CONTRAST SECURITY

パス(ディレクトリ)トラバーサルの解説と対策

この動画で説明すること

パストラバーサルの説明と脆弱性の修正

1. パストラバーサルが起こる原因
2. Contrast Assessで検知
3. 検査パターンでパストラバーサルを確認
4. パストラバーサルを修正
5. Contrast Assessで修正を確認
6. 検査パターンでパストラバーサルの修正を確認

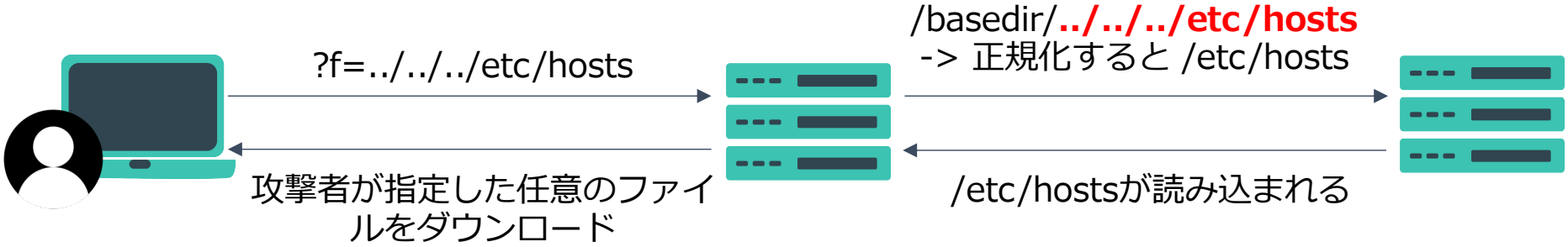
3,6については任意です


パストラバーサルの原因

パストラバーサルの原因 (1/2)

パストラバーサルの概要

パストラバーサルはファイル操作(ダウンロード/アップロード)の際に外部の入力値をそのまま使用しているが原因で発生します。



ファイルアップロードも同様に、攻撃者の指定したファイル名をそのまま使用すると、スクリプトファイルなどをアップロードされそのまま任意のコードを実行される可能性があります。

圧縮ファイルの展開後のファイル名を使用したファイル操作も同様です。

パストラバーサルの原因 (2/2)

原因はユーザ入力のファイルパスをそのまま使用しているため

以下のJavaのコードはパストラバーサルの脆弱性があります。

```
final String STATEMENT_DIR = "/safe_directory/";
String statement = request.getParameter("statement");
// 指定したファイルの読込み
File file = new File(STATEMENT_DIR, statement);
FileInputStream fis = new FileInputStream(file);
byte[] fileBytes = new byte[file.length()];
fis.read(fileBytes);
response.getOutputStream().write(fileBytes);
```

先ほどのページのように、「../../../」といったパスを遡るような入力を受け付けた場合、本来想定しているパスとは異なるパスのファイルを読み込んだり、書き込んだりしてしまいます。


Contrast Assessでの検知

パストラバーサルの検知(1/2)

