Dependency Check

-The open source vulnerability scanner

Expectations.

- History of Dependency Check
- Importance of Dependency Check.
  - Why to care about the Dependencies which we use in our daily coding.
- To understand what is Dependency Checker by
  - Supported Languages/tech.
  - Relation to OWASP top 10.
  - Reviewing How it works.
    - Vulnerability Data Source.
    - Library Identification and issues.
    - Evidence based identification, issues and Remediation.
  - Using Dependency Check.
    - Components of Dependency Check.
    - Use Cases of Dependency Check.
    - Enterprise Deployments.
  - How to read the reports.
  - Demo.

History of Dependency Check

- Dependency-Check is a Software Composition Analysis (SCA) tool that attempts to detect publicly disclosed vulnerabilities contained within a projects dependencies. It does this by determining if there is a Common Platform Enumeration (CPE) identifier for a given dependency. If found, it will generate a report linking to the associated CVE entries.

Importance of Dependency Check

- **95% of applications include open source**
- **67% of applications contained open source vulnerabilities**
- **Average age of open source vulnerability identified: 1,894 days**

- **CVE-2018-2815 – JAVA SE DOS via Serialization.**
- **CVE-2016-5000 -** Apache POI Information Disclosure via External Entity Expansion (XXE)
- **CVE-2016-4216 -** Adobe XMP Toolkit for Java Information Disclosure via External Entity Expansion (XXE)
- **CVE-2016-3081 -** Remote code execution vulnerability in Apache Struts when dynamic method invocation is enabled
- **CVE-2015-8103 -** Remote code execution vulnerability in Jenkins remoting; related to the Apache commons-collections

Understanding Dependency Check
- Supported Languages/tech.

- Fully supported: Java & .NET
- Experimental Analyzers:
  - CocoaPods
  - Swift Package Manager
  - Python
  - PHP (composer)
  - Node.js
  - Ruby

Relation to OWASP top 10.

- Most critical web application risks
- A9 – Using components with known vulnerabilities
  - Prevalence: Widespread
  - Detectability: Difficult
- Difficult for 4 reasons
  - Awareness
  - Visibility
  - Lack of tooling in 2012/2013

How it works.

Vulnerability Data Source.

- National Vulnerability Database (NVD)
  - https://nvd.nist.gov
- Contains a listing of Common Vulnerability and Exposures (CVE)
- Each CVE entry contains
  - A description of the vulnerability or exposure
  - A Common Vulnerability Scoring System (CVSS) score
  - A list of the affected platforms identified by their Common Platform Enumeration (CPE)

Library Identification and issues.

- Identification :

  Reporting on known/published vulnerabilities requires the correct identification of the libraries used
- Issues :
  - Development & Security use different identifiers
  - Development (GAV coordinates):
    - org.springframework:spring-core:3.2.0.RELEASE
  - Security uses Common Platform Enumeration (CPE):
    - cpe:/a:springsource:spring_framework:3.2.0
    - cpe:/a:pivotal:spring_framework:3.2.0
    - cpe:/a:pivotal_software:spring_framework:3.2.0
  - No publicly available database exists to map between the two

Evidence based identification, issues and Remediation.

- Identification :
  - Evidence is extracted from dependencies
    - File name, manifest, POM, package names, etc.
    - Evidence is grouped into Vendor, Product, and Version collections
  - Local copy of the NVD CVE is maintained
  - Lucene Index of the CPE information is created
  - Evidence collected is used to search the index and identify the library by CPE

Evidence based identification, issues and Remediation.

- Issues :
  - False Positives
    - Evidence extracted may cause incorrect identification
  - False Negatives
    - If key elements are not included in the dependency (e.g. jar, dll) the library will not be identified and may result in un-reported risk

Evidence based identification, issues and Remediation.

- Remediation

  Invalid dependency identification can be resolved using a suppression file:

```
<suppress>
   <notes><![CDATA[
   This suppresses false positives identified on spring security.
   ]]></notes>
   <gav regex="true">org\.springframework\.security:spring.*</gav>
   <cpe>cpe:/a:mod_security:mod_security</cpe>
   <cpe>cpe:/a:springsource:spring_framework</cpe>
   <cpe>cpe:/a:vmware:springsource_spring_framework</cpe>
</suppress>
```

  More Details.

