# **Jenkins Automation Security**

Researched by: Karl Ivan Aquino

Contents

[**Jenkins Automation Security** 1](#_Toc120346315)

[**Introduction** 3](#_Toc120346316)

[**How does Jenkins automation work?** 3](#_Toc120346317)

[**Advantages and Disadvantages of Jenkins automation** 4](#_Toc120346318)

[**Jenkins Plugins** 5](#_Toc120346319)

[**Basic Jenkins security management** 6](#_Toc120346320)

[**Jenkins Security CVE** 7](#_Toc120346321)

[**Jenkins Security recommendations** 7](#_Toc120346322)

[**Conclusion** 9](#_Toc120346323)

[**Reference list** 9](#_Toc120346324)

## **Introduction**

Jenkins is a free and open-source automation tool that has built-in plugins for use with continuous integration. Your project is built, compiled, and tested using it, making it simple for developers to make modifications to the project. A tool for automation, Jenkins is an expansive server that enables any developers to create, test, and release software. It utilises Java because it was created in Java. Jenkins enables end-to-end automation of projects (jobs) or other types of initiatives. No matter what platform you are using, Jenkins is a potent tool that enables continuous integration and delivery of projects. Any build or continuous integration can be handled by this free source. A variety of testing and deployment solutions are compatible with Jenkins. Your software projects will be regularly built and tested using Jenkins.

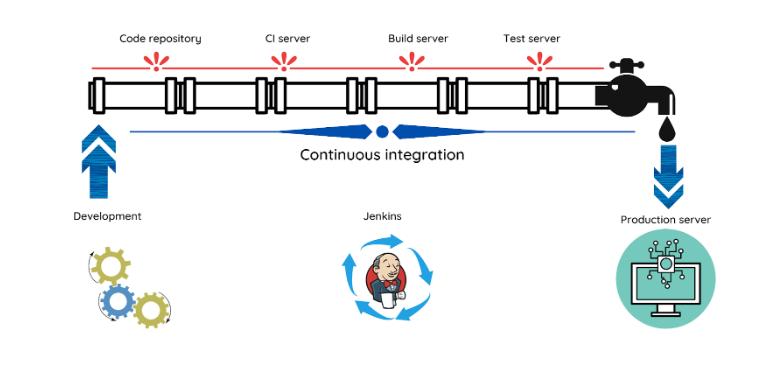
## **How does Jenkins automation work?**

Jenkins is software that enables continuous integration. Developers are required to recurrently incorporate new code into a shared repository in accordance with the continuous integration development methodology. Because of the increase in release frequency, manual deployment of all these releases is now extremely time-consuming, underscoring the urgent need for automated deployment. This concept was created to address the problem of identifying errors later in the build process. Deploying an application or component to test servers and then to end users enables it to be available for testing, making it a crucial phase in the software development life cycle (by deploying on the production server).

Diagram

Description automatically generated

Automation cycle overview [3]



Jenkins architecture [2]

Jenkins can automate these routine deployments using predetermined triggers and conditions. We can fully or partially ship the code across multiple development stages, from fundamental development through production, using a method called automatic deployment. When our deployments are automated, they are more dependable and effective. This functionality is automated to the greatest extent possible to prevent flawed functionality during the transition of code from development to production. The quantity of testing conducted prior to release has an impact on the dependability and stability of an application. Automated testing and cloud-native technologies have made it possible for testing to grow faster and more thorough over the past few decades.

## **Advantages and Disadvantages of Jenkins automation**

ADVANTAGES:

* Constant delivery is made simpler with Jenkins because updates to software are automatically tested and released as soon as they meet all requirements. This lessens the need for human interaction and the likelihood of human error.
* Focused on developers. Because Jenkins was made by and for developers, it provides the capabilities that developers would require and would expect to work. Developers utilising Jenkins can thus concentrate on actualizing their innovations rather than laborious, repeated testing procedures.
* Reports. Users can get a detailed understanding of the automated tests and how successfully they were carried out through the display of test results summary and trends. The patterns of the automated tests, including prior versions' failure points, the duration of various tests, etc., are visible to testers. Such knowledge greatly aids users in enhancing the pipeline
* Jenkins does not require developers' access to be restricted for test automation purposes, therefore testing is not intrusive. Jenkins will handle the rest while QA teams can specify what they want to automate.
* Small modifications are simple to handle; the smaller the modification, the less eager a developer is to wait for a build. Tests required for a minor change can be done automatically in a matter of seconds.
* Jenkins' scheduling functionality enables testers to schedule tests to run at predetermined times. This is a quicker and more practical method.

