**Target**

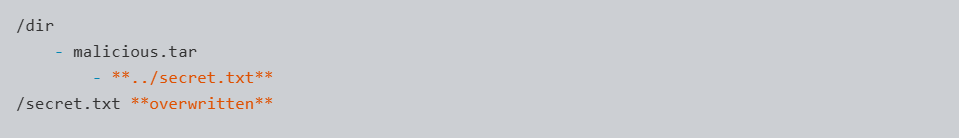
java-security-toolkit ≤ 1.1.1

**Explain**

Java Code Secutiry Toolkit은 Java로 개발할 때 발생할 수 있는 다양한 범주의 취약점들을 시큐어코딩을 통해 예방할 수 있도록 입력 데이터의 검증이나 필터링 등의 기능을 하는 API를 제공합니다.

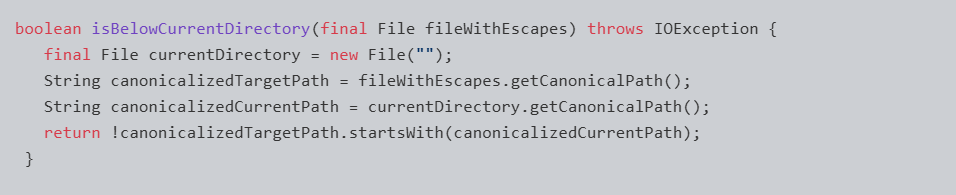
개발자는 이러한 API를 사용해서 사용자로부터 전달된 데이터를 필터링하거나 ObjectInputStream을 통한 역직렬화처럼 매우 취약한 기존 방식을 대체할 수 있습니다.

예를 들어 HtmlEncoder.java에서는 텍스트에 포함된 \t, < 등을 HTML 엔티티로 변환시켜 XSS나 CSRF 등의 취약점을 방지하는데 사용할 수 있는 API를 제공하고 이번에 취약점이 발견된 ZipSecurity.java는 Zip slip 공격을 검사하는 API를 제공합니다.



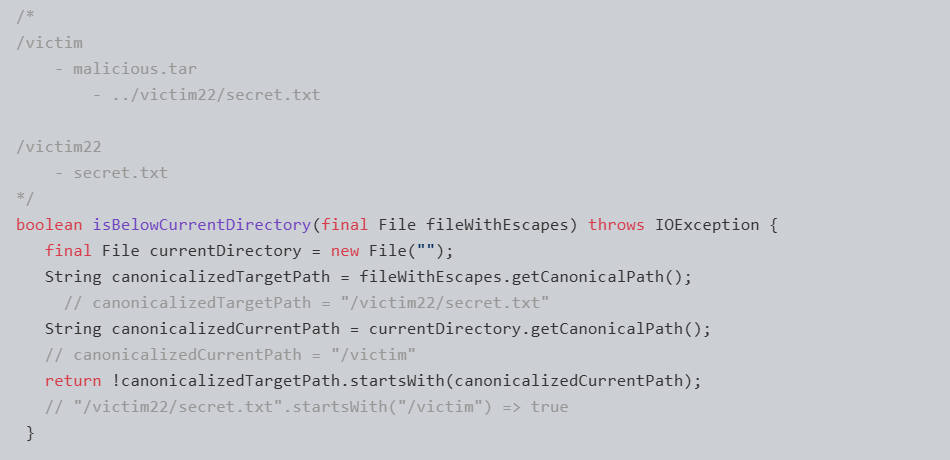
Zip slip이란 .zip ,.tar 등의 아카이브 파일에 위처럼 path traversal 구문이 포함되어 압축이 해제될 때 해당 경로의 파일을 덮어 쓰는 공격으로 malicious.tar을 dir에 압축 해제하게 되면 ../secret의 path traversal 구문에 의해 /secret.txt가 덮어 쓰일 것입니다.

ZipSecutiry.java에서는 zip slip을 감지하기 위해 아래의 isBelowCurrentDirectory 메소드를 통해 압축 해제될 경로가 현재 디렉터리와 일치하는지 확인했습니다.



정확히는 아카이브 파일에 포함된 파일의 경로 문자열이 현재 디렉터리로 시작(startsWith)하는지 검사하는 방식을 사용했고 getCanonicalPath를 통해 ./나 ../ 가 정리된 파일의 압축 해제 경로와 현재 디렉터리의 경로를 얻었습니다.

문제는 getCanonicalPath가 반환하는 경로 문자열은 맨 끝에 /가 없기 때문에 아래처럼 현재 디렉터리와 앞부분은 똑같은 형제 디렉터리가 있다면 이곳으로의 path traversal은 탐지하지 못합니다



따라서 아카이브 파일을 추출할 디렉터리의 이름 + @인 형제 디렉터리가 있다면 해당 디렉터리로 path traversal이 가능합니다.

이 취약점은 현재는 아래와 같이 String 클래스의 startsWith이 아닌 Path 클래스의 startsWith 메소드를 사용하도록 패치되었습니다.

private boolean isBelowOrSisterToCurrentDirectory(final String untrustedFileWithEscapes) throws IOException {

// Get the absolute path of the current directory

final File currentDirectory = new File("").getCanonicalFile();

final Path currentPathRoot = currentDirectory.toPath();

// Get the absolute path of the untrusted file

final File untrustedFile = new File(currentDirectory, untrustedFileWithEscapes);

final Path pathWithEscapes = untrustedFile.getCanonicalFile().toPath();

return !pathWithEscapes.startsWith(currentPathRoot);

}