Journal of Theoretical and Applied Information Technology
15th July 2023. Vol.101. No 13

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

IMPROVING INTEGRITY, SECURITY, AND ACCURACY DURING DEVOPS PROCESS

HANAN FAHMY[1], SAMAR SAMIR[2], MONA NASR[3]

[1, 2, 3] Department of Information Systems, Faculty of Computers and Artificial Intelligence, Cairo, Egypt

E-mail: [1]hanan.fahmy@fci.helwan.edu.eg, [2]samar.samir@fci.helwan.edu.eg,

[3]drmona_nasr@fci.helwan.edu.eg

ABSTRACT

DevOps is a culture that aids in increasing the service delivery speed of an organization. It encourages the development and operations teams for working and coordinates them together to achieve faster and more automated activities. Although DevOps becomes an essential building block in every software life cycle, its processes suffer from data security attacks over time. Therefore, companies and organizations exert considerable effort including investigation time, buying licenses for third-party defending tools, hiring highly skilled security engineers, etc. to avoid these security attacks. Meanwhile, many studies suggest building DevOps frameworks that concentrate on enabling automated processes without giving much focus to the integrity and validation of the data, which might be lost or manipulated due to significant security attacks during the deployment process, the ignorance of handling security assaults costs a lot of damage to the organizations. In the current study, a DevOps framework DevHash is proposed to improve the previous DevOps frameworks by adding new phases detecting and validating common security attacks, such as data manipulation attacks that might occur during the deployment process. DevHash can detect data manipulation attacks much faster than the other previous DevOps frameworks as the proposed framework DevHash includes four main phases: "Coding", "Build Pipeline and Hashing", "Deployment Pipeline and Hashing" and "Health Check". These phases are important in enhancing and validating data security during the deployment process. To assess the performance of the proposed framework DevHash, a comparison has been made between the DevHash framework and a previous traditional DevOps framework; it is applied to six cases including a collection of software companies from various domains with different technologies. Finally, the results showed that the proposed framework DevHash consumes less time than the traditional DevOps framework in detecting major security attacks such as data manipulation attacks, and it helps in increasing the integrity and the accuracy of data as well as the overall reliability of the DevOps processes.

Keywords: *DevOps, Validation, Deployment Process, Security Attacks, Data Manipulation Attacks.*

1. INTRODUCTION

The term "DevOps" stands for "development and operations", and it refers to a method of close cooperation between software developers and operations [1]. The objective is to improve communication and integration between development and operations [2]; consequently, all benefits of modern software development methodologies using frequent releases of new software features to the end-users and then learning from them can be realized [3]. Although the DevOps concept first appeared in the context of rapid releases in 2009 [4], the vast majority of the addressed problems had previously been identified [5-7]. These previous studies identified the issue of poor collaboration and early involvement of the operations team in the software development process as having a detrimental impact on software release time and quality in the production environment [6,7]. Because most agile approaches do not address the system in use (operations) life-cycle phase, much of the new attention on DevOps occurs after the broad adoption of agile methods [8,9]. As a result, several notions such as agile infrastructure [10] and boundary spanning [11] have been used to fill the gap in the agile literature but not consistently. Instead of focusing solely on the operations unit, the agile literature on boundary spanning focuses on all units interacting with it and they are external to the development unit [11,12]. The problem was initially addressed by developing well-defined and formal methods for involving operations in the development process [5,11].

DevOps aims at improving the speed with which production updates are delivered and automating the software development process [13].

DevOps arose from the collaboration of development and operations team members to reduce the gap between developers and operators [14]; they not only accelerate new software releases but also reduce the taken time responding to client requests and changes. Development teams are tasked with introducing new features, whereas operations teams are concerned with maintaining stability [15].

Recently, DevOps frameworks are concentrating on automating and enhancing the software development and deployment processes without paying much attention to security attacks, including attacks that could harm and manipulate the data while transferring between different DevOps processes. As a result, data reliability and data integrity flowing inside the deployment pipeline will be at risk and the company's processes will be in jeopardy.

Because of this, companies try to ensure the existence of the appropriate measures to prevent data tampering and data loss, and provide a secure deployment pipeline to run its operations, so in the current study, a new framework "DevHash" is proposed; it can validate and detect one of the most common security attacks known as data manipulation attacks where "DevHash" can detect any manipulation that occurred during the transferring of the build files between the source and the target environments. As a result, the "DevHash" framework will improve the integrity and accuracy of data, as well as enhance the reliability of the CI/CD processes.

2. PREVIOUS WORK

Several studies have been done to enhance and handle the challenges affecting the DevOps process. In 2019, the authors suggested a DevOps approach for handling security measures along the DevOps pipeline. This method is introducing security into the DevOps process based on source code inspection during the integration phase. The outcomes of this strategy created for an actual scenario involving the health industry were promising [16].

