Life Insurance Association Singapore
LIFE IS WORTH PROTECTING. INVEST IN IT.

GENERAL INSURANCE ASSOCIATION

DevOps Security Framework (Insurance)

By *Insurance Standing Committee for Cyber Security (ISCCS)*

13 April 2020

Change History

Version	Change Description	by	Date
1.0	Initial Official release	ISCCS	13 Apr 2020

Background

The definition of DevOps describes the practice of how the development, testing, security and operation team would communicate and collaborate to transform the source code to a run-time that would be tested before being deployed to production. The term DevOps was coined along with the evolution of Agile development methodology that is made easier and possible with the accessibility of automation tools. Without a doubt, automation is the key cornerstone to ensure a successful DevOps / Agile implementation. A project would typically be broken down into many phases/ functionalities. Taking the advantage of this automated process, the time take from development, to testing, to deployment of a phase or functionalities would be greatly reduced.

However, in a highly regulated industry, frequent deployment of changes is viewed as a risk to security and governance processes or controls. Despite this concern, a DevOps concept where continuous integration and continuous delivery (CI/CD) can still be implemented securely. The difference would be to extend the implementation to manage more complex governance and compliance requirements.

Minimally, the implementation should address the following:-

- Ensure maker is not the checker.
- Ensure the right source code base that represent the current production runtime is used for development.
- Ensure the integrity of the build pipeline when the production runtime is updated (i.e. applied to both serialised and parallel developments).
- Ensure security tests been performed and meet the threshold before migrating to production.

Objective

The objective of this guideline is to establish a guideline or framework identifying the minimum and appropriate security controls required to ensure the DevOps environment meets the regulatory requirements.

This document will be focusing on the security requirement and will not examine the environment management and build management within the DevOps framework.

Boundary

For this framework, it will focus on the security controls to be in place to support the DevOps practice. The fundamental level-1 DevOps (e.g. Continuous monitoring, Agile development methodology,

application life-cycle management, agile operation, and etc). Below diagram illustrate the fundamental DevOps structure with the 6 focus areas.

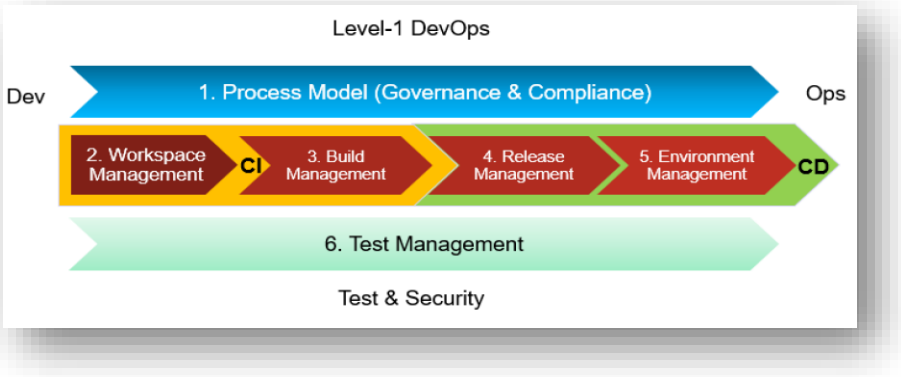

Figure 1 Fundamental DevOps Structure

Process Model

The process model defines a digitised workflow that will orchestrate the level-1 DevOps practice at the activity level (end-to-end). Below diagram depicts an example of the workflow that support the requirements for segregation of duties.

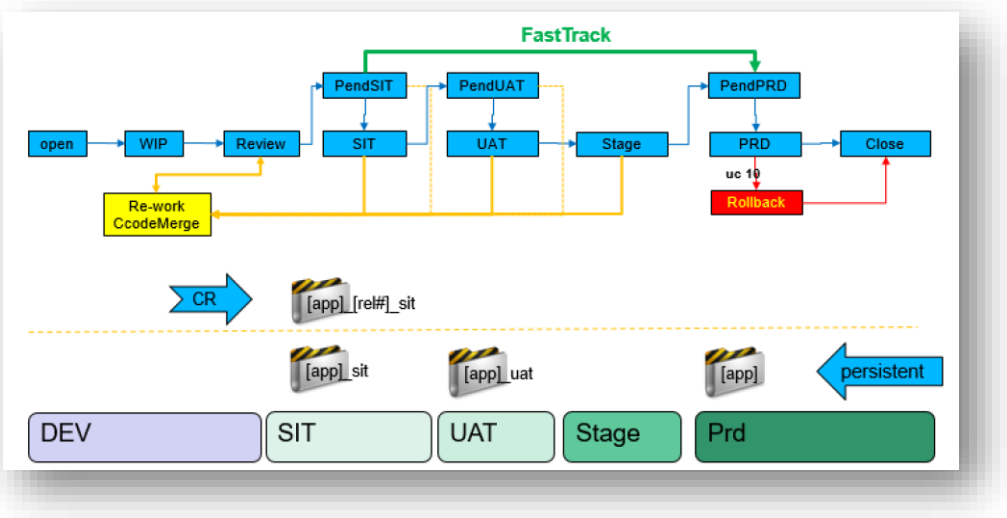

Figure 2 DevOps Process Model

The workflow shall apply to the dev, test, security and ops team to promote the application release from dev to test, and to the production environment. The workflow shall also support emergency release, production rollback situation and the need to support re-work. (test failure, source package e change, code merge)

This process model will be the base model for determine the appropriate security controls to be implemented.

