

開発者の負担を軽減し、脆弱性対応をよりスマートに



“診断して終わり”じゃない。
Aviatorは、次の一手まで示します。

脆弱性の内容だけでなく、修正コード例まで自動提示。
開発者の「どう直せば？」にすぐに応えます。

脆弱性対応は重要である一方、セキュリティ担当や開発者に大きな負担となります。
OpenText™ Application Security Aviator は、AIと大規模言語モデル（LLM）の力を活用することで、これら負担の軽減を実現します。

OpenText Application Security Aviatorとは？



AI/LLMを活用した監査

ソースコードや対応履歴の自然言語処理による自動分析と要約により、監査プロセスが飛躍的に効率化されます。



テキスト解析による明確な説明と対策提案

AIがセキュリティ修正の理由と対応履歴を文脈に基づいて要約・提示。開発者の説明負担を軽減し、監査準備も効率化されます。



開発者の生産性と実用性を向上

Application Security Aviatorは、開発者にとって“パーソナルセキュリティチャンピオン”のような存在です。セキュリティ課題をコードのコンテキストに即して説明し、開発者が理解しやすい言葉で示すことで、対応のしやすさを大きく向上させます。



シームレスな統合で滑らかな体験を実現

AIによる修正提案は、単なる脆弱性リストではなく、開発者のワークフローに自然に組み込まれます。これにより、煩雑な作業を減らし、セキュリティ対応がより円滑になります。

OpenText Application Security Aviatorの利用は簡単？



OpenText Core Application Security（パブリック型SaaS）画面



OpenText Static Application Security Testing（オンプレ版）にもバージョン25.2より利用が開始され、順次機能拡張中！



OpenText Application Security Aviatorの実際の結果は？



誤検知例（ハードコードパスワード）

Application Security Aviator によって、脆弱性の可能性がある項目の内容を正確に理解し、誤検知かどうかを自動的に判断します。このケースでは誤検知と判断されたため、ステータスは自動的に「問題ではない」に変更され、さらに表示が抑制されることで、人手によるトリージ（選別）作業の負荷が軽減されています。

また、その判断理由についても丁寧に説明されるため、開発者は診断結果を安心して確認し、その正確性を的確に判断することができます。

226470791 → src/main/java/com/fortify/aviator_fod_demo/HomeController.java: 22

S.O. WITH AVIATOR High Password Management: Hardcoded Password SMART FIX

Vulnerability Recommendations Code Diagram More Evidence History

Path 1 of 3 Word Wrap Stacked View Inline Analysis Trace

HomeController.java:22 - Field Access: PASSWORD

```
15 public class HomeController {
16
17     private Connection connection;
18
19     private final static String ATTRIB_MESSAGE = "message";
20     private final static String ATTRIB_USERNAME = "username";
21     private final static String USERNAME = "Please provide a username.";
22     private final static String PASSWORD = "Please provide a password.";
23     private final static String WRONG_CREDS = "Wrong credentials.";
24     private final static String SUCCESS = "Logged in successfully.";
25
26     @RequestMapping(value = "/auth/login", method = RequestMethod.POST)
27     public ModelAndView loginSubmit(@RequestBody LoginCredentials credentials,
28                                     @RequestParam String redirectUrl,
29                                     @HttpSession Request request,
30                                     HttpSession session) throws SQLException {
31         String username = credentials.getUsername();
32         String password = credentials.getPassword();
33         if(username.isEmpty()) {
34             request.setAttribute(ATTRIB_MESSAGE, USERNAME);
35         } else if(password.isEmpty()) {
36             request.setAttribute(ATTRIB_MESSAGE, PASSWORD);
37         } else {
38             System.out.println("username: " + username);
39             System.out.println("password: " + password);
40             boolean credentialsCorrect = false;
41             try{Connection connection = DriverManager.getConnection("localhost:1234", "dbuser", "secretpwd");
42                 try (Statement statement = connection.createStatement()) {
43                     try (ResultSet rs = statement.executeQuery(
44                         "SELECT 1 FROM users WHERE username = '" + username + "' AND password = '" + password + "'")) {
45                         credentialsCorrect = rs.next();
46                     }
47                 }
48             }
49         }
50     }
51 }
```

Aviator フィードバック

Fortify Aviator 2024/07/05 02:35:55 PM
Changed Auditor Status from 'Pending Review' to 'Not an Issue'

Fortify Aviator 2024/07/05 02:35:55 PM
The code line flagged by Fortify:

private final static String PASSWORD = "Please provide a password.;"

is not actually a hardcoded password. It is simply a string constant used as a human-readable label or message. This does not pose a security risk.

