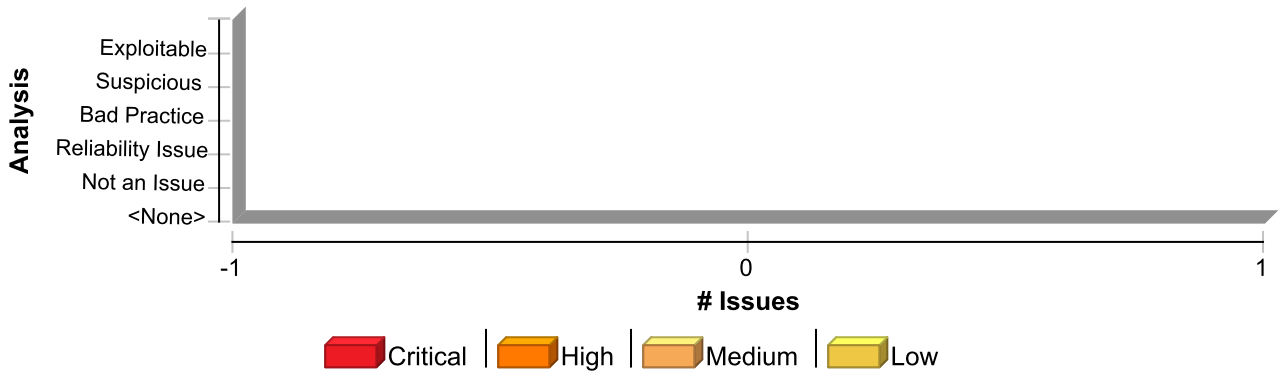
```
...
private static final String encryptionKey = "lakdsljkalkjlksdfkl";
byte[] keyBytes = encryptionKey.getBytes();
SecretKeySpec key = new SecretKeySpec(keyBytes, "AES");
Cipher encryptCipher = Cipher.getInstance("AES");
encryptCipher.init(Cipher.ENCRYPT_MODE, key);
...
```

Anyone with access to the code has access to the encryption key. After the application has shipped, there is no way to change the encryption key unless the program is patched. An employee with access to this information can use it to break into the system. If attackers had access to the executable for the application, they could extract the encryption key value.

Recommendation

Encryption keys should never be hardcoded and should be obfuscated and managed in an external source. Storing encryption keys in plain text anywhere on the system allows anyone with sufficient permissions to read and potentially misuse the encryption key.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Key Management: Hardcoded Encryption Key	1	0	0	1
Total	**1**	**0**	**0**	**1**

Key Management: Hardcoded Encryption Key — High

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/EncryptedPasswordUtils.java, line 41 (Key Management: Hardcoded Encryption Key)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction	Not Predicted

Sink Details

Sink: FunctionCall: SecretKeySpec
Enclosing Method: ()
File: src/main/java/com/microfocus/example/utils/EncryptedPasswordUtils.java:41

```
38 public class EncryptedPasswordUtils {
39
40  private static final byte[] iv = { 22, 33, 11, 44, 55, 99, 66, 77 };
41  private static final SecretKey keySpec = new SecretKeySpec(iv, "DES");
42
43  public static String encryptPassword(String password) {
44  byte[] encrypted = null;
```

Log Forging (10 issues)

Abstract

Writing unvalidated user input to log files can allow an attacker to forge log entries or inject malicious content into the logs.

Explanation

Log forging vulnerabilities occur when:

1. Data enters an application from an untrusted source.

2. The data is written to an application or system log file.

Applications typically use log files to store a history of events or transactions for later review, statistics gathering, or debugging. Depending on the nature of the application, the task of reviewing log files may be performed manually on an as-needed basis or automated with a tool that automatically culls logs for important events or trending information.

Interpretation of the log files may be hindered or misdirected if an attacker can supply data to the application that is subsequently logged verbatim. In the most benign case, an attacker may be able to insert false entries into the log file by providing the application with input that includes appropriate characters. If the log file is processed automatically, the attacker may be able to render the file unusable by corrupting the format of the file or injecting unexpected characters. A more subtle attack might involve skewing the log file statistics. Forged or otherwise, corrupted log files can be used to cover an attacker's tracks or even to implicate another party in the commission of a malicious act [1]. In the worst case, an attacker may inject code or other commands into the log file and take advantage of a vulnerability in the log processing utility [2].

Example 1: The following web application code attempts to read an integer value from a request object. If the value fails to parse as an integer, then the input is logged with an error message indicating what happened.

```
...
        String val = request.getParameter("val");
        try {
                int value = Integer.parseInt(val);
        }
        catch (NumberFormatException nfe) {
                log.info("Failed to parse val = " + val);
        }
...
```

If a user submits the string "`twenty-one`" for `val`, the following entry is logged:

```
INFO: Failed to parse val=twenty-one
```

However, if an attacker submits the string "`twenty-one%0a%0aINFO:+User+logged+out%3dbadguy`", the following entry is logged:

```
INFO: Failed to parse val=twenty-one

INFO: User logged out=badguy
```

Clearly, attackers may use this same mechanism to insert arbitrary log entries.

Some think that in the mobile world, classic web application vulnerabilities, such as log forging, do not make sense -- why would the user attack themself? However, keep in mind that the essence of mobile platforms is applications that are downloaded from various sources and run alongside each other on the same device. The likelihood of running a piece of malware next to a banking application is high, which necessitates expanding the attack surface of mobile applications to include inter-process communication.

Example 2: The following code adapts `Example 1` to the Android platform.

```
...
        String val = this.getIntent().getExtras().getString("val");
        try {
                int value = Integer.parseInt();
        }
        catch (NumberFormatException nfe) {
                Log.e(TAG, "Failed to parse val = " + val);
        }
...
```

Recommendation

Prevent log forging attacks with indirection: create a set of legitimate log entries that correspond to different events that must be logged and only log entries from this set. To capture dynamic content, such as users logging out of the system, always use server-controlled values rather than user-supplied data. This ensures that the input provided by the user is never used directly in a log entry.

Example 1 can be rewritten to use a pre-defined log entry that corresponds to a `NumberFormatException` as follows:

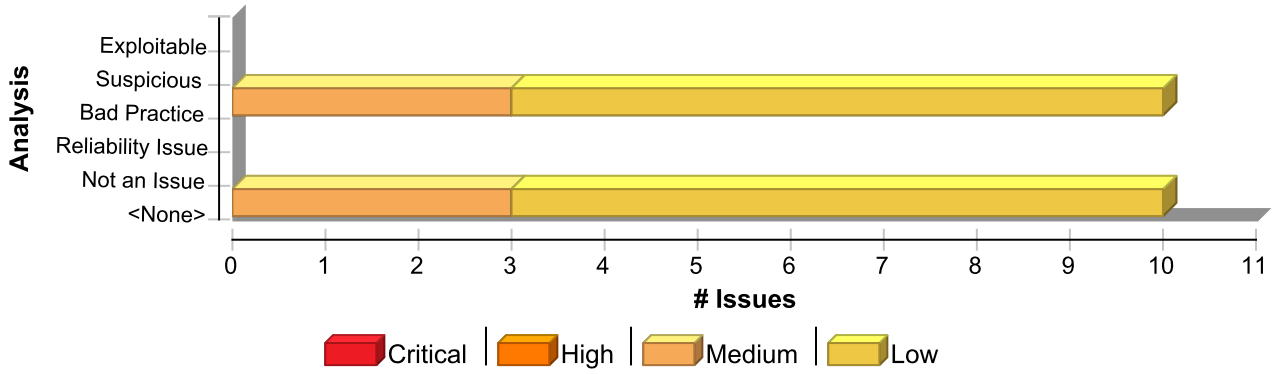
```
...
        public static final String NFE = "Failed to parse val. The input is
required to be an integer value."
...
        String val = request.getParameter("val");
        try {
                int value = Integer.parseInt(val);
        }
        catch (NumberFormatException nfe) {
                log.info(NFE);
        }
..
```

And here is an Android equivalent:

```
...
        public static final String NFE = "Failed to parse val. The input is
required to be an integer value."
...
        String val = this.getIntent().getExtras().getString("val");
        try {
                int value = Integer.parseInt();
        }
        catch (NumberFormatException nfe) {
                Log.e(TAG, NFE);
        }
...
```

In some situations this approach is impractical because the set of legitimate log entries is too large or complicated. In these situations, developers often fall back on implementing a deny list. A deny list is used to selectively reject or escape potentially dangerous characters before using the input. However, a list of unsafe characters can quickly become incomplete or outdated. A better approach is to create a list of characters that are permitted to appear in log entries and accept input composed exclusively of characters in the approved set. The most critical character in most log forging attacks is the '\n' (newline) character, which should never appear on a log entry allow list.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Log Forging	10	0	0	10
Total	**10**	**0**	**0**	**10**

Log Forging — Medium

URL: null

src/main/java/com/microfocus/example/utils/AdminUtils.java, line 67 (Log Forging)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.784

Source Details

Source: runDbBackup(0)
From: com.microfocus.example.web.controllers.admin.AdminDefaultController.runDbBackup
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java:117
URL: null

```
114   }
115
116   @PostMapping("/runDbBackup")
117   public String runDbBackup(@Valid @ModelAttribute("backupForm") BackupForm backupForm,
118   BindingResult bindingResult, Model model,
119   RedirectAttributes redirectAttributes,
120   Principal principal) {
```

Sink Details

Log Forging — Medium

URL: null

src/main/java/com/microfocus/example/utils/AdminUtils.java, line 67 (Log Forging)

Sink: org.slf4j.Logger.info()
Enclosing Method: startDbBackup()
File: src/main/java/com/microfocus/example/utils/AdminUtils.java:67
Taint Flags: WEB, XSS

```
64  String[] cleanupCommand = {
65  "cmd.exe", "/K", "c:\\util\\cleanup.bat"
66  };
67  log.info("Running: " + Arrays.toString(backupCommand));
68  // call backup tool API
69  log.info("Running: " + Arrays.toString(cleanupCommand));
70
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 516 (Log Forging)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.714

Source Details

Source: verifyUser(0)
From: com.microfocus.example.web.controllers.UserController.verifyUser
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:477
URL: null

```
474  }
475
476  @GetMapping("/verify")
477  public String verifyUser(@RequestParam("email") Optional<String>
usersEmail,
478  @RequestParam("code") Optional<String> verificationCode,
479  @RequestParam("status") Optional<String> statusCode,
480  RedirectAttributes redirectAttributes,
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: verifyUser()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:516
Taint Flags: NON_STRING_PARAMETERIZED_TYPE, WEB, XSS

Log Forging — Medium

URL: null

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 516 (Log Forging)

```
513  redirectAttributes.addFlashAttribute("alertClass", "alert-danger");
514  }
515  } catch (UserNotFoundException ex) {
516  log.error("Could not find user '" + email + "' to verify: " + ex.getLocalizedMessage());
517  redirectAttributes.addFlashAttribute("message", "The account being verified does not
exist. Please try registering again or contact support.");
518  redirectAttributes.addFlashAttribute("alertClass", "alert-danger");
519  }
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java, line 182 (Log Forging)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.799

Source Details

Source: ssrfExploit(1)
From: com.microfocus.example.web.controllers.admin.AdminDefaultController.ssrfExploit
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java:172
URL: null

```
169  }
170
171  @GetMapping("/log")
172  public String ssrfExploit(Model model, @Param("val") String val) {
173  int intVal = -1;
174  String strLog = "";
175  try {
```

Sink Details

Sink: org.slf4j.Logger.info()
Enclosing Method: ssrfExploit()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java:182
Taint Flags: WEB, XSS

Log Forging — Medium

URL: null

src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java, line 182 (Log Forging)

```
179  }
180  catch (NumberFormatException nfe) {
181  strLog = "Failed to parse val = " + val;
182  log.info("Failed to parse val = " + val);
183  }
184
185  model.addAttribute("val", val);
```

Log Forging — Low

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/UserUtils.java, line 133 (Log Forging)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.521

Source Details

Source: java.util.zip.ZipFile.entries()
From: com.microfocus.example.utils.UserUtils.logZipContents
File: src/main/java/com/microfocus/example/utils/UserUtils.java:131

```
128  throws IOException, SecurityException, IllegalStateException,
NoSuchElementException {
129  ZipFile zf = new ZipFile(fName);
130  @SuppressWarnings("unchecked")
131  Enumeration<ZipEntry> e = (Enumeration<ZipEntry>) zf.entries();
132  while (e.hasMoreElements()) {
133  log.info(e.nextElement().toString());
134  }
```

Sink Details

Sink: org.slf4j.Logger.info()
Enclosing Method: logZipContents()
File: src/main/java/com/microfocus/example/utils/UserUtils.java:133
Taint Flags: FILE_SYSTEM

Log Forging — Low

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/UserUtils.java, line 133 (Log Forging)

```
130  @SuppressWarnings("unchecked")
131  Enumeration<ZipEntry> e = (Enumeration<ZipEntry>) zf.entries();
132  while (e.hasMoreElements()) {
133  log.info(e.nextElement().toString());
134  }
135  }
136
```

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 398 (Log Forging)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.56

Source Details

Source: Read this.AUTHENTICATION_ERROR
From: com.microfocus.example.web.controllers.UserController.userSavePassword
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:398

```
395  redirectAttributes.addFlashAttribute("alertClass", "alert-success");
396  return "redirect:/logout";
397  } catch (InvalidPasswordException ex) {
398  log.error(AUTHENTICATION_ERROR);
399  FieldError passwordError = new FieldError("passwordForm", "password",
ex.getMessage());
400  bindingResult.addError(passwordError);
401  } catch (UserNotFoundException ex) {
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: userSavePassword()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:398
Taint Flags: ARGS, ENVIRONMENT, PROPERTY

Log Forging — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 398 (Log Forging)

```
395  redirectAttributes.addFlashAttribute("alertClass", "alert-success");
396  return "redirect:/logout";
397  } catch (InvalidPasswordException ex) {
398  log.error(AUTHENTICATION_ERROR);
399  FieldError passwordError = new FieldError("passwordForm", "password", ex.getMessage());
400  bindingResult.addError(passwordError);
401  } catch (UserNotFoundException ex) {
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 366 (Log Forging)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.56

Source Details

Source: Read this.AUTHENTICATION_ERROR
From: com.microfocus.example.web.controllers.UserController.userSaveProfile
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:366

```
363  this.setModelDefaults(model, principal, "profile");
364  return "redirect:/user/profile";
365  } catch (InvalidPasswordException ex) {
366  log.error(AUTHENTICATION_ERROR);
367  FieldError passwordError = new FieldError("userForm", "password",
ex.getMessage());
368  bindingResult.addError(passwordError);
369  } catch (UserNotFoundException ex) {
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: userSaveProfile()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:366
Taint Flags: ARGS, ENVIRONMENT, PROPERTY

Log Forging — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 366 (Log Forging)

```
363  this.setModelDefaults(model, principal, "profile");
364  return "redirect:/user/profile";
365  } catch (InvalidPasswordException ex) {
366  log.error(AUTHENTICATION_ERROR);
367  FieldError passwordError = new FieldError("userForm", "password", ex.getMessage());
368  bindingResult.addError(passwordError);
369  } catch (UserNotFoundException ex) {
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 370 (Log Forging)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.56

Source Details

Source: Read this.USER_NOT_FOUND_ERROR
From: com.microfocus.example.web.controllers.UserController.userSaveProfile
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:370

```
367  FieldError passwordError = new FieldError("userForm", "password",
ex.getMessage());
368  bindingResult.addError(passwordError);
369  } catch (UserNotFoundException ex) {
370  log.error(USER_NOT_FOUND_ERROR);
371  FieldError usernameError = new FieldError("userForm", "username",
ex.getMessage());
372  bindingResult.addError(usernameError);
373  }
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: userSaveProfile()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:370
Taint Flags: ARGS, ENVIRONMENT, PROPERTY

Log Forging — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 370 (Log Forging)

```
367  FieldError passwordError = new FieldError("userForm", "password", ex.getMessage());
368  bindingResult.addError(passwordError);
369  } catch (UserNotFoundException ex) {
370  log.error(USER_NOT_FOUND_ERROR);
371  FieldError usernameError = new FieldError("userForm", "username", ex.getMessage());
372  bindingResult.addError(usernameError);
373  }
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 463 (Log Forging)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.56

Source Details

Source: Read this.USERNAME_TAKEN_ERROR
From: com.microfocus.example.web.controllers.UserController.registerUser
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:463

```
460  this.setModelDefaults(model, null, "verify");
461  return "redirect:/user/verify?email="+u.getEmail()+"&status=new";
462  } catch (UsernameTakenException ex) {
463  log.error(USERNAME_TAKEN_ERROR);
464  FieldError usernameError = new FieldError("registerUserForm",
"username", ex.getMessage());
465  bindingResult.addError(usernameError);
466  } catch (EmailAddressTakenException ex) {
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: registerUser()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:463
Taint Flags: ARGS, ENVIRONMENT, PROPERTY

Log Forging — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 463 (Log Forging)

```
460  this.setModelDefaults(model, null, "verify");
461  return "redirect:/user/verify?email="+u.getEmail()+"&status=new";
462  } catch (UsernameTakenException ex) {
463  log.error(USERNAME_TAKEN_ERROR);
464  FieldError usernameError = new FieldError("registerUserForm", "username",
ex.getMessage());
465  bindingResult.addError(usernameError);
466  } catch (EmailAddressTakenException ex) {
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 467 (Log Forging)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.56

Source Details

Source: Read this.EMAIL_ADDRESS_TAKEN_ERROR
From: com.microfocus.example.web.controllers.UserController.registerUser
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:467

```
464  FieldError usernameError = new FieldError("registerUserForm",
"username", ex.getMessage());
465  bindingResult.addError(usernameError);
466  } catch (EmailAddressTakenException ex) {
467  log.error(EMAIL_ADDRESS_TAKEN_ERROR);
468  FieldError emailError = new FieldError("registerUserForm", "email",
ex.getMessage());
469  bindingResult.addError(emailError);
470  }
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: registerUser()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:467
Taint Flags: ARGS, ENVIRONMENT, PROPERTY

Log Forging — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 467 (Log Forging)

```
464  FieldError usernameError = new FieldError("registerUserForm", "username",
ex.getMessage());
465  bindingResult.addError(usernameError);
466  } catch (EmailAddressTakenException ex) {
467  log.error(EMAIL_ADDRESS_TAKEN_ERROR);
468  FieldError emailError = new FieldError("registerUserForm", "email", ex.getMessage());
469  bindingResult.addError(emailError);
470  }
```

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminUserController.java, line 155 (Log Forging)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.56

Source Details

Source: Read this.AUTHENTICATION_ERROR
From: com.microfocus.example.web.controllers.admin.AdminUserController.userSavePassword
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminUserController.java:155

```
152  redirectAttributes.addFlashAttribute("alertClass", "alert-success");
153  return "redirect:/admin/users/" + userId;
154  } catch (InvalidPasswordException ex) {
155  log.error(AUTHENTICATION_ERROR);
156  FieldError passwordError = new FieldError("adminPasswordForm",
"password", ex.getMessage());
157  bindingResult.addError(passwordError);
158  } catch (UserNotFoundException ex) {
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: userSavePassword()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminUserController.java:155
Taint Flags: ARGS, ENVIRONMENT, PROPERTY

Log Forging — Low

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminUserController.java, line 155 (Log Forging)

```
152  redirectAttributes.addFlashAttribute("alertClass", "alert-success");
153  return "redirect:/admin/users/" + userId;
154  } catch (InvalidPasswordException ex) {
155  log.error(AUTHENTICATION_ERROR);
156  FieldError passwordError = new FieldError("adminPasswordForm", "password",
ex.getMessage());
157  bindingResult.addError(passwordError);
158  } catch (UserNotFoundException ex) {
```

Log Forging (debug) (24 issues)

Abstract

Writing unvalidated user input to log files can allow an attacker to forge log entries or inject malicious content into the logs.

Explanation

Log forging vulnerabilities occur when:

1. Data enters an application from an untrusted source.

2. The data is written to an application or system log file.

Applications typically use log files to store a history of events or transactions for later review, statistics gathering, or debugging. Depending on the nature of the application, the task of reviewing log files may be performed manually on an as-needed basis or automated with a tool that automatically culls logs for important events or trending information.

Interpretation of the log files may be hindered or misdirected if an attacker can supply data to the application that is subsequently logged verbatim. In the most benign case, an attacker may be able to insert false entries into the log file by providing the application with input that includes appropriate characters. If the log file is processed automatically, the attacker may be able to render the file unusable by corrupting the format of the file or injecting unexpected characters. A more subtle attack might involve skewing the log file statistics. Forged or otherwise, corrupted log files can be used to cover an attacker's tracks or even to implicate another party in the commission of a malicious act [1]. In the worst case, an attacker may inject code or other commands into the log file and take advantage of a vulnerability in the log processing utility [2].

Example 1: The following web application code attempts to read an integer value from a request object. If the value fails to parse as an integer, then the input is logged with an error message indicating what happened.

```
...
        String val = request.getParameter("val");
        try {
                int value = Integer.parseInt(val);
        }
        catch (NumberFormatException nfe) {
                log.info("Failed to parse val = " + val);
        }
...
```

If a user submits the string "`twenty-one`" for `val`, the following entry is logged:

```
INFO: Failed to parse val=twenty-one
```

However, if an attacker submits the string "`twenty-one%0a%0aINFO:+User+logged+out%3dbadguy`", the following entry is logged:

```
INFO: Failed to parse val=twenty-one

INFO: User logged out=badguy
```

Clearly, attackers may use this same mechanism to insert arbitrary log entries.

Some think that in the mobile world, classic web application vulnerabilities, such as log forging, do not make sense -- why would the user attack themself? However, keep in mind that the essence of mobile platforms is applications that are downloaded from various sources and run alongside each other on the same device. The likelihood of running a piece of malware next to a banking application is high, which necessitates expanding the attack surface of mobile applications to include inter-process communication.

Example 2: The following code adapts `Example 1` to the Android platform.

```
...
        String val = this.getIntent().getExtras().getString("val");
        try {
                int value = Integer.parseInt();
        }
        catch (NumberFormatException nfe) {
                Log.e(TAG, "Failed to parse val = " + val);
        }
...
```

Recommendation

Prevent log forging attacks with indirection: create a set of legitimate log entries that correspond to different events that must be logged and only log entries from this set. To capture dynamic content, such as users logging out of the system, always use server-controlled values rather than user-supplied data. This ensures that the input provided by the user is never used directly in a log entry.

Example 1 can be rewritten to use a pre-defined log entry that corresponds to a `NumberFormatException` as follows:

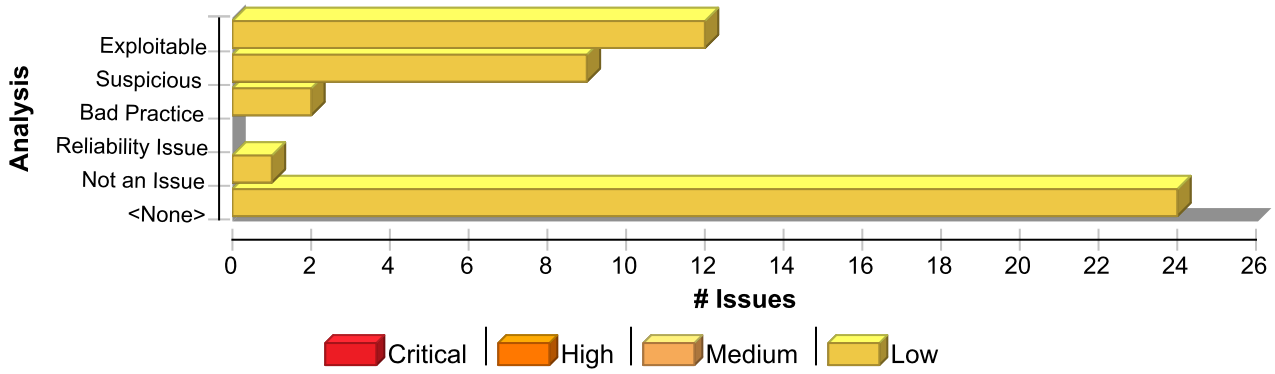
```
...
        public static final String NFE = "Failed to parse val. The input is
required to be an integer value."
...
        String val = request.getParameter("val");
        try {
                int value = Integer.parseInt(val);
        }
        catch (NumberFormatException nfe) {
                log.info(NFE);
        }
..
```

And here is an Android equivalent:

```
...
        public static final String NFE = "Failed to parse val. The input is
required to be an integer value."
...
        String val = this.getIntent().getExtras().getString("val");
        try {
                int value = Integer.parseInt();
        }
        catch (NumberFormatException nfe) {
                Log.e(TAG, NFE);
        }
...
```

In some situations this approach is impractical because the set of legitimate log entries is too large or complicated. In these situations, developers often fall back on implementing a deny list. A deny list is used to selectively reject or escape potentially dangerous characters before using the input. However, a list of unsafe characters can quickly become incomplete or outdated. A better approach is to create a list of characters that are permitted to appear in log entries and accept input composed exclusively of characters in the approved set. The most critical character in most log forging attacks is the '\n' (newline) character, which should never appear on a log entry allow list.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Log Forging (debug)	24	0	0	24
Total	**24**	**0**	**0**	**24**

Log Forging (debug) — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/UrlAuthenticationSuccessHandler.java, line 122 (Log Forging (debug))

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.705

Source Details

Source: javax.servlet.ServletRequest.getParameter()
From: com.microfocus.example.config.handlers.UrlAuthenticationSuccessHandler.handle
File: src/main/java/com/microfocus/example/config/handlers/UrlAuthenticationSuccessHandler.java:82

```
79
80  boolean isUser = false;
81  boolean isAdmin = false;
82  String targetUrl = request.getParameter("referer");
83  //if (targetUrl.endsWith("/")) targetUrl = targetUrl.substring(0, targetUrl.length());
84  String targetPath = new URL(targetUrl).getPath();
85
```

Sink Details

Log Forging (debug) — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/UrlAuthenticationSuccessHandler.java, line 122 (Log Forging (debug))

Sink: org.slf4j.Logger.debug()
Enclosing Method: handle()
File: src/main/java/com/microfocus/example/config/handlers/UrlAuthenticationSuccessHandler.java:122
Taint Flags: VALIDATED_PORTABILITY_FLAW_FILE_SEPARATOR, WEB, XSS

```
119  return;
120  }
121
122  log.debug("Redirecting to: " + targetUrl);
123  redirectStrategy.sendRedirect(request, response, targetUrl);
124  }
125
```

src/main/java/com/microfocus/example/config/handlers/AuthenticationTokenFilter.java, line 77 (Log Forging (debug))

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.761

Source Details

Source: javax.servlet.http.HttpServletRequest.getHeader()
From: com.microfocus.example.config.handlers.AuthenticationTokenFilter.parseJwt
File: src/main/java/com/microfocus/example/config/handlers/AuthenticationTokenFilte
r.java:74

```
71  }
72
73  private String parseJwt(HttpServletRequest request) {
74  String headerAuth = request.getHeader("Authorization");
75
76  if (StringUtils.hasText(headerAuth) && headerAuth.startsWith("Bearer ")) {
77  log.debug("Found jwtToken in header: " + headerAuth.substring(7,
headerAuth.length()));
```

Sink Details

Sink: org.slf4j.Logger.debug()
Enclosing Method: parseJwt()
File: src/main/java/com/microfocus/example/config/handlers/AuthenticationTokenFilter.java:77
Taint Flags: START_CHECKED_STRING, WEB, XSS

Log Forging (debug) — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/AuthenticationTokenFilter.java, line 77 (Log Forging (debug))

```
74  String headerAuth = request.getHeader("Authorization");
75
76  if (StringUtils.hasText(headerAuth) && headerAuth.startsWith("Bearer ")) {
77  log.debug("Found jwtToken in header: " + headerAuth.substring(7, headerAuth.length()));
78  return headerAuth.substring(7, headerAuth.length());
79  }
80
```

src/main/java/com/microfocus/example/config/handlers/UrlAuthenticationSuccessHandler.java, line 99 (Log Forging (debug))

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.717

Source Details

Source: javax.servlet.ServletRequest.getParameter()
From: com.microfocus.example.config.handlers.UrlAuthenticationSuccessHandler.handle
File: src/main/java/com/microfocus/example/config/handlers/UrlAuthenticationSuccess Handler.java:82

```
79
80  boolean isUser = false;
81  boolean isAdmin = false;
82  String targetUrl = request.getParameter("referer");
83  //if (targetUrl.endsWith("/")) targetUrl = targetUrl.substring(0, targetUrl.length());
84  String targetPath = new URL(targetUrl).getPath();
85
```

Sink Details

Sink: org.slf4j.Logger.debug()
Enclosing Method: handle()
File: src/main/java/com/microfocus/example/config/handlers/UrlAuthenticationSuccessHandler.java:99
Taint Flags: WEB, XSS

Log Forging (debug) — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/UrlAuthenticationSuccessHandler.java, line 99 (Log Forging (debug))

```
96  targetUrl = "/admin";
97  } else if (isUser) {
98  log.debug("targetPath=" + targetPath);
99  log.debug("targetUrl=" + targetUrl);
100  if (targetUrl.contains("?")) targetUrl = targetUrl.substring(0, targetUrl.indexOf("?"));
101  if (targetPath.endsWith("/cart")) {
102  targetUrl = targetUrl.replace("/cart", "/cart/checkout");
```

src/main/java/com/microfocus/example/config/handlers/CustomAuthenticationSuccessHandler.java, line 162 (Log Forging (debug))

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.525

Source Details

Source: javax.servlet.http.HttpServletRequest.getHeader()
From: com.microfocus.example.web.controllers.DefaultController.login
File: src/main/java/com/microfocus/example/web/controllers/DefaultController.java:99

```
96  @GetMapping("/login")
97  public String login(HttpServletRequest request, Model model, Principal principal) {
98  HttpSession session = request.getSession(false);
99  String referer = (String) request.getHeader("referer");
100  session.setAttribute("loginReferer", referer);
101  this.setModelDefaults(model, principal, "login");
102  return "login";
```

Sink Details

Sink: org.slf4j.Logger.debug()
Enclosing Method: bypassVerification()
File: src/main/java/com/microfocus/example/config/handlers/CustomAuthenticationSuccessHandler.java:162
Taint Flags: VALIDATED_PORTABILITY_FLAW_FILE_SEPARATOR, WEB, XSS

Log Forging (debug) — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/CustomAuthenticationSuccessHandler.java, line 162 (Log Forging (debug))

```
159  Authentication authentication) throws IOException {
160  String jwtToken = jwtUtils.generateAndSetSession(request, response, authentication);
161  String targetUrl = getTargetUrl(request, response, authentication);
162  log.debug("Redirecting to: " + targetUrl);
163  redirectStrategy.sendRedirect(request, response, targetUrl);
164  clearAuthenticationAttributes(request);
165  }
```

src/main/java/com/microfocus/example/config/handlers/UrlAuthenticationSuccessHandler.java, line 118 (Log Forging (debug))

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.705

Source Details

Source: javax.servlet.ServletRequest.getParameter()
From: com.microfocus.example.config.handlers.UrlAuthenticationSuccessHandler.handle
File: src/main/java/com/microfocus/example/config/handlers/UrlAuthenticationSuccessHandler.java:82

```
79
80  boolean isUser = false;
81  boolean isAdmin = false;
82  String targetUrl = request.getParameter("referer");
83  //if (targetUrl.endsWith("/")) targetUrl = targetUrl.substring(0, targetUrl.length());
84  String targetPath = new URL(targetUrl).getPath();
85
```

Sink Details

Sink: org.slf4j.Logger.debug()
Enclosing Method: handle()
File: src/main/java/com/microfocus/example/config/handlers/UrlAuthenticationSuccessHandler.java:118
Taint Flags: VALIDATED_PORTABILITY_FLAW_FILE_SEPARATOR, WEB, XSS

Log Forging (debug) — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/UrlAuthenticationSuccessHandler.java, line 118 (Log Forging (debug))

```
115  }
116
117  if (response.isCommitted()) {
118  log.debug("Response has already been committed. Unable to redirect to "+ targetUrl);
119  return;
120  }
121
```

src/main/java/com/microfocus/example/config/handlers/UrlAuthenticationSuccessHandler.java, line 98 (Log Forging (debug))

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Exploitable
AA_Prediction	Exploitable
AA_Confidence	0.846

Source Details

Source: javax.servlet.ServletRequest.getParameter()
From: com.microfocus.example.config.handlers.UrlAuthenticationSuccessHandler.handle
File: src/main/java/com/microfocus/example/config/handlers/UrlAuthenticationSuccessHandler.java:82

```
79
80   boolean isUser = false;
81   boolean isAdmin = false;
82   String targetUrl = request.getParameter("referer");
83   //if (targetUrl.endsWith("/")) targetUrl = targetUrl.substring(0, targetUrl.length());
84   String targetPath = new URL(targetUrl).getPath();
85
```

Sink Details

Sink: org.slf4j.Logger.debug()
Enclosing Method: handle()
File: src/main/java/com/microfocus/example/config/handlers/UrlAuthenticationSuccessHandler.java:98
Taint Flags: WEB, XSS

Log Forging (debug) — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/UrlAuthenticationSuccessHandler.java, line 98 (Log Forging (debug))

```
95  if (isAdmin) {
96  targetUrl = "/admin";
97  } else if (isUser) {
98  log.debug("targetPath=" + targetPath);
99  log.debug("targetUrl=" + targetUrl);
100  if (targetUrl.contains("?")) targetUrl = targetUrl.substring(0, targetUrl.indexOf("?"));
101  if (targetPath.endsWith("/cart")) {
```

Package: com.microfocus.example.repository

src/main/java/com/microfocus/example/repository/UserRepositoryImpl.java, line 107 (Log Forging (debug))

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.694

Source Details

Source: java.sql.Statement.executeQuery()
From: com.microfocus.example.repository.UserRepositoryImpl$1.execute
File: src/main/java/com/microfocus/example/repository/UserRepositoryImpl.java:77

```
74  Integer authorityCount = 0;
75  try {
76  Statement stmt = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
77  ResultSet results = stmt.executeQuery(
78  "SELECT u.*, a.name as authority " +
79  "FROM users u, authorities a INNER JOIN user_authorities ua on a.id = ua.authority_id " +
80  "WHERE u.id = ua.user_id AND u.username LIKE '" + username + "'");
```

Sink Details

Sink: org.slf4j.Logger.debug()
Enclosing Method: execute()
File: src/main/java/com/microfocus/example/repository/UserRepositoryImpl.java:107
Taint Flags: DATABASE, XSS

Log Forging (debug) — Low

Package: com.microfocus.example.repository

src/main/java/com/microfocus/example/repository/UserRepositoryImpl.java, line 107 (Log Forging (debug))

```
104  utmp.setAddress(results.getString("address"));
105  utmp.setState(results.getString("state"));
106  utmp.setZip(results.getString("zip"));
107  log.debug("Adding authority " + results.getString("authority") + " for user");
108  authorities.add(new Authority(AuthorityType.valueOf(results.getString("authority"))));
109  authorityCount++;
110  } else {
```

src/main/java/com/microfocus/example/repository/UserRepositoryImpl.java, line 111 (Log Forging (debug))

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.694

Source Details

Source: java.sql.Statement.executeQuery()
From: com.microfocus.example.repository.UserRepositoryImpl$1.execute
File: src/main/java/com/microfocus/example/repository/UserRepositoryImpl.java:77

```
74  Integer authorityCount = 0;
75  try {
76  Statement stmt = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);
77  ResultSet results = stmt.executeQuery(
78  "SELECT u.*, a.name as authority " +
79  "FROM users u, authorities a INNER JOIN user_authorities ua on a.id =
ua.authority_id " +
80  "WHERE u.id = ua.user_id AND u.username LIKE '" + username + "'");
```

Sink Details

Sink: org.slf4j.Logger.debug()
Enclosing Method: execute()
File: src/main/java/com/microfocus/example/repository/UserRepositoryImpl.java:111
Taint Flags: DATABASE, XSS

Log Forging (debug) — Low

Package: com.microfocus.example.repository

src/main/java/com/microfocus/example/repository/UserRepositoryImpl.java, line 111 (Log Forging (debug))

```
108  authorities.add(new Authority(AuthorityType.valueOf(results.getString("authority"))));
109  authorityCount++;
110  } else {
111  log.debug("Adding authority " + results.getString("authority") + " for user");
112  authorities.add(new Authority(AuthorityType.valueOf(results.getString("authority"))));
113  }
114  }
```

Package: com.microfocus.example.service

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 39 (Log Forging (debug))

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Not an Issue
AA_Prediction	Not an Issue
AA_Confidence	0.826

Source Details

Source: Read this.location
From: com.microfocus.example.config.StorageProperties.getLocation
File: src/main/java/com/microfocus/example/config/StorageProperties.java:16

```
13  private String location = System.getProperty("user.home") +
File.separatorChar + "upload-dir";
14
15  public String getLocation() {
16  return location;
17  }
18
19  public void setLocation(String location) {
```

Sink Details

Sink: org.slf4j.Logger.debug()
Enclosing Method: FileSystemStorageService()
File: src/main/java/com/microfocus/example/service/FileSystemStorageService.java:39
Taint Flags: PROPERTY

Log Forging (debug) — Low

Package: com.microfocus.example.service

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 39 (Log Forging (debug))

```
36  public FileSystemStorageService(StorageProperties properties) {
37  this.rootLocation = Paths.get(properties.getLocation());
38  if (!Files.exists(this.rootLocation)) {
39  log.debug("Creating storage service directory: " + rootLocation.toString());
40  try {
41  Files.createDirectory(rootLocation);
42  } catch (IOException e) {
```

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/DefaultController.java, line 158 (Log Forging (debug))

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.534

Source Details

Source: otpLogin(2)
From: com.microfocus.example.web.controllers.DefaultController.otpLogin
File: src/main/java/com/microfocus/example/web/controllers/DefaultController.java:1
40

```
137
138  @PostMapping("/login_mfa")
139  public String otpLogin(HttpServletRequest request, HttpServletResponse
response,
140  @RequestParam("otp") Optional<String> otp,
141  Model model, Principal principal) {
142  Authentication authentication = (Authentication) principal;
143  CustomUserDetails loggedInUser = (CustomUserDetails) ((Authentication)
principal).getPrincipal();
```

Sink Details

Sink: org.slf4j.Logger.debug()
Enclosing Method: otpLogin()
File: src/main/java/com/microfocus/example/web/controllers/DefaultController.java:158
Taint Flags: WEB, XSS

Log Forging (debug) — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/DefaultController.java, line 158 (Log Forging (debug))

```
155  int otpNum = Integer.valueOf(optStr).intValue();
156  // validate OTP "one-time-password" for user
157  if (otpNum > 0) {
158  log.debug("Verifying otp '" + otp + "' of user with id: " + userId);
159  int serverOtp = verificationService.getOtp(userId);
160  if (serverOtp > 0) {
161  if (otpNum == serverOtp) {
```

URL: null

src/main/java/com/microfocus/example/api/controllers/ApiOrderController.java, line 129 (Log Forging (debug))

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Exploitable
AA_Prediction	Exploitable
AA_Confidence	0.975

Source Details

Source: createOrder(0)
From: com.microfocus.example.api.controllers.ApiOrderController.createOrder
File: src/main/java/com/microfocus/example/api/controllers/ApiOrderController.java:128
URL: null

```
125  @PostMapping(value = {""}, produces = {"application/json"}, consumes = {"application/json"})
126  @ResponseStatus(HttpStatus.CREATED)
127  public ResponseEntity<OrderResponse> createOrder(
128  @io.swagger.v3.oas.annotations.parameters.RequestBody(description = "") @Valid @RequestBody OrderRequest newOrder) {
129  log.debug("API::Creating new order: " + newOrder.toString());
130  return new ResponseEntity<>(new OrderResponse(productService.saveOrderFromApi(null, newOrder)), HttpStatus.OK);
131  }
```

Sink Details

Sink: org.slf4j.Logger.debug()
Enclosing Method: createOrder()
File: src/main/java/com/microfocus/example/api/controllers/ApiOrderController.java:129
Taint Flags: WEB, XSS

Log Forging (debug) — Low

URL: null

src/main/java/com/microfocus/example/api/controllers/ApiOrderController.java, line 129 (Log Forging (debug))

```
126  @ResponseStatus(HttpStatus.CREATED)
127  public ResponseEntity<OrderResponse> createOrder(
128  @io.swagger.v3.oas.annotations.parameters.RequestBody(description = "") @Valid
@RequestBody OrderRequest newOrder) {
129  log.debug("API::Creating new order: " + newOrder.toString());
130  return new ResponseEntity<>(new OrderResponse(productService.saveOrderFromApi(null,
newOrder)), HttpStatus.OK);
131  }
132
```

src/main/java/com/microfocus/example/api/controllers/ApiProductController.java, line 122 (Log Forging (debug))

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Exploitable
AA_Prediction	Exploitable
AA_Confidence	0.975

Source Details

Source: createProduct(0)
From: com.microfocus.example.api.controllers.ApiProductController.createProduct
File: src/main/java/com/microfocus/example/api/controllers/ApiProductController.jav
a:121
URL: null

```
118  @PostMapping(value = {""}, produces = {"application/json"}, consumes =
{"application/json"})
119  @ResponseStatus(HttpStatus.CREATED)
120  public ResponseEntity<ProductResponse> createProduct(
121  @io.swagger.v3.oas.annotations.parameters.RequestBody(description = "")
@Valid @RequestBody ProductRequest newProduct) {
122  log.debug("API::Creating new product: " + newProduct.toString());
123  return new ResponseEntity<>(new
ProductResponse(productService.saveProductFromApi(null, newProduct)),
HttpStatus.OK);
124  }
```

Sink Details

Sink: org.slf4j.Logger.debug()
Enclosing Method: createProduct()
File: src/main/java/com/microfocus/example/api/controllers/ApiProductController.java:122
Taint Flags: WEB, XSS

Log Forging (debug) — Low

URL: null

src/main/java/com/microfocus/example/api/controllers/ApiProductController.java, line 122 (Log Forging (debug))

```
119  @ResponseStatus(HttpStatus.CREATED)
120  public ResponseEntity<ProductResponse> createProduct(
121  @io.swagger.v3.oas.annotations.parameters.RequestBody(description = "") @Valid
@RequestBody ProductRequest newProduct) {
122  log.debug("API::Creating new product: " + newProduct.toString());
123  return new ResponseEntity<>(new ProductResponse(productService.saveProductFromApi(null,
newProduct)), HttpStatus.OK);
124  }
125
```

src/main/java/com/microfocus/example/api/controllers/ApiUserController.java, line 123 (Log Forging (debug))

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Exploitable
AA_Prediction	Exploitable
AA_Confidence	0.969

Source Details

Source: createUser(0)
From: com.microfocus.example.api.controllers.ApiUserController.createUser
File: src/main/java/com/microfocus/example/api/controllers/ApiUserController.java:121
URL: null

```
118  @PostMapping(value = {""}, produces = {"application/json"}, consumes =
{"application/json"})
119  @ResponseStatus(HttpStatus.CREATED)
120  public ResponseEntity<UserResponse> createUser(
121  @io.swagger.v3.oas.annotations.parameters.RequestBody(description = "")
@Valid @RequestBody User newUser) {
122  //newUser.setId(new UUID()); // set to 0 for sequence id generation
123  log.debug("API::Creating new user: " + newUser.toString());
124  return new ResponseEntity<>(new
UserResponse(userService.saveUser(newUser)), HttpStatus.OK);
```

Sink Details

Sink: org.slf4j.Logger.debug()
Enclosing Method: createUser()
File: src/main/java/com/microfocus/example/api/controllers/ApiUserController.java:123
Taint Flags: WEB, XSS

Log Forging (debug) — Low

URL: null

src/main/java/com/microfocus/example/api/controllers/ApiUserController.java, line 123 (Log Forging (debug))

```
120  public ResponseEntity<UserResponse> createUser(
121  @io.swagger.v3.oas.annotations.parameters.RequestBody(description = "") @Valid
@RequestBody User newUser) {
122  //newUser.setId(new UUID()); // set to 0 for sequence id generation
123  log.debug("API::Creating new user: " + newUser.toString());
124  return new ResponseEntity<>(new UserResponse(userService.saveUser(newUser)),
HttpStatus.OK);
125  }
126
```

src/main/java/com/microfocus/example/api/controllers/ApiSiteController.java, line 157 (Log Forging (debug))

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Exploitable
AA_Prediction	Exploitable
AA_Confidence	0.934

Source Details

Source: emailIsTaken(0)
From: com.microfocus.example.api.controllers.ApiSiteController.emailIsTaken
File: src/main/java/com/microfocus/example/api/controllers/ApiSiteController.java:156
URL: null

```
153  })
154  @GetMapping(value = {"/email-already-exists/{email}"}, produces =
{"application/json"})
155  public ResponseEntity<Boolean> emailIsTaken(
156  @Parameter(description = "Email address to check. Cannot be empty.",
example = "user1@localhost.com", required = true) @PathVariable("email")
String email) {
157  log.debug("API::Checking for user with email: " + email);
158  Optional<User> user = userService.findUserByEmail(email);
159  if (user.isPresent()) {
```

Sink Details

Sink: org.slf4j.Logger.debug()
Enclosing Method: emailIsTaken()
File: src/main/java/com/microfocus/example/api/controllers/ApiSiteController.java:157
Taint Flags: WEB, XSS

Log Forging (debug) — Low

URL: null

src/main/java/com/microfocus/example/api/controllers/ApiSiteController.java, line 157 (Log Forging (debug))

```
154  @GetMapping(value = {"/email-already-exists/{email}"}, produces = {"application/json"})
155  public ResponseEntity<Boolean> emailIsTaken(
156  @Parameter(description = "Email address to check. Cannot be empty.", example =
"user1@localhost.com", required = true) @PathVariable("email") String email) {
157  log.debug("API::Checking for user with email: " + email);
158  Optional<User> user = userService.findUserByEmail(email);
159  if (user.isPresent()) {
160  return new ResponseEntity<Boolean>(Boolean.TRUE, HttpStatus.OK);
```

src/main/java/com/microfocus/example/api/controllers/ApiMessageController.java, line 127 (Log Forging (debug))

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.675

Source Details

Source: createMessage(0)
From: com.microfocus.example.api.controllers.ApiMessageController.createMessage
File: src/main/java/com/microfocus/example/api/controllers/ApiMessageController.jav
a:126
URL: null

```
123  @PostMapping(value = {""}, produces = {"application/json"}, consumes =
{"application/json"})
124  @ResponseStatus(HttpStatus.CREATED)
125  public ResponseEntity<MessageResponse> createMessage(
126  @io.swagger.v3.oas.annotations.parameters.RequestBody(description = "")
@Valid @RequestBody MessageRequest newMessage) {
127  log.debug("API::Creating new message: " + newMessage.toString());
128  return new ResponseEntity<>(new
MessageResponse(userService.saveMessageFromApi(null, newMessage)),
HttpStatus.OK);
129  }
```

Sink Details

Sink: org.slf4j.Logger.debug()
Enclosing Method: createMessage()
File: src/main/java/com/microfocus/example/api/controllers/ApiMessageController.java:127
Taint Flags: WEB, XSS

Log Forging (debug) — Low

URL: null

src/main/java/com/microfocus/example/api/controllers/ApiMessageController.java, line 127 (Log Forging (debug))

```
124  @ResponseStatus(HttpStatus.CREATED)
125  public ResponseEntity<MessageResponse> createMessage(
126  @io.swagger.v3.oas.annotations.parameters.RequestBody(description = "") @Valid
@RequestBody MessageRequest newMessage) {
127  log.debug("API::Creating new message: " + newMessage.toString());
128  return new ResponseEntity<>(new MessageResponse(userService.saveMessageFromApi(null,
newMessage)), HttpStatus.OK);
129  }
130
```

src/main/java/com/microfocus/example/api/controllers/ApiRoleController.java, line 117 (Log Forging (debug))

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Exploitable
AA_Prediction	Exploitable
AA_Confidence	0.967

Source Details

Source: createRole(0)
From: com.microfocus.example.api.controllers.ApiRoleController.createRole
File: src/main/java/com/microfocus/example/api/controllers/ApiRoleController.java:115
URL: null

