

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



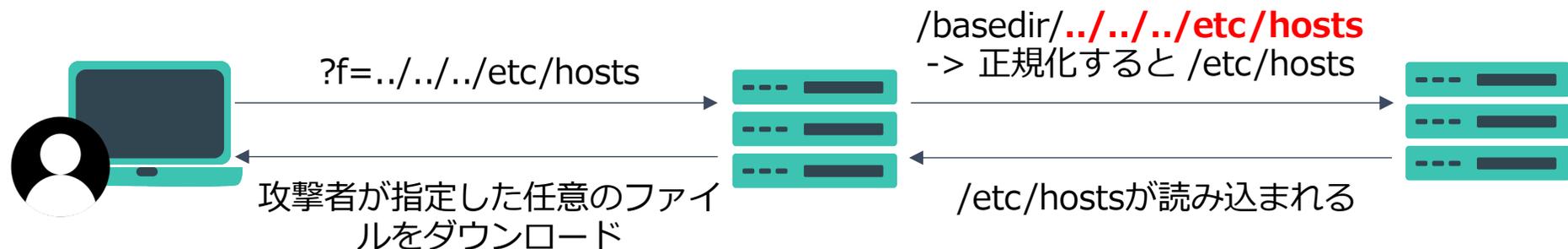
パストラバーサルの原因



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み込み
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)

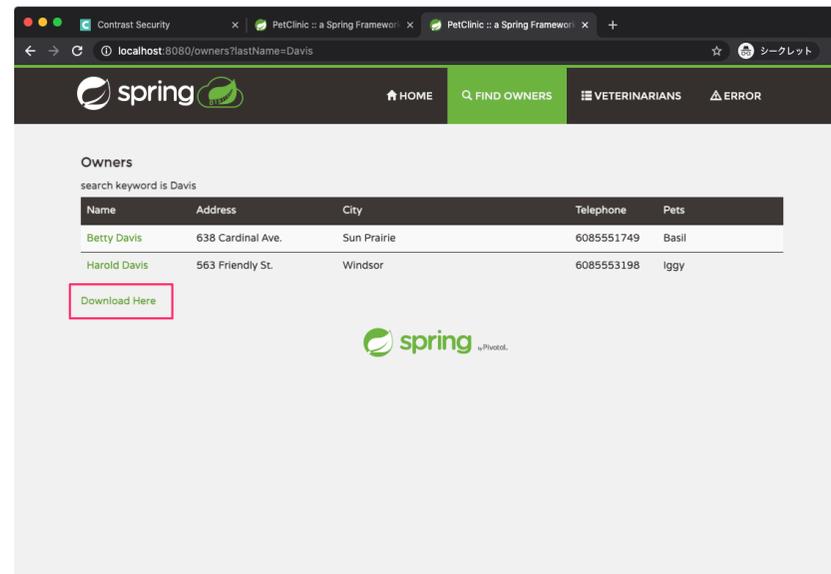
実際にContrast Assessを用いて検知

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。

The screenshot displays the Contrast Security interface for a project named 'spring-petclinic-yuya'. The main view shows a list of detected vulnerabilities. One vulnerability is highlighted with a red box:

深堀度	脆弱性	最後の検出	ステータス	データ別
重大	HQLインジェクション: 「owners」ペ...	9分前	報告済	main
高	パストラバーサル: 「owners/download...	46秒前	報告済	main
中	「SHA-1」ハッシュアルゴリズムを使用...	3分前	報告済	main
中	ハードコードされたパスワードを使用...	11分前	報告済	main
注意	キャッシュ防止制御の欠如を検出	49秒前	報告済	main
注意	クリックジャッキング対策の制御がない...	35秒前	報告済	main

The detailed view of the highlighted vulnerability shows the following information:

- 環境:** Development
- 最初の検出:** FEB 21 2021
- 最後の検出:** FEB 21 2021
- 露出期間:** 0+ 日

何が起ったか? 脆弱性を検出するまでにエージェントが観測したデータの流れ

「file」パラメータの次のデータを追跡しました:

```
GET /owners/download?file=OwnerResult_20210221145834.json
```

このデータは、次のコード内でアクセスされました:

```
org.springframework.samples.petclinic.owner.OwnerFileService.getFileContents(), line 81
```

そして、最終的に次のファイルをオープンするパスの一部として使用されていました:

```
/private/tmp/OwnerResult_20210221145834.json
```

どんなリスクであるか?