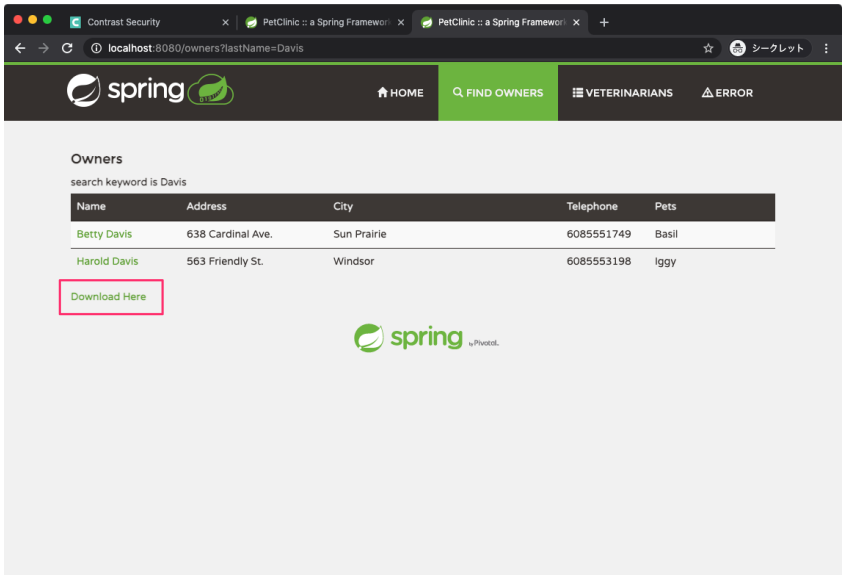
実際にContrast Assessを用いて検知

脆弱性のあるアプリケーションにContrastエージェントを組み込んで、アプリケーションを起動します。

その後、アプリケーションを操作し脆弱性を検知します。

ファイルダウンロード機能をテストします

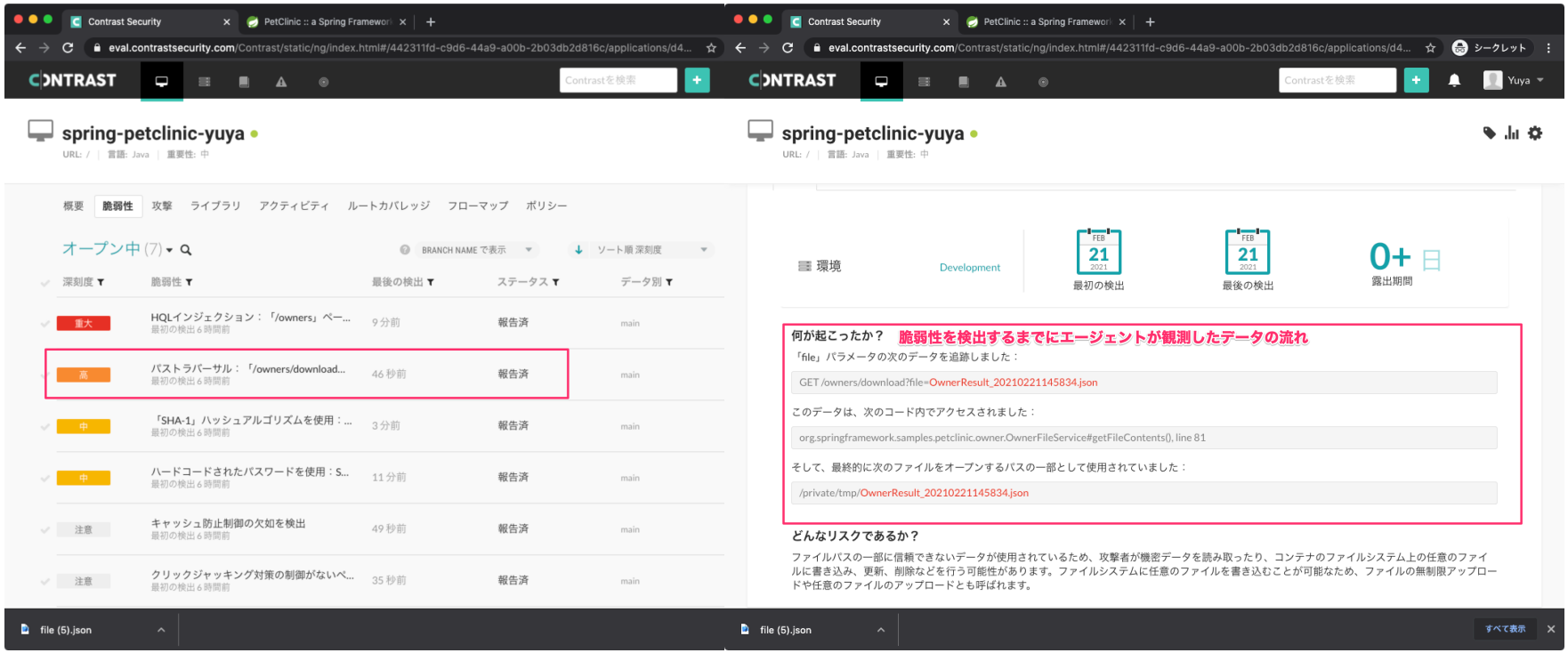
DASTなどで使用される攻撃文字列を使用する必要はありません。



パストラバーサルの検知(2/2)

Contrast Assessで検知した結果

Contrast UIにアクセスし検知した脆弱性を確認します。




アプリケーションでの確認

検査パターンで確認(1/2)

実際にExploitableか確認

検査パターンベースでアプリケーションにパストラバーサルが存在するかは主に下記の観点で確認します。

観点	入力値