```
112  @PostMapping(value = {""}, produces = {"application/json"}, consumes =
{"application/json"})
113  @ResponseStatus(HttpStatus.CREATED)
114  public ResponseEntity<Authority> createRole(
115  @io.swagger.v3.oas.annotations.parameters.RequestBody(description = "")
@Valid @RequestBody Authority newRole) {
116  //newRole.setId(0); // set to 0 for sequence id generation
117  log.debug("API::Creating new role: " + newRole.toString());
118  return new ResponseEntity<>(roleService.saveRole(newRole),
HttpStatus.OK);
```

Sink Details

Sink: org.slf4j.Logger.debug()
Enclosing Method: createRole()
File: src/main/java/com/microfocus/example/api/controllers/ApiRoleController.java:117
Taint Flags: WEB, XSS

Log Forging (debug) — Low

URL: null

src/main/java/com/microfocus/example/api/controllers/ApiRoleController.java, line 117 (Log Forging (debug))

```
114  public ResponseEntity<Authority> createRole(
115  @io.swagger.v3.oas.annotations.parameters.RequestBody(description = "") @Valid
@RequestBody Authority newRole) {
116  //newRole.setId(0); // set to 0 for sequence id generation
117  log.debug("API::Creating new role: " + newRole.toString());
118  return new ResponseEntity<>(roleService.saveRole(newRole), HttpStatus.OK);
119  }
120
```

src/main/java/com/microfocus/example/api/controllers/ApiSiteController.java, line 177 (Log Forging (debug))

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Exploitable
AA_Prediction	Exploitable
AA_Confidence	0.975

Source Details

Source: registerUser(0)
From: com.microfocus.example.api.controllers.ApiSiteController.registerUser
File: src/main/java/com/microfocus/example/api/controllers/ApiSiteController.java:176
URL: null

```
173  @PostMapping(value = {"/register-user"}, produces = {"application/
json"}, consumes = {"application/json"})
174  @ResponseStatus(HttpStatus.CREATED)
175  public ResponseEntity<ApiStatusResponse> registerUser(
176  @io.swagger.v3.oas.annotations.parameters.RequestBody(description = "")
@Valid @RequestBody RegisterUserRequest newUser) {
177  log.debug("API::Registering new user: " + newUser.toString());
178  RegisterUserResponse user = userService.registerUser(newUser);
179  ApiStatusResponse response = new ApiStatusResponse();
```

Sink Details

Sink: org.slf4j.Logger.debug()
Enclosing Method: registerUser()
File: src/main/java/com/microfocus/example/api/controllers/ApiSiteController.java:177
Taint Flags: WEB, XSS

Log Forging (debug) — Low

URL: null

src/main/java/com/microfocus/example/api/controllers/ApiSiteController.java, line 177 (Log Forging (debug))

```
174  @ResponseStatus(HttpStatus.CREATED)
175  public ResponseEntity<ApiStatusResponse> registerUser(
176  @io.swagger.v3.oas.annotations.parameters.RequestBody(description = "") @Valid
@RequestBody RegisterUserRequest newUser) {
177  log.debug("API::Registering new user: " + newUser.toString());
178  RegisterUserResponse user = userService.registerUser(newUser);
179  ApiStatusResponse response = new ApiStatusResponse();
180  if (user.getEmail().equals(newUser.getEmail())) response.setSuccess(true);
```

src/main/java/com/microfocus/example/api/controllers/ApiReviewController.java, line 100 (Log Forging (debug))

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.673

Source Details

Source: getReviewsByKeywords(0)
From: com.microfocus.example.api.controllers.ApiReviewController.getReviewsByKeywords
File: src/main/java/com/microfocus/example/api/controllers/ApiReviewController.java:96
URL: null

```
93  })
94  @GetMapping(value = {""}, produces = {"application/json"})
95  public ResponseEntity<List<ReviewResponse>> getReviewsByKeywords(
96  @Parameter(description = "UUID of the product to find reviews for.",
example = "eec467c8-5de9-4c7c-8541-7b31614d31a0") @RequestParam("pid")
Optional<UUID> pid,
97  @Parameter(description = "Keyword(s) search for reviews to be found.")
@RequestParam("keywords") Optional<String> keywords,
98  @Parameter(description = "Offset of the starting record. 0 indicates the
first record.") @RequestParam("offset") Optional<Integer> offset,
99  @Parameter(description = "Maximum records to return. The maximum value
allowed is 50.") @RequestParam("limit") Optional<Integer> limit) {
```

Sink Details

Log Forging (debug) — Low

URL: null

src/main/java/com/microfocus/example/api/controllers/ApiReviewController.java, line 100 (Log Forging (debug))

Sink: org.slf4j.Logger.debug()
Enclosing Method: getReviewsByKeywords()
File: src/main/java/com/microfocus/example/api/controllers/ApiReviewController.java:100
Taint Flags: WEB, XSS

```
97  @Parameter(description = "Keyword(s) search for reviews to be found.")
@RequestParam("keywords") Optional<String> keywords,
98  @Parameter(description = "Offset of the starting record. 0 indicates the first record.")
@RequestParam("offset") Optional<Integer> offset,
99  @Parameter(description = "Maximum records to return. The maximum value allowed is 50.")
@RequestParam("limit") Optional<Integer> limit) {
100  log.debug("API::Retrieving reviews by keyword(s)" + (pid.map(value -> " for product id:"
+ value).orElse("")));
101  if (limit.isPresent()) {
102  productService.setPageSize(limit.get());
103  } else {
```

src/main/java/com/microfocus/example/api/controllers/ApiSiteController.java, line 138 (Log Forging (debug))

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Exploitable
AA_Prediction	Exploitable
AA_Confidence	0.934

Source Details

Source: usernameIsTaken(0)
From: com.microfocus.example.api.controllers.ApiSiteController.usernameIsTaken
File: src/main/java/com/microfocus/example/api/controllers/ApiSiteController.java:137
URL: null

```
134  })
135  @GetMapping(value = {"/username-already-exists/{username}"}, produces =
{"application/json"})
136  public ResponseEntity<Boolean> usernameIsTaken(
137  @Parameter(description = "Username to check. Cannot be empty.", example
= "user1", required = true) @PathVariable("username") String username) {
138  log.debug("API::Checking for user with username: " + username);
139  Optional<User> user = userService.findUserByUsername(username);
140  if (user.isPresent()) {
```

Sink Details

Log Forging (debug) — Low

URL: null

src/main/java/com/microfocus/example/api/controllers/ApiSiteController.java, line 138 (Log Forging (debug))

Sink: org.slf4j.Logger.debug()
Enclosing Method: usernameIsTaken()
File: src/main/java/com/microfocus/example/api/controllers/ApiSiteController.java:138
Taint Flags: WEB, XSS

```
135  @GetMapping(value = {"/username-already-exists/{username}"}, produces = {"application/
json"})
136  public ResponseEntity<Boolean> usernameIsTaken(
137  @Parameter(description = "Username to check. Cannot be empty.", example = "user1",
required = true) @PathVariable("username") String username) {
138  log.debug("API::Checking for user with username: " + username);
139  Optional<User> user = userService.findUserByUsername(username);
140  if (user.isPresent()) {
141  return new ResponseEntity<Boolean>(Boolean.TRUE, HttpStatus.OK);
```

src/main/java/com/microfocus/example/api/controllers/ApiSiteController.java, line 195 (Log Forging (debug))

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Exploitable
AA_Prediction	Exploitable
AA_Confidence	0.977

Source Details

Source: subscribeUser(0)
From: com.microfocus.example.api.controllers.ApiSiteController.subscribeUser
File: src/main/java/com/microfocus/example/api/controllers/ApiSiteController.java:194
URL: null

```
191  @PostMapping(value = {"/subscribe-user"}, produces = {"application/
json"}, consumes = {"application/json"})
192  @ResponseStatus(HttpStatus.OK)
193  public ResponseEntity<ApiStatusResponse> subscribeUser(
194  @io.swagger.v3.oas.annotations.parameters.RequestBody(description = "")
@Valid @RequestBody SubscribeUserRequest newUser) {
195  log.debug("API::Subscribing a user to the newsletter: " +
newUser.toString());
196  SubscribeUserResponse user = userService.subscribeUser(newUser);
197  ApiStatusResponse response = new ApiStatusResponse();
```

Sink Details

Log Forging (debug) — Low

URL: null

src/main/java/com/microfocus/example/api/controllers/ApiSiteController.java, line 195 (Log Forging (debug))

Sink: org.slf4j.Logger.debug()
Enclosing Method: subscribeUser()
File: src/main/java/com/microfocus/example/api/controllers/ApiSiteController.java:195
Taint Flags: WEB, XSS

```
192  @ResponseStatus(HttpStatus.OK)
193  public ResponseEntity<ApiStatusResponse> subscribeUser(
194  @io.swagger.v3.oas.annotations.parameters.RequestBody(description = "") @Valid
@RequestBody SubscribeUserRequest newUser) {
195  log.debug("API::Subscribing a user to the newsletter: " + newUser.toString());
196  SubscribeUserResponse user = userService.subscribeUser(newUser);
197  ApiStatusResponse response = new ApiStatusResponse();
198  if ((user.getEmail().equals(newUser.getEmail()))) response.setSuccess(true);
```

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 95 (Log Forging (debug))

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Exploitable
AA_Prediction	Exploitable
AA_Confidence	0.934

Source Details

Source: firstaid(1)
From: com.microfocus.example.web.controllers.ProductController.firstaid
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:94
URL: null

```
91  }
92
93  @GetMapping("/firstaid")
94  public String firstaid(Model model, @Param("keywords") String keywords,
@Param("limit") Integer limit, Principal principal) {
95  log.debug("Searching for products using keywords: " + ((keywords == null
|| keywords.isEmpty()) ? "none" : keywords));
96  productService.setPageSize((limit == null ? defaultPageSize : limit));
97  List<Product> products = productService.getAllActiveProducts(0, keywords);
```

Sink Details

Log Forging (debug) — Low

URL: null

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 95 (Log Forging (debug))

Sink: org.slf4j.Logger.debug()
Enclosing Method: firstaid()
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:95
Taint Flags: WEB, XSS

```
92
93  @GetMapping("/firstaid")
94  public String firstaid(Model model, @Param("keywords") String keywords, @Param("limit") Integer limit, Principal principal) {
95  log.debug("Searching for products using keywords: " + ((keywords == null || keywords.isEmpty()) ? "none" : keywords));
96  productService.setPageSize((limit == null ? defaultPageSize : limit));
97  List<Product> products = productService.getAllActiveProducts(0, keywords);
98  model.addAttribute("keywords", keywords);
```

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 108 (Log Forging (debug))

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Exploitable
AA_Prediction	Exploitable
AA_Confidence	0.934

Source Details

Source: index(1)
From: com.microfocus.example.web.controllers.ProductController.index
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:107
URL: null

```
104  }
105
106  @GetMapping(value = {"", "/"})
107  public String index(Model model, @Param("keywords") String keywords, @Param("limit") Integer limit, Principal principal) {
108  log.debug("Searching for products using keywords: " + ((keywords == null || keywords.isEmpty()) ? "none" : keywords));
109  productService.setPageSize((limit == null ? defaultPageSize : limit));
110  List<Product> products = productService.getAllActiveProducts(0, keywords);
```

Sink Details

Log Forging (debug) — Low

URL: null

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 108 (Log Forging (debug))

Sink: org.slf4j.Logger.debug()
Enclosing Method: index()
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:108
Taint Flags: WEB, XSS

```
105
106  @GetMapping(value = {"", "/"})
107  public String index(Model model, @Param("keywords") String keywords, @Param("limit")
Integer limit, Principal principal) {
108  log.debug("Searching for products using keywords: " + ((keywords == null ||
keywords.isEmpty()) ? "none" : keywords));
109  productService.setPageSize((limit == null ? defaultPageSize : limit));
110  List<Product> products = productService.getAllActiveProducts(0, keywords);
111  model.addAttribute("keywords", keywords);
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 507 (Log Forging (debug))

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.768

Source Details

Source: verifyUser(0)
From: com.microfocus.example.web.controllers.UserController.verifyUser
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:477
URL: null

```
474  }
475
476  @GetMapping("/verify")
477  public String verifyUser(@RequestParam("email") Optional<String>
usersEmail,
478  @RequestParam("code") Optional<String> verificationCode,
479  @RequestParam("status") Optional<String> statusCode,
480  RedirectAttributes redirectAttributes,
```

Sink Details

Log Forging (debug) — Low

URL: null

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 507 (Log Forging (debug))

Sink: org.slf4j.Logger.debug()
Enclosing Method: verifyUser()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:507
Taint Flags: NON_STRING_PARAMETERIZED_TYPE, WEB, XSS

```
504  try {
505  User u = userService.verifyUserRegistration(email, code);
506  if (u != null) {
507  log.debug("Successfully verified user '" + email + "'");
508  redirectAttributes.addFlashAttribute("message", "Your account has been successfully
verified. Please login.");
509  redirectAttributes.addFlashAttribute("alertClass", "alert-success");
510  } else {
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java, line 124 (Log Forging (debug))

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Exploitable
AA_Prediction	Exploitable
AA_Confidence	0.982

Source Details

Source: runDbBackup(0)
From: com.microfocus.example.web.controllers.admin.AdminDefaultController.runDbBackup
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java:117
URL: null

```
114  }
115
116  @PostMapping("/runDbBackup")
117  public String runDbBackup(@Valid @ModelAttribute("backupForm")
BackupForm backupForm,
118  BindingResult bindingResult, Model model,
119  RedirectAttributes redirectAttributes,
120  Principal principal) {
```

Sink Details

Log Forging (debug) — Low

URL: null

src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java, line 124 (Log Forging (debug))

Sink: org.slf4j.Logger.debug()
Enclosing Method: runDbBackup()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java:124
Taint Flags: WEB, XSS

```
121  if (bindingResult.hasErrors()) {
122  return "admin/backup";
123  } else {
124  log.debug("Backup profile: " + backupForm.getProfile());
125  int backUpId = 0;
126  try {
127  backUpId = AdminUtils.startDbBackup(backupForm.getProfile());
```

Mass Assignment: Request Parameters Bound into Persisted Objects (2 issues)

Abstract

Allowing database persistent entities to be auto-populated by request parameters will let an attacker create unintended records in association entities or update unintended fields in the entity object.

Explanation

Persistent objects are bound to the underlying database and updated automatically by the persistence framework, such as Hibernate or JPA. Allowing these objects to be dynamically bound to the request by Spring MVC will let an attacker inject unexpected values into the database by providing additional request parameters. **Example 1:** The `Order`, `Customer`, and `Profile` are Hibernate persisted classes.

```
public class Order {
        String ordered;
        List lineItems;
        Customer cust;
...
}
public class Customer {
        String customerId;
        ...
    Profile p;
...
}
public class Profile {
        String profileId;
        String username;
        String password;
        ...
}
```

`OrderController` is the Spring controller class handling the request:

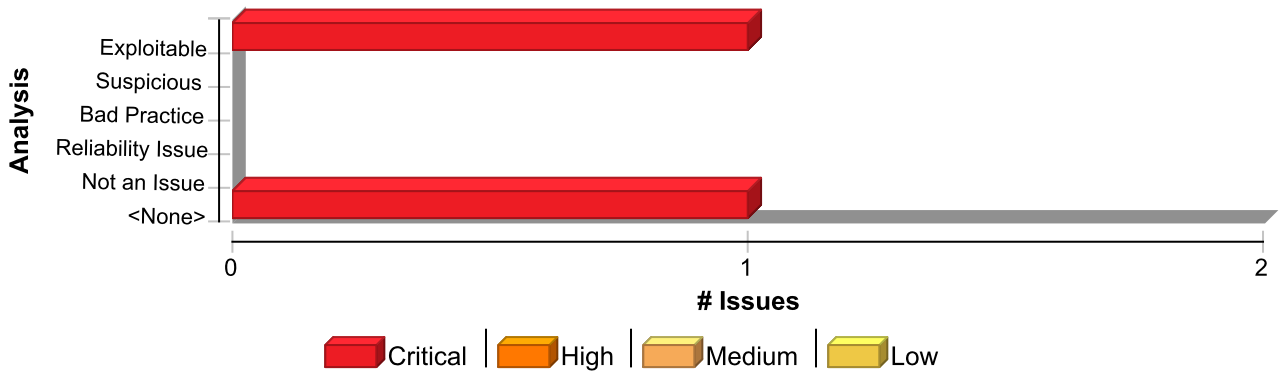
```
@Controller
public class OrderController {
...
        @RequestMapping("/updateOrder")
        public String updateOrder(Order order) {
                ...
                session.save(order);
        }
}
```

Because command classes are automatically bound to the request, an attacker may use this vulnerability to update another user's password by adding the following request parameters to the request: "http://www.yourcorp.com/webApp/updateOrder?order.customer.profile.profileId=1234&order.customer.profile.password=urpowned"

Recommendation

Do not use persistent entity objects as your request bound objects. Manually copy the attributes which you are interested in persisting from your request bound objects to your persistent entity objects. An alternative would be to explicitly define which attributes on the request bound object are settable via request parameters.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Mass Assignment: Request Parameters Bound into Persisted Objects	2	0	0	2
Total	**2**	**0**	**0**	**2**

Mass Assignment: Request Parameters Bound into Persisted Objects — Critical

Package: com.microfocus.example.entity

src/main/java/com/microfocus/example/entity/User.java, line 45 (Mass Assignment: Request Parameters Bound into Persisted Objects)

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Audit Details

JiraBugLink	
Analysis	Exploitable
AA_Prediction	Not Predicted

Audit Comments

admin: Wed Jun 28 2023 10:08:38 GMT-0000 (UTC)
test

admin: Wed Jun 28 2023 10:09:52 GMT-0000 (UTC)
test

Sink Details

Sink: Class: User
File: src/main/java/com/microfocus/example/entity/User.java:45

```
42 @Entity
43 @Table(name = "users")
44 @JsonIdentityInfo(generator = ObjectIdGenerators.PropertyGenerator.class, property = "id")
45 public class User implements Serializable {
46
47  private static final long serialVersionUID = 1L;
48
```

Mass Assignment: Request Parameters Bound into Persisted Objects — Critical

Package: com.microfocus.example.entity

src/main/java/com/microfocus/example/entity/Authority.java, line 36 (Mass Assignment: Request Parameters Bound into Persisted Objects)

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction	Not Predicted

Sink Details

Sink: Class: Authority
File: src/main/java/com/microfocus/example/entity/Authority.java:36

```
33 @Entity
34 @Table(name = "authorities")
35 @JsonIdentityInfo(generator = ObjectIdGenerators.PropertyGenerator.class, property = "id")
36 public class Authority {
37
38  private static final long serialVersionUID = 1L;
39
```

Missing XML Validation (2 issues)

Abstract

Failure to enable validation when parsing XML gives an attacker the opportunity to supply malicious input.

Explanation

Most successful attacks begin with a violation of the programmer's assumptions. By accepting an XML document without validating it against a DTD or XML schema, the programmer leaves a door open for attackers to provide unexpected, unreasonable, or malicious input. It is not possible for an XML parser to validate all aspects of a document's content; a parser cannot understand the complete semantics of the data. However, a parser can do a complete and thorough job of checking the document's structure and therefore guarantee to the code that processes the document that the content is well-formed.

Recommendation

Always enable validation when you create an XML parser or parser factory. If enabling validation causes problems because the rules for defining a well-formed document are Byzantine or altogether unknown, chances are good that there are security errors nearby.

The following examples demonstrate how to enable validation for the Xerces parsers (both DOM and SAX):

```
org.apache.xerces.framework.XMLParser: parser.setValidation(true);
org.apache.xerces.framework.XMLParser: parser.setValidationSchema(true);
```

The following examples demonstrate how to enable validation for the SAX and DOM parser factories in the javax library.

javax SAX parser factory:

```
javax.xml.parsers.SAXParserFactory: factory.setValidating(true);
javax.xml.parsers.SAXParserFactory: factory.setFeature("http://xml.org/sax/
features/validation", true);
```

javax DOM parser factory:

```
javax.xml.parsers.DocumentBuilderFactory: factory.setValidating(true);
```

The following examples demonstrate how to enable validation for individual parsers and XMLReaders in the javax library.

Note: The Fortify Software Security Research group does not recommend enabling validation by this method. Instead, you should enable validation at the parser factory.

javax SAX parser and reader:

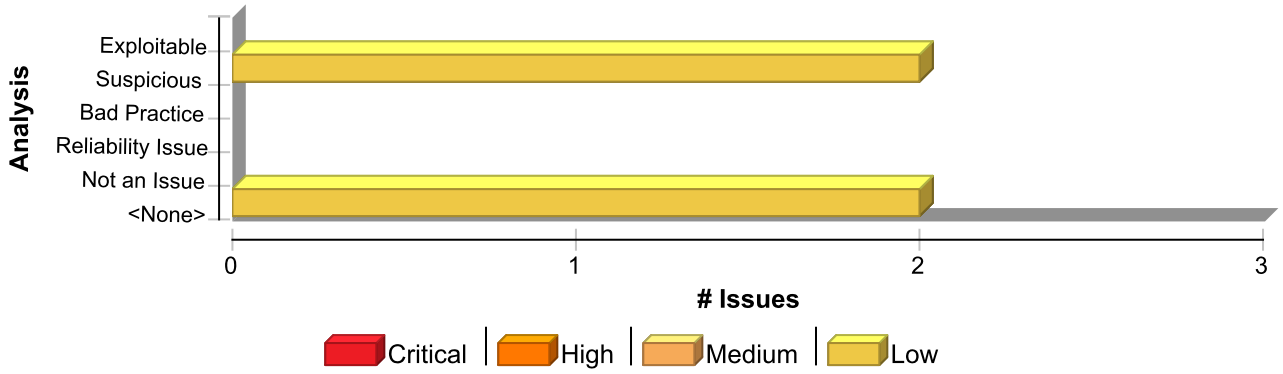
```
javax.xml.parsers.SAXParser: parser.setProperty("http://xml.org/sax/features/
validation", new Boolean(true));
org.xml.sax.XMLReader: reader.setFeature("http://xml.org/sax/features/
validation", true);
```

The following examples demonstrate how to set the XML return type for Apache Axis.

Axis client Call:

```
call.addParameter("testParam", org.apache.axis.Constants.XSD_STRING,
javax.xml.rpc.ParameterMode.IN);
call.setReturnType(org.apache.axis.Constants.XSD_STRING);
String ret = (String) call.invoke( new Object[] { "Hello!" } );
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Missing XML Validation	2	0	0	2
Total	**2**	**0**	**0**	**2**

Missing XML Validation — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 662 (Missing XML Validation)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Control Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.69

Sink Details

Sink: db.parse(...)
Enclosing Method: handleXMLUpdate()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:662

```
659  try {
660  dbf.setFeature(XMLConstants.FEATURE_SECURE_PROCESSING, false);
661  DocumentBuilder db = dbf.newDocumentBuilder();
662  Document doc = db.parse(new InputSource(new StringReader(newXMLContent)));
663  Path temp = Files.createTempFile("iwa", ".xml");
664  try (FileOutputStream outStream = new FileOutputStream(temp.toString())) {
665  writeXml(doc, outStream);
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 600 (Missing XML Validation)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Control Flow)

Missing XML Validation — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 600 (Missing XML Validation)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.69

Sink Details

Sink: db.parse(...)
Enclosing Method: getXMLFileContent()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:600

```
597  try {
598  dbf.setFeature(XMLConstants.FEATURE_SECURE_PROCESSING, false);
599  DocumentBuilder db = dbf.newDocumentBuilder();
600  Document doc = db.parse(fpath.toFile());
601  try (ByteArrayOutputStream bytesOutStream = new ByteArrayOutputStream()) {
602  writeXml(doc, bytesOutStream);
603  xmlContent = bytesOutStream.toString();
```

Null Dereference (1 issue)

Abstract

The program can potentially dereference a null-pointer, thereby causing a null-pointer exception.

Explanation

Null-pointer exceptions usually occur when one or more of the programmer's assumptions is violated. A dereference-after-store error occurs when a program explicitly sets an object to `null` and dereferences it later. This error is often the result of a programmer initializing a variable to `null` when it is declared.

Most null-pointer issues result in general software reliability problems, but if attackers can intentionally trigger a null-pointer dereference, they can use the resulting exception to bypass security logic or to cause the application to reveal debugging information that will be valuable in planning subsequent attacks.

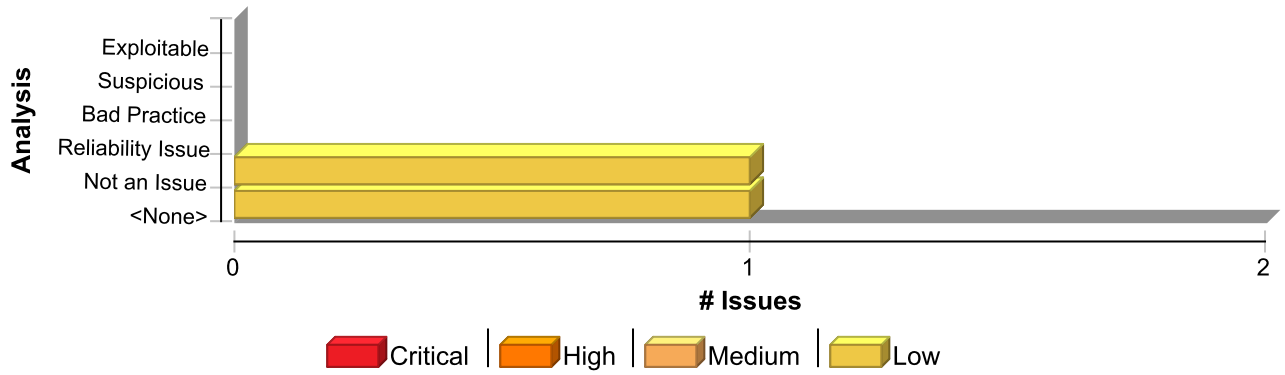
Example: In the following code, the programmer explicitly sets the variable `foo` to `null`. Later, the programmer dereferences `foo` before checking the object for a `null` value.

```
Foo foo = null;
...
foo.setBar(val);
...
}
```

Recommendation

Implement careful checks before dereferencing objects that might be `null`. When possible, abstract `null` checks into wrappers around code that manipulates resources to ensure that they are applied in all cases and to minimize the places where mistakes can occur.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Null Dereference	1	0	0	1
Total	**1**	**0**	**0**	**1**

Null Dereference — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/CustomAuthenticationSuccessHandler.java, line 122 (Null Dereference)

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Audit Details

Analysis	Not an Issue
AA_Prediction	Not an Issue
AA_Confidence	0.833

Sink Details

Sink: Dereferenced : targetPath
Enclosing Method: getTargetUrl()
File: src/main/java/com/microfocus/example/config/handlers/CustomAuthenticationSuccessHandler.java:122

```
119  log.error(ex.getLocalizedMessage());
120  }
121  if (targetUrl.contains("?")) targetUrl = targetUrl.substring(0, targetUrl.indexOf("?"));
122  if (targetPath.endsWith("/cart")) {
123  targetUrl = targetUrl.replace("/cart", "/cart/checkout");
124  } else if (targetPath.endsWith("/login")) {
125  targetUrl = targetUrl.replace("/login", "/user");
```

Often Misused: Boolean.getBoolean() (1 issue)

Abstract

The method `Boolean.getBoolean()` is often confused with `Boolean.valueOf()` or `Boolean.parseBoolean()` method calls.

Explanation

In most cases, a call to `Boolean.getBoolean()` is often misused as it is assumed to return the boolean value represented by the specified string argument. However, as stated in the Javadoc `Boolean.getBoolean(String)` method "Returns true if and only if the system property named by the argument exists and is equal to the string 'true'."

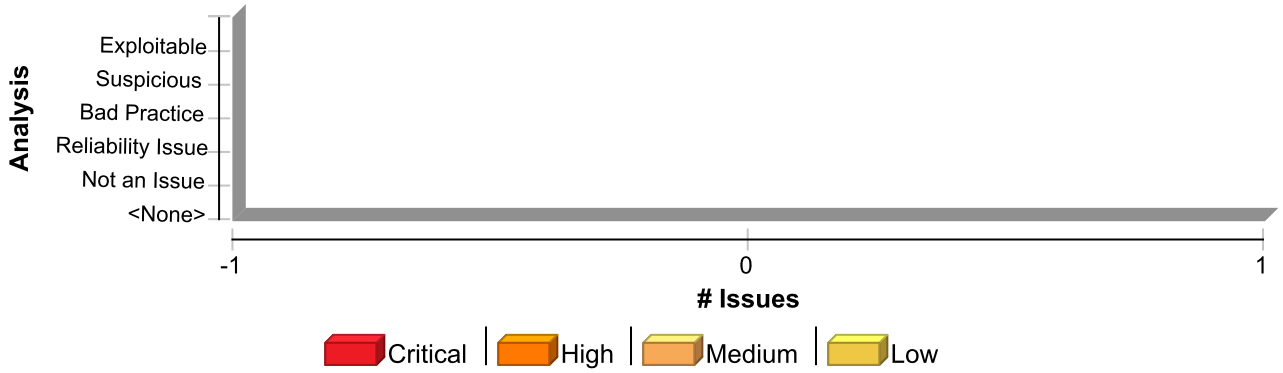
Most often what the developer intended to use was a call to `Boolean.valueOf(String)` or `Boolean.parseBoolean(String)` method. **Example 1:** The following code will not behave as expected. It will print "FALSE" as `Boolean.getBoolean(String)` does not translate a String primitive. It only translates system property.

```
...
String isValid = "true";
if ( Boolean.getBoolean(isValid) ) {
    System.out.println("TRUE");
}
else {
    System.out.println("FALSE");
}
...
```

Recommendation

Please ensure that you intend to call the method `Boolean.getBoolean(String)` and the specified string argument is a system property. Else the method call you are most likely looking for is `Boolean.valueOf(String)` or `Boolean.parseBoolean(String)`.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Often Misused: Boolean.getBoolean()	1	0	0	1
Total	**1**	**0**	**0**	**1**

Often Misused: Boolean.getBoolean() — Medium

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/AdminUtils.java, line 91 (Often Misused: Boolean.getBoolean())

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Audit Details

AA_Prediction	Not Predicted

Sink Details

Sink: getBoolean()
Enclosing Method: getDbStatus()
File: src/main/java/com/microfocus/example/utils/AdminUtils.java:91

```
88   }
89 */
90
91   if(Boolean.getBoolean(isLocked(backupId))){
92   return"LOCKED";
93   }
94   return isReady(backupId);
```

Often Misused: File Upload (2 issues)

Abstract

Permitting users to upload files can allow attackers to inject dangerous content or malicious code to run on the server.

Explanation

Regardless of the language a program is written in, the most devastating attacks often involve remote code execution, whereby an attacker succeeds in executing malicious code in the program's context. If attackers are allowed to upload files to a directory that is accessible from the Web and cause these files to be passed to a code interpreter (e.g. JSP/ASPX/PHP), then they can cause malicious code contained in these files to execute on the server.

Example: The following Spring MVC controller class has a parameter than can be used to handle uploaded files.

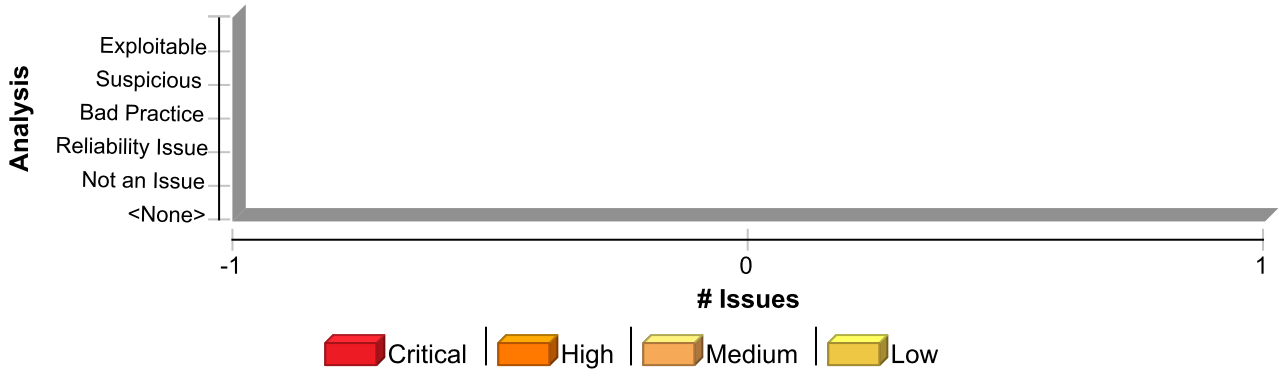
```
@Controller
public class MyFormController {
    ...
    @RequestMapping("/test")
    public String uploadFile (org.springframework.web.multipart.MultipartFile
file) {
       ...
    }    ...
}
```

Even if a program stores uploaded files under a directory that isn't accessible from the Web, attackers might still be able to leverage the ability to introduce malicious content into the server environment to mount other attacks. If the program is susceptible to path manipulation, command injection, or dangerous file inclusion vulnerabilities, then an attacker might upload a file with malicious content and cause the program to read or execute it by exploiting another vulnerability.

Recommendation

Do not accept attachments if they can be avoided. If a program must accept attachments, then restrict the ability of an attacker to supply malicious content by only accepting the specific types of content the program expects. Most attacks that rely on uploaded content require that attackers be able to supply content of their choosing. Placing restrictions on the content the program will accept will greatly limit the range of possible attacks. Check file names, extensions, and file content to make sure they are all expected and acceptable for use by the application. Make it difficult for the attacker to determine the name and location of uploaded files. Such solutions are often program-specific and vary from storing uploaded files in a directory with a name generated from a strong random value when the program is initialized to assigning each uploaded file a random name and tracking them with entries in a database.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Often Misused: File Upload	2	0	0	2
Total	**2**	**0**	**0**	**2**

Often Misused: File Upload — Medium

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 554 (Often Misused: File Upload)

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: Function: handleFileUpload
Enclosing Method: handleFileUpload()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:554

```
551  }
552
553  @PostMapping("/files/upload")
554  public String handleFileUpload(@RequestParam("file") MultipartFile file,
555  RedirectAttributes redirectAttributes) {
556
557  storageService.store(file);
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 626 (Often Misused: File Upload)

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Often Misused: File Upload — Medium

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 626 (Often Misused: File Upload)

Sink Details

Sink: Function: handleXMLFileUpload
Enclosing Method: handleXMLFileUpload()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:626

```
623  }
624
625  @PostMapping("/files/upload-xml")
626  public String handleXMLFileUpload(@RequestParam("file") MultipartFile file,
627  RedirectAttributes redirectAttributes) {
628
629  storageService.store(file);
```

Open Redirect (1 issue)

Abstract

Allowing unvalidated input to control the URL used in a redirect can aid phishing attacks.

Explanation

Redirects allow web applications to direct users to different pages within the same application or to external sites. Applications utilize redirects to aid in site navigation and, in some cases, to track how users exit the site. Open redirect vulnerabilities occur when a web application redirects clients to any arbitrary URL that can be controlled by an attacker.

Attackers might utilize open redirects to trick users into visiting a URL to a trusted site, but then redirecting them to a malicious site. By encoding the URL, an attacker can make it difficult for end-users to notice the malicious destination of the redirect, even when it is passed as a URL parameter to the trusted site. Open redirects are often abused as part of phishing scams to harvest sensitive end-user data.

Example 1: The following JSP code instructs the user's browser to open a URL parsed from the `dest` request parameter when a user clicks the link.

```
<%
    ...
    String strDest = request.getParameter("dest");
    pageContext.forward(strDest);
    ...
%>
```

If a victim received an email instructing them to follow a link to "http://trusted.example.com/ecommerce/redirect.asp?dest=www.wilyhacker.com", the user would likely click on the link believing they would be transferred to the trusted site. However, when the victim clicks the link, the code in `Example 1` will redirect the browser to "http://www.wilyhacker.com".

Many users have been educated to always inspect URLs they receive in emails to make sure the link specifies a trusted site they know. However, if the attacker Hex encoded the destination url as follows: "http://trusted.example.com/ecommerce/redirect.asp?dest=%77%69%6C%79%68%61%63%6B%65%72%2E%63%6F%6D"

then even a savvy end-user may be fooled into following the link.

Recommendation

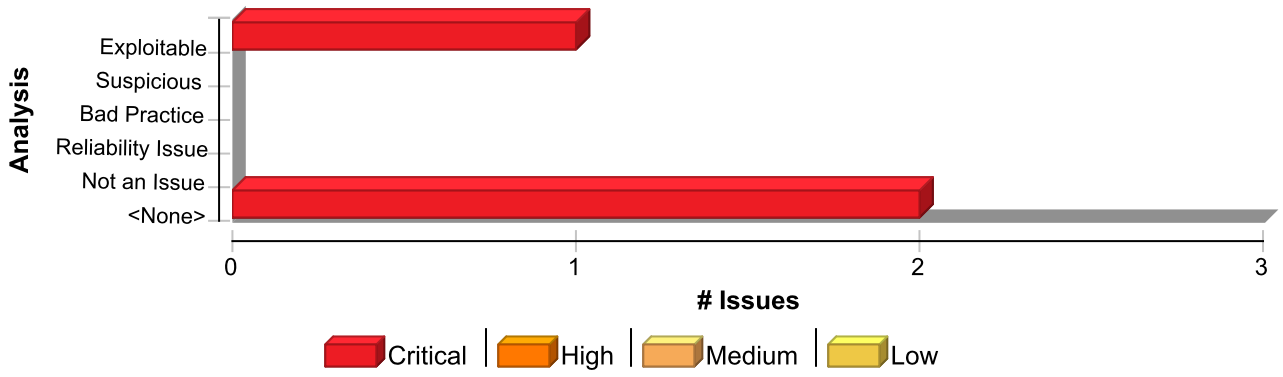
Unvalidated user input should not be allowed to control the destination URL in a redirect. Instead, use a level of indirection: create a list of legitimate URLs that users are allowed to specify and only allow users to select from the list. With this approach, input provided by users is never used directly to specify a URL for redirects.

Example 2: The following code references an array populated with valid URLs. The link the user clicks passes in the array index that corresponds to the desired URL.

```
<%
    ...
    try {
        int strDest = Integer.parseInt(request.getParameter("dest"));
        if((strDest >= 0) && (strDest <= strURLArray.length -1 ))
        {
            strFinalURL = strURLArray[strDest];
            pageContext.forward(strFinalURL);
        }
    }
    catch (NumberFormatException nfe) {
        // Handle exception
        ...
    }
    ...
%>
```

In some situations this approach is impractical because the set of legitimate URLs is too large or too hard to keep track of. In such cases, use a similar approach to restrict the domains that users can be redirected to, which can at least prevent attackers from sending users to malicious external sites.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Open Redirect	1	0	0	1
Total	**1**	**0**	**0**	**1**

Open Redirect — Critical

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/DefaultController.java, line 178 (Open Redirect)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

JiraBugLink	
Analysis	Exploitable
AA_Training	Include
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.678

Source Details

Source: javax.servlet.http.HttpServletRequest.getHeader()
From: com.microfocus.example.web.controllers.DefaultController.login
File: src/main/java/com/microfocus/example/web/controllers/DefaultController.java:99

```
96  @GetMapping("/login")
97  public String login(HttpServletRequest request, Model model, Principal principal) {
98  HttpSession session = request.getSession(false);
99  String referer = (String) request.getHeader("referer");
100  session.setAttribute("loginReferer", referer);
101  this.setModelDefaults(model, principal, "login");
102  return "login";
```

Sink Details

Sink: Return
Enclosing Method: otpLogin()
File: src/main/java/com/microfocus/example/web/controllers/DefaultController.java:178
Taint Flags: VALIDATED_PORTABILITY_FLAW_FILE_SEPARATOR, WEB, XSS

```
175  }
176  String jwtToken = jwtUtils.generateAndSetSession(request, response, authentication);
177  String targetUrl = CustomAuthenticationSuccessHandler.getTargetUrl(request, response, authentication);
178  return "redirect:"+targetUrl;
179  }
180
181  @GetMapping("/services")
```

Password Management: Empty Password in Configuration File (1 issue)

Abstract

Using an empty string as a password is insecure.

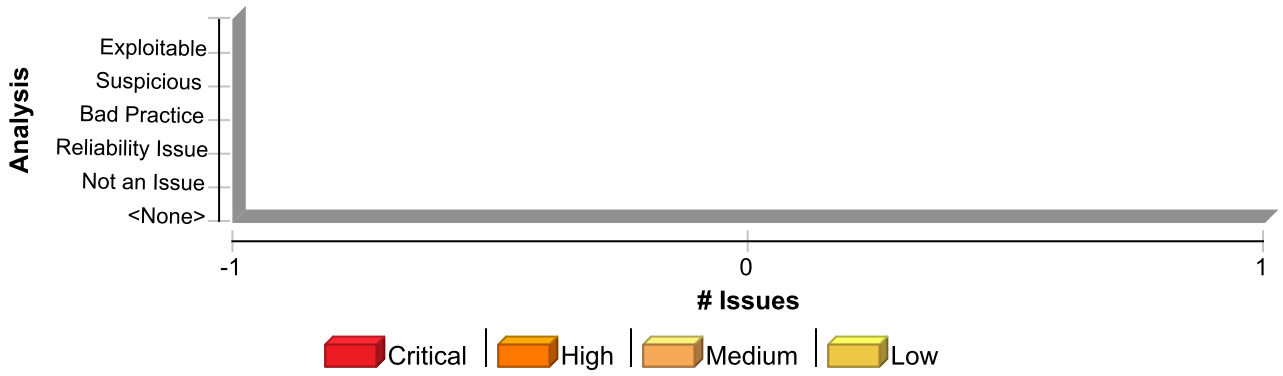
Explanation

It is never appropriate to use an empty string as a password. It is too easy to guess.

Recommendation

Require that sufficiently hard-to-guess passwords protect all accounts and system resources. Consult the references to help establish appropriate password guidelines.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Password Management: Empty Password in Configuration File	1	0	0	1
Total	**1**	**0**	**0**	**1**

Password Management: Empty Password in Configuration File — High

Package: src.main.resources

src/main/resources/application.yml, line 59 (Password Management: Empty Password in Configuration File)

Issue Details

Kingdom: Environment
Scan Engine: SCA (Scripted)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: spring.mail.password
File: src/main/resources/application.yml:59

Password Management: Empty Password in Configuration File — High

Package: src.main.resources

src/main/resources/application.yml, line 59 (Password Management: Empty Password in Configuration File)

```
56  default-encoding: UTF-8
57  host: smtp.sendgrid.net
58  username: apikey
59  password: # Enter SendGrid API Password here
60  port: 587
61  test-connection: false
62  debug: true
```

Password Management: Hardcoded Password (9 issues)

Abstract

Hardcoded passwords can compromise system security in a way that is difficult to remedy.

Explanation

Never hardcode passwords. Not only does it expose the password to all of the project's developers, it also makes fixing the problem extremely difficult. After the code is in production, a program patch is probably the only way to change the password. If the account the password protects is compromised, the system owners must choose between security and availability.

Example: The following YAML uses a hardcoded password:

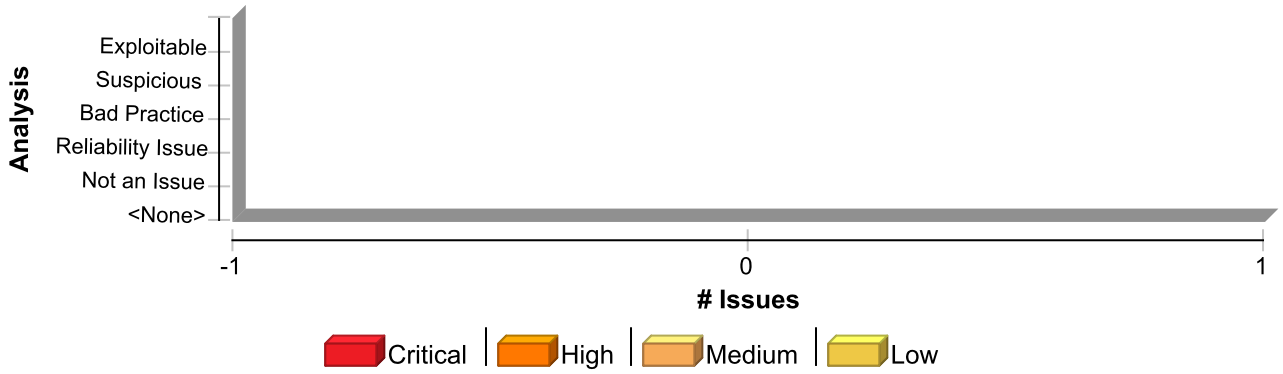
```
...
credential_settings:
  username: scott
  password: tiger
...
```

This configuration may be valid, but anyone who has access to the configuration will have access to the password. After the program is released, changing the default user account "scott" with a password of "tiger" is difficult. Anyone with access to this information can use it to break into the system.