ファイルパスの一部に信頼できないデータが使用されているため、攻撃者が機密データを読み取ったり、コンテナのファイルシステム上の任意のファイルに書き込み、更新、削除などを行う可能性があります。ファイルシステムに任意のファイルを書き込むことが可能なため、ファイルの無制限アップロードや任意のファイルのアップロードとも呼ばれます。



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	<ol style="list-style-type: none">1. ../../../../../../ect/hosts2. ../../../../../../etc/hosts%003. ../../../../../../etc/hosts%25004. ../../../../../../windows/win.ini5. ../../../../../../windows/win.ini%006. ../../../../../../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。

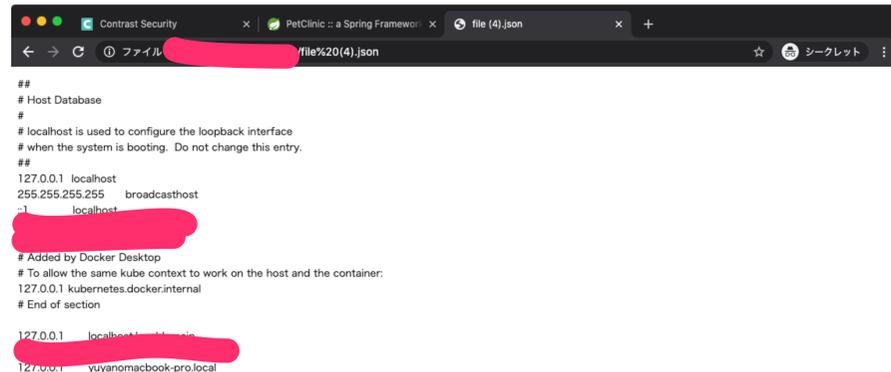


検査パターンで確認(2/2)

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」ファイルがダウンロードされます。



```
##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting. Do not change this entry.
##
127.0.0.1 localhost
255.255.255.255 broadcasthost
-1 localhost
# Added by Docker Desktop
# To allow the same kube context to work on the host and the container:
127.0.0.1 kubernetes.docker.internal
# End of section

127.0.0.1 localhost
127.0.0.1 yuyanomacbook-pro.local
```



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ため、脆弱性を検知しています。

The screenshot shows the Contrast Security web interface. The browser address bar displays the URL: `eval.contrastsecurity.com/Contrast/static/ng/index.html#/442311fd-c9d6-44a9-a00b-2b03db2d816c/application...`. The page title is "spring-petclinic-yuya". The main content area displays a vulnerability report titled "パストラバーサル: 「/owners/download」ページの「file」パラメータ". The report includes the following details:

- Severity:** 高 (High)
- Date:** 2021/02/21 09:55 午前
- Status:** 報告済
- ID:** GU8H-G3HY-MGOI-97FN

The report lists three events:

Event	Code Snippet	File Path
HTTPのパラメータを取得	<pre>string[] = wrapper.getParameterValues("file")</pre>	<code>file = OwnerResult_20210221145834.json</code>
文字列操作が発生	<pre>getFileContents() @ OwnerFileService.java:66</pre>	<code>/private/tmp/OwnerResult_20210221145834.json</code>
ファイルオープンに信頼できないデータを使用	<pre>file = new File("/private/tmp/OwnerResult_20210221145834.json")</pre>	<code>/private/tmp/OwnerResult_20210221145834.json</code>



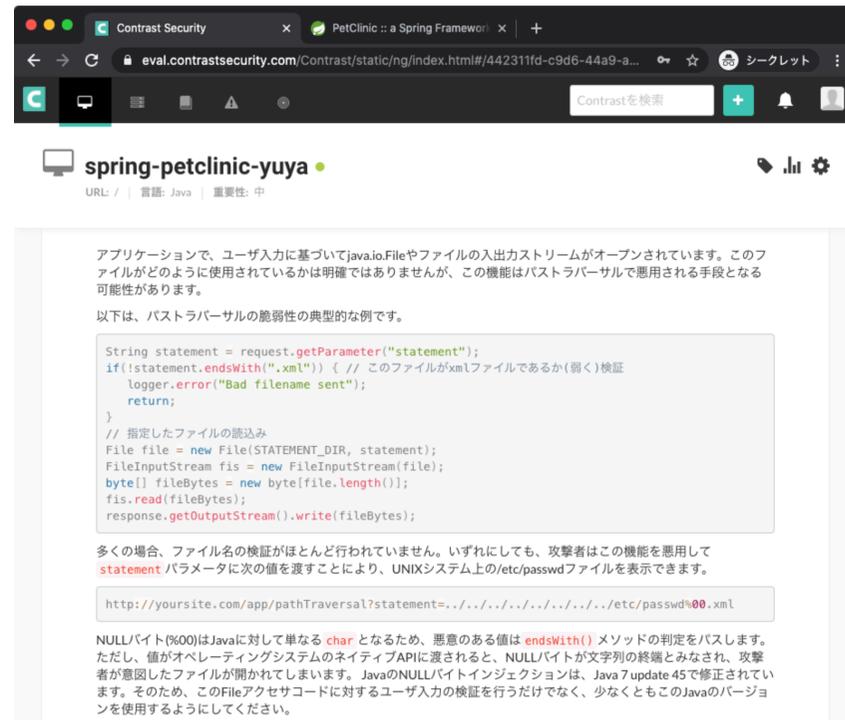
パストラバーサルを修正(2/3)