本来公開されていないファイルを取得できないか	1. ../../../../../../../../ect/hosts 2. ../../../../../../../../etc/hosts%00 3. ../../../../../../../../etc/hosts%2500 4. ../../../../../../../../windows/win.ini 5. ../../../../../../../../windows/win.ini%00 6. ../../../../../../../../windows/win.ini%2500

※%00はNULL文字をURLエンコードした値です。

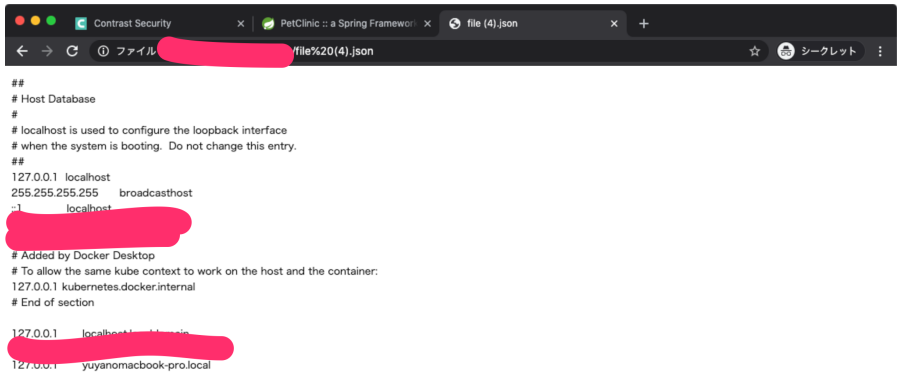
NULL文字を挿入した場合、プログラミング言語やそのバージョンによってはNULL文字列を挿入されることにより、拡張子の検証を迂回される場合があります。

検査パターンで確認(2/2)

