Farhad Ahmed (fa961)
Redwanul Mutee (rm4243)
Application Security
03/24/19

Lab1b: Scanning the Code

Identification

For ImageResizer java app, we used two static program analysis scanners. The first scanner was *ck* which calculates object oriented metrics by processing source Java files. The second scanner used was the OWASP dependency check. The following report details the results of the two scanners after being run on the app.

ck

This scanner produces four csv files that contain metrics on classes, fields, variables, and also methods contained within the source code. The fields csv file was empty and contained no relevant information after the scan. The github link:https://github.com/mauricioaniche/ck , contains the meanings and explanations for the metrics presented.

The classes file contained the following:

file	class	type	cbo	wmc	dit	rfc	lcom	totalMethods	staticMethods	publicMethods	privateMethods	protectedMethods	defaultMethods	abstractMethods	finalMethods
/Users/Farhad_Ahmed/Desktop/APPSEC/LAB1/ApplicationSecurity/ImageResizer/ImageResizer.java	ImageResizer	class	1	4	1	10	1	2	1	2	0	0	0	0	0

synchronizedMethods	totalFields	staticFields	publicFields	privateFields	protectedFields	defaultFields	finalFields	synchronizedFields	nosi	loc	returnQty	loopQty	comparisonsQty	tryCatchQty	parenthesizedExpsQty
0	0	0	0	0	0	0	0	0	3	35	1	0	1	1	0

stringLiteralsQty	numbersQty	assignmentsQty	mathOperationsQty	variablesQty	maxNestedBlocks	anonymousClassesQty	subClassesQty	lambdasQty	uniqueWordsQty	
4	11	11	2	2	8	1	0	0	0	32

The methods file appeared as such:

file	class	method	line	cbo	wmc	rfc	loc	returns	variables	parameters	startLine	loopQty	comparison	tryCatchQty	parenthesiz	stringLiteral
/Users/Farhad_Ahmed/Desktop/APPSEC/LAB1/ApplicationSecurity/ImageResizer/ImageResizer.java	ImageResizer	sqTransformImg/3[java.lang.String	java.lang.String	int]	15	0	1	7	8	0	4	3	15	0	0	0
/Users/Farhad_Ahmed/Desktop/APPSEC/LAB1/ApplicationSecurity/ImageResizer/ImageResizer.java	ImageResizer	main/1[java.lang.String[]]	24	1	3	3	24	1	4	1	24	0	1	1	0	3