Recommendation

Never hardcode password. Passwords should generally be obfuscated and managed in an external source. Storing passwords in plain text anywhere on the system allows anyone with sufficient permissions to read and potentially misuse the password.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Password Management: Hardcoded Password	9	0	0	9
Total	**9**	**0**	**0**	**9**

Password Management: Hardcoded Password — Critical

Package: src.main.resources

src/main/resources/data.sql, line 14 (Password Management: Hardcoded Password)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Configuration)

Audit Details

AA_Prediction Not Predicted

Sink Details

File: src/main/resources/data.sql:14

```
11 VALUES ('32e7db01-86bc-4687-9ecb-d79b265ac14f', 'user1',
'$2a$10$YFhTnHpCL.Z0Ev0j1CbEUub7sIWmN7Qd5RmnU8g5ekuoapV7Zdx32',
12  'Sam', 'Shopper', 'user1@localhost.com', '+44808123456', '1 Somewhere Street', 'London',
'Greater London', 'SW1', 'United Kingdom', CURDATE(), 1);
13 INSERT INTO users (id, username, password, first_name, last_name, email, phone, address,
city, state, zip, country, date_created, enabled, mfa)
14 VALUES ('db4cfab1-ff1d-4bca-a662-394771841383', 'user2',
'$2a$10$YFhTnHpCL.Z0Ev0j1CbEUub7sIWmN7Qd5RmnU8g5ekuoapV7Zdx32',
15  'Sarah', 'Shopper', 'user2@localhost.com', '+44808123456', '1 Somewhere Street', 'London',
'Greater London', 'SW1', 'United Kingdom', CURDATE(), 1, 1);
16 INSERT INTO users (id, username, password, first_name, last_name, email, phone, address,
city, state, zip, country, date_created, enabled)
17 VALUES ('92a82f45-7a03-42f3-80f8-ce4e9892409d', 'api',
'$2a$10$YFhTnHpCL.Z0Ev0j1CbEUub7sIWmN7Qd5RmnU8g5ekuoapV7Zdx32',
```

src/main/resources/data.sql, line 8 (Password Management: Hardcoded Password)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Configuration)

Audit Details

AA_Prediction Not Predicted

Sink Details

File: src/main/resources/data.sql:8

Password Management: Hardcoded Password — Critical

Package: src.main.resources

src/main/resources/data.sql, line 8 (Password Management: Hardcoded Password)

```
5 INSERT INTO authorities (name, id)
6 values ('ROLE_API', 'dfc1d81b-4a7e-4248-80f7-8445ee5cb68e');
7 INSERT INTO users (id, username, password, first_name, last_name, email, phone, address,
city, state, zip, country, date_created, enabled)
8 VALUES ('e18c8bcc-935d-444d-a194-3a32a3b35a49', 'admin',
'$2a$10$YFhTnHpCL.Z0Ev0j1CbEUub7sIWmN7Qd5RmnU8g5ekuoapV7Zdx32',
9  'Admin', 'User', 'admin@localhost.com', '+44808123456', '', '', '', '', 'United Kingdom',
CURDATE(), 1);
10 INSERT INTO users (id, username, password, first_name, last_name, email, phone, address,
city, state, zip, country, date_created, enabled)
11 VALUES ('32e7db01-86bc-4687-9ecb-d79b265ac14f', 'user1',
'$2a$10$YFhTnHpCL.Z0Ev0j1CbEUub7sIWmN7Qd5RmnU8g5ekuoapV7Zdx32',
```

src/main/resources/data.sql, line 20 (Password Management: Hardcoded Password)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Configuration)

Audit Details

AA_Prediction Not Predicted

Sink Details

File: src/main/resources/data.sql:20

```
17 VALUES ('92a82f45-7a03-42f3-80f8-ce4e9892409d', 'api',
'$2a$10$YFhTnHpCL.Z0Ev0j1CbEUub7sIWmN7Qd5RmnU8g5ekuoapV7Zdx32',
18  'Api', 'User', 'api@localhost.com', '+44808123456', '1 Somewhere Street', 'London',
'Greater London', 'SW1', 'United Kingdom', CURDATE(), 1);
19 INSERT INTO users (id, username, password, first_name, last_name, email, phone, address,
city, state, zip, country, date_created, enabled, verify_code)
20 VALUES ('a730c051-b5c2-454c-b669-679f06d99731', 'user3',
'$2a$10$YFhTnHpCL.Z0Ev0j1CbEUub7sIWmN7Qd5RmnU8g5ekuoapV7Zdx32',
21  'Steve', 'Shopper', 'user3@localhost.com', '+44808123456', '1 Somewhere Street', 'London',
'Greater London', 'SW1', 'United Kingdom', CURDATE(), 0, 'AwUjqPvDLVxjzTEChhQXMDMJxBlWvZoG');
22 INSERT INTO user_authorities (authority_id, user_id)
23 VALUES ('05970e74-c82b-4e21-b100-f8184d6e3454', 'e18c8bcc-935d-444d-a194-3a32a3b35a49');
```

src/main/resources/data.sql, line 11 (Password Management: Hardcoded Password)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Configuration)

Audit Details

AA_Prediction Not Predicted

Sink Details

Password Management: Hardcoded Password — Critical

Package: src.main.resources

src/main/resources/data.sql, line 11 (Password Management: Hardcoded Password)

File: src/main/resources/data.sql:11

```
8 VALUES ('e18c8bcc-935d-444d-a194-3a32a3b35a49', 'admin',
'$2a$10$YFhTnHpCL.Z0Ev0j1CbEUub7sIWmN7Qd5RmnU8g5ekuoapV7Zdx32',
9  'Admin', 'User', 'admin@localhost.com', '+44808123456', '', '', '', '', 'United Kingdom',
CURDATE(), 1);
10 INSERT INTO users (id, username, password, first_name, last_name, email, phone, address,
city, state, zip, country, date_created, enabled)
11 VALUES ('32e7db01-86bc-4687-9ecb-d79b265ac14f', 'user1',
'$2a$10$YFhTnHpCL.Z0Ev0j1CbEUub7sIWmN7Qd5RmnU8g5ekuoapV7Zdx32',
12  'Sam', 'Shopper', 'user1@localhost.com', '+44808123456', '1 Somewhere Street', 'London',
'Greater London', 'SW1', 'United Kingdom', CURDATE(), 1);
13 INSERT INTO users (id, username, password, first_name, last_name, email, phone, address,
city, state, zip, country, date_created, enabled, mfa)
14 VALUES ('db4cfab1-ff1d-4bca-a662-394771841383', 'user2',
'$2a$10$YFhTnHpCL.Z0Ev0j1CbEUub7sIWmN7Qd5RmnU8g5ekuoapV7Zdx32',
```

src/main/resources/data.sql, line 17 (Password Management: Hardcoded Password)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Configuration)

Audit Details

AA_Prediction Not Predicted

Sink Details

File: src/main/resources/data.sql:17

```
14 VALUES ('db4cfab1-ff1d-4bca-a662-394771841383', 'user2',
'$2a$10$YFhTnHpCL.Z0Ev0j1CbEUub7sIWmN7Qd5RmnU8g5ekuoapV7Zdx32',
15  'Sarah', 'Shopper', 'user2@localhost.com', '+44808123456', '1 Somewhere Street', 'London',
'Greater London', 'SW1', 'United Kingdom', CURDATE(), 1, 1);
16 INSERT INTO users (id, username, password, first_name, last_name, email, phone, address,
city, state, zip, country, date_created, enabled)
17 VALUES ('92a82f45-7a03-42f3-80f8-ce4e9892409d', 'api',
'$2a$10$YFhTnHpCL.Z0Ev0j1CbEUub7sIWmN7Qd5RmnU8g5ekuoapV7Zdx32',
18  'Api', 'User', 'api@localhost.com', '+44808123456', '1 Somewhere Street', 'London',
'Greater London', 'SW1', 'United Kingdom', CURDATE(), 1);
19 INSERT INTO users (id, username, password, first_name, last_name, email, phone, address,
city, state, zip, country, date_created, enabled, verify_code)
20 VALUES ('a730c051-b5c2-454c-b669-679f06d99731', 'user3',
'$2a$10$YFhTnHpCL.Z0Ev0j1CbEUub7sIWmN7Qd5RmnU8g5ekuoapV7Zdx32',
```

Password Management: Hardcoded Password — High

Package: src.main.resources

src/main/resources/application-dev.yml, line 35 (Password Management: Hardcoded Password)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: ConfigPair
File: src/main/resources/application-dev.yml:35

```
32  # h2 database
33  driver-class-name: org.h2.Driver
34  url: jdbc:h2:mem:iwa_dev
35  username: sa
36  password: password
37  initialization-mode: always
38  jpa:
```

src/main/resources/application-dev.yml, line 73 (Password Management: Hardcoded Password)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: ConfigPair
File: src/main/resources/application-dev.yml:73

```
70  name: IWA Pharmacy Direct
71  url: http://localhost:8888
72  version: 1.1
73  currency: GBP
74  invalidPasswordList: "/invalid-password-list.txt"
75  data:
76  page-size: 25
```

src/main/resources/application-test.yml, line 35 (Password Management: Hardcoded Password)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Password Management: Hardcoded Password — High

Package: src.main.resources

src/main/resources/application-test.yml, line 35 (Password Management: Hardcoded Password)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: ConfigPair
File: src/main/resources/application-test.yml:35

```
32  # h2 database
33  driver-class-name: org.h2.Driver
34  url: jdbc:h2:mem:iwa_test
35  username: sa
36  password: password
37  initialization-mode: always
38  jpa:
```

src/main/resources/application-test.yml, line 65 (Password Management: Hardcoded Password)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: ConfigPair
File: src/main/resources/application-test.yml:65

```
62 app:
63  name: IWA Pharmacy Direct
64  version: 1.0
65  currency: GBP
66  invalidPasswordList: "/invalid-password-list.txt"
67  data:
68  page-size: 25
```

Password Management: Password in Comment (14 issues)

Abstract

Storing passwords or password details in plain text anywhere in the system or system code might compromise system security in a way that cannot be easily remedied.

Explanation

It is never a good idea to hardcode a password. Storing password details within comments is equivalent to hardcoding passwords. Not only is the password visible to the project's developers, it also makes fixing the problem extremely difficult. After the code is in production, the password is leaked to the outside world and cannot be protected or changed without patching the software. If the account protected by the password is compromised, the owners of the system must choose between security and availability.

Example: The following comment specifies the default password to connect to a database:

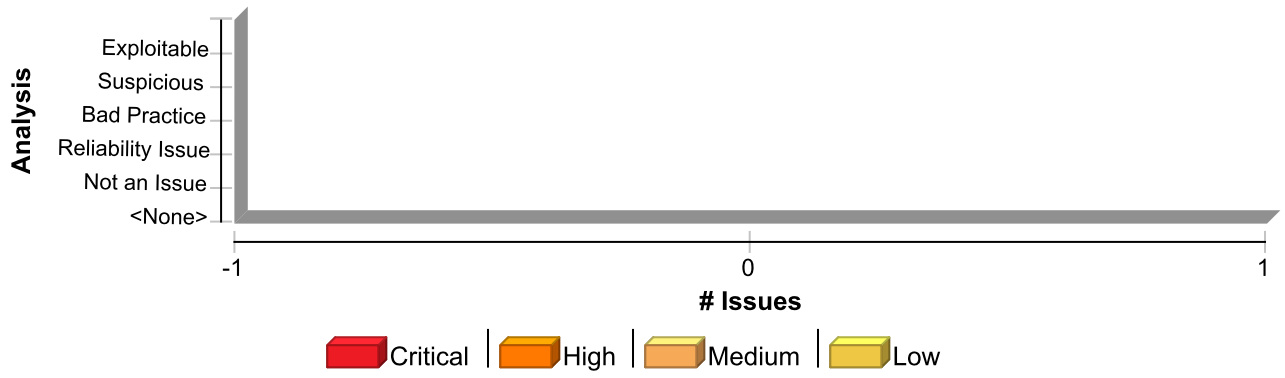
```
...
<config>
  <!-- Default username/password is scott/tiger -->
  <username></username>
  <password></password>
</config>
...
```

This code will run successfully, but anyone who has access to it will have access to the password. An employee with access to this information can use it to break into the system.

Recommendation

Passwords should never be hardcoded and should generally be obfuscated and managed in an external source. Storing passwords in plain text anywhere on the system allows anyone with sufficient permissions to read and potentially misuse the password.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Password Management: Password in Comment	14	0	0	14
Total	**14**	**0**	**0**	**14**

Password Management: Password in Comment — Low

Package: .com.microfocus.example.entity

src/main/java/com/microfocus/example/entity/User.java, line 297 (Password Management: Password in Comment)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction	Not Predicted

Sink Details

Sink: Comment
File: src/main/java/com/microfocus/example/entity/User.java:297

```
294   }
295   User utmp = new User();
296   utmp.setUsername(user.getUsername());
297   //utmp.setPassword(user.getPassword());
298   utmp.setAuthorities(authorities);
299   utmp.setEnabled(user.isEnabled());
300   return utmp;
```

Package: .com.microfocus.example.exception

src/main/java/com/microfocus/example/exception/BackupException.java, line 22 (Password Management: Password in Comment)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction	Not Predicted

Sink Details

Sink: Comment
File: src/main/java/com/microfocus/example/exception/BackupException.java:22

```
19
20 package com.microfocus.example.exception;
21
22 /**
23  * Generic Exception for handling Password errors
24  * @author Kevin A. Lee
25  */
```

src/main/java/com/microfocus/example/exception/InvalidPasswordException.java, line 22 (Password Management: Password in Comment)

Issue Details

Password Management: Password in Comment — Low

Package: .com.microfocus.example.exception

src/main/java/com/microfocus/example/exception/InvalidPasswordException.java, line 22 (Password Management: Password in Comment)

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction	Not Predicted

Sink Details

Sink: Comment
File: src/main/java/com/microfocus/example/exception/InvalidPasswordException.java:22

```
19
20 package com.microfocus.example.exception;
21
22 /**
23  * Generic Exception for handling Password errors
24  * @author Kevin A. Lee
25  */
```

src/main/java/com/microfocus/example/exception/UserLockedOutException.java, line 22 (Password Management: Password in Comment)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction	Not Predicted

Sink Details

Sink: Comment
File: src/main/java/com/microfocus/example/exception/UserLockedOutException.java:22

```
19
20 package com.microfocus.example.exception;
21
22 /**
23  * Generic Exception for handling Password errors
24  * @author Kevin A. Lee
25  */
```

Package: .com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/EncryptedPasswordUtils.java, line 32 (Password Management: Password in Comment)

Issue Details

Password Management: Password in Comment — Low

Package: .com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/EncryptedPasswordUtils.java, line 32 (Password Management: Password in Comment)

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: Comment
File: src/main/java/com/microfocus/example/utils/EncryptedPasswordUtils.java:32

```
29 import javax.crypto.SecretKey;
30 import javax.crypto.spec.SecretKeySpec;
31
32 /**
33  * Encrypted Password Utilities using BCryptPasswordEncoder
34  *
35  * @author Kevin A. Lee
```

Package: .com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/DefaultController.java, line 156 (Password Management: Password in Comment)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: Comment
File: src/main/java/com/microfocus/example/web/controllers/DefaultController.java:156

```
153  }
154
155  int otpNum = Integer.valueOf(optStr).intValue();
156  // validate OTP "one-time-password" for user
157  if (otpNum > 0) {
158  log.debug("Verifying otp '" + otp + "' of user with id: " + userId);
159  int serverOtp = verificationService.getOtp(userId);
```

src/main/java/com/microfocus/example/web/controllers/DefaultController.java, line 117 (Password Management: Password in Comment)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Password Management: Password in Comment — Low

Package: .com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/DefaultController.java, line 117 (Password Management: Password in Comment)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: Comment
File: src/main/java/com/microfocus/example/web/controllers/DefaultController.java:117

```
114  } else {
115  // create an otp
116  try {
117  // generate OTP "one-time-password" for user
118  int otp = verificationService.generateOTP(userId);
119  log.debug("Generated OTP '" + String.valueOf(otp) + "' for user id: " + userId);
120
```

Package: .com.microfocus.example.web.form

src/main/java/com/microfocus/example/web/form/PasswordForm.java, line 30 (Password Management: Password in Comment)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: Comment
File: src/main/java/com/microfocus/example/web/form/PasswordForm.java:30

```
27 import com.microfocus.example.entity.User;
28 import com.microfocus.example.web.validation.ValidPassword;
29
30 /**
31  * Form backing entity/DTO for changing password
32  * @author Kevin A. Lee
33  */
```

Package: .com.microfocus.example.web.form.admin

src/main/java/com/microfocus/example/web/form/admin/AdminPasswordForm.java, line 31 (Password Management: Password in Comment)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Password Management: Password in Comment — Low

Package: .com.microfocus.example.web.form.admin

src/main/java/com/microfocus/example/web/form/admin/AdminPasswordForm.java, line 31 (Password Management: Password in Comment)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: Comment
File: src/main/java/com/microfocus/example/web/form/admin/AdminPasswordForm.java:31

```
28 import javax.validation.constraints.NotEmpty;
29 import java.util.UUID;
30
31 /**
32  * Form backing entity/DTO for changing password
33  * @author Kevin A. Lee
34  */
```

Package: .com.microfocus.example.web.validation

src/main/java/com/microfocus/example/web/validation/ValidPassword.java, line 32 (Password Management: Password in Comment)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: Comment
File: src/main/java/com/microfocus/example/web/validation/ValidPassword.java:32

```
29 import static java.lang.annotation.ElementType.FIELD;
30 import static java.lang.annotation.RetentionPolicy.RUNTIME;
31
32 /**
33  * Interface for custom Password Validator
34  * @author Kevin A. Lee
35  */
```

src/main/java/com/microfocus/example/web/validation/PasswordConstraintValidator.java, line 121 (Password Management: Password in Comment)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Password Management: Password in Comment — Low

Package: .com.microfocus.example.web.validation

src/main/java/com/microfocus/example/web/validation/PasswordConstraintValidator.java, line 121 (Password Management: Password in Comment)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: Comment
File: src/main/java/com/microfocus/example/web/validation/PasswordConstraintValidator.java:121

```
118  // no whitespace
119  new WhitespaceRule()
120
121  // no common passwords
122  //dictionaryRule
123  ));
124
```

src/main/java/com/microfocus/example/web/validation/PasswordConstraintValidator.java, line 58 (Password Management: Password in Comment)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: Comment
File: src/main/java/com/microfocus/example/web/validation/PasswordConstraintValidator.java:58

```
55
56  private DictionaryRule dictionaryRule;
57
58 /*
59  @Value("${app.invalidPasswordList}")
60  private String invalidPasswordList = "/invalid-password-list.txt";
61
```

src/main/java/com/microfocus/example/web/validation/PasswordConstraintValidator.java, line 48 (Password Management: Password in Comment)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Password Management: Password in Comment — Low

Package: .com.microfocus.example.web.validation

src/main/java/com/microfocus/example/web/validation/PasswordConstraintValidator.java, line 48 (Password Management: Password in Comment)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: Comment
File: src/main/java/com/microfocus/example/web/validation/PasswordConstraintValidator.java:48

```
45 import org.slf4j.LoggerFactory;
46 import org.springframework.beans.factory.annotation.Value;
47
48 /**
49  * Custom Password Validator (using org.passay)
50  * @author Kevin A. Lee
51  */
```

Package: <none>

pom.xml, line 601 (Password Management: Password in Comment)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Configuration)

Audit Details

AA_Prediction Not Predicted

Sink Details

File: pom.xml:601

```
598  <includes>
599  <include>**/*IT.java</include>
600  </includes>
601  </configuration-->
602  </execution>
603  </executions>
604  </plugin>
```

Path Manipulation (19 issues)

Abstract

Allowing user input to control paths used in file system operations could enable an attacker to access or modify otherwise protected system resources.

Explanation

Path manipulation errors occur when the following two conditions are met:

1. An attacker can specify a path used in an operation on the file system.

2. By specifying the resource, the attacker gains a capability that would not otherwise be permitted.

For example, the program might give the attacker the ability to overwrite the specified file or run with a configuration controlled by the attacker.

Example 1: The following code uses input from an HTTP request to create a file name. The programmer has not considered the possibility that an attacker could provide a file name such as "`../../tomcat/conf/server.xml`", which causes the application to delete one of its own configuration files.

```
String rName = request.getParameter("reportName");
File rFile = new File("/usr/local/apfr/reports/" + rName);
...
rFile.delete();
```

Example 2: The following code uses input from a configuration file to determine which file to open and echo back to the user. If the program runs with adequate privileges and malicious users can change the configuration file, they can use the program to read any file on the system that ends with the extension `.txt`.

```
fis = new FileInputStream(cfg.getProperty("sub")+".txt");
amt = fis.read(arr);
out.println(arr);
```

Some think that in the mobile environment, classic vulnerabilities, such as path manipulation, do not make sense -- why would the user attack themself? However, keep in mind that the essence of mobile platforms is applications that are downloaded from various sources and run alongside each other on the same device. The likelihood of running a piece of malware next to a banking application is high, which necessitates expanding the attack surface of mobile applications to include inter-process communication.

Example 3: The following code adapts `Example 1` to the Android platform.

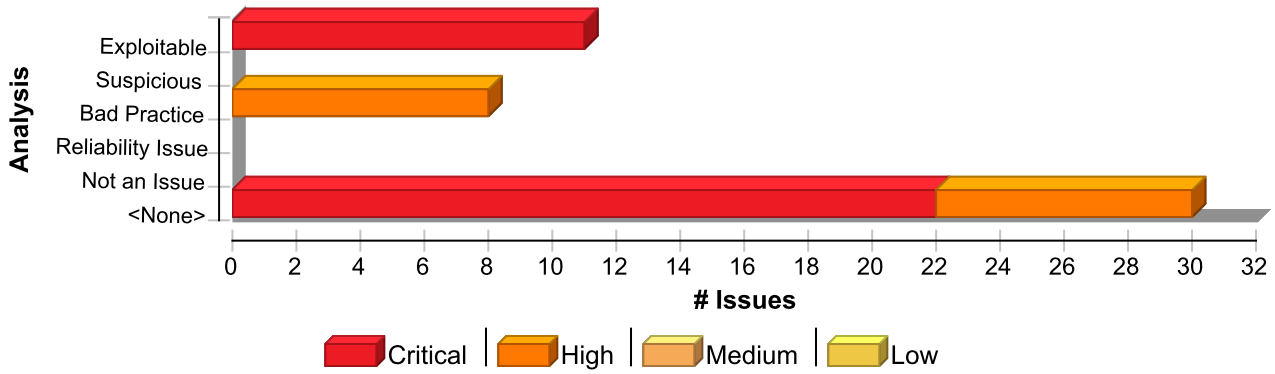
```
...
        String rName = this.getIntent().getExtras().getString("reportName");
        File rFile = getBaseContext().getFileStreamPath(rName);
...
        rFile.delete();
...
```

Recommendation

The best way to prevent path manipulation is with a level of indirection: create a list of legitimate values from which the user must select. With this approach, the user-provided input is never used directly to specify the resource name.

In some situations this approach is impractical because the set of legitimate resource names is too large or too hard to maintain. Programmers often resort to implementing a deny list in these situations. A deny list is used to selectively reject or escape potentially dangerous characters before using the input. However, any such list of unsafe characters is likely to be incomplete and will almost certainly become out of date. A better approach is to create a list of characters that are permitted to appear in the resource name and accept input composed exclusively of characters in the approved set.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Path Manipulation	19	0	0	19
Total	**19**	**0**	**0**	**19**

Path Manipulation — Critical

URL: null

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 144 (Path Manipulation)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

JiraBugLink	
Analysis	Exploitable
AA_Prediction	Exploitable
AA_Confidence	0.806

Audit Comments

admin: Thu Jul 13 2023 07:58:06 GMT-0000 (UTC)
exploitable issue

Source Details

Path Manipulation — Critical

URL: null

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 144 (Path Manipulation)

Source: serveXMLFile(0)
From: com.microfocus.example.web.controllers.UserController.serveXMLFile
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:618
URL: null

```
615
616  @GetMapping("/files/xml/{filename:.+}")
617  @ResponseBody
618  public ResponseEntity<Resource> serveXMLFile(@PathVariable String
filename) {
619
620  Resource file = storageService.loadAsResource(filename);
621  return ResponseEntity.ok().header(HttpHeaders.CONTENT_DISPOSITION,
```

Sink Details

Sink: java.nio.file.Path.resolve()
Enclosing Method: load()
File: src/main/java/com/microfocus/example/service/FileSystemStorageService.java:144
Taint Flags: WEB, XSS

```
141
142  @Override
143  public Path load(String filename) {
144  return rootLocation.resolve(filename);
145  }
146
147  @Override
```

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 55 (Path Manipulation)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

JiraBugLink	
Analysis	Exploitable
AA_Prediction	Exploitable
AA_Confidence	0.85

Source Details

Source: handleFileUpload(0)
From: com.microfocus.example.web.controllers.UserController.handleFileUpload
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:554
URL: null

Path Manipulation — Critical

URL: null

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 55 (Path Manipulation)

```
551   }
552
553   @PostMapping("/files/upload")
554   public String handleFileUpload(@RequestParam("file") MultipartFile file,
555   RedirectAttributes redirectAttributes) {
556
557   storageService.store(file);
```

Sink Details

Sink: java.nio.file.Paths.get()
Enclosing Method: store()
File: src/main/java/com/microfocus/example/service/FileSystemStorageService.java:55
Taint Flags: WEB, XSS

```
52   throw new StorageException("Failed to store empty file.");
53   }
54   Path destinationFile = this.rootLocation.resolve(
55   Paths.get(file.getOriginalFilename()))
56   .normalize().toAbsolutePath();
57   if (!destinationFile.getParent().equals(this.rootLocation.toAbsolutePath())) {
58   // This is a security check
```

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 79 (Path Manipulation)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

JiraBugLink	
Analysis	Exploitable
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.694

Source Details

Source: handleXMLUpdate(0)
From: com.microfocus.example.web.controllers.UserController.handleXMLUpdate
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:652
URL: null

Path Manipulation — Critical

URL: null

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 79 (Path Manipulation)

```
649   }
650
651   @PostMapping("/files/xml/update")
652   public String handleXMLUpdate(@RequestParam("filename") String fileName,
653   @RequestParam("fcontent") String newXMLContent,
654   RedirectAttributes redirectAttributes) {
655
```

Sink Details

Sink: java.nio.file.Paths.get()
Enclosing Method: store()
File: src/main/java/com/microfocus/example/service/FileSystemStorageService.java:79
Taint Flags: WEB, XSS

```
76   throw new StorageException("Failed to store empty file.");
77   }
78
79   Path destinationFile = Paths.get(dstFileName)
80   .normalize().toAbsolutePath();
81
82   if (!destinationFile.getParent().equals(this.rootLocation.toAbsolutePath())) {
```

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 144 (Path Manipulation)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

JiraBugLink	
Analysis	Exploitable
AA_Prediction	Exploitable
AA_Confidence	0.806

Source Details

Source: serveFile(0)
From: com.microfocus.example.web.controllers.UserController.serveFile
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:546
URL: null

Path Manipulation — Critical

URL: null

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 144 (Path Manipulation)

```
543
544  @GetMapping("/files/{filename:.+}")
545  @ResponseBody
546  public ResponseEntity<Resource> serveFile(@PathVariable String filename)
{
547
548  Resource file = storageService.loadAsResource(filename);
549  return ResponseEntity.ok().header(HttpHeaders.CONTENT_DISPOSITION,
```

Sink Details

Sink: java.nio.file.Path.resolve()
Enclosing Method: load()
File: src/main/java/com/microfocus/example/service/FileSystemStorageService.java:144
Taint Flags: WEB, XSS

```
141
142  @Override
143  public Path load(String filename) {
144  return rootLocation.resolve(filename);
145  }
146
147  @Override
```

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 144 (Path Manipulation)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

JiraBugLink	
Analysis	Exploitable
AA_Prediction	Exploitable
AA_Confidence	0.821

Source Details

Source: serveUnverifiedFile(0)
From: com.microfocus.example.web.controllers.UserController.serveUnverifiedFile
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:696
URL: null

Path Manipulation — Critical

URL: null

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 144 (Path Manipulation)

```
693   }
694
695   @GetMapping("/files/download/unverified")
696   public ResponseEntity<?> serveUnverifiedFile(@Param("file") String file)
{
697
698   if (Objects.isNull(file) || file.isEmpty()) {
699   return ResponseEntity.badRequest().build();
```

Sink Details

Sink: java.nio.file.Path.resolve()
Enclosing Method: load()
File: src/main/java/com/microfocus/example/service/FileSystemStorageService.java:144
Taint Flags: WEB, XSS

```
141
142   @Override
143   public Path load(String filename) {
144   return rootLocation.resolve(filename);
145   }
146
147   @Override
```

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 55 (Path Manipulation)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

JiraBugLink	
Analysis	Exploitable
AA_Prediction	Exploitable
AA_Confidence	0.85

Source Details

Source: handleXMLFileUpload(0)
From: com.microfocus.example.web.controllers.UserController.handleXMLFileUpload
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:626
URL: null

Path Manipulation — Critical

URL: null

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 55 (Path Manipulation)

```
623   }
624
625   @PostMapping("/files/upload-xml")
626   public String handleXMLFileUpload(@RequestParam("file") MultipartFile
file,
627   RedirectAttributes redirectAttributes) {
628
629   storageService.store(file);
```

Sink Details

Sink: java.nio.file.Paths.get()
Enclosing Method: store()
File: src/main/java/com/microfocus/example/service/FileSystemStorageService.java:55
Taint Flags: WEB, XSS

```
52   throw new StorageException("Failed to store empty file.");
53   }
54   Path destinationFile = this.rootLocation.resolve(
55   Paths.get(file.getOriginalFilename()))
56   .normalize().toAbsolutePath();
57   if (!destinationFile.getParent().equals(this.rootLocation.toAbsolutePath())) {
58   // This is a security check
```

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 144 (Path Manipulation)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

JiraBugLink	
Analysis	Exploitable
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.739

Source Details

Source: handleXMLUpdate(0)
From: com.microfocus.example.web.controllers.UserController.handleXMLUpdate
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:652
URL: null

Path Manipulation — Critical

URL: null

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 144 (Path Manipulation)

```
649   }
650
651   @PostMapping("/files/xml/update")
652   public String handleXMLUpdate(@RequestParam("filename") String fileName,
653   @RequestParam("fcontent") String newXMLContent,
654   RedirectAttributes redirectAttributes) {
655
```

Sink Details

Sink: java.nio.file.Path.resolve()
Enclosing Method: load()
File: src/main/java/com/microfocus/example/service/FileSystemStorageService.java:144
Taint Flags: WEB, XSS

```
141
142   @Override
143   public Path load(String filename) {
144   return rootLocation.resolve(filename);
145   }
146
147   @Override
```

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 160 (Path Manipulation)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

JiraBugLink	
Analysis	Exploitable
AA_Prediction	Exploitable
AA_Confidence	0.821

Source Details

Source: serveFile(0)
From: com.microfocus.example.web.controllers.UserController.serveFile
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:546
URL: null

Path Manipulation — Critical

URL: null

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 160 (Path Manipulation)

```
543
544  @GetMapping("/files/{filename:.+}")
545  @ResponseBody
546  public ResponseEntity<Resource> serveFile(@PathVariable String filename)
{
547
548  Resource file = storageService.loadAsResource(filename);
549  return ResponseEntity.ok().header(HttpHeaders.CONTENT_DISPOSITION,
```

Sink Details

Sink: java.nio.file.Paths.get()
Enclosing Method: loadAsResource()
File: src/main/java/com/microfocus/example/service/FileSystemStorageService.java:160
Taint Flags: WEB, XSS

```
157  try {
158  Path file = null;
159  if (traverse) {
160  file = Paths.get(filename);
161  } else {
162  file = load(filename);
163  }
```

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 160 (Path Manipulation)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

JiraBugLink	
Analysis	Exploitable
AA_Prediction	Exploitable
AA_Confidence	0.821

Source Details

Source: serveXMLFile(0)
From: com.microfocus.example.web.controllers.UserController.serveXMLFile
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:618
URL: null

Path Manipulation — Critical

URL: null

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 160 (Path Manipulation)

```
615
616   @GetMapping("/files/xml/{filename:.+}")
617   @ResponseBody
618   public ResponseEntity<Resource> serveXMLFile(@PathVariable String
filename) {
619
620   Resource file = storageService.loadAsResource(filename);
621   return ResponseEntity.ok().header(HttpHeaders.CONTENT_DISPOSITION,
```

Sink Details

Sink: java.nio.file.Paths.get()
Enclosing Method: loadAsResource()
File: src/main/java/com/microfocus/example/service/FileSystemStorageService.java:160
Taint Flags: WEB, XSS

```
157   try {
158   Path file = null;
159   if (traverse) {
160   file = Paths.get(filename);
161   } else {
162   file = load(filename);
163   }
```

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 160 (Path Manipulation)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

JiraBugLink	
Analysis	Exploitable
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.739

Source Details

Source: serveUnverifiedFile(0)
From: com.microfocus.example.web.controllers.UserController.serveUnverifiedFile
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:696
URL: null

Path Manipulation — Critical

URL: null

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 160 (Path Manipulation)

```
693  }
694
695  @GetMapping("/files/download/unverified")
696  public ResponseEntity<?> serveUnverifiedFile(@Param("file") String file)
{
697
698  if (Objects.isNull(file) || file.isEmpty()) {
699  return ResponseEntity.badRequest().build();
```

Sink Details

Sink: java.nio.file.Paths.get()
Enclosing Method: loadAsResource()
File: src/main/java/com/microfocus/example/service/FileSystemStorageService.java:160
Taint Flags: WEB, XSS

```
157  try {
158  Path file = null;
159  if (traverse) {
160  file = Paths.get(filename);
161  } else {
162  file = load(filename);
163  }
```

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 148 (Path Manipulation)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

JiraBugLink	
Analysis	Exploitable
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.539

Source Details

Source: downloadFile(1)
From: com.microfocus.example.web.controllers.ProductController.downloadFile
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:136
URL: null

Path Manipulation — Critical

URL: null

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 148 (Path Manipulation)

```
133
134  @GetMapping("/{id}/download/{fileName:.+}")
135  public ResponseEntity<Resource> downloadFile(@PathVariable(value = "id")
UUID productId,
136  @PathVariable String fileName, HttpServletRequest request) {
137  Resource resource;
138  File dataDir;
139  try {
```

Sink Details

Sink: java.nio.file.Paths.get()
Enclosing Method: downloadFile()
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:148
Taint Flags: WEB, XSS

```
145
146  log.debug("Using data directory: " + dataDir.getAbsolutePath());
147  String fileBasePath = dataDir.getAbsolutePath() + File.separatorChar +
productId.toString() + File.separatorChar;
148  Path path = Paths.get(fileBasePath + fileName);
149  try {
150  resource = new UrlResource(path.toUri());
151  } catch (MalformedURLException e) {
```

Path Manipulation — High

Package: com.microfocus.example.service

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 79 (Path Manipulation)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.567

Source Details

Source: Read this.location
From: com.microfocus.example.config.StorageProperties.getLocation
File: src/main/java/com/microfocus/example/config/StorageProperties.java:16

Path Manipulation — High

Package: com.microfocus.example.service

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 79 (Path Manipulation)

```
13  private String location = System.getProperty("user.home") +
File.separatorChar + "upload-dir";
14
15  public String getLocation() {
16  return location;
17  }
18
19  public void setLocation(String location) {
```

Sink Details

Sink: java.nio.file.Paths.get()
Enclosing Method: store()
File: src/main/java/com/microfocus/example/service/FileSystemStorageService.java:79
Taint Flags: PROPERTY

```
76  throw new StorageException("Failed to store empty file.");
77  }
78
79  Path destinationFile = Paths.get(dstFileName)
80  .normalize().toAbsolutePath();
81
82  if (!destinationFile.getParent().equals(this.rootLocation.toAbsolutePath())) {
```

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 144 (Path Manipulation)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.634

Source Details

Source: Read this.location
From: com.microfocus.example.config.StorageProperties.getLocation
File: src/main/java/com/microfocus/example/config/StorageProperties.java:16

Path Manipulation — High

Package: com.microfocus.example.service

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 144 (Path Manipulation)

```
13  private String location = System.getProperty("user.home") +
File.separatorChar + "upload-dir";
14
15  public String getLocation() {
16  return location;
17  }
18
19  public void setLocation(String location) {
```

Sink Details

Sink: java.nio.file.Path.resolve()
Enclosing Method: load()
File: src/main/java/com/microfocus/example/service/FileSystemStorageService.java:144
Taint Flags: PROPERTY

```
141
142  @Override
143  public Path load(String filename) {
144  return rootLocation.resolve(filename);
145  }
146
147  @Override
```

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 182 (Path Manipulation)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.664

Source Details

Source: Read this.location
From: com.microfocus.example.config.StorageProperties.getLocation
File: src/main/java/com/microfocus/example/config/StorageProperties.java:16

Path Manipulation — High

Package: com.microfocus.example.service

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 182 (Path Manipulation)

```
13  private String location = System.getProperty("user.home") +
File.separatorChar + "upload-dir";
14
15  public String getLocation() {
16  return location;
17  }
18
19  public void setLocation(String location) {
```

Sink Details

Sink: java.nio.file.Path.toFile()
Enclosing Method: deleteAll()
File: src/main/java/com/microfocus/example/service/FileSystemStorageService.java:182
Taint Flags: PROPERTY

```
179
180  @Override
181  public void deleteAll() {
182  FileSystemUtils.deleteRecursively(rootLocation.toFile());
183  }
184
185  @Override
```

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 37 (Path Manipulation)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.611

Source Details

Source: Read this.location
From: com.microfocus.example.config.StorageProperties.getLocation
File: src/main/java/com/microfocus/example/config/StorageProperties.java:16

Path Manipulation — High

Package: com.microfocus.example.service

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 37 (Path Manipulation)

```
13  private String location = System.getProperty("user.home") +
File.separatorChar + "upload-dir";
14
15  public String getLocation() {
16  return location;
17  }
18
19  public void setLocation(String location) {
```

Sink Details

Sink: java.nio.file.Paths.get()
Enclosing Method: FileSystemStorageService()
File: src/main/java/com/microfocus/example/service/FileSystemStorageService.java:37
Taint Flags: PROPERTY

```
34
35  @Autowired
36  public FileSystemStorageService(StorageProperties properties) {
37  this.rootLocation = Paths.get(properties.getLocation());
38  if (!Files.exists(this.rootLocation)) {
39  log.debug("Creating storage service directory: " + rootLocation.toString());
40  try {
```

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/UserUtils.java, line 79 (Path Manipulation)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.698

Source Details

Source: java.lang.System.getProperty()
From: com.microfocus.example.utils.UserUtils.getFilePath
File: src/main/java/com/microfocus/example/utils/UserUtils.java:138

Path Manipulation — High

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/UserUtils.java, line 79 (Path Manipulation)

```
135   }
136
137   private static String getFilePath(String relativePath) {
138   return System.getProperty("user.home") + File.separatorChar +
relativePath;
139   }
140
141 }
```

Sink Details

Sink: java.io.File.File()
Enclosing Method: registerUser()
File: src/main/java/com/microfocus/example/utils/UserUtils.java:79
Taint Flags: FILE_SYSTEM, NO_NEW_LINE, SYSTEMINFO

```
76   JSONParser jsonParser = new JSONParser();
77   JSONArray jsonArray = new JSONArray();
78
79   File dataFile = new File(getFilePath(NEWSLETTER_USER_FILE));
80   if (dataFile.exists()) {
81   jsonArray = (JSONArray) jsonParser.parse(new
FileReader(getFilePath(NEWSLETTER_USER_FILE)));
82   } else {
```

src/main/java/com/microfocus/example/utils/UserUtils.java, line 50 (Path Manipulation)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.698

Source Details

Source: java.lang.System.getProperty()
From: com.microfocus.example.utils.UserUtils.getFilePath
File: src/main/java/com/microfocus/example/utils/UserUtils.java:138

Path Manipulation — High

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/UserUtils.java, line 50 (Path Manipulation)

```
135   }
136
137   private static String getFilePath(String relativePath) {
138   return System.getProperty("user.home") + File.separatorChar +
relativePath;
139   }
140
141 }
```

Sink Details

Sink: java.io.File.File()
Enclosing Method: writeUser()
File: src/main/java/com/microfocus/example/utils/UserUtils.java:50
Taint Flags: FILE_SYSTEM, NO_NEW_LINE, SYSTEMINFO

```
47  public static void writeUser(String username, String password) throws IOException {
48  JsonFactory jsonFactory = new JsonFactory();
49
50  File dataFile = new File(getFilePath(USER_INFO_FILE));
51  if (dataFile.createNewFile()){
52  log.debug("Created: " + getFilePath(USER_INFO_FILE));
53  }
```

src/main/java/com/microfocus/example/utils/UserUtils.java, line 81 (Path Manipulation)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.698

Source Details

Source: java.lang.System.getProperty()
From: com.microfocus.example.utils.UserUtils.getFilePath
File: src/main/java/com/microfocus/example/utils/UserUtils.java:138

Path Manipulation — High

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/UserUtils.java, line 81 (Path Manipulation)

```
135   }
136
137   private static String getFilePath(String relativePath) {
138   return System.getProperty("user.home") + File.separatorChar +
relativePath;
139   }
140
141 }
```

Sink Details

Sink: java.io.FileReader.FileReader()
Enclosing Method: registerUser()
File: src/main/java/com/microfocus/example/utils/UserUtils.java:81
Taint Flags: FILE_SYSTEM, NO_NEW_LINE, SYSTEMINFO

```
78
79   File dataFile = new File(getFilePath(NEWSLETTER_USER_FILE));
80   if (dataFile.exists()) {
81   jsonArray = (JSONArray) jsonParser.parse(new
FileReader(getFilePath(NEWSLETTER_USER_FILE)));
82   } else {
83   dataFile.createNewFile();
84   log.debug("Created: " + getFilePath(NEWSLETTER_USER_FILE));
```

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 600 (Path Manipulation)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.666

Source Details

Source: Read this.location
From: com.microfocus.example.config.StorageProperties.getLocation
File: src/main/java/com/microfocus/example/config/StorageProperties.java:16

Path Manipulation — High

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 600 (Path Manipulation)

```
13  private String location = System.getProperty("user.home") +
File.separatorChar + "upload-dir";
14
15  public String getLocation() {
16  return location;
17  }
18
19  public void setLocation(String location) {
```

Sink Details

Sink: java.nio.file.Path.toFile()
Enclosing Method: getXMLFileContent()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:600
Taint Flags: PROPERTY

```
597  try {
598  dbf.setFeature(XMLConstants.FEATURE_SECURE_PROCESSING, false);
599  DocumentBuilder db = dbf.newDocumentBuilder();
600  Document doc = db.parse(fpath.toFile());
601  try (ByteArrayOutputStream bytesOutStream = new ByteArrayOutputStream()) {
602  writeXml(doc, bytesOutStream);
603  xmlContent = bytesOutStream.toString();
```

Poor Error Handling: Empty Catch Block (1 issue)

Abstract

Ignoring an exception can cause the program to overlook unexpected states and conditions.

Explanation

Just about every serious attack on a software system begins with the violation of a programmer's assumptions. After the attack, the programmer's assumptions seem flimsy and poorly founded, but before an attack many programmers would defend their assumptions well past the end of their lunch break.

Two dubious assumptions that are easy to spot in code are "this method call can never fail" and "it doesn't matter if this call fails". When a programmer ignores an exception, they implicitly state that they are operating under one of these assumptions.

Example 1: The following code excerpt ignores a rarely-thrown exception from `doExchange()`.

```
try {
  doExchange();
}
catch (RareException e) {
  // this can never happen
}
```

If a `RareException` were to ever be thrown, the program would continue to execute as though nothing unusual had occurred. The program records no evidence indicating the special situation, potentially frustrating any later attempt to explain the program's behavior.

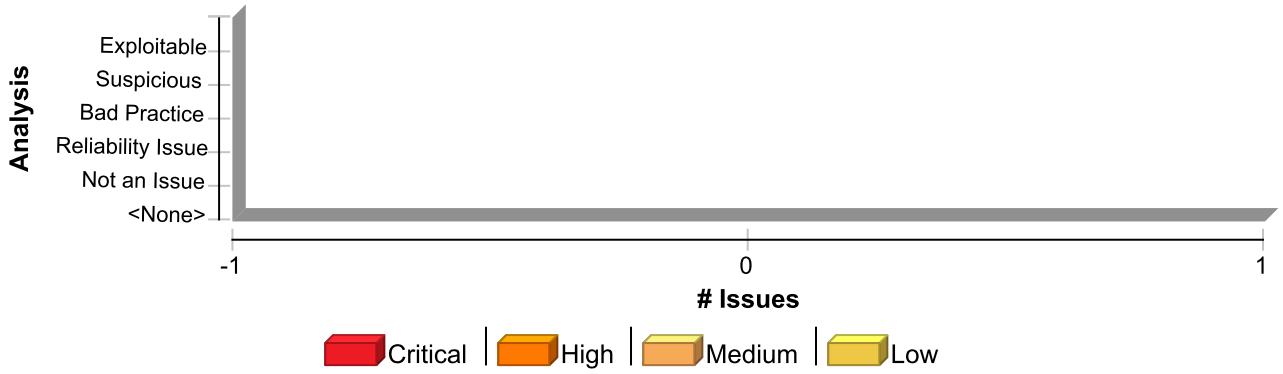
Recommendation

At a minimum, log the fact that the exception was thrown so that it will be possible to come back later and make sense of the resulting program behavior. Better yet, abort the current operation. If the exception is being ignored because the caller cannot properly handle it but the context makes it inconvenient or impossible for the caller to declare that it throws the exception itself, consider throwing a `RuntimeException` or an `Error`, both of which are unchecked exceptions. As of JDK 1.4, `RuntimeException` has a constructor that makes it easy to wrap another exception.

Example 2: The code in `Example 1` could be rewritten in the following way:

```
try {
  doExchange();
}
catch (RareException e) {
  throw new RuntimeException("This can never happen", e);
}
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Poor Error Handling: Empty Catch Block	1	0	0	1
Total	**1**	**0**	**0**	**1**

Poor Error Handling: Empty Catch Block — Low

Package: com.microfocus.example.service

src/main/java/com/microfocus/example/service/CustomUserDetailsService.java, line 61 (Poor Error Handling: Empty Catch Block)

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: CatchBlock
Enclosing Method: loadUserByUsername()
File: src/main/java/com/microfocus/example/service/CustomUserDetailsService.java:61

```
58  if (!user.isPresent()) {
59  throw new UsernameNotFoundException("User with email: " + username + " not found.");
60  }
61  } catch (UserLockedOutException ignored) {
62  // Do something here
63  }
64  return new CustomUserDetails(user.get());
```

Poor Error Handling: Overly Broad Catch (2 issues)

Abstract

The catch block handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

Explanation

Multiple catch blocks can get repetitive, but "condensing" catch blocks by catching a high-level class such as `Exception` can obscure exceptions that deserve special treatment or that should not be caught at this point in the program. Catching an overly broad exception essentially defeats the purpose of Java's typed exceptions, and can become particularly dangerous if the program grows and begins to throw new types of exceptions. The new exception types will not receive any attention.

Example: The following code excerpt handles three types of exceptions in an identical fashion.

```
try {
  doExchange();
}
catch (IOException e) {
  logger.error("doExchange failed", e);
}
catch (InvocationTargetException e) {
  logger.error("doExchange failed", e);
}
catch (SQLException e) {
  logger.error("doExchange failed", e);
}
```

At first blush, it may seem preferable to deal with these exceptions in a single catch block, as follows:

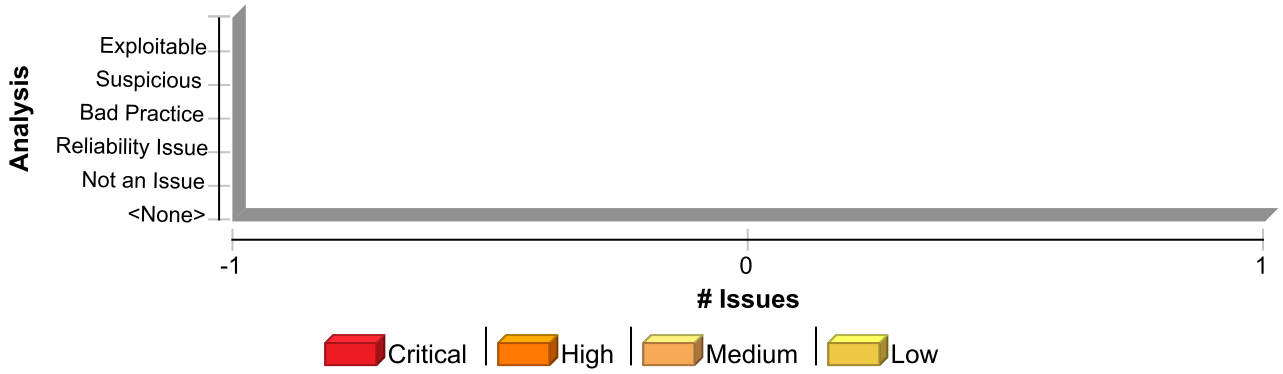
```
try {
  doExchange();
}
catch (Exception e) {
  logger.error("doExchange failed", e);
}
```

However, if `doExchange()` is modified to throw a new type of exception that should be handled in some different kind of way, the broad catch block will prevent the compiler from pointing out the situation. Further, the new catch block will now also handle exceptions derived from `RuntimeException` such as `ClassCastException`, and `NullPointerException`, which is not the programmer's intent.

Recommendation

Do not catch broad exception classes such as `Exception`, `Throwable`, `Error`, or `RuntimeException` except at the very top level of the program or thread.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Poor Error Handling: Overly Broad Catch	2	0	0	2
Total	**2**	**0**	**0**	**2**

Poor Error Handling: Overly Broad Catch — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/AuthenticationTokenFilter.java, line 66 (Poor Error Handling: Overly Broad Catch)

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: CatchBlock
Enclosing Method: doFilterInternal()
File: src/main/java/com/microfocus/example/config/handlers/AuthenticationTokenFilter.java:66

```
63
64  SecurityContextHolder.getContext().setAuthentication(authentication);
65  }
66  } catch (Exception e) {
67  logger.error("Cannot set user authentication: {}", e);
68  }
69
```

Package: com.microfocus.example.service

src/main/java/com/microfocus/example/service/VerificationService.java, line 69 (Poor Error Handling: Overly Broad Catch)

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Poor Error Handling: Overly Broad Catch — Low

Package: com.microfocus.example.service

src/main/java/com/microfocus/example/service/VerificationService.java, line 69 (Poor Error Handling: Overly Broad Catch)

Sink Details

Sink: CatchBlock
Enclosing Method: getOtp()
File: src/main/java/com/microfocus/example/service/VerificationService.java:69

```
66  public int getOtp(String key){
67  try {
68  return otpCache.get(key);
69  } catch (Exception ex){
70  log.error(ex.getLocalizedMessage());
71  return 0;
72  }
```

Poor Error Handling: Overly Broad Throws (3 issues)

Abstract

The method throws a generic exception making it harder for callers to do a good job of error handling and recovery.

Explanation

Declaring a method to throw `Exception` or `Throwable` makes it difficult for callers to do good error handling and error recovery. Java's exception mechanism is set up to make it easy for callers to anticipate what can go wrong and write code to handle each specific exceptional circumstance. Declaring that a method throws a generic form of exception defeats this system.

Example: The following method throws three types of exceptions.