Using Dependency Check.

Components of Dependency Check

- Command line interface
- a Maven plugin
- an Ant task
- Jenkins plugin
- Core engine containing series of analyzers.
  - Steps to run
    - Extract the bat file obtained from link.
    - Go to bin.
    - Execute the command :

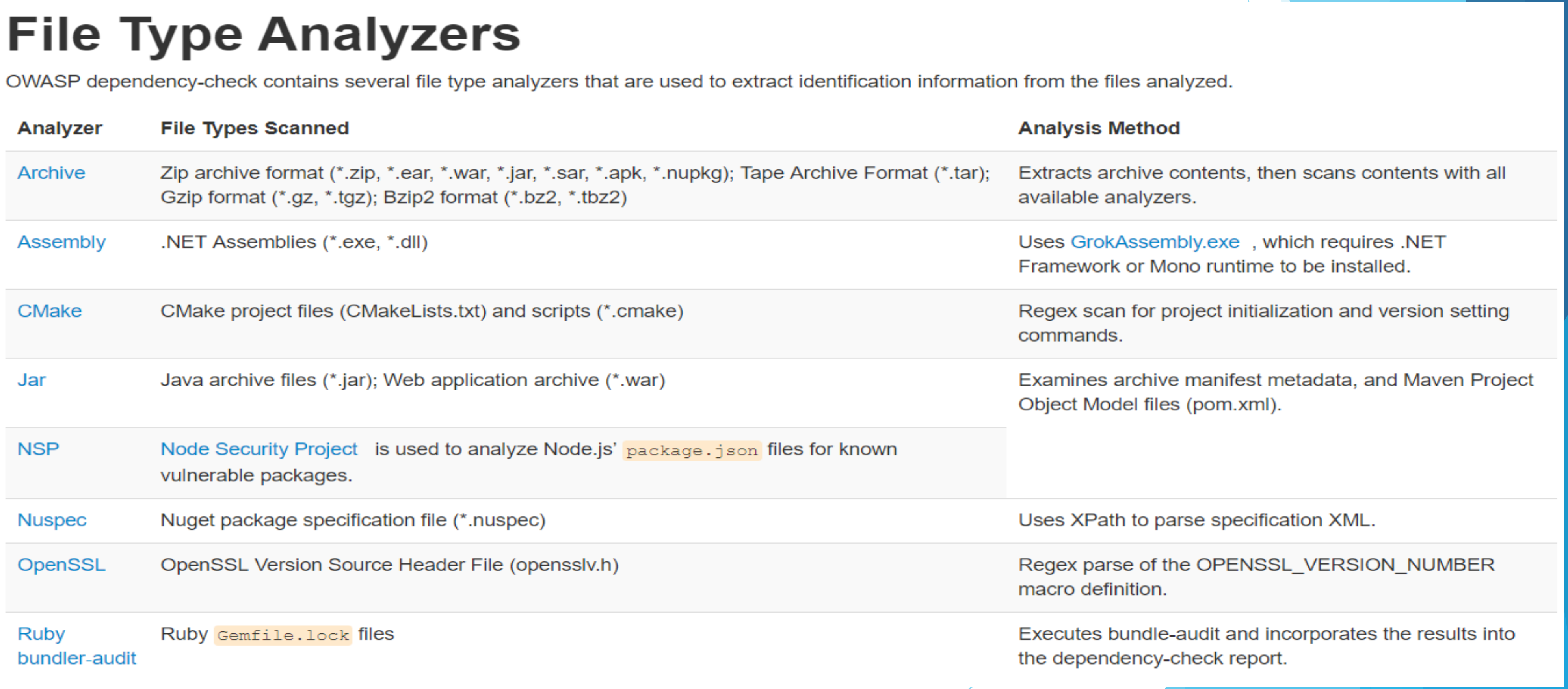
```
Dependency.bat  --format <HTML or PDF> --out "<Location for extracting report>" --scan "<location of jar/dependent files>" --project <name of report.>
```

e.g.

```
Dependency.bat  --format HTML --out "C:\Users\Administrator\Desktop\Security Testing" --scan "C:\Users\Administrator\Desktop\Security Testing\*.*" --project SecurityScannerToolCommand
```

Using Dependency Check.

