OWASP Dependency-Check For Your DevOps Environment

By: Michael Nee

What is OWASP Dependency-Check?

OWASP Dependency-Check is an open source software composition analysis(SCA) tool built and maintained by the OWASP foundation. It attempts to determine if a project contains vulnerable code within a project's dependencies. It does so in a variety of ways using hashes and package managers in order to determine if there is an existing CPE and what CVEs are associated with that CPE.

Value of SCA Tools In a Development Environment

SCA tools have multiple uses within organizations. The primary use of SCA tools is the ability to list an inventory of a product, allowing engineers to quickly list what code is brought in to their products without interrupting work. Another important use case is for finding any vulnerabilities within that inventory so they can quickly be addressed by the relevant teams. Lastly it helps developers to prioritize when and how vulnerabilities should be addressed.

What Are the Downsides to Using It?

OWASP Dependency-Check is a fantastic tool which produces very good results however since it is an open source product maintained by an online community it doesn't have an easily hostable option. It's two main competitors BlackDuck and WhiteSource both offer very good remote hosting solutions or on-site installations of their servers. This is the main drawback to Dependency-Check as it measures up to its competitors in most other areas. Without a way to easily display results across an organization Dependency-Check quickly runs into an issue where every developer runs their own scans and getting different parts. Without centralization of scan results an organization does not get as much value out of their SCA tools as they could be.

The Solution(ish)

Intro

So what's the solution you might ask? If you said "spend a lot of money on a very expensive solution which is solid", well you might be correct. Another answer however could be to implement your own centralized solution for a *free* tool. The rest of this blog will be focused

on how I went about doing so as well as some drawbacks and where there can be room for improvement.

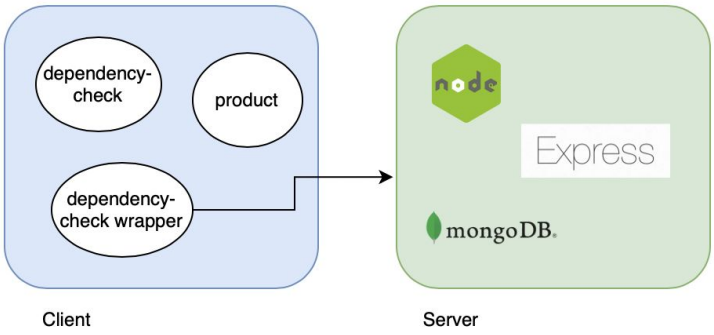
Addressing The Problem

The problem with Dependency-Check as stated above is it's lack of centralized reporting, so I'm going to step through how we might begin to address this issue. I initially had two thoughts which were to set up a notification solution which would email scan results to developers. This would allow for the scan to sit in the middle of a CI/CD deployment model quite well. As the code is built it is scanned and then developers are notified. However I thought this model would really suffer from a lack of versatility for later improvements, such as improved reports, record keeping, and efficiently reducing friction for developers. So from there I moved to a centralized model more similar to the other SCA tools, which is a centralized portal. This addresses problems with the previous model by allowing reports to be in one spot so that some post processing can happen on them as well as keeping a history and allowing the data to be viewed by anyone in the organization.

Design Time

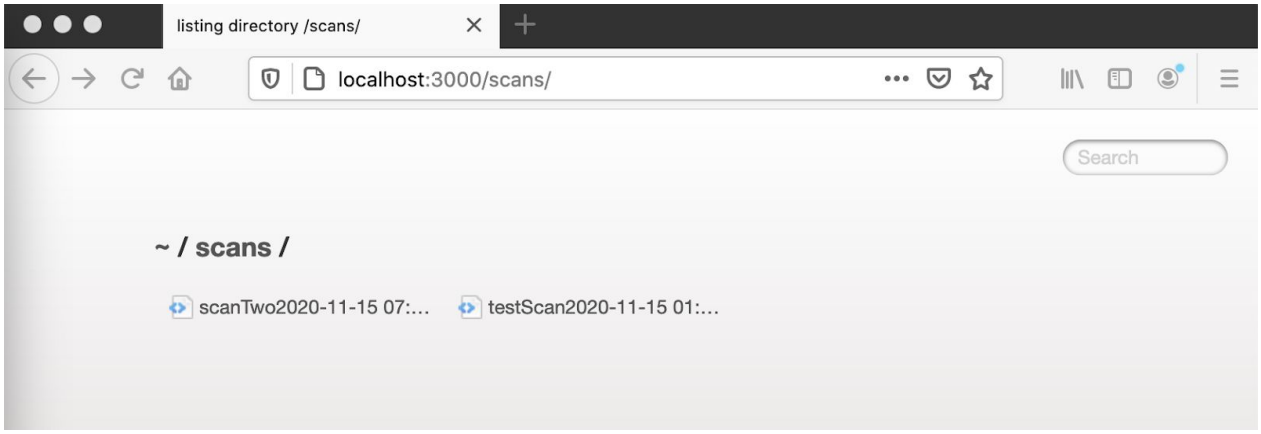
So for initial thoughts this program was going to be significantly more complicated than it ended up. Where I ended was a proof of concept for what an organization might do to implement Dependency-Checker, towards the end I will review what improvements I believe would be necessary for this solution to be practical.

