

Contrast Assess セッションメタデータ機能

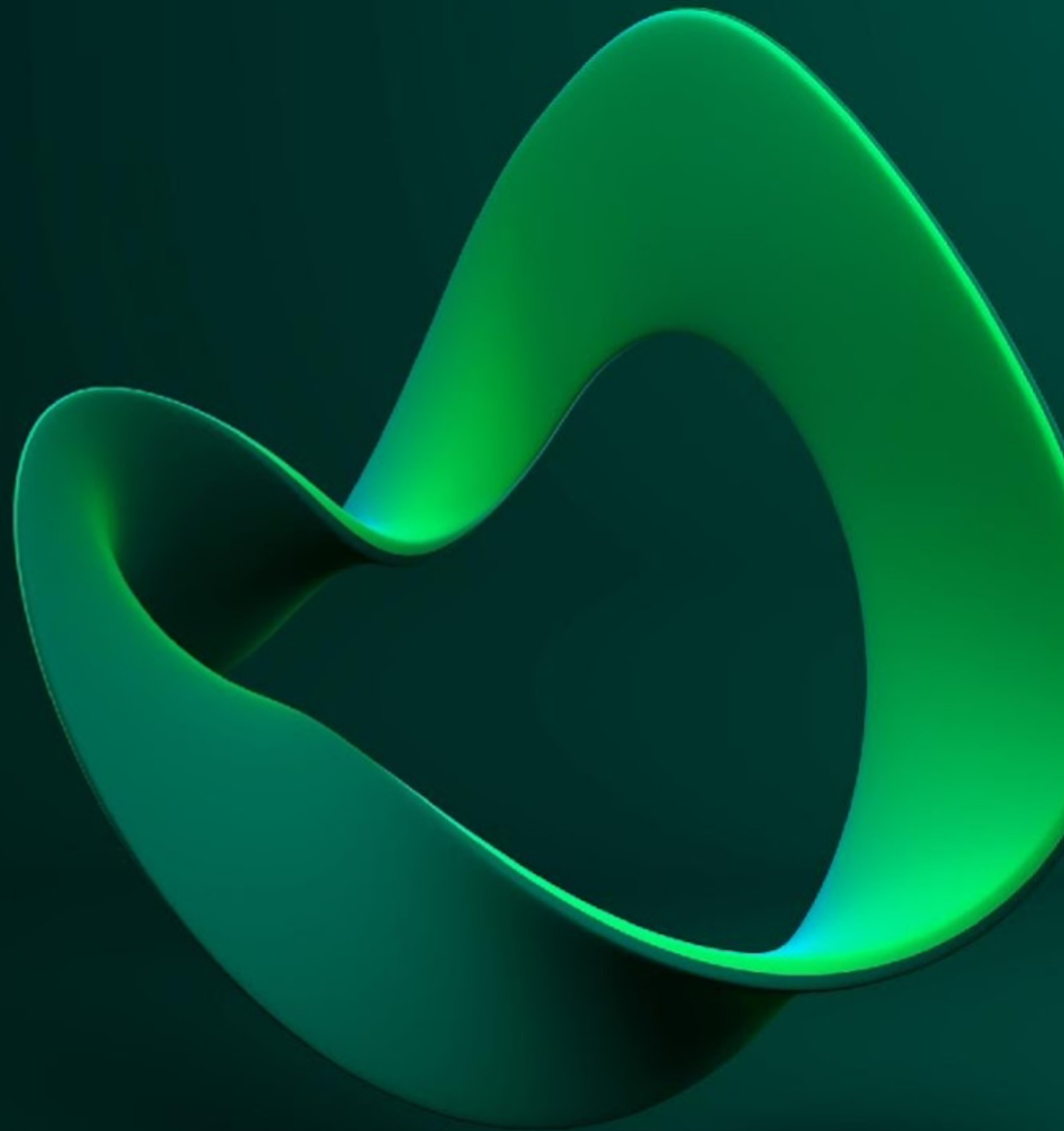
Contrast Security Japan 合同会社



本資料について

- この資料は、Contrast Assessの「セッションメタデータ」機能の利用目的や設定方法、ユースケースについて概要を説明します。

機能概要



セッションメタデータ機能の概要

- セッションメタデータ - Assessエージェントによる検査時に、その検査セッション（アプリの起動～エージェントによる検査～アプリの終了の一連の流れ）に関して、下記のような付加情報を追加することで、セッションごとの検査結果を区別できるようにする仕組み
 - ビルド番号
 - ブランチ名
 - アプリのバージョン
 - ... etc.
- 脆弱性やルートカバレッジを閲覧する際に、メタデータの値を元にしたフィルタリングなどが可能
- 主な目的
 - 脆弱性が修正されたかどうかの確認を容易にする
 - 複数の開発ブランチが並存しているような環境で、どのブランチのビルドがどの脆弱性を持っているか、区別できるようにする

セッションメタデータの設定

Contrastエージェントの動作パラメータ `session_metadata` に、キー/値ペア形式で渡します。

指定できるメタデータの一覧 設定例

Name	Value
Commit Hash	<code>commitHash</code>
Committer	<code>committer</code>
Branch Name	<code>branchName</code>
Git Tag	<code>gitTag</code>
Repository	<code>repository</code>
Test Run	<code>testRun</code>
Version	<code>version</code>
Build Number	<code>buildNumber</code>

Javaシステムプロパティとして指定

```
-Dcontrast.application.session_metadata="branchName=feature/some-new-thing,committer=Jane,repository=Contrast-Java"
```