Components of Dependency Check



File Type Analyzers

OWASP dependency-check contains several file type analyzers that are used to extract identification information from the files analyzed.

Analyzer	File Types Scanned	Analysis Method
Archive	Zip archive format (*.zip, *.ear, *.war, *.jar, *.sar, *.apk, *.nupkg); Tape Archive Format (*.tar); Gzip format (*.gz, *.tgz); Bzip2 format (*.bz2, *.tbz2)	Extracts archive contents, then scans contents with all available analyzers.
Assembly	.NET Assemblies (*.exe, *.dll)	Uses GrokAssembly.exe , which requires .NET Framework or Mono runtime to be installed.
CMake	CMake project files (CMakeLists.txt) and scripts (*.cmake)	Regex scan for project initialization and version setting commands.
Jar	Java archive files (*.jar); Web application archive (*.war)	Examines archive manifest metadata, and Maven Project Object Model files (pom.xml).
NSP	Node Security Project is used to analyze Node.js' `package.json` files for known vulnerable packages.	
Nuspec	Nuget package specification file (*.nuspec)	Uses XPath to parse specification XML.
OpenSSL	OpenSSL Version Source Header File (opensslv.h)	Regex parse of the OPENSSL_VERSION_NUMBER macro definition.
Ruby bundler-audit	Ruby `Gemfile.lock` files	Executes bundle-audit and incorporates the results into the dependency-check report.

Using Dependency Check.

Use Cases of Dependency Check

- Prove the existence of the problem
- Baseline test when conducting POCs with commercial solutions
- OWASP dependency-check is used as the primary tool to identify known vulnerable components

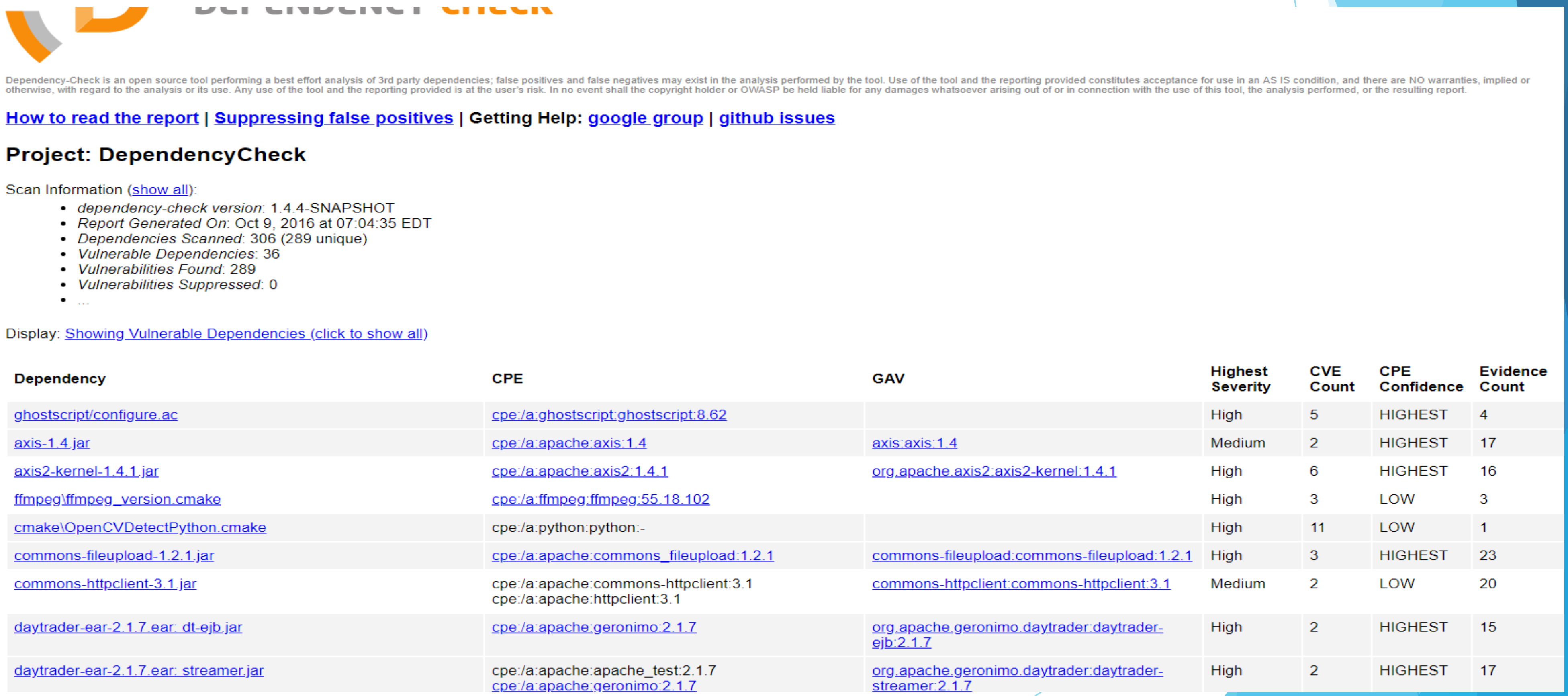
Using Dependency Check.