In 2020, a paper presented some contributions for professionals and practitioners, including the primary DevOps capabilities, DevOps areas, and the relationships between them. They identified and detailed the main DevOps capabilities as continuous integration, continuous deployment, and continuous testing. The authors, on the other hand, elicited and described the following DevOps areas: culture, measurement, sharing, automation, technology, people, and process [15].

In 2020, another study covered the key components of DevOps in securing cloud solutions and provided examples of real-world applications from businesses like IBM. The importance of this research to the United States is also discussed, particularly for organizations looking to secure their serverless systems [17].

In 2021, a study started to look into the main problems users encounter when integrating security tools into a DevOps workflow to offer suggestions for fixing these problems. Consequently, they conducted a study involving 31 carefully selected webinars on integrating security tools in DevOps. They also used thematic analysis, a qualitative data analysis method, to define the issues and emerging solutions related to integrating security tools in rapid deployment environments [18].

To rationalize and systematize security and privacy analyses in multi-cloud, a study in 2022 introduced an innovative DevOps framework and methodology. This will enable an informed decision-making process for a risk-cost balanced selection of the system components requested from Cloud Service Providers. The primary focus of the current work is multi-cloud application development [19].

Another study in 2022, is presented to investigate the advantages of adopting both the DevOps and Agile methodologies simultaneously. By incorporating twelve case studies from global software engineering firms, a qualitative methodology is used. The advantages of adopting both approaches simultaneously are determined via thematic analysis. Twelve advantages are revealed by the data, with the automation of procedures, better team communication, and a decrease in time to market due to process integration and faster software delivery cycles being highlighted. The Agile and DevOps methodologies, while addressing various objectives and difficulties, can benefit organizations when properly merged and integrated. This study is innovative in that it systematizes the advantages of adopting Agile and DevOps together while considering various viewpoints of the business environment for software engineering. [20].

In earlier studies, the suggested DevOps frameworks concentrated on implementing the deployment pipeline and automating the processes of developing and deploying the software without paying much attention to security attacks or modifications that could harm the data while transferring it from one phase to another during various DevOps processes.

Lack of data integrity, besides the absence of proper techniques to avoid data manipulation and data loss, could have a significant effect on the organization's reputation, trust, and profits, which eventually will increase customer churn.

As a result, organizations must consider security attacks as a threat to their existence, and suitable approaches have to be taken to ensure safe and smooth operations. So in this research, "DevHash" added new validation phases to the recent DevOps methodologies, to face these threats and raise the security of the current deployment process.

3. BACKGROUND

The modern software development process is increasingly characterized by frequent and quick software modifications that enable end-user input [3,24]. A fundamental motivator for enterprises to adopt DevOps [4] and continuous procedures [3] is the need to be able to release software frequently, rapidly, and automatically as soon as changes are checked into the mainline. This section presents the DevOps concept, DevOps exercises and practices, CI, CD, and terminologies used in developing the proposed framework DevHash and they are as follows:

3.1 The DevOps Main Concept

The DevOps concept seeks to fill gaps driven by the organizational separation between software development, release, and operations processes [9]. Throughout this perspective, the development and operations teams must contend with numerous competing goals of 'agility vs. stability' and a slew of roadblocks such as insufficient information flow and test environments [6].

The scientific definition of DevOps is "a mindset that encourages cross-functional collaboration between teams within a software development organization, particularly development and IT operations, to reduce ambiguity, operate resilient systems, and accelerate change delivery [23]. Collaboration and communication between operations and development are the central ideas of DevOps [23].

3.2 DevOps Exercises Practices

The term "DevOps" refers to both technical and non-technical processes [22]. The categories distinguish between common practices carried out jointly by development and operations teams, as well as practices exclusive to development and operations. The category of common practices is further subdivided into collaborative practices involving human interactions and mostly automated procedural practices. The common practices of the procedural category receive more attention than their unique practices [24].

DevOps approaches are incorporated into the deployment pipeline because the focus is not only on the design and implementation of system features but also on the environments and tools that support the development, deployment, and operation of software features [13]. The deployment pipeline is an automated representation of the entire software process, including all stages of getting software updates from version control to end-user visibility [24]. DevOps methods include deployment pipeline automation, such as automatic environment provisioning, intending to avoid (or minimize) human system handovers from development to operations. It can also be used to keep developers away from the intricacies of deployment procedures, resulting in a shorter learning curve [21,25]. The automated deployment mechanism is built into the continuous integration (CI) server with pre-defined triggers to allow for the automatic deployment of changes to virtual machines (VM) in production or other cloud environments [13] in cloud-based systems, with which the concept of DevOps is frequently associated [26].

