International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

DevSecOps Support on Continuous Integration Deployment of TRAC Applications for Mobile iOS and Android with Continuous Integration Method

Bagus Permadi[1]*
[1]Master of Information Systems Management Department, Business Information System,Gunadarma University, Indonesia
*e-mail: bagus.permadi5 @ gmail.com, ssiregar @ staff.gunadarma.ac.id

Abstract— *In the business world, the application of information and communication technology is currently needed as a tool so that the organization can be more advanced and develop. This can be helped by the presence of an IT vendor. Since its establishment in 1986, Trac has developed a car rental application named Trac Service Auto and was developed directly by the IT Vendor PT. Agit As fellow members of the Astra Group. From several problems related to processes that are often delayed due to problems with External and Internal System Integration, manual deployment and security issues that have not been implemented, this encourages the author to analyze and utilize the DevSecOps system which allows a shorter and safer development cycle. In this paper the authors use the Continuous Integration method in the need for continuous deployment code and integration to server apps as well as explore how to focus on code security incident management and limited access issues for the developer team by emphasizing collaboration with server or system access rights owners (Internal team) so that can contribute to sticking to the software security standards that are applied.*

Keywords— *Continuous Deployment Azure Pipeline, Continuous Deployment, DevSecOps, Continuous Integration.*

I. INTRODUCTION

The term DevSecOps was first used in 2012 by Neil MacDonald to integrate security into DevOps practices without affecting the speed and agility of the software development process [1]. DevSecOps is a metrology file where security is integrated into the entire application development lifecycle and not just a concept or theory [2]. Many experts and organizations have adopted DevOps and DevSecOps in the Software Development Life Cycle (SDLC). The DevOps digital marketing market may grow from $3.4 billion in 2018 to $10.3 billion in 2023 and DevSecOps will grow from $1.5 billion to $5.9 billion in 2023 [3].

DevSecOps practice is based on the 4 principles of Culture, Automation, Measurement and Sharing (CAMS) for the successful implementation of security into the development life cycle [3]. Devsecops Culture focuses on collaboration between development, security and operational teams to share responsibility, Devsecops Automation focuses on automating Security control systems and development processes that do not reduce the speed and effectiveness of development flow, Devsecops Measurement focuses on optimizing the use of monitoring and metrics to measure vulnerability and threats, and Devsecops Sharing is a part that supports knowledge sharing among all related teams with the aim of integrating security throughout the development process.

DevSecOps refers to the integration of security practices into the DevOps software delivery model. The foundation is a culture in which development and operations are enabled through processes and tools to share in the shared responsibility to deliver secure software. Before focusing on the understanding and implementation of DevSecOps, we must first understand the origins of DevOps.

DevOps is not a new development paradigm, but in recent years in DevOps the boundaries between developers and operations teams are being lowered, sometimes to the point where the slogan "You build it - you run it" becomes a reality. even on most DevOps systems operating in the Cloud, this is also coupled with continuous deployment, where new versions of system software can be deployed multiple times a day. This working mode really takes system reliability and agility to a new level, although doubts remain about how to ensure the security of new versions of software without changing the security test standards (Security). The answer to these doubts lies in implementing incident management.

In connection with Devops and Devsecops, this study uses Continuous Deployment (CD) for software development practices that enable organizations to deploy software to customers continuously, automatically, and reliably [4]. A number of innovative organizations such as Facebook, Microsoft, and IBM are adopting CDs to frequently deliver value to their customers. CD brings several benefits to organizations [5]. These benefits include reducing developer effort, improving software quality, and reducing costs [6]. Continuous Deployment Pipeline (CDP) is a core concept for successfully implementing CD practices [7].

CDP automatically transfers code changes from the repository to the production environment. In addition, CDP allows team members to keep an eye on every aspect (e.g., build, deploy, test, etc.) of the system, and get quick feedback on the implemented software. CDP also promotes collaboration between different groups of developers who work together to fix bugs and problems and deliver software by increasing the visibility of changes [8]. CDP is a collection of stages (e.g., build, package, and test) supported by tools (GitHub, Azure Devops, AWS, etc.) and technologies to enable continuous and automated deployment of changes into production. The number and nature of the stages involved in CDP vary from organization to organization [9]. Likewise, the tools and technologies incorporated for CDP implementation also vary from project to project and organization to organization.

The company that is the subject of this research discussion is Trac, a company that is one of the Corporate Operation business units of PT Serasi Autoraya Tbk and is part of PT Astra International Tbk Group as the logistics business unit of the Astra Group. engaged in four-wheeled vehicle rental, since 1986 has developed the Trac Service Auto application in the form of development in the form of a mobile application on iOS and Android, the application has been implemented in most Trac branches spread across Indonesia. The company that is the vendor of Trac's system development is AGIT, Agit is a Digital Service Provider, providing one-stop Solutions for Digital Services. Just like TRAC, AGIT is part of PT Astra International Tbk Group. PT Astra International Tbk Group is a conglomerate company in Indonesia which currently has 235 subsidiaries and affiliated companies with 7 business fields. In relation to the implementation phase of the TRAC application and as part of the information system development life cycle, it is necessary to integrate source code management (SCM) into internal server, Code security testing and the deloyment process which is monitored until the process is completed in the post-development build. The results of this process will then be used as a reference for the development of the Trac Application in other branches and for the evaluation of the Trac Internal Team itself.

II. Literatur Review

A. *DevSecOps*

Embedding security in the CI/CD pipeline, automated monitoring and setup in both Development and Production Environments is very useful in finding bugs and vulnerabilities especially in teams with no experience or without the security knowledge or expertise to manually find and assess vulnerabilities. Myrbakken and Colomo-Palacios explain the meaning of DevSecOps, the benefits and challenges of adopting the practice and how it has evolved since it was first introduced [10]. DevSecOps is meant to change the mindset of everyone in ensuring the security of automated development processes. secure DevOps workflows and how organizations can embed continuous security testing. in it Continuous delivery pipeline[11]. Perhaps one of the most relevant jobs in terms of integrating security in a DevOps environment. Khan discusses security controls, tools, automated checks/testing and best practices to ensure software is tested at every stage of development[11]. While there are several studies and publications on DevOps and DevSecOps, none of them address or focus on security-savvy development teams. focus on optimizing deployment of flow deployments effectively by adapting DevSecOps, benefits and challenges they may face.

B. *Continuous Integration*

Continuous Integration is a software development process where developers integrate code into a shared repository several times a day, allowing the software development team to detect problems early [12]. Continuous integration recommends that build automation and test automation be part of the development pipeline. It also recommends the use of a revision control system. The DevSecOps journal presents how to implement continuous dynamic security testing in a CI/CD pipeline and investigates the pitfalls of such testing.

The related journal theory explains that not much literature focuses on the dynamics of Static Application Security Testing (SAST) security testing and describes how to integrate appropriate tools to scan for vulnerabilities in workflows [13]. Static Application Security Testing (SAST) is used to secure software by reviewing software source code to identify the source of vulnerabilities. Although the static process of analyzing source code has been around for as long as computers have existed, the technique spread to security in the late 90s and the first public discussion of SQL injection in 1998 when Web applications integrated new technologies such as JavaScript and Flash[14]. DevSecOps practice is based on four principles (CAMS) for the successful implementation of security into the development life cycle[13].

Culture: The DevSecOps culture promotes shared responsibility for security and promotes collaboration between development, security, and operations teams. Every department has to integrate security in their work and that means security people have to be involved from the early stages of the project [13]. DevSecOps is about inclusion and working together as a team [15] and tends to dispel the traditional practice of having separate silos.

