This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

This report was prepared for the SEI Administrative Agent AFLCMC/AZS 5 Eglin Street Hanscom AFB, MA 01731-2100

DM18-1200

Recommended Implementation Priorities for Dev(Sec)Ops

Process – DevOps is about people, not about tooling. It is important that you define your software development process, requirements, and goals.
Compatibility – When designing a DevOps pipeline and selecting development tools, it is imperative that tools will smoothly operate in all environments. If this is not possible, it might be necessary to use more than one tool. However, one must be careful; each development group or contractor could pick a different tool. This will be management disaster.
Consistency – Consistent software requires consistent processes around the SDLC. Each team and team member need to understand the chosen software process. Documentation and process enforcement is the key to success.
Security – An important part of the software deployment process is security. It is also an integral part of RMF and ATO. Depending on the hardware, software, and environment, there are numerous ways to implement software assurances.
Communications – DevOps is about people communicating. Open communications are required to have a successful DevOps environment. The DevOps way is to ensure that there is transparency across all stakeholders of a project. In addition, developers need to visualize how and when their builds will be tested and deployed, system operators need to be aware of upcoming installations, and end user will need to be trained for new features.
Deployment – Software deployment needs to be an automated and repeatable process. Automation reduces the chance of error, and it will allow for an automated version rollback if needed.
Auditing – Security and accountability require auditing to understand were bugs and security flaws were introduced into the software. This is to improve the software process, calculate insider threats, and help produce teachable lessons.
Scalability – Software builds should not cause a long delay in testing and deployment. If developers have to wait until free slots are available in the pipeline, productivity is lost. This can jeopardize the project delivery window.

Example priorities for implementation

NOTE: Any products mentioned in this document are examples. They are not recommendations.

Process

Define gates for the pipeline

- Block build for peer code review
- Block build for security deficiencies
- Block for any manual or physical testing
- Block deployment until ATO is granted

Define behavior and unit testing requirements

- Minimum test coverage of a module's source code
- Manual testing requirements for physical devices
- Behavioral test frameworks, ie: Cucumber or SpecFlow,
- Matching behavioral test to the requirement

Define acceptable risk for security baseline

- Patching or update expectations for zero-day security flaws or critical bugs
- Acceptable risk for security flaw that cannot be leveraged in a specific environment

Define ownership of feature and system requirements

- Define the Agile process and sprint expectations
- Define the product owner that is responsible for the service or application

Define ATO-ready

- Define the required reporting needed for ATO
- Different organizations have various RMF requirements for ATO
- Approving parties need to understand the risks and benefits of the deployment

Issue and feature tracking

- JIRA
- Rally
- JAMA

Compatibility

Tooling interoperability between systems and contractors

- Source control should work seamlessly in every development environment

Tooling must be compatible all facets of the system

- Deployment to a Windows system should suppose UNC paths
- Deployment to a Linux/UNIX system should support SFTP and NFS over SSH/TLS
- Container deployment should support a container registry for deployment to orchestrations like Docker, Docker Swarm, or Kubernetes

Users must be able to use the tools; if they are too complex or unusable, all efficiencies will be lost

- Native TFS on Linux requires a Git plugin or a custom application. This is complicated and should be avoided

Consistent software builds

Software development methodology

- Define a sprint
- Define a release
- Define a feature

Common source control

- Git
- SVN
- Mercurial

Common artifact repository

- Nexus
- VMware Harbor

Development, build, and test environment parity

- Use Infrastructure as Code tools to build each environment
- Developers will use a similar environment as production; if the application runs in a container, it should be developed in a container

Peer review

- Formal process that can be reviewed after completion, similar to a pull request
- Another developer that is familiar with the project should review source code submissions

Build tooling

- gcc
- clang
- make
- maven
- msbuild

Automated testing frameworks

- Selenium web driver
- pytest
- JUnit
- XUnit
- NUnit
- MSTest

Logging of all processes

- Build outputs are stored
- Test outputs are stored
- Security and penetration testing results are stored
- Logging is required to automate the generation of an RMF report

Security

Static code analysis

- Brakeman
- SonarQube
- RIPS
- FindBugs

Dynamic code and runtime analysis

- Valgrind
- Purify
- Load testing of software

Container vulnerability analysis

- Clair
- Sysdig Falco
- Dagda
- Twistlock
- JFrog XRay

Communication between tools and humans

RMF report generation

- The final step in the build process should generate the RMF report needed for ATO

- Report on automatic testing
- Report on security scanning
- Report on penetration tests
- Report artifact should provide all requirements needed for ATO

System status and monitoring overview of pipeline

- All stakeholders should have visibility of the DevOps CD/CI process and pipeline

Interoperability between tools

- Source code commits that fail any test should reject back to the developer
- Issues and feature tracking tickets should be updated when a feature or fix moves through the pipeline

On-demand reporting

- Stakeholders and users should be able to view logs and reports from the build system

Documentation

- The project, along with its build and test processes, should be thoroughly documented
- Documentation should be easily accessible and searched
- Wiki-style products and markdown make it easy for developers to update the knowledge base
- The build process can generate documentation from the source code comments

Deployment

Deploy new builds to different environments

- Builds should be deployable to all environments
- Tool should be configured to block deployments to production until the build is approved

Automated testing and deployment of testing environment and software

- Testing tools should be able to automatically build an environment, test the software, and tear down the environment

Feature and Code audit trail

- Build results integrated into issue tracker
- Auditing of builds, source control, and testing

Scalability

- Pipeline should be able to handle parallel builds without blocking development
- Easily replicated for usage by different teams or contractors
- Accessible by all developers