Software Assurance Tips

A product of the Software Assurance Tips Team[5]

Jon Hood

Monday 18th September, 2023

1 A History of Verification, Validation, and Code Scanning

Updated Tuesday 27th January, 2026

One of the DoD and NASA trends we have seen lately is the confusion of functional VV&A with cybersecurity tasks. The blurring of these two distinct efforts from organizations implementing an IV&V program and a cybersecurity program can cause collisions with the definitions of what is being accomplished.

Static code analysis is one of those definitions. The meaning of static code analysis is very different from a Functional, VV&A IV&V program when compared to the static code analysis conducted by a cybersecurity program. The etymology of static analysis comes from completely diverging goals.

1.1 Verification and Validation

"Haste makes waste." In the 1960s and 1970s, a static code walkthrough was recommended before ever compiling the module. "The verification sessions should occur before the first compilation of the module." Since "few if any compilers are capable of detecting every syntax error," syntax errors were considered logic errors in the software.[8, pp. 144, 149, 292] The static analysis and formal walkthrough of code from a V&V perspective has the goal of verifying the integrity of the logic that goes into the design of the software module.

This ties into the current DoD VV&A process for Verification, Validation, and Accreditation in DoD Instruction 5000.61. These are defined as:[2]

- *Verification*—The process of determining that a model or simulation implementation and its associated data accurately represent the developer's conceptual description and specifications.
- *Validation*—The process of determining the degree to which a model or simulation and its associated data are an accurate representation of the real world from the perspective of the intended uses of the model.
- *Accreditation*—The official certification that a model or simulation and its associated data are acceptable for use for a specific purpose.

One way to accomplish these goals is by implementing IEEE 1012, the IEEE Standard for System, Software, and Hardware Verification and Validation. This lays out the purpose of V&V: "to help the organization build quality into the system during the life cycle. V&V processes provide an objective assessment of products and processes throughout the life cycle. This assessment demonstrates whether requirements are correct, complete, accurate, consistent, and testable." This standard recommends a process for inspecting source code to "verify that the source code implementation is traceable to the design" and a static walkthrough of the code for the same types of logic issues Myers mentions in his book.[6, pp. 10, 213, 216] Cybersecurity static code analysis scans are circumstantial evidence used to demonstrate the testability of the codebase.[6, p. 206]

If you want to certify that a simulation or software model correctly reflects the real-world scenario and requirements that define the software module, VV&A is the way to go. Static code walkthroughs are often employed as a tool to accomplish this task.

Static analysis tools for proving the logic of code include flow provability tools such as GNATprove, property checkers[4] such as RapidCheck and QuickCheck[7], SlithIR SSA, Prusti, and many more static analysis tools.

1.2 Cybersecurity

Static analysis is defined in the cybersecurity realm by DoDI 8500.01 as part of the Risk Management Framework which implements NIST 800-53.[3] Static analysis and the tools which conduct it are defined in controls RA-5, SA-11, and SA-11(1). "Static code analysis provides a technology and methodology for security reviews and includes checking for weaknesses in the code as well as for the incorporation of libraries or other included code with known vulnerabilities or that are out-of-date and not supported. Static code analysis can be used to identify vulnerabilities and enforce secure coding practices."[10, p. 277]

Cybersecurity static analysis usually maps issues against Common Weakness Enumerations (CWEs), Common Vulnerabilities and Exposures (CVEs), and software assurance defects. A good framework for defining these secure code security analysis tools is NIST 500-268 which defines functional requirements of cybersecurity static analysis tools. Buffer overflows (such as stack and heap overflows, CWEs 121 and 122), uninitialized variables (CWE-457), TOCTOU (CWE-367), and injections (such as command, SQL, and LDAP injections, CWEs 78, 89, and 90) are examples of cybersecurity concerns.[9]