Automation: DevSecOps focuses on 100% automation of security controls and processes in a way that won't compromise speed and agility [16]. Software testing activities are carried out automatically by using test equipment (software) to do whatever human testers do manually. And it's not just about testing and deployment, it includes release management, configuration management, monitoring [13]. However, test automation cannot eliminate or completely replace manual testing because it is not possible to automate all test cases. Manual checking is essential in some cases because certain errors or problems such as authentication and authorization are impossible to detect by automated testing tools [16]. Measurement: DevSecOps encourages team use of monitoring and metrics to measure vulnerabilities and threats, which is important for keeping performance records and improving software quality [16]. Everything relevant has to be measured and teams cannot improve their product if measurement is abandoned. Sharing: DevSecOps supports knowledge sharing between all teams with the aim of integrating security into every process. This is education and cross-training for each member of the development, operations, and security team [17] on their security duties. Process security can only be improved when teams are constantly sharing the challenges they face and how they can help each other [17].

C. *DevSecOps System Building tool*

a. *Sonarqube*

he most common Open Source application code analysis tools adopted both in academia [18], [19] and in industry [20]. SonarQube is provided as a service from sonarcloud.io or can be downloaded and run on a private server. SonarQube calculates metrics such as line number of code and code complexity, and verifies code compliance with a specific set of "coding rules" defined for the most common development languages. In case the analyzed source code violates the coding rules or if the metric falls outside a predefined threshold, SonarQube generates an "issue". SonarQube covers Reliability, Maintenance and Security Rules. Sonarqube's security scan process which consists of GET and SEND analysis. Scanning results are schematically presented in Figure 1.

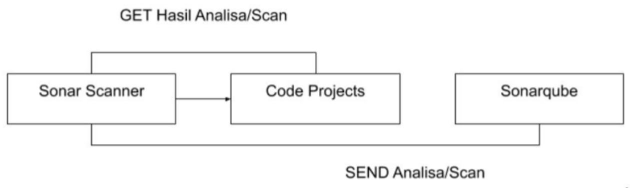


Fig. 1. Security Scan Sonarqube Process

b. Azure Devops

Microsoft Azure DevOps is a software as a service SaaS (Software as a service) platform that offers users an end-to-end DevOps tool chain for developing and deploying software. Azure DevOps is not a single program but consists of the following services: Azure Board: It includes agile planning, work item tracking, and visualization, as well as reporting tools. Azure Pipelines: It is a cloud-agnostic language, platform, and CI/CD platform with support for containers or Kubernetes. Azure Repos: It offers a private Git repository hosted in the cloud, with pull requests, advanced file management, and other benefits. Azure Artifacts: Artifacts in question provide developers with integrated package management, including support for Maven, npm, Python, and NuGet package feeds from public or private sources. Azure Test Plans: This service provides a unified, all-in-one planned, and exploratory testing solution [21]

c. Gradle

Gradle is a flexible and powerful build tool. Gradle is a common Android library building tool that allows developers to build software with any tool, because Gradle only creates a small amount of Logic Code about what the user/developer is trying to create or how to build it. The most notable limitation is that dependency management currently only supports Maven and Ivy compatible repositories and file systems. This doesn't mean developers have to do a lot of work to create build code. Gradle makes it easy to create common projects - for example Java libraries - by adding layers of conventions and built-in functionality via plugins. Users and developers can even create or publish custom plugins to encapsulate your own conventions and build functionality. 2. Core models are based on tasks Gradle models its builds as Directed Acyclic Graphs (DAGs) of tasks (work units). That is, the build essentially configures a set of tasks and splices them together - based on their dependencies - to create those DAGs. After the task graph is created, Gradle determines which tasks need to be run in which order and then proceeds to execute them[22].

d. Xcode

Xcode is Apple's IDE, created for producing software on Macs for use on iOS, iPadOS, macOS, tvOS, and watchOS. Free to download and use, the IDE is primarily used by developers to create iPhone and iPad applications, as well as programs for Mac [23].

e. Git

Git was created in 2005 by Linus Torvalds with the aim of sustaining the development of the Linux kernel [24]. Since then, it has been used to maintain thousands of projects. Various online Git repositories exist, including GitHub, GitLab, and BitBucket. The advantage of Git is that everything is stored locally, in the .git directory. This makes adding functionality simple as one only needs to access the files in the .git directory

f. App Center

Visual Studio App Center is an integrated mobile development lifecycle solution for iOS, Android, Windows, and macOS applications. It brings together several services commonly used by mobile developers, including building, testing, distributing, monitoring, diagnostics, etc., into one single integrated cloud solution [25].

g. Android

Android According to Arifianto (2011), Android is a mobile device on an operating system for Linux-based cellular phones. According to Hermawan (2011), Android is a Mobile OS (Operating System) that is growing in the midst of other OSes that are developing today. Android is an operating system designed by the Google company based on the Linux kernel and also various software such as Open Source and others. Mobile phones that use Android can be used for devices with touch screens such as smartphones and tablet computers [13].

h. iOS

iOS, is an operating system developed and distributed by Apple Inc. [24] which was launched in 2007 for the iPhone and iPod Touch, and has been developed to support other Apple devices such as the iPad and Apple TV. Apple does not license iOS to be installed on non-Apple hardware or in other words only specifically for Apple's own products.

D. Security Testing

Local Security Testing To prevent security experts on the software project team from wasting their time finding types of software vulnerabilities that are not sophisticated, we recommend that these types of vulnerabilities be detected and fixed before being added to any branch of the main project repository. To achieve this, the security experts on the project team must create a Security testng test case for the type of vulnerability relevant to the project and distribute it to all developers. Every developer, then, has to run these test cases against their code before they commit them to the main repository. If a vulnerability is detected in their code, Security Testing will inform them with easy-to-understand information about the type of vulnerability, its location, and how it can be triggered (Lescisin, Mahmoud and Ciorac, 2019).

E. Deployment

In programming, this process is called deployment. Deployment is an activity that aims to deploy applications that have been done by people who are experts in the field of programmers. The method of distribution is also very diverse, depending on the type of application. If you choose a Web application, then you will be hosted on a server. Meanwhile, if the application is mobile, there will be two deployments. The first is deployment for the application to the Playstore or Appstore, and the second is the deployment of the API (backend) to the server. To deploy you have to be extra patient because there will be a lot of unwanted things happening. An example of a constraint that is often experienced is a system that suddenly goes down, that's why it takes a long time to deploy a program (Wilde, et al., 2016).

III. Research Method

A. Research Flow

The research flow describes the sequence of the research process in detail and the relationship between a process (instructions) and other processes, as shown in Figure 2. This research begins with analyzing the ongoing TRAC application development process. This process is a collaboration between TRAC's internal team and the Agit team. This ongoing process analysis aims to determine which processes will be optimized and implemented in the DevSecOps system, the second stage is the Technology Selection stage that will be implemented in the DevSecOps process by selecting tools that suit the needs of Continuous Integration and Continuous Deployment as well as selecting tools for scanning code security compliant with DevSecOps standards.

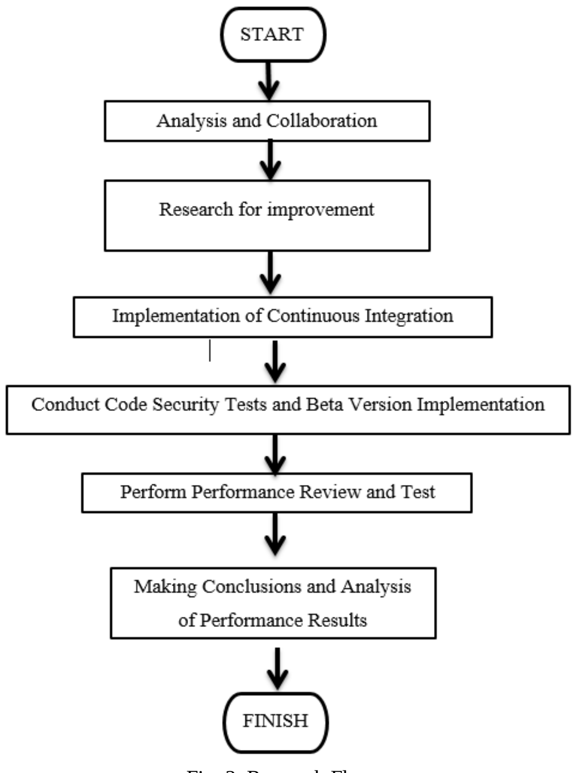

START
Analysis and Collaboration
Research for improvement
Implementation of Continuous Integration
Conduct Code Security Tests and Beta Version Implementation
Perform Performance Review and Test
Making Conclusions and Analysis of Performance Results
FINISH

Fig. 2. Research Flow

After the technology is selected according to the needs, then the third stage is carried out, the stage when it is carried out by testing the selected technological process. Testing is carried out from the beginning of the flow to the end of the flow in the local environment by applying the Continuous Integration method. After the entire cycle or full cycle of the process has been carried out in the local environment, proceed to the fourth stage. The fourth stage is the implementation of code security scanning according to standard i in a full full cycle which has been successful. If the full DevSecOps cycle is confirmed to be running well in the local environment, the next stage is the Review stage with the Agit Team and testing the application on the Server Staging Environment to compare the performance of the new system. At this final stage, analysis and conclusions are also made with the Agit team: whether the flow that has been implemented in the Staging Environment is in accordance with the needs of the user (Agit Developer). The research flow is schematically presented in Figure 2.