```
public void doExchange()
  throws IOException, InvocationTargetException,
         SQLException {
  ...
}
```

While it might seem tidier to write

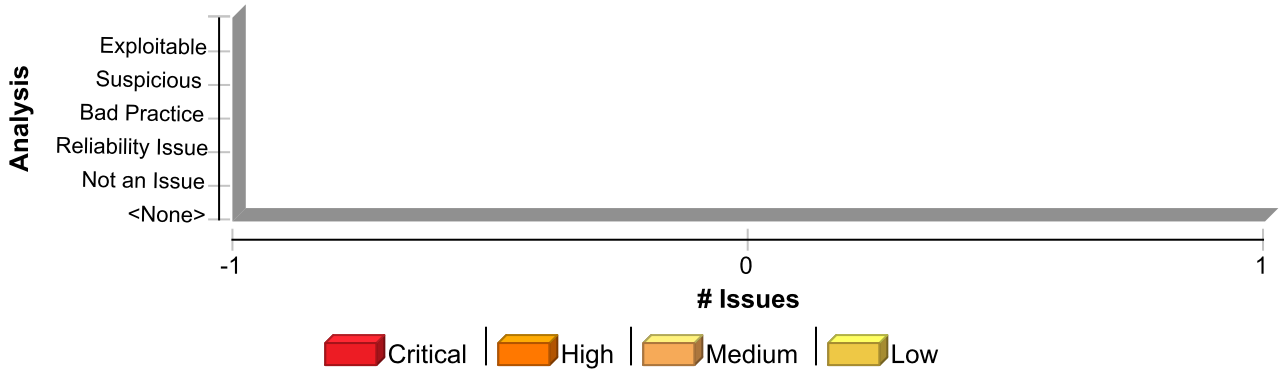
```
public void doExchange()
  throws Exception {
  ...
}
```

doing so hampers the caller's ability to understand and handle the exceptions that occur. Further, if a later revision of `doExchange()` introduces a new type of exception that should be treated differently than previous exceptions, there is no easy way to enforce this requirement.

Recommendation

Do not declare methods to throw `Exception` or `Throwable`. If the exceptions thrown by a method are not recoverable or should not generally be caught by the caller, consider throwing unchecked exceptions rather than checked exceptions. This can be accomplished by implementing exception classes that extend `RuntimeException` or `Error` instead of `Exception`, or add a try/catch wrapper in your method to convert checked exceptions to unchecked exceptions.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Poor Error Handling: Overly Broad Throws	3	0	0	3
Total	**3**	**0**	**0**	**3**

Poor Error Handling: Overly Broad Throws — Low

Package: com.microfocus.example.config

src/main/java/com/microfocus/example/config/WebSecurityConfiguration.java, line 84 (Poor Error Handling: Overly Broad Throws)

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: Function: configureGlobal
Enclosing Method: configureGlobal()
File: src/main/java/com/microfocus/example/config/WebSecurityConfiguration.java:84

```
81  private String activeProfile;
82
83  @Autowired
84  public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
85  auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());
86  }
87
```

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/GlobalExceptionHandler.java, line 41 (Poor Error Handling: Overly Broad Throws)

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Poor Error Handling: Overly Broad Throws — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/GlobalExceptionHandler.java, line 41 (Poor Error Handling: Overly Broad Throws)

Sink Details

Sink: Function: handleAll
Enclosing Method: handleAll()
File: src/main/java/com/microfocus/example/config/handlers/GlobalExceptionHandler.java:41

```
38  public static final String DEFAULT_ERROR_VIEW = "error/default";
39
40  @ExceptionHandler({Exception.class})
41  public ModelAndView handleAll(HttpServletRequest request, final Exception ex) throws
Exception {
42  log.debug("GlobalExceptionHandler::handleAll");
43  log.error("error: " + ex.toString());
44  // If the exception is annotated with @ResponseStatus rethrow it and let
```

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/EmailUtils.java, line 77 (Poor Error Handling: Overly Broad Throws)

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction	Not Predicted

Sink Details

Sink: Function: sendEmail
Enclosing Method: sendEmail()
File: src/main/java/com/microfocus/example/utils/EmailUtils.java:77

```
74  EmailUtils.EMAIL_PASSWORD = emailPassword;
75  }
76
77  public static void sendEmail(EmailRequest request) throws Exception {
78
79  Email server = new SimpleEmail();
80  server.setHostName(EMAIL_SERVER);
```

Poor Style: Value Never Read (7 issues)

Abstract

The variable's value is assigned but never used, making it a dead store.

Explanation

This variable's value is not used. After the assignment, the variable is either assigned another value or goes out of scope.

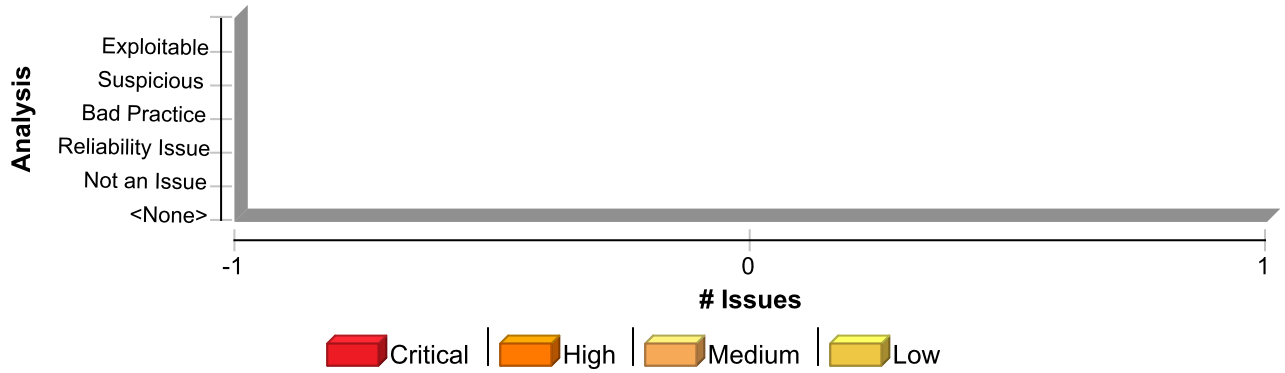
Example: The following code excerpt assigns to the variable `r` and then overwrites the value without using it.

```
r = getName();
r = getNewBuffer(buf);
```

Recommendation

Remove unnecessary assignments in order to make the code easier to understand and maintain.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Poor Style: Value Never Read	7	0	0	7
Total	**7**	**0**	**0**	**7**

Poor Style: Value Never Read — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/CustomAuthenticationSuccessHandler.java, line 73 (Poor Style: Value Never Read)

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction	Not Predicted

Poor Style: Value Never Read — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/CustomAuthenticationSuccessHandler.java, line 73 (Poor Style: Value Never Read)

Sink Details

Sink: VariableAccess: userId
Enclosing Method: onAuthenticationSuccess()
File: src/main/java/com/microfocus/example/config/handlers/CustomAuthenticationSuccessHandler.java:73

```
70
71  CustomUserDetails customUserDetails = (CustomUserDetails) authentication.getPrincipal();
72  Boolean mfa = customUserDetails.getMfa();
73  UUID userId = customUserDetails.getId();
74  String mobile = customUserDetails.getMobile();
75  boolean isAdmin = customUserDetails.getAuthorities().stream().anyMatch(a ->
a.getAuthority().equals("ROLE_ADMIN"));
76
```

src/main/java/com/microfocus/example/config/handlers/CustomAuthenticationSuccessHandler.java, line 103 (Poor Style: Value Never Read)

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: VariableAccess: isUser
Enclosing Method: getTargetUrl()
File: src/main/java/com/microfocus/example/config/handlers/CustomAuthenticationSuccessHandler.java:103

```
100  HttpSession session = request.getSession(false);
101  CustomUserDetails customUserDetails = (CustomUserDetails) authentication.getPrincipal();
102  boolean isAdmin = customUserDetails.getAuthorities().stream().anyMatch(a ->
a.getAuthority().equals("ROLE_ADMIN"));
103  boolean isUser = !isAdmin;
104  String targetUrl = INDEX_URL;
105
106  if (isAdmin) {
```

src/main/java/com/microfocus/example/config/handlers/CustomAuthenticationSuccessHandler.java, line 160 (Poor Style: Value Never Read)

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Poor Style: Value Never Read — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/CustomAuthenticationSuccessHandler.java, line 160 (Poor Style: Value Never Read)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: VariableAccess: jwtToken
Enclosing Method: bypassVerification()
File: src/main/java/com/microfocus/example/config/handlers/CustomAuthenticationSuccessHandler.java:160

```
157
158  private void bypassVerification(HttpServletRequest request, HttpServletResponse response,
159  Authentication authentication) throws IOException {
160  String jwtToken = jwtUtils.generateAndSetSession(request, response, authentication);
161  String targetUrl = getTargetUrl(request, response, authentication);
162  log.debug("Redirecting to: " + targetUrl);
163  redirectStrategy.sendRedirect(request, response, targetUrl);
```

Package: com.microfocus.example.repository

src/main/java/com/microfocus/example/repository/UserRepositoryImpl.java, line 70 (Poor Style: Value Never Read)

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: VariableAccess: authorityCount
Enclosing Method: findUserByUsername()
File: src/main/java/com/microfocus/example/repository/UserRepositoryImpl.java:70

```
67  List<User> users = new ArrayList<>();
68
69  Session session = entityManager.unwrap(Session.class);
70  Integer authorityCount = session.doReturningWork(new ReturningWork<Integer>() {
71
72  @Override
73  public Integer execute(Connection con) throws SQLException {
```

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/DefaultController.java, line 109 (Poor Style: Value Never Read)

Issue Details

Poor Style: Value Never Read — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/DefaultController.java, line 109 (Poor Style: Value Never Read)

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: VariableAccess: email
Enclosing Method: otpLogin()
File: src/main/java/com/microfocus/example/web/controllers/DefaultController.java:109

```
106  public String otpLogin(HttpServletRequest request, Model model, Principal principal) {
107  CustomUserDetails loggedInUser = (CustomUserDetails) ((Authentication)
principal).getPrincipal();
108  String userId = loggedInUser.getId().toString();
109  String email = loggedInUser.getEmail();
110  String mobile = loggedInUser.getMobile();
111  log.debug("Verifying user with id: " + userId);
112  if (model.containsAttribute("otp")) {
```

src/main/java/com/microfocus/example/web/controllers/DefaultController.java, line 126 (Poor Style: Value Never Read)

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: VariableAccess: sid
Enclosing Method: otpLogin()
File: src/main/java/com/microfocus/example/web/controllers/DefaultController.java:126

```
123  sms.setMessage("Your IWA Pharmacy Direct security code is " + String.valueOf(otp));
124
125  try {
126  String sid = smsSenderService.sendSms(sms);
127  } catch (Exception ex) {
128  log.error(ex.getLocalizedMessage());
129  }
```

src/main/java/com/microfocus/example/web/controllers/DefaultController.java, line 176 (Poor Style: Value Never Read)

Issue Details

Poor Style: Value Never Read — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/DefaultController.java, line 176 (Poor Style: Value Never Read)

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: VariableAccess: jwtToken
Enclosing Method: otpLogin()
File: src/main/java/com/microfocus/example/web/controllers/DefaultController.java:176

```
173  } else {
174  // TODO: fail
175  }
176  String jwtToken = jwtUtils.generateAndSetSession(request, response, authentication);
177  String targetUrl = CustomAuthenticationSuccessHandler.getTargetUrl(request, response,
authentication);
178  return "redirect:"+targetUrl;
179  }
```

Privacy Violation (1 issue)

Abstract

Mishandling private information, such as customer passwords or social security numbers, can compromise user privacy and is often illegal.

Explanation

Privacy violations occur when:

1. Private user information enters the program.

2. The data is written to an external location, such as the console, file system, or network.

Example 1: The following code contains a logging statement that tracks the records added to a database by storing the contents in a log file.

```
pass = getPassword();
...
dbmsLog.println(id+":"+pass+":"+type+":"+tstamp);
```

The code in `Example 1` logs a plain text password to the file system. Although many developers trust the file system as a safe storage location for data, it should not be trusted implicitly, particularly when privacy is a concern.

Privacy is one of the biggest concerns in the mobile world for a couple of reasons. One of them is a much higher chance of device loss. The other has to do with inter-process communication between mobile applications. With mobile platforms, applications are downloaded from various sources and are run alongside each other on the same device. The likelihood of running a piece of malware next to a banking application is high, which is why application authors need to be careful about what information they include in messages addressed to other applications running on the device. Sensitive information should never be part of inter-process communication between mobile applications.

Example 2: The following code reads username and password for a given site from an Android WebView store and broadcasts them to all the registered receivers.

```
...
webview.setWebViewClient(new WebViewClient() {
  public void onReceivedHttpAuthRequest(WebView view,
        HttpAuthHandler handler, String host, String realm) {
    String[] credentials = view.getHttpAuthUsernamePassword(host, realm);
    String username = credentials[0];
    String password = credentials[1];
    Intent i = new Intent();
    i.setAction("SEND_CREDENTIALS");
    i.putExtra("username", username);
    i.putExtra("password", password);
    view.getContext().sendBroadcast(i);
  }
});
...
```

This example demonstrates several problems. First of all, by default, WebView credentials are stored in plain text and are not hashed. If a user has a rooted device (or uses an emulator), they can read stored passwords for given sites. Second, plain text credentials are broadcast to all the registered receivers, which means that any receiver registered to listen to intents with the `SEND_CREDENTIALS` action will receive the message. The broadcast is not even protected with a permission to limit the number of recipients, although in this case we do not recommend using permissions as a fix.

Private data can enter a program in a variety of ways:

- Directly from the user in the form of a password or personal information

- Accessed from a database or other data store by the application

- Indirectly from a partner or other third party

Typically, in the context of the mobile environment, this private information includes (along with passwords, SSNs, and other general personal information):

- Location

- Cell phone number

- Serial numbers and device IDs

- Network Operator information

- Voicemail information

Sometimes data that is not labeled as private can have a privacy implication in a different context. For example, student identification numbers are usually not considered private because there is no explicit and publicly-available mapping to an individual student's personal information. However, if a school generates identification numbers based on student social security numbers, then the identification numbers should be considered private.

Security and privacy concerns often seem to compete with each other. From a security perspective, you should record all important operations so that any anomalous activity can later be identified. However, when private data is involved, this practice can create risk.

Although there are many ways in which private data can be handled unsafely, a common risk stems from misplaced trust. Programmers often trust the operating environment in which a program runs, and therefore believe that it is acceptable to store private information on the file system, in the registry, or in other locally-controlled resources. However, even if access to certain resources is restricted, this does not guarantee that the individuals who do have access can be trusted. For example, in 2004, an unscrupulous employee at AOL sold approximately 92 million private customer email addresses to a spammer marketing an offshore gambling web site [1].

In response to such high-profile exploits, the collection and management of private data is becoming increasingly regulated. Depending on its location, the type of business it conducts, and the nature of any private data it handles, an organization may be required to comply with one or more of the following federal and state regulations:

- Safe Harbor Privacy Framework [3]

- Gramm-Leach Bliley Act (GLBA) [4]

- Health Insurance Portability and Accountability Act (HIPAA) [5]

- California SB-1386 [6]

Despite these regulations, privacy violations continue to occur with alarming frequency.

Recommendation

When security and privacy demands clash, privacy should usually be given the higher priority. To accomplish this and still maintain required security information, cleanse any private information before it exits the program.

To enforce good privacy management, develop and strictly adhere to internal privacy guidelines. The guidelines should specifically describe how an application should handle private data. If your organization is regulated by federal or state law, ensure that your privacy guidelines are sufficiently strenuous to meet the legal requirements. Even if your organization is not regulated, you must protect private information or risk losing customer confidence.

The best policy with respect to private data is to minimize its exposure. Applications, processes, and employees should not be granted access to any private data unless the access is required for the tasks that they are to perform. Just as the principle of least privilege dictates that no operation should be performed with more than the necessary privileges, access to private data should be restricted to the smallest possible group.

For mobile applications, make sure they never communicate any sensitive data to other applications running on the device. When private data needs to be stored, it should always be encrypted. For Android, as well as any other platform that uses SQLite database, SQLCipher is a good alternative. SQLCipher is an extension to the SQLite database that provides transparent 256-bit AES encryption of database files. Thus, credentials can be stored in an encrypted database.

Example 3: The following code demonstrates how to integrate SQLCipher into an Android application after downloading the necessary binaries, and store credentials into the database file.

```
import net.sqlcipher.database.SQLiteDatabase;
...
  SQLiteDatabase.loadLibs(this);
  File dbFile = getDatabasePath("credentials.db");
  dbFile.mkdirs();
  dbFile.delete();
  SQLiteDatabase db = SQLiteDatabase.openOrCreateDatabase(dbFile,
"credentials", null);
  db.execSQL("create table credentials(u, p)");
  db.execSQL("insert into credentials(u, p) values(?, ?)", new Object[]
{username, password});
...
```

Note that references to `android.database.sqlite.SQLiteDatabase` are substituted with those of `net.sqlcipher.database.SQLiteDatabase`.

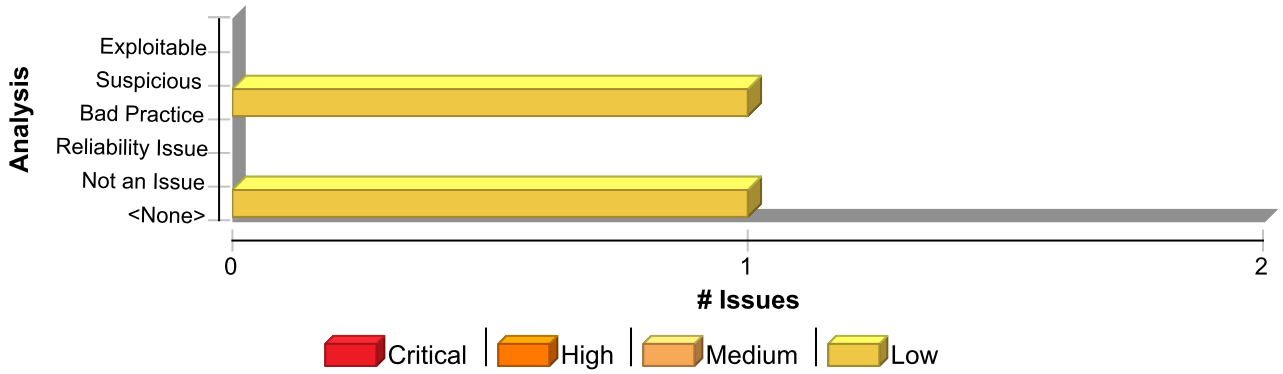
To enable encryption on the WebView store, you must recompile WebKit with the `sqlcipher.so` library.

Example 4: The following code reads username and password for a given site from an Android WebView store and instead of broadcasting them to all the registered receivers, it only broadcasts internally so that the broadcast is only seen by other parts of the same application.

```
...
webview.setWebViewClient(new WebViewClient() {
  public void onReceivedHttpAuthRequest(WebView view,
        HttpAuthHandler handler, String host, String realm) {
    String[] credentials = view.getHttpAuthUsernamePassword(host, realm);
    String username = credentials[0];
    String password = credentials[1];
    Intent i = new Intent();
    i.setAction("SEND_CREDENTIALS");
    i.putExtra("username", username);
```

```
    i.putExtra("password", password);
    LocalBroadcastManager.getInstance(view.getContext()).sendBroadcast(i);
  }
});
...
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Privacy Violation	1	0	0	1
Total	**1**	**0**	**0**	**1**

Privacy Violation — Low

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/UserUtils.java, line 63 (Privacy Violation)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.658

Source Details

Source: Read password
From: com.microfocus.example.utils.UserUtils.writeUser
File: src/main/java/com/microfocus/example/utils/UserUtils.java:63

Privacy Violation — Low

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/UserUtils.java, line 63 (Privacy Violation)

```
60  jGenerator.writeRawValue("\"" + username + "\"");
61
62  jGenerator.writeFieldName("password");
63  jGenerator.writeRawValue("\"" + password + "\"");
64
65  jGenerator.writeFieldName("role");
66  jGenerator.writeRawValue("\"default\"");
```

Sink Details

Sink: com.fasterxml.jackson.core.JsonGenerator.writeRawValue()
Enclosing Method: writeUser()
File: src/main/java/com/microfocus/example/utils/UserUtils.java:63
Taint Flags: PRIVATE

```
60  jGenerator.writeRawValue("\"" + username + "\"");
61
62  jGenerator.writeFieldName("password");
63  jGenerator.writeRawValue("\"" + password + "\"");
64
65  jGenerator.writeFieldName("role");
66  jGenerator.writeRawValue("\"default\"");
```

Race Condition: Singleton Member Field (1 issue)

Abstract

Servlet member fields might allow one user to see another user's data.

Explanation

Many Servlet developers do not understand that a Servlet is a singleton. There is only one instance of the Servlet, and that single instance is used and re-used to handle multiple requests that are processed simultaneously by different threads.

A common result of this misunderstanding is that developers use Servlet member fields in such a way that one user may inadvertently see another user's data. In other words, storing user data in Servlet member fields introduces a data access race condition.

Example 1: The following Servlet stores the value of a request parameter in a member field and then later echoes the parameter value to the response output stream.

```
public class GuestBook extends HttpServlet {

   String name;

   protected void doPost (HttpServletRequest req, HttpServletResponse res) {
     name = req.getParameter("name");
     ...
     out.println(name + ", thanks for visiting!");
   }
}
```

While this code will work perfectly in a single-user environment, if two users access the Servlet at approximately the same time, it is possible for the two request handler threads to interleave in the following way:

Thread 1: assign "`Dick`" to `name` Thread 2: assign "`Jane`" to `name` Thread 1: print "`Jane, thanks for visiting!`" Thread 2: print "`Jane, thanks for visiting!`"

Thereby showing the first user the second user's name.

Recommendation

Do not use Servlet member fields for anything but constants. (i.e. make all member fields `static final`).

Developers are often tempted to use Servlet member fields for user data when they need to transport data from one region of code to another. If this is your aim, consider declaring a separate class and using the Servlet only to "wrap" this new class.

Example 2: The bug in `Example 1` can be corrected in the following way:

```
public class GuestBook extends HttpServlet {

   protected void doPost (HttpServletRequest req, HttpServletResponse res) {
        GBRequestHandler handler = new GBRequestHandler();
        handler.handle(req, res);
   }
}

public class GBRequestHandler {

   String name;

   public void handle(HttpServletRequest req, HttpServletResponse res) {
     name = req.getParameter("name");
     ...
     out.println(name + ", thanks for visiting!");
   }

}
```

Alternatively, a Servlet can utilize synchronized blocks to access servlet instance variables but using synchronized blocks may cause significant performance problems.

Please notice that wrapping the field access within a synchronized block will only prevent the issue if all read and write operations on that member are performed within the same synchronized block or method.

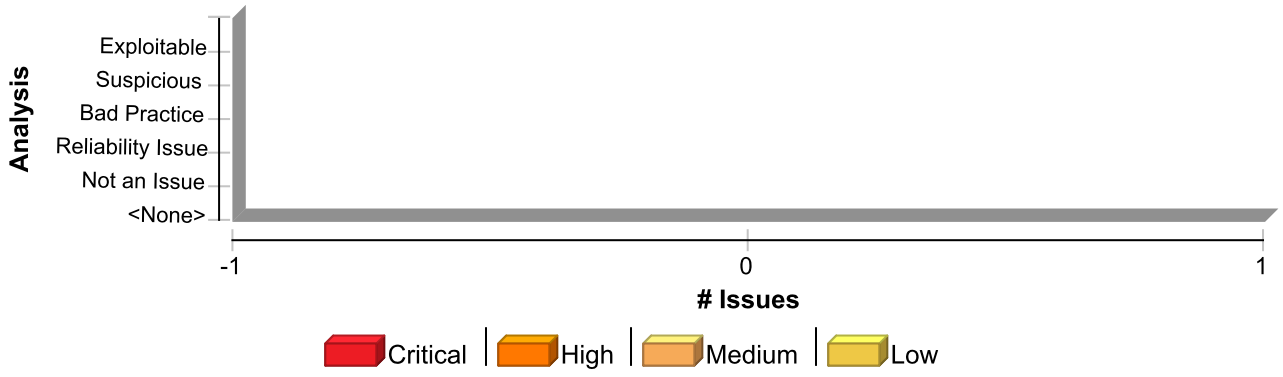
Example 3: Wrapping the `Example 1` write operation (assignment) in a synchronized block will not fix the problem since the threads will have to get a lock to modify `name` field, but they will release the lock afterwards, allowing a second thread to change the value again. If, after changing the `name` value, the first thread resumes execution, the value printed will be the one assigned by the second thread:

```
public class GuestBook extends HttpServlet {

   String name;

   protected void doPost (HttpServletRequest req, HttpServletResponse res) {
     synchronized(name) {
        name = req.getParameter("name");
     }
     ...
     out.println(name + ", thanks for visiting!");
   }
}
```

In order to fix the race condition, all the write and read operations on the shared member field should be run atomically within the same synchronized block:

```
public class GuestBook extends HttpServlet {

   String name;

   protected void doPost (HttpServletRequest req, HttpServletResponse res) {
     synchronized(name) {
        name = req.getParameter("name");
        ...
        out.println(name + ", thanks for visiting!");
     }
   }
}
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Race Condition: Singleton Member Field	1	0	0	1
Total	**1**	**0**	**0**	**1**

Race Condition: Singleton Member Field — High

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java, line 165 (Race Condition: Singleton Member Field)

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction	Not Predicted

Sink Details

Sink: AssignmentStatement
Enclosing Method: executeCommandShell()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java:165

Race Condition: Singleton Member Field — High

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java, line 165 (Race Condition: Singleton Member Field)

```
162  public String executeCommandShell(@RequestParam("cmdshell") String cmd,
163  RedirectAttributes redirectAttributes) {
164
165  this.thRCECMD = cmd;
166  redirectAttributes.addFlashAttribute("message",
167  "You successfully executed " + cmd + "!");
168  return "redirect:/admin/command-shell";
```

Resource Injection (2 issues)

Abstract

Allowing user input to control resource identifiers could enable an attacker to access or modify otherwise protected system resources.

Explanation

A resource injection issue occurs when the following two conditions are met:

1. An attacker is able to specify the identifier used to access a system resource.

For example, an attacker may be able to specify a port number to be used to connect to a network resource.

2. By specifying the resource, the attacker gains a capability that would not otherwise be permitted.

For example, the program may give the attacker the ability to transmit sensitive information to a third-party server.

Note: Resource injections involving resources stored on the file system are reported in a separate category named path manipulation. See the path manipulation description for further details of this vulnerability.

Example 1: The following code uses a port number read from an HTTP request to create a socket.

```
String remotePort = request.getParameter("remotePort");
...
ServerSocket srvr = new ServerSocket(remotePort);
Socket skt = srvr.accept();
...
```

Some think that in the mobile world, classic web application vulnerabilities, such as resource injection, do not make sense -- why would the user attack themself? However, keep in mind that the essence of mobile platforms is applications that are downloaded from various sources and run alongside each other on the same device. The likelihood of running a piece of malware next to a banking application is high, which necessitates expanding the attack surface of mobile applications to include inter-process communication.

Example 2: The following code uses a URL read from an Android intent to load the page in `WebView`.

```
...
        WebView webview = new WebView(this);
        setContentView(webview);
        String url = this.getIntent().getExtras().getString("url");
        webview.loadUrl(url);
...
```

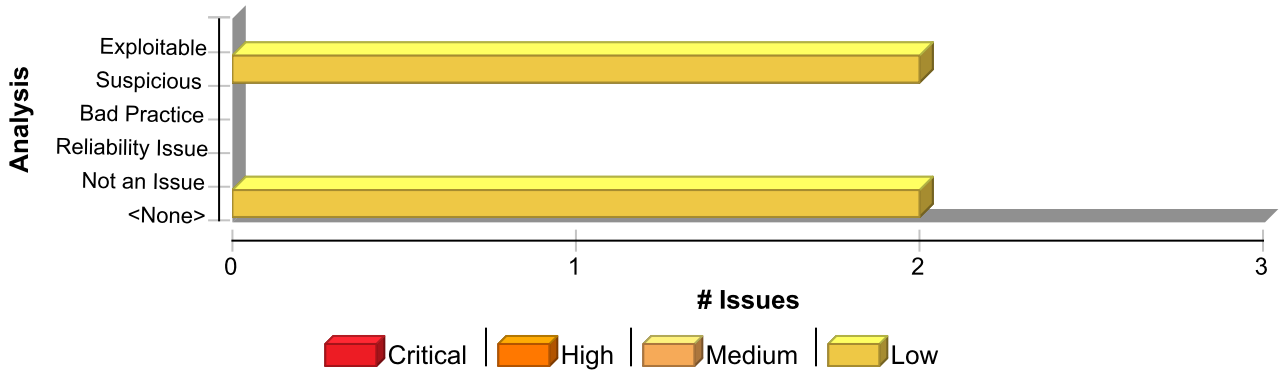
The kind of resource affected by user input indicates the kind of content that may be dangerous. For example, data containing special characters like period, slash, and backslash are risky when used in methods that interact with the file system. Similarly, data that contains URLs and URIs is risky for functions that create remote connections.

Recommendation

The best way to prevent resource injection is with a level of indirection: create a list of legitimate resource names that a user is allowed to specify, and only allow the user to select from the list. With this approach the input provided by the user is never used directly to specify the resource name.

In some situations this approach is impractical because the set of legitimate resource names is too large or too hard to maintain. Programmers often resort to implementing a deny list in these situations. A deny list is used to selectively reject or escape potentially dangerous characters before using the input. However, any such list of unsafe characters is likely to be incomplete and will almost certainly become out of date. A better approach is to create a list of characters that are permitted to appear in the resource name and accept input composed exclusively of characters in the approved set.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Resource Injection	2	0	0	2
Total	**2**	**0**	**0**	**2**

Resource Injection — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/CustomAuthenticationSuccessHandler.java, line 117 (Resource Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.69

Source Details

Source: javax.servlet.http.HttpServletRequest.getHeader()
From: com.microfocus.example.web.controllers.DefaultController.login
File: src/main/java/com/microfocus/example/web/controllers/DefaultController.java:99

Resource Injection — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/CustomAuthenticationSuccessHandler.java, line 117 (Resource Injection)

```
96  @GetMapping("/login")
97  public String login(HttpServletRequest request, Model model, Principal
principal) {
98  HttpSession session = request.getSession(false);
99  String referer = (String) request.getHeader("referer");
100  session.setAttribute("loginReferer", referer);
101  this.setModelDefaults(model, principal, "login");
102  return "login";
```

Sink Details

Sink: java.net.URL.URL()
Enclosing Method: getTargetUrl()
File: src/main/java/com/microfocus/example/config/handlers/CustomAuthenticationSuccessHandler.java:117
Taint Flags: WEB, XSS

```
114  targetUrl = loginReferer;
115  String targetPath = null;
116  try {
117  targetPath = new URL(targetUrl).getPath();
118  } catch (MalformedURLException ex) {
119  log.error(ex.getLocalizedMessage());
120  }
```

src/main/java/com/microfocus/example/config/handlers/UrlAuthenticationSuccessHandler.java, line 84 (Resource Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.52

Source Details

Source: javax.servlet.ServletRequest.getParameter()
From: com.microfocus.example.config.handlers.UrlAuthenticationSuccessHandler.handle
File: src/main/java/com/microfocus/example/config/handlers/UrlAuthenticationSuccessHandler.java:82

Resource Injection — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/UrlAuthenticationSuccessHandler.java, line 84 (Resource Injection)

```
79
80  boolean isUser = false;
81  boolean isAdmin = false;
82  String targetUrl = request.getParameter("referer");
83  //if (targetUrl.endsWith("/")) targetUrl = targetUrl.substring(0,
targetUrl.length());
84  String targetPath = new URL(targetUrl).getPath();
85
```

Sink Details

Sink: java.net.URL.URL()
Enclosing Method: handle()
File: src/main/java/com/microfocus/example/config/handlers/UrlAuthenticationSuccessHandler.java:84
Taint Flags: WEB, XSS

```
81  boolean isAdmin = false;
82  String targetUrl = request.getParameter("referer");
83  //if (targetUrl.endsWith("/")) targetUrl = targetUrl.substring(0, targetUrl.length());
84  String targetPath = new URL(targetUrl).getPath();
85
86  Collection<? extends GrantedAuthority> authorities = authentication.getAuthorities();
87  for (GrantedAuthority grantedAuthority : authorities) {
```

SQL Injection (14 issues)

Abstract

Constructing a dynamic SQL statement with input that comes from an untrusted source could allow an attacker to modify the statement's meaning or to execute arbitrary SQL commands.

Explanation

SQL injection errors occur when:

1. Data enters a program from an untrusted source.

In this case, Fortify Static Code Analyzer could not determine that the source of the data is trusted.

2. The data is used to dynamically construct a SQL query.

Example 1: The following code dynamically constructs and executes a SQL query that searches for items matching a specified name. The query restricts the items displayed to those where the owner matches the user name of the currently-authenticated user.

```
...
        String userName = ctx.getAuthenticatedUserName();
        String itemName = request.getParameter("itemName");
        String query = "SELECT * FROM items WHERE owner = '"
                               + userName + "' AND itemname = '"
                               + itemName + "'";
        ResultSet rs = stmt.execute(query);
...
```

The query intends to execute the following code:

```
        SELECT * FROM items
        WHERE owner = <userName>
        AND itemname = <itemName>;
```

However, because the query is constructed dynamically by concatenating a constant base query string and a user input string, the query only behaves correctly if `itemName` does not contain a single-quote character. If an attacker with the user name `wiley` enters the string "`name' OR 'a'='a`" for `itemName`, then the query becomes the following:

```
        SELECT * FROM items
        WHERE owner = 'wiley'
        AND itemname = 'name' OR 'a'='a';
```

The addition of the `OR 'a'='a'` condition causes the where clause to always evaluate to true, so the query becomes logically equivalent to the much simpler query:

```
        SELECT * FROM items;
```

This simplification of the query allows the attacker to bypass the requirement that the query must only return items owned by the authenticated user. The query now returns all entries stored in the `items` table, regardless of their specified owner.

Example 2: This example examines the effects of a different malicious value passed to the query constructed and executed in `Example 1`. If an attacker with the user name `wiley` enters the string "`name'; DELETE FROM items; --`" for `itemName`, then the query becomes the following two queries:

```
        SELECT * FROM items
        WHERE owner = 'wiley'
        AND itemname = 'name';
```

```
  DELETE FROM items;

      --'
```

Many database servers, including Microsoft(R) SQL Server 2000, allow multiple SQL statements separated by semicolons to be executed at once. While this attack string results in an error on Oracle and other database servers that do not allow the batch-execution of statements separated by semicolons, on databases that do allow batch execution, this type of attack allows the attacker to execute arbitrary commands against the database.

Notice the trailing pair of hyphens (--), which specifies to most database servers that the remainder of the statement is to be treated as a comment and not executed [4]. In this case the comment character serves to remove the trailing single-quote left over from the modified query. On a database where comments are not allowed to be used in this way, the general attack could still be made effective using a trick similar to the one used in `Example 1`. If an attacker enters the string "`name'); DELETE FROM items; SELECT * FROM items WHERE 'a'='a`", the following three valid statements will be created:

```
        SELECT * FROM items
        WHERE owner = 'wiley'
        AND itemname = 'name';

  DELETE FROM items;

      SELECT * FROM items WHERE 'a'='a';
```

Some think that in the mobile world, classic web application vulnerabilities, such as SQL injection, do not make sense -- why would the user attack themself? However, keep in mind that the essence of mobile platforms is applications that are downloaded from various sources and run alongside each other on the same device. The likelihood of running a piece of malware next to a banking application is high, which necessitates expanding the attack surface of mobile applications to include inter-process communication.

Example 3: The following code adapts `Example 1` to the Android platform.

```
...
        PasswordAuthentication pa = authenticator.getPasswordAuthentication();
        String userName = pa.getUserName();
        String itemName = this.getIntent().getExtras().getString("itemName");
        String query = "SELECT * FROM items WHERE owner = '"
                               + userName + "' AND itemname = '"
                               + itemName + "'";
        SQLiteDatabase db = this.openOrCreateDatabase("DB", MODE_PRIVATE,
null);
        Cursor c = db.rawQuery(query, null);
...
```

One traditional approach to preventing SQL injection attacks is to handle them as an input validation problem and either accept only characters from an allow list of safe values or identify and escape a list of potentially malicious values (deny list). Checking an allow list can be a very effective means of enforcing strict input validation rules, but parameterized SQL statements require less maintenance and can offer more guarantees with respect to security. As is almost always the case, implementing a deny list is riddled with loopholes that make it ineffective at preventing SQL injection attacks. For example, attackers may:

- Target fields that are not quoted - Find ways to bypass the need for certain escaped metacharacters - Use stored procedures to hide the injected metacharacters

Manually escaping characters in input to SQL queries can help, but it will not make your application secure from SQL injection attacks.

Another solution commonly proposed for dealing with SQL injection attacks is to use stored procedures. Although stored procedures prevent some types of SQL injection attacks, they fail to protect against many others. Stored procedures typically help prevent SQL injection attacks by limiting the types of statements that can be passed to their parameters. However, there are many ways around the limitations and many interesting statements that can still be passed to stored procedures. Again, stored procedures can prevent some exploits, but they will not make your application secure against SQL injection attacks.

Recommendation

The root cause of a SQL injection vulnerability is the ability of an attacker to change context in the SQL query, causing a value that the programmer intended to be interpreted as data to be interpreted as a command instead. When a SQL query is constructed, the programmer knows what should be interpreted as part of the command and what should be interpreted as data. Parameterized SQL statements can enforce this behavior by disallowing data-directed context changes and preventing nearly all SQL injection attacks. Parameterized SQL statements are constructed using strings of regular SQL, but when user-supplied data needs to be included, they create bind parameters, which are placeholders for data that is subsequently inserted. Bind parameters allow the program to explicitly specify to the database what should be treated as a command and what should be treated as data. When the program is ready to execute a statement, it specifies to the database the runtime values to use for the value of each of the bind parameters, without the risk of the data being interpreted as commands.

Example 1 can be rewritten to use parameterized SQL statements (instead of concatenating user supplied strings) as follows:

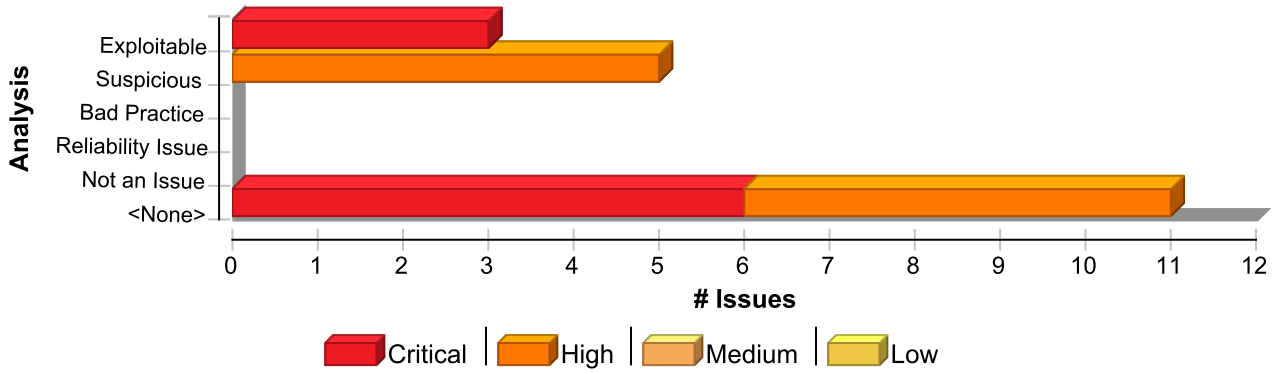
```
...
        String userName = ctx.getAuthenticatedUserName();
        String itemName = request.getParameter("itemName");
        String query =
                "SELECT * FROM items WHERE itemname=? AND owner=?";
        PreparedStatement stmt = conn.prepareStatement(query);
        stmt.setString(1, itemName);
        stmt.setString(2, userName);
        ResultSet results = stmt.execute();
...
```

And here is an Android equivalent:

```
...
        PasswordAuthentication pa = authenticator.getPasswordAuthentication();
        String userName = pa.getUserName();
        String itemName = this.getIntent().getExtras().getString("itemName");
        String query = "SELECT * FROM items WHERE itemname=? AND owner=?";
        SQLiteDatabase db = this.openOrCreateDatabase("DB", MODE_PRIVATE,
null);
        Cursor c = db.rawQuery(query, new Object[]{itemName, userName});
...
```

More complicated scenarios, often found in report generation code, require that user input affect the command structure of the SQL statement, such as the addition of dynamic constraints in the `WHERE` clause. Do not use this requirement to justify concatenating user input into query strings. Prevent SQL injection attacks where user input must affect statement command structure with a level of indirection: create a set of legitimate strings that correspond to different elements you might include in a SQL statement. When constructing a statement, use input from the user to select from this set of application-controlled values.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
SQL Injection	14	0	0	14
Total	**14**	**0**	**0**	**14**

SQL Injection — Critical

URL: null

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 128 (SQL Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

JiraBugLink	
Analysis	Exploitable
AA_Prediction	Exploitable
AA_Confidence	0.827

Source Details

Source: firstaid(1)
From: com.microfocus.example.web.controllers.ProductController.firstaid
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:94
URL: null

```
91  }
92
93  @GetMapping("/firstaid")
94  public String firstaid(Model model, @Param("keywords") String keywords, @Param("limit") Integer limit, Principal principal) {
95  log.debug("Searching for products using keywords: " + ((keywords == null || keywords.isEmpty()) ? "none" : keywords));
96  productService.setPageSize((limit == null ? defaultPageSize : limit));
97  List<Product> products = productService.getAllActiveProducts(0, keywords);
```

Sink Details

SQL Injection — Critical

URL: null

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 128 (SQL Injection)

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findAvailableByKeywords()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:128
Taint Flags: WEB, XSS

```
125  " OR lower(description) LIKE '%" + query + "%'" +
126  " AND available = true " +
127  " LIMIT " + limit + " OFFSET " + offset;
128  return jdbcTemplate.query(sqlQuery, new ProductMapper());
129  }
130
131  public List<Product> findAvailableByKeywordsFromProductName(String keywords) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 128 (SQL Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

JiraBugLink	
Analysis	Exploitable
AA_Prediction	Exploitable
AA_Confidence	0.827

Source Details

Source: index(1)
From: com.microfocus.example.web.controllers.ProductController.index
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:107
URL: null

```
104  }
105
106  @GetMapping(value = {"", "/"})
107  public String index(Model model, @Param("keywords") String keywords, @Param("limit") Integer limit, Principal principal) {
108  log.debug("Searching for products using keywords: " + ((keywords == null || keywords.isEmpty()) ? "none" : keywords));
109  productService.setPageSize((limit == null ? defaultPageSize : limit));
110  List<Product> products = productService.getAllActiveProducts(0, keywords);
```

Sink Details

SQL Injection — Critical

URL: null

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 128 (SQL Injection)

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findAvailableByKeywords()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:128
Taint Flags: WEB, XSS

```
125  " OR lower(description) LIKE '%" + query + "%'" +
126  " AND available = true " +
127  " LIMIT " + limit + " OFFSET " + offset;
128  return jdbcTemplate.query(sqlQuery, new ProductMapper());
129  }
130
131  public List<Product> findAvailableByKeywordsFromProductName(String keywords) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 110 (SQL Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

JiraBugLink	
Analysis	Exploitable
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.787

Source Details

Source: getProductsByKeywords(0)
From: com.microfocus.example.api.controllers.ApiProductController.getProductsByKeywords
File: src/main/java/com/microfocus/example/api/controllers/ApiProductController.java:75
URL: null

```
72  })
73  @GetMapping(value = {""}, produces = {"application/json"})
74  public ResponseEntity<List<ProductResponse>> getProductsByKeywords(
75  @Parameter(description = "Keyword(s) search for products to be found.") @RequestParam("keywords") Optional<String> keywords,
76  @Parameter(description = "Offset of the starting record. 0 indicates the first record.") @RequestParam("offset") Optional<Integer> offset,
77  @Parameter(description = "Maximum records to return. The maximum value allowed is 50.") @RequestParam("limit") Optional<Integer> limit) {
78  log.debug("API::Retrieving products by keyword(s)");
```

Sink Details

SQL Injection — Critical

URL: null

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 110 (SQL Injection)

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findByKeywords()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:110
Taint Flags: WEB, XSS

```
107  " OR lower(summary) LIKE '%" + query + "%'" +
108  " OR lower(description) LIKE '%" + query + "%'" +
109  " LIMIT " + limit + " OFFSET " + offset;
110  return jdbcTemplate.query(sqlQuery, new ProductMapper());
111  }
112
113  public List<Product> findByKeywordsFromProductName(String keywords) {
```

SQL Injection — High

Package: com.microfocus.example.repository

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 55 (SQL Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.674

Source Details

Source: Read this.pageSize
From: com.microfocus.example.service.ProductService.getAllProducts
File: src/main/java/com/microfocus/example/service/ProductService.java:100

```
97   if (keywords != null && !keywords.isEmpty()) {
98   return productRepository.findByKeywords(keywords, offset, pageSize);
99   } else {
100   return productRepository.findAll(offset, pageSize);
101   }
102   }
103
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findAll()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:55
Taint Flags: ARGS, ENVIRONMENT, PROPERTY

SQL Injection — High

Package: com.microfocus.example.repository

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 55 (SQL Injection)

```
52  public List<Product> findAll(int offset, int limit) {
53  String sqlQuery = "select * from products" +
54  " LIMIT " + limit + " OFFSET " + offset;
55  return jdbcTemplate.query(sqlQuery, new ProductMapper());
56  }
57
58  public List<Product> findAvailable(int offset, int limit) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 110 (SQL Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.674

Source Details

Source: Read this.pageSize
From: com.microfocus.example.service.ProductService.getAllProducts
File: src/main/java/com/microfocus/example/service/ProductService.java:98

```
95
96  public List<Product> getAllProducts(Integer offset, String keywords) {
97  if (keywords != null && !keywords.isEmpty()) {
98  return productRepository.findByKeywords(keywords, offset, pageSize);
99  } else {
100  return productRepository.findAll(offset, pageSize);
101  }
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findByKeywords()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:110
Taint Flags: ARGS, ENVIRONMENT, PROPERTY

SQL Injection — High

Package: com.microfocus.example.repository

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 110 (SQL Injection)

```
107  " OR lower(summary) LIKE '%" + query + "%'" +
108  " OR lower(description) LIKE '%" + query + "%'" +
109  " LIMIT " + limit + " OFFSET " + offset;
110  return jdbcTemplate.query(sqlQuery, new ProductMapper());
111  }
112
113  public List<Product> findByKeywordsFromProductName(String keywords) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 128 (SQL Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.674

Source Details

Source: Read this.pageSize
From: com.microfocus.example.service.ProductService.getAllActiveProducts
File: src/main/java/com/microfocus/example/service/ProductService.java:106

```
103
104  public List<Product> getAllActiveProducts(Integer offset, String
keywords) {
105  if (keywords != null && !keywords.isEmpty()) {
106  return productRepository.findAvailableByKeywords(keywords, offset,
pageSize);
107  }
108  return productRepository.findAvailable(offset, pageSize);
109  }
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findAvailableByKeywords()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:128
Taint Flags: ARGS, ENVIRONMENT, PROPERTY

SQL Injection — High

Package: com.microfocus.example.repository

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 128 (SQL Injection)

```
125  " OR lower(description) LIKE '%" + query + "%'" +
126  " AND available = true " +
127  " LIMIT " + limit + " OFFSET " + offset;
128  return jdbcTemplate.query(sqlQuery, new ProductMapper());
129  }
130
131  public List<Product> findAvailableByKeywordsFromProductName(String keywords) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 62 (SQL Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.674

Source Details

Source: Read this.pageSize
From: com.microfocus.example.service.ProductService.getAllActiveProducts
File: src/main/java/com/microfocus/example/service/ProductService.java:108

```
105  if (keywords != null && !keywords.isEmpty()) {
106  return productRepository.findAvailableByKeywords(keywords, offset,
pageSize);
107  }
108  return productRepository.findAvailable(offset, pageSize);
109  }
110
111  public long count() {
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findAvailable()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:62
Taint Flags: ARGS, ENVIRONMENT, PROPERTY

SQL Injection — High

Package: com.microfocus.example.repository

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 62 (SQL Injection)

```
59  String sqlQuery = "select * from products" +
60  " where available = true " +
61  " LIMIT " + limit + " OFFSET " + offset;
62  return jdbcTemplate.query(sqlQuery, new ProductMapper());
63  }
64
65  public Optional<Product> findById(UUID id) {
```

URL: null

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 95 (SQL Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.565

Source Details

Source: productSave(0)
From: com.microfocus.example.web.controllers.admin.AdminProductController.productSave
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminProductController.java:102
URL: null

```
99   }
100
101  @PostMapping("/{id}/save")
102  public String productSave(@Valid @ModelAttribute("adminProductForm")
AdminProductForm adminProductForm,
103  BindingResult bindingResult, Model model,
104  RedirectAttributes redirectAttributes,
105  Principal principal) {
```

Sink Details

Sink: org.springframework.jdbc.core.JdbcTemplate.query()
Enclosing Method: findByCode()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:95
Taint Flags: WEB, XSS

SQL Injection — High

URL: null

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 95 (SQL Injection)

```
92  String query = code.toLowerCase();
93  String sqlQuery = "SELECT * FROM " + getTableName() +
94  " WHERE lower(code) = '" + query + "'";
95  result = jdbcTemplate.query(sqlQuery, new ProductMapper());
96  Optional<Product> optionalProduct = Optional.empty();
97  if (!result.isEmpty()) {
98  optionalProduct = Optional.of(result.get(0));
```

SQL Injection — Low

Package: com.microfocus.example.repository

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 70 (SQL Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Audit Details

AA_Prediction	Not Predicted

Sink Details

Sink: query()
Enclosing Method: findById()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:70

```
67  String query = id.toString();
68  String sqlQuery = "SELECT * FROM " + getTableName() +
69  " WHERE id = '" + query + "'";
70  result = jdbcTemplate.query(sqlQuery, new ProductMapper());
71  Optional<Product> optionalProduct = Optional.empty();
72  if (!result.isEmpty()) {
73  optionalProduct = Optional.of(result.get(0));
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 135 (SQL Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Audit Details

AA_Prediction	Not Predicted

Sink Details

SQL Injection — Low

Package: com.microfocus.example.repository

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 135 (SQL Injection)

Sink: query()
Enclosing Method: findAvailableByKeywordsFromProductName()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:135

```
132  String query = keywords.toLowerCase();
133  String sqlQuery = "SELECT * FROM " + getTableName() +
134  " WHERE available = true AND lower(name) LIKE '%" + query + "%' ";
135  return jdbcTemplate.query(sqlQuery, new ProductMapper());
136  }
137
138  public Product save(Product p) {
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 83 (SQL Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Audit Details

AA_Prediction	Not Predicted

Sink Details

Sink: query()
Enclosing Method: existsById()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:83

```
80  String query = id.toString().toLowerCase();
81  String sqlQuery = "SELECT * FROM " + getTableName() +
82  " WHERE id = '" + query + "'";
83  result = jdbcTemplate.query(sqlQuery, new ProductMapper());
84  if (result.isEmpty()) {
85  return false;
86  }
```

src/main/java/com/microfocus/example/repository/UserRepositoryImpl.java, line 77 (SQL Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Audit Details

AA_Prediction	Not Predicted

Sink Details

SQL Injection — Low

Package: com.microfocus.example.repository

src/main/java/com/microfocus/example/repository/UserRepositoryImpl.java, line 77 (SQL Injection)

Sink: executeQuery()
Enclosing Method: execute()
File: src/main/java/com/microfocus/example/repository/UserRepositoryImpl.java:77

```
74  Integer authorityCount = 0;
75  try {
76  Statement stmt = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);
77  ResultSet results = stmt.executeQuery(
78  "SELECT u.*, a.name as authority " +
79  "FROM users u, authorities a INNER JOIN user_authorities ua on a.id = ua.authority_id " +
80  "WHERE u.id = ua.user_id AND u.username LIKE '" + username + "'");
```

src/main/java/com/microfocus/example/repository/ProductRepository.java, line 117 (SQL Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Audit Details

AA_Prediction	Not Predicted

Sink Details

Sink: query()
Enclosing Method: findByKeywordsFromProductName()
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:117

```
114  String query = keywords.toLowerCase();
115  String sqlQuery = "SELECT * FROM " + getTableName() +
116  " WHERE lower(name) LIKE '%" + query + "%' ";
117  return jdbcTemplate.query(sqlQuery, new ProductMapper());
118  }
119
120  public List<Product> findAvailableByKeywords(String keywords, int offset, int limit) {
```

src/main/java/com/microfocus/example/repository/ReviewRepositoryImpl.java, line 130 (SQL Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Audit Details

AA_Prediction	Not Predicted

Sink Details

SQL Injection — Low

Package: com.microfocus.example.repository

src/main/java/com/microfocus/example/repository/ReviewRepositoryImpl.java, line 130 (SQL Injection)

Sink: createNativeQuery()
Enclosing Method: addProductReview()
File: src/main/java/com/microfocus/example/repository/ReviewRepositoryImpl.java:130

```
127
128  public Review addProductReview(UUID productId, UUID userId, String comment, int rating) {
129  UUID reviewId = UUID.randomUUID();
130  entityManager.createNativeQuery(
131  "INSERT INTO reviews (id, product_id, user_id, comment, rating) " +
132  "VALUES (" +
133  reviewId + "," +
```

Session Puzzling: Spring (14 issues)

Abstract

Attackers may modify Spring session attributes which may lead to application logic abuse.