DISADVANTAGES:

* Jenkins is free software that receives a lot of contributions. Because of the various plugins that have been created for a single tool, consumers may find their options confusing.
* Plugins have less versatility because they can't be customised. The plugins offered in Jenkins do not allow much modification or customization to make them more unique because most of them do not have enough documentation to assist the users.
* Jenkins has a challenging configuration process that requires significant learning. Jenkins has a lot of features and options that must be examined, so getting a handle on it takes some time.
* A cloud-based service is not itself hosted by Jenkins. Jenkins, which is offered as a service by cloud service providers like Azure, Cloudbees, etc., must be used by users. As well Jenkins's Docker integration needs to be improved
* Rules and options for authentication and permission are lacking

## **Jenkins Plugins**

Jenkins plugin is a package in java-archive format that adheres to a predetermined structure while being developed. Each plugin includes all the necessary data, including files, graphics, code, and other details. Jenkins essentially describes the collection of interfaces that developers in the Jenkins community implement and expand with original code. Therefore, plugins are created by community developers in accordance with the functional requirements, and we can download those plugins to add that feature to Jenkins.

Here are some commonly used Jenkins plugins:[5] <https://spectralops.io/blog/top-25-jenkins-plugins-for-2021/>

[](https://www.youtube.com/embed/9R6FZQu_jGc?feature=oembed)

A demonstration of Jenkin Plugin for scanning vulnerabilities in the project [9]

## **Basic Jenkins security management**

From workstations on corporate intranets to powerful servers connected to the public internet, Jenkins is utilised everywhere. Jenkins provides a variety of configuration options for enabling, changing, or disabling various security elements to safely serve this vast range of security and threat profiles.  Since Jenkins 2.0, numerous security features have been enabled by default to keep Jenkins environments secure unless a system administrator specifically turns off safeguards. When passing the interactive setup wizard, several of the security parameters are turned on by default to make sure Jenkins is secure. Others depend on specialised use cases enabled by unique Jenkins instances and include environment-specific installation and exchange.

Jenkins has two sections for access control:

Users **authenticate** themselves by demonstrating their identity utilising a security realm. User identification and group memberships are governed by the security domain.

An a**uthorisation** approach allows users to be authorised (permitted to perform something). This regulates whether a user has a permission (directly or via group memberships).

Jenkins Security, Authentication, Authorization, and Enabling Project Security Matrix is explained in this video tutorial:

[](https://www.youtube.com/embed/cyV6eIaj28s?feature=oembed)

Jenkins Security, Authentication, Authorization, and Enabling Project Security Matrix video tutorial [6]

## **Jenkins Security CVE**

These are the following latest Jenkins CVE including plugins (up to date):

Graphical user interface

Description automatically generated with medium confidence

Jenkins CVE list [7]

To discover more in details [7]: <https://www.jenkins.io/security/advisory/2022-06-22/>

## **Jenkins Security recommendations**

* Update Outdated and Vulnerable Core Plugins. Jenkins may be divided into two parts for the purpose of looking for known vulnerabilities: the basic automation platform of Jenkins and the plugins that sit on top of it. Maintaining a vulnerable version is particularly risky since malevolent users could employ publicly accessible exploits to take advantage of your server. While updating the core version requires manual labor, updating plugins only requires a few mouse clicks on the Jenkins UI. Most publicly disclosed vulnerabilities are plugin-related, so removing them will fix many security issues.
* Authentication and authorization. Jenkins offers numerous built-in authentication options, including "Jenkins' user database" and "Delegate to servlet container" (together referred to as "Security Realms" in Jenkins). Jenkins' built-in authentication mechanisms should not be used; instead, users should authenticate against a centralised third-party provider, such as GitLab, Github, LDAP, SAML, or Google. These techniques enable restrictions, such as password complexity, to be applied to the passwords, preventing unauthorised users from accessing the server. Jenkins has the built-in permission options "Legacy mode," "Anyone can do everything," and "Logged-in users can do anything." Specialists advise against using these built-in techniques in favour of plugins for more sophisticated permission procedures. The two most well-known plugins for this purpose are Matrix Authorization Strategy and Role-based Authorization Strategy. These plugins offer significant flexibility for implementing PoLP (Principle of least privilege) by defining the privileges of anonymous users, authenticated users, or specified ones. Additionally, you can assign roles that you create for each user or set privileges for each project. It is also feasible to use GitHub-based and GitLab-based authentication (using previously mentioned plugins).
* Limiting Agent Privileges. Build pipelines will by default be operating under the SYSTEM internal user's permissions. As a result, builds can now start and stop other builds, create, and delete jobs, run code on any node, and do other operations. If, for instance, Jenkins draws malicious build pipelines from the SCM platform, which the Jenkins administrator doesn't monitor, running builds with such rights can lead to major security risks. You can specify which user will perform the build and, consequently, what permissions, by using the Authorize Project plugin. Setting the least privilege possible for each project would be the general rule of thumb