Tools that conduct these checks include Fortify, Checkmarx, and many more. The performance of these scans is implemented as part of the Application Security and Development STIG[1] which provides even more guidance on how to establish coding standards (SV-222653r879887_rule), independent testing (SV-222627r879887_rule), and even specifies some minimum requirements for static code analysis such as race conditions (SV-222567r879887_rule), storing sensitive information in hidden fields (SV-222601r879812_rule), cross-site scripting (XSS) (SV-222602r879652_rule), cross-site request forgeries (CSRF) (SV-222603r879652_rule), command injections (SV-222604r879652_rule), canonical representation vulnerabilities (SV-222605r879652_rule), input validation/handling (SV-222606r879652_rule/SV-222609r879818_rule), SQL injections (SV-222607r879652_rule), XML-based attacks (SV-222608r879652_rule), and overflows (including buffer overflow, heap overflows, stack overflows, and wrap-around issues) (SV-222612r879821_rule).

1.3 Conclusion

We have seen an increase in VV&A organizations claiming that they're conducting the static analysis defined by VV&A processes yet instead conduct cybersecurity static analyses. A good program will incorporate the DoDI 5000.61 requirements, and static analysis tools that speak the language of logic, proofs, and contracts is one small portion of the capabilities such organizations should implement. If your program needs accreditation or authorization from a cybersecurity perspective (such as an Assess and Authorize under RMF), a cybersecurity program will employ static analysis tools that speak the language of weaknesses, vulnerabilities, and coding standards.

Perform a litmus test on your own programs today. Is your VV&A program using cybersecurity tools such as Cppcheck, Code Dx, Checkmarx, Fortify, or Coverity? Suppose your program has a requirement to draw a circle. These are the tools that allow you to say, "Yeah, verily, this is the most secure square that has ever been drawn on a screen." But these tools do not allow you to say, "You drew a square when the requirements contract for the code is that you draw a circle." That involves provability and qualification testing that your VV&A program should be concentrating on.

References

[1] Application Security and Development STIG V5R3. Tech. rep. July 2023. URL: https://dl.dod.cyber.mil/wp-content/uploads/stigs/zip/U_ASD_V5R3_STIG.zip.

[2] Department of Defense. Department of Defense Instruction 5000.61. Tech. rep. Incorporating Change 1, October 15, 2018. Washington, D.C.: Department of Defense, 2018. URL: https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/500061p.pdf.

[3] Department of Defense. Department of Defense Instruction 8500.01. Tech. rep. Incorporating Change 1, October 7, 2019. Washington, D.C.: Department of Defense, 2019. URL: https://www.esd.whs.mil/portals/54/documents/dd/issuances/dodi/850001_2014.pdf.

[4] Harry D Foster. "Trends in functional verification: A 2014 industry study". In: Proceedings of the 52nd Annual
2015, pp. 1–6.

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

[6] IEEE Standards Association. "IEEE Standard for System, Software, and Hardware Verification and Validation". In: IEEE Std 1012-2016 (Revision of IEEE Std 1012-2012/ Incorporates IEEE Std 1012-2016/Cor
(2017), pp. 1–260. DOI: 10.1109/IEEESTD.2017.8055462.

[7] Andrew Jones and Jeremy Sonander. "An Introduction To Property Checkers For Functional Verification". In: (2003). URL: https://averant.com:8443/assets/pdf/Intro_PropVer.pdf.

[8] Glenford J Myers. Software Reliability. John Wiley & Sons, Inc., 1976. ISBN: 978-0-471-62765-4.

[9] National Institute of Standards and Technology. Security and Privacy Controls for Information Systems and O
Tech. rep. Special Publication (SP) 500-268 v1.1. Washington, D.C.: U.S. Department of Commerce, 2011. DOI: 10.6028/NIST.SP.500-268v1.1. URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication500-268v1.1.pdf.

[10] National Institute of Standards and Technology. Security and Privacy Controls for Information Systems and O
Tech. rep. Special Publication (SP) 800-53 Revision 5. Washington, D.C.: U.S. Department of Commerce, 2020. DOI: 10.6028/NIST.SP.800-53r5. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r5.pdf.