Explanation

A class annotated with `@SessionAttributes` will mean Spring replicates changes to model attributes in the session object. If an attacker is able to store arbitrary values within a model attribute, these changes will be replicated in the session object where they may be trusted by the application. If the session attribute is initialized with trusted data which the user should not be able to modify, the attacker may be able to conduct a Session Puzzling attack and abuse the application logic.

Example 1: The following controller contains a method which loads the user data into the session upon a successful login.

```
@Controller
@SessionAttributes("user")
public class HomeController {
    ...
    @RequestMapping(value= "/auth", method=RequestMethod.POST)
    public String authHandler(@RequestParam String username, @RequestParam
String password, RedirectAttributes attributes, Model model) {
        User user = userService.findByNamePassword(username, password);
        if (user == null) {
            // Handle error
            ...
        } else {
            // Handle success
            attributes.addFlashAttribute("user", user);
            return "redirect:home";
        }
    }
    ...
}
```

A different controller handles the reset password feature. It tries to load the `User` instance from the session since the class is annotated with `@SessionAttributes("user")` and uses it to verify the reset password question.

```
@Controller
@SessionAttributes("user")
public class ResetPasswordController {

    @RequestMapping(value = "/resetQuestion", method = RequestMethod.POST)
    public String resetQuestionHandler(@RequestParam String answerReset,
SessionStatus status, User user, Model model) {

        if (!user.getAnswer().equals(answerReset)) {
            // Handle error
            ...
        } else {
            // Handle success
            ...
        }
    }
}
```

The developer's intention was to load the `user` instance from the session where it was stored during the login process. However Spring will check the request and will try to bind its data into the model `user` instance. If the received request contains data that can be bound to the `User` class, Spring will merge the received data into the user session attribute. This scenario can be abused by submitting both an arbitrary

answer in the `answerReset` query parameter and the same value to override the value stored in the session. This way, the attacker may set an arbitrary new password for random users.

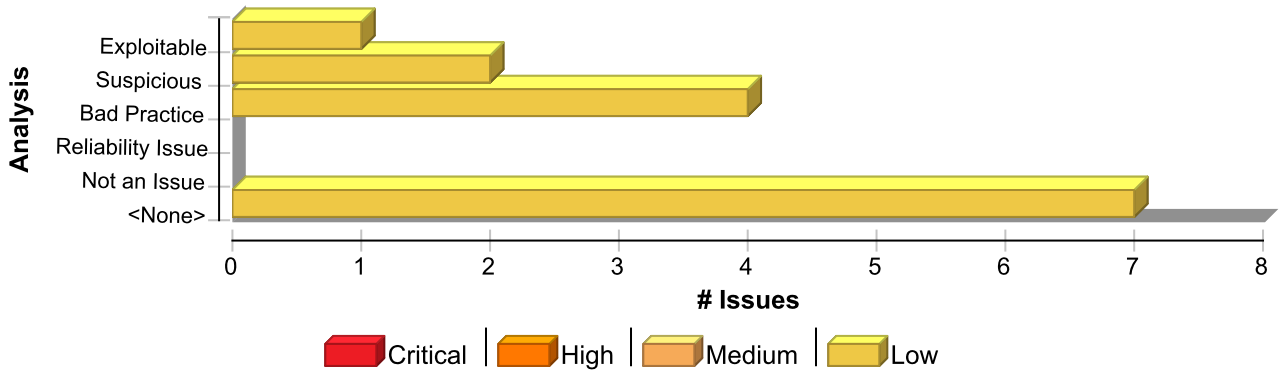
Recommendation

When using `@SessionAttributes` annotation, pay special attention to the session attributes that the user will be able to override. Make sure these attributes are meant to be modified by the user and do not contain any security sensitive data that may get manipulated by an attacker by abusing Spring auto-binding capabilities.

Example 2: The following controller uses `@SessionAttribute` annotation (on the method parameter, not to be confused with `@SessionAttributes`) to load the user instance from the session object without merging any incoming request parameters.

```
@Controller
public class ResetPasswordController {

    @RequestMapping(value = "/resetQuestion", method = RequestMethod.POST)
    public String resetQuestionHandler(@RequestParam String answerReset,
SessionStatus status, @SessionAttribute User user, Model model) {

        if (!user.getAnswer().equals(answerReset)) {
            // Handle error
            ...
        } else {
            // Handle success
            ...
        }
    }
}
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Session Puzzling: Spring	14	0	0	14
Total	**14**	**0**	**0**	**14**

Session Puzzling: Spring — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 565 (Session Puzzling: Spring)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: Variable: uploadForm
Enclosing Method: listUploadedXMLFiles()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:565

```
562  }
563
564  @GetMapping("/upload-xml-file")
565  public String listUploadedXMLFiles(@Valid @ModelAttribute("uploadForm") UploadForm uploadForm,
566  BindingResult bindingResult, Model model,
567  RedirectAttributes redirectAttributes,
568  Principal principal) throws IOException {
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 380 (Session Puzzling: Spring)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: Variable: passwordForm
Enclosing Method: userSavePassword()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:380

```
377  }
378
379  @PostMapping("/savePassword")
380  public String userSavePassword(@Valid @ModelAttribute("passwordForm") PasswordForm passwordForm,
381  BindingResult bindingResult, Model model,
382  RedirectAttributes redirectAttributes,
383  Principal principal) {
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 429 (Session Puzzling: Spring)

Issue Details

Session Puzzling: Spring — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 429 (Session Puzzling: Spring)

Kingdom: Input Validation and Representation
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: Variable: registerUserForm
Enclosing Method: registerUser()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:429

```
426  }
427
428  @PostMapping("/register")
429  public String registerUser(@Valid @ModelAttribute("registerUserForm") RegisterUserForm
registerUserForm,
430  BindingResult bindingResult, Model model,
431  RedirectAttributes redirectAttributes,
432  Principal principal) {
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 696 (Session Puzzling: Spring)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: Function: serveUnverifiedFile
Enclosing Method: serveUnverifiedFile()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:696

```
693  }
694
695  @GetMapping("/files/download/unverified")
696  public ResponseEntity<?> serveUnverifiedFile(@Param("file") String file) {
697
698  if (Objects.isNull(file) || file.isEmpty()) {
699  return ResponseEntity.badRequest().build();
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 531 (Session Puzzling: Spring)

Issue Details

Session Puzzling: Spring — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 531 (Session Puzzling: Spring)

Kingdom: Input Validation and Representation
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: Variable: uploadForm
Enclosing Method: listUploadedFiles()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:531

```
528  //
529
530  @GetMapping("/uploadFile")
531  public String listUploadedFiles(@Valid @ModelAttribute("uploadForm") UploadForm
uploadForm,
532  BindingResult bindingResult, Model model,
533  RedirectAttributes redirectAttributes,
534  Principal principal) throws IOException {
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 685 (Session Puzzling: Spring)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: Function: handleStorageFileNotFound
Enclosing Method: handleStorageFileNotFound()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:685

```
682  }
683
684  @ExceptionHandler(StorageFileNotFoundException.class)
685  public ResponseEntity<?> handleStorageFileNotFound(StorageFileNotFoundException exc) {
686  return ResponseEntity.notFound().build();
687  }
688
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 351 (Session Puzzling: Spring)

Issue Details

Session Puzzling: Spring — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 351 (Session Puzzling: Spring)

Kingdom: Input Validation and Representation
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction	Not Predicted

Sink Details

Sink: Variable: userForm
Enclosing Method: userSaveProfile()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:351

```
348  //
349
350  @PostMapping("/saveProfile")
351  public String userSaveProfile(@Valid @ModelAttribute("userForm") UserForm userForm,
352  BindingResult bindingResult, Model model,
353  RedirectAttributes redirectAttributes,
354  Principal principal) {
```

URL: null

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 559 (Session Puzzling: Spring)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.76

Source Details

Source: handleFileUpload(0)
From: com.microfocus.example.web.controllers.UserController.handleFileUpload
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:554
URL: null

```
551  }
552
553  @PostMapping("/files/upload")
554  public String handleFileUpload(@RequestParam("file") MultipartFile file,
555  RedirectAttributes redirectAttributes) {
556
557  storageService.store(file);
```

Session Puzzling: Spring — Low

URL: null

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 559 (Session Puzzling: Spring)

Sink Details

Sink: org.springframework.web.servlet.mvc.support.RedirectAttributes.addFlashAttribute()
Enclosing Method: handleFileUpload()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:559
Taint Flags: WEB, XSS

```
556
557  storageService.store(file);
558  redirectAttributes.addFlashAttribute("message",
559  "You successfully uploaded " + file.getOriginalFilename() + "!");
560
561  return "redirect:/user/upload-file";
562  }
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 491 (Session Puzzling: Spring)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.509

Source Details

Source: verifyUser(1)
From: com.microfocus.example.web.controllers.UserController.verifyUser
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:478
URL: null

```
475
476  @GetMapping("/verify")
477  public String verifyUser(@RequestParam("email") Optional<String>
usersEmail,
478  @RequestParam("code") Optional<String> verificationCode,
479  @RequestParam("status") Optional<String> statusCode,
480  RedirectAttributes redirectAttributes,
481  Model model) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: verifyUser()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:491
Taint Flags: WEB, XSS

Session Puzzling: Spring — Low

URL: null

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 491 (Session Puzzling: Spring)

```
488  model.addAttribute("message", "Your registration details have been stored. Please check
your email to verify your details.");
489  model.addAttribute("alertClass", "alert-success");
490  VerifyUserForm verifyUserForm = new VerifyUserForm(usersEmail, verificationCode);
491  model.addAttribute("verifyUserForm", verifyUserForm);
492  this.setModelDefaults(model, null, "verify");
493  return "user/verify";
494  } else if ((email == null || email.isEmpty()) || (code == null || code.isEmpty())) {
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 499 (Session Puzzling: Spring)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.509

Source Details

Source: verifyUser(1)
From: com.microfocus.example.web.controllers.UserController.verifyUser
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:478
URL: null

```
475
476  @GetMapping("/verify")
477  public String verifyUser(@RequestParam("email") Optional<String>
usersEmail,
478  @RequestParam("code") Optional<String> verificationCode,
479  @RequestParam("status") Optional<String> statusCode,
480  RedirectAttributes redirectAttributes,
481  Model model) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: verifyUser()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:499
Taint Flags: WEB, XSS

Session Puzzling: Spring — Low

URL: null

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 499 (Session Puzzling: Spring)

```
496  model.addAttribute("message", "You need to supply both an email address and verification code.");
497  model.addAttribute("alertClass", "alert-danger");
498  VerifyUserForm verifyUserForm = new VerifyUserForm(usersEmail, verificationCode);
499  model.addAttribute("verifyUserForm", verifyUserForm);
500  this.setModelDefaults(model, null, "verify");
501  return "user/verify";
502  } else {
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 491 (Session Puzzling: Spring)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.509

Source Details

Source: verifyUser(0)
From: com.microfocus.example.web.controllers.UserController.verifyUser
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:477
URL: null

```
474  }
475
476  @GetMapping("/verify")
477  public String verifyUser(@RequestParam("email") Optional<String> usersEmail,
478  @RequestParam("code") Optional<String> verificationCode,
479  @RequestParam("status") Optional<String> statusCode,
480  RedirectAttributes redirectAttributes,
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: verifyUser()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:491
Taint Flags: WEB, XSS

Session Puzzling: Spring — Low

URL: null

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 491 (Session Puzzling: Spring)

```
488  model.addAttribute("message", "Your registration details have been stored. Please check
your email to verify your details.");
489  model.addAttribute("alertClass", "alert-success");
490  VerifyUserForm verifyUserForm = new VerifyUserForm(usersEmail, verificationCode);
491  model.addAttribute("verifyUserForm", verifyUserForm);
492  this.setModelDefaults(model, null, "verify");
493  return "user/verify";
494  } else if ((email == null || email.isEmpty()) || (code == null || code.isEmpty())) {
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 499 (Session Puzzling: Spring)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.509

Source Details

Source: verifyUser(0)
From: com.microfocus.example.web.controllers.UserController.verifyUser
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:477
URL: null

```
474  }
475
476  @GetMapping("/verify")
477  public String verifyUser(@RequestParam("email") Optional<String>
usersEmail,
478  @RequestParam("code") Optional<String> verificationCode,
479  @RequestParam("status") Optional<String> statusCode,
480  RedirectAttributes redirectAttributes,
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: verifyUser()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:499
Taint Flags: WEB, XSS

Session Puzzling: Spring — Low

URL: null

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 499 (Session Puzzling: Spring)

```
496  model.addAttribute("message", "You need to supply both an email address and verification code.");
497  model.addAttribute("alertClass", "alert-danger");
498  VerifyUserForm verifyUserForm = new VerifyUserForm(usersEmail, verificationCode);
499  model.addAttribute("verifyUserForm", verifyUserForm);
500  this.setModelDefaults(model, null, "verify");
501  return "user/verify";
502  } else {
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 679 (Session Puzzling: Spring)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Exploitable
AA_Prediction	Exploitable
AA_Confidence	0.819

Source Details

Source: handleXMLUpdate(0)
From: com.microfocus.example.web.controllers.UserController.handleXMLUpdate
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:652
URL: null

```
649  }
650
651  @PostMapping("/files/xml/update")
652  public String handleXMLUpdate(@RequestParam("filename") String fileName,
653  @RequestParam("fcontent") String newXMLContent,
654  RedirectAttributes redirectAttributes) {
655
```

Sink Details

Sink: org.springframework.web.servlet.mvc.support.RedirectAttributes.addFlashAttribute()
Enclosing Method: handleXMLUpdate()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:679
Taint Flags: WEB, XSS

Session Puzzling: Spring — Low

URL: null

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 679 (Session Puzzling: Spring)

```
676
677
678  redirectAttributes.addFlashAttribute("message",
679  "Successfully updated " + fileName + "!");
680
681  return "redirect:/user/upload-xml-file";
682  }
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 631 (Session Puzzling: Spring)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.76

Source Details

Source: handleXMLFileUpload(0)
From: com.microfocus.example.web.controllers.UserController.handleXMLFileUpload
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:626
URL: null

```
623  }
624
625  @PostMapping("/files/upload-xml")
626  public String handleXMLFileUpload(@RequestParam("file") MultipartFile
file,
627  RedirectAttributes redirectAttributes) {
628
629  storageService.store(file);
```

Sink Details

Sink: org.springframework.web.servlet.mvc.support.RedirectAttributes.addFlashAttribute()
Enclosing Method: handleXMLFileUpload()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:631
Taint Flags: WEB, XSS

Session Puzzling: Spring — Low

URL: null

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 631 (Session Puzzling: Spring)

```
628
629  storageService.store(file);
630  redirectAttributes.addFlashAttribute("message",
631  "You successfully uploaded " + file.getOriginalFilename() + "!");
632
633  return "redirect:/user/upload-xml-file";
634  }
```

Spring Boot Misconfiguration: DevTools Enabled (1 issue)

Abstract

The Spring Boot application is configured in developer mode.

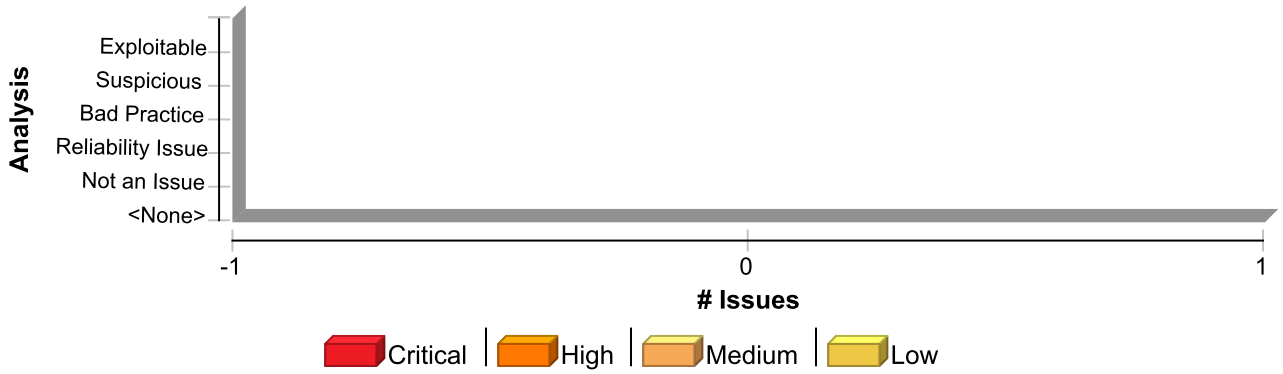
Explanation

The Spring Boot application has DevTools enabled. DevTools include an additional set of tools which can make the application development experience a little more pleasant, but DevTools are not recommended to use on applications in a production environment. As stated in the official Spring Boot documentation: "Enabling `spring-boot-devtools` on a remote application is a security risk. You should never enable support on a production deployment."

Recommendation

Remove `spring-boot-devtools` dependency on production deployments.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Spring Boot Misconfiguration: DevTools Enabled	1	0	0	1
Total	**1**	**0**	**0**	**1**

Spring Boot Misconfiguration: DevTools Enabled — High

Package: <none>

pom.xml, line 285 (Spring Boot Misconfiguration: DevTools Enabled)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Configuration)

Audit Details

AA_Prediction Not Predicted

Sink Details

File: pom.xml:285

Spring Boot Misconfiguration: DevTools Enabled — High

Package: <none>

pom.xml, line 285 (Spring Boot Misconfiguration: DevTools Enabled)

```
282  <dependencies>
283  <dependency>
284  <groupId>org.springframework.boot</groupId>
285  <artifactId>spring-boot-devtools</artifactId>
286  <optional>true</optional>
287  </dependency>
288  </dependencies>
```

Spring Security Misconfiguration: Lack of Fallback Check (1 issue)

Abstract

Spring Security configuration lacks a fallback check to apply to unmatched requests.

Explanation

Spring Security uses an expression-based access control that lets developers define a set of checks that must be applied to every request. To determine if the access control must be applied to the request, Spring Security attempts to match the request with the request matcher defined for every security check. If the request matches, the access control is applied to the request. A special request matcher exists to always match against any requests: `anyRequest()`. Failing to define a fallback check that uses the `anyRequest()` matcher, might leave endpoints unprotected.

Example 1: The following code defines a Spring Security configuration that fails to define a fallback check:

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    ...
    http.authorizeRequests()
        .mvcMatchers("/admin").hasRole("ADMIN");
    ...
}
```

In the previous `Example 1` example, current or future endpoints such as `/admin/panel` might be left unprotected.

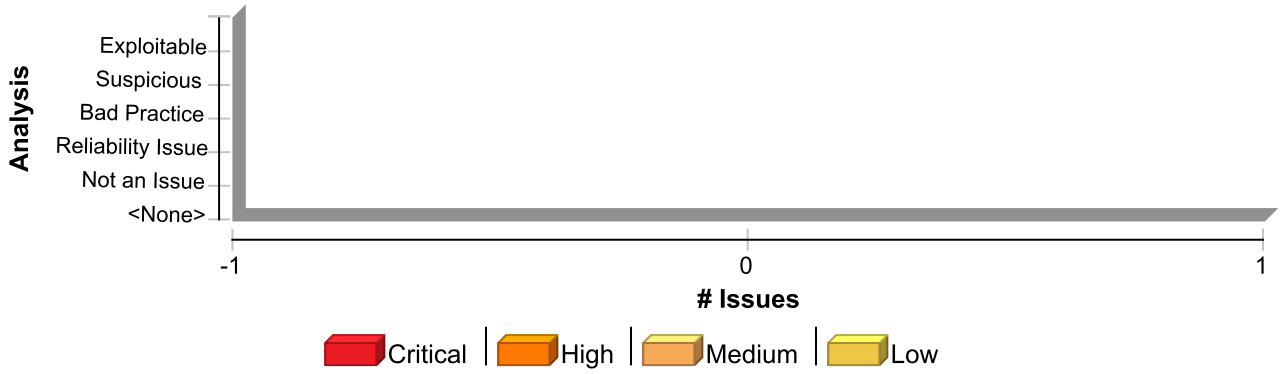
Recommendation

As a security best practice, always include a catch-all matcher that denies access to any previously unmatched requests.

Example 2: The following code defines a Spring Security configuration that defaults to deny access to any unmatched requests:

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    ...
    http.authorizeRequests()
        .mvcMatchers("/admin").hasRole("ADMIN")
        .mvcMatchers("/home").anonymous()
        .anyRequest().denyAll();
    ...
}
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Spring Security Misconfiguration: Lack of Fallback Check	1	0	0	1
Total	**1**	**0**	**0**	**1**

Spring Security Misconfiguration: Lack of Fallback Check — Low

Package: com.microfocus.example.config

src/main/java/com/microfocus/example/config/WebSecurityConfiguration.java, line 99 (Spring Security Misconfiguration: Lack of Fallback Check)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: Function: configure
Enclosing Method: configure()
File: src/main/java/com/microfocus/example/config/WebSecurityConfiguration.java:99

```
96  public class ApiConfigurationAdapter extends WebSecurityConfigurerAdapter {
97
98  @Override
99  protected void configure(HttpSecurity httpSecurity) throws Exception {
100
101  /*http.cors().and().csrf().disable()
102  .exceptionHandling().authenticationEntryPoint(unauthorizedHandler).and()
```

Spring Security Misconfiguration: Overly Permissive Firewall Policy (1 issue)

Abstract

Spring Security HTTP firewall is configured with a lax policy.

Explanation

Spring Security includes an HTTP firewall that helps protect the application by sanitizing requests that contain potentially malicious characters. Spring achieves this by including the `HttpFirewall` into its `FilterChainProxy`, which processes the requests before they are sent through the filter chain. Sprint Security uses the `StrictHttpFirewall` implementation by default.

Example: The following code relaxes the firewall policy to allow `%2F` and `;` characters:

```
@Override
public void configure(WebSecurity web) throws Exception {
    super.configure(web);
    StrictHttpFirewall firewall = new StrictHttpFirewall();
    firewall.setAllowUrlEncodedSlash(true);
    firewall.setAllowSemicolon(true);
    web.httpFirewall(firewall);
}
```

Allowing potentially malicious characters can lead to vulnerabilities if these characters are incorrectly on inconsistently processed. For example, allowing semicolons enables path parameters (as defined in RFC 2396), which are not consistently processed by frontend web servers such as nginx and application servers such as Apache Tomcat. Attackers can use these inconsistencies for path traversal attacks or access control bypasses.

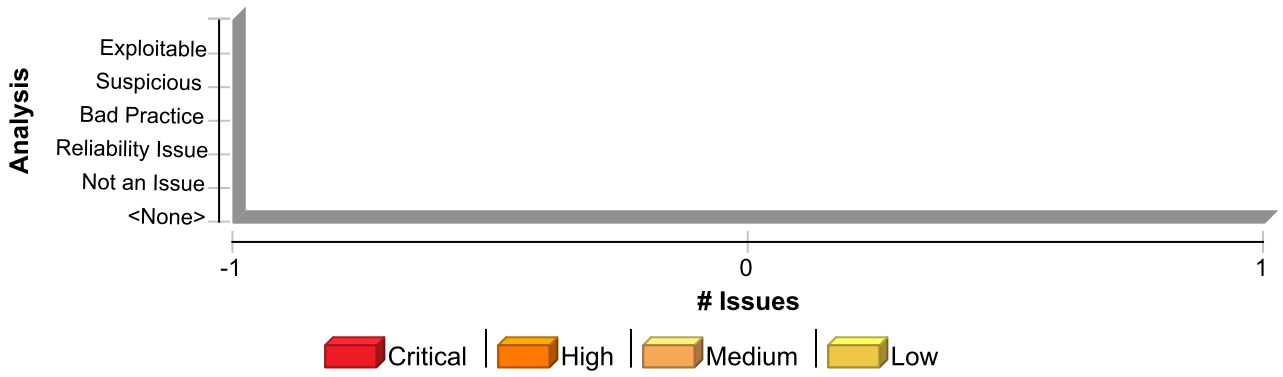
Recommendation

Do not relax the default HTTP firewall policy.

Note that regardless of the class name, `DefaultHttpFirewall` is not the default. The Spring documentation states:

"Users should consider using `StrictHttpFirewall` because rather than trying to sanitize a malicious URL it rejects the malicious URL providing better security guarantees."

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Spring Security Misconfiguration: Overly Permissive Firewall Policy	1	0	0	1
Total	**1**	**0**	**0**	**1**

Spring Security Misconfiguration: Overly Permissive Firewall Policy — Medium

Package: com.microfocus.example.config

src/main/java/com/microfocus/example/config/WebSecurityConfiguration.java, line 231 (Spring Security Misconfiguration: Overly Permissive Firewall Policy)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: FunctionCall: DefaultHttpFirewall
Enclosing Method: getHttpFirewall()
File: src/main/java/com/microfocus/example/config/WebSecurityConfiguration.java:231

```
228
229  @Bean
230  public HttpFirewall getHttpFirewall() {
231  return new DefaultHttpFirewall();
232  }
233
234  }
```

System Information Leak (11 issues)

Abstract

Revealing system data or debugging information helps an adversary learn about the system and form a plan of attack.

Explanation

An information leak occurs when system data or debug information leaves the program through an output stream or logging function.

Example 1: The following code writes an exception to the standard error stream:

```
try {
    ...
} catch (Exception e) {
    e.printStackTrace();
}
```

Depending upon the system configuration, this information can be dumped to a console, written to a log file, or exposed to a remote user. For example, with scripting mechanisms it is trivial to redirect output information from "Standard error" or "Standard output" into a file or another program. Alternatively, the system that the program runs on could have a remote logging mechanism such as a "syslog" server that sends the logs to a remote device. During development, you have no way of knowing where this information might end up being displayed.

In some cases, the error message provides the attacker with the precise type of attack to which the system is vulnerable. For example, a database error message can reveal that the application is vulnerable to a SQL injection attack. Other error messages can reveal more oblique clues about the system. In `Example 1`, the leaked information could imply information about the type of operating system, the applications installed on the system, and the amount of care that the administrators have put into configuring the program.

Here is another scenario, specific to the mobile world. Most mobile devices now implement a Near-Field Communication (NFC) protocol for quickly sharing information between devices using radio communication. It works by bringing devices to close proximity or simply having them touch each other. Even though the communication range of NFC is limited to just a few centimeters, eavesdropping, data modification and various other types of attacks are possible, since NFC alone does not ensure secure communication.

Example 2: The Android platform provides support for NFC. The following code creates a message that gets pushed to the other device within the range.

```
...
public static final String TAG = "NfcActivity";
private static final String DATA_SPLITTER = "__:DATA:__";
private static final String MIME_TYPE = "application/my.applications.mimetype";
...
public NdefMessage createNdefMessage(NfcEvent event) {
    TelephonyManager tm =
(TelephonyManager)Context.getSystemService(Context.TELEPHONY_SERVICE);
    String VERSION = tm.getDeviceSoftwareVersion();
    String text = TAG + DATA_SPLITTER + VERSION;
    NdefRecord record = new NdefRecord(NdefRecord.TNF_MIME_MEDIA,
            MIME_TYPE.getBytes(), new byte[0], text.getBytes());
    NdefRecord[] records = { record };
    NdefMessage msg = new NdefMessage(records);
    return msg;
}
...
```

NFC Data Exchange Format (NDEF) message contains typed data, a URI, or a custom application payload. If the message contains information about the application, such as its name, MIME type, or device

software version, this information could be leaked to an eavesdropper. In `Example 2`, Fortify Static Code Analyzer reports a System Information Leak vulnerability on the return statement.

Recommendation

Write error messages with security in mind. In production environments, turn off detailed error information in favor of brief messages. Restrict the generation and storage of detailed output that can help administrators and programmers diagnose problems. Debug traces can sometimes appear in non-obvious places (embedded in comments in the HTML for an error page, for example).

Even brief error messages that do not reveal stack traces or database dumps can potentially aid an attacker. For example, an "Access Denied" message can reveal that a file or user exists on the system.

If you are concerned about leaking system data via NFC on an Android device, you could do one of the following three things. Do not include system data in the messages pushed to other devices in range, encrypt the payload of the message, or establish a secure communication channel at a higher layer.

Issue Summary

Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
System Information Leak	11	0	0	11
Total	**11**	**0**	**0**	**11**

System Information Leak — Low

Package: com.microfocus.example.service

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 43 (System Information Leak)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Semantic)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: printStackTrace()
Enclosing Method: FileSystemStorageService()
File: src/main/java/com/microfocus/example/service/FileSystemStorageService.java:43

System Information Leak — Low

Package: com.microfocus.example.service

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 43 (System Information Leak)

```
40  try {
41  Files.createDirectory(rootLocation);
42  } catch (IOException e) {
43  e.printStackTrace();
44  }
45  }
46  }
```

src/main/java/com/microfocus/example/service/UserService.java, line 188 (System Information Leak)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Semantic)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: printStackTrace()
Enclosing Method: subscribeUser()
File: src/main/java/com/microfocus/example/service/UserService.java:188

```
185  try {
186  UserUtils.registerUser(null, null, newUser.getEmail());
187  } catch (IOException | ParseException e) {
188  e.printStackTrace();
189  }
190  SubscribeUserResponse subscribedUser = new SubscribeUserResponse(newUser.getId(), null,
null, newUser.getEmail());
191  return subscribedUser;
```

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 105 (System Information Leak)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Semantic)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: printStackTrace()
Enclosing Method: checkMimeType()
File: src/main/java/com/microfocus/example/service/FileSystemStorageService.java:105

System Information Leak — Low

Package: com.microfocus.example.service

src/main/java/com/microfocus/example/service/FileSystemStorageService.java, line 105 (System Information Leak)

```
102  fileMimeType = tika.detect(p.toFile());
103  } catch (IOException e) {
104  // TODO Auto-generated catch block
105  e.printStackTrace();
106  }
107  if (mimeTypeList.contains(fileMimeType))
108  return true;
```

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/EncryptedPasswordUtils.java, line 68 (System Information Leak)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Semantic)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: printStackTrace()
Enclosing Method: matches()
File: src/main/java/com/microfocus/example/utils/EncryptedPasswordUtils.java:68

```
65  encPassword1 = new String(encrypted);
66  } catch (NoSuchAlgorithmException | NoSuchPaddingException | InvalidKeyException |
IllegalBlockSizeException | BadPaddingException e) {
67  // TODO Auto-generated catch block
68  e.printStackTrace();
69  return false;
70  }
71
```

src/main/java/com/microfocus/example/utils/EncryptedPasswordUtils.java, line 51 (System Information Leak)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Semantic)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: printStackTrace()
Enclosing Method: encryptPassword()
File: src/main/java/com/microfocus/example/utils/EncryptedPasswordUtils.java:51

System Information Leak — Low

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/EncryptedPasswordUtils.java, line 51 (System Information Leak)

```
48  encrypted = desCipher.doFinal(password.getBytes());
49  } catch (NoSuchAlgorithmException | NoSuchPaddingException | InvalidKeyException |
IllegalBlockSizeException | BadPaddingException e) {
50  // TODO Auto-generated catch block
51  e.printStackTrace();
52  return null;
53  }
54
```

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 674 (System Information Leak)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Semantic)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: printStackTrace()
Enclosing Method: handleXMLUpdate()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:674

```
671  Files.delete(temp);
672  } catch (ParserConfigurationException | IOException | SAXException e) {
673  // TODO Auto-generated catch block
674  e.printStackTrace();
675  }
676
677
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 667 (System Information Leak)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Semantic)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: printStackTrace()
Enclosing Method: handleXMLUpdate()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:667

System Information Leak — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 667 (System Information Leak)

```
664  try (FileOutputStream outStream = new FileOutputStream(temp.toString())) {
665  writeXml(doc, outStream);
666  } catch (IOException | TransformerException e) {
667  e.printStackTrace();
668  }
669
670  storageService.store(temp, fpath.toString());
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 609 (System Information Leak)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Semantic)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: printStackTrace()
Enclosing Method: getXMLFileContent()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:609

```
606  }
607  } catch (ParserConfigurationException | IOException | SAXException e) {
608  // TODO Auto-generated catch block
609  e.printStackTrace();
610  }
611
612  return xmlContent;
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 605 (System Information Leak)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Semantic)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: printStackTrace()
Enclosing Method: getXMLFileContent()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:605

System Information Leak — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 605 (System Information Leak)

```
602  writeXml(doc, bytesOutStream);
603  xmlContent = bytesOutStream.toString();
604  } catch (IOException | TransformerException e) {
605  e.printStackTrace();
606  }
607  } catch (ParserConfigurationException | IOException | SAXException e) {
608  // TODO Auto-generated catch block
```

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 152 (System Information Leak)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Semantic)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: printStackTrace()
Enclosing Method: downloadFile()
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:152

```
149  try {
150  resource = new UrlResource(path.toUri());
151  } catch (MalformedURLException e) {
152  e.printStackTrace();
153  return ResponseEntity.notFound().build();
154  }
155
```

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 142 (System Information Leak)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Semantic)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: printStackTrace()
Enclosing Method: downloadFile()
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:142

System Information Leak — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 142 (System Information Leak)

```
139  try {
140  dataDir = ResourceUtils.getFile("classpath:data");
141  } catch (FileNotFoundException e) {
142  e.printStackTrace();
143  return ResponseEntity.notFound().build();
144  }
145
```

System Information Leak: External (20 issues)

Abstract

Revealing system data or debugging information helps an adversary learn about the system and form a plan of attack.

Explanation

An external information leak occurs when system data or debug information leaves the program to a remote machine via a socket or network connection. External leaks can help an attacker by revealing specific data about operating systems, full pathnames, the existence of usernames, or locations of configuration files, and are more serious than internal information leaks, which are more difficult for an attacker to access.

Example 1: The following code leaks Exception information in the HTTP response:

```
protected void doPost (HttpServletRequest req, HttpServletResponse res) throws
IOException {
    ...
    PrintWriter out = res.getWriter();
    try {
        ...
    } catch (Exception e) {
      out.println(e.getMessage());
    }
}
```

This information can be exposed to a remote user. In some cases, the error message provides the attacker with the precise type of attack to which the system is vulnerable. For example, a database error message can reveal that the application is vulnerable to a SQL injection attack. Other error messages can reveal more oblique clues about the system. In `Example 1`, the leaked information could imply information about the type of operating system, the applications installed on the system, and the amount of care that the administrators have put into configuring the program.

Information leaks are also a concern in a mobile computing environment. With mobile platforms, applications are downloaded from various sources and are run alongside each other on the same device. The likelihood of running a piece of malware next to a banking application is high, which is why application authors need to be careful about what information they include in messages addressed to other applications running on the device.

Example 2: The following code broadcasts the stack trace of a caught exception to all the registered Android receivers.

```
...
try {
  ...
} catch (Exception e) {
    String exception = Log.getStackTraceString(e);
    Intent i = new Intent();
    i.setAction("SEND_EXCEPTION");
    i.putExtra("exception", exception);
    view.getContext().sendBroadcast(i);
}
...
```

This is another scenario specific to the mobile environment. Most mobile devices now implement a Near-Field Communication (NFC) protocol for quickly sharing information between devices using radio communication. It works by bringing devices in close proximity or having the devices touch each other. Even though the communication range of NFC is limited to just a few centimeters, eavesdropping, data modification and various other types of attacks are possible, because NFC alone does not ensure secure communication.

Example 3: The Android platform provides support for NFC. The following code creates a message that

gets pushed to the other device within range.

```
...
public static final String TAG = "NfcActivity";
private static final String DATA_SPLITTER = "__:DATA:__";
private static final String MIME_TYPE = "application/my.applications.mimetype";
...
TelephonyManager tm =
(TelephonyManager)Context.getSystemService(Context.TELEPHONY_SERVICE);
String VERSION = tm.getDeviceSoftwareVersion();
...
NfcAdapter nfcAdapter = NfcAdapter.getDefaultAdapter(this);
if (nfcAdapter == null)
  return;

String text = TAG + DATA_SPLITTER + VERSION;
NdefRecord record = new NdefRecord(NdefRecord.TNF_MIME_MEDIA,
            MIME_TYPE.getBytes(), new byte[0], text.getBytes());
NdefRecord[] records = { record };
NdefMessage msg = new NdefMessage(records);
nfcAdapter.setNdefPushMessage(msg, this);
...
```

An NFC Data Exchange Format (NDEF) message contains typed data, a URI, or a custom application payload. If the message contains information about the application, such as its name, MIME type, or device software version, this information could be leaked to an eavesdropper.

Recommendation

Write error messages with security in mind. In production environments, turn off detailed error information in favor of brief messages. Restrict the generation and storage of detailed output that can help administrators and programmers diagnose problems. Debug traces can sometimes appear in non-obvious places (embedded in comments in the HTML for an error page, for example).

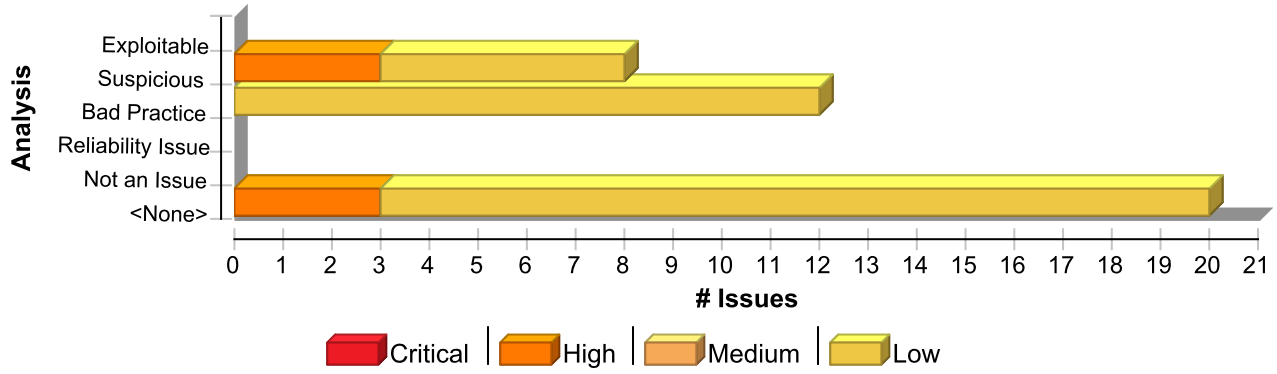
Even brief error messages that do not reveal stack traces or database dumps can potentially aid an attacker. For example, an "Access Denied" message can reveal that a file or user exists on the system. Because of this, never send information to a resource directly outside the program.

Example 4: The following code broadcasts the stack trace of a caught exception within your application only, so that it cannot be leaked to other apps on the system. Additionally, this technique is more efficient than globally broadcasting through the system.