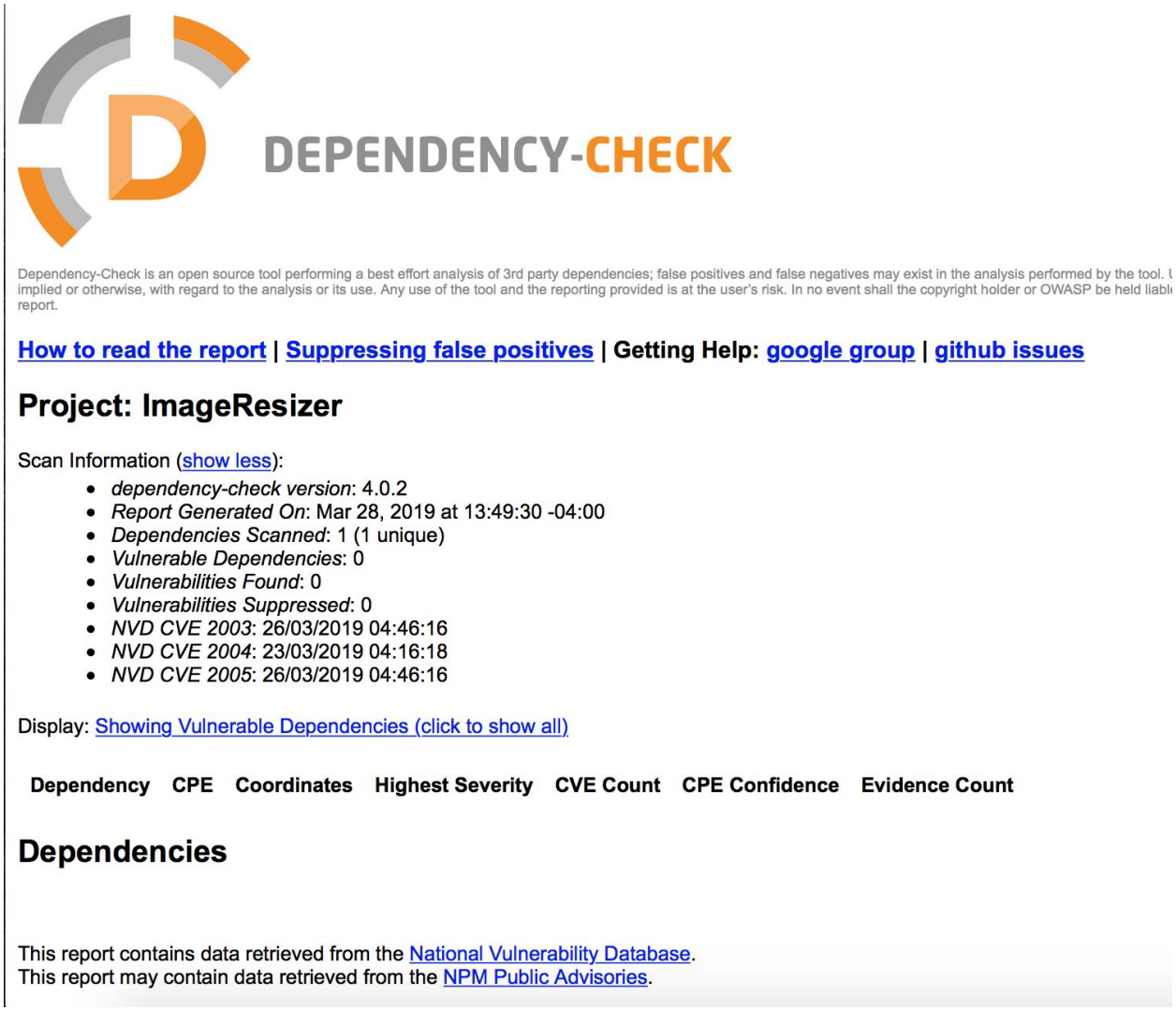
numbersQty	assignments	mathOperat	maxNested	anonymous	subClassesQ	lambdasQty	uniqueWor		
0	1	3	4	1	0	0	0	0	18
8	7	1	1	0	0	0	21		

And finally, the variables file listed all the used variables in the source code.