Security Controls Consideration

1. Roles and Responsibilities

Each organisation may implement the DevOps process differently, this framework will not attempt to list out all the roles and their responsibilities. Instead, each organisation is responsible to establish a RACI matrix that identified the roles, associate responsibilities and ownership in the DevOps process.

Below are some tasks or functions that would form part of the DevOps process. Organisation may wish to use this list as a starting point to identify detailed functions specific to its DevOps process. A RACI matrix can then be established identifying the different roles or user groups responsible for the various identified tasks/functions:

- Development of the programs
  (Detail functions, considering; Who are the developers? What access required? What tools required?);
- Managing the source code repository
  (detail functions, considering; who can access the repository? Version controls? Code merging controls? etc);
- Define and creation of the workflow
  (detail functions, considering; how to start a project? what controls must be fulfilled before promoting to production? whose approval before promoting? etc)
- Define and creation of the operating environment and system configuration
  (detail functions, considering; how many environments? Who can spin up an environment? What system parameters to include? etc);
- Managing the security controls (e.g. passcode, certificate, etc)
  (detail functions, considering; who is managing the credential to other systems such as databases? Who is managing the certificates e.g. SSL? Who can define the security testing parameters, etc)

In addition, when establishing the RACI matrix, organisation should ensure proper segregation of duties/tasks to minimise the risk of:-

- Unauthorised promotion of fraudulent codes to production environment;
- Unauthorised modification of the environment parameters;
- Unauthorised modification of the security controls.

2. Audit Trail

Organisation should enable auditing/tracking features throughout the whole DevOps process. If not, the minimum audit trails captured must be enough for forensic investigation to determine when an individual access the system, what activities were performed by the individual, when were the activities executed, etc.

Furthermore, Organisation may also wish to explore the possibility of piping the audit log a central log tracking system to eliminate the possibility of unauthorised access and modification of the audit log.

3. Access Controls

Appropriate access right, in accordance with the identified role, should be granted to the role to enable the user, assigned with the role, to carry out his responsibilities.

The privileges granted should be based on the "least privileges" approach. In addition, processes should be established and formalised to govern:-

- The proper creation of user account and assignment of roles;
- Proper deletion of user accounts;
- Period review to identify dormant user account or user account no longer required at an earliest possible time.

Organisation should consider stronger authentication mechanism (i.e. 2 factor authentication) when accessing the pipeline from external network/Internet to perform critical functions such as modification to the workflow, modification to the security acceptance threshold, configuration, etc.

4. Integrity

In the agile environment, various tasks could be in developments at the same time. In such cases, a single point of truth is important. The deployed process model should also track the current ***production runtime baseline*** (PBL) to ensure each development team uses the source code that represent the current PBL. From security aspect, it ensure the development by each team has achieved certain level of security.

Access to the source code repository should be restricted to ensure the integrity of the source code and the operating environment in each controlled environments: SIT, UAT, staging and Production. This is to ensure the configuration of the test and production environment is consistent and the testing is done in a state that represents the final deployment in production.

Besides establishing access control to the source code repository and systems configuration, organisation should establish processes to govern these operations. For example:-

- A process to define and update the production runtime baseline of the application (i.e. check in and check out of source code);

In addition, organisation should established a formal configuration management process to ensure update to the environment (i.e. patching) is consistent across all environments.

5. Security Testing

Security testing, one of the controls, that can be used to provide some level of assurance on the security of the developed application or software. Traditionally in waterfall methodology, security testing can be scheduled on the development/application/software before promoting it to production. However, in agile methodology, several releases may be occurring within a short period of time. This would make security testing prior to promotion costlier, time consuming and ineffective (i.e. another release before the issues in the previous release were resolved).

To balance between the DevOps principle of shorter development, faster release to production and assurance of the application security, this framework proposed the following approach instead of performing security testing prior to each promotion to production.

Releases	Penetration testing	bugs bounty	Source code review/scan	Vulnerability scanning
First launch (base release)	✓	✓	✓	✓
Major release***	✓	*	✓	✓
Minor release***	*	*	✓	✓
Periodic (e.g. 6 monthly, yearly, bi-yearly, etc)**	+	+	✓	✓

** denote : Organisation to determine the testing scope (i.e. only delta changes or a group of delta changes);*

*** denote : Full scope testing to be performed*

**** denote : Organisation to define what is major or minor releases. It can be based on its risk appetites or by percentage of changes or impact of changes or financial values*

+ denote : Alternate testing (e.g. year 1 perform penetration testing, year 2 perform bug bounty, etc)

Type of testing

- Penetration testing:
  - Objective is to find the weaknesses within the application or system by attacking it. The attacks can be performed by selective security team/consultants. The test should be performed in a secured environment.
  - Instead of performing this testing before every promotion to production environment, it is recommended to perform at least once, on the full application, before it first introduction to the production environment. Subsequently, on each major release or periodically.