ファイルダウンロードで確認

Contrast Assessがパストラバーサルを検知した機能を前ページの観点で確認してみます。

「../../../../etc/hosts」を指定した場合、
本来はダウンロードを想定していない
「hosts」ファイルがダウロードされます。




パストラバーサルを修正

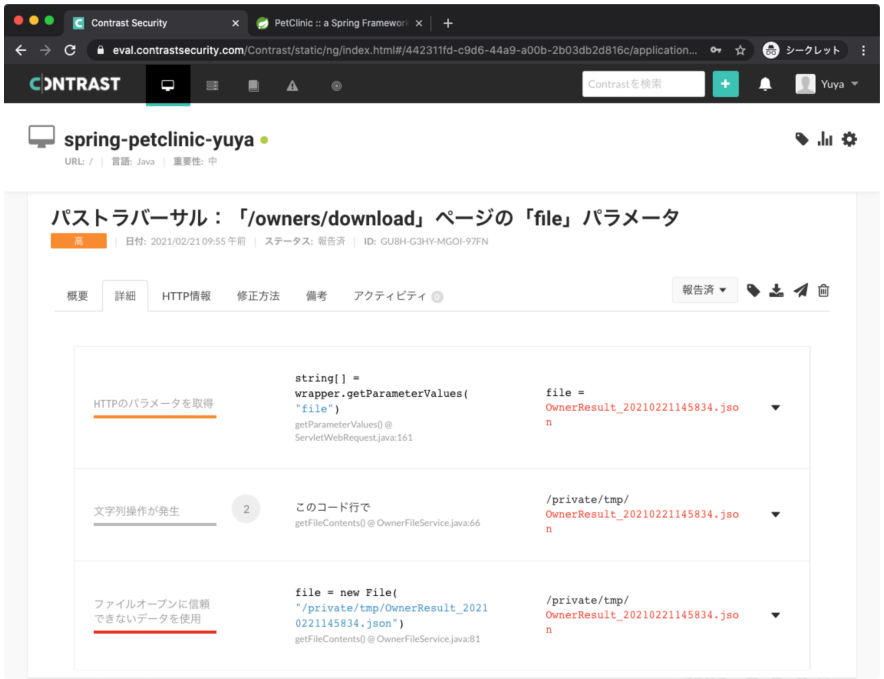
パストラバーサルを修正(1/3)

Contrast UIを確認し修正箇所を特定

脆弱性の詳細タブを確認すると、Contrast Assessのエージェントが観測したイベントを確認できます。

ユーザの入力値をそのまま使用して、ファイルパスの構築を行っていることがわかります。

その後、構築したパスを使用してファイルオープンしているしているため、脆弱性を検知しています。



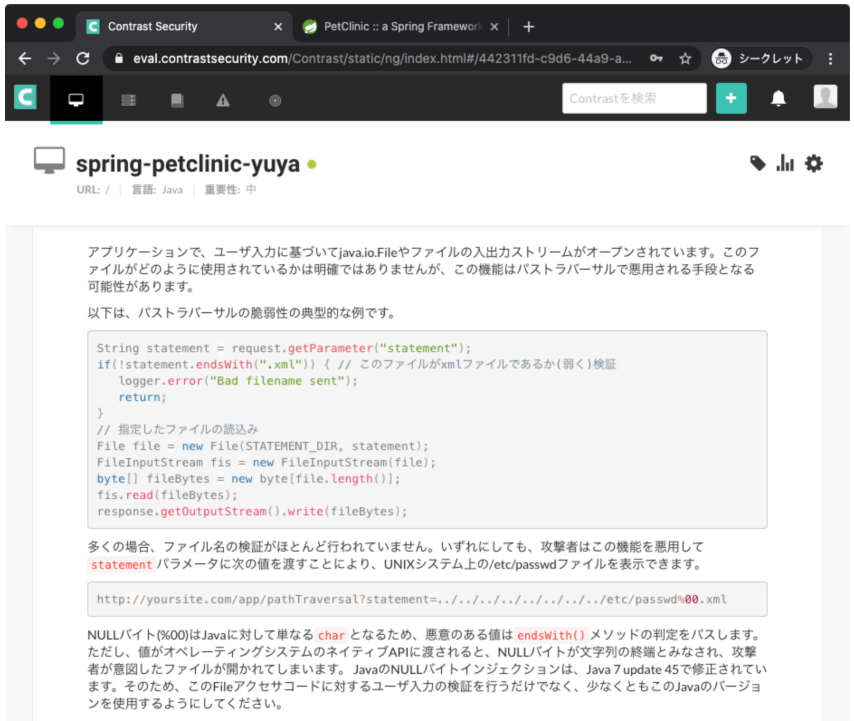
パストラバーサルを修正(2/3)

