SYNOPSYS®

Initial minimum requirements for testing software source code

When created, all software has an intended purpose and deployment model dictating how it should be operated. At the point of creation, that software was tested to ensure it met those intended objectives. With each release iteration, testing continues, but over time the intended purpose may shift, leading to usage different from the original intent. In addition to iterative evolution, software can be assembled and combined creating novel solutions, each in turn differing from the original intent of any constituent component. Both the iterative and derivative models testing paradigms need to account for both the original intent in addition to changes in behavior resulting from the new usage pattern.

Unless response to Executive Order 14028 addresses modern development paradigms and provides transparency into the intent of code and the associated testing performed at each stage, the order may fail to meet its longer-term objectives. Meeting that desired outcome requires specific testing scenarios be factored into a baseline for software assurance, including:

- An understanding that no single testing technique will identify all potential defects or weaknesses within a given software codebase. Rather than focus on specific testing frameworks, priority should be given to how any findings would be remediated and communicated to those teams downstream from the tests being executed;
- A recognition that a “trust but verify” model allows for one testing technique to confirm both the results from a different technique in addition to confirming whether any mitigations or compensating controls are effective in reducing the impact from unresolved defects. For example, IAST or penetration testing can be used to confirm mitigations required to address unpatched weaknesses identified by a SAST tool;
- An understanding that modern DevOps, DevSecOps, and “cloud-native” development paradigms prioritize feature velocity over continuous security testing. One implication of this reality is that multiple software releases can occur in short order with the tests performed varying between releases.

Focus for testing efforts at scale should include the context of any code changes made between releases. For software following an agile release-on-demand or continuous deployment model, that context should include an understanding of how interim code changes impact existing threat models, monitoring solutions and deployment protections while providing that context to automated testing solutions to confirm compliance with security targets. In the event of contextual changes involving monitoring solutions and deployment protections, deployment of any changes should be deferred until monitoring and deployment adjustments are made.

While this discussion is focused on new feature development and deployment, consideration needs to be made to rollback mechanisms such that threat models, monitoring solutions, and deployment realities always match the specific deployed version of code.

Additionally, it must be recognized that many suppliers aren’t comfortable sharing their source code with buyers for any number of reasons. Where source code isn’t available for review, evidence of source code testing should be provided with each release, patch or update to an application, where that evidence could be in the form of test results, third-party test attestations, or from a third-party repository of associated abstract syntax trees. Any source code testing results should include attestations about how exploitable weaknesses and vulnerabilities were addressed or mitigated such that those remediation and mitigation efforts can be validated prior to deployment.