- Bugs Bounty:
  - Like Penetration testing, the objective of this exercise is to identify weaknesses within the system (i.e. application and servers). However, the differences been that this exercise would be performed by a larger and wider group of testers with various experience and skills and can be performed from any corner of the earth.

- Secured source code review/scanning: (secured) at the built stage
  - Objective is to find any security weakness associated with the way the application is coded.
  - Security source code review can be performed by:

- Manually (i.e. eyeballing)
  Review is performed by staff eyeballing the source codes. The effectiveness of this type of review depend very much on the reviewer's experiences and attentiveness during the review;

- Automatically (using scanning tools)
  Scanning of the application source codes is performed using scanning tools. The advantages of using automated tools are that testing are faster and consistent in testing methods and coverage. As such, it is commonly incorporated into the DevOps pipes.

  At times, the pipeline can be fully automated to promote the application to production environment basing on the threshold of the alerts or exception thrown up by the scanning tools. When doing so, organisation should determine the threshold level and establish process and procedures to:-
  - Periodically update the scanning tools;
  - Handle the exception or alert thrown up by the scanning tools (i.e. risk assessment and acceptance process).

- Vulnerability scanning:
  - Objective is to find any security weakness associated with the associated Operating Systems or middleware applications.
  - Nowadays, this type of scanning is commonly done using automated scanning tools. The difference is whether to perform the scan from external or internally, including infrastructure and OS registries.
  - As the tools very much depend on the databases of known vulnerabilities, organisation will have to establish process and procedure to periodically update the tools with latest vulnerabilities signatures.
  - Like any scanning tools depending on known signature, these scanning tool would have difficulty in identifying zero-day vulnerabilities. As such, organisation will have to incorporate other means to protect the system against these zero-day vulnerabilities.

Security checklist for DevOps

No	Controls Expectation	Comments
1	*DevOps Workflow Management*	
1.1	*Has the organisation established the DevOps flow?*	
1.2	*Control and checks points incorporated within the workflow?*	
1.3	*Is security management incorporated as part of the workflow?*	
1.4	*Has the necessary functions and access been identified and assigned?* - *Who can administer the workflow?* - *Who can change or update the workflow?* - *Who can create/remove an environment?*	
2	RACI Matrix	
2.1	*Has organisation established the RACI matrix?* *Example:* - *Software development team;* - *Repository management team;* - *Security team;* - *Configuration management team;* *etc*	
3	Source Code Management	
3.1	*Is there a source code repository?*	
3.2	*Does it have version control feature?*	
3.3	*Does it have source code merger feature?*	
3.4	*Does it track who check out the codes and when?*	
3.5	*Is access control in place over this repository?*	
3.6	*Is access setup in accordance with RACI matrix?*	
4	Configuration Management	
4.1	*Is there a mean to establish the production runtime baseline ("PRB") (i.e. OS, database, web, middleware, built baseline, workflow pipeline, tools, components, containers, etc)?*	
4.2	*Continuous security scanning and update of the golden image*	
4.3	*Is there a mean to ensure all systems and environments are having the same PRB?*	
4.4	*Is there process in place to test patches or changes before upgrade to PRB?*	
4.5	*Has the necessary functions and access been identified and assigned?* - *Who can access this repository?* - *Who can create the PRB?* - *Who can replace or update the PRB?*	

	- *Who can deploy the PRB?*	
5	Built/System Management	
5.1	*Is there a strategy for build-once and deploy multiple?*	
5.2	*Is there a built repository?*	
5.3	*Has the necessary functions and access been identified and assigned?* - *Who can access this repository?* - *Who can create the build?* - *Who can deploy the build?*	
5.4	*Is there means to ensure the built complies with the organisation's security standards?*	
6	Security Management	
6.1	*Has the software/system overall architecture been assessed?*	
6.2	*Is there security scanning tools (e.g. source code scanning and vulnerabilities scanning)?*	
6.3	*Has the requirement of the security scanning tools been established?*	
6.4	*How would these tools provide assurance to the systems or applications?*	
6.5	*Is source code review part of the workflow requirement before promoting to production?*	
6.6	*Is independent penetration testing a requirement of the workflow before promoting to production?*	
6.7	*Has the threshold on security findings set on the security scanning tool (i.e. low or informative)?*	
6.8	*Has the organisation formalised an exception handling process?*	
7	Code Promotion Management	
7.1	*Has organisation identified the key controls/factors that are to be met before promoting the codes? (e.g. testing sign off, security testing sign off, etc)*	