file	class	method	variable	usage		
/Users/Farhad_Ahmed/Desktop/APPSEC/LAB1/ApplicationSecurity/ImageResizer/ImageResizer.java	ImageResizer	sqTransformImg/3[java.lang.String	java.lang.String	int]	sqDim	4
/Users/Farhad_Ahmed/Desktop/APPSEC/LAB1/ApplicationSecurity/ImageResizer/ImageResizer.java	ImageResizer	sqTransformImg/3[java.lang.String	java.lang.String	int]	basePath	1
/Users/Farhad_Ahmed/Desktop/APPSEC/LAB1/ApplicationSecurity/ImageResizer/ImageResizer.java	ImageResizer	sqTransformImg/3[java.lang.String	java.lang.String	int]	squarePath	3
/Users/Farhad_Ahmed/Desktop/APPSEC/LAB1/ApplicationSecurity/ImageResizer/ImageResizer.java	ImageResizer	sqTransformImg/3[java.lang.String	java.lang.String	int]	sqrImg	2
/Users/Farhad_Ahmed/Desktop/APPSEC/LAB1/ApplicationSecurity/ImageResizer/ImageResizer.java	ImageResizer	sqTransformImg/3[java.lang.String	java.lang.String	int]	baseImgFile	1
/Users/Farhad_Ahmed/Desktop/APPSEC/LAB1/ApplicationSecurity/ImageResizer/ImageResizer.java	ImageResizer	sqTransformImg/3[java.lang.String	java.lang.String	int]	fillSqrImg	1
/Users/Farhad_Ahmed/Desktop/APPSEC/LAB1/ApplicationSecurity/ImageResizer/ImageResizer.java	ImageResizer	sqTransformImg/3[java.lang.String	java.lang.String	int]	baseImg	2
/Users/Farhad_Ahmed/Desktop/APPSEC/LAB1/ApplicationSecurity/ImageResizer/ImageResizer.java	ImageResizer	main/1[java.lang.String[]]	args	7		
/Users/Farhad_Ahmed/Desktop/APPSEC/LAB1/ApplicationSecurity/ImageResizer/ImageResizer.java	ImageResizer	main/1[java.lang.String[]]	ImageResizeInst	1		
/Users/Farhad_Ahmed/Desktop/APPSEC/LAB1/ApplicationSecurity/ImageResizer/ImageResizer.java	ImageResizer	main/1[java.lang.String[]]	sqSideLen	3		
/Users/Farhad_Ahmed/Desktop/APPSEC/LAB1/ApplicationSecurity/ImageResizer/ImageResizer.java	ImageResizer	main/1[java.lang.String[]]	e	0		
/Users/Farhad_Ahmed/Desktop/APPSEC/LAB1/ApplicationSecurity/ImageResizer/ImageResizer.java	ImageResizer	main/1[java.lang.String[]]	imgFilePath	3		
/Users/Farhad_Ahmed/Desktop/APPSEC/LAB1/ApplicationSecurity/ImageResizer/ImageResizer.java	ImageResizer	main/1[java.lang.String[]]	sqFilePath	3		

OWASP Dependency Checker

For the dependency checker, the source files of the app were compiled into a .jar file in order for the dependency checker to be able to scan it. After passing in argument dictating where the project path was as well as giving it a name, the scanner produced an html file with the following results:



DEPENDENCY-CHECK

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. U
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 liabl
report.

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

Project: ImageResizer

Scan Information (show less):

- *dependency-check version*: 4.0.2
- *Report Generated On*: Mar 28, 2019 at 13:49:30 -04:00
- *Dependencies Scanned*: 1 (1 unique)
- *Vulnerable Dependencies*: 0
- *Vulnerabilities Found*: 0
- *Vulnerabilities Suppressed*: 0
- *NVD CVE 2003*: 26/03/2019 04:46:16
- *NVD CVE 2004*: 23/03/2019 04:16:18
- *NVD CVE 2005*: 26/03/2019 04:46:16

Display: Showing Vulnerable Dependencies (click to show all)

Dependency	CPE	Coordinates	Highest Severity	CVE Count	CPE Confidence	Evidence Count

Dependencies

This report contains data retrieved from the National Vulnerability Database.
This report may contain data retrieved from the NPM Public Advisories.

Verification

In this portion of the report, the results of the two scanners will be discussed and their accuracies will be reviewed. Starting with the ck scanner, it provides a lot of useful information regarding the code metrics. Some of the most interesting metrics includes *WMC,* or rather McCabe's complexity. This metric reports the number of branch instructions in the class and gives a low-level insight of how the program is executing. Although this app was simple and did not delve into inheritance and derived classes, the metrics also report the depth inheritance tree. One important feature, is also reporting the number of static invocations since managing static classes/methods/variables can be the cause of some errors leading to the program not even compiling sometimes. Overall, running the ck scanner of the java source files is a good way to get an overall view of your project and see where/what is causing it to be hefty.

The OWASP dependency checker detects publicly disclosed vulnerabilities contained within a project's 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. The report generated on the ImageResizer.jar file detected 0 vulnerabilities in the program. This does make sense since the source code written did not delve into complex behavior. The simple resizing of the image was contained within the java basic scope and was never really exposed to anything that was vulnerable or could be exploited.