Contrastエージェントのyaml設定ファイルで指定

```
application:  
  session_metadata: branchName=feature/some-new-thing,committer=Jane,  
  repository=Contrast-Ruby
```

CIシステム (Jenkinsなど) の環境変数をビルドスクリプト内で渡す

```
-Dcontrast.application.session_metadata="branchName=$GIT_BRANCH,committer=$GIT_COMMITTER_NAME,commitHash=$GIT_COMMIT_HASH,repository=$GIT_URL,buildNumber=$BUILD_NUMBER"
```

製品ドキュメント (<https://docs.contrastsecurity.com/en/configure-session-metadata.html>) より

脆弱性一覧画面でのセッションメタデータの表示

1. 上部のドロップダウンから、表示するメタデータを選択します。

The screenshot shows the '脆弱性' (Vulnerabilities) tab in the petclinicdemo5 application. A dropdown menu is open, showing options for metadata: REPOSITORY, COMMITTER, BRANCH NAME, BUILD NUMBER, and DISSOCIATED. The 'セッション' (Session) column header is also visible.

深刻度	脆弱性	ステータス	セッション
重大	HQLインジェクション: 「/owners」ページの「lastName」	報告済	Contrast-Java
中	「MD5」ハッシュアルゴリズムを使用: GranteeManager	報告済	Contrast-Java
注意	キャッシュ防止制御の欠如を検出	報告済	Contrast-Java
注意	クリックジャッキング対策の制御がないページを検出	報告済	Contrast-Java
注意	オートコンプリート防止のないフォームを検出	報告済	Contrast-Java

