Software Assurance Tips

A product of the Software Assurance Tips Team[1]

Jon Hood

Monday 6th September, 2021

1 When Code Analysis Fails

Updated Monday 6th September, 2021

This article contains content that originally appeared in the August 31, 2017 Software Assurance Tips

One of the first open source projects I ever made was a webpage visit counter. But this was no ordinary counter: a user could upload a picture, and the code would overlay the webpage "hits" on that picture. The project was hosted on an old Compaq 133MHz 4U rackmount server that a friend and I "purchased" (i.e., rescued from the dumpster), installed Linux+Apache+PHP on, and connected to his state-of-the-art Cable Alabama broadband connection. The code used ImageMagick to dynamically rebuild the image every time a request came in to show the picture and the number of hits.

Everything worked great…until we started getting several hundred hits per second! The server could not keep up with the request load, and during peak usage, it was practically inaccessible. I learned an important lesson in resource management!

Now that development projects are more security-aware, our SwA team would identify any unsanitized image coming from the user as a potential for code injection, resource management errors, or even a steganography attack. Nevertheless, software sometimes has the requirement of accepting images from users. Using an up-to-date image processing library and limiting the types of input that can be received are critical for managing a system's resources.

Recently, we've evaluated some image and map manipulation software that draws pictures and graphs on the supplied images. The systems were marked for not using an up-to-date version of their graphics manipulation software. Suppose that developers decide to implement GraphicsMagick (`http://www.graphicsmagick.org/`) as their graphics manipulation library. The software advertises a maintainable codebase (even mentioning the famous David A. Wheeler on their main page) and that their recent Coverity scans reveal 0 defects per 1000 lines of code. With credentials like that, it can't be insecure! Nevertheless, CVE-2017-13777 is out detailing a resource consumption attack with XBM (bitmap) images.[2] A pernicious bitmap would have brought my old server to a halt!

Even if the security scans don't find an issue, we continue to mark references to unmaintained versions of software as a potential security concern. It indicates that a development team is not staying up-to-date with the latest dependency developments and may miss important CVEs and security-relevant software updates. CVE-2017-13777 is a prime example of what happens when precautionary software assurance techniques don't identify an underlying issue in the code. If someone were to take the Juliet SARD test suite and run it against an analysis tool, it is very likely that the tool would not identify the flaw in more than 50% of the examples. Therefore, a competent software assurance plan does not merely take a proactive, preventative approach to software security; it must also have a reactionary, continuing plan for software maintenance.

References

[1] Jon Hood, ed. SwATips. https://www.SwATips.com/.

[2] MITRE. CVE-2017-13777. Aug. 2017. URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-13777.