```
...
try {
  ...
} catch (Exception e) {
    String exception = Log.getStackTraceString(e);
    Intent i = new Intent();
    i.setAction("SEND_EXCEPTION");
    i.putExtra("exception", exception);
    LocalBroadcastManager.getInstance(view.getContext()).sendBroadcast(i);
}
...
```

If you are concerned about leaking system data via NFC on an Android device, you could do one of the following three things. Do not include system data in the messages pushed to other devices in range, encrypt the payload of the message, or establish a secure communication channel at a higher layer.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
System Information Leak: External	20	0	0	20
Total	**20**	**0**	**0**	**20**

System Information Leak: External — High

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/BasicAuthenticationEntryPointCustom.java, line 69 (System Information Leak: External)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.787

Source Details

Source: java.lang.Throwable.getLocalizedMessage()
From: com.microfocus.example.config.handlers.BasicAuthenticationEntryPointCustom.commence
File: src/main/java/com/microfocus/example/config/handlers/BasicAuthenticationEntryPointCustom.java:56

```
53  response.addHeader("WWW-Authenticate", "Basic realm='" + getRealmName() + "'");
54  response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
55  ArrayList<String> errors = new ArrayList<>();
56  errors.add(ex.getLocalizedMessage());
57  ApiStatusResponse apiStatusResponse = new ApiStatusResponse
58  .ApiResponseBuilder()
59  .withSuccess(false)
```

Sink Details

Sink: java.io.PrintWriter.println()
Enclosing Method: commence()
File: src/main/java/com/microfocus/example/config/handlers/BasicAuthenticationEntryPointCustom.java:69
Taint Flags: EXCEPTIONINFO, SYSTEMINFO, VALIDATED_SERVER_SIDE_REQUEST_FORGERY

```
66  mapper.registerModule(new JavaTimeModule());
67  String jsonString = mapper.writeValueAsString(apiError.getBody());
68  PrintWriter writer = response.getWriter();
69  writer.println(jsonString);
70  }
71
72  @Override
```

src/main/java/com/microfocus/example/config/handlers/AuthenticationEntryPointJwt.java, line 69 (System Information Leak: External)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

System Information Leak: External — High

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/AuthenticationEntryPointJwt.java, line 69 (System Information Leak: External)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.787

Source Details

Source: java.lang.Throwable.getLocalizedMessage()
From: com.microfocus.example.config.handlers.AuthenticationEntryPointJwt.commence
File: src/main/java/com/microfocus/example/config/handlers/AuthenticationEntryPoint Jwt.java:56

```
53  //response.sendError(HttpServletResponse.SC_UNAUTHORIZED, "Error:
Unauthorized");
54  response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
55  ArrayList<String> errors = new ArrayList<>();
56  errors.add(ex.getLocalizedMessage());
57  ApiStatusResponse apiStatusResponse = new ApiStatusResponse
58  .ApiResponseBuilder()
59  .withSuccess(false)
```

Sink Details

Sink: java.io.PrintWriter.println()
Enclosing Method: commence()
File: src/main/java/com/microfocus/example/config/handlers/AuthenticationEntryPointJwt.java:69
Taint Flags: EXCEPTIONINFO, SYSTEMINFO, VALIDATED_SERVER_SIDE_REQUEST_FORGERY

```
66  mapper.registerModule(new JavaTimeModule());
67  String jsonString = mapper.writeValueAsString(apiError.getBody());
68  PrintWriter writer = response.getWriter();
69  writer.println(jsonString);
70  }
71
72 }
```

src/main/java/com/microfocus/example/config/handlers/ApiAccessDeniedHandler.java, line 66 (System Information Leak: External)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.787

System Information Leak: External — High

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/ApiAccessDeniedHandler.java, line 66 (System Information Leak: External)

Source Details

Source: java.lang.Throwable.getLocalizedMessage()
From: com.microfocus.example.config.handlers.ApiAccessDeniedHandler.handle
File: src/main/java/com/microfocus/example/config/handlers/ApiAccessDeniedHandler.java:53

```
50  throws IOException, ServletException {
51  response.setStatus(HttpServletResponse.SC_FORBIDDEN);
52  ArrayList<String> errors = new ArrayList<>();
53  errors.add(ex.getLocalizedMessage());
54  ApiStatusResponse apiStatusResponse = new ApiStatusResponse
55  .ApiResponseBuilder()
56  .withSuccess(false)
```

Sink Details

Sink: java.io.PrintWriter.println()
Enclosing Method: handle()
File: src/main/java/com/microfocus/example/config/handlers/ApiAccessDeniedHandler.java:66
Taint Flags: EXCEPTIONINFO, SYSTEMINFO, VALIDATED_SERVER_SIDE_REQUEST_FORGERY

```
63  mapper.registerModule(new JavaTimeModule());
64  String jsonString = mapper.writeValueAsString(apiError.getBody());
65  PrintWriter writer = response.getWriter();
66  writer.println(jsonString);
67  }
68
69 }
```

System Information Leak: External — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/BasicAuthenticationEntryPointCustom.java, line 63 (System Information Leak: External)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.603

Source Details

System Information Leak: External — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/BasicAuthenticationEntryPointCustom.java, line 63 (System Information Leak: External)

Source: java.lang.Throwable.getLocalizedMessage()
From: com.microfocus.example.config.handlers.BasicAuthenticationEntryPointCustom.commence
File: src/main/java/com/microfocus/example/config/handlers/BasicAuthenticationEntryPointCustom.java:56

```
53  response.addHeader("WWW-Authenticate", "Basic realm='" + getRealmName() + "'");
54  response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
55  ArrayList<String> errors = new ArrayList<>();
56  errors.add(ex.getLocalizedMessage());
57  ApiStatusResponse apiStatusResponse = new ApiStatusResponse
58  .ApiResponseBuilder()
59  .withSuccess(false)
```

Sink Details

Sink: org.springframework.http.ResponseEntity.ResponseEntity()
Enclosing Method: commence()
File: src/main/java/com/microfocus/example/config/handlers/BasicAuthenticationEntryPointCustom.java:63
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
60  .atTime(LocalDateTime.now(ZoneOffset.UTC))
61  .withErrors(errors)
62  .build();
63  ResponseEntity<ApiStatusResponse> apiError = new ResponseEntity<ApiStatusResponse>(apiStatusResponse, HttpStatus.UNAUTHORIZED);
64  response.setContentType(MediaType.APPLICATION_JSON_UTF8_VALUE);
65  ObjectMapper mapper = new ObjectMapper();
66  mapper.registerModule(new JavaTimeModule());
```

src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java, line 77 (System Information Leak: External)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.603

Source Details

System Information Leak: External — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java, line 77 (System Information Leak: External)

Source: java.lang.Throwable.getLocalizedMessage()
From: com.microfocus.example.config.handlers.GlobalRestExceptionHandler.userNotFound
File: src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java:70

```
67  @ExceptionHandler(UserNotFoundException.class)
68  public ResponseEntity<ApiStatusResponse> userNotFound(final UserNotFoundException ex, final WebRequest request) {
69  ArrayList<String> errors = new ArrayList<>();
70  errors.add(ex.getLocalizedMessage());
71  final ApiStatusResponse apiStatusResponse = new ApiStatusResponse
72  .ApiResponseBuilder()
73  .withSuccess(false)
```

Sink Details

Sink: org.springframework.http.ResponseEntity.ResponseEntity()
Enclosing Method: userNotFound()
File: src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java:77
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
74  .atTime(LocalDateTime.now(ZoneOffset.UTC))
75  .withErrors(errors)
76  .build();
77  return new ResponseEntity<ApiStatusResponse>(apiStatusResponse, HttpStatus.NOT_FOUND);
78  }
79
80  @ExceptionHandler(RoleNotFoundException.class)
```

src/main/java/com/microfocus/example/config/handlers/AuthenticationEntryPointJwt.java, line 63 (System Information Leak: External)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.603

Source Details

Source: java.lang.Throwable.getLocalizedMessage()
From: com.microfocus.example.config.handlers.AuthenticationEntryPointJwt.commence
File: src/main/java/com/microfocus/example/config/handlers/AuthenticationEntryPointJwt.java:56

System Information Leak: External — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/AuthenticationEntryPointJwt.java, line 63 (System Information Leak: External)

```
53  //response.sendError(HttpServletResponse.SC_UNAUTHORIZED, "Error:
Unauthorized");
54  response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
55  ArrayList<String> errors = new ArrayList<>();
56  errors.add(ex.getLocalizedMessage());
57  ApiStatusResponse apiStatusResponse = new ApiStatusResponse
58  .ApiResponseBuilder()
59  .withSuccess(false)
```

Sink Details

Sink: org.springframework.http.ResponseEntity.ResponseEntity()
Enclosing Method: commence()
File: src/main/java/com/microfocus/example/config/handlers/AuthenticationEntryPointJwt.java:63
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
60  .atTime(LocalDateTime.now(ZoneOffset.UTC))
61  .withErrors(errors)
62  .build();
63  ResponseEntity<ApiStatusResponse> apiError = new
ResponseEntity<ApiStatusResponse>(apiStatusResponse, HttpStatus.UNAUTHORIZED);
64  response.setContentType(MediaType.APPLICATION_JSON_UTF8_VALUE);
65  ObjectMapper mapper = new ObjectMapper();
66  mapper.registerModule(new JavaTimeModule());
```

src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java, line 116 (System Information Leak: External)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.603

Source Details

Source: java.lang.Throwable.getLocalizedMessage()
From: com.microfocus.example.config.handlers.GlobalRestExceptionHandler.productNotFound
File: src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java:109

System Information Leak: External — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java, line 116 (System Information Leak: External)

```
106  @ExceptionHandler(ProductNotFoundException.class)
107  public ResponseEntity<ApiStatusResponse> productNotFound(final
ProductNotFoundException ex, final WebRequest request) {
108  ArrayList<String> errors = new ArrayList<>();
109  errors.add(ex.getLocalizedMessage());
110  final ApiStatusResponse apiStatusResponse = new ApiStatusResponse
111  .ApiResponseBuilder()
112  .withSuccess(false)
```

Sink Details

Sink: org.springframework.http.ResponseEntity.ResponseEntity()
Enclosing Method: productNotFound()
File: src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java:116
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
113  .atTime(LocalDateTime.now(ZoneOffset.UTC))
114  .withErrors(errors)
115  .build();
116  return new ResponseEntity<ApiStatusResponse>(apiStatusResponse, HttpStatus.NOT_FOUND);
117  }
118
119  // Generic HTTP exception handlers
```

src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java, line 103 (System Information Leak: External)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.603

Source Details

Source: java.lang.Throwable.getLocalizedMessage()
From: com.microfocus.example.config.handlers.GlobalRestExceptionHandler.messageNotFound
File: src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java:96

System Information Leak: External — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java, line 103 (System Information Leak: External)

```
93  @ExceptionHandler(MessageNotFoundException.class)
94  public ResponseEntity<ApiStatusResponse> messageNotFound(final
MessageNotFoundException ex, final WebRequest request) {
95  ArrayList<String> errors = new ArrayList<>();
96  errors.add(ex.getLocalizedMessage());
97  final ApiStatusResponse apiStatusResponse = new ApiStatusResponse
98  .ApiResponseBuilder()
99  .withSuccess(false)
```

Sink Details

Sink: org.springframework.http.ResponseEntity.ResponseEntity()
Enclosing Method: messageNotFound()
File: src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java:103
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
100  .atTime(LocalDateTime.now(ZoneOffset.UTC))
101  .withErrors(errors)
102  .build();
103  return new ResponseEntity<ApiStatusResponse>(apiStatusResponse, HttpStatus.NOT_FOUND);
104  }
105
106  @ExceptionHandler(ProductNotFoundException.class)
```

src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java, line 116 (System Information Leak: External)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.535

Source Details

Source: java.lang.Throwable.getLocalizedMessage()
From: com.microfocus.example.config.handlers.GlobalRestExceptionHandler.productNotFound
File: src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java:109

System Information Leak: External — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java, line 116 (System Information Leak: External)

```
106  @ExceptionHandler(ProductNotFoundException.class)
107  public ResponseEntity<ApiStatusResponse> productNotFound(final
ProductNotFoundException ex, final WebRequest request) {
108  ArrayList<String> errors = new ArrayList<>();
109  errors.add(ex.getLocalizedMessage());
110  final ApiStatusResponse apiStatusResponse = new ApiStatusResponse
111  .ApiResponseBuilder()
112  .withSuccess(false)
```

Sink Details

Sink: Return
Enclosing Method: productNotFound()
File: src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java:116
Taint Flags: EXCEPTIONINFO, SYSTEMINFO, VALIDATED_SERVER_SIDE_REQUEST_FORGERY

```
113  .atTime(LocalDateTime.now(ZoneOffset.UTC))
114  .withErrors(errors)
115  .build();
116  return new ResponseEntity<ApiStatusResponse>(apiStatusResponse, HttpStatus.NOT_FOUND);
117  }
118
119  // Generic HTTP exception handlers
```

src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java, line 90 (System Information Leak: External)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.535

Source Details

Source: java.lang.Throwable.getLocalizedMessage()
From: com.microfocus.example.config.handlers.GlobalRestExceptionHandler.roleNotFound
File: src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java:83

System Information Leak: External — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java, line 90 (System Information Leak: External)

```
80  @ExceptionHandler(RoleNotFoundException.class)
81  public ResponseEntity<ApiStatusResponse> roleNotFound(final
RoleNotFoundException ex, final WebRequest request) {
82  ArrayList<String> errors = new ArrayList<>();
83  errors.add(ex.getLocalizedMessage());
84  final ApiStatusResponse apiStatusResponse = new ApiStatusResponse
85  .ApiResponseBuilder()
86  .withSuccess(false)
```

Sink Details

Sink: Return
Enclosing Method: roleNotFound()
File: src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java:90
Taint Flags: EXCEPTIONINFO, SYSTEMINFO, VALIDATED_SERVER_SIDE_REQUEST_FORGERY

```
87  .atTime(LocalDateTime.now(ZoneOffset.UTC))
88  .withErrors(errors)
89  .build();
90  return new ResponseEntity<ApiStatusResponse>(apiStatusResponse, HttpStatus.NOT_FOUND);
91  }
92
93  @ExceptionHandler(MessageNotFoundException.class)
```

src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java, line 342 (System Information Leak: External)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.603

Source Details

Source: java.lang.Throwable.getLocalizedMessage()
From: com.microfocus.example.config.handlers.GlobalRestExceptionHandler.handleAll
File: src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java:335

System Information Leak: External — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java, line 342 (System Information Leak: External)

```
332  log.debug("GlobalRestExceptionHandler::handleAll");
333  log.error("error:" + ex.toString());
334  ArrayList<String> errors = new ArrayList<>();
335  errors.add(ex.getLocalizedMessage());
336  final ApiStatusResponse apiStatusResponse = new ApiStatusResponse
337  .ApiResponseBuilder()
338  .withSuccess(false)
```

Sink Details

Sink: org.springframework.http.ResponseEntity.ResponseEntity()
Enclosing Method: handleAll()
File: src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java:342
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
339  .atTime(LocalDateTime.now(ZoneOffset.UTC))
340  .withErrors(errors)
341  .build();
342  return new ResponseEntity<Object>(apiStatusResponse, new HttpHeaders(),
HttpStatus.INTERNAL_SERVER_ERROR);
343  }
344
345 }
```

src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java, line 267 (System Information Leak: External)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.603

Source Details

Source: java.lang.Throwable.getLocalizedMessage()
From: com.microfocus.example.config.handlers.GlobalRestExceptionHandler.handleHttpMessageNotReadable
File: src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java:260

System Information Leak: External — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java, line 267 (System Information Leak: External)

```
257  protected ResponseEntity<Object> handleHttpMessageNotReadable(HttpMessageNotReadableException ex, HttpHeaders headers, HttpStatus status, WebRequest request) {
258  ArrayList<String> errors = new ArrayList<>();
259  //final String error = "No handler found for " + ex.getHttpMethod() + " " + ex.getRequestURL();
260  errors.add(ex.getLocalizedMessage());
261  final ApiStatusResponse apiStatusResponse = new ApiStatusResponse
262  .ApiResponseBuilder()
263  .withSuccess(false)
```

Sink Details

Sink: org.springframework.http.ResponseEntity.ResponseEntity()
Enclosing Method: handleHttpMessageNotReadable()
File: src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java:267
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
264  .atTime(LocalDateTime.now(ZoneOffset.UTC))
265  .withErrors(errors)
266  .build();
267  return new ResponseEntity<Object>(apiStatusResponse, new HttpHeaders(), HttpStatus.NOT_FOUND);
268  }
269
270  @Override
```

src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java, line 103 (System Information Leak: External)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.535

Source Details

Source: java.lang.Throwable.getLocalizedMessage()
From: com.microfocus.example.config.handlers.GlobalRestExceptionHandler.messageNotFound
File: src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java:96

System Information Leak: External — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java, line 103 (System Information Leak: External)

```
93  @ExceptionHandler(MessageNotFoundException.class)
94  public ResponseEntity<ApiStatusResponse> messageNotFound(final
MessageNotFoundException ex, final WebRequest request) {
95  ArrayList<String> errors = new ArrayList<>();
96  errors.add(ex.getLocalizedMessage());
97  final ApiStatusResponse apiStatusResponse = new ApiStatusResponse
98  .ApiResponseBuilder()
99  .withSuccess(false)
```

Sink Details

Sink: Return
Enclosing Method: messageNotFound()
File: src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java:103
Taint Flags: EXCEPTIONINFO, SYSTEMINFO, VALIDATED_SERVER_SIDE_REQUEST_FORGERY

```
100  .atTime(LocalDateTime.now(ZoneOffset.UTC))
101  .withErrors(errors)
102  .build();
103  return new ResponseEntity<ApiStatusResponse>(apiStatusResponse, HttpStatus.NOT_FOUND);
104  }
105
106  @ExceptionHandler(ProductNotFoundException.class)
```

src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java, line 221 (System Information Leak: External)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.672

Source Details

Source: org.springframework.web.bind.MissingServletRequestParameterException.getParameterName()
From: com.microfocus.example.config.handlers.GlobalRestExceptionHandler.handleMissingServletRequestParameter
File: src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java:213

System Information Leak: External — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java, line 221 (System Information Leak: External)

```
210  final HttpHeaders headers, final HttpStatus status,
211  final WebRequest request) {
212  ArrayList<String> errors = new ArrayList<>();
213  final String error = ex.getParameterName() + " parameter is missing";
214  errors.add(error);
215  final ApiStatusResponse apiStatusResponse = new ApiStatusResponse
216  .ApiResponseBuilder()
```

Sink Details

Sink: org.springframework.http.ResponseEntity.ResponseEntity()
Enclosing Method: handleMissingServletRequestParameter()
File: src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java:221
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
218  .atTime(LocalDateTime.now(ZoneOffset.UTC))
219  .withErrors(errors)
220  .build();
221  return new ResponseEntity<>(apiStatusResponse, new HttpHeaders(), HttpStatus.BAD_REQUEST);
222  }
223
224  @ExceptionHandler({MethodArgumentTypeMismatchException.class})
```

src/main/java/com/microfocus/example/config/handlers/ApiAccessDeniedHandler.java, line 60 (System Information Leak: External)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.603

Source Details

Source: java.lang.Throwable.getLocalizedMessage()
From: com.microfocus.example.config.handlers.ApiAccessDeniedHandler.handle
File: src/main/java/com/microfocus/example/config/handlers/ApiAccessDeniedHandler.java:53

System Information Leak: External — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/ApiAccessDeniedHandler.java, line 60 (System Information Leak: External)

```
50  throws IOException, ServletException {
51  response.setStatus(HttpServletResponse.SC_FORBIDDEN);
52  ArrayList<String> errors = new ArrayList<>();
53  errors.add(ex.getLocalizedMessage());
54  ApiStatusResponse apiStatusResponse = new ApiStatusResponse
55  .ApiResponseBuilder()
56  .withSuccess(false)
```

Sink Details

Sink: org.springframework.http.ResponseEntity.ResponseEntity()
Enclosing Method: handle()
File: src/main/java/com/microfocus/example/config/handlers/ApiAccessDeniedHandler.java:60
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
57  .atTime(LocalDateTime.now(ZoneOffset.UTC))
58  .withErrors(errors)
59  .build();
60  ResponseEntity<ApiStatusResponse> apiError = new
ResponseEntity<ApiStatusResponse>(apiStatusResponse, HttpStatus.FORBIDDEN);
61  response.setContentType(MediaType.APPLICATION_JSON_UTF8_VALUE);
62  ObjectMapper mapper = new ObjectMapper();
63  mapper.registerModule(new JavaTimeModule());
```

src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java, line 90 (System Information Leak: External)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.603

Source Details

Source: java.lang.Throwable.getLocalizedMessage()
From: com.microfocus.example.config.handlers.GlobalRestExceptionHandler.roleNotFound
File: src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java:83

System Information Leak: External — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java, line 90 (System Information Leak: External)

```
80  @ExceptionHandler(RoleNotFoundException.class)
81  public ResponseEntity<ApiStatusResponse> roleNotFound(final
RoleNotFoundException ex, final WebRequest request) {
82  ArrayList<String> errors = new ArrayList<>();
83  errors.add(ex.getLocalizedMessage());
84  final ApiStatusResponse apiStatusResponse = new ApiStatusResponse
85  .ApiResponseBuilder()
86  .withSuccess(false)
```

Sink Details

Sink: org.springframework.http.ResponseEntity.ResponseEntity()
Enclosing Method: roleNotFound()
File: src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java:90
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
87  .atTime(LocalDateTime.now(ZoneOffset.UTC))
88  .withErrors(errors)
89  .build();
90  return new ResponseEntity<ApiStatusResponse>(apiStatusResponse, HttpStatus.NOT_FOUND);
91  }
92
93  @ExceptionHandler(MessageNotFoundException.class)
```

src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java, line 342 (System Information Leak: External)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.535

Source Details

Source: java.lang.Throwable.getLocalizedMessage()
From: com.microfocus.example.config.handlers.GlobalRestExceptionHandler.handleAll
File: src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java:335

System Information Leak: External — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java, line 342 (System Information Leak: External)

```
332  log.debug("GlobalRestExceptionHandler::handleAll");
333  log.error("error:" + ex.toString());
334  ArrayList<String> errors = new ArrayList<>();
335  errors.add(ex.getLocalizedMessage());
336  final ApiStatusResponse apiStatusResponse = new ApiStatusResponse
337  .ApiResponseBuilder()
338  .withSuccess(false)
```

Sink Details

Sink: Return
Enclosing Method: handleAll()
File: src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java:342
Taint Flags: EXCEPTIONINFO, SYSTEMINFO, VALIDATED_SERVER_SIDE_REQUEST_FORGERY

```
339  .atTime(LocalDateTime.now(ZoneOffset.UTC))
340  .withErrors(errors)
341  .build();
342  return new ResponseEntity<Object>(apiStatusResponse, new HttpHeaders(),
HttpStatus.INTERNAL_SERVER_ERROR);
343  }
344
345 }
```

src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java, line 77 (System Information Leak: External)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.535

Source Details

Source: java.lang.Throwable.getLocalizedMessage()
From: com.microfocus.example.config.handlers.GlobalRestExceptionHandler.userNotFoun
d
File: src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandl
er.java:70

System Information Leak: External — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java, line 77 (System Information Leak: External)

```
67  @ExceptionHandler(UserNotFoundException.class)
68  public ResponseEntity<ApiStatusResponse> userNotFound(final
UserNotFoundException ex, final WebRequest request) {
69  ArrayList<String> errors = new ArrayList<>();
70  errors.add(ex.getLocalizedMessage());
71  final ApiStatusResponse apiStatusResponse = new ApiStatusResponse
72  .ApiResponseBuilder()
73  .withSuccess(false)
```

Sink Details

Sink: Return
Enclosing Method: userNotFound()
File: src/main/java/com/microfocus/example/config/handlers/GlobalRestExceptionHandler.java:77
Taint Flags: EXCEPTIONINFO, SYSTEMINFO, VALIDATED_SERVER_SIDE_REQUEST_FORGERY

```
74  .atTime(LocalDateTime.now(ZoneOffset.UTC))
75  .withErrors(errors)
76  .build();
77  return new ResponseEntity<ApiStatusResponse>(apiStatusResponse, HttpStatus.NOT_FOUND);
78  }
79
80  @ExceptionHandler(RoleNotFoundException.class)
```

Package: com.microfocus.example.service

src/main/java/com/microfocus/example/service/EmailSenderService.java, line 65 (System Information Leak: External)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.502

Source Details

Source: Read this.emailFromAddress
From: com.microfocus.example.web.controllers.UserController.registerUser
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:445

System Information Leak: External — Low

Package: com.microfocus.example.service

src/main/java/com/microfocus/example/service/EmailSenderService.java, line 65 (System Information Leak: External)

```
442  Mail mail = new Mail();
443  mail.setMailTo(u.getEmail());
444  mail.setFrom(emailFromAddress);
445  mail.setReplyTo(emailFromAddress);
446  mail.setSubject("[IWA Pharmacy Direct] Verify your account");
447
448  Map<String, Object> props = new HashMap<String, Object>();
```

Sink Details

Sink: org.springframework.mail.javamail.MimeMessageHelper.setReplyTo()
Enclosing Method: sendEmail()
File: src/main/java/com/microfocus/example/service/EmailSenderService.java:65
Taint Flags: ARGS, ENVIRONMENT, PROPERTY

```
62  helper.setText(html, true);
63  helper.setSubject(mail.getSubject());
64  helper.setFrom(mail.getFrom());
65  helper.setReplyTo(mail.getReplyTo());
66  emailSender.send(message);
67  }
68 }
```

src/main/java/com/microfocus/example/service/EmailSenderService.java, line 64 (System Information Leak: External)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.502

Source Details

Source: Read this.emailFromAddress
From: com.microfocus.example.web.controllers.UserController.registerUser
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:444

System Information Leak: External — Low

Package: com.microfocus.example.service

src/main/java/com/microfocus/example/service/EmailSenderService.java, line 64 (System Information Leak: External)

```
441
442   Mail mail = new Mail();
443   mail.setMailTo(u.getEmail());
444   mail.setFrom(emailFromAddress);
445   mail.setReplyTo(emailFromAddress);
446   mail.setSubject("[IWA Pharmacy Direct] Verify your account");
447
```

Sink Details

Sink: org.springframework.mail.javamail.MimeMessageHelper.setFrom()
Enclosing Method: sendEmail()
File: src/main/java/com/microfocus/example/service/EmailSenderService.java:64
Taint Flags: ARGS, ENVIRONMENT, PROPERTY

```
61  helper.setTo(mail.getMailTo());
62  helper.setText(html, true);
63  helper.setSubject(mail.getSubject());
64  helper.setFrom(mail.getFrom());
65  helper.setReplyTo(mail.getReplyTo());
66  emailSender.send(message);
67  }
```

System Information Leak: Internal (24 issues)

Abstract

Revealing system data or debugging information helps an adversary learn about the system and form a plan of attack.

Explanation

An internal information leak occurs when system data or debug information is sent to a local file, console, or screen via printing or logging.

Example 1: The following code writes an exception to the standard error stream:

```
try {
    ...
} catch (Exception e) {
    e.printStackTrace();
}
```

Depending upon the system configuration, this information can be dumped to a console, written to a log file, or exposed to a user. In some cases, the error message provides the attacker with the precise type of attack to which the system is vulnerable. For example, a database error message can reveal that the application is vulnerable to a SQL injection attack. Other error messages can reveal more oblique clues about the system. In `Example 1`, the leaked information could imply information about the type of operating system, the applications installed on the system, and the amount of care that the administrators have put into configuring the program.

Information leaks are also a concern in a mobile computing environment.

Example 2: The following code logs the stack trace of a caught exception on the Android platform.

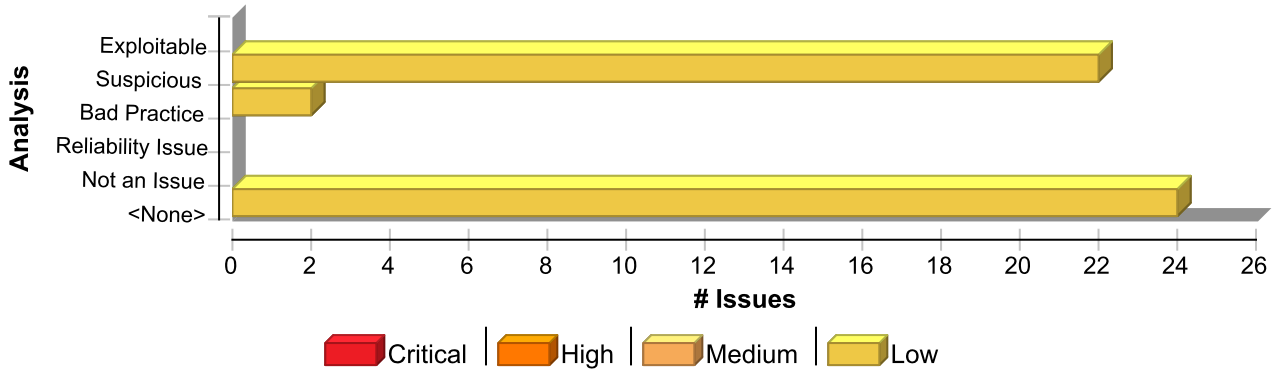
```
...
try {
  ...
} catch (Exception e) {
    Log.e(TAG, Log.getStackTraceString(e));
}
...
```

Recommendation

Write error messages with security in mind. In production environments, turn off detailed error information in favor of brief messages. Restrict the generation and storage of detailed output that can help administrators and programmers diagnose problems. Debug traces can sometimes appear in non-obvious places (embedded in comments in the HTML for an error page, for example).

Even brief error messages that do not reveal stack traces or database dumps can potentially aid an attacker. For example, an "Access Denied" message can reveal that a file or user exists on the system.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
System Information Leak: Internal	24	0	0	24
Total	**24**	**0**	**0**	**24**

System Information Leak: Internal — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/CustomAuthenticationSuccessHandler.java, line 119 (System Information Leak: Internal)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.635

Source Details

Source: java.lang.Throwable.getLocalizedMessage()
From: com.microfocus.example.config.handlers.CustomAuthenticationSuccessHandler.getTargetUrl
File: src/main/java/com/microfocus/example/config/handlers/CustomAuthenticationSuccessHandler.java:119

```
116  try {
117  targetPath = new URL(targetUrl).getPath();
118  } catch (MalformedURLException ex) {
119  log.error(ex.getLocalizedMessage());
120  }
121  if (targetUrl.contains("?")) targetUrl = targetUrl.substring(0, targetUrl.indexOf("?"));
122  if (targetPath.endsWith("/cart")) {
```

Sink Details

System Information Leak: Internal — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/CustomAuthenticationSuccessHandler.java, line 119 (System Information Leak: Internal)

Sink: org.slf4j.Logger.error()
Enclosing Method: getTargetUrl()
File: src/main/java/com/microfocus/example/config/handlers/CustomAuthenticationSuccessHandler.java:119
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
116  try {
117  targetPath = new URL(targetUrl).getPath();
118  } catch (MalformedURLException ex) {
119  log.error(ex.getLocalizedMessage());
120  }
121  if (targetUrl.contains("?")) targetUrl = targetUrl.substring(0, targetUrl.indexOf("?"));
122  if (targetPath.endsWith("/cart")) {
```

src/main/java/com/microfocus/example/config/handlers/AuthenticationTokenFilter.java, line 67 (System Information Leak: Internal)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.582

Source Details

Source: Read e
From: com.microfocus.example.config.handlers.AuthenticationTokenFilter.doFilterInternal
File: src/main/java/com/microfocus/example/config/handlers/AuthenticationTokenFilter.java:67

```
64  SecurityContextHolder.getContext().setAuthentication(authentication);
65  }
66  } catch (Exception e) {
67  logger.error("Cannot set user authentication: {}", e);
68  }
69
70  filterChain.doFilter(request, response);
```

Sink Details

Sink: org.apache.commons.logging.Log.error()
Enclosing Method: doFilterInternal()
File: src/main/java/com/microfocus/example/config/handlers/AuthenticationTokenFilter.java:67
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

System Information Leak: Internal — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/AuthenticationTokenFilter.java, line 67 (System Information Leak: Internal)

```
64  SecurityContextHolder.getContext().setAuthentication(authentication);
65  }
66  } catch (Exception e) {
67  logger.error("Cannot set user authentication: {}", e);
68  }
69
70  filterChain.doFilter(request, response);
```

src/main/java/com/microfocus/example/config/handlers/CustomAuthenticationSuccessHandler.java, line 153 (System Information Leak: Internal)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.635

Source Details

Source: java.lang.Throwable.getLocalizedMessage()
From: com.microfocus.example.config.handlers.CustomAuthenticationSuccessHandler.requestAndRegisterVerification
File: src/main/java/com/microfocus/example/config/handlers/CustomAuthenticationSuccessHandler.java:153

```
150  log.debug("Generated OTP '" + String.valueOf(otp) + "' for user id: " + userId.toString());
151  return (otp != 0);
152  } catch (VerificationRequestFailedException ex) {
153  log.error(ex.getLocalizedMessage());
154  return false;
155  }
156  }
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: requestAndRegisterVerification()
File: src/main/java/com/microfocus/example/config/handlers/CustomAuthenticationSuccessHandler.java:153
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

System Information Leak: Internal — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/CustomAuthenticationSuccessHandler.java, line 153 (System Information Leak: Internal)

```
150  log.debug("Generated OTP '" + String.valueOf(otp) + "' for user id: " +
userId.toString());
151  return (otp != 0);
152  } catch (VerificationRequestFailedException ex) {
153  log.error(ex.getLocalizedMessage());
154  return false;
155  }
156  }
```

Package: com.microfocus.example.repository

src/main/java/com/microfocus/example/repository/UserRepositoryImpl.java, line 123 (System Information Leak: Internal)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.635

Source Details

Source: java.lang.Throwable.getLocalizedMessage()
From: com.microfocus.example.repository.UserRepositoryImpl$1.execute
File: src/main/java/com/microfocus/example/repository/UserRepositoryImpl.java:123

```
120  log.debug("No matching users found");
121  }
122  } catch (SQLException ex) {
123  log.error(ex.getLocalizedMessage());
124  }
125  return authorityCount;
126  }
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: execute()
File: src/main/java/com/microfocus/example/repository/UserRepositoryImpl.java:123
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

System Information Leak: Internal — Low

Package: com.microfocus.example.repository

src/main/java/com/microfocus/example/repository/UserRepositoryImpl.java, line 123 (System Information Leak: Internal)

```
120  log.debug("No matching users found");
121  }
122  } catch (SQLException ex) {
123  log.error(ex.getLocalizedMessage());
124  }
125  return authorityCount;
126  }
```

Package: com.microfocus.example.service

src/main/java/com/microfocus/example/service/VerificationService.java, line 70 (System Information Leak: Internal)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.635

Source Details

Source: java.lang.Throwable.getLocalizedMessage()
From: com.microfocus.example.service.VerificationService.getOtp
File: src/main/java/com/microfocus/example/service/VerificationService.java:70

```
67  try {
68  return otpCache.get(key);
69  } catch (Exception ex){
70  log.error(ex.getLocalizedMessage());
71  return 0;
72  }
73  }
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: getOtp()
File: src/main/java/com/microfocus/example/service/VerificationService.java:70
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

System Information Leak: Internal — Low

Package: com.microfocus.example.service

src/main/java/com/microfocus/example/service/VerificationService.java, line 70 (System Information Leak: Internal)

```
67  try {
68  return otpCache.get(key);
69  } catch (Exception ex){
70  log.error(ex.getLocalizedMessage());
71  return 0;
72  }
73  }
```

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/JwtUtils.java, line 90 (System Information Leak: Internal)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.635

Source Details

Source: java.lang.Throwable.getMessage()
From: com.microfocus.example.utils.JwtUtils.validateJwtToken
File: src/main/java/com/microfocus/example/utils/JwtUtils.java:90

```
87  } catch (SignatureException e) {
88  log.error("Invalid JWT signature: {}", e.getMessage());
89  } catch (MalformedJwtException e) {
90  log.error("Invalid JWT token: {}", e.getMessage());
91  } catch (ExpiredJwtException e) {
92  log.error("JWT token is expired: {}", e.getMessage());
93  } catch (UnsupportedJwtException e) {
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: validateJwtToken()
File: src/main/java/com/microfocus/example/utils/JwtUtils.java:90
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

System Information Leak: Internal — Low

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/JwtUtils.java, line 90 (System Information Leak: Internal)

```
87  } catch (SignatureException e) {
88  log.error("Invalid JWT signature: {}", e.getMessage());
89  } catch (MalformedJwtException e) {
90  log.error("Invalid JWT token: {}", e.getMessage());
91  } catch (ExpiredJwtException e) {
92  log.error("JWT token is expired: {}", e.getMessage());
93  } catch (UnsupportedJwtException e) {
```

src/main/java/com/microfocus/example/utils/JwtUtils.java, line 96 (System Information Leak: Internal)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.635

Source Details

Source: java.lang.Throwable.getMessage()
From: com.microfocus.example.utils.JwtUtils.validateJwtToken
File: src/main/java/com/microfocus/example/utils/JwtUtils.java:96

```
93  } catch (UnsupportedJwtException e) {
94  log.error("JWT token is unsupported: {}", e.getMessage());
95  } catch (IllegalArgumentException e) {
96  log.error("JWT claims string is empty: {}", e.getMessage());
97  }
98
99  return false;
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: validateJwtToken()
File: src/main/java/com/microfocus/example/utils/JwtUtils.java:96
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

System Information Leak: Internal — Low

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/JwtUtils.java, line 96 (System Information Leak: Internal)

```
93  } catch (UnsupportedJwtException e) {
94  log.error("JWT token is unsupported: {}", e.getMessage());
95  } catch (IllegalArgumentException e) {
96  log.error("JWT claims string is empty: {}", e.getMessage());
97  }
98
99  return false;
```

src/main/java/com/microfocus/example/utils/JwtUtils.java, line 94 (System Information Leak: Internal)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.635

Source Details

Source: java.lang.Throwable.getMessage()
From: com.microfocus.example.utils.JwtUtils.validateJwtToken
File: src/main/java/com/microfocus/example/utils/JwtUtils.java:94

```
91  } catch (ExpiredJwtException e) {
92  log.error("JWT token is expired: {}", e.getMessage());
93  } catch (UnsupportedJwtException e) {
94  log.error("JWT token is unsupported: {}", e.getMessage());
95  } catch (IllegalArgumentException e) {
96  log.error("JWT claims string is empty: {}", e.getMessage());
97  }
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: validateJwtToken()
File: src/main/java/com/microfocus/example/utils/JwtUtils.java:94
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

System Information Leak: Internal — Low

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/JwtUtils.java, line 94 (System Information Leak: Internal)

```
91  } catch (ExpiredJwtException e) {
92  log.error("JWT token is expired: {}", e.getMessage());
93  } catch (UnsupportedJwtException e) {
94  log.error("JWT token is unsupported: {}", e.getMessage());
95  } catch (IllegalArgumentException e) {
96  log.error("JWT claims string is empty: {}", e.getMessage());
97  }
```

src/main/java/com/microfocus/example/utils/UserUtils.java, line 84 (System Information Leak: Internal)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.631

Source Details

Source: java.lang.System.getProperty()
From: com.microfocus.example.utils.UserUtils.getFilePath
File: src/main/java/com/microfocus/example/utils/UserUtils.java:138

```
135   }
136
137   private static String getFilePath(String relativePath) {
138   return System.getProperty("user.home") + File.separatorChar +
relativePath;
139   }
140
141 }
```

Sink Details

Sink: org.slf4j.Logger.debug()
Enclosing Method: registerUser()
File: src/main/java/com/microfocus/example/utils/UserUtils.java:84
Taint Flags: FILE_SYSTEM, NO_NEW_LINE, SYSTEMINFO

System Information Leak: Internal — Low

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/UserUtils.java, line 84 (System Information Leak: Internal)

```
81  jsonArray = (JSONArray) jsonParser.parse(new
FileReader(getFilePath(NEWSLETTER_USER_FILE)));
82  } else {
83  dataFile.createNewFile();
84  log.debug("Created: " + getFilePath(NEWSLETTER_USER_FILE));
85  }
86
87  try (OutputStream fos = new FileOutputStream(dataFile, false)) {
```

src/main/java/com/microfocus/example/utils/JwtUtils.java, line 92 (System Information Leak: Internal)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.635

Source Details

Source: java.lang.Throwable.getMessage()
From: com.microfocus.example.utils.JwtUtils.validateJwtToken
File: src/main/java/com/microfocus/example/utils/JwtUtils.java:92

```
89  } catch (MalformedJwtException e) {
90  log.error("Invalid JWT token: {}", e.getMessage());
91  } catch (ExpiredJwtException e) {
92  log.error("JWT token is expired: {}", e.getMessage());
93  } catch (UnsupportedJwtException e) {
94  log.error("JWT token is unsupported: {}", e.getMessage());
95  } catch (IllegalArgumentException e) {
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: validateJwtToken()
File: src/main/java/com/microfocus/example/utils/JwtUtils.java:92
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

System Information Leak: Internal — Low

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/JwtUtils.java, line 92 (System Information Leak: Internal)

```
89  } catch (MalformedJwtException e) {
90  log.error("Invalid JWT token: {}", e.getMessage());
91  } catch (ExpiredJwtException e) {
92  log.error("JWT token is expired: {}", e.getMessage());
93  } catch (UnsupportedJwtException e) {
94  log.error("JWT token is unsupported: {}", e.getMessage());
95  } catch (IllegalArgumentException e) {
```

src/main/java/com/microfocus/example/utils/UserUtils.java, line 52 (System Information Leak: Internal)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.631

Source Details

Source: java.lang.System.getProperty()
From: com.microfocus.example.utils.UserUtils.getFilePath
File: src/main/java/com/microfocus/example/utils/UserUtils.java:138

```
135   }
136
137   private static String getFilePath(String relativePath) {
138   return System.getProperty("user.home") + File.separatorChar +
relativePath;
139   }
140
141 }
```

Sink Details

Sink: org.slf4j.Logger.debug()
Enclosing Method: writeUser()
File: src/main/java/com/microfocus/example/utils/UserUtils.java:52
Taint Flags: FILE_SYSTEM, NO_NEW_LINE, SYSTEMINFO

System Information Leak: Internal — Low

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/UserUtils.java, line 52 (System Information Leak: Internal)

```
49
50  File dataFile = new File(getFilePath(USER_INFO_FILE));
51  if (dataFile.createNewFile()){
52  log.debug("Created: " + getFilePath(USER_INFO_FILE));
53  }
54
55  JsonGenerator jGenerator = jsonFactory.createGenerator(dataFile, JsonEncoding.UTF8);
```

src/main/java/com/microfocus/example/utils/JwtUtils.java, line 88 (System Information Leak: Internal)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.635

Source Details

Source: java.lang.Throwable.getMessage()
From: com.microfocus.example.utils.JwtUtils.validateJwtToken
File: src/main/java/com/microfocus/example/utils/JwtUtils.java:88

```
85  Jwts.parser().setSigningKey(jwtSecret).parseClaimsJws(authToken);
86  return true;
87  } catch (SignatureException e) {
88  log.error("Invalid JWT signature: {}", e.getMessage());
89  } catch (MalformedJwtException e) {
90  log.error("Invalid JWT token: {}", e.getMessage());
91  } catch (ExpiredJwtException e) {
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: validateJwtToken()
File: src/main/java/com/microfocus/example/utils/JwtUtils.java:88
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

System Information Leak: Internal — Low

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/JwtUtils.java, line 88 (System Information Leak: Internal)

```
85  Jwts.parser().setSigningKey(jwtSecret).parseClaimsJws(authToken);
86  return true;
87  } catch (SignatureException e) {
88  log.error("Invalid JWT signature: {}", e.getMessage());
89  } catch (MalformedJwtException e) {
90  log.error("Invalid JWT token: {}", e.getMessage());
91  } catch (ExpiredJwtException e) {
```

src/main/java/com/microfocus/example/utils/UserUtils.java, line 83 (System Information Leak: Internal)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.565

Source Details

Source: java.lang.System.getProperty()
From: com.microfocus.example.utils.UserUtils.getFilePath
File: src/main/java/com/microfocus/example/utils/UserUtils.java:138

```
135   }
136
137   private static String getFilePath(String relativePath) {
138   return System.getProperty("user.home") + File.separatorChar +
relativePath;
139   }
140
141 }
```

Sink Details

Sink: java.io.File.createNewFile()
Enclosing Method: registerUser()
File: src/main/java/com/microfocus/example/utils/UserUtils.java:83
Taint Flags: FILE_SYSTEM, NO_NEW_LINE, SYSTEMINFO, TAINTED_PATH

System Information Leak: Internal — Low

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/UserUtils.java, line 83 (System Information Leak: Internal)

```
80  if (dataFile.exists()) {
81  jsonArray = (JSONArray) jsonParser.parse(new
FileReader(getFilePath(NEWSLETTER_USER_FILE)));
82  } else {
83  dataFile.createNewFile();
84  log.debug("Created: " + getFilePath(NEWSLETTER_USER_FILE));
85  }
86
```

src/main/java/com/microfocus/example/utils/UserUtils.java, line 51 (System Information Leak: Internal)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.565

Source Details

Source: java.lang.System.getProperty()
From: com.microfocus.example.utils.UserUtils.getFilePath
File: src/main/java/com/microfocus/example/utils/UserUtils.java:138

```
135  }
136
137  private static String getFilePath(String relativePath) {
138  return System.getProperty("user.home") + File.separatorChar +
relativePath;
139  }
140
141 }
```

Sink Details

Sink: java.io.File.createNewFile()
Enclosing Method: writeUser()
File: src/main/java/com/microfocus/example/utils/UserUtils.java:51
Taint Flags: FILE_SYSTEM, NO_NEW_LINE, SYSTEMINFO, TAINTED_PATH

System Information Leak: Internal — Low

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/UserUtils.java, line 51 (System Information Leak: Internal)

```
48  JsonFactory jsonFactory = new JsonFactory();
49
50  File dataFile = new File(getFilePath(USER_INFO_FILE));
51  if (dataFile.createNewFile()){
52  log.debug("Created: " + getFilePath(USER_INFO_FILE));
53  }
54
```

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 370 (System Information Leak: Internal)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.582

Source Details

Source: Read this.USER_NOT_FOUND_ERROR
From: com.microfocus.example.web.controllers.UserController.userSaveProfile
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:370

```
367  FieldError passwordError = new FieldError("userForm", "password",
ex.getMessage());
368  bindingResult.addError(passwordError);
369  } catch (UserNotFoundException ex) {
370  log.error(USER_NOT_FOUND_ERROR);
371  FieldError usernameError = new FieldError("userForm", "username",
ex.getMessage());
372  bindingResult.addError(usernameError);
373  }
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: userSaveProfile()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:370
Taint Flags: ARGS, ENVIRONMENT, PROPERTY

System Information Leak: Internal — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 370 (System Information Leak: Internal)

```
367  FieldError passwordError = new FieldError("userForm", "password", ex.getMessage());
368  bindingResult.addError(passwordError);
369  } catch (UserNotFoundException ex) {
370  log.error(USER_NOT_FOUND_ERROR);
371  FieldError usernameError = new FieldError("userForm", "username", ex.getMessage());
372  bindingResult.addError(usernameError);
373  }
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 457 (System Information Leak: Internal)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.635

Source Details

Source: java.lang.Throwable.getLocalizedMessage()
From: com.microfocus.example.web.controllers.UserController.registerUser
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:457

```
454  try {
455  emailSenderService.sendEmail(mail, "email/default");
456  } catch (Exception ex) {
457  log.error(ex.getLocalizedMessage());
458  }
459
460  this.setModelDefaults(model, null, "verify");
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: registerUser()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:457
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

System Information Leak: Internal — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 457 (System Information Leak: Internal)

```
454  try {
455  emailSenderService.sendEmail(mail, "email/default");
456  } catch (Exception ex) {
457  log.error(ex.getLocalizedMessage());
458  }
459
460  this.setModelDefaults(model, null, "verify");
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 366 (System Information Leak: Internal)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.582

Source Details

Source: Read this.AUTHENTICATION_ERROR
From: com.microfocus.example.web.controllers.UserController.userSaveProfile
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:366

```
363  this.setModelDefaults(model, principal, "profile");
364  return "redirect:/user/profile";
365  } catch (InvalidPasswordException ex) {
366  log.error(AUTHENTICATION_ERROR);
367  FieldError passwordError = new FieldError("userForm", "password",
ex.getMessage());
368  bindingResult.addError(passwordError);
369  } catch (UserNotFoundException ex) {
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: userSaveProfile()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:366
Taint Flags: ARGS, ENVIRONMENT, PROPERTY

System Information Leak: Internal — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 366 (System Information Leak: Internal)

```
363  this.setModelDefaults(model, principal, "profile");
364  return "redirect:/user/profile";
365  } catch (InvalidPasswordException ex) {
366  log.error(AUTHENTICATION_ERROR);
367  FieldError passwordError = new FieldError("userForm", "password", ex.getMessage());
368  bindingResult.addError(passwordError);
369  } catch (UserNotFoundException ex) {
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 398 (System Information Leak: Internal)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.582

Source Details

Source: Read this.AUTHENTICATION_ERROR
From: com.microfocus.example.web.controllers.UserController.userSavePassword
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:398

```
395  redirectAttributes.addFlashAttribute("alertClass", "alert-success");
396  return "redirect:/logout";
397  } catch (InvalidPasswordException ex) {
398  log.error(AUTHENTICATION_ERROR);
399  FieldError passwordError = new FieldError("passwordForm", "password",
ex.getMessage());
400  bindingResult.addError(passwordError);
401  } catch (UserNotFoundException ex) {
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: userSavePassword()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:398
Taint Flags: ARGS, ENVIRONMENT, PROPERTY

System Information Leak: Internal — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 398 (System Information Leak: Internal)

```
395  redirectAttributes.addFlashAttribute("alertClass", "alert-success");
396  return "redirect:/logout";
397  } catch (InvalidPasswordException ex) {
398  log.error(AUTHENTICATION_ERROR);
399  FieldError passwordError = new FieldError("passwordForm", "password", ex.getMessage());
400  bindingResult.addError(passwordError);
401  } catch (UserNotFoundException ex) {
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 463 (System Information Leak: Internal)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.582

Source Details

Source: Read this.USERNAME_TAKEN_ERROR
From: com.microfocus.example.web.controllers.UserController.registerUser
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:463

```
460  this.setModelDefaults(model, null, "verify");
461  return "redirect:/user/verify?email="+u.getEmail()+"&status=new";
462  } catch (UsernameTakenException ex) {
463  log.error(USERNAME_TAKEN_ERROR);
464  FieldError usernameError = new FieldError("registerUserForm",
"username", ex.getMessage());
465  bindingResult.addError(usernameError);
466  } catch (EmailAddressTakenException ex) {
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: registerUser()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:463
Taint Flags: ARGS, ENVIRONMENT, PROPERTY

System Information Leak: Internal — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 463 (System Information Leak: Internal)

```
460  this.setModelDefaults(model, null, "verify");
461  return "redirect:/user/verify?email="+u.getEmail()+"&status=new";
462  } catch (UsernameTakenException ex) {
463  log.error(USERNAME_TAKEN_ERROR);
464  FieldError usernameError = new FieldError("registerUserForm", "username",
ex.getMessage());
465  bindingResult.addError(usernameError);
466  } catch (EmailAddressTakenException ex) {
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 516 (System Information Leak: Internal)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.635

Source Details

Source: java.lang.Throwable.getLocalizedMessage()
From: com.microfocus.example.web.controllers.UserController.verifyUser
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:516

```
513  redirectAttributes.addFlashAttribute("alertClass", "alert-danger");
514  }
515  } catch (UserNotFoundException ex) {
516  log.error("Could not find user '" + email + "' to verify: " +
ex.getLocalizedMessage());
517  redirectAttributes.addFlashAttribute("message", "The account being
verified does not exist. Please try registering again or contact support.");
518  redirectAttributes.addFlashAttribute("alertClass", "alert-danger");
519  }
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: verifyUser()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:516
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

System Information Leak: Internal — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 516 (System Information Leak: Internal)

```
513  redirectAttributes.addFlashAttribute("alertClass", "alert-danger");
514  }
515  } catch (UserNotFoundException ex) {
516  log.error("Could not find user '" + email + "' to verify: " + ex.getLocalizedMessage());
517  redirectAttributes.addFlashAttribute("message", "The account being verified does not
exist. Please try registering again or contact support.");
518  redirectAttributes.addFlashAttribute("alertClass", "alert-danger");
519  }
```

src/main/java/com/microfocus/example/web/controllers/DefaultController.java, line 131 (System Information Leak: Internal)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.635

Source Details

Source: java.lang.Throwable.getLocalizedMessage()
From: com.microfocus.example.web.controllers.DefaultController.otpLogin
File: src/main/java/com/microfocus/example/web/controllers/DefaultController.java:131

```
128  log.error(ex.getLocalizedMessage());
129  }
130  } catch (VerificationRequestFailedException ex) {
131  log.error(ex.getLocalizedMessage());
132  // TODO: handle
133  }
134  }
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: otpLogin()
File: src/main/java/com/microfocus/example/web/controllers/DefaultController.java:131
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

System Information Leak: Internal — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/DefaultController.java, line 131 (System Information Leak: Internal)

```
128  log.error(ex.getLocalizedMessage());
129  }
130  } catch (VerificationRequestFailedException ex) {
131  log.error(ex.getLocalizedMessage());
132  // TODO: handle
133  }
134  }
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 467 (System Information Leak: Internal)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.582

Source Details

Source: Read this.EMAIL_ADDRESS_TAKEN_ERROR
From: com.microfocus.example.web.controllers.UserController.registerUser
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:467

```
464  FieldError usernameError = new FieldError("registerUserForm",
"username", ex.getMessage());
465  bindingResult.addError(usernameError);
466  } catch (EmailAddressTakenException ex) {
467  log.error(EMAIL_ADDRESS_TAKEN_ERROR);
468  FieldError emailError = new FieldError("registerUserForm", "email",
ex.getMessage());
469  bindingResult.addError(emailError);
470  }
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: registerUser()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:467
Taint Flags: ARGS, ENVIRONMENT, PROPERTY

System Information Leak: Internal — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 467 (System Information Leak: Internal)

```
464  FieldError usernameError = new FieldError("registerUserForm", "username",
ex.getMessage());
465  bindingResult.addError(usernameError);
466  } catch (EmailAddressTakenException ex) {
467  log.error(EMAIL_ADDRESS_TAKEN_ERROR);
468  FieldError emailError = new FieldError("registerUserForm", "email", ex.getMessage());
469  bindingResult.addError(emailError);
470  }
```

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminUserController.java, line 155 (System Information Leak: Internal)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.582

Source Details

Source: Read this.AUTHENTICATION_ERROR
From: com.microfocus.example.web.controllers.admin.AdminUserController.userSavePassword
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminUserController.java:155

```
152  redirectAttributes.addFlashAttribute("alertClass", "alert-success");
153  return "redirect:/admin/users/" + userId;
154  } catch (InvalidPasswordException ex) {
155  log.error(AUTHENTICATION_ERROR);
156  FieldError passwordError = new FieldError("adminPasswordForm",
"password", ex.getMessage());
157  bindingResult.addError(passwordError);
158  } catch (UserNotFoundException ex) {
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: userSavePassword()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminUserController.java:155
Taint Flags: ARGS, ENVIRONMENT, PROPERTY

System Information Leak: Internal — Low

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminUserController.java, line 155 (System Information Leak: Internal)

```
152  redirectAttributes.addFlashAttribute("alertClass", "alert-success");
153  return "redirect:/admin/users/" + userId;
154  } catch (InvalidPasswordException ex) {
155  log.error(AUTHENTICATION_ERROR);
156  FieldError passwordError = new FieldError("adminPasswordForm", "password",
ex.getMessage());
157  bindingResult.addError(passwordError);
158  } catch (UserNotFoundException ex) {
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java, line 129 (System Information Leak: Internal)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.635

Source Details

Source: java.lang.Throwable.getMessage()
From: com.microfocus.example.web.controllers.admin.AdminDefaultController.runDbBackup
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java:129

```
126  try {
127  backUpId = AdminUtils.startDbBackup(backupForm.getProfile());
128  } catch (BackupException ignored) {
129  log.error(ignored.getMessage());
130  }
131  log.debug("Backup id: " + backUpId);
132  redirectAttributes.addFlashAttribute("message", "Database backup started
successfully.");
```

Sink Details

Sink: org.slf4j.Logger.error()
Enclosing Method: runDbBackup()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java:129
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

System Information Leak: Internal — Low

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java, line 129 (System Information Leak: Internal)

```
126  try {
127  backUpId = AdminUtils.startDbBackup(backupForm.getProfile());
128  } catch (BackupException ignored) {
129  log.error(ignored.getMessage());
130  }
131  log.debug("Backup id: " + backUpId);
132  redirectAttributes.addFlashAttribute("message", "Database backup started successfully.");
```

Trust Boundary Violation (52 issues)

Abstract

Commingling trusted and untrusted data in the same data structure encourages programmers to mistakenly trust unvalidated data.