3.3 Continuous Integration (CI)

The term "Continuous Integration" (CI) refers to a software development practice in which team members integrate regularly their work, usually at least once per day, resulting in multiple integrations per day. An automated build validates each integration to uncover integration flaws as quickly as possible [30]. It seeks to continuously integrate source code into the main branch [29], enabling the developer to commit the code multiple times daily, followed by an automatic build and testing, and giving the developer prompt feedback whenever a defect is found. In case no bugs are discovered, the committed code is released to production. Continuous Delivery necessitates Continuous Integration [28].

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3.4 Continuous Delivery or Continuous Deployment (CD)

The term "Continuous Deployment" (CD) refers to the ability to release a feature as soon as it is ready, assuming a sufficient development and deployment infrastructure in place [14]. It is an operations practice in which release candidates reviewed in continuous delivery are placed in a production environment on a regular and timely basis, the nature of which may vary depending on the technical setting. This usually entails making it widely available to consumers, but not always, and in some cases, it is not even applicable as a concept [27]. The Continuous Deployment pipeline is a set of tools enabling a workflow to continuously transfer source code from a version control system to production via the build system after the code passes the testing phase, it will be ready to be deployed into the target environments [29]. After a successful continuous integration pass, continuous deployment normally puts the new build into production, but it can also include automated processes such as deploying virtual machines and installing and configuring operating systems, supporting software, and libraries [14].

3.5 Hashing

The term "Hashing" refers to the process of dividing a data file into small chunks and combining them to produce a numeric value that can be used to identify the original data file. In the broadest sense, hashing has many applications, including efficient data storage and retrieval, data integrity, and data matching [30].

3.6 Hashing Function

The hash function is one of the most widely used concepts in information security today. Hash functions are heavily used in modern cryptography; in the field of information security, the Hash function method is currently in widespread use. The hashing process is carried out by the Hash function [30,31].

3.7 Hashing Process

The hashing process in information sharing primarily verifies the accuracy and integrity of data (detection of changes). Calculating the hash code of the data you want to keep safe and secure, is the core idea; after that, you should compare it to the hash code stored in memory as a standard. The mismatch in Hash values compared here is unquestionably a breach of data integrity. To ensure the security of various types of data, all companies that provide online services on the network employ a wide range of hashing algorithms [30,31].

4. THE PROPOSED FRAMEWORK

4.1 Overview

The current paper introduces a new framework DevHash, that presents new steps to detect any data manipulation attack in an earlier phase, saves a lot of time consumed in detecting such attacks, and it also improves data integrity during the deployment process. The proposed framework DevHash and its integrated sequential phases are described in detail in the following section. As shown in **Figure. 1**, the framework is broken down into four major sequential phases: “Coding”, “Build Pipeline and Hashing”, “Deployment Pipeline and Hashing”, and “Health Check”.

Journal of Theoretical and Applied Information Technology
15th July 2023. Vol.101. No 13