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



The screenshot shows a browser window with the URL `eval.contrastsecurity.com/Contrast/static/ng/index.html/#/442311fd-c9d6-44a9-a...`. The page title is "spring-petclinic-yuya". The main content area displays a security warning in Japanese: "アプリケーションで、ユーザ入力に基づいてjava.io.Fileやファイルの入出力ストリームがオープンされています。このファイルがどのように使用されているかは明確ではありませんが、この機能はパストラバーサルで悪用される手段となる可能性があります。以下は、パストラバーサルの脆弱性の典型的な例です。"

```
String statement = request.getParameter("statement");
if(!statement.endsWith(".xml")) { // このファイルがxmlファイルであるか(弱く)検証
    logger.error("Bad filename sent");
    return;
}
// 指定したファイルの読み込み
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);
```

多くの場合、ファイル名の検証がほとんど行われていません。いずれにしても、攻撃者はこの機能を悪用して `statement` パラメータに次の値を渡すことにより、UNIXシステム上の/etc/passwdファイルを表示できます。

```
http://yoursite.com/app/pathTraversal?statement=../../../../../../../../etc/passwd%00.xml
```

NULLバイト(%00)はJavaに対して単なる `char` となるため、悪意のある値は `endsWith()` メソッドの判定をパスします。ただし、値がオペレーティングシステムのネイティブAPIに渡されると、NULLバイトが文字列の終端とみなされ、攻撃者が意図したファイルが開かれてしまいます。JavaのNULLバイトインジェクションは、Java 7 update 45で修正されています。そのため、このFileアクセスコードに対するユーザ入力の検証を行うだけでなく、少なくともこのJavaのバージョンを使用するようにしてください。



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。

```
spring-petclinic - OwnerController.java
149
150 @GetMapping("/owners/download")
151 public String fileDownload(
152     /*@RequestParam("file") String filePath,*/
153     @RequestParam("file") String fileKey,
154     HttpServletResponse response
155 ) {
156     response.setContentType("application/octet-stream");
157     response.setHeader( name: "Content-Disposition", value: "attachment; filename=");
158     byte[] fileContent = null;
159     try {
160         // fileContent = fileService.getFileContents(filePath);
161         fileContent = fileService.getFileContents(fileKey);
162     } catch (IOException e) {
163         e.printStackTrace();
164         return "redirect:/oops";
165     }
166     response.setContentLength(fileContent.length);
167     OutputStreamWrite(response, fileContent);
168     return null;
169 }
170
171 public void OutputStreamWrite(HttpServletResponse response, byte[] fileContent)
172     try (OutputStream stream = response.getOutputStream()){
173         stream.write(fileContent);
174     }

spring-petclinic - OwnerFileService.java
public byte[] getFileContents(
    /*final String fileName*/
    final String fileKey) throws IOException {
    final String filePath = BASE_PATH + read(fileKey);
    try (FileInputStream inputStream = new FileInputStream(filePath)) {
        return IOUtils.toByteArray(inputStream);
    } catch (IOException e) {
        throw new IOException("File not found: " + e.getMessage());
    }
}

// get file name
private String read(String key) {
    return redisTemplate.opsForValue().get(key);
}

// save file name
private String write(String value) {
    String key = UUID.randomUUID().toString();
    redisTemplate.opsForValue().set(key, value);
    return key;
}
```