Explanation

A trust boundary can be thought of as line drawn through a program. On one side of the line, data is untrusted. On the other side of the line, data is assumed to be trustworthy. The purpose of validation logic is to allow data to safely cross the trust boundary--to move from untrusted to trusted.

A trust boundary violation occurs when a program blurs the line between what is trusted and what is untrusted. The most common way to make this mistake is to allow trusted and untrusted data to commingle in the same data structure.

Example: The following Java code accepts an HTTP request and stores the `usrname` parameter in the HTTP session object before checking to ensure that the user has been authenticated.

```
usrname = request.getParameter("usrname");
if (session.getAttribute(ATTR_USR) != null) {
    session.setAttribute(ATTR_USR, usrname);
}
```

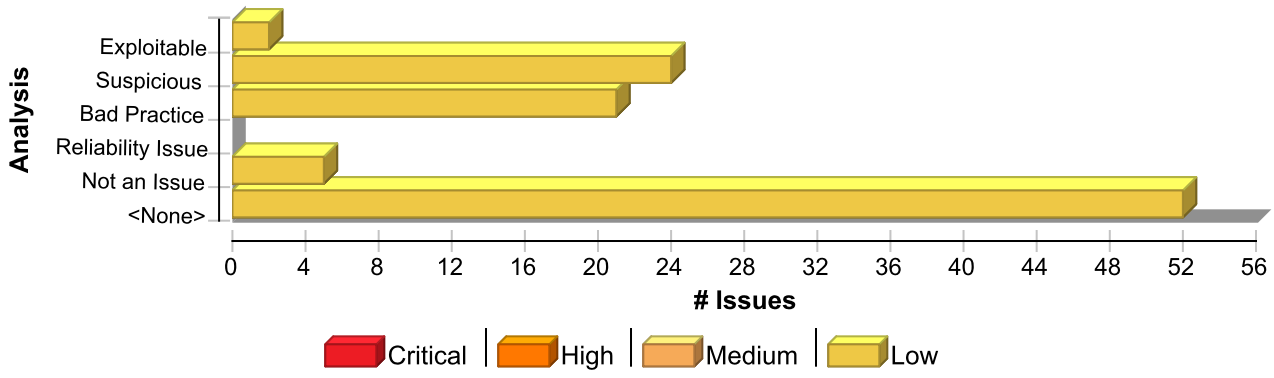
Without well-established and maintained trust boundaries, programmers will inevitably lose track of which pieces of data have been validated and which have not. This confusion eventually allows some data to be used without first being validated.

Recommendation

Define clear trust boundaries in the application. Do not use the same data structure to hold trusted data in some contexts and untrusted data in other contexts. Minimize the number of ways that data can move across a trust boundary.

Trust boundary violations sometimes occur when input needs to be built up over a series of user interactions before being processed. It may not be possible to do complete input validation until all of the data has arrived. In these situations, it is still important to maintain a trust boundary. The untrusted data should be built up in a single untrusted data structure, validated, and then moved into a trusted location.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Trust Boundary Violation	52	0	0	52
Total	**52**	**0**	**0**	**52**

Trust Boundary Violation — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/GlobalExceptionHandler.java, line 53 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.512

Source Details

Source: javax.servlet.http.HttpServletRequest.getRequestURL()
From: com.microfocus.example.config.handlers.GlobalExceptionHandler.handleAll
File: src/main/java/com/microfocus/example/config/handlers/GlobalExceptionHandler.java:53

```
50  // Otherwise setup and send the user to a default error-view.
51  ModelAndView mav = new ModelAndView();
52  mav.addObject("exception", ex);
53  mav.addObject("url", request.getRequestURL());
54  mav.setViewName(DEFAULT_ERROR_VIEW);
55  return mav;
56  }
```

Sink Details

Sink: org.springframework.web.servlet.ModelAndView.addObject()
Enclosing Method: handleAll()
File: src/main/java/com/microfocus/example/config/handlers/GlobalExceptionHandler.java:53
Taint Flags: NO_NEW_LINE, POORVALIDATION, URL_ENCODE, VALIDATED_CROSS_SITE_SCRIPTING_DOM, VALIDATED_CROSS_SITE_SCRIPTING_INTER_COMPONENT_COMMUNICATION, VALIDATED_CROSS_SITE_SCRIPTING_PERSISTENT, VALIDATED_CROSS_SITE_SCRIPTING_REFLECTED, VALIDATED_HTTP_PARAMETER_POLLUTION, WEB, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.config.handlers

src/main/java/com/microfocus/example/config/handlers/GlobalExceptionHandler.java, line 53 (Trust Boundary Violation)

```
50  // Otherwise setup and send the user to a default error-view.
51  ModelAndView mav = new ModelAndView();
52  mav.addObject("exception", ex);
53  mav.addObject("url", request.getRequestURL());
54  mav.setViewName(DEFAULT_ERROR_VIEW);
55  return mav;
56  }
```

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 153 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.606

Source Details

Source: org.springframework.data.repository.CrudRepository.findById()
From: com.microfocus.example.service.UserService.findUserById
File: src/main/java/com/microfocus/example/service/UserService.java:92

```
89  }
90
91  public Optional<User> findUserById(UUID id) {
92  return userRepository.findById(id);
93  }
94
95  public Optional<User> findUserByUsername(String username) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: userHome()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:153
Taint Flags: DATABASE, NUMBER, PRIMARY_KEY, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 153 (Trust Boundary Violation)

```
150  Optional<User> optionalUser = userService.findUserById(user.getId());
151  if (optionalUser.isPresent()) {
152  UserForm userForm = new UserForm(optionalUser.get());
153  model.addAttribute("username", userForm.getUsername());
154  model.addAttribute("fullname", userForm.getFirstName() + " " + userForm.getLastName());
155  model.addAttribute("userInfo", WebUtils.toString(user.getUserDetails()));
156  model.addAttribute("unreadMessageCount",
userService.getUserUnreadMessageCount(user.getId()));
```

src/main/java/com/microfocus/example/web/controllers/CartController.java, line 90 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Exploitable
AA_Prediction	Exploitable
AA_Confidence	0.931

Source Details

Source: org.springframework.data.repository.CrudRepository.findById()
From: com.microfocus.example.service.UserService.findUserById
File: src/main/java/com/microfocus/example/service/UserService.java:92

```
89  }
90
91  public Optional<User> findUserById(UUID id) {
92  return userRepository.findById(id);
93  }
94
95  public Optional<User> findUserByUsername(String username) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: checkout()
File: src/main/java/com/microfocus/example/web/controllers/CartController.java:90
Taint Flags: DATABASE, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/CartController.java, line 90 (Trust Boundary Violation)

```
87  User utmp = optionalUser.get();
88  OrderForm orderForm = new OrderForm();
89  orderForm.setUser(utmp);
90  model.addAttribute("orderForm", orderForm);
91  model.addAttribute("userInfo", WebUtils.toString(user.getUserDetails()));
92  } else {
93  model.addAttribute("message", "Internal error accessing user!");
```

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 112 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Not an Issue
AA_Prediction	Not an Issue
AA_Confidence	0.819

Source Details

Source: org.springframework.jdbc.core.JdbcTemplate.query()
From: com.microfocus.example.repository.ProductRepository.findAvailable
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:62

```
59  String sqlQuery = "select * from products" +
60  " where available = true " +
61  " LIMIT " + limit + " OFFSET " + offset;
62  return jdbcTemplate.query(sqlQuery, new ProductMapper());
63  }
64
65  public Optional<Product> findById(UUID id) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: index()
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:112
Taint Flags: DATABASE, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 112 (Trust Boundary Violation)

```
109  productService.setPageSize((limit == null ? defaultPageSize : limit));
110  List<Product> products = productService.getAllActiveProducts(0, keywords);
111  model.addAttribute("keywords", keywords);
112  model.addAttribute("products", products);
113  model.addAttribute("productCount", products.size());
114  model.addAttribute("productTotal", productService.count());
115  this.setModelDefaults(model, principal, "index");
```

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 99 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Not an Issue
AA_Prediction	Not an Issue
AA_Confidence	0.819

Source Details

Source: org.springframework.jdbc.core.JdbcTemplate.query()
From: com.microfocus.example.repository.ProductRepository.findAvailable
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:62

```
59  String sqlQuery = "select * from products" +
60  " where available = true " +
61  " LIMIT " + limit + " OFFSET " + offset;
62  return jdbcTemplate.query(sqlQuery, new ProductMapper());
63  }
64
65  public Optional<Product> findById(UUID id) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: firstaid()
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:99
Taint Flags: DATABASE, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 99 (Trust Boundary Violation)

```
96  productService.setPageSize((limit == null ? defaultPageSize : limit));
97  List<Product> products = productService.getAllActiveProducts(0, keywords);
98  model.addAttribute("keywords", keywords);
99  model.addAttribute("products", products);
100  model.addAttribute("productCount", products.size());
101  model.addAttribute("productTotal", productService.count());
102  this.setModelDefaults(model, principal, "index");
```

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 112 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Not an Issue
AA_Prediction	Not an Issue
AA_Confidence	0.819

Source Details

Source: org.springframework.jdbc.core.JdbcTemplate.query()
From: com.microfocus.example.repository.ProductRepository.findAvailableByKeywords
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:128

```
125  " OR lower(description) LIKE '%" + query + "%'" +
126  " AND available = true " +
127  " LIMIT " + limit + " OFFSET " + offset;
128  return jdbcTemplate.query(sqlQuery, new ProductMapper());
129  }
130
131  public List<Product> findAvailableByKeywordsFromProductName(String
keywords) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: index()
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:112
Taint Flags: DATABASE, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 112 (Trust Boundary Violation)

```
109  productService.setPageSize((limit == null ? defaultPageSize : limit));
110  List<Product> products = productService.getAllActiveProducts(0, keywords);
111  model.addAttribute("keywords", keywords);
112  model.addAttribute("products", products);
113  model.addAttribute("productCount", products.size());
114  model.addAttribute("productTotal", productService.count());
115  this.setModelDefaults(model, principal, "index");
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 193 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.657

Source Details

Source: org.springframework.data.repository.CrudRepository.findById()
From: com.microfocus.example.service.UserService.findUserById
File: src/main/java/com/microfocus/example/service/UserService.java:92

```
89  }
90
91  public Optional<User> findUserById(UUID id) {
92  return userRepository.findById(id);
93  }
94
95  public Optional<User> findUserByUsername(String username) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: userEditProfile()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:193
Taint Flags: DATABASE, NUMBER, PRIMARY_KEY, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 193 (Trust Boundary Violation)

```
190  Optional<User> optionalUser = userService.findUserById(user.getId());
191  if (optionalUser.isPresent()) {
192  UserForm userForm = new UserForm(optionalUser.get());
193  model.addAttribute("userForm", userForm);
194  model.addAttribute("userInfo", WebUtils.toString(user.getUserDetails()));
195  } else {
196  model.addAttribute("message", "Internal error accessing user!");
```

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 101 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.716

Source Details

Source: org.springframework.jdbc.core.JdbcTemplate.queryForObject()
From: com.microfocus.example.repository.ProductRepository.count
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:49

```
46
47  public int count() {
48  String sqlQuery = "select count(*) from products";
49  return jdbcTemplate.queryForObject(sqlQuery, Integer.class);
50  }
51
52  public List<Product> findAll(int offset, int limit) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: firstaid()
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:101
Taint Flags: DATABASE, NUMBER

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 101 (Trust Boundary Violation)

```
98  model.addAttribute("keywords", keywords);
99  model.addAttribute("products", products);
100  model.addAttribute("productCount", products.size());
101  model.addAttribute("productTotal", productService.count());
102  this.setModelDefaults(model, principal, "index");
103  return "products/firstaid";
104  }
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 174 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.657

Source Details

Source: org.springframework.data.repository.CrudRepository.findById()
From: com.microfocus.example.service.UserService.findUserById
File: src/main/java/com/microfocus/example/service/UserService.java:92

```
89  }
90
91  public Optional<User> findUserById(UUID id) {
92  return userRepository.findById(id);
93  }
94
95  public Optional<User> findUserByUsername(String username) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: userProfile()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:174
Taint Flags: DATABASE, NUMBER, PRIMARY_KEY, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 174 (Trust Boundary Violation)

```
171  Optional<User> optionalUser = userService.findUserById(user.getId());
172  if (optionalUser.isPresent()) {
173  UserForm userForm = new UserForm(optionalUser.get());
174  model.addAttribute("userForm", userForm);
175  model.addAttribute("userInfo", WebUtils.toString(user.getUserDetails()));
176  model.addAttribute("unreadMessageCount",
userService.getUserUnreadMessageCount(user.getId()));
177  } else {
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 335 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.624

Source Details

Source: org.springframework.data.repository.CrudRepository.findById()
From: com.microfocus.example.service.UserService.findOrderById
File: src/main/java/com/microfocus/example/service/UserService.java:438

```
435  }
436
437  public Optional<Order> findOrderById(UUID id) {
438  return orderRepository.findById(id);
439  }
440
441  public Order saveOrder(Order order) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: viewOrder()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:335
Taint Flags: DATABASE, NUMBER, PRIMARY_KEY, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 335 (Trust Boundary Violation)

```
332  return "user/orders/access-denied";
333  }
334  OrderForm orderForm = new OrderForm(optionalOrder.get());
335  model.addAttribute("orderForm", orderForm);
336  } else {
337  model.addAttribute("message", "Internal error accessing order!");
338  model.addAttribute("alertClass", "alert-danger");
```

src/main/java/com/microfocus/example/web/controllers/DefaultController.java, line 91 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Exploitable
AA_Prediction	Exploitable
AA_Confidence	0.957

Source Details

Source: Read this.message
From: com.microfocus.example.web.controllers.DefaultController.index
File: src/main/java/com/microfocus/example/web/controllers/DefaultController.java:91

```
88
89  @GetMapping("/")
90  public String index(Model model, Principal principal) {
91  model.addAttribute("message", message);
92  this.setModelDefaults(model, principal, "index");
93  return "index";
94  }
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: index()
File: src/main/java/com/microfocus/example/web/controllers/DefaultController.java:91
Taint Flags: ARGS, ENVIRONMENT, PROPERTY

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/DefaultController.java, line 91 (Trust Boundary Violation)

```
88
89  @GetMapping("/")
90  public String index(Model model, Principal principal) {
91  model.addAttribute("message", message);
92  this.setModelDefaults(model, principal, "index");
93  return "index";
94  }
```

src/main/java/com/microfocus/example/web/controllers/DefaultController.java, line 100 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.769

Source Details

Source: javax.servlet.http.HttpServletRequest.getHeader()
From: com.microfocus.example.web.controllers.DefaultController.login
File: src/main/java/com/microfocus/example/web/controllers/DefaultController.java:99

```
96  @GetMapping("/login")
97  public String login(HttpServletRequest request, Model model, Principal principal) {
98  HttpSession session = request.getSession(false);
99  String referer = (String) request.getHeader("referer");
100  session.setAttribute("loginReferer", referer);
101  this.setModelDefaults(model, principal, "login");
102  return "login";
```

Sink Details

Sink: javax.servlet.http.HttpSession.setAttribute()
Enclosing Method: login()
File: src/main/java/com/microfocus/example/web/controllers/DefaultController.java:100
Taint Flags: WEB, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/DefaultController.java, line 100 (Trust Boundary Violation)

```
97  public String login(HttpServletRequest request, Model model, Principal principal) {
98  HttpSession session = request.getSession(false);
99  String referer = (String) request.getHeader("referer");
100  session.setAttribute("loginReferer", referer);
101  this.setModelDefaults(model, principal, "login");
102  return "login";
103  }
```

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 123 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.648

Source Details

Source: org.springframework.jdbc.core.JdbcTemplate.query()
From: com.microfocus.example.repository.ProductRepository.findById
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:70

```
67  String query = id.toString();
68  String sqlQuery = "SELECT * FROM " + getTableName() +
69  " WHERE id = '" + query + "'";
70  result = jdbcTemplate.query(sqlQuery, new ProductMapper());
71  Optional<Product> optionalProduct = Optional.empty();
72  if (!result.isEmpty()) {
73  optionalProduct = Optional.of(result.get(0));
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: viewProduct()
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:123
Taint Flags: DATABASE, NON_STRING_PARAMETERIZED_TYPE, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 123 (Trust Boundary Violation)

```
120  public String viewProduct(@PathVariable("id") UUID productId, Model model, Principal
principal) {
121  Optional<Product> optionalProduct = productService.findProductById(productId);
122  if (optionalProduct.isPresent()) {
123  model.addAttribute("product", optionalProduct.get());
124  } else {
125  model.addAttribute("message", "Internal error accessing product!");
126  model.addAttribute("alertClass", "alert-danger");
```

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 99 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Not an Issue
AA_Prediction	Not an Issue
AA_Confidence	0.819

Source Details

Source: org.springframework.jdbc.core.JdbcTemplate.query()
From: com.microfocus.example.repository.ProductRepository.findAvailableByKeywords
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:128

```
125  " OR lower(description) LIKE '%" + query + "%'" +
126  " AND available = true " +
127  " LIMIT " + limit + " OFFSET " + offset;
128  return jdbcTemplate.query(sqlQuery, new ProductMapper());
129  }
130
131  public List<Product> findAvailableByKeywordsFromProductName(String
keywords) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: firstaid()
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:99
Taint Flags: DATABASE, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 99 (Trust Boundary Violation)

```
96  productService.setPageSize((limit == null ? defaultPageSize : limit));
97  List<Product> products = productService.getAllActiveProducts(0, keywords);
98  model.addAttribute("keywords", keywords);
99  model.addAttribute("products", products);
100  model.addAttribute("productCount", products.size());
101  model.addAttribute("productTotal", productService.count());
102  this.setModelDefaults(model, principal, "index");
```

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 114 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.716

Source Details

Source: org.springframework.jdbc.core.JdbcTemplate.queryForObject()
From: com.microfocus.example.repository.ProductRepository.count
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:49

```
46
47  public int count() {
48  String sqlQuery = "select count(*) from products";
49  return jdbcTemplate.queryForObject(sqlQuery, Integer.class);
50  }
51
52  public List<Product> findAll(int offset, int limit) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: index()
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:114
Taint Flags: DATABASE, NUMBER

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 114 (Trust Boundary Violation)

```
111  model.addAttribute("keywords", keywords);
112  model.addAttribute("products", products);
113  model.addAttribute("productCount", products.size());
114  model.addAttribute("productTotal", productService.count());
115  this.setModelDefaults(model, principal, "index");
116  return "products/index";
117  }
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 211 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.677

Source Details

Source: org.springframework.data.repository.CrudRepository.findById()
From: com.microfocus.example.service.UserService.findUserById
File: src/main/java/com/microfocus/example/service/UserService.java:92

```
89  }
90
91  public Optional<User> findUserById(UUID id) {
92  return userRepository.findById(id);
93  }
94
95  public Optional<User> findUserByUsername(String username) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: userChangePassword()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:211
Taint Flags: DATABASE, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 211 (Trust Boundary Violation)

```
208  Optional<User> optionalUser = userService.findUserById(user.getId());
209  if (optionalUser.isPresent()) {
210  PasswordForm passwordForm = new PasswordForm(optionalUser.get());
211  model.addAttribute("passwordForm", passwordForm);
212  model.addAttribute("userInfo", WebUtils.toString(user.getUserDetails()));
213  } else {
214  model.addAttribute("message", "Internal error accessing user!");
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 154 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.606

Source Details

Source: org.springframework.data.repository.CrudRepository.findById()
From: com.microfocus.example.service.UserService.findUserById
File: src/main/java/com/microfocus/example/service/UserService.java:92

```
89  }
90
91  public Optional<User> findUserById(UUID id) {
92  return userRepository.findById(id);
93  }
94
95  public Optional<User> findUserByUsername(String username) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: userHome()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:154
Taint Flags: DATABASE, NUMBER, PRIMARY_KEY, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 154 (Trust Boundary Violation)

```
151  if (optionalUser.isPresent()) {
152  UserForm userForm = new UserForm(optionalUser.get());
153  model.addAttribute("username", userForm.getUsername());
154  model.addAttribute("fullname", userForm.getFirstName() + " " + userForm.getLastName());
155  model.addAttribute("userInfo", WebUtils.toString(user.getUserDetails()));
156  model.addAttribute("unreadMessageCount",
userService.getUserUnreadMessageCount(user.getId()));
157  model.addAttribute("unshippedOrderCount",
userService.getUserUnshippedOrderCount(user.getId()));
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 274 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.662

Source Details

Source: org.springframework.data.repository.CrudRepository.findById()
From: com.microfocus.example.service.UserService.findMessageById
File: src/main/java/com/microfocus/example/service/UserService.java:374

```
371  }
372
373  public Optional<Message> findMessageById(UUID id) {
374  return messageRepository.findById(id);
375  }
376
377  public Message saveMessage(Message message) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: viewMessage()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:274
Taint Flags: DATABASE, NUMBER, PRIMARY_KEY, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 274 (Trust Boundary Violation)

```
271  return "user/messages/access-denied";
272  }
273  MessageForm messageForm = new MessageForm(optionalMessage.get());
274  model.addAttribute("messageForm", messageForm);
275  // mark messages as read
276  userService.markMessageAsReadById(messageId);
277  } else {
```

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminOrderController.java, line 126 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.624

Source Details

Source: org.springframework.data.repository.CrudRepository.findById()
From: com.microfocus.example.service.ProductService.findOrderById
File: src/main/java/com/microfocus/example/service/ProductService.java:301

```
298  }
299
300  public Optional<Order> findOrderById(UUID id) {
301  return orderRepository.findById(id);
302  }
303
304  public Optional<Order> findOrderByNumber(String number) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: deleteOrder()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminOrderController.java:126
Taint Flags: DATABASE, NUMBER, PRIMARY_KEY, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminOrderController.java, line 126 (Trust Boundary Violation)

```
123  Optional<Order> optionalOrder = productService.findOrderById(orderId);
124  if (optionalOrder.isPresent()) {
125  AdminOrderForm adminOrderForm = new AdminOrderForm(optionalOrder.get());
126  model.addAttribute("adminOrderForm", adminOrderForm);
127  } else {
128  model.addAttribute("message", "Internal error accessing order!");
129  model.addAttribute("alertClass", "alert-danger");
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminOrderController.java, line 63 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.556

Source Details

Source: org.springframework.data.jpa.repository.JpaRepository.findAll()
From: com.microfocus.example.service.ProductService.getAllOrders
File: src/main/java/com/microfocus/example/service/ProductService.java:308

```
305  return orderRepository.findByNumber(number);
306  }
307
308  public List<Order> getAllOrders() { return orderRepository.findAll(); }
309
310  public List<Order> getAllOrders(Integer offset, String keywords) {
311  if (keywords != null && !keywords.isEmpty()) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: listOrders()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminOrderController.java:63
Taint Flags: DATABASE, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminOrderController.java, line 63 (Trust Boundary Violation)

```
60  @GetMapping(value = {"", "/"})
61  public String listOrders(Model model, Principal principal) {
62  List<Order> orders = productService.getAllOrders();
63  model.addAttribute("orders", orders);
64  this.setModelDefaults(model, principal, "Admin", "orders");
65  return "admin/orders/index";
66  }
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminReviewController.java, line 90 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.662

Source Details

Source: org.springframework.data.repository.CrudRepository.findById()
From: com.microfocus.example.service.ProductService.findReviewById
File: src/main/java/com/microfocus/example/service/ProductService.java:200

```
197  //
198
199  public Optional<Review> findReviewById(UUID id) {
200  return reviewRepository.findById(id);
201  }
202
203  public List<Review> findReviewsByProductId(UUID productId) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: editReview()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminReviewController.java:90
Taint Flags: DATABASE, NUMBER, PRIMARY_KEY, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminReviewController.java, line 90 (Trust Boundary Violation)

```
87  Optional<Review> optionalReview = productService.findReviewById(reviewId);
88  if (optionalReview.isPresent()) {
89  AdminReviewForm adminReviewForm = new AdminReviewForm(optionalReview.get());
90  model.addAttribute("adminReviewForm", adminReviewForm);
91  } else {
92  model.addAttribute("message", "Internal error accessing review!");
93  model.addAttribute("alertClass", "alert-danger");
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminUserController.java, line 80 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.657

Source Details

Source: org.springframework.data.repository.CrudRepository.findById()
From: com.microfocus.example.service.UserService.findUserById
File: src/main/java/com/microfocus/example/service/UserService.java:92

```
89  }
90
91  public Optional<User> findUserById(UUID id) {
92  return userRepository.findById(id);
93  }
94
95  public Optional<User> findUserByUsername(String username) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: viewUser()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminUserController.java:80
Taint Flags: DATABASE, NUMBER, PRIMARY_KEY, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminUserController.java, line 80 (Trust Boundary Violation)

```
77  Optional<User> optionalUser = userService.findUserById(userId);
78  if (optionalUser.isPresent()) {
79  UserForm userForm = new UserForm(optionalUser.get());
80  model.addAttribute("userForm", userForm);
81  } else {
82  model.addAttribute("message", "Internal error accessing user!");
83  model.addAttribute("alertClass", "alert-danger");
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminOrderController.java, line 63 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.556

Source Details

Source: org.springframework.data.jpa.repository.JpaRepository.findAll()
From: com.microfocus.example.service.ProductService.getAllOrders
File: src/main/java/com/microfocus/example/service/ProductService.java:308

```
305  return orderRepository.findByNumber(number);
306  }
307
308  public List<Order> getAllOrders() { return orderRepository.findAll(); }
309
310  public List<Order> getAllOrders(Integer offset, String keywords) {
311  if (keywords != null && !keywords.isEmpty()) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: listOrders()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminOrderController.java:63
Taint Flags: DATABASE, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminOrderController.java, line 63 (Trust Boundary Violation)

```
60  @GetMapping(value = {"", "/"})
61  public String listOrders(Model model, Principal principal) {
62  List<Order> orders = productService.getAllOrders();
63  model.addAttribute("orders", orders);
64  this.setModelDefaults(model, principal, "Admin", "orders");
65  return "admin/orders/index";
66  }
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminUserController.java, line 69 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.556

Source Details

Source: org.springframework.data.jpa.repository.JpaRepository.findAll()
From: com.microfocus.example.service.UserService.getAllUsers
File: src/main/java/com/microfocus/example/service/UserService.java:102

```
99  public Optional<User> findUserByEmail(String email) { return
userRepository.findUserByEmail(email); }
100
101  public List<User> getAllUsers() {
102  return (List<User>) userRepository.findAll();
103  }
104
105  public List<User> getAllUsers(Integer offset, String keywords) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: listUsers()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminUserController.java:69
Taint Flags: DATABASE, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminUserController.java, line 69 (Trust Boundary Violation)

```
66  @GetMapping(value = {"", "/"})
67  public String listUsers(Model model, Principal principal) {
68  List<User> users = userService.getAllUsers();
69  model.addAttribute("users", users);
70  this.setModelDefaults(model, principal, "Admin", "users");
71  return "admin/users/index";
72  }
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminUserController.java, line 96 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.657

Source Details

Source: org.springframework.data.repository.CrudRepository.findById()
From: com.microfocus.example.service.UserService.findUserById
File: src/main/java/com/microfocus/example/service/UserService.java:92

```
89  }
90
91  public Optional<User> findUserById(UUID id) {
92  return userRepository.findById(id);
93  }
94
95  public Optional<User> findUserByUsername(String username) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: userEditProfile()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminUserController.java:96
Taint Flags: DATABASE, NUMBER, PRIMARY_KEY, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminUserController.java, line 96 (Trust Boundary Violation)

```
93  Optional<User> optionalUser = userService.findUserById(userId);
94  if (optionalUser.isPresent()) {
95  AdminUserForm adminUserForm = new AdminUserForm(optionalUser.get());
96  model.addAttribute("adminUserForm", adminUserForm);
97  } else {
98  model.addAttribute("message", "Internal error accessing user!");
99  model.addAttribute("alertClass", "alert-danger");
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminReviewController.java, line 126 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.662

Source Details

Source: org.springframework.data.repository.CrudRepository.findById()
From: com.microfocus.example.service.ProductService.findReviewById
File: src/main/java/com/microfocus/example/service/ProductService.java:200

```
197  //
198
199  public Optional<Review> findReviewById(UUID id) {
200  return reviewRepository.findById(id);
201  }
202
203  public List<Review> findReviewsByProductId(UUID productId) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: deleteReview()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminReviewController.java:126
Taint Flags: DATABASE, NUMBER, PRIMARY_KEY, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminReviewController.java, line 126 (Trust Boundary Violation)

```
123  Optional<Review> optionalReview = productService.findReviewById(reviewId);
124  if (optionalReview.isPresent()) {
125  AdminReviewForm adminReviewForm = new AdminReviewForm(optionalReview.get());
126  model.addAttribute("adminReviewForm", adminReviewForm);
127  } else {
128  model.addAttribute("message", "Internal error accessing review!");
129  model.addAttribute("alertClass", "alert-danger");
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminProductController.java, line 127 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.531

Source Details

Source: org.springframework.jdbc.core.JdbcTemplate.query()
From: com.microfocus.example.repository.ProductRepository.findById
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:70

```
67  String query = id.toString();
68  String sqlQuery = "SELECT * FROM " + getTableName() +
69  " WHERE id = '" + query + "'";
70  result = jdbcTemplate.query(sqlQuery, new ProductMapper());
71  Optional<Product> optionalProduct = Optional.empty();
72  if (!result.isEmpty()) {
73  optionalProduct = Optional.of(result.get(0));
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: productDelete()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminProductController.java:127
Taint Flags: DATABASE, NON_STRING_PARAMETERIZED_TYPE, NUMBER, PRIMARY_KEY, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminProductController.java, line 127 (Trust Boundary Violation)

```
124  Optional<Product> optionalProduct = productService.findProductById(productId);
125  if (optionalProduct.isPresent()) {
126  AdminProductForm adminProductForm = new AdminProductForm(optionalProduct.get());
127  model.addAttribute("adminProductForm", adminProductForm);
128  } else {
129  model.addAttribute("message", "Internal error accessing product!");
130  model.addAttribute("alertClass", "alert-danger");
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminOrderController.java, line 74 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.624

Source Details

Source: org.springframework.data.repository.CrudRepository.findById()
From: com.microfocus.example.service.ProductService.findOrderById
File: src/main/java/com/microfocus/example/service/ProductService.java:301

```
298  }
299
300  public Optional<Order> findOrderById(UUID id) {
301  return orderRepository.findById(id);
302  }
303
304  public Optional<Order> findOrderByNumber(String number) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: viewOrder()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminOrderController.java:74
Taint Flags: DATABASE, NUMBER, PRIMARY_KEY, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminOrderController.java, line 74 (Trust Boundary Violation)

```
71  Optional<Order> optionalOrder = productService.findOrderById(orderId);
72  if (optionalOrder.isPresent()) {
73  AdminOrderForm adminOrderForm = new AdminOrderForm(optionalOrder.get());
74  model.addAttribute("adminOrderForm", adminOrderForm);
75  } else {
76  model.addAttribute("message", "Internal error accessing order!");
77  model.addAttribute("alertClass", "alert-danger");
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminProductController.java, line 75 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.531

Source Details

Source: org.springframework.jdbc.core.JdbcTemplate.query()
From: com.microfocus.example.repository.ProductRepository.findById
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:70

```
67  String query = id.toString();
68  String sqlQuery = "SELECT * FROM " + getTableName() +
69  " WHERE id = '" + query + "'";
70  result = jdbcTemplate.query(sqlQuery, new ProductMapper());
71  Optional<Product> optionalProduct = Optional.empty();
72  if (!result.isEmpty()) {
73  optionalProduct = Optional.of(result.get(0));
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: viewProduct()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminProductController.java:75
Taint Flags: DATABASE, NON_STRING_PARAMETERIZED_TYPE, NUMBER, PRIMARY_KEY, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminProductController.java, line 75 (Trust Boundary Violation)

```
72  Optional<Product> optionalProduct = productService.findProductById(productId);
73  if (optionalProduct.isPresent()) {
74  AdminProductForm adminProductForm = new AdminProductForm(optionalProduct.get());
75  model.addAttribute("adminProductForm", adminProductForm);
76  } else {
77  model.addAttribute("message", "Internal error accessing product!");
78  model.addAttribute("alertClass", "alert-danger");
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminReviewController.java, line 63 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.556

Source Details

Source: org.springframework.data.jpa.repository.JpaRepository.findAll()
From: com.microfocus.example.service.ProductService.getReviews
File: src/main/java/com/microfocus/example/service/ProductService.java:211

```
208  return reviewRepository.findByUserId(userId);
209  }
210
211  public List<Review> getReviews() { return reviewRepository.findAll(); }
212
213  public List<Review> getReviews(Integer offset, String keywords) {
214  if (keywords != null && !keywords.isEmpty()) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: listReviews()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminReviewController.java:63
Taint Flags: DATABASE, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminReviewController.java, line 63 (Trust Boundary Violation)

```
60  @GetMapping(value = {"", "/"})
61  public String listReviews(Model model, Principal principal) {
62  List<Review> reviews = productService.getReviews();
63  model.addAttribute("reviews", reviews);
64  this.setModelDefaults(model, principal, "Admin", "reviews");
65  return "admin/reviews/index";
66  }
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminProductController.java, line 64 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Not an Issue
AA_Prediction	Not an Issue
AA_Confidence	0.819

Source Details

Source: org.springframework.jdbc.core.JdbcTemplate.query()
From: com.microfocus.example.repository.ProductRepository.findAll
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:55

```
52  public List<Product> findAll(int offset, int limit) {
53  String sqlQuery = "select * from products" +
54  " LIMIT " + limit + " OFFSET " + offset;
55  return jdbcTemplate.query(sqlQuery, new ProductMapper());
56  }
57
58  public List<Product> findAvailable(int offset, int limit) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: listProducts()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminProductController.java:64
Taint Flags: DATABASE, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminProductController.java, line 64 (Trust Boundary Violation)

```
61  @GetMapping(value = {"", "/"})
62  public String listProducts(Model model, Principal principal) {
63  List<Product> products = productService.getAllProducts();
64  model.addAttribute("products", products);
65  this.setModelDefaults(model, principal, "Admin", "products");
66  return "admin/products/index";
67  }
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminOrderController.java, line 90 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.624

Source Details

Source: org.springframework.data.repository.CrudRepository.findById()
From: com.microfocus.example.service.ProductService.findOrderById
File: src/main/java/com/microfocus/example/service/ProductService.java:301

```
298  }
299
300  public Optional<Order> findOrderById(UUID id) {
301  return orderRepository.findById(id);
302  }
303
304  public Optional<Order> findOrderByNumber(String number) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: editOrder()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminOrderController.java:90
Taint Flags: DATABASE, NUMBER, PRIMARY_KEY, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminOrderController.java, line 90 (Trust Boundary Violation)

```
87  Optional<Order> optionalOrder = productService.findOrderById(orderId);
88  if (optionalOrder.isPresent()) {
89  AdminOrderForm adminOrderForm = new AdminOrderForm(optionalOrder.get());
90  model.addAttribute("adminOrderForm", adminOrderForm);
91  } else {
92  model.addAttribute("message", "Internal error accessing order!");
93  model.addAttribute("alertClass", "alert-danger");
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminMessageController.java, line 55 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.556

Source Details

Source: org.springframework.data.jpa.repository.JpaRepository.findAll()
From: com.microfocus.example.service.UserService.getAllMessages
File: src/main/java/com/microfocus/example/service/UserService.java:354

```
351  //
352
353  public List<Message> getAllMessages() {
354  return messageRepository.findAll();
355  }
356
357  public long getUserMessageCount(UUID userId) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: adminMessages()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminMessageController.java:55
Taint Flags: DATABASE, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminMessageController.java, line 55 (Trust Boundary Violation)

```
52  @GetMapping(value = {"", "/"})
53  public String adminMessages(Model model, Principal principal) {
54  List<Message> messages = userService.getAllMessages();
55  model.addAttribute("messages", messages);
56  this.setModelDefaults(model, principal, "Admin", "messages");
57  return "admin/messages/index";
58  }
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminUserController.java, line 173 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.657

Source Details

Source: org.springframework.data.repository.CrudRepository.findById()
From: com.microfocus.example.service.UserService.findUserById
File: src/main/java/com/microfocus/example/service/UserService.java:92

```
89  }
90
91  public Optional<User> findUserById(UUID id) {
92  return userRepository.findById(id);
93  }
94
95  public Optional<User> findUserByUsername(String username) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: userDelete()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminUserController.java:173
Taint Flags: DATABASE, NUMBER, PRIMARY_KEY, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminUserController.java, line 173 (Trust Boundary Violation)

```
170  Optional<User> optionalUser = userService.findUserById(userId);
171  if (optionalUser.isPresent()) {
172  AdminUserForm adminUserForm = new AdminUserForm(optionalUser.get());
173  model.addAttribute("adminUserForm", adminUserForm);
174  } else {
175  model.addAttribute("message", "Internal error accessing user!");
176  model.addAttribute("alertClass", "alert-danger");
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminMessageController.java, line 55 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.556

Source Details

Source: org.springframework.data.jpa.repository.JpaRepository.findAll()
From: com.microfocus.example.service.UserService.getAllMessages
File: src/main/java/com/microfocus/example/service/UserService.java:354

```
351  //
352
353  public List<Message> getAllMessages() {
354  return messageRepository.findAll();
355  }
356
357  public long getUserMessageCount(UUID userId) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: adminMessages()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminMessageController.java:55
Taint Flags: DATABASE, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminMessageController.java, line 55 (Trust Boundary Violation)

```
52  @GetMapping(value = {"", "/"})
53  public String adminMessages(Model model, Principal principal) {
54  List<Message> messages = userService.getAllMessages();
55  model.addAttribute("messages", messages);
56  this.setModelDefaults(model, principal, "Admin", "messages");
57  return "admin/messages/index";
58  }
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminProductController.java, line 91 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.531

Source Details

Source: org.springframework.jdbc.core.JdbcTemplate.query()
From: com.microfocus.example.repository.ProductRepository.findById
File: src/main/java/com/microfocus/example/repository/ProductRepository.java:70

```
67  String query = id.toString();
68  String sqlQuery = "SELECT * FROM " + getTableName() +
69  " WHERE id = '" + query + "'";
70  result = jdbcTemplate.query(sqlQuery, new ProductMapper());
71  Optional<Product> optionalProduct = Optional.empty();
72  if (!result.isEmpty()) {
73  optionalProduct = Optional.of(result.get(0));
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: productEdit()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminProductController.java:91
Taint Flags: DATABASE, NON_STRING_PARAMETERIZED_TYPE, NUMBER, PRIMARY_KEY, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminProductController.java, line 91 (Trust Boundary Violation)

```
88  Optional<Product> optionalProduct = productService.findProductById(productId);
89  if (optionalProduct.isPresent()) {
90  AdminProductForm adminProductForm = new AdminProductForm(optionalProduct.get());
91  model.addAttribute("adminProductForm", adminProductForm);
92  } else {
93  model.addAttribute("message", "Internal error accessing product!");
94  model.addAttribute("alertClass", "alert-danger");
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminUserController.java, line 132 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.62

Source Details

Source: org.springframework.data.repository.CrudRepository.findById()
From: com.microfocus.example.service.UserService.findUserById
File: src/main/java/com/microfocus/example/service/UserService.java:92

```
89  }
90
91  public Optional<User> findUserById(UUID id) {
92  return userRepository.findById(id);
93  }
94
95  public Optional<User> findUserByUsername(String username) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: userChangePassword()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminUserController.java:132
Taint Flags: DATABASE, PRIMARY_KEY, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminUserController.java, line 132 (Trust Boundary Violation)

```
129  Optional<User> optionalUser = userService.findUserById(userId);
130  if (optionalUser.isPresent()) {
131  AdminPasswordForm adminPasswordForm = new AdminPasswordForm(optionalUser.get());
132  model.addAttribute("adminPasswordForm", adminPasswordForm);
133  } else {
134  model.addAttribute("message", "Internal error accessing user!");
135  model.addAttribute("alertClass", "alert-danger");
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminUserController.java, line 69 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.556

Source Details

Source: org.springframework.data.jpa.repository.JpaRepository.findAll()
From: com.microfocus.example.service.UserService.getAllUsers
File: src/main/java/com/microfocus/example/service/UserService.java:102

```
99  public Optional<User> findUserByEmail(String email) { return
userRepository.findUserByEmail(email); }
100
101  public List<User> getAllUsers() {
102  return (List<User>) userRepository.findAll();
103  }
104
105  public List<User> getAllUsers(Integer offset, String keywords) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: listUsers()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminUserController.java:69
Taint Flags: DATABASE, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminUserController.java, line 69 (Trust Boundary Violation)

```
66  @GetMapping(value = {"", "/"})
67  public String listUsers(Model model, Principal principal) {
68  List<User> users = userService.getAllUsers();
69  model.addAttribute("users", users);
70  this.setModelDefaults(model, principal, "Admin", "users");
71  return "admin/users/index";
72  }
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminReviewController.java, line 63 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.556

Source Details

Source: org.springframework.data.jpa.repository.JpaRepository.findAll()
From: com.microfocus.example.service.ProductService.getReviews
File: src/main/java/com/microfocus/example/service/ProductService.java:211

```
208  return reviewRepository.findByUserId(userId);
209  }
210
211  public List<Review> getReviews() { return reviewRepository.findAll(); }
212
213  public List<Review> getReviews(Integer offset, String keywords) {
214  if (keywords != null && !keywords.isEmpty()) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: listReviews()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminReviewController.java:63
Taint Flags: DATABASE, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminReviewController.java, line 63 (Trust Boundary Violation)

```
60  @GetMapping(value = {"", "/"})
61  public String listReviews(Model model, Principal principal) {
62  List<Review> reviews = productService.getReviews();
63  model.addAttribute("reviews", reviews);
64  this.setModelDefaults(model, principal, "Admin", "reviews");
65  return "admin/reviews/index";
66  }
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminMessageController.java, line 66 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.662

Source Details

Source: org.springframework.data.repository.CrudRepository.findById()
From: com.microfocus.example.service.UserService.findMessageById
File: src/main/java/com/microfocus/example/service/UserService.java:374

```
371  }
372
373  public Optional<Message> findMessageById(UUID id) {
374  return messageRepository.findById(id);
375  }
376
377  public Message saveMessage(Message message) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: viewMessage()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminMessageController.java:66
Taint Flags: DATABASE, NUMBER, PRIMARY_KEY, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminMessageController.java, line 66 (Trust Boundary Violation)

```
63  Optional<Message> optionalMessage = userService.findMessageById(messageId);
64  if (optionalMessage.isPresent()) {
65  MessageForm messageForm = new MessageForm(optionalMessage.get());
66  model.addAttribute("messageForm", messageForm);
67  } else {
68  model.addAttribute("message", "Internal error accessing message!");
69  model.addAttribute("alertClass", "alert-danger");
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminReviewController.java, line 74 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.662

Source Details

Source: org.springframework.data.repository.CrudRepository.findById()
From: com.microfocus.example.service.ProductService.findReviewById
File: src/main/java/com/microfocus/example/service/ProductService.java:200

```
197  //
198
199  public Optional<Review> findReviewById(UUID id) {
200  return reviewRepository.findById(id);
201  }
202
203  public List<Review> findReviewsByProductId(UUID productId) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: viewRevier()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminReviewController.java:74
Taint Flags: DATABASE, NUMBER, PRIMARY_KEY, XSS

Trust Boundary Violation — Low

Package: com.microfocus.example.web.controllers.admin

src/main/java/com/microfocus/example/web/controllers/admin/AdminReviewController.java, line 74 (Trust Boundary Violation)

```
71  Optional<Review> optionalReview = productService.findReviewById(reviewId);
72  if (optionalReview.isPresent()) {
73  AdminReviewForm adminReviewForm = new AdminReviewForm(optionalReview.get());
74  model.addAttribute("adminReviewForm", adminReviewForm);
75  } else {
76  model.addAttribute("message", "Internal error accessing review!");
77  model.addAttribute("alertClass", "alert-danger");
```

URL: null

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 491 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.623

Source Details

Source: verifyUser(0)
From: com.microfocus.example.web.controllers.UserController.verifyUser
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:477
URL: null