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

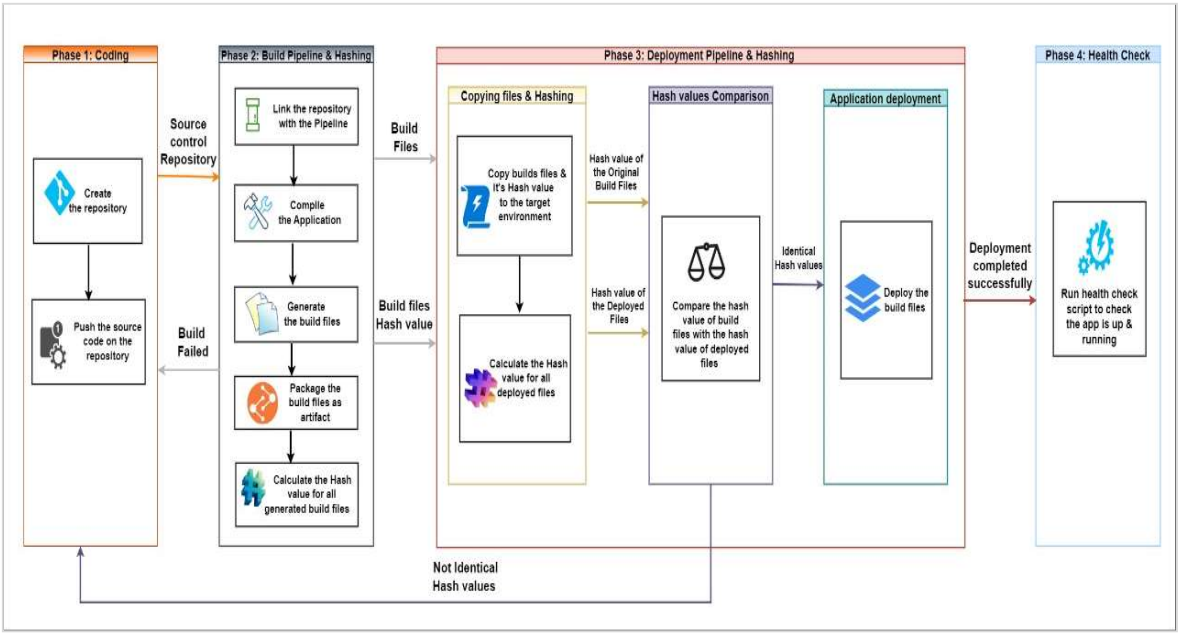

Figure 1: Proposed Framework "DevHash" Architecture

4.2 Phase 1: Coding

Companies primarily use GitHub and other continuous integration tools during this phase to store, track and collaborate on software projects. Developers can use it to upload their code files and work together on projects with other developers. In this phase, a GitHub repository has been created and the source code has been pushed into it.

4.3 Phase 2: Build Pipeline and Hashing

In this phase, "AWS" (Amazon Web Services), "Azure DevOps" or any platform can be used for pipelines creation; in the proposed framework DevHash, the build pipeline is linked with the GitHub repository through the repository URL to get the latest version of the source code. The configuration file of the build pipeline is created to build the source code, package the build files as artifacts, and then calculates the Hash value of the generated build files using the Hash Algorithm (SHA-1) [32] as well as storing it in a temporary variable named "sha1sum" as shown in **Routine 1**.

Routine 1 Build Pipeline and Hashing

```
version: 0.1
phases:
  build:
    commands:
      - echo Build started on `date`
      - mvn package
  post_build:
    commands:
      - echo Build completed on `date`
      - sha1sum target/SampleMavenTomcatApp.war | awk '{ print $1 }' > target/shaBuild
artifacts:
  files:
    - '**/*'
```

There are two paths for this step: the first one is when the build process is done successfully and the build files are generated, the Hash value is calculated for these files successfully, and then the process will proceed to the next phase, 'Deployment Pipeline and Hashing' in which the generated build files and the calculated Hash value will be considered the inputs for it.

The second path is that when the build process fails, the process will get the latest source code from the source control tool, recompile the application, create and package new build files again and proceed with the process steps.

4.4 Phase 3: Deployment Pipeline and Hashing

This phase consists of 3 main steps; they are "Coping Files and Hashing", "Hash Values Comparison" and "Application Deployment". All these steps are integrated to complete the deployment process and make a successful deployment for a new application release into the target environment and make sure that there are no

data attacks or differentiation between the original build files and the deployed files. The Deployment pipeline is created, and its configuration file includes all the commands needed to complete all the 3 main steps; it contains the target environment(s) details (for example, QA, Staging, or Production environment details), the files that should be moved to the target environment, and the script files that should be executed after the files' coping process is completed, as shown in **Routine 2**.

Routine 2 Deployment Pipeline and Hashing

```
1. version: 0.0
2. os: linux
3. files:
4.       - source: target/SampleMavenTomcatApp.war
5.         destination: /opt/
6.       - source: target/shaBuild
7.         destination: /opt/
8.       - source: Dockerfile
9.         destination: /opt/
10.      - source: Hello.war
11.        destination: /opt/
12. hooks:
13.    # ApplicationStop:
14.    #   - location: scripts/stop_application
15.    #     timeout: 300
16.    AfterInstall:
17.    - location: scripts/compare.sh
18.      timeout: 300
19.    ApplicationStart:
20.     - location: scripts/start_application
21.       timeout: 300
22.    ValidateService:
23.     - location: scripts/basic_health_check.sh
```

After completion of the copying files process in the targeted environment, the Hash value will be calculated for these files using the (SHA-1) algorithm [32], and then a process will start to compare the Hash value of generated build files in **Phase 2** with the calculated one.

Provided the two Hash values are not identical, this case means data manipulation has occurred and the deployed files differ from the original build files. So, the deployment process will be stopped and the whole process will retrigger once more starting from getting the latest version of the source code from the used source control tool, as shown in **Routine 3**.