2. 「セッション」列に、選択したメタデータの値が表示されます。

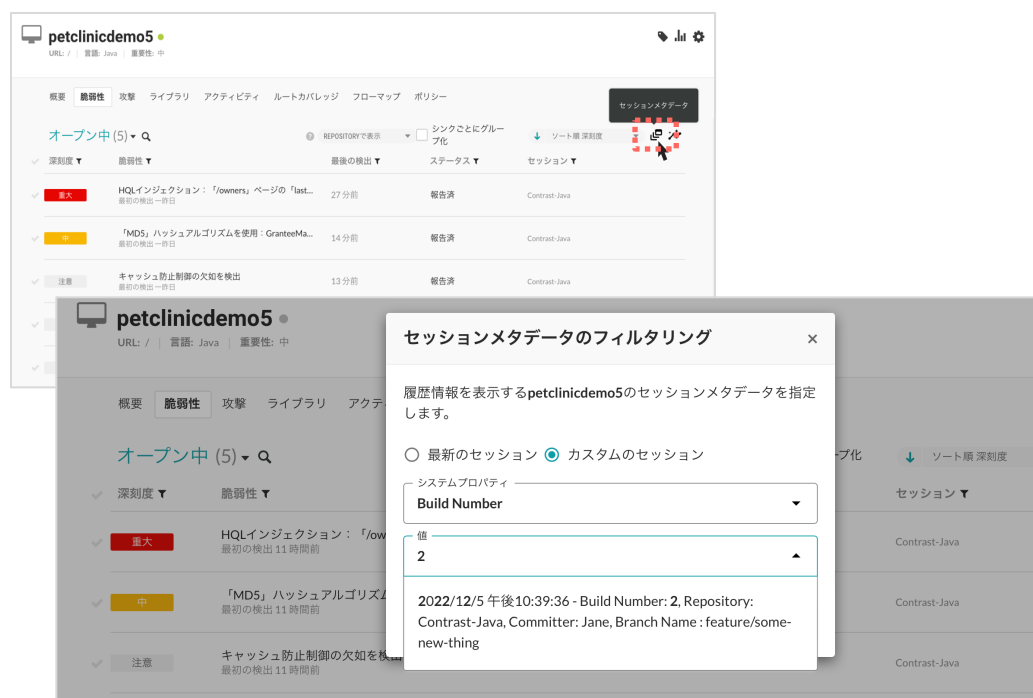
The screenshot shows the same vulnerability list interface, but now the 'セッション' column displays the selected metadata values: 1, 2.1, 2.1, 2.1, 2.1, and 2.1. The dropdown menu is now set to 'BUILD NUMBER'.

深刻度	脆弱性	最後の検出	ステータス	セッション
重大	HQLインジェクション: 「/owners」ページの「lastName」パラ...	6時間前	報告済	1
中	「MD5」ハッシュアルゴリズムを使用: GranteeManager	4時間前	報告済	2.1
注意	キャッシュ防止制御の欠如を検出	4時間前	報告済	2.1
注意	クリックジャッキング対策の制御がないページを検出	4時間前	報告済	2.1
注意	オートコンプリート防止のないフォームを検出	4時間前	報告済	2.1

セッションメタデータを使った脆弱性のフィルタリング

1. 右上の四角が重なったアイコンをクリックして、フィルタに使うプロパティ名と値を指定します。

2. 該当するセッションからの検出結果のみが表示されます。



「脆弱性」画面のほか、「ルートカバレッジ」画面でも同じ要領でセッションを特定して確認ができます。

特定検査セッションに遡った脆弱性詳細情報 (脆弱性インスタンス)の確認

脆弱性詳細情報を特定セッション時点に遡って確認したい場合、脆弱性ページの「備考」タブから「脆弱性インスタンス」を指定します。

1. petclinicdemo5 ●
URL: / | 言語: Java | 重要性: 中

概要 脆弱性 攻撃 ライブラリ アクティビティ ルートカバレッジ フローマップ ポリシー

< 検索に戻る

HQLインジェクション: 「/owners」ページの「lastName」パラメータ

重大 | 日付: 12/12/2022 09:29 午後 | ステータス: 報告済 | ID: 2W9W-XA1B-BQVH-IWRK

概要 詳細 HTTP情報 修正方法 **備考** アクティビティ

環境 Development

最初の検出 12 2022 最後の検出 13 2022

何が起こったか?

「lastName」パラメータの次のデータを追跡しました:

```
GET /owners?lastName=contrast-redacted-name
```

このデータは、次のコード内でアクセスされました:

```
org.hibernate.jpa.spi.AbstractEntityManagerImpl#createQuery(), 305行目
```

2.

- PCI - 3.2.1 セキュリティ基準: 6.5.1
- PCI - 3.0 セキュリティ基準: 6.5.1
- セキュリティ基準: API8, API10
- PCI - 2.0 セキュリティ基準: 6.5.1
- OWASP Top Ten - 2017 セキュリティ基準: A1
- IPA - 7.0 セキュリティ基準: 1