B. Collaboration and Analysis

The Collaboration phase with the Dev team from the PT. Agit Vendor is needed to analyze several processes that can be optimized with the DevSecOps System from existing manual processes that have been carried out by the vendor team with clients such as:

- Push Code process for Fixing or Enhance system with manual clone Source Code directly from Server
- The Build Code process is carried out directly on the Client Server and is carried out by the System Administrator from the Internal Client team
- The Code security scanning process is still being carried out individually from the dev team by making Unit tests on a personal locale without the Quality Code Static Application Security Testing (SAST) standard.
- The process of Publishing and Sharing Beta Versions of iOS (.ipa) and Android (.apk) which is still manual Using Flash Drive media
- The process of the Report Monitoring Log and Result from the Build which is still manual is attached via email

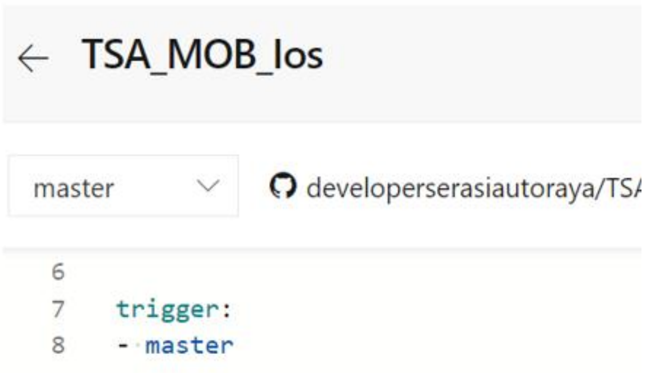
C. Research and Explore Improvement

Research and Explore carried out to find several main components of the DevSecOps system that can be implemented for the Continuous Integration method on existing flow deployments are aimed at reducing developer work, such as:

- Perform manual Testing Code and change it to automatic
- Doing manual build code becomes automatic
- Publishing (.ipa) and (.apk) manually becomes automatic

D. Continuous Integration Implementation

The Github Source Code Management used by the vendor team has a feature to create branches based on the environment they are working on which is enabled for Continuous Integration needs to several tools such as Azure DevOps. The trigger will be connected according to the action on a particular branch that is pushed or on a pull request. This process is the last point of the developer and the next process will be carried out entirely by the Devesecops system with the continuous integration method. Figure 3 shows the Trigger Branch Azure Devops Pipeline.



```
trigger:
- master
```

Fig. 3. Trigger Branch Azure Devops Pipeline

E. Conducting Code Security Tests and Beta Version Implementation

In Azure DevOps, the github webhook that is set in the Azure Devops pipeline will automatically trigger the build according to the trigger config created in the azure pipeline with actions or commit code activities to branches such as 'master' or 'dev' on github. And after the Azure pipeline build is started, the sonar scan or security testing process in the code section using the sonarqube plugin that has been added in the Continuos Deployment step will scan the code in several steps according to the metrics and options that have been designed in the scan quality code parameter. will be used. Figure 4 shows the Sonarqube Analysis Flowchart.

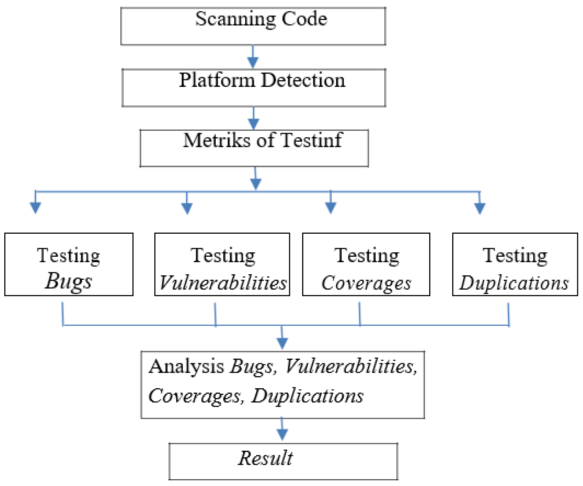
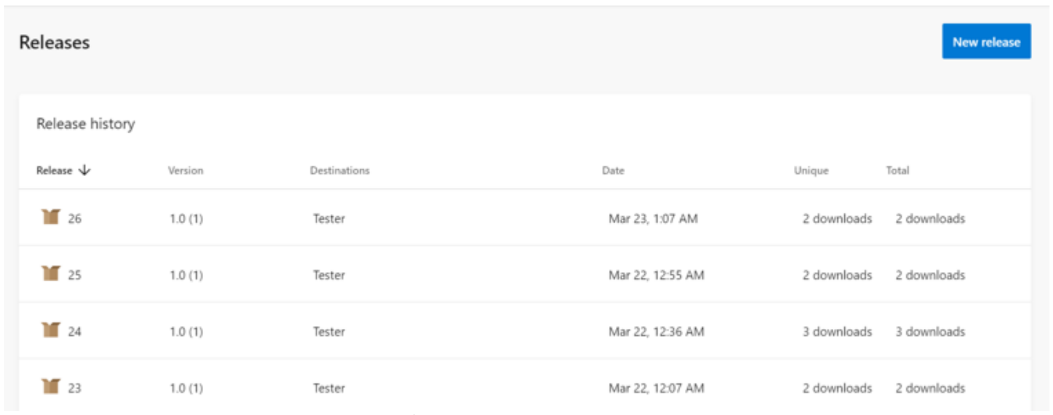


Fig. 4. Flowchart Analisis Sonarqube

- Azure Pipeline Gradle Once Code Passed Test Results it will continue for Build Gradle if deployment is for Android Mobile
- Azure Pipeline Xcode After Code Passed Test Results it will continue to Build Xcode if the deployment is for iOS Mobile
- AppCenter After Build Completed, Files (.ipa) and (.apk) will be published to AppCenter tools. Figure 5 shows sample files (.ipa) and (.apk) that have been successfully published to the App Center from the Automatic Build results.



Release history

Release	Version	Destinations	Date	Unique	Total
26	1.0 (1)	Tester	Mar 23, 1:07 AM	2 downloads	2 downloads
25	1.0 (1)	Tester	Mar 22, 12:55 AM	2 downloads	2 downloads
24	1.0 (1)	Tester	Mar 22, 12:36 AM	3 downloads	3 downloads
23	1.0 (1)	Tester	Mar 22, 12:07 AM	2 downloads	2 downloads

Fig. 5. list of build success publish (ipa) asnd (.apk)

F. Conducting Performance Review and Test

In the Review and Validation Test Phase, the Agit Developer Team will review or review the Continuous Deployment cycle. The review is carried out from the beginning of the scenario until the entire cycle is completed assisted by the Internal DevOps team. This is to ensure that all processes carried out are in accordance with Automation Deployment Optimization as a substitute for every step that was previously done manually. In addition, time optimization analysis of the DevSecOps implementation was also carried out for each step. Figure 6 shows the process validation sequence for each Automation step.