修正方法を確認

ファイルパスを遡る「../」が入り込まない、評価しない仕組みを実装する必要があります。

それに基づき、パストラバーサルの修正には下記の方法が考えられます。

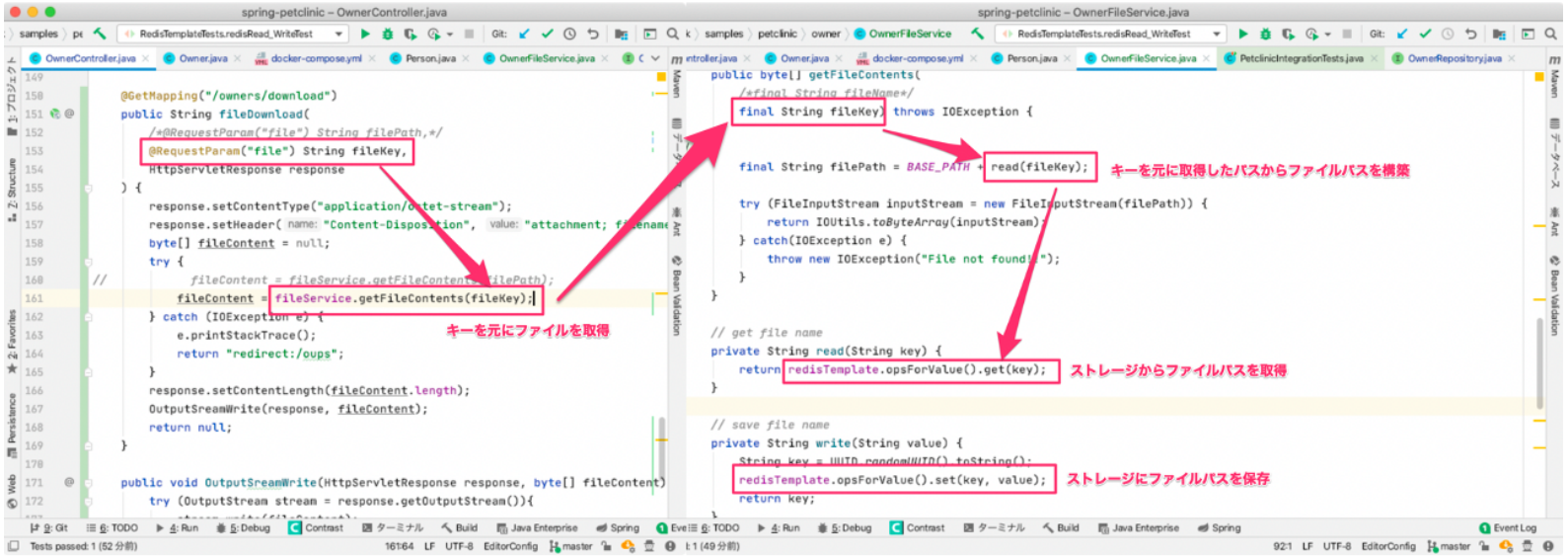
1. ファイルパスを指定せず、ファイルパスに紐づくキーを指定する
2. ファイル名を英字のみにする
3. 入力されたファイル名をBasenameで評価する
4. パスを正規化後、ベースパスが変わっていないことを検証する



パストラバーサルを修正(3/3)