The actual password used for authentication is not hardcoded, but is retrieved from user input and passed to the database query:

try (ResultSet rs = statement.executeQuery(
 "SELECT 1 FROM users WHERE username = '" + username + "' AND password = '" + password + "'")) {
 credentialsCorrect = rs.next();
}

While this code has other security issues (e.g. SQL injection vulnerability), it does not contain a hardcoded password vulnerability.

Status設定 :
Not an Issue (問題ではない)

説明 : ハードコードされたパスワードではない

※不毛で時間のかかる誤検知除去の自動化

脆弱性例（SQLインジェクション）

このケースでは脆弱性と判断されたため、ステータスは自動的に「要対策」に変更され、脆弱性の説明に加えて、該当コード行および修正コードの例も提示されます。

これにより、通常であれば開発者自身が脆弱性の説明やサンプルコードをもとに修正方法を検討する必要がありますが、その作業を技術的に支援し、負荷の軽減につなげています。

226470796 → src/main/java/com/fortify/aviator_fod_demo/HomeController.java: 44

S.O. WITH AVIATOR Critical SQL Injection SMART FIX

Vulnerability Recommendations Code Diagram More Evidence History

Path 1 of 3 Word Wrap Stacked View Inline Analysis Trace

HomeController.java:44 - executeQuery()

```
25
26 @RequestMapping(value = "/auth/login", method = RequestMethod.POST)
27 public ModelAndView loginSubmit(@RequestBody LoginCredentials credentials,
28                                 @RequestParam String redirectUrl,
29                                 @HttpSession Request request,
30                                 HttpSession session) throws SQLException {
31     String username = credentials.getUsername();
32     String password = credentials.getPassword();
33     if(username.isEmpty()) {
34         request.setAttribute(ATTRIB_MESSAGE, USERNAME);
35     } else if(password.isEmpty()) {
36         request.setAttribute(ATTRIB_MESSAGE, PASSWORD);
37     } else {
38         System.out.println("username: " + username);
39         System.out.println("password: " + password);
40         boolean credentialsCorrect = false;
41         try{Connection connection = DriverManager.getConnection("localhost:1234", "dbuser", "secretpwd");
42             try (Statement statement = connection.createStatement()) {
43                 try (ResultSet rs = statement.executeQuery(
44                     "SELECT 1 FROM users WHERE username = '" + username + "' AND password = '" + password + "'")) {
45                     credentialsCorrect = rs.next();
46                 }
47             }
48             if(credentialsCorrect) {
49                 request.setAttribute(ATTRIB_MESSAGE, SUCCESS);
50                 session.setAttribute(ATTRIB_USERNAME, username);
51             } else {
52                 request.setAttribute(ATTRIB_MESSAGE, WRONG_CREDS);
53             }
54         }
55     }
56 }
```

Aviator フィードバック

Fortify Aviator 2024/07/05 02:35:57 PM
Changed Auditor Status from 'Pending Review' to 'Remediation Required'

Fortify Aviator 2024/07/05 02:35:57 PM
The code is vulnerable to SQL injection because it constructs a SQL query by concatenating user input (username and password) directly into the query string. This allows an attacker to inject malicious SQL code that could modify the intended query or execute arbitrary SQL commands.

To fix this vulnerability, the code should use a prepared statement with parameter binding instead of string concatenation. Here's how the code can be modified:

In file src/main/java/com/fortify/aviator_fod_demo/HomeController.java, replace lines 44-47 by the following lines:

String query = "SELECT 1 FROM users WHERE username = ? AND password = ?";
try (PreparedStatement stmt = connection.prepareStatement(query)) {
 stmt.setString(1, username);
 stmt.setString(2, password);
 try (ResultSet rs = stmt.executeQuery()) {
 credentialsCorrect = rs.next();
 }
}

※実コードに即したサンプル提示

Status設定 :
Remediation Required (要対策)

説明 : SQLインジェクション

コードを含む対策案

オープンテキスト株式会社

メールアドレス : jpmkt-group@opentext.com

OpenTextグループ（マイクロフォーカスエンタープライズ株式会社）

お問い合わせは[こちら](#)から