G. Making Conclucions and Analyzing the results of performance

In the final stage of this research, conclusions and recommendations are drawn regarding the success of the DevSecOps System process on the Continuous Deployment of Mobile Apps iOS and Android Trac from the user side or Dev who is a direct user of this system. Standards required for a successful implementation of DevSevOps From a Code

Security point of view and system effectiveness in helping to increase the productivity of TRAC Application Deployment.

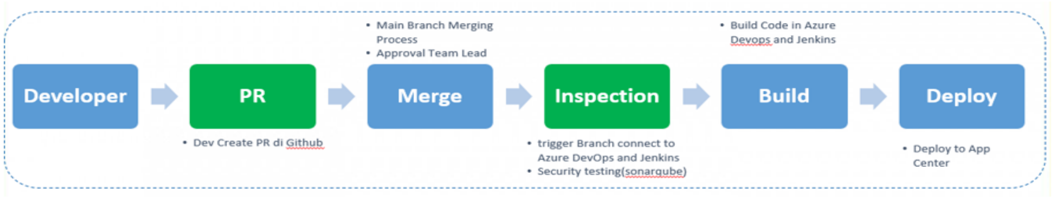

Developer
PR
Merge
Inspection
Build
Deploy
• Main Branch Merging Process
• Approval Team Lead
• Build Code in Azure Devops and Jenkins
• Dev Create PR di Github
• trigger Branch connect to Azure DevOps and Jenkins
• Security testing(sonarqube)
• Deploy to App Center

Fig. 6. Validation Process for every Autoamtion Step

IV. Research Results and Discussion

A. *Result of Analysis and Collaboration*

The Trac Service Auto application is an application that focuses on car rental for Corporate, Daily and BUS rental needs. In this Trac, what is the Trac Fleet Management Solution product for vehicle tracking and has been implemented in most Trac branches spread throughout Indonesia. This application has been implemented in the Beta version for Mobile Apps and iOS Apps in the form of enhancement (Rewrite). And in this Trac application, several development processes in this Beta version application will be investigated, namely the Integration process and Security Checking Code. The Trac Application Development process flow before and after DevSecOps implementation can be seen in figure 7.

B. *Result of Explore and Improvement*

The implementation process according to research begins with code installation on Azure Devops and the use of the Azure Pipeline Feature for the build process and creates a continuous integration scenario at each step according to the needs of the automation process, and the installation of the sonarqube plugin in the middle of the build process. At the end of the app center installation is done as an additional step for publication (.apk) and (.ipa) for each release. The design scenario is presented in Figure 8.

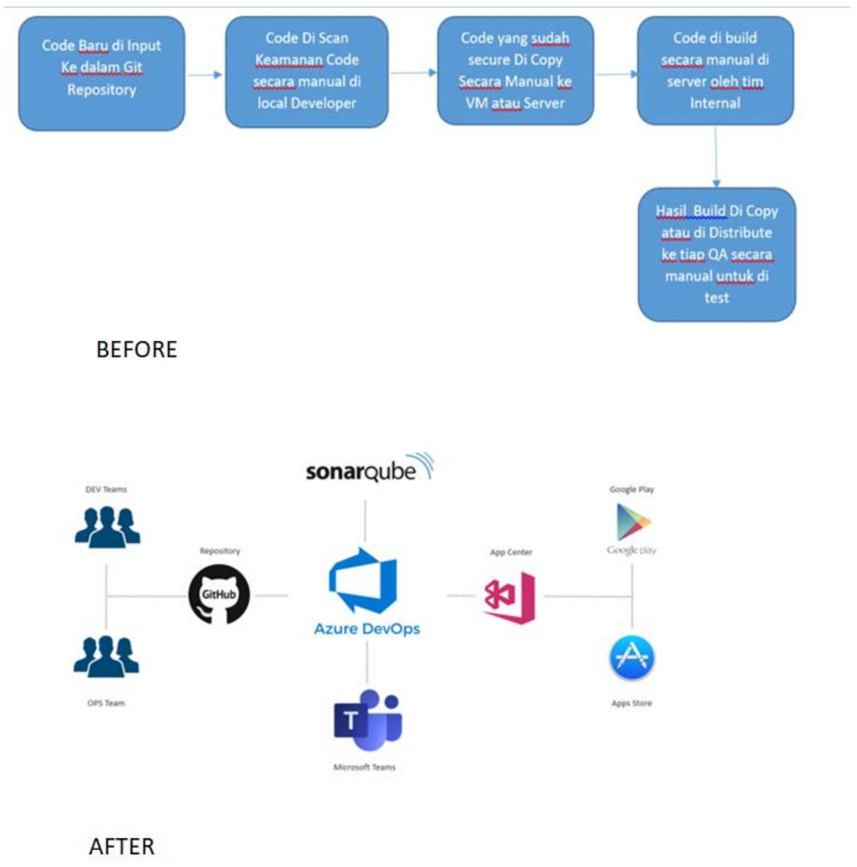

Code Baru di Input Ke dalam Git Repository
Code Di Scan Keamanan Code secara manual di local Developer
Code yang sudah secure Di Copy Secara Manual ke VM atau Server
Code di build secara manual di server oleh tim Internal
Hasil Build Di Copy atau di Distribute ke tiap QA secara manual untuk di test
BEFORE
DEV Teams
Repository
GitHub
sonarqube
Azure DevOps
App Center
Google Play
OPS Team
Microsoft Teams
Apps Store
AFTER

Fig. 7. Before and After Implementation of DevSecOps

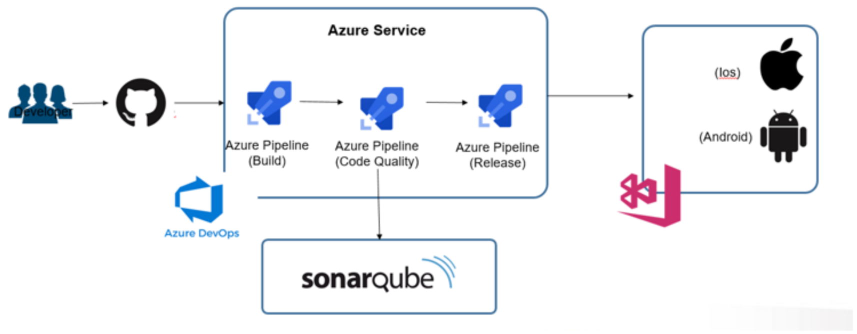


Fig. 8. Scenario Design of Devsecops

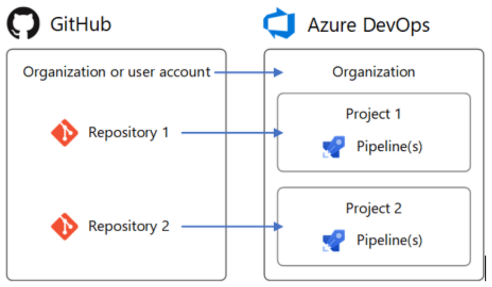


Fig. 9. Azure Devops Github Service Connection

C. *Implementation of Continuous Integration*

The Continuous Integration method implemented at the Source Code Management stage such as the use of webhook from Azure DevOps To change the Cloning code process which is still done manually from Server Apps and also change the scenario for the editing process or feature enhancement from the code side from manual to automation using Continuous Integration is explained in Figure 9 for automated deployments based on trigger actions from github.