コードを修正

確認した修正方法でパストラバーサルを修正していきます。



ユーザに直接ファイルパスを操作しないよう修正しました。


Contrast Assessで修正を確認

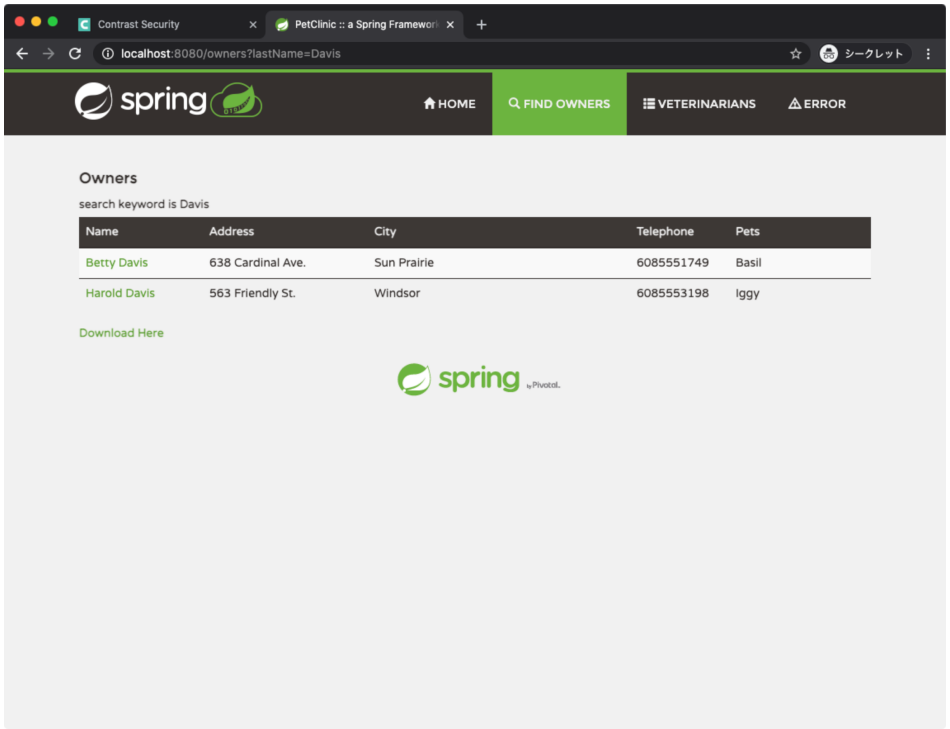
Contrastで修正を確認(1/2)

脆弱性が修正されているかを確認

脆弱性を修正後に再ビルドを行い、エージェントを組み込みアプリケーションを起動します。

脆弱性検出時と同じ操作を行います。

リクエストを再現する場合、HTTP情報タブにある「リクエストを再生」機能からも実施可能です。



Contrastで修正を確認(2/2)

脆弱性が修正されているかを確認

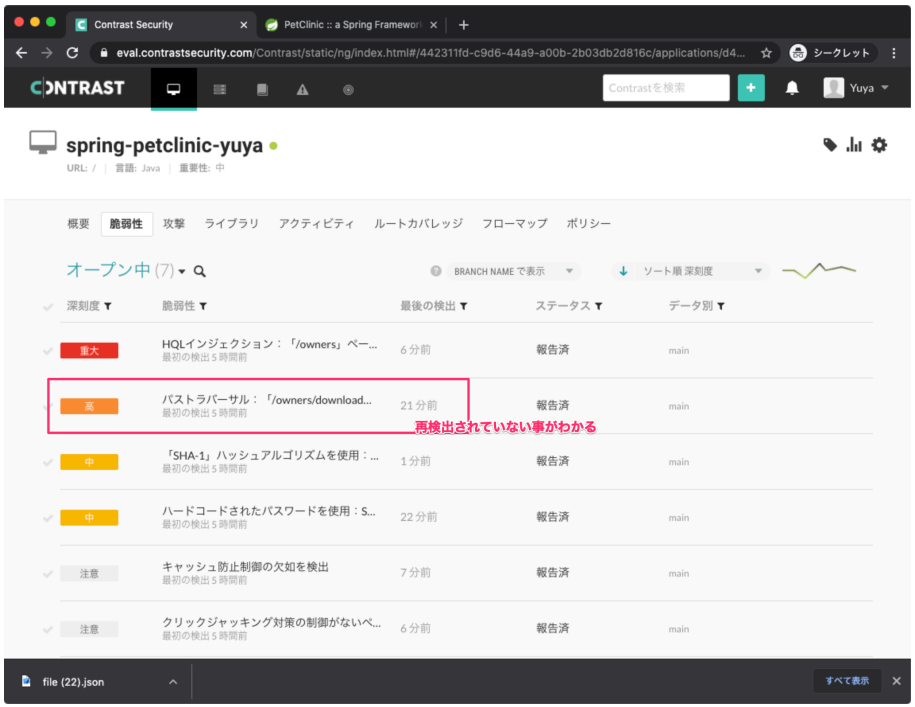
脆弱性を修正後に再ビルドを行い、エージェントを組み込みアプリケーションを起動します。