Routine 3 Comparison Script

```
1. sha1sum  /opt/SampleMavenTomcatApp.war | awk '{ print $1 }' > /opt/shaDeploy
2. build_sha=$(cat /opt/shaBuild)
3. deploy_sha=$(cat /opt/shaDeploy)
4. if [ $build_sha = $deploy_sha ]
5. then
6.     echo Deployment successed, Go for testing
7. else
8.     echo Deployment failed, trigger Pipeline again
9.     echo ShaValues are not identical
10.    apt install awscli
11.    aws codepipeline start-pipeline-execution --name  SHA-Pipline --region us-west-2
12.    exit 1
13. fi
```

Now that the two values are identical, this means there is no data manipulation occurring on the build files and the deployed files are the same as they generated from the build pipeline. Thus, the process will proceed to the next step, "Application Deployment" and the completion of the deployment process will be done in the targeted environment, as shown in **Routine 4**.

Routine 4 Application Starter

```
1. #!/bin/bash
2. set -e
3. # CATALINA_HOME='/usr/share/tomcat/'
4. # cp/opt/SampleMavenTomcatApp.war $CATALINA_HOME/webapps/
5. # chown tomcat:tomcat-R /usr/share/tomcat/
6. # $CATALINA_HOME/bin/catalina.sh start
7. docker kill $(docker ps -q)
8. cd /opt/
9. docker build -t hello
10.  docker run -itd -p 80:8080 hello
```

4.5 Phase 4: Health Check

In this phase, a health check script is created to check that the deployed application is up and running on the target environment and working properly, as shown in **Routine 5**.

Routine 5 Health Check

```
1. #!/bin/bash
2. for i in `seq 1 10`;
3. do
4.  HTTP_CODE=`curl --write-out '%{http_code}' -o /dev/null -m 10 -q -s http://localhost:80/Hello/`
5.  if [ "$HTTP_CODE" == "200" ]; then
6.    echo "Successfully pulled root page."
7.    rm -rf /opt/SampleMavenTomcatApp.war
8.     rm -rf /opt/shaBuild
9.     rm -rf /opt/shaDeploy
10.    rm -rf /opt/Dockerfile
11.    rm -rf /opt/Hello.war
12.    exit 0;
13.  fi
14.  echo "Attempt to curl endpoint returned HTTP Code $HTTP_CODE. Backing off and retrying."
15.  sleep 10
16. done
17. echo "Server did not come up after the expected time. Failing."
18. exit 1
```

4.6 The Proposed Execution Scenario is as Below:

Step 1: In the beginning, after the execution of the build pipeline, a number of build files will be generated, and then the Hash value will be calculated for the generated build files.

Step 2: After that, during the execution of the deployment pipeline, the build files and their Hash value will be transferred to the target environment(s).

Step 3: In the target environment(s), the comparison script will be executed. It will calculate the Hash value for the deployed files, and then compare it with the Hash value calculated before in **step 1**.

Step 4: In case the two Hash values are identical, the application starter will be executed to start up the application on the target environment. After that, the health check script will be executed to check that the application is up and running. However, if the two Hash values are not identical, the whole process will be restarted.

5. RESULTS

The proposed framework was evaluated using six cases including a group of software companies in different domains using different technologies to assess the proposed framework DevHash efficiency. In each case, the development team of each project in **Table 1** made some edits in the configurations of the pipelines to apply the proposed routines to work with its project.

The results have been collected and analyzed after applying the recent DevOps framework presented in [20] and comparing it with the proposed framework "DevHash", using the six cases mentioned in **Table 1**.

Table 1: Companies and Projects Description.

Aspect	Case A	Case B	Case C	Case D	Case E	Case F
Project Size	Small (5 modules)	Small (7 modules)	Medium (10 modules)	Medium (14 modules)	Large (21 modules)	Large (25 modules)
System	Web-based-service	Rest APIU for media content	Web-based-service	Web-based-service	Web-based-service	Web-based-service
Methodology	Agile	Agile	Agile	Agile	Agile	Agile
Release Cycle	Fixed schedule project of 6 months 2-week sprint	Serval time to production 2-week sprint	Serval time to production 4-week sprint	Serval time to production 4-week sprint	Serval time to production 4-week sprint	Serval time to production 4-week sprint

Journal of Theoretical and Applied Information Technology
15th July 2023. Vol.101. No 13

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Team size	3 full stack developers, 1 project manager	3 full stack developers, 1 project manager, 1 UX designer	5 full stack developers, 1 project manager, 1 UX designer, 1 product owner	4 backend developers, 1 UX designer, 2 frontend developers, 1 project manager	6 backend developers, 2 UX designers, 3 frontend developers, 1 project manager, 1 product owner	10 full stack developers, 2 UX designers, 1 project manager, 1 product owner
Build output	JAR	JAR	WAR	WAR	WAR	AWS Machine Image
Tools	GitHub, Java, Selenium, Jenkins	GitHub, Java, JavaScript Selenium, Jenkins	GitHub, Java, Docker, JavaScript Selenium, Jenkins	GitHub, Java, Selenium, Angular, Jenkins	GitHub, Java, Selenium, React, Jenkins	Bitbucket, Jenkins, Python, AWS, Java, CloudFormation, Vue Docker
Languages	java	Java, JavaScript, Nodejs.js	Java, JavaScript, Nodejs.js	Java, angular	Java, React	Python, java, Vue

In the current study, a data manipulation attack [33] has been selected to test the efficiency of the proposed framework in all cases described in **Table 1**. To simulate the data manipulation attack, a tool commonly used by attackers for data manipulation purposes has been used.