D. *Conduct Code Security Tests and Beta Version Implementation*

Conducting Security Testing Code Security CU testing is performed in conjunction with the Continuous Deployment process using the Sonarqube plugin from Azure DevOps on the full lifecycle of the DevSecOps system. The flowchart of the iOS Automation build process to publish files (.ipa) and the Android build process to publish files (.apk) is presented in Figure 10 and Beta Version Implementation

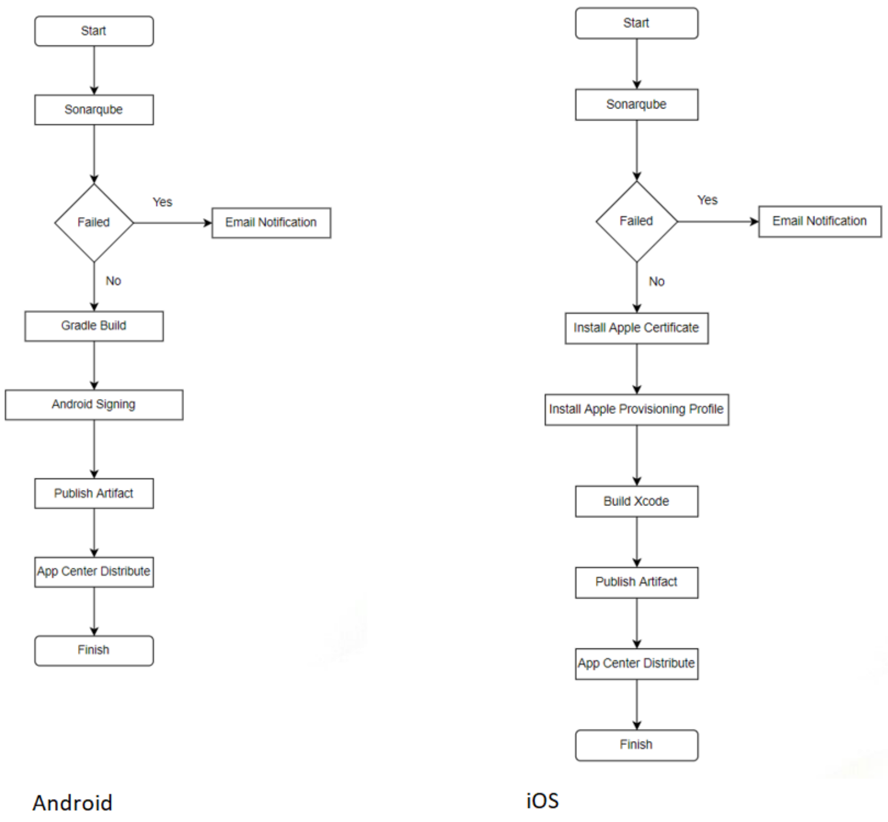


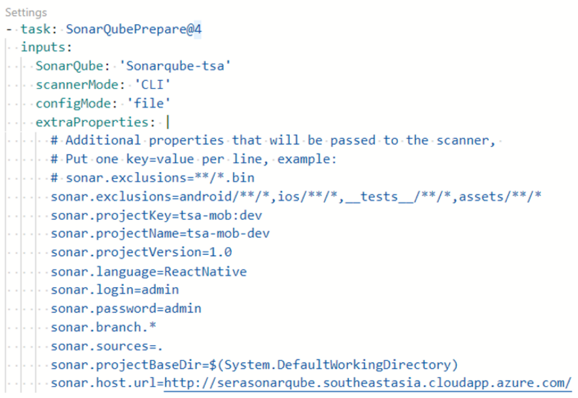
Fig. 10. Flowchart Build Azure Devops iOS and Android

1. *Sonarqube Testing Security Code*

Security testing using Sonarqube in the DevSecOps Build process flow with Azure DevOps focused on scanning security code according to the programming language that can be defined in the sonarqube azure pipeline properties. The full code of the azure pipeline for Sonar Properties is presented in Figure 11.

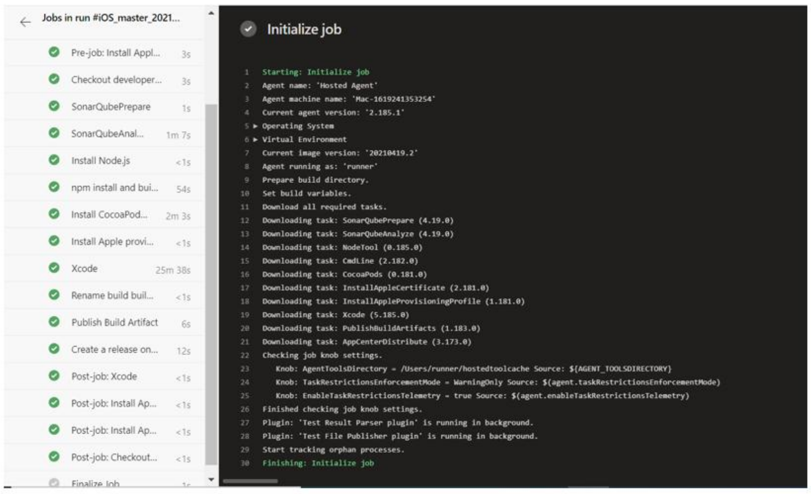
2. *iOS and Android DevSecOps Deployment full cycle*

Changed All Manual Build Processes that are still Performed on the apps server and on local developers also added the Security Testing process to a full reorganize process in Azure DevOps Pipeline by sorting processes according to dependencies from Android for builds (.apk) and adjusting iOS dependencies for builds (. ipa). Full logs of the azure pipeline for Android Build Properties and iOS Properties are presented in Figure 12 and Figure 13.



```
Settings
- task: SonarQubePrepare@4
  inputs:
    SonarQube: 'Sonarqube-tsa'
    scannerMode: 'CLI'
    configMode: 'file'
    extraProperties: |
      # Additional properties that will be passed to the scanner,
      # Put one key=value per line, example:
      # sonar.exclusions=**/*.bin
      sonar.exclusions=android/**/*,ios/**/*,__tests__/**/*,assets/**/*
      sonar.projectKey=tsa-mob:dev
      sonar.projectName=tsa-mob-dev
      sonar.projectVersion=1.0
      sonar.language=ReactNative
      sonar.login=admin
      sonar.password=admin
      sonar.branch.*
      sonar.sources=.
      sonar.projectBaseDir=$(System.DefaultWorkingDirectory)
      sonar.host.url=http://serasonarqube.southeastasia.cloudapp.azure.com/
```

Fig. 11. Sonar Properties Azure Pipeline

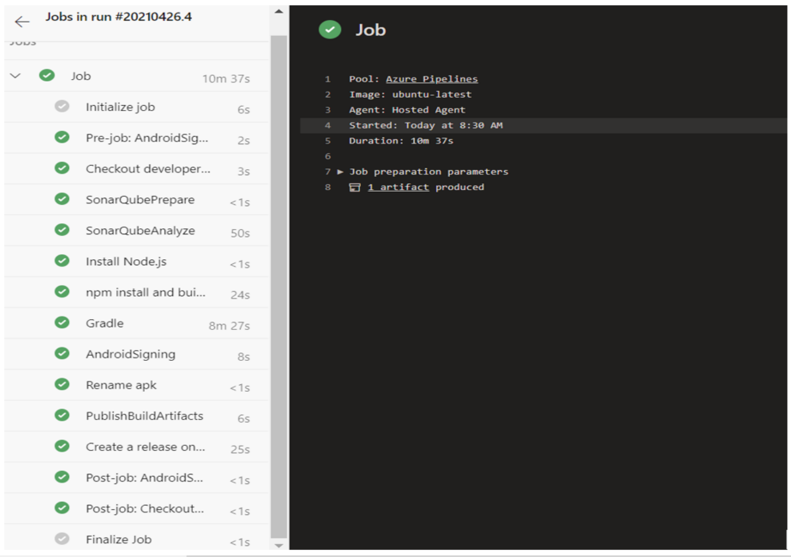


```
Jobs in run #iOS_master_2021...
Pre-job: Install Appl...      3s
Checkout developer...         3s
SonarQubePrepare              1s
SonarQubeAnal...          1m 7s
Install Node.js             <1s
npm install and bui...      54s
Install CocoaPod...       2m 3s
Install Apple provi...      <1s
Xcode                   25m 38s
Rename build buil...        <1s
Publish Build Artifact       6s
Create a release on...      12s
Post-job: Xcode             <1s
Post-job: Install Ap...     <1s
Post-job: Install Ap...     <1s
Post-job: Checkout...       <1s
Finalize Job                 1s
```

```
Initialize job

Starting: Initialize job
Agent name: 'Hosted Agent'
Agent machine name: 'Mac-1619241353254'
Current agent version: '2.185.1'
▶ Operating System
▶ Virtual Environment
Current image version: '20210419.2'
Agent running as: 'runner'
Prepare build directory.
Set build variables.
Download all required tasks.
Downloading task: SonarQubePrepare (4.19.0)
Downloading task: SonarQubeAnalyze (4.19.0)
Downloading task: NodeTool (0.185.0)
Downloading task: CmdLine (2.182.0)
Downloading task: CocoaPods (0.181.0)
Downloading task: InstallAppleCertificate (2.181.0)
Downloading task: InstallAppleProvisioningProfile (1.181.0)
Downloading task: Xcode (5.185.0)
Downloading task: PublishBuildArtifacts (1.183.0)
Downloading task: AppCenterDistribute (3.173.0)
Checking job knob settings.
   Knob: AgentToolsDirectory = /Users/runner/hostedtoolcache Source: ${AGENT_TOOLSDIRECTORY}
   Knob: TaskRestrictionsEnforcementMode = WarningOnly Source: $(agent.taskRestrictionsEnforcementMode)
   Knob: EnableTaskRestrictionsTelemetry = true Source: $(agent.enableTaskRestrictionsTelemetry)
Finished checking job knob settings.
Plugin: 'Test Result Parser plugin' is running in background.
Plugin: 'Test File Publisher plugin' is running in background.
Start tracking orphan processes.
Finishing: Initialize job
```