```
474  }
475
476  @GetMapping("/verify")
477  public String verifyUser(@RequestParam("email") Optional<String>
usersEmail,
478  @RequestParam("code") Optional<String> verificationCode,
479  @RequestParam("status") Optional<String> statusCode,
480  RedirectAttributes redirectAttributes,
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: verifyUser()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:491
Taint Flags: WEB, XSS

Trust Boundary Violation — Low

URL: null

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 491 (Trust Boundary Violation)

```
488  model.addAttribute("message", "Your registration details have been stored. Please check
your email to verify your details.");
489  model.addAttribute("alertClass", "alert-success");
490  VerifyUserForm verifyUserForm = new VerifyUserForm(usersEmail, verificationCode);
491  model.addAttribute("verifyUserForm", verifyUserForm);
492  this.setModelDefaults(model, null, "verify");
493  return "user/verify";
494  } else if ((email == null || email.isEmpty()) || (code == null || code.isEmpty())) {
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 499 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.623

Source Details

Source: verifyUser(0)
From: com.microfocus.example.web.controllers.UserController.verifyUser
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:477
URL: null

```
474  }
475
476  @GetMapping("/verify")
477  public String verifyUser(@RequestParam("email") Optional<String>
usersEmail,
478  @RequestParam("code") Optional<String> verificationCode,
479  @RequestParam("status") Optional<String> statusCode,
480  RedirectAttributes redirectAttributes,
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: verifyUser()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:499
Taint Flags: WEB, XSS

Trust Boundary Violation — Low

URL: null

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 499 (Trust Boundary Violation)

```
496  model.addAttribute("message", "You need to supply both an email address and verification code.");
497  model.addAttribute("alertClass", "alert-danger");
498  VerifyUserForm verifyUserForm = new VerifyUserForm(usersEmail, verificationCode);
499  model.addAttribute("verifyUserForm", verifyUserForm);
500  this.setModelDefaults(model, null, "verify");
501  return "user/verify";
502  } else {
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 491 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.623

Source Details

Source: verifyUser(1)
From: com.microfocus.example.web.controllers.UserController.verifyUser
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:478
URL: null

```
475
476  @GetMapping("/verify")
477  public String verifyUser(@RequestParam("email") Optional<String> usersEmail,
478  @RequestParam("code") Optional<String> verificationCode,
479  @RequestParam("status") Optional<String> statusCode,
480  RedirectAttributes redirectAttributes,
481  Model model) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: verifyUser()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:491
Taint Flags: WEB, XSS

Trust Boundary Violation — Low

URL: null

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 491 (Trust Boundary Violation)

```
488  model.addAttribute("message", "Your registration details have been stored. Please check
your email to verify your details.");
489  model.addAttribute("alertClass", "alert-success");
490  VerifyUserForm verifyUserForm = new VerifyUserForm(usersEmail, verificationCode);
491  model.addAttribute("verifyUserForm", verifyUserForm);
492  this.setModelDefaults(model, null, "verify");
493  return "user/verify";
494  } else if ((email == null || email.isEmpty()) || (code == null || code.isEmpty())) {
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 499 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.623

Source Details

Source: verifyUser(1)
From: com.microfocus.example.web.controllers.UserController.verifyUser
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:478
URL: null

```
475
476  @GetMapping("/verify")
477  public String verifyUser(@RequestParam("email") Optional<String>
usersEmail,
478  @RequestParam("code") Optional<String> verificationCode,
479  @RequestParam("status") Optional<String> statusCode,
480  RedirectAttributes redirectAttributes,
481  Model model) {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: verifyUser()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:499
Taint Flags: WEB, XSS

Trust Boundary Violation — Low

URL: null

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 499 (Trust Boundary Violation)

```
496  model.addAttribute("message", "You need to supply both an email address and verification code.");
497  model.addAttribute("alertClass", "alert-danger");
498  VerifyUserForm verifyUserForm = new VerifyUserForm(usersEmail, verificationCode);
499  model.addAttribute("verifyUserForm", verifyUserForm);
500  this.setModelDefaults(model, null, "verify");
501  return "user/verify";
502  } else {
```

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 111 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.674

Source Details

Source: index(1)
From: com.microfocus.example.web.controllers.ProductController.index
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:107
URL: null

```
104  }
105
106  @GetMapping(value = {"", "/"})
107  public String index(Model model, @Param("keywords") String keywords, @Param("limit") Integer limit, Principal principal) {
108  log.debug("Searching for products using keywords: " + ((keywords == null || keywords.isEmpty()) ? "none" : keywords));
109  productService.setPageSize((limit == null ? defaultPageSize : limit));
110  List<Product> products = productService.getAllActiveProducts(0, keywords);
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: index()
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:111
Taint Flags: WEB, XSS

Trust Boundary Violation — Low

URL: null

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 111 (Trust Boundary Violation)

```
108  log.debug("Searching for products using keywords: " + ((keywords == null ||
keywords.isEmpty()) ? "none" : keywords));
109  productService.setPageSize((limit == null ? defaultPageSize : limit));
110  List<Product> products = productService.getAllActiveProducts(0, keywords);
111  model.addAttribute("keywords", keywords);
112  model.addAttribute("products", products);
113  model.addAttribute("productCount", products.size());
114  model.addAttribute("productTotal", productService.count());
```

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 98 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.674

Source Details

Source: firstaid(1)
From: com.microfocus.example.web.controllers.ProductController.firstaid
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:94
URL: null

```
91   }
92
93   @GetMapping("/firstaid")
94   public String firstaid(Model model, @Param("keywords") String keywords,
@Param("limit") Integer limit, Principal principal) {
95   log.debug("Searching for products using keywords: " + ((keywords == null
|| keywords.isEmpty()) ? "none" : keywords));
96   productService.setPageSize((limit == null ? defaultPageSize : limit));
97   List<Product> products = productService.getAllActiveProducts(0, keywords);
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: firstaid()
File: src/main/java/com/microfocus/example/web/controllers/ProductController.java:98
Taint Flags: WEB, XSS

Trust Boundary Violation — Low

URL: null

src/main/java/com/microfocus/example/web/controllers/ProductController.java, line 98 (Trust Boundary Violation)

```
95  log.debug("Searching for products using keywords: " + ((keywords == null ||
keywords.isEmpty()) ? "none" : keywords));
96  productService.setPageSize((limit == null ? defaultPageSize : limit));
97  List<Product> products = productService.getAllActiveProducts(0, keywords);
98  model.addAttribute("keywords", keywords);
99  model.addAttribute("products", products);
100  model.addAttribute("productCount", products.size());
101  model.addAttribute("productTotal", productService.count());
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java, line 156 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.61

Source Details

Source: executeCommandShell(0)
From: com.microfocus.example.web.controllers.admin.AdminDefaultController.executeCommandShell
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java:162
URL: null

```
159  }
160
161  @PostMapping("/command-shell")
162  public String executeCommandShell(@RequestParam("cmdshell") String cmd,
163  RedirectAttributes redirectAttributes) {
164
165  this.thRCECMD = cmd;
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: getCommandShell()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java:156
Taint Flags: WEB, XSS

Trust Boundary Violation — Low

URL: null

src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java, line 156 (Trust Boundary Violation)

```
153  if (Objects.nonNull(this.thRCECMD) && this.thRCECMD.length() > 2) {
154  cmdWrapper = String.format("T (java.lang.Runtime).getRuntime().exec('%s')",
this.thRCECMD);
155  }
156  model.addAttribute("shellcmd", cmdWrapper);
157  model.addAttribute("usercmd", this.thRCECMD);
158  return "admin/command-shell";
159  }
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java, line 187 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.51

Source Details

Source: ssrfExploit(1)
From: com.microfocus.example.web.controllers.admin.AdminDefaultController.ssrfExploit
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java:172
URL: null

```
169  }
170
171  @GetMapping("/log")
172  public String ssrfExploit(Model model, @Param("val") String val) {
173  int intVal = -1;
174  String strLog = "";
175  try {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: ssrfExploit()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java:187
Taint Flags: NUMBER, WEB, XSS

Trust Boundary Violation — Low

URL: null

src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java, line 187 (Trust Boundary Violation)

```
184
185  model.addAttribute("val", val);
186  model.addAttribute("intval", intVal);
187  model.addAttribute("logwritten", strLog);
188
189  return "admin/log";
190  }
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java, line 185 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.674

Source Details

Source: ssrfExploit(1)
From: com.microfocus.example.web.controllers.admin.AdminDefaultController.ssrfExploit
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java:172
URL: null

```
169  }
170
171  @GetMapping("/log")
172  public String ssrfExploit(Model model, @Param("val") String val) {
173  int intVal = -1;
174  String strLog = "";
175  try {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: ssrfExploit()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java:185
Taint Flags: WEB, XSS

Trust Boundary Violation — Low

URL: null

src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java, line 185 (Trust Boundary Violation)

```
182  log.info("Failed to parse val = " + val);
183  }
184
185  model.addAttribute("val", val);
186  model.addAttribute("intval", intVal);
187  model.addAttribute("logwritten", strLog);
188
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java, line 186 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.5

Source Details

Source: ssrfExploit(1)
From: com.microfocus.example.web.controllers.admin.AdminDefaultController.ssrfExploit
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java:172
URL: null

```
169  }
170
171  @GetMapping("/log")
172  public String ssrfExploit(Model model, @Param("val") String val) {
173  int intVal = -1;
174  String strLog = "";
175  try {
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: ssrfExploit()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java:186
Taint Flags: NUMBER, WEB

Trust Boundary Violation — Low

URL: null

src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java, line 186 (Trust Boundary Violation)

```
183  }
184
185  model.addAttribute("val", val);
186  model.addAttribute("intval", intVal);
187  model.addAttribute("logwritten", strLog);
188
189  return "admin/log";
```

src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java, line 157 (Trust Boundary Violation)

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.591

Source Details

Source: executeCommandShell(0)
From: com.microfocus.example.web.controllers.admin.AdminDefaultController.executeCommandShell
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java:162
URL: null

```
159  }
160
161  @PostMapping("/command-shell")
162  public String executeCommandShell(@RequestParam("cmdshell") String cmd,
163  RedirectAttributes redirectAttributes) {
164
165  this.thRCECMD = cmd;
```

Sink Details

Sink: org.springframework.ui.Model.addAttribute()
Enclosing Method: getCommandShell()
File: src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java:157
Taint Flags: WEB, XSS

Trust Boundary Violation — Low

URL: null

src/main/java/com/microfocus/example/web/controllers/admin/AdminDefaultController.java, line 157 (Trust Boundary Violation)

```
154  cmdWrapper = String.format("T (java.lang.Runtime).getRuntime().exec('%s')",
this.thRCECMD);
155  }
156  model.addAttribute("shellcmd", cmdWrapper);
157  model.addAttribute("usercmd", this.thRCECMD);
158  return "admin/command-shell";
159  }
160
```

Unreleased Resource: Files (1 issue)

Abstract

The program can potentially fail to release a file handle.

Explanation

The program can potentially fail to release a file handle.

Resource leaks have at least two common causes:

- Error conditions and other exceptional circumstances.

- Confusion over which part of the program is responsible for releasing the resource.

Most unreleased resource issues result in general software reliability problems. However, if an attacker can intentionally trigger a resource leak, the attacker may be able to launch a denial of service attack by depleting the resource pool.

Example 1: The following method never closes the file handle it opens. The `finalize()` method for `ZipFile` eventually calls `close()`, but there is no guarantee as to how long it will take before the `finalize()` method will be invoked. In a busy environment, this can result in the JVM using up all of its file handles.

```
public void printZipContents(String fName)
    throws ZipException, IOException, SecurityException,
IllegalStateException, NoSuchElementException
{
  ZipFile zf = new ZipFile(fName);
  Enumeration<ZipEntry> e = zf.entries();

  while (e.hasMoreElements()) {
    printFileInfo(e.nextElement());
  }
}
```

Example 2: Under normal conditions, the following fix properly closes the file handle after printing out all the zip file entries. But if an exception occurs while iterating through the entries, the zip file handle will not be closed. If this happens often enough, the JVM can still run out of available file handles.

```
public void printZipContents(String fName)
    throws ZipException, IOException, SecurityException,
IllegalStateException, NoSuchElementException
{
  ZipFile zf = new ZipFile(fName);
  Enumeration<ZipEntry> e = zf.entries();

  while (e.hasMoreElements()) {
    printFileInfo(e.nextElement());
  }
}
```

Recommendation

1. Never rely on `finalize()` to reclaim resources. In order for an object's `finalize()` method to be invoked, the garbage collector must determine that the object is eligible for garbage collection. Because the garbage collector is not required to run unless the JVM is low on memory, there is no guarantee that an object's `finalize()` method will be invoked in an expedient fashion. When the garbage collector finally does run, it may cause a large number of resources to be reclaimed in a short period of time, which can lead to "bursty" performance and lower overall system throughput. This effect becomes more pronounced as the load on the system increases.

Finally, if it is possible for a resource reclamation operation to hang (if it requires communicating over a network, for example), then the thread that is executing the `finalize()` method will hang.

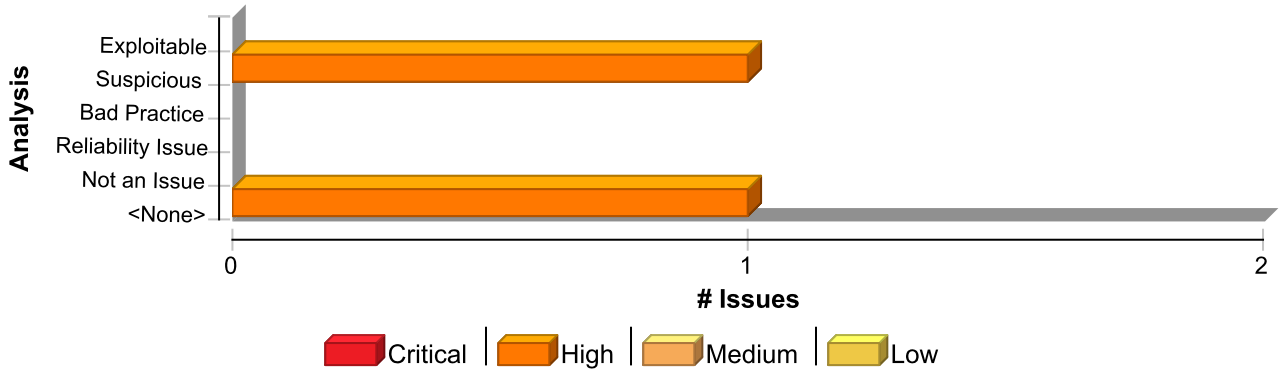
2. Release resources in a `finally` block. The code for `Example 2` should be rewritten as follows:

```
public void printZipContents(String fName)
    throws ZipException, IOException, SecurityException,
IllegalStateException, NoSuchElementException
{
  ZipFile zf;
  try {
    zf = new ZipFile(fName);
    Enumeration<ZipEntry> e = zf.entries();
    ...
  }
  finally {
    if (zf != null) {
      safeClose(zf);
    }
  }
}

public static void safeClose(ZipFile zf) {
  if (zf != null) {
    try {
      zf.close();
    } catch (IOException e) {
      log(e);
    }
  }
}
```

This solution uses a helper function to log the exceptions that might occur when trying to close the file. Presumably this helper function will be reused whenever a file needs to be closed.

Also, the `printZipContents` method does not initialize the `zf` object to `null`. Instead, it checks to ensure that `zf` is not `null` before calling `safeClose()`. Without the `null` check, the Java compiler reports that `zf` might not be initialized. This choice takes advantage of Java's ability to detect uninitialized variables. If `zf` is initialized to `null` in a more complex method, cases in which `zf` is used without being initialized will not be detected by the compiler.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Unreleased Resource: Files	1	0	0	1
Total	**1**	**0**	**0**	**1**

Unreleased Resource: Files — High

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/UserUtils.java, line 129 (Unreleased Resource: Files)

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.528

Sink Details

Sink: zf = new ZipFile(...)
Enclosing Method: logZipContents()
File: src/main/java/com/microfocus/example/utils/UserUtils.java:129

```
126
127  public void logZipContents(String fName)
128  throws IOException, SecurityException, IllegalStateException, NoSuchElementException {
129  ZipFile zf = new ZipFile(fName);
130  @SuppressWarnings("unchecked")
131  Enumeration<ZipEntry> e = (Enumeration<ZipEntry>) zf.entries();
132  while (e.hasMoreElements()) {
```

Unreleased Resource: Streams (1 issue)

Abstract

The program can potentially fail to release a system resource.

Explanation

The program can potentially fail to release a system resource.

Resource leaks have at least two common causes:

- Error conditions and other exceptional circumstances.

- Confusion over which part of the program is responsible for releasing the resource.

Most unreleased resource issues result in general software reliability problems. However, if an attacker can intentionally trigger a resource leak, the attacker may be able to launch a denial of service attack by depleting the resource pool.

Example: The following method never closes the file handle it opens. The `finalize()` method for `FileInputStream` eventually calls `close()`, but there is no guarantee as to how long it will take before the `finalize()` method will be invoked. In a busy environment, this can result in the JVM using up all of its file handles.

```
private void processFile(String fName) throws FileNotFoundException,
IOException {
  FileInputStream fis = new FileInputStream(fName);
  int sz;
  byte[] byteArray = new byte[BLOCK_SIZE];
  while ((sz = fis.read(byteArray)) != -1) {
    processBytes(byteArray, sz);
  }
}
```

Recommendation

1. Never rely on `finalize()` to reclaim resources. In order for an object's `finalize()` method to be invoked, the garbage collector must determine that the object is eligible for garbage collection. Because the garbage collector is not required to run unless the JVM is low on memory, there is no guarantee that an object's `finalize()` method will be invoked in an expedient fashion. When the garbage collector finally does run, it may cause a large number of resources to be reclaimed in a short period of time, which can lead to "bursty" performance and lower overall system throughput. This effect becomes more pronounced as the load on the system increases.

Finally, if it is possible for a resource reclamation operation to hang (if it requires communicating over a network to a database, for example), then the thread that is executing the `finalize()` method will hang.

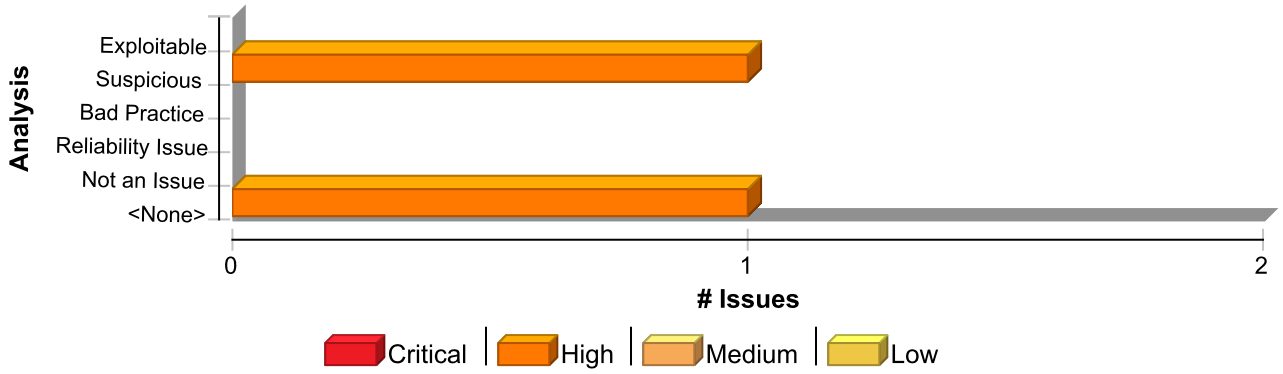
2. Release resources in a `finally` block. The code for the Example should be rewritten as follows:

```
public void processFile(String fName) throws FileNotFoundException,
IOException {
  FileInputStream fis;
  try {
    fis = new FileInputStream(fName);
    int sz;
    byte[] byteArray = new byte[BLOCK_SIZE];
    while ((sz = fis.read(byteArray)) != -1) {
      processBytes(byteArray, sz);
    }
  }
  finally {
    if (fis != null) {
      safeClose(fis);
    }
  }
}

public static void safeClose(FileInputStream fis) {
  if (fis != null) {
    try {
      fis.close();
    } catch (IOException e) {
      log(e);
    }
  }
}
```

This solution uses a helper function to log the exceptions that might occur when trying to close the stream. Presumably this helper function will be reused whenever a stream needs to be closed.

Also, the `processFile` method does not initialize the `fis` object to `null`. Instead, it checks to ensure that `fis` is not `null` before calling `safeClose()`. Without the `null` check, the Java compiler reports that `fis` might not be initialized. This choice takes advantage of Java's ability to detect uninitialized variables. If `fis` is initialized to `null` in a more complex method, cases in which `fis` is used without being initialized will not be detected by the compiler.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Unreleased Resource: Streams	1	0	0	1
Total	**1**	**0**	**0**	**1**

Unreleased Resource: Streams — High

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/UserUtils.java, line 81 (Unreleased Resource: Streams)

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Audit Details

Analysis	Suspicious
AA_Training	Include
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.526

Sink Details

Sink: new FileReader(...)
Enclosing Method: registerUser()
File: src/main/java/com/microfocus/example/utils/UserUtils.java:81

```
78
79  File dataFile = new File(getFilePath(NEWSLETTER_USER_FILE));
80  if (dataFile.exists()) {
81  jsonArray = (JSONArray) jsonParser.parse(new
FileReader(getFilePath(NEWSLETTER_USER_FILE)));
82  } else {
83  dataFile.createNewFile();
84  log.debug("Created: " + getFilePath(NEWSLETTER_USER_FILE));
```

Weak Encryption (3 issues)

Abstract

The identified call uses a weak encryption algorithm that cannot guarantee the confidentiality of sensitive data.

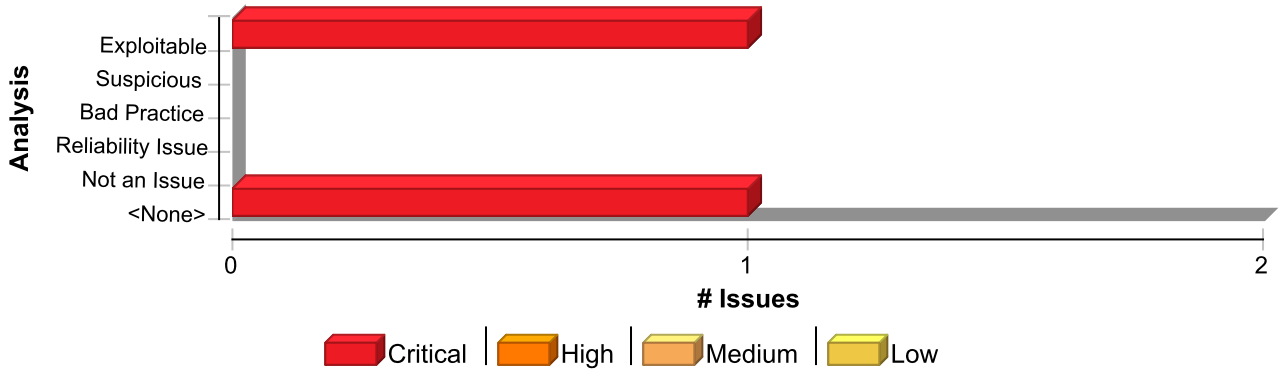
Explanation

Antiquated encryption algorithms such as DES no longer provide sufficient protection for use with sensitive data. Encryption algorithms rely on key size as one of the primary mechanisms to ensure cryptographic strength. Cryptographic strength is often measured by the time and computational power needed to generate a valid key. Advances in computing power have made it possible to obtain small encryption keys in a reasonable amount of time. For example, the 56-bit key used in DES posed a significant computational hurdle in the 1970s when the algorithm was first developed, but today DES can be cracked in less than a day using commonly available equipment.

Recommendation

Use strong encryption algorithms with large key sizes to protect sensitive data. A strong alternative to DES is AES (Advanced Encryption Standard, formerly Rijndael). Before selecting an algorithm, first determine if your organization has standardized on a specific algorithm and implementation.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Weak Encryption	3	0	0	3
Total	**3**	**0**	**0**	**3**

Weak Encryption	Critical

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/EncryptedPasswordUtils.java, line 62 (Weak Encryption)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Weak Encryption — Critical

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/EncryptedPasswordUtils.java, line 62 (Weak Encryption)

Audit Details

JiraBugLink	
Analysis	Exploitable
AA_Prediction	Not Predicted

Audit Comments

admin: Wed Jun 14 2023 14:47:36 GMT-0000 (UTC)
Its a security issue that needs to be fixed.

Sink Details

Sink: getInstance()
Enclosing Method: matches()
File: src/main/java/com/microfocus/example/utils/EncryptedPasswordUtils.java:62

```
59  byte[] encrypted = null;
60  String encPassword1 = "";
61  try {
62  Cipher desCipher = Cipher.getInstance("DES");
63  desCipher.init(Cipher.ENCRYPT_MODE, keySpec);
64  encrypted = desCipher.doFinal(password1.getBytes());
65  encPassword1 = new String(encrypted);
```

src/main/java/com/microfocus/example/utils/EncryptedPasswordUtils.java, line 46 (Weak Encryption)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Audit Details

AA_Prediction	Not Predicted

Sink Details

Sink: getInstance()
Enclosing Method: encryptPassword()
File: src/main/java/com/microfocus/example/utils/EncryptedPasswordUtils.java:46

```
43  public static String encryptPassword(String password) {
44  byte[] encrypted = null;
45  try {
46  Cipher desCipher = Cipher.getInstance("DES");
47  desCipher.init(Cipher.ENCRYPT_MODE, keySpec);
48  encrypted = desCipher.doFinal(password.getBytes());
49  } catch (NoSuchAlgorithmException | NoSuchPaddingException | InvalidKeyException |
IllegalBlockSizeException | BadPaddingException e) {
```

Weak Encryption — Critical

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/EncryptedPasswordUtils.java, line 41 (Weak Encryption)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Audit Details

AA_Prediction	Not Predicted

Sink Details

Sink: SecretKeySpec()
Enclosing Method: ()
File: src/main/java/com/microfocus/example/utils/EncryptedPasswordUtils.java:41

```
38 public class EncryptedPasswordUtils {
39
40  private static final byte[] iv = { 22, 33, 11, 44, 55, 99, 66, 77 };
41  private static final SecretKey keySpec = new SecretKeySpec(iv, "DES");
42
43  public static String encryptPassword(String password) {
44  byte[] encrypted = null;
```

Weak Encryption: Insecure Mode of Operation (2 issues)

Abstract

Do not use cryptographic encryption algorithms with an insecure mode of operation.

Explanation

The mode of operation of a block cipher is an algorithm that describes how to repeatedly apply a cipher's single-block operation to securely transform amounts of data larger than a block. Some modes of operation include Electronic Codebook (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), and Counter (CTR).

ECB mode is inherently weak, as it produces the same ciphertext for identical blocks of plain text. CBC mode is vulnerable to padding oracle attacks. CTR mode is the superior choice because it does not have these weaknesses.

Example 1: The following code uses the AES cipher with ECB mode:

```
...
SecretKeySpec key = new SecretKeySpec(keyBytes, "AES");
Cipher cipher = Cipher.getInstance("AES/ECB/PKCS7Padding", "BC");
cipher.init(Cipher.ENCRYPT_MODE, key);
...
```

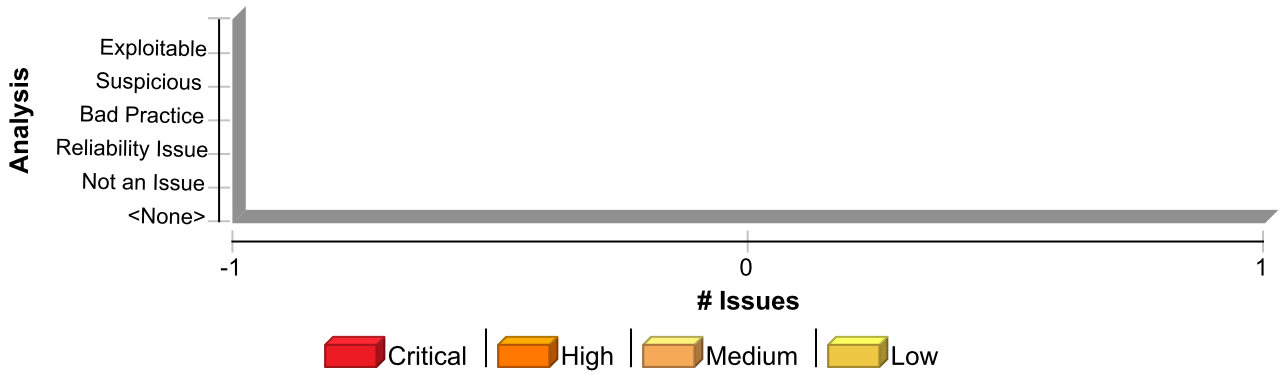
Recommendation

Avoid using ECB and CBC modes of operation when encrypting data larger than a block. CBC mode is somewhat inefficient and poses a serious risk if used with SSL [1]. Instead, use CCM (Counter with CBC-MAC) mode or, if performance is a concern, GCM (Galois/Counter Mode) mode where they are available.

Example 2: The following code uses the AES cipher with GCM mode:

```
...
SecretKeySpec key = new SecretKeySpec(keyBytes, "AES");
Cipher cipher = Cipher.getInstance("AES/GCM/PKCS5Padding", "BC");
cipher.init(Cipher.ENCRYPT_MODE, key);
...
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Weak Encryption: Insecure Mode of Operation	2	0	0	2
Total	**2**	**0**	**0**	**2**

Weak Encryption: Insecure Mode of Operation — Critical

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/EncryptedPasswordUtils.java, line 62 (Weak Encryption: Insecure Mode of Operation)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: getInstance()
Enclosing Method: matches()
File: src/main/java/com/microfocus/example/utils/EncryptedPasswordUtils.java:62

```
59  byte[] encrypted = null;
60  String encPassword1 = "";
61  try {
62  Cipher desCipher = Cipher.getInstance("DES");
63  desCipher.init(Cipher.ENCRYPT_MODE, keySpec);
64  encrypted = desCipher.doFinal(password1.getBytes());
65  encPassword1 = new String(encrypted);
```

src/main/java/com/microfocus/example/utils/EncryptedPasswordUtils.java, line 46 (Weak Encryption: Insecure Mode of Operation)

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Audit Details

AA_Prediction Not Predicted

Sink Details

Sink: getInstance()
Enclosing Method: encryptPassword()
File: src/main/java/com/microfocus/example/utils/EncryptedPasswordUtils.java:46

Weak Encryption: Insecure Mode of Operation — Critical

Package: com.microfocus.example.utils

src/main/java/com/microfocus/example/utils/EncryptedPasswordUtils.java, line 46 (Weak Encryption: Insecure Mode of Operation)

```
43  public static String encryptPassword(String password) {
44  byte[] encrypted = null;
45  try {
46  Cipher desCipher = Cipher.getInstance("DES");
47  desCipher.init(Cipher.ENCRYPT_MODE, keySpec);
48  encrypted = desCipher.doFinal(password.getBytes());
49  } catch (NoSuchAlgorithmException | NoSuchPaddingException | InvalidKeyException |
IllegalBlockSizeException | BadPaddingException e) {
```

XML Entity Expansion Injection (5 issues)

Abstract

Using XML parsers configured to not prevent nor limit Document Type Definition (DTD) entity resolution can expose the parser to an XML Entity Expansion injection

Explanation

XML Entity Expansion injection also known as XML Bombs are Denial Of Service (DoS) attacks that benefit from valid and well-formed XML blocks that expand exponentially until they exhaust the server allocated resources. XML allows to define custom entities which act as string substitution macros. By nesting recurrent entity resolutions, an attacker may easily crash the server resources.

The following XML document shows an example of an XML Bomb.

```
<?xml version="1.0"?>
<!DOCTYPE lolz [
  <!ENTITY lol "lol">
  <!ENTITY lol2 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
  <!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
  <!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
  <!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
  <!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
  <!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
  <!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
  <!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">
]>
<lolz>&lol9;</lolz>
```

This test could crash the server by expanding the small XML document into more than 3GB in memory.

Recommendation

An XML parser should be configured securely so that it does not allow document type definition (DTD) custom entities as part of an incoming XML document.

To avoid XML Entity Expansion injection the "secure-processing" property should be set for an XML factory, parser or reader:

```
factory.setFeature("http://javax.xml.XMLConstants/feature/secure-processing",
true);
```

In JAXP 1.3 and earlier versions, when the secure processing feature is on, default limitations are set for DOM and SAX parsers. These limits are:

```
entityExpansionLimit = 64,000
elementAttributeLimit = 10,000
```

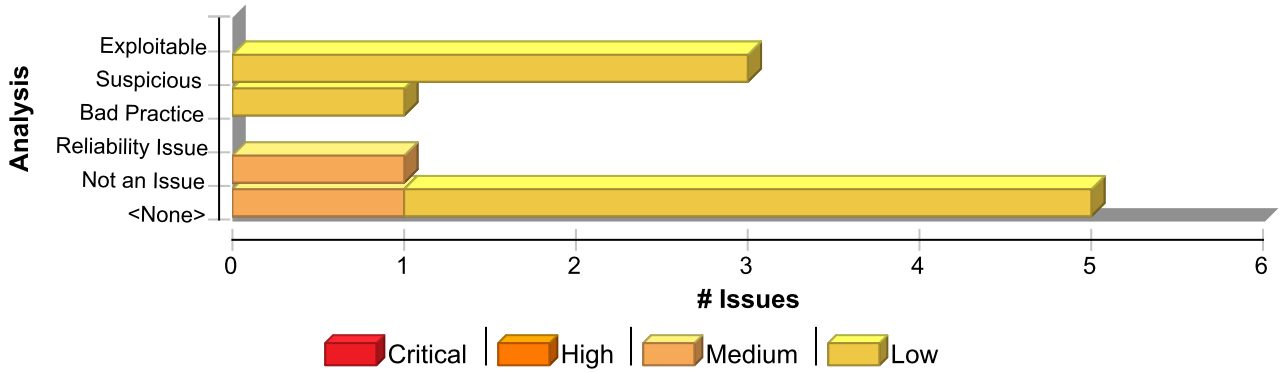
Since JAXP 1.4, the secure processing feature is turned on by default. In addition to the preceding limits, a new maxOccur limit is added to the validating parser. The limit is:

```
maxOccur = 5,000
```

If inline DOCTYPE declaration is not needed, it can be completely disabled with the following property:

```
factory.setFeature("http://apache.org/xml/features/disallow-doctype-decl",
true);
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
XML Entity Expansion Injection	5	0	0	5
Total	**5**	**0**	**0**	**5**

XML Entity Expansion Injection — Medium

URL: null

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 662 (XML Entity Expansion Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Not an Issue
AA_Prediction	Not an Issue
AA_Confidence	0.901

Source Details

Source: handleXMLUpdate(1)
From: com.microfocus.example.web.controllers.UserController.handleXMLUpdate
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:653
URL: null

```
650
651  @PostMapping("/files/xml/update")
652  public String handleXMLUpdate(@RequestParam("filename") String fileName,
653  @RequestParam("fcontent") String newXMLContent,
654  RedirectAttributes redirectAttributes) {
655
656  Path fpath = storageService.load(fileName);
```

Sink Details

XML Entity Expansion Injection — Medium

URL: null

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 662 (XML Entity Expansion Injection)

Sink: javax.xml.parsers.DocumentBuilder.parse()
Enclosing Method: handleXMLUpdate()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:662
Taint Flags: WEB, XSS

```
659  try {
660  dbf.setFeature(XMLConstants.FEATURE_SECURE_PROCESSING, false);
661  DocumentBuilder db = dbf.newDocumentBuilder();
662  Document doc = db.parse(new InputSource(new StringReader(newXMLContent)));
663  Path temp = Files.createTempFile("iwa", ".xml");
664  try (FileOutputStream outStream = new FileOutputStream(temp.toString())) {
665  writeXml(doc, outStream);
```

XML Entity Expansion Injection — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 642 (XML Entity Expansion Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Control Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.52

Sink Details

Sink: transformerFactory.newTransformer() : XML document parsed allowing external entity resolution
Enclosing Method: writeXml()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:642

```
639  throws TransformerException {
640
641  TransformerFactory transformerFactory = TransformerFactory.newInstance();
642  Transformer transformer = transformerFactory.newTransformer();
643
644  DOMSource source = new DOMSource(doc);
645  StreamResult result = new StreamResult(output);
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 600 (XML Entity Expansion Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

XML Entity Expansion Injection — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 600 (XML Entity Expansion Injection)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.56

Source Details

Source: Read this.location
From: com.microfocus.example.config.StorageProperties.getLocation
File: src/main/java/com/microfocus/example/config/StorageProperties.java:16

```
13  private String location = System.getProperty("user.home") +
File.separatorChar + "upload-dir";
14
15  public String getLocation() {
16  return location;
17  }
18
19  public void setLocation(String location) {
```

Sink Details

Sink: javax.xml.parsers.DocumentBuilder.parse()
Enclosing Method: getXMLFileContent()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:600
Taint Flags: PROPERTY, TAINTED_PATH

```
597  try {
598  dbf.setFeature(XMLConstants.FEATURE_SECURE_PROCESSING, false);
599  DocumentBuilder db = dbf.newDocumentBuilder();
600  Document doc = db.parse(fpath.toFile());
601  try (ByteArrayOutputStream bytesOutStream = new ByteArrayOutputStream()) {
602  writeXml(doc, bytesOutStream);
603  xmlContent = bytesOutStream.toString();
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 660 (XML Entity Expansion Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Control Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.52

Sink Details

XML Entity Expansion Injection — Low

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 660 (XML Entity Expansion Injection)

Sink: dbf.setFeature(...) : External entity resolution allowed
Enclosing Method: handleXMLUpdate()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:660

```
657
658  DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
659  try {
660  dbf.setFeature(XMLConstants.FEATURE_SECURE_PROCESSING, false);
661  DocumentBuilder db = dbf.newDocumentBuilder();
662  Document doc = db.parse(new InputSource(new StringReader(newXMLContent)));
663  Path temp = Files.createTempFile("iwa", ".xml");
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 598 (XML Entity Expansion Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Control Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.52

Sink Details

Sink: dbf.setFeature(...) : External entity resolution allowed
Enclosing Method: getXMLFileContent()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:598

```
595  String xmlContent = "";
596  DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
597  try {
598  dbf.setFeature(XMLConstants.FEATURE_SECURE_PROCESSING, false);
599  DocumentBuilder db = dbf.newDocumentBuilder();
600  Document doc = db.parse(fpath.toFile());
601  try (ByteArrayOutputStream bytesOutStream = new ByteArrayOutputStream()) {
```

XML External Entity Injection (5 issues)

Abstract

Using XML parsers configured to not prevent nor limit external entities resolution can expose the parser to an XML External Entities attack

Explanation

XML External Entities attacks benefit from an XML feature to build documents dynamically at the time of processing. An XML entity allows inclusion of data dynamically from a given resource. External entities allow an XML document to include data from an external URI. Unless configured to do otherwise, external entities force the XML parser to access the resource specified by the URI, e.g., a file on the local machine or on a remote system. This behavior exposes the application to XML External Entity (XXE) attacks, which can be used to perform denial of service of the local system, gain unauthorized access to files on the local machine, scan remote machines, and perform denial of service of remote systems.

The following XML document shows an example of an XXE attack.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///dev/random" >]><foo>&xxe;</foo>
```

This example could crash the server (on a UNIX system), if the XML parser attempts to substitute the entity with the contents of the /dev/random file.

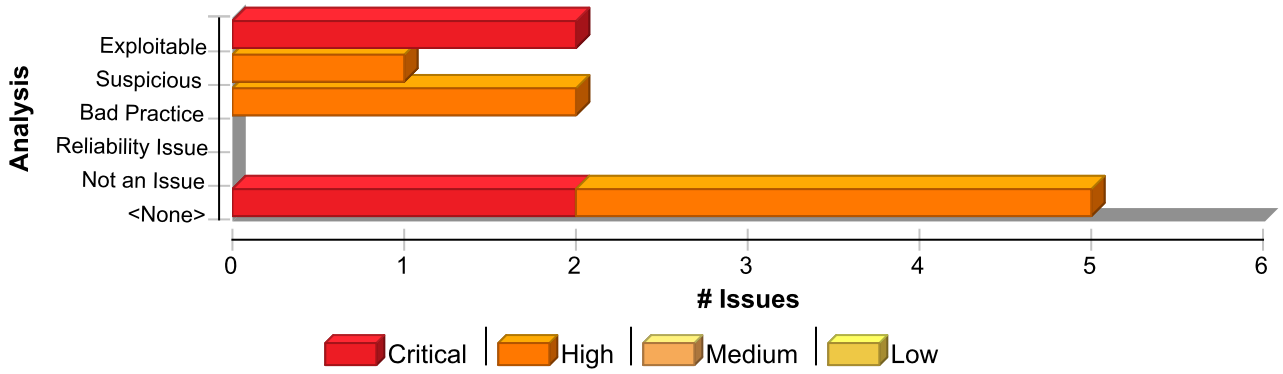
Recommendation

The XML unmarshaller should be configured securely so that it does not allow external entities as part of an incoming XML document.

To avoid XXE injection do not use unmarshal methods that process an XML source directly as java.io.File, java.io.Reader or java.io.InputStream. Parse the document with a securely configured parser and use an unmarshal method that takes the secure parser as the XML source as shown in the following example:

```
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
dbf.setFeature("http://apache.org/xml/features/disallow-doctype-decl", true);
DocumentBuilder db = dbf.newDocumentBuilder();
Document document = db.parse(<XML Source>);
Model model = (Model) u.unmarshal(document);
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
XML External Entity Injection	5	0	0	5
Total	**5**	**0**	**0**	**5**

XML External Entity Injection — Critical

URL: null

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 647 (XML External Entity Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Exploitable
AA_Prediction	Exploitable
AA_Confidence	0.823

Source Details

Source: handleXMLUpdate(1)
From: com.microfocus.example.web.controllers.UserController.handleXMLUpdate
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:653
URL: null

```
650
651  @PostMapping("/files/xml/update")
652  public String handleXMLUpdate(@RequestParam("filename") String fileName,
653  @RequestParam("fcontent") String newXMLContent,
654  RedirectAttributes redirectAttributes) {
655
656  Path fpath = storageService.load(fileName);
```

Sink Details

Sink: javax.xml.transform.Transformer.transform()
Enclosing Method: writeXml()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:647
Taint Flags: WEB, XSS

XML External Entity Injection — Critical

URL: null

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 647 (XML External Entity Injection)

```
644  DOMSource source = new DOMSource(doc);
645  StreamResult result = new StreamResult(output);
646
647  transformer.transform(source, result);
648
649  }
650
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 662 (XML External Entity Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Exploitable
AA_Prediction	Exploitable
AA_Confidence	0.803

Source Details

Source: handleXMLUpdate(1)
From: com.microfocus.example.web.controllers.UserController.handleXMLUpdate
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:653
URL: null

```
650
651  @PostMapping("/files/xml/update")
652  public String handleXMLUpdate(@RequestParam("filename") String fileName,
653  @RequestParam("fcontent") String newXMLContent,
654  RedirectAttributes redirectAttributes) {
655
656  Path fpath = storageService.load(fileName);
```

Sink Details

Sink: javax.xml.parsers.DocumentBuilder.parse()
Enclosing Method: handleXMLUpdate()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:662
Taint Flags: WEB, XSS

XML External Entity Injection — Critical

URL: null

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 662 (XML External Entity Injection)

```
659  try {
660  dbf.setFeature(XMLConstants.FEATURE_SECURE_PROCESSING, false);
661  DocumentBuilder db = dbf.newDocumentBuilder();
662  Document doc = db.parse(new InputSource(new StringReader(newXMLContent)));
663  Path temp = Files.createTempFile("iwa", ".xml");
664  try (FileOutputStream outStream = new FileOutputStream(temp.toString())) {
665  writeXml(doc, outStream);
```

XML External Entity Injection — High

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 647 (XML External Entity Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Suspicious
AA_Prediction	Indeterminate (Below Exploitable threshold)
AA_Confidence	0.511

Source Details

Source: Read this.location
From: com.microfocus.example.config.StorageProperties.getLocation
File: src/main/java/com/microfocus/example/config/StorageProperties.java:16

```
13  private String location = System.getProperty("user.home") +
File.separatorChar + "upload-dir";
14
15  public String getLocation() {
16  return location;
17  }
18
19  public void setLocation(String location) {
```

Sink Details

Sink: javax.xml.transform.Transformer.transform()
Enclosing Method: writeXml()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:647
Taint Flags: PROPERTY, TAINTED_PATH

XML External Entity Injection — High

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 647 (XML External Entity Injection)

```
644  DOMSource source = new DOMSource(doc);
645  StreamResult result = new StreamResult(output);
646
647  transformer.transform(source, result);
648
649  }
650
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 642 (XML External Entity Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Control Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.534

Sink Details

Sink: transformerFactory.newTransformer() : XML document parsed allowing external entity resolution
Enclosing Method: writeXml()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:642

```
639  throws TransformerException {
640
641  TransformerFactory transformerFactory = TransformerFactory.newInstance();
642  Transformer transformer = transformerFactory.newTransformer();
643
644  DOMSource source = new DOMSource(doc);
645  StreamResult result = new StreamResult(output);
```

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 600 (XML External Entity Injection)

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis	Bad Practice
AA_Prediction	Indeterminate (Below Not An Issue threshold)
AA_Confidence	0.557

Source Details

XML External Entity Injection — High

Package: com.microfocus.example.web.controllers

src/main/java/com/microfocus/example/web/controllers/UserController.java, line 600 (XML External Entity Injection)

Source: Read this.location
From: com.microfocus.example.config.StorageProperties.getLocation
File: src/main/java/com/microfocus/example/config/StorageProperties.java:16

```
13  private String location = System.getProperty("user.home") +
File.separatorChar + "upload-dir";
14
15  public String getLocation() {
16  return location;
17  }
18
19  public void setLocation(String location) {
```

Sink Details

Sink: javax.xml.parsers.DocumentBuilder.parse()
Enclosing Method: getXMLFileContent()
File: src/main/java/com/microfocus/example/web/controllers/UserController.java:600
Taint Flags: PROPERTY, TAINTED_PATH

```
597  try {
598  dbf.setFeature(XMLConstants.FEATURE_SECURE_PROCESSING, false);
599  DocumentBuilder db = dbf.newDocumentBuilder();
600  Document doc = db.parse(fpath.toFile());
601  try (ByteArrayOutputStream bytesOutStream = new ByteArrayOutputStream()) {
602  writeXml(doc, bytesOutStream);
603  xmlContent = bytesOutStream.toString();
```

Description of Key Terminology

Likelihood and Impact

Likelihood
Likelihood is the probability that a vulnerability will be accurately identified and successfully exploited.

Impact
Impact is the potential damage an attacker could do to assets by successfully exploiting a vulnerability. This damage can be in the form of, but not limited to, financial loss, compliance violation, loss of brand reputation, and negative publicity.

Fortify Priority Order

Critical
Critical-priority issues have high impact and high likelihood. Critical-priority issues are easy to detect and exploit and result in large asset damage. These issues represent the highest security risk to the application. As such, they should be remediated immediately.

SQL Injection is an example of a critical issue.

High
High-priority issues have high impact and low likelihood. High-priority issues are often difficult to detect and exploit, but can result in large asset damage. These issues represent a high security risk to the application. High-priority issues should be remediated in the next scheduled patch release.

Password Management: Hardcoded Password is an example of a high issue.

Medium
Medium-priority issues have low impact and high likelihood. Medium-priority issues are easy to detect and exploit, but typically result in small asset damage. These issues represent a moderate security risk to the application. Medium-priority issues should be remediated in the next scheduled product update.

Path Manipulation is an example of a medium issue.

Low
Low-priority issues have low impact and low likelihood. Low-priority issues can be difficult to detect and exploit and typically result in small asset damage. These issues represent a minor security risk to the application. Low-priority issues should be remediated as time allows.

Dead Code is an example of a low issue.

About Fortify Solutions

Fortify is the leader in end-to-end application security solutions with the flexibility of testing on-premise and on-demand to cover the entire software development lifecycle. Learn more at www.microfocus.com/solutions/application-security.