In the beginning, when applying the recent DevOps framework presented in [20], various security attacks such as the data manipulation attack have been detected after the project is deployed. It has consumed a lot of time until the attack is detected, where the consumed time includes the deployment time and the rollback time, as shown in **Table 2**.

Table 2: Consumed time for detecting the attack before applying "DevHash".

Cases	Deployment Time (Minutes)	Rollback Time (Minutes)	Total Consumed Time (Minutes)
Case A	3.1	3.2	6.3
Case B	4.7	4.72	9.42
Case C	6.0	6.2	12.2
Case D	6.9	7.0	13.9
Case E	8.3	8.42	16.72
Case F	9.52	9.50	19.02

And the representation of the total consumed time for detecting the attack before applying "DevHash" is shown in **Figure. 2**.

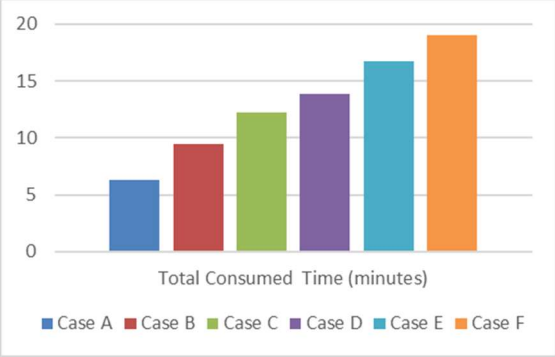

Figure 2: Total Consumed Time Before Applying "DevHash"

On the other hand, after applying the proposed framework DevHash, the security attacks have been detected even before the project is deployed. It saves a lot of time in comparison to the recent DevOps framework presented in [20], as shown in **Table 3**. DevHash will discover the security attacks, in the 'Comparison Script'step, as shown in **Routine 3**.

Table 3: Consumed time for detecting the attack after applying "DevHash"

Cases	Consumed Time (Minutes)
Case A	1.2
Case B	1.32
Case C	1.8

Case D	1.9
Case E	2.6
Case F	3.0

The consumed time for detecting the attack after applying "DevHash" is presented in **Figure. 3**.

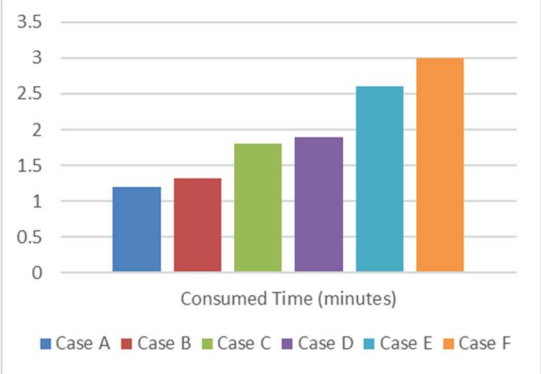

Figure 3: Consumed Time After Applying "DevHash"

A comparison between the consumed time for detecting the data manipulation attack before and after applying the proposed framework DevHash in each case is summarized in **Table 4**.

Table 4: Consumed time for detecting the attack before and after applying "DevHash"

Cases	Total Consumed Time before applying "DevHash" (Minutes)	Total Consumed Time After applying "DevHash" (Minutes)
Case A	6.3	1.2
Case B	9.42	1.32
Case C	12.2	1.8
Case D	13.9	1.9
Case E	16.72	2.6
Case F	19.02	3.0

The comparison between the consumed time for detecting the attack before and after applying the proposed framework "DevHash" is presented in **Figure. 4**.