Fig. 12. Azure DevOps Build iOS



```
Jobs in run #20210426.4
Job                     10m 37s
Initialize job               6s
Pre-job: AndroidSig...       2s
Checkout developer...        3s
SonarQubePrepare            <1s
SonarQubeAnalyze            50s
Install Node.js             <1s
npm install and bui...      24s
Gradle                   8m 27s
AndroidSigning               8s
Rename apk                  <1s
PublishBuildArtifacts        6s
Create a release on...      25s
Post-job: AndroidS...       <1s
Post-job: Checkout...       <1s
Finalize Job                <1s
```

```
Job

Pool: Azure Pipelines
Image: ubuntu-latest
Agent: Hosted Agent
Started: Today at 8:30 AM
Duration: 10m 37s

▶ Job preparation parameters
1 artifact produced
```

Fig. 13. Azure DevOps Build Android

E. *Conductiong Review and Performance Test*

To obtain information on the performance of System Devsecops that has been implemented on Android and iOS deployments, performance tests are carried out by comparing the implemented System DevSecOps with the manual process usually carried out by the developer team by looking at the overall process and the total time required for publication. (.apk) and (.ipa) and analyze several steps in the optimization of the devsecops system to make the developer's work easier.

1. *Review implementasi DevSecOps*

From the results of the implementation of the Devsecops System to build APKs, there are several steps that are optimized to facilitate developer performance and can be handled directly by the devsecops system, as follows. Some detailed deployment steps that are automated with the Devsecops system for android and iOS builds are presented in Figure 14.

F. *Making Conclusions and Analysis of Performance Results*

The process of implementing the Continuous Integration method using Service Connection from Github and Azure DevOps which is described in Figure 15.

Is a process to support increasing Deployment Frequency with scenarios when developers make changes to coding or feature enhancements for development needs will trigger automatic deployments and carry out all processes automatically without much effort to be done manually, reducing failures on new releases because there is no Human Touch and Human Error in the deployment process, and also shorten the total time of deployment. A detailed comparison of the total time required for manual deployment and automation can be seen in Figure 16 and Figure 17.

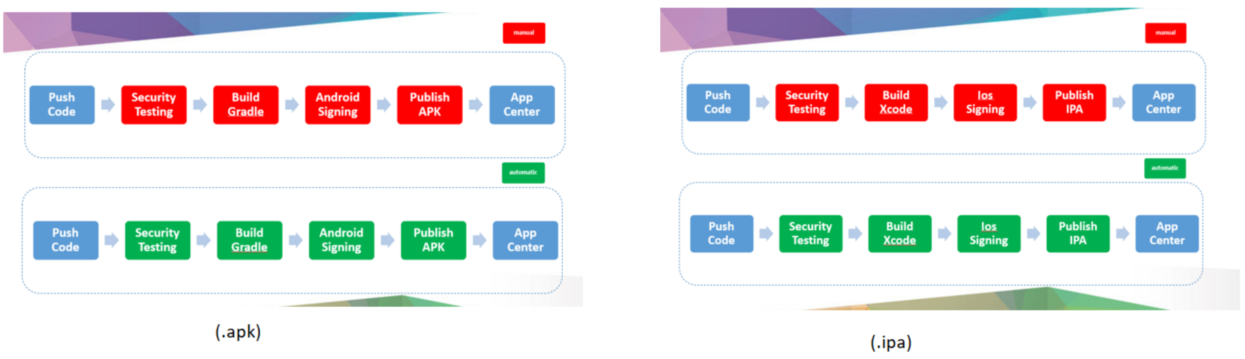
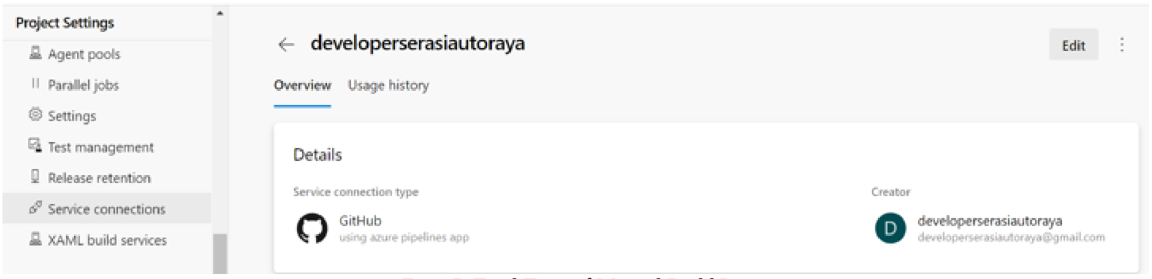


Fig. 14. Automation Transition from Manual Process



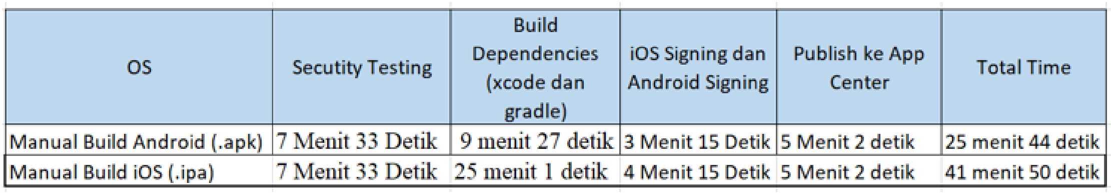
developerserasiautoraya

Details

Service connection type
GitHub
using azure pipelines app

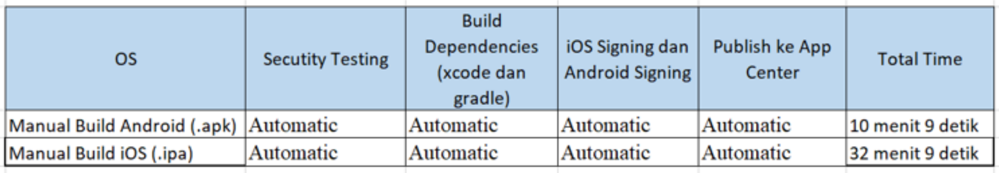
Creator
developerserasiautoraya
developerserasiautoraya@gmail.com

Fig. 15. Total Time of Manual Build Process



OS	Secutity Testing	Build Dependencies (xcode dan gradle)	iOS Signing dan Android Signing	Publish ke App Center	Total Time
Manual Build Android (.apk)	7 Menit 33 Detik	9 menit 27 detik	3 Menit 15 Detik	5 Menit 2 detik	25 menit 44 detik
Manual Build iOS (.ipa)	7 Menit 33 Detik	25 menit 1 detik	4 Menit 15 Detik	5 Menit 2 detik	41 menit 50 detik