Enterprise Deployments.

- Use a centralized database to maintain the local copy of the NVD
  - Single instance of dependency-check used to update
  - Scanning instances do not need to update
- Use an internal Nexus instead of Maven Central
- Run dependency-check within their CI
- Continuous monitoring/reporting using **OWASP dependency-check sonar plugin**, OWASP dependency-track, or ThreadFix

How to read the reports.

- Not sure about what to write here. I am attaching a screenshot. May help you.



Dependency-Check is an open source tool performing a best effort analysis of 3rd party dependencies; false positives and false negatives may exist in the analysis performed by the tool. Use of the tool and the reporting provided constitutes acceptance for use in an AS IS condition, and there are NO warranties, implied or otherwise, with regard to the analysis or its use. Any use of the tool and the reporting provided is at the user's risk. In no event shall the copyright holder or OWASP be held liable for any damages whatsoever arising out of or in connection with the use of this tool, the analysis performed, or the resulting report.

How to read the report | Suppressing false positives | Getting Help: google group | github issues

Project: DependencyCheck

Scan Information (show all):

- *dependency-check version*: 1.4.4-SNAPSHOT
- *Report Generated On*: Oct 9, 2016 at 07:04:35 EDT
- *Dependencies Scanned*: 306 (289 unique)
- *Vulnerable Dependencies*: 36
- *Vulnerabilities Found*: 289
- *Vulnerabilities Suppressed*: 0
- ...

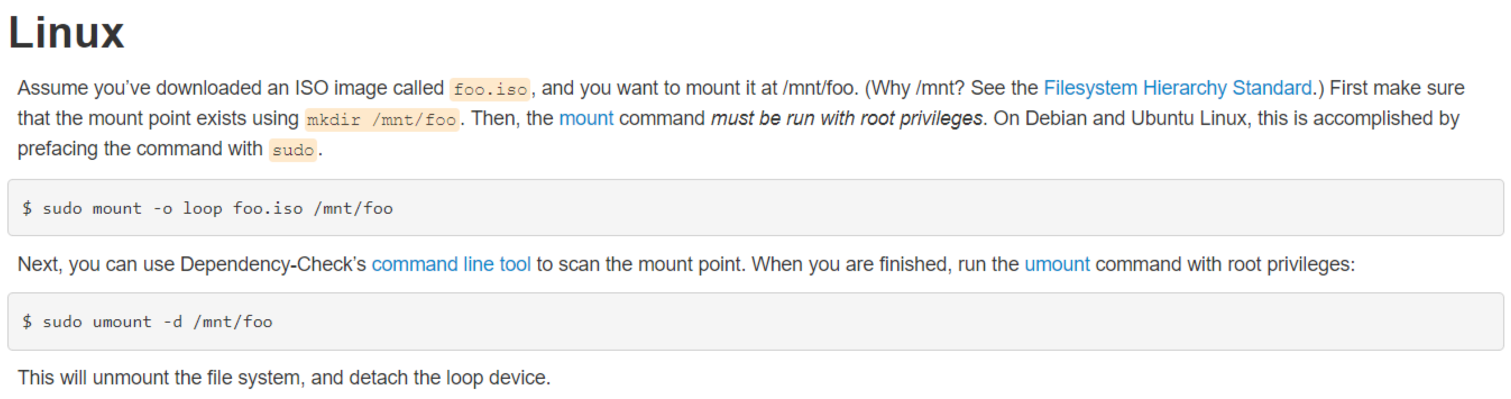
Display: Showing Vulnerable Dependencies (click to show all)