メタデータ

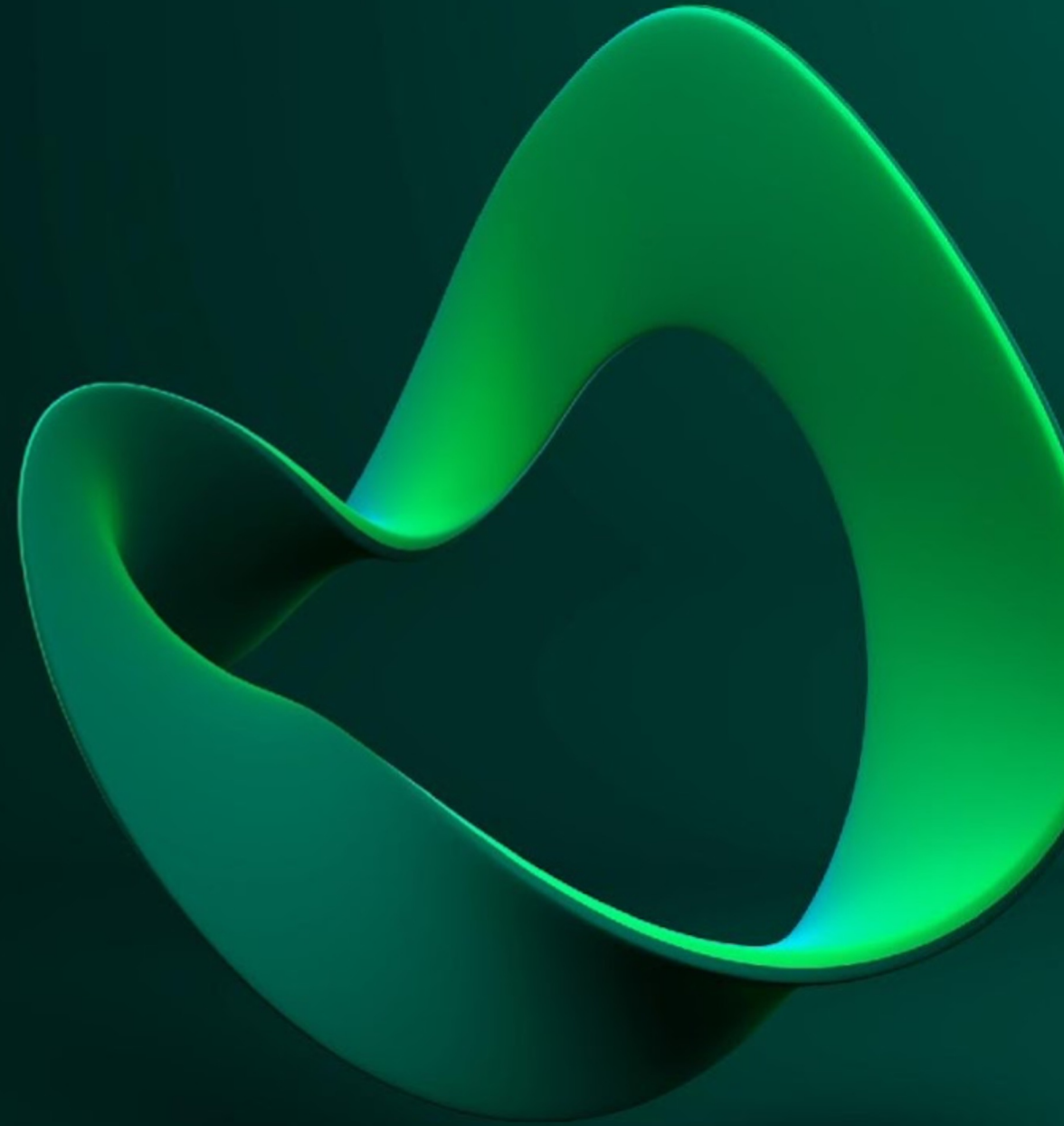
- Build Number: 3
- Repository: Contrast-Java
- Committer: Jane
- Branch Name: feature/some-new-thing

セッションID 41c2d2b75383f3b56257071857a1323d

脆弱性インスタンス

- D006-7CC2-VJTV-CZ3I (1時間前)
- 2W9W-XA1B-BQVH-IWRK (1時間前)
- J8Q0-3D3U-VSCL-8ZQW (15時間前)

利用シナリオ例



利用シナリオ例 – 脆弱性修正の追跡 (1/3)