The original design for this project was written down on a post-it note in sharpie, unfortunately due to a roommate related mishap it is no longer with us so I will recreate the original diagram to the right. It should be noted that my drawings on the post-it note were exactly as seen to the right. In later iterations mongodb was scrapped to accommodate for time, and replaced with just a standard JSON file. For practical uses a database should absolutely be used in the place of a JSON file. In the original design Dependency-Check was going to be integrated into the wrapper, but again for the sake of time, it was implied that the client machine would have it installed. In this case clients may refer to many places the best of which for this application would be to be added somewhere in a CI/CD pipeline so scans are run as code is built. However a developer's machine is also an acceptable and common use case.



The workflow for this diagram is that a developer or an automated tool would trigger a scan with node to run the Dependency-Check wrapper. The wrapper would then run on a specified folder where the product should be installed. From there the wrapper would run Dependency-Check and read in the created file. From there some meta data will be added and

that file will be sent over to the node server running express. Once on the server the file will be created in the public folder and served. The view of all scans will be shown below.



Each of these scans links to a Dependency-Check produced html page of a directory which is being scanned. For the purposes of demonstrating Dependency-Check I create a NPM package called [vulnerable-js] which contains multiple vulnerable libraries with varying severity. Both scans seen above were against a project which imported vulnerable-js. Again the complete report will be featured below.

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. 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: github issues

Project:

Scan Information (show less):

- *dependency-check version*: 6.0.1
- *Report Generated On*: Sat, 14 Nov 2020 20:53:24 -0500
- *Dependencies Scanned*: 2414 (2260 unique)
- *Vulnerable Dependencies*: 25
- *Vulnerabilities Found*: 66
- *Vulnerabilities Suppressed*: 0
- *NVD CVE Checked*: 2020-11-14T19:57:06
- *NVD CVE Modified*: 2020-11-14T18:01:18
- *VersionCheckOn*: 2020-11-14T19:57:06

Summary

Display: Showing Vulnerable Dependencies (click to show all)

Dependency	Vulnerability IDs	Package
bestzip:2.1.6	cpe:2.3:a:bestzip_project:bestzip:2.1.6:*:*:*:*:*:*:*	pkg:npm/bestzip@2.1.6
bestzip:2.1.6		pkg:npm/bestzip@2.1.6
deep-extend:0.6.0	cpe:2.3:a:deep_extend_project:deep_extend:0.6.0:*:*:*:*:*:*:* cpe:2.3:a:extend_project:extend:0.6.0:*:*:*:*:*:*:*	pkg:npm/deep-extend@0.6.0

Reflection

Room for Improvement

Now obviously this project is far from perfect, there are many tweaks and updates that should be done before it can be implemented in production anywhere. For one as of right now only the html files produced are being uploaded, however if the JSON result of all the scans were to be uploaded there is a lot of room for post processing. Using data from an uploaded JSON more accurate inventories could be created as well as there could be an attempt at creating an attribution report to list licenses. In addition to that there is no functionality as of now to have multiple products. If you were to have two products the only sorting you could do would be by what the scan is named, or have a separate instance of the server running. If you or anyone you know would be interested in making a contribution all the source code is available on my Github.

Final Takeaways

This project was very interesting to me because it definitely helped me to learn more about some of the technologies I used to complete it, but also much more about Dependency-Check. And while there are certainly some glaring issues this does seem to be a great improvement, and is certainly very useful for anyone who doesn't have the money to shell out for one of the big name brand competitors.

Sources

Some of the sources I used for this project are listed below.

White Source SCA Description
Dependency-Check Description
Dependency-Check Command Help
Help With Some Frontend