Graphical user interface, text, application, email

Description automatically generated

Example of authorizing user permissions [8]

* The use of containerized agents. The agents can be run using a variety of techniques and instructions, including Kubernetes clusters, virtual machines, containers, and physical machines. From a security standpoint, we aim to reduce the influence that a hacked build has on other builds or the system. As a result, we favor starting from scratch with each build environment. Creating a container image with all necessary dependencies and delivering the task to a distant docker service could do this. By doing that, you can be guaranteed that every build will run on a fresh, independent container.
* Security credentials. The Jenkins core application does not offer practical ways to limit the exposure of credentials for users and builds, but several well-liked plugins do so admirably. Every new credential that is introduced to Jenkins has two options: "Global," which makes it available for Jenkins, nodes, items, all children's items, basically everything, or "System," which limits access to Jenkins and nodes only. To manage and access credentials, you can also build "Domains" for them.

## **Conclusion**

An open-source automation tool called Jenkins is free to use and comes with built-in plugins for usage with continuous integration. Because it is used to build, compile, and test your product, developers can easily make changes to it. Jenkins is a server that allows any developer to build, test, and release software. It is a tool for automation. Software called Jenkins allows for continuous integration, so In order to solve the issue of discovering mistakes later in the construction process, automated deployment was developed. One of the most important stages in the software development life cycle is the deployment of an application or component to test servers and later to end users, which makes it possible for testing to take place. Jenkins plugin is a Java-archive package that while being produced follows a specified structure. Each plugin contains all the required information, such as files, graphics, code, and other specifics. Jenkins still has shortcomings such the lack of Rules and alternatives for authentication and authorisation, even with the highlighted capabilities. As a result, by putting security standards and regulations in place, potential restrictions could be lessened, and the application's effectiveness increased.

## **Reference list**

[1] H. Sheth, “What is Jenkins? how & why to use it?,” LambdaTest, 02-Sep-2022. [Online]. Available: https://www.lambdatest.com/blog/what-is-jenkins/. [Accessed: 23-Nov-2022].

[2] S. Ndungu, “Jenkins automation for high-quality builds: Blazemeter by perforce,” Blazemeter, 07-Feb-2022. [Online]. Available: https://www.blazemeter.com/blog/jenkins-automation. [Accessed: 23-Nov-2022].

[3] B. Shrikanth, “Jenkins for test automation : Tutorial,” BrowserStack, 07-Sep-2022. [Online]. Available: https://www.browserstack.com/guide/jenkins-for-test-automation. [Accessed: 23-Nov-2022].

[4] Jagrat, “Jenkins manage plugins - how to manage, update & uninstall,” TOOLSQA, 11-Sep-2021. [Online]. Available: https://toolsqa.com/jenkins/jenkins-manage-plugins/. [Accessed: 23-Nov-2022].

[5] E. Katz, “Top 25 jenkins plugins for 2021,” Spectral, 22-Dec-2020. [Online]. Available: https://spectralops.io/blog/top-25-jenkins-plugins-for-2021/. [Accessed: 23-Nov-2022].

[6] “Jenkins security: Enabling security & project security matrix,” Software Testing Help, 25-Oct-2022. [Online]. Available: https://www.softwaretestinghelp.com/jenkins-security-tutorial/. [Accessed: 23-Nov-2022].

[7] “Jenkins Security Advisory 2022-06-22,” Jenkins security advisory 2022-06-22. [Online]. Available: https://www.jenkins.io/security/advisory/2022-06-22/. [Accessed: 23-Nov-2022].

[8] A. Ilgayev, “Jenkins security best practices,” Cycode, 24-Aug-2022. [Online]. Available: https://cycode.com/blog/jenkins-security-best-practices/. [Accessed: 23-Nov-2022].

[9] Jenkins Plugin for Fortify SCA (v 19.2). YouTube, 2019. [Online video]. Available: https://www.youtube.com/watch?v=9R6FZQu\_jGc. [Accessed: 23-Nov-2022].