① 脆弱性のあるコードをテスト

1. 脆弱性のあるコード

```
19 public Collection<Owner> findByLastName(String lastName) {
20     System.out.println("Vulnerable method 1");
21     // 脆弱なクエリ構築 ← ここから
22     String sqlQuery = "SELECT DISTINCT owner FROM Owner owner left
23         join fetch owner.pets WHERE owner.lastName LIKE '" + lastName +
24         "%'";
25     TypedQuery<Owner> query = this.entityManager.createQuery(sqlQuery,
26         Owner.class);
27     // 脆弱なクエリ構築 ← ここまで
28     return query.getResultList();
29 }
```

2. エージェントオプションで、セッションメタデータ Build Number = 1 を指定して起動・検査

```
java -javaagent:contrast.jar \
-Dcontrast.config.path=contrast_security.yaml \
-Dcontrast.application.session_metadata="branchName=feature/
some-new-thing,committer=Jane,repository=Contrast-Java,buildNumber=1" \
-jar ./target/spring-petclinic-1.5.1.jar
```

3. 検出結果の表示

The screenshot displays the Contrast Security interface for a project named 'petclinicdemo5'. The top navigation bar includes '概要', '脆弱性', '攻撃', 'ライブラリ', 'アクティビティ', 'ルートカバレッジ', 'フローマップ', and 'ポリシー'. The main content area shows a list of vulnerabilities under the heading 'オープン中 (5)'. The first vulnerability is highlighted in red, indicating a critical issue: 'HQLインジェクション: 「owners」ページの「lastName」パラメータ'. Below this, a detailed view of this vulnerability is shown, including the HTTP request parameters, the code snippet where the injection occurs, and the resulting SQL query.

脆弱性	攻撃	ライブラリ	アクティビティ	ルートカバレッジ	フローマップ	ポリシー
重大	HQLインジェクション: 「owners」ページの「lastName」パラメータ					
中	「MDS」ハッシュアルゴリズムを使用: GranteeManager					
注意	キャッシュ防止制御の欠如を検出					
注意	クリックジャッキング対策の制					
注意	オートコンプリート防止の					

HQLインジェクション: 「owners」ページの「lastName」パラメータ

概要 | 詳細 | HTTP情報 | 修正方法 | 備考 | アクティビティ

HTTPのパラメータを取得

```
string[] = facade.getParameterValues("lastName");
getParametersStartingWith() @ WebUtils.java:672
```

文字列操作が発生

このコード行で

```
findByLastName() @ OwnerRepositoryCustomImpl.java:22
```

```
SELECT DISTINCT owner FR...E
owner.lastName LIKE 'davis%'
```

ルール違反の検出

```
query = impl.createQuery("SELECT
DISTINCT owner FROM Owner owner left
join...")
createQuery() @ AbstractEntityManagerImpl.java:305
```

利用シナリオ例 – 脆弱性修正の追跡 (2/3)

② 脆弱なコードを修正し、再テスト

1. 修正されたコード

```
19 public Collection<Owner> findByLastName(String lastName) {
20     System.out.println("Vulnerable method 1");
21
22     // 脆弱なクエリ構築 -- ここから
23     //String sqlQuery = "SELECT DISTINCT owner FROM Owner owner left
24     //join fetch owner.pets WHERE owner.lastName LIKE '" + lastName +
25     //"%";
26     //TypedQuery<Owner> query = this.entityManager.createQuery(
27     //sqlQuery, Owner.class);
28     // 脆弱なクエリ構築 -- ここまで
29
30     // 修正済み -- ここから
31     String sqlQuery = "SELECT DISTINCT owner FROM Owner owner left
32     join fetch owner.pets WHERE owner.lastName LIKE :lastName";
33     TypedQuery<Owner> query = this.entityManager.createQuery(sqlQuery,
34     Owner.class);
35     query.setParameter("lastName", lastName + "%");
36     // 修正済み -- ここまで
37
38     return query.getResultList();
39 }
```

2. エージェントオプションで、セッションメタデータ Build Number = 2を指定して起動・検査

```
java -javaagent:contrast.jar \
-Dcontrast.config.path=contrast_security.yaml \
-Dcontrast.application.session_metadata="branchName=feature/
some-new-thing,committer=Jane,repository=Contrast-Java,buildNumber=2" \
-jar ./target/spring-petclinic-1.5.1.jar
```