Fig. 16. Total Time of Manual Build Full Process

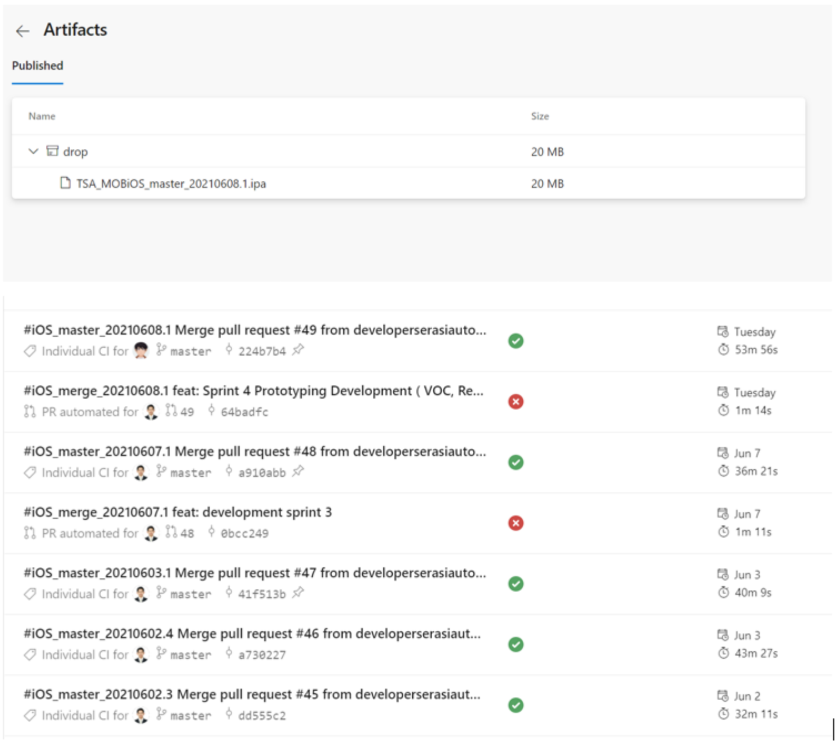


OS	Secutity Testing	Build Dependencies (xcode dan gradle)	iOS Signing dan Android Signing	Publish ke App Center	Total Time
Manual Build Android (.apk)	Automatic	Automatic	Automatic	Automatic	10 menit 9 detik
Manual Build iOS (.ipa)	Automatic	Automatic	Automatic	Automatic	32 menit 9 detik

Fig. 17. Total Time of Automation Build Full Process

The use of Artifact in Azure DevOps described in Figure 4.38 is a process that supports increasing recovery time or Rollback to the most stable release state for problematic deployment scenarios so that it can be immediately returned to normal conditions if business traffic is high. Figure 18 also shows a list of Artifacts with code numbers such as '22rb7b4' and '64badfc' as identification for rollback scenarios when selecting the most stable and successful artifact release.

Feature Secure File Support and Addition of Automation Security Testing presented in Figure 19 is a process to support standard Security Deployment by using the DevSecOps system to increase Code security in terms of Secret Transparency or key (Password) that need to be protected from Hackers or teams other than developers who have access to azure devops and improve code quality that supports application security standards on the source code side.



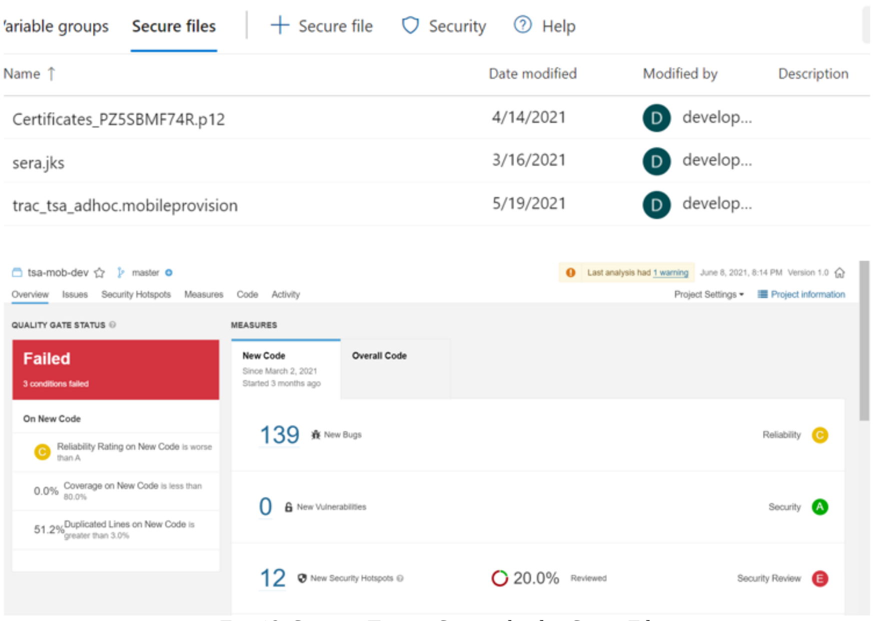
Artifacts

Published

Name	Size
drop	20 MB
TSA_MOBiOS_master_20210608.1.ipa	20 MB

#iOS_master_20210608.1 Merge pull request #49 from developerserasiauto... Individual CI for master 224b7b4	✓	Tuesday 53m 56s
#iOS_merge_20210608.1 feat: Sprint 4 Prototyping Development (VOC, Re... PR automated for 49 64badfc	✗	Tuesday 1m 14s
#iOS_master_20210607.1 Merge pull request #48 from developerserasiauto... Individual CI for master a910abb	✓	Jun 7 36m 21s
#iOS_merge_20210607.1 feat: development sprint 3 PR automated for 48 0bcc249	✗	Jun 7 1m 11s
#iOS_master_20210603.1 Merge pull request #47 from developerserasiauto... Individual CI for master 41f513b	✓	Jun 3 40m 9s
#iOS_master_20210602.4 Merge pull request #46 from developerserasiaut... Individual CI for master a730227	✓	Jun 3 43m 27s
#iOS_master_20210602.3 Merge pull request #45 from developerserasiaut... Individual CI for master dd555c2	✓	Jun 2 32m 11s

Fig. 18. Artifact Azure DevOps



'ariable groups   Secure files   + Secure file   Security   Help

Name ↑	Date modified	Modified by	Description
Certificates_PZ5SBMF74R.p12	4/14/2021	D develop...	
sera.jks	3/16/2021	D develop...	
trac_tsa_adhoc.mobileprovision	5/19/2021	D develop...	

tsa-mob-dev   master   Last analysis had 1 warning   June 8, 2021, 8:14 PM   Version 1.0

Overview   Issues   Security Hotspots   Measures   Code   Activity   Project Settings   Project information

QUALITY GATE STATUS

Failed
3 conditions failed

On New Code
C Reliability Rating on New Code is worse than A
0.0% Coverage on New Code is less than 80.0%
51.2% Duplicated Lines on New Code is greater than 3.0%

MEASURES

New Code
Since March 2, 2021
Started 3 months ago

Overall Code

139 New Bugs — Reliability C
0 New Vulnerabilities — Security A
12 New Security Hotspots — 20.0% Reviewed — Security Review E

Fig. 19. Security Testing Sonarqube dan SecureFile

V. CONCLUSIONS

Based on the testing and analysis results that have been carried out, the conclusions of this study are as follows.

a) The factors that influence the successful implementation of the DevSecOps System on the Trac application, including the Total Build Time variable with a value of 10 minutes 9 seconds for APK builds and 32 minutes 9 seconds for iOS builds have the advantage of increasing the deployment frequency of the Agit team which requires a total manual build time for APK is 9 minutes 27 seconds and 25 minutes 1 second for IPA but does not include the Security testing process which on average takes about 7 minutes if done manually and must be done by the Internal team and there needs to be more effort from the dev team to request via Email to the Internal team to run the test.
b) The success rate of DevSecOps system implementation The Trac application is influenced by several factors in the Continuous Integration method and the application of Continuous Deployment, namely the full Automation Deployment process to help reduce more effort from the user at each step. which provides several benefits such as:
    1. Reducing User Interference or the number of accesses that take part in the deployment process which causes frequent failure of the deployment process due to Human Errors and reduces the level of system security by changing all manual deployment processes to automatic.
    2. Facilitate the work of the Agit team in terms of time effectiveness because all processes that are usually done manually and require an approval process from the security team with a total amount of time can be completed quickly and easily.
c) Recommended Use of Azure DevOps Artifact Feature for DevSecOps System Implementation.
    1. With Artifact Publication in every deployment process the Recovery Scenario becomes easier and faster.
    2. Recovery or Rollback security is more secure because Developers no longer need to do backups or make mistakes because they forget to backup.