Dependency	CPE	GAV	Highest Severity	CVE Count	CPE Confidence	Evidence Count
ghostscript/configure.ac	cpe:/a:ghostscript:ghostscript:8.62		High	5	HIGHEST	4
axis-1.4.jar	cpe:/a:apache:axis:1.4	axis:axis:1.4	Medium	2	HIGHEST	17
axis2-kernel-1.4.1.jar	cpe:/a:apache:axis2:1.4.1	org.apache.axis2:axis2-kernel:1.4.1	High	6	HIGHEST	16
ffmpeg\ffmpeg_version.cmake	cpe:/a:ffmpeg:ffmpeg:55.18.102		High	3	LOW	3
cmake\OpenCVDetectPython.cmake	cpe:/a:python:python:-		High	11	LOW	1
commons-fileupload-1.2.1.jar	cpe:/a:apache:commons_fileupload:1.2.1	commons-fileupload:commons-fileupload:1.2.1	High	3	HIGHEST	23
commons-httpclient-3.1.jar	cpe:/a:apache:commons-httpclient:3.1 cpe:/a:apache:httpclient:3.1	commons-httpclient:commons-httpclient:3.1	Medium	2	LOW	20
daytrader-ear-2.1.7.ear: dt-ejb.jar	cpe:/a:apache:geronimo:2.1.7	org.apache.geronimo.daytrader:daytrader-ejb:2.1.7	High	2	HIGHEST	15
daytrader-ear-2.1.7.ear: streamer.jar	cpe:/a:apache:apache_test:2.1.7 cpe:/a:apache:geronimo:2.1.7	org.apache.geronimo.daytrader:daytrader-streamer:2.1.7	High	2	HIGHEST	17

DEMO

BACK UP

How to run a scan on ISO image.



Linux

Assume you've downloaded an ISO image called `foo.iso`, and you want to mount it at /mnt/foo. (Why /mnt? See the Filesystem Hierarchy Standard.) First make sure that the mount point exists using `mkdir /mnt/foo`. Then, the mount command *must be run with root privileges*. On Debian and Ubuntu Linux, this is accomplished by prefacing the command with `sudo`.

```
$ sudo mount -o loop foo.iso /mnt/foo
```

Next, you can use Dependency-Check's command line tool to scan the mount point. When you are finished, run the umount command with root privileges:

```
$ sudo umount -d /mnt/foo
```

This will unmount the file system, and detach the loop device.

Mac OS X

Using the GUI

Simply double-click on the image file in Mac OS X Finder.

Using a Terminal Window

Use the hdiutil command.

```
$ hdiutil attach foo.iso
```

The output will show the `/dev` entry assigned as well as the mount point, which is where you may now read the files in the image's file system.

To detach:

```
$ hdiutil detach foo.iso
```

Windows

Windows 8 and later versions support mounting ISO images as a virtual drive.

Using the GUI

1. In *File Explorer*, right-click on “foo.iso”.
2. Select “Mount”

File Explorer then redirects to showing the files on your virtual drive. You can then use the command line tool to scan the virtual drive. When finished, “Windows-E” will open File Explorer showing the various drives on your computer. To eject the virtual drive:

1. Right-click on the virtual drive.
2. Select “Eject”

Using PowerShell

To mount, use the Mount-DiskImage cmdlet:

```
$ Mount-DiskImage -ImagePath C:\Full\Path\to\foo.iso
```

To view all drives (and find your virtual drive), use the Get-PSDrive cmdlet:

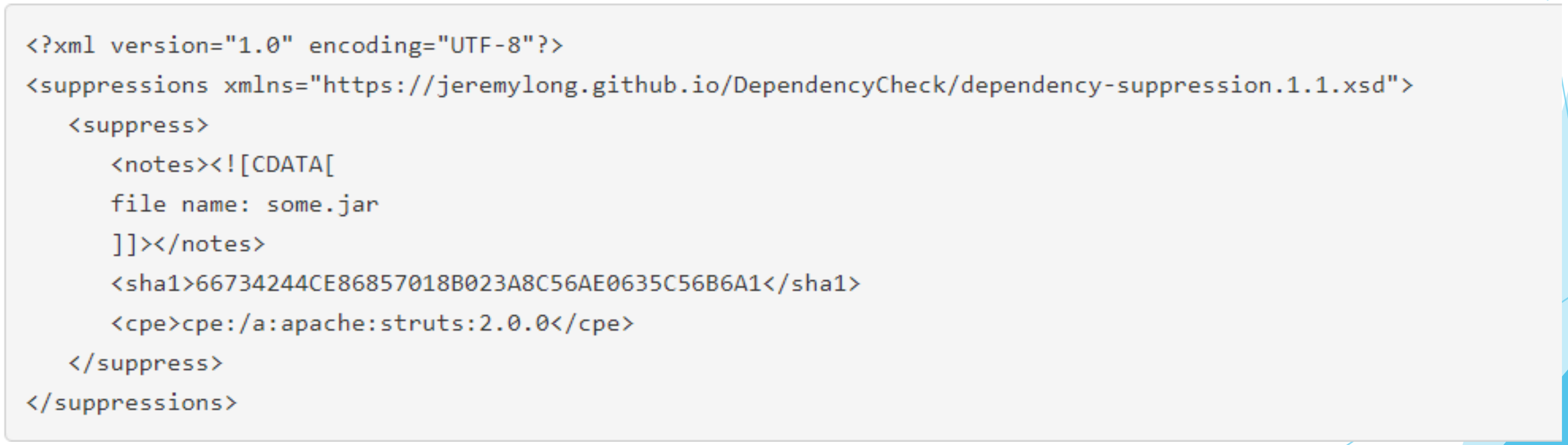
```
$ Get-PSDrive -PSProvider 'FileSystem'
```

To dismount, use the Dismount-DiskImage cmdlet:

```
$ Dismount-DiskImage -ImagePath C:\Full\Path\to\file.iso
```

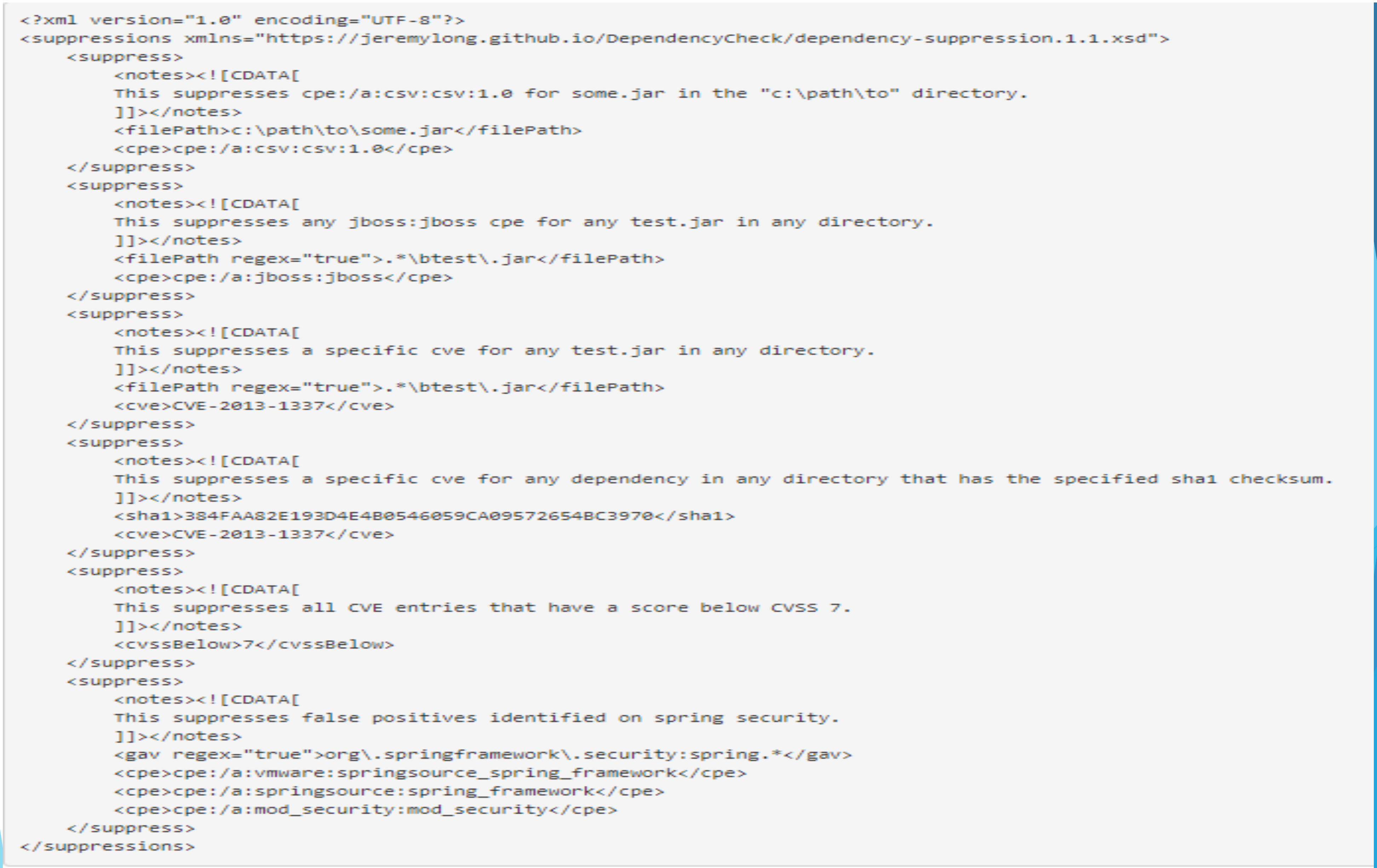
Suppression of false positive.