キーを元にファイルを取得

キーを元に取得したパスからファイルパスを構築

ストレージからファイルパスを取得

ストレージにファイルパスを保存

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



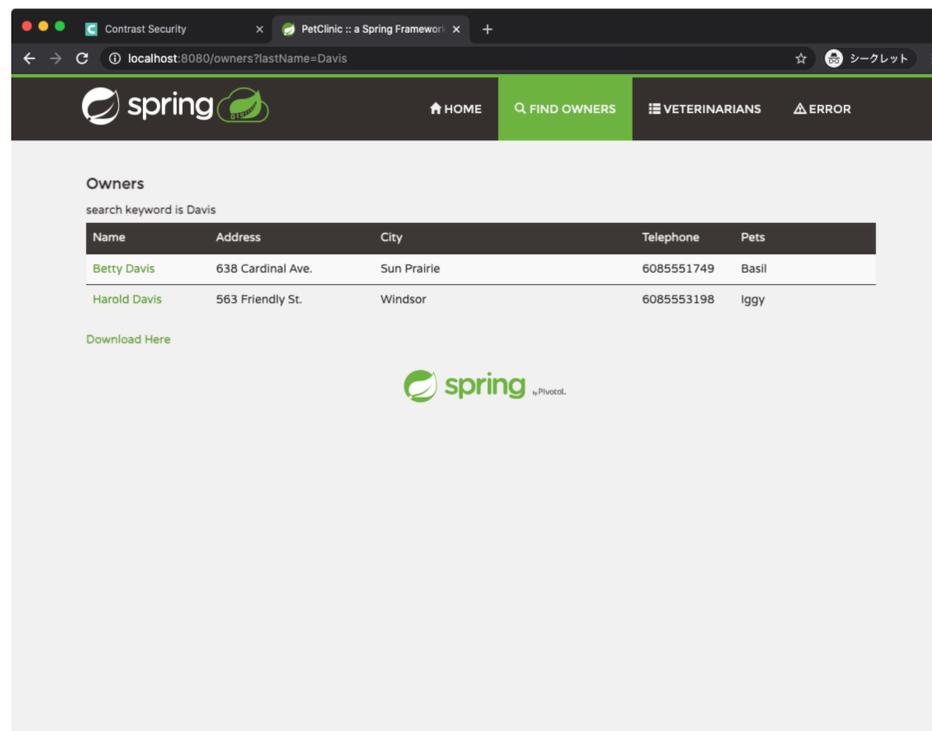
Contrastで修正を確認(1/2)

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「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。

The screenshot shows the Contrast Security dashboard for 'spring-petclinic-yuya'. The '脆弱性' (Vulnerabilities) tab is active, displaying a table of detected issues. The table has columns for severity, description, last detected time, status, and data type. The following table represents the data shown in the image:

深刻度	脆弱性	最後の検出	ステータス	データ別
重大	HQLインジェクション: 「/owners」ペー...	6分前	報告済	main
高	パストラバーサル: 「/owners/download...	21分前	報告済	main
中	「SHA-1」ハッシュアルゴリズムを使用: ...	1分前	報告済	main
中	ハードコードされたパスワードを使用: S...	22分前	報告済	main
注意	キャッシュ防止制御の欠知を検出	7分前	報告済	main
注意	クリックジャッキング対策の制御がないべ...	6分前	報告済	main

A red box highlights the 'High' severity vulnerability 'パストラバーサル: 「/owners/download...」'. A red note next to it says '再検出されていない事がわかる' (It can be seen that it has not been re-detected).



検査パターンで修正を確認



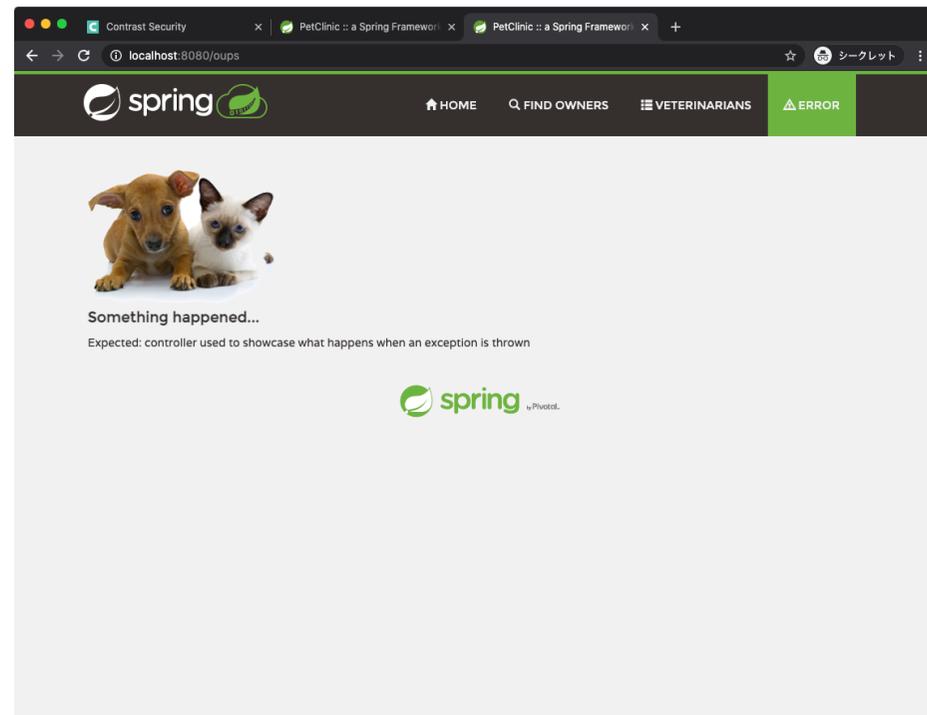
検査パターンで修正を確認(1/1)

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

