Assignment 2: OWASP Open Source Review Project

Software Security 2015

Group Number: 6

Student Name		Student Id	
		TU/e Eindhoven	RU Nijmegen
1	RAMAKRISHNAN LAKSHMINARYANAN	979826	4677080
2	PRASANNA SURYANARAYANAN	980938	4677099
3	MILTON SABIITI	847370	4320654
4	JASDHEER SINGH MAAN	979995	4674952

1. Organization

The approach used for this project was twofold:-

1. Generic review of the source code for trivial /obvious security vulnerabilities. This involved the individual scan through of the whole TestCMS code for trivial vulnerabilities .This was done simultaneously with pre –reads of OWASP guidelines and PHP self-study. The team met and discussed the findings and challenges.
2. ASVS categories of security requirements were equally subdivided between the group members to make a thorough review of the TestCMS web Application. The results were populated in the ASVS-checklist individually. This process took around 1 month.

The team met for a review of the findings and re-allocated the topics for cross –review. The findings were discussed and this report has been generated to that effect. – This process has taken us approximately three weeks.

Tools

The team utilized fortify together with RIPS (each member used the tools individually). However, we have relied more on the results of Fortify in our verdict because RIPS was giving too many false positives thus making our review tricky.

The application was run by each one of us and this was found handy in our review. About 50% of our verdicts are based on navigation through the deployed TestCMS application and comparing with the related source code.

2. Verdict

We have moved the motivation of all the verdicts to the “Motivation/Comments” column in the spreadsheet ASVS-checklist_group_6.xlsx.

Overall we would judge the TestCMS application as having ASVS Level 1 security.

3. Reflection

ASVS

The ASVS in general was clear and requirements were able to be verified. Almost all the areas of security were covered which forms a very good rule system also since the verifications are classified into 3 levels of security.
But some requirements were quite unclear. It kind of misguided in understanding the requirement that it tried to convey. Maybe an example along with the requirement would solve this concern. Overall ASVS is well constructed, brief and to the point guide intended for security experts.

Tools used (Fortify and RIPS)

Fortify was quite useful in identifying the vulnerabilities and security issues. It was also clear in the way it brought out the issues. In addition to pointing out issues, it also gives in corrections and suggestions to make corrections.
The tool categorizes the issues as Hot, warning and info. Fortify provided around 116 issues, but if we carefully check them many of them are like just information or false positives. But at the same time it did point at important issues too. So as a best practice one could start with the Hot and warning issues and leave the info issues as most of the info issues tend to be false positives.

We also used RIPS, it is a tool which is a kind of web application in itself which runs on the server. RIPS also provided results with some good warnings. Most of them were an overlap of what we found in the Fortify. But the amount of false positives in RIPS was more as compare to Fortify.

The approach we consider in Fortify tool is to educate the programmer and fix errors, unsafe and unstructured code. Even though RIPS did the same, but Fortify found many more warnings and attention points in the source code. Apart from that, the explanation in Fortify is more extensive and deep. We consider Fortify very useful.

Bottlenecks

In judging some of the requirements, we could not come to a conclusion in deciding the verdict. For example, in the issues regarding inclusion attacks, there were a number of security information leak issues reported by Fortify. But, we got stuck up in finding out the way of exploiting the vulnerability even after going for many different trials.

Verification requirements for categories like Authentication and Access control were quite easy to judge since they could be done by directly checking with the application without using any tools. Some categories like Files and Resources had issues that required the use of tools like Fortify. Some requirements in Communication Security were not able to be judged since they demanded the information about the Certificate Authority and communication protocols used.

Future Improvements

If we were asked to do a task like this again then we would rather assign/divide the work in pairs. Because there are many such requirements which are on the borderline of the verdict, which needs a quite lot of discussion with the fellow mates. This time we first judged the verdicts individually and then figured out in the end altogether, by this there was change in the judgement for many requirements which was a kind of rework for all of us.

TestCMS Web Application Overview

In TestCMS, one main security issue was Access control failure. This could be sorted out through delegation of rights depending on the roles and implementation of principle of least privilege.
Similarly, in Authentication there existed only a primary authentication mechanism.
So, apart from this, secondary authentication, two-factor authentication, offline recovery or any other authentication mechanism could be implemented additionally.

Given a task of developing an application, we might consider applying the basic security factors like a strong authentication mechanism, access control with least privilege, inclusive of a proven security mechanism.
Another issue was no validations on the client side, all the validations were carried out on server side. Validations should be used on both side.
For data security, authentication mechanisms like resource governor and request throttle might be used along with an account lockout mechanism. To prevent inclusion attacks, validation of data from other sources could be done. Also filtering out the type of input data using whitelist could be handy.