- Two ways of using the suppression.
  1) Using SHA hash and
  2) Using filepath
- Check for false positive manually if fixed or not.
- Click the suppress button in the xml report and save it in a txt doc.



```
<?xml version="1.0" encoding="UTF-8"?>
<suppressions xmlns="https://jeremylong.github.io/DependencyCheck/dependency-suppression.1.1.xsd">
   <suppress>
      <notes><![CDATA[
      file name: some.jar
      ]]></notes>
      <sha1>66734244CE86857018B023A8C56AE0635C56B6A1</sha1>
      <cpe>cpe:/a:apache:struts:2.0.0</cpe>
   </suppress>
</suppressions>
```

The above XML file will suppress the cpe:/a:apache:struts:2.0.0 from any file with the a matching SHA1 hash.



```xml
<?xml version="1.0" encoding="UTF-8"?>
<suppressions xmlns="https://jeremylong.github.io/DependencyCheck/dependency-suppression.1.1.xsd">
    <suppress>
        <notes><![CDATA[
        This suppresses cpe:/a:csv:csv:1.0 for some.jar in the "c:\path\to" directory.
        ]]></notes>
        <filePath>c:\path\to\some.jar</filePath>
        <cpe>cpe:/a:csv:csv:1.0</cpe>
    </suppress>
    <suppress>
        <notes><![CDATA[
        This suppresses any jboss:jboss cpe for any test.jar in any directory.
        ]]></notes>
        <filePath regex="true">.*\btest\.jar</filePath>
        <cpe>cpe:/a:jboss:jboss</cpe>
    </suppress>
    <suppress>
        <notes><![CDATA[
        This suppresses a specific cve for any test.jar in any directory.
        ]]></notes>
        <filePath regex="true">.*\btest\.jar</filePath>
        <cve>CVE-2013-1337</cve>
    </suppress>
    <suppress>
        <notes><![CDATA[
        This suppresses a specific cve for any dependency in any directory that has the specified sha1 checksum.
        ]]></notes>
        <sha1>384FAA82E193D4E4B0546059CA09572654BC3970</sha1>
        <cve>CVE-2013-1337</cve>
    </suppress>
    <suppress>
        <notes><![CDATA[
        This suppresses all CVE entries that have a score below CVSS 7.
        ]]></notes>
        <cvssBelow>7</cvssBelow>
    </suppress>
    <suppress>
        <notes><![CDATA[
        This suppresses false positives identified on spring security.
        ]]></notes>
        <gav regex="true">org\.springframework\.security:spring.*</gav>
        <cpe>cpe:/a:vmware:springsource_spring_framework</cpe>
        <cpe>cpe:/a:springsource:spring_framework</cpe>
        <cpe>cpe:/a:mod_security:mod_security</cpe>
    </suppress>
</suppressions>
```