2. Contrast Serverで脆弱性一覧画面にナビゲート。(一見変化がない。)

The screenshot shows the Contrast Server interface for a project named 'petclinicdemo5'. The '脆弱性' (Vulnerabilities) tab is selected, showing a list of 5 open vulnerabilities. The table columns are: Severity, Vulnerability Description, Last Detected, Status, and Session. The first vulnerability is 'HQLインジェクション: 「/owners」ページの「lastName」パラ...' with a severity of '重大' (Critical) and a status of '報告済' (Reported). Other vulnerabilities include '「MD5」ハッシュアルゴリズムを使用: GranteeManager', 'キャッシュ防止制御の欠如を検出', 'クリックジャッキング対策の制御がないページを検出', and 'オートコンプリート防止のないフォームを検出'.

深さ	脆弱性	最後の検出	ステータス	セッション
重大	HQLインジェクション: 「/owners」ページの「lastName」パラ...	2時間前	報告済	Contrast-Java
中	「MD5」ハッシュアルゴリズムを使用: GranteeManager	38分前	報告済	Contrast-Java
注意	キャッシュ防止制御の欠如を検出	37分前	報告済	Contrast-Java
注意	クリックジャッキング対策の制御がないページを検出	37分前	報告済	Contrast-Java
注意	オートコンプリート防止のないフォームを検出	37分前	報告済	Contrast-Java

利用シナリオ例 – 脆弱性修正の追跡 (3/3)

③ セッションメタデータを使って脆弱性が修正されたことを確認

1. 「セッション」列にBuild Numberを表示。HQLインジェクション脆弱性は修正後の「2」には該当しないことが確認できる。

2. フィルタ(Build Number = 2)を適用。HQLインジェクション脆弱性は表示されなくなっている。

深さ	脆弱性	最後の検出	ステータス	セッション
重大	HQLインジェクション: 「/owners」ページの「lastName」パラ...	6時間前	報告済	1
中	「MD5」ハッシュアルゴリズムを使用: GranteeManager	4時間前	報告済	2.1
注意	キャッシュ防止制御の欠如を検出	4時間前	報告済	2.1
注意	クリックジャッキング対策の制御がないページを検出	4時間前	報告済	2.1
注意	オートコンプリート防止のないフォームを検出	4時間前	報告済	2.1

深さ	脆弱性	最後の検出	ステータス	セッション
中	「MD5」ハッシュアルゴリズムを使用: GranteeManager	5時間前	報告済	2.1
注意	キャッシュ防止制御の欠如を検出	4時間前	報告済	2.1
注意	クリックジャッキング対策の制御がないページを検出	4時間前	報告済	2.1
注意	オートコンプリート防止のないフォームを検出	4時間前	報告済	2.1

補足



補足 - セッションメタデータ機能を使わない場合

セッションメタデータを使って特定の脆弱性インスタンスに遡る機能は、修正確認が簡単ではないケース、特に、同じ実行ルートに複数の脆弱性があるようなケースで有用です。

ここでは、そのような状況でセッションメタデータを使わないとどのようになるかを取り上げます。

1. 同じメソッドに脆弱なHQL呼び出しが2件ある状態でテスト（セッションメタデータ指定なし）
2. 2つの脆弱性のうち1つのみ修正してテスト（セッションメタデータ指定なし）

```
public Collection<Owner> findByLastName(String lastName) {
    System.out.println("Vulnerable method 1 - Two vulnerable queries");

    String lastName1 = lastName;
    String lastName2 = lastName;

    // クエリ1(脆弱) -- start
    String sqlQuery1 = "SELECT DISTINCT owner FROM Owner owner left join fetch owner.pets
        WHERE owner.lastName LIKE '" + lastName1 + "%'";
    TypedQuery<Owner> query1 = this.entityManager.createQuery(sqlQuery1, Owner.class);
    // クエリ1(脆弱) -- end

    // クエリ2(脆弱) -- start
    String sqlQuery2 = "SELECT DISTINCT owner FROM Owner owner left join fetch owner.pets
        WHERE owner.lastName LIKE '" + lastName2 + "%'";
    TypedQuery<Owner> query2 = this.entityManager.createQuery(sqlQuery2, Owner.class);
    // クエリ2(脆弱) -- end

    Collection result1 = query1.getResultList();
    Collection result2 = query2.getResultList();

    return result1;
}
```

```
public Collection<Owner> findByLastName(String lastName) {
    System.out.println("Vulnerable method 1 - Two vulnerable queries");

    String lastName1 = lastName;
    String lastName2 = lastName;

    // クエリ1(安全) -- start
    String sqlQuery1 = "SELECT DISTINCT owner FROM Owner owner left join fetch owner.pets
        WHERE owner.lastName LIKE :lastName";
    TypedQuery<Owner> query1 = this.entityManager.createQuery(sqlQuery1, Owner.class);
    query1.setParameter("lastName", lastName1 + "%");
    // クエリ1(安全) -- end

    // クエリ2(脆弱) -- start
    String sqlQuery2 = "SELECT DISTINCT owner FROM Owner owner left join fetch owner.pets
        WHERE owner.lastName LIKE '" + lastName2 + "%'";
    TypedQuery<Owner> query2 = this.entityManager.createQuery(sqlQuery2, Owner.class);
    // クエリ2(脆弱) -- end

    Collection result1 = query1.getResultList();
    Collection result2 = query2.getResultList();

    return result1;
}
```

補足 - セッションメタデータ機能を使わない場合

1. 脆弱性の詳細は変化せず (コード行の表示は変わらず)

2. 脆弱性インスタンスも1つのみ (時点を遡った詳細比較ができない)

petclinicdemo6
URL: / | 言語: Java | 重要性: 中

HQLインジェクション: 「/owners」ページの「lastName」パラメータ
重大 | 日付: 12/15/2022 12:21 午後 | ステータス: 報告済 | ID: L07V-6LIQ-2562-NXB7

概要 詳細 HTTP情報 修正方法 備考 アクティビティ

HTTPのパラメータを取得
string() = facade.getParameterValues("lastName")
lastName = davis
getParametersStartingWith() @ WebUtils.java:672

文字列操作が発生
このコード行で
findByLastName() @ OwnerRepositoryCustomImpl.java:26
SELECT DISTINCT owner FR...E owner.lastName LIKE 'davis%'

ルール違反の検出
query = impl.createQuery("SELECT DISTINCT owner FROM Owner owner left joi...")
createQuery() @ AbstractEntityManagerImpl.java:305
SELECT DISTINCT owner FR...E owner.lastName LIKE 'davis%'

petclinicdemo6
URL: / | 言語: Java | 重要性: 中

HQLインジェクション: 「/owners」ページの「lastName」パラメータ
重大 | 日付: 12/15/2022 12:21 午後 | ステータス: 報告済 | ID: L07V-6LIQ-2562-NXB7

概要 脆弱性 攻撃 ライブラリ アクティビティ ルートカバレッジ フローマップ ポリシー

検索に戻る 5件中1件目

HQLインジェクション: 「/owners」ページの「lastName」パラメータ
重大 | 日付: 12/15/2022 12:21 午後 | ステータス: 報告済 | ID: L07V-6LIQ-2562-NXB7

概要 詳細 HTTP情報 修正方法 備考 アクティビティ

最初の検出 30分前
最後の検出 3分前

アプリケーションのバージョン
アプリケーションのバージョンはありません。

次のサーバにより報告

- SSHOTA-C02G90HEMD6R (数: 1 最後の検出 30分前)
報告されている最新版 - 4.0.0
- OWASP Top Ten - 2017 セキュリティ基準: A1
- IPA - 7.0 セキュリティ基準: 1

脆弱性インスタンス

- L07V-6LIQ-2562-NXB7 (33分前)

このように、セッションメタデータを使わないと、施した修正がうまくいっていないのか、それとも今回の修正自体はうまくいったものの他にも脆弱性がある状況なのか、切り分けがしづらくなります。

さまざまな状況で円滑に脆弱性対応ができるよう、セッションメタデータ機能を活用することが推奨されます。