パストラバーサルの「最後の検出」が更新されていないことがわかります。

脆弱性が修正されたかどうかを確認する方法はいくつかあります。

- session_metadataを使用する
- アプリケーションのバージョンを指定する

などの方法があります。




検査パターンで修正を確認

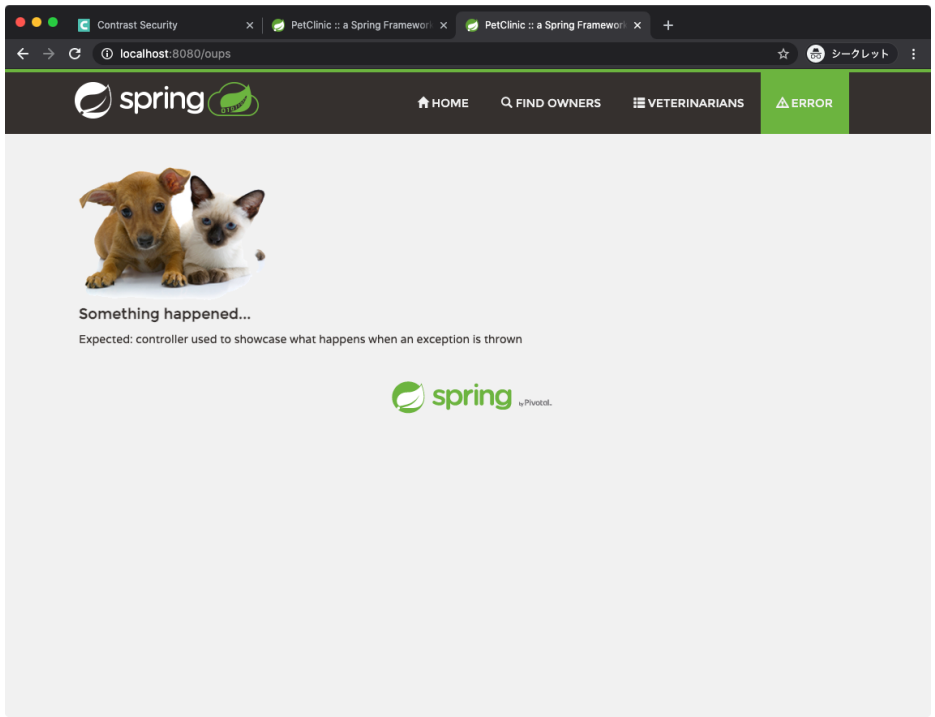
検査パターンで修正を確認(1/1)

