Secure software development lifecycle standards

SDLC challenges and solutions have long existed. This is an area where guidelines exist and should be followed but are simply not. The first most important requirement should be sound change control, and this should be enforced by government, not by providers. It is simply too important to be left to 3rd parties, and has costs that far exceed the ability of providers to provide. This is a national laboratory level issue in R&D and an NSA issue in execution. Having said that, here is an outline of the standard approaches that work when applied.[1]

Sound change control:

- Generically, the system includes separate areas with code only passing in one direction:
  - R&D → Change control → Testing → Production.
  - Failure at any step is indicated outside of the system along with the basis, and any change/component failing along the way is rejected and not allowed to progress except starting again at R&D.
- All changes (and components) should be specified with a justification for the change that is accepted as necessary for the identified applications of the mechanism.
  - This implies that all mechanisms should be identified as to their purpose and thus Trojan horses or 'spyware' and similar intentionally embedded capabilities of the mechanisms must be revealed openly as a basis for the inclusion of relevant code.
- All changes (and components) should only provide the mechanisms necessary to perform the identified purpose.
  - If they do other things, the changes (and components) should be rejected and the change (component) not allowed to go into testing or production.
- Change control should include code review by automation and humans examining every step in the context of the situation within the program and paths to and from the change, as well as in the context of the overall system the component is intended to operate within.
  - Individuals performing such review should have to identify and document what they did and certify that they did it. They should be subject to random verification and audit, and any failure to do what they said they did should be treated with an optional one-time warning, and immediate termination from the change control process with no return. This is different from imperfection in the carrying out of their tasks, which cannot be expected to be perfect.
- Only after a change is approved, it should be passed into testing. The test environment should compile / interpret / execute the code independently of anything in the R&D environment other than the actual change and prior accepted version.
  - The test environment should have identified historical testing criteria, including regression testing for all previous known fault types and failures resulting from them, and random and other testing methodologies with defined coverage requirements over specified fault models.
- Only after successful testing with regression for all prior known causes of failures and additional testing with known coverage on test data, then samples from live data, can the change be propagated into live systems.
  - Deployment should provide the means to roll back to prior versions and content, or in cases of forward-only deployments, be able to operate in a reduced mode with prior versions in emergencies.

---

1 Fred Cohen – CEO – fc@all.net – please consider me as a speaker at the workshop

Rolling deployments:

- With rare exception, deployments should be done in a rolling sequence, starting with the most urgent and least important circumstances, then going to urgent important, then less urgent more important, then to less urgent less important. The reason for this ordering is that the initial deployment of changes often reveals problems not detected in change control due to complexities and conditions of the operating environment that may differ from the test environment. Urgent situations should be undertaken first, but it's better to have negative effects detected on less important situations first. After that, the sequence is naturally highest overall priority to lowest.
- The size of deployments in the step-wise roll-out approach should be limited by risk aggregation limitations specified in duties to protect and risk management. No single step of the roll-out should exceed the risk aggregation threshold for the entity or its hierarchy (e.g., an agency or the Federal government as a whole).

Inventory and provenance and attribution channels:

- Inventory is fundamental to effective control, and security inventory should be in place for all systems, content, people, and business relationships. The inventory mechanisms identified in the Standards of Practice along with relevant lifecycle controls, and all relevant elements of the COP table should be used in this inventory system, which should be a system of records and operated and maintained within the standards for archival and records management.
- The entire supply chain of software and systems should be supported by provenance information and attestations as to source and path. Existing methodologies including SBOM and DBOM may be immediately adopted as a path forward, with augmented mechanisms applied over time.

Integrity controls:

- Integrity controls, particularly in the form of detecting unauthorized changes, should be applied to all systems, content, and the entire software stack, from firmware through user coded components, and at all levels in between. At a minimum, this should include cryptographic checksums or hash functions and real-time prior to use detection of changes with proper automatic responses based on requirements for operational modalities, including fail safe modes and taking into account the different failures associated with ceasing operations and continuing them under the loss of integrity based on detection. The detection mechanisms themselves should be protected so as to eliminate all feasible methods of corrupting them, which would produce outages caused by the integrity mechanisms themselves.

Additional areas in the EO:

- Authentication and authorization should be controlled based on risk aggregation limits of individual responsibility and escalation of authentication (quantity and frequency of uses and number and types of factors) based on the consequences associated with failures. Matching surety to risk is the basic concept here.
- Providers to the Federal system should have uniform requirements for always providing and updating artifacts demonstrative of conformance of all requirements, including those of their dependent supply chain. This should be integrated into DBOM and SBOM or similar attestation channels and audited with frequency and statistical requirements commensurate with the consequences of failures. A failure to fulfill this requirement should lead to a limiter form of disbarment or other similar level of punishment. These artifacts should be generated all the time and not limited to periodic or upon request responses.

These are all standard approaches that should have been codified long ago. And they are now under mandate for codification. Hopefully this will help.