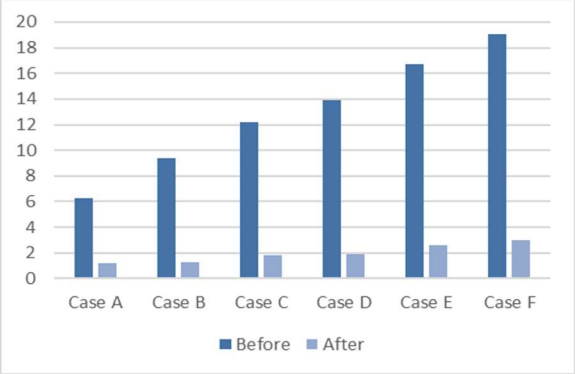

Figure 4: comparison between the consumed time for detecting the attack before and after applying "DevHash"

6. CONCLUSION AND FUTURE WORK

DevHash framework is proposed to enhance traditional DevOps frameworks when dealing with data security attacks; the implementation of the new phases can validate and detect one of the most common data security attacks known as data manipulation attacks. In the current study, a comparison has been made between the DevHash framework and a previous traditional DevOps framework to evaluate the performance of each framework when using a simulated data manipulation attack. This comparison is applied to six cases which include a set of software companies in different domains with different technologies.

Finally, the results showed that the proposed framework DevHash consumes less time than the traditional DevOps framework in detecting major security attacks, such as data manipulation attacks, which helps in increasing the integrity and the accuracy of data, as well as the overall reliability of the DevOps process. DevHash has shown promising results regarding data security attacks compared with the previous DevOps frameworks.

Although, the "DevHash" framework showed good results as mentioned before. But it still can be extended, enhanced, and installed in many other potential areas in the future, especially projects that include sensitive data and require high levels of security such as banking systems, stock markets, medical research, and cryptocurrency organizations.

In addition, the "DevHash" framework has been used already in a number of case studies as described above in the research, but applying the "DevHash" framework in large-scale projects becomes an interesting challenge that needs to be considered.

However, applying the validation steps to the production environment, was the main focus of this research. But the validation steps can also be

added to secure the deployment process to other environments, such as staging, and QA environments, which present an interesting point that needs to be noted and applied. Afterward, results need to be collected and analyzed to evaluate the performance of the "DevHash" framework in different environments.

REFERENCES:

[1] C. Amrit, M. Daneva, A mapping study on cooperation between information system development and operations, in: International Conference on Product-Focused Software Process Improvement, Springer, 2014, pp. 277–280, doi:10.1007/978-3-319-13835-0_21.

[2] J. Humble, D. Farley, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, First ed., Addison-Wesley Professional, Boston, 2010.

[3] A. Haghighatkhah, L.E. Lwakatare, S. Teppola, T. Suomalainen, J. Eskeli, T. Karvonen, P. Kuvaja, J.M. Verner, M. Oivo, Continuous deployment of software intensive products and services: a systematic mapping study, J. Syst. Softw. 123 (2017) 265–291.

[4] J. Humble, J. Molesky, Why enterprises must adopt DevOps to enable continuous delivery, Cutt. IT J. 24 (8) (2011) 6–12.

[5] B. Tessem, J. Iden, Cooperation between developers and operations in software engineering projects, in: Proceedings of the International Workshop on Cooperative and Human Aspects of Software Engineering,ACM,2008,pp. 105–108, doi:10.1145/1370114.1370141.

[6] J.Iden, B.Tessem, T. Päivärinta, Problems in the interplay of development and IT operations in system development projects: a Delphi study of Norwegian IT experts, Inf. Softw. Technol. 53 (4) (2011) 394–406.

[7] J. Hamilton, on designing and deploying internet-scale services, in:Proceedings of the 21st Large Installation System Administration Conference, USENIX Association, Dallas, Texas, 2007, pp. 231–242.

[8] P. Abrahamsson, J. Warsta, M. Siponen, J. Ronkainen, New directions on agile methods: a comparative analysis, in: 25th International Conference on Software Engineering, 2003. Proceedings., IEEE, 2003, pp. 244–254, doi:10.1109/ICSE.2003.1201204.

[9] O. Gotel, D. Leip, Agilesoftware development meets corporate deployment procedures: stretching the agile envelope, in: Agile Processes in Software Engineering and Extreme Programming, Springer, Berlin, Heidelberg, 2007, pp. 24–27.

[10] P. Debois, Agileinfrastructure and operations: how infragile are you? in: Agile 2008 Conference, IEEE, 2008, pp. 202–207, doi:10.1109/Agile.2008.42.

[11] D.E. Strode, S.L. Huff, B. Hope, S. Link, Coordination in co-located agile software development projects, J. Syst. Softw. 85 (6) (2012) 1222–1238.

[12] G. van Waardenburg, H. van Vliet, When agile meets the enterprise, Inf. Softw. Technol. 55 (12) (2013) 2154–2171.

[13] L. Bass, I. Weber, L. Zhu, DevOps: A Software Architect's Perspective, Addison-Wesley Professional, 2015.

[14] TeemuLaukkarinen, Kati Kuusinen and TommiMikkonen (2017 IEEE/ACM), DevOps in Regulated Software Development: Case Medical Devices, 978-1-5386-2675-7/17 $31.00 © 2017 IEEE, DOI 10.1109/ICSE-NIER.2017.20

[15] D. Teixeira, R. Pereira, T. A. Henriques, M. Silva and J. Faustino, "A Systematic Literature Review on DevOps Capabilities and Areas", International Journal of Human Capital and Information TechnologyProfessionals, Volume 11, Issue 3, July-September 2020, PP. 1-22.

[16] X. Larrucea, A. Berreteaga, and I. Santamaria, "Dealing with Security in a Real DevOps Environment", EuroSPI 2019: Systems, Software and Services Process Improvement, Springer, vol: 1060, pp: 453–464.

[17] Yarlagadda, R. Teja, "DevOps for Better Software Security in the Cloud", International Journal of Emerging Technologies and Innovative Research (www.jetir.org), ISSN:2349-5162, Vol.7, Issue 9, page no.1081-1085, September 2020, Available at SSRN: https://ssrn.com/abstract=3807615.

[18] R. N. Rajapakse, M. Zahedi, M. A. Babar, "An Empirical Analysis of Practitioners' Perspectives on Security Tool Integration into DevOps", Proceedings of the 15th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), October 2021, Article No: 6, pp: 1–12, https://doi.org/10.1145/3475716.3475776.

[19] S. F. Ahamed, M. Dhar M S, S. K. Kishore, M. P. Borawake, T. D. R and M. Thenmozhi, "DevOps Security and Privacy in the Development of Multicloud Applications," 2022 International Conference on Electronics and

Renewable Systems (ICEARS), Tuticorin, India, 2022, pp. 1631-1635, doi: 10.1109/ICEARS53579.2022.9752387.T. Okubo, H. Kaiya, "Efficient secure DevOps using process mining and Attack Defense Trees", Procedia Computer Science, Elsevier, 2022, vol: 207, pp: 446-455.

[20] Almeida, F.; Simões, J.; Lopes, S. Exploring the Benefits of Combining DevOps and Agile. Future Internet 2022, 14, 63. https://doi.org/10.3390/fi14020063

[21] L.E. Lwakatare, P. Kuvaja, M. Oivo, an exploratory study of DevOps: extending the dimensions of DevOps with practices, in: The Eleventh International Conference on Software Engineering Advances, IARIA, Rome, 2016, pp. 91–99.

[22] R. Penners, A. Dyck, Release engineering vs. DevOps-an approach to define both terms, Full-Scale Softw. Eng. (2015). https://www2.swc.rwth-aachen.de/docs/teaching/seminar2015/FsSE2015papers.pdf#page← 53.

[23] L. Bass, R. Jeffery, H. Wada, I. Weber, L. Zhu, Eliciting operations requirements for applications, in 1st International Workshop on Release Engineering, IEEE, 2013, pp. 5–8.

[24] Lucy Ellen Lwakatare, TerhiKilamo, TeemuKarvonen, DevOps in practice: A multiple case study of five companies, Information and Software Technology 114 (2019) 217–230.

[25] J. Cito, P. Leitner, T. Fritz, H.C. Gall, The making of cloud applications: an empirical study on software development for the cloud, in: 10th Joint Meeting on Foundations of Software Engineering, ACM Press, 2015, pp. 393–403.

[26] D. Stahl, TorvaldMartensson and Jan Bosch, Continuous Prac- tices and DevOps: Beyond the Buzz, What Does It All Mean? 978-1- 5386-2141-7/17 $31.00, IEEE DOI 10.1109/SEAA.2017.78,2017.

[27] Aayush Agarwal, SubHashGupta, Tanupriya Choudhury (ICACCE- 2018), Continuous and Integrated Software Development using DevOps, 978-1-5386-4485-0/18/$31.00 ©2018 IEEE.

[28] Rickard Bremer and Johan Eriksson, Understandings and Implemen- tations of Continuous Delivery, June 2015.

[29] Kim Rejstrom, Implementing Continuous Integration in a SmallCompany, Aalto University School of Electrical Engineering, 2016.

[30] H. Farid,An Overview of Perceptual Hashing, in: Journal of Online Trust and Safety, October 2021, pp.1-22,doi:10.54501/jots.v1i1.24.

[31] N. Quliyev,Z.Shamilov, S.Akbarova,The Role of Hashing Algorithms in File Security,in:Scientific Community: Interdisciplinary Researc,2021, pp.518-524.

[32] D. Eastlake and P. Jones, "U.S. secure hash algorithm 1 (SHA1)", The Internet Society, 2001, pp:1-22.

[33] Chatterjee, R. (2021). Security in DevOps and Automation. In: Red Hat and IT Security. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-6434-8_3.