ファイルダウンロードで確認

検索機能に存在していたパストラバーサルが修正されているかを検査パターンベースで確認します。

検出時と同様に「../../../etc/hosts」を入力し検索します。

修正前にダウンロードできていたhostsファイルがダウンロードできなくなっていることが確認できます。




まとめ

まとめ

パストラバーサルの原因と対策

1. パストラバーサルが起こる原因

   ユーザーの入力値をそのまましようしてファイルパスを構築しているため。

2. 脆弱性の検出

   Contrast Assessを使用することで、UIを操作するだけで脆弱性を検出できます。

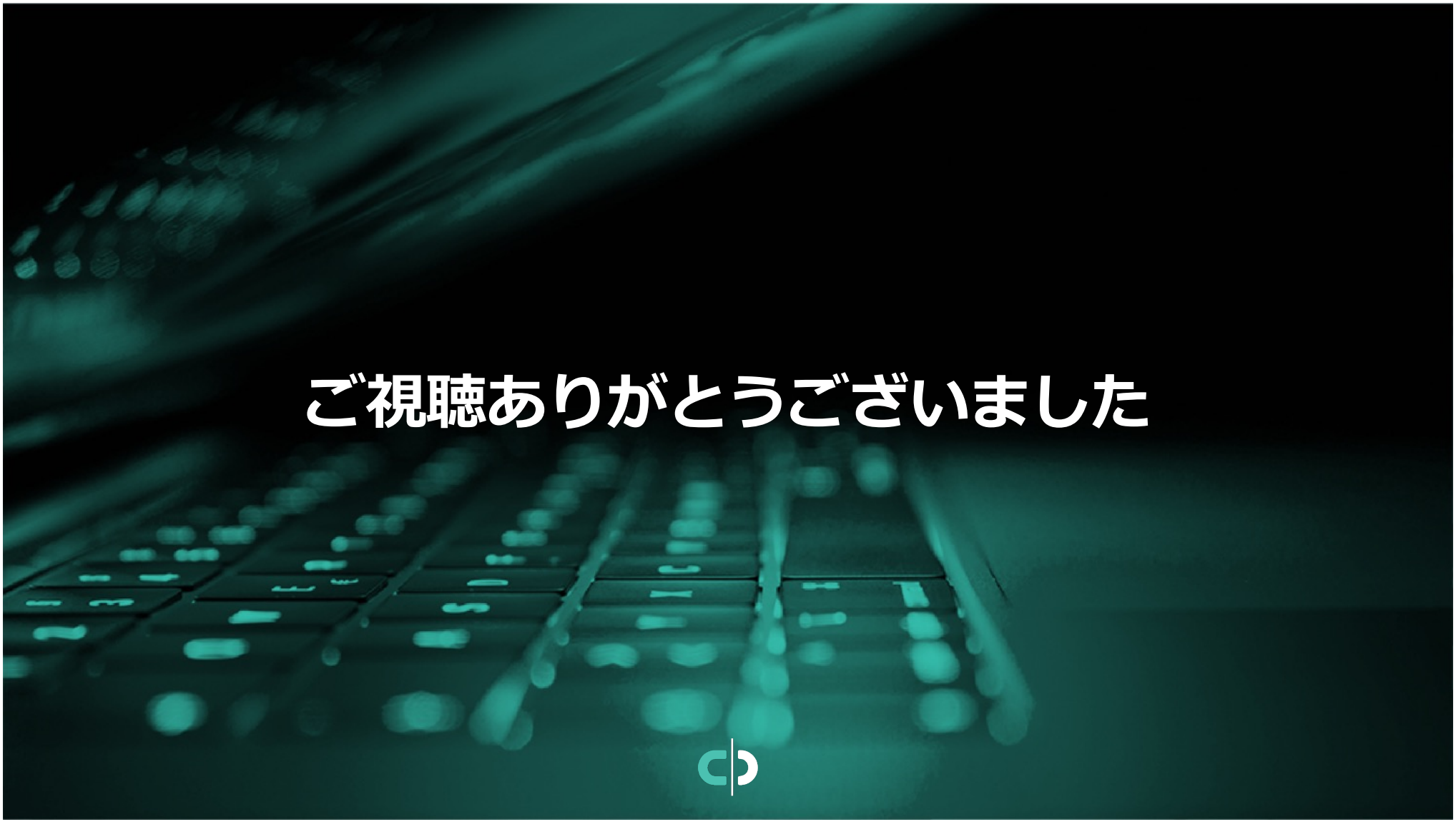
3. 脆弱性の修正

   Contrastが観測した情報、修正方法のサンプルコードをもとに脆弱性を修正し、検出時と同じ操作を実施することで脆弱性の対策が実施されていることを確認できます。

この動画で使用した素材

アプリケーションは下記のソースコードに脆弱性を追加したものを使用しております。本来のコードに脆弱性は含まれておりません。

https://github.com/spring-projects/spring-petclinic


ご視聴ありがとうございました