Jenkins General Hardening Steps

Summary: Jenkins is a huge asset in a company based on its sheer power for a CI/CD pipeline. Like any software, there are going to be inherent problems with the implementation. There are certain built in functions and concepts that need to be secured to reduce the risk of a catastrophic breach.

General Communication Pattern