Security Testing Automation process using Sonarqube and the use of Secure File Azure Devops Feature for securing key and password files to support code security standardization for the deployment process in accordance with the implementation of System DevSecOps.

REFERENCES

[1] R. Kumar and R. Goyal, “Modeling continuous security: A conceptual model for automated *DevSecOps* using open-source software over cloud (ADOC),” Comput. Secur., vol. 97, p. 101967, 2020, doi: 10.1016/j.cose.2020.101967.

[2] J. Caraballo-vega, “Pipelines Use Case : Docker Container Scanning BUILD CLEANUP Use Case : Black Box Enumeration of System,” no. August, 2019.

[3] T. Rangnau, R. v. Buijtenen, F. Fransen, and F. Turkmen, “Continuous Security Testing: A Case Study on Integrating Dynamic Security Testing Tools in CI/CD Pipelines,” pp. 145–154, 2020, doi: 10.1109/edoc49727.2020.00026.

[4] Claps, G. G., Svensson, R. B. and Aurum, A. (2015) 'On the journey to continuous deployment: Technical and social challenges along the way', Information and Software Technology, 57, pp. 21-31.

[5] Anderson, K. H., et al. (2014) 'Continuous deployment system for software development.', U.S. Patent No. 8,677,315

[6] Chen, L. (2015) 'Continuous delivery: Huge benefits, but challenges too', IEEE Software, 32(2), pp. 50-54.

[7] Humble, J. and Farley, D. (2010) Continuous delivery: reliable software releases through build, test, and deployment automation. Pearson Education.

[8] Fowler, M. (2013) 'Deployment pipeline. Available at http://martinfowler.com/bliki/DeploymentPipeline.htm l [Last Accessed: 24th Oct, 2016]'.

[9] Adams, B. and McIntosh, S. (2016) 'Modern Release Engineering in a Nutshell -- Why Researchers Should Care', IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), pp. 78-90.

[10] H. Myrbakken and R. Colomo-Palacios, “*DevSecOps*: A multivocal literature review,” Commun. Comput. Inf. Sci., vol. 770, no. September, pp. 17–29, 2017, doi: 10.1007/978-3-319-67383-7_2.

[11] M. O. Khan, “Fast Delivery, Continuously Build, Testing and Deployment with *DevOps* Pipeline Techniques on Cloud,” Indian J. Sci. Technol., vol. 13, no. 5, pp. 552–575, 2020, doi: 10.17485/ijst/2020/v13i05/148983

[12] ThoughtWorks. Continuous Integration. Available online: https://www.thoughtworks.com/continuousintegration (accessed on 20 September 2019)

[13] T. Rangnau, R. v. Buijtenen, F. Fransen, and F. Turkmen, “Continuous Security Testing: A Case Study on Integrating Dynamic Security Testing Tools in CI/CD Pipelines,” pp. 145–154, 2020, doi: 10.1109/edoc49727.2020.00026.

[14] Static application security testing. Available online: https://en.wikipedia.org/wiki/Static_application_security_testing (accessed on 4 April 2021).

[15] R. B. Salesforce and K. Carter, “SOFTWARE ENGINEERING Francois Raynaud on DevSecOps,” no. October, pp. 93–96, 2017.

[16] H. Yasar and K. Kontostathis, “Where to Integrate Security Practices on DevOps Platform,” Int. J. Secur. Softw. Eng., vol. 7, no. 4, pp. 39–50, 2017, doi: 10.4018/ijsse.2016100103.

[17] M. Sánchez-Gordón and R. Colomo-Palacios, “Security as Culture: A Systematic Literature Review of DevSecOps,” Proceedings - 2020 IEEE/ACM 42nd International Conference on Software Engineering Workshops, ICSEW 2020, pp. 266–269, 2020.

[18] Valentina Lenarduzzi, Alberto Sillitti, and Davide Taibi. Analyzing forty years of software maintenance models. In 39th International Conference on Software Engineering Companion, ICSE-C ’17, pages 146–148, Piscataway, NJ, USA, 2017. IEEE Press.

[19] Valentina Lenarduzzi, Alberto Sillitti, and Davide Taibi. A survey on code analysis tools for software maintenance prediction. In 6th International Conference in Software Engineering for Defence Applications, pages 165–175. Springer International Publishing, 2020.

[20] Carmine Vassallo, Sebastiano Panichella, Fabio Palomba, Sebastian Proksch, Harald C. Gall, and Andy Zaidman. How Developers Engage with Static Analysis Tools in Different Contexts. In Empirical Software Engineering, 2019.

[21] Azure Devops. Available online: https://www.simplilearn.com/azure-devops-article (accessed on 4 April 2021).

[22] Five things you need to know about Gradle. Available online: https://docs.gradle.org/current/userguide/what_is_gradle.html (accessed on 4 April 2021).

[23] Xcode. Available online: https://appleinsider.com/inside/xcode (accessed on 4 April 2021).

[24] Chacon, S.; Straub, B. A Short History of Git. Available online: https://git-scm.com/book/en/v2/GettingStarted-A-Short-History-of-Git (accessed on 4 April 2021).

[25] Build-Test-Distribute Mobile Apps using App Center. Available online: https://www.azuredevopslabs.com/labs/vstsextend/appcenter (accessed on 4 April 2021).

[26] Michael Lescisin, Qusay H. Mahmoud , and Anca Cioraca, Design and Implementation of SFCI: A Tool forSecurity Focused Continuous Integration 1 November 2019.

[27] Faheem Ullah1, Adam Johannes Raft2, Mojtaba Shahin1, Mansooreh Zahedi2 and Muhammad Ali Babar1,2 Security Support in Continuous Deployment Pipeline March 2017.

[28] Martin Gilje Jaatun, Software Security Activities that Support Incident Management in Secure DevOps, August 2018

[29] Norman Wilde, Brian Eddy, Khyati Patel, Nathan Cooper, Valeria Gamboa, Bhavyansh Mishra, Keenal Shah Security For *Devops* Deployment Processes: Defenses, Risks, Research Directions November 2016.

[30] Bakary Jammeh *DevSecOps*: Security Expertise a Key to Automated Testing in CI/CD Pipeline. December 2020.

[31] Paul Swartout (2012), Continuous Delivery and *DevOps*: A Quickstart Guide, Packt Publishing Birmingham – Mumbai.

[32] R. B. Salesforce and K. Carter, "Software Engineering Francois Raynaud on DevSecOps," no. October, pp. 93–96, 2017.

[33] H. Yasar and K. Kontostathis, "Where to Integrate Security Practices on DevOps Platform," Int. J. Secur. Softw. Eng., vol. 7, no. 4, pp. 39–50, 2017, doi: 10.4018/ijsse.2016100103.

[34] Suren Machiraju,Suraj Gaurav (2018), *DevOps* for Azure Applications, Apress

[35] Tarun Arora, Utkarsh Shigihalli (2019), Azure *DevOps* Server 2019, , Packt Publishing Birmingham – Mumbai.

[36] A. Jl, P. Wr, and S. Dm, "The emerging soft tissue paradigm in orthodontic diagnosis and treatment planning," Clinical Orthodontics and Research, vol. 2, no. 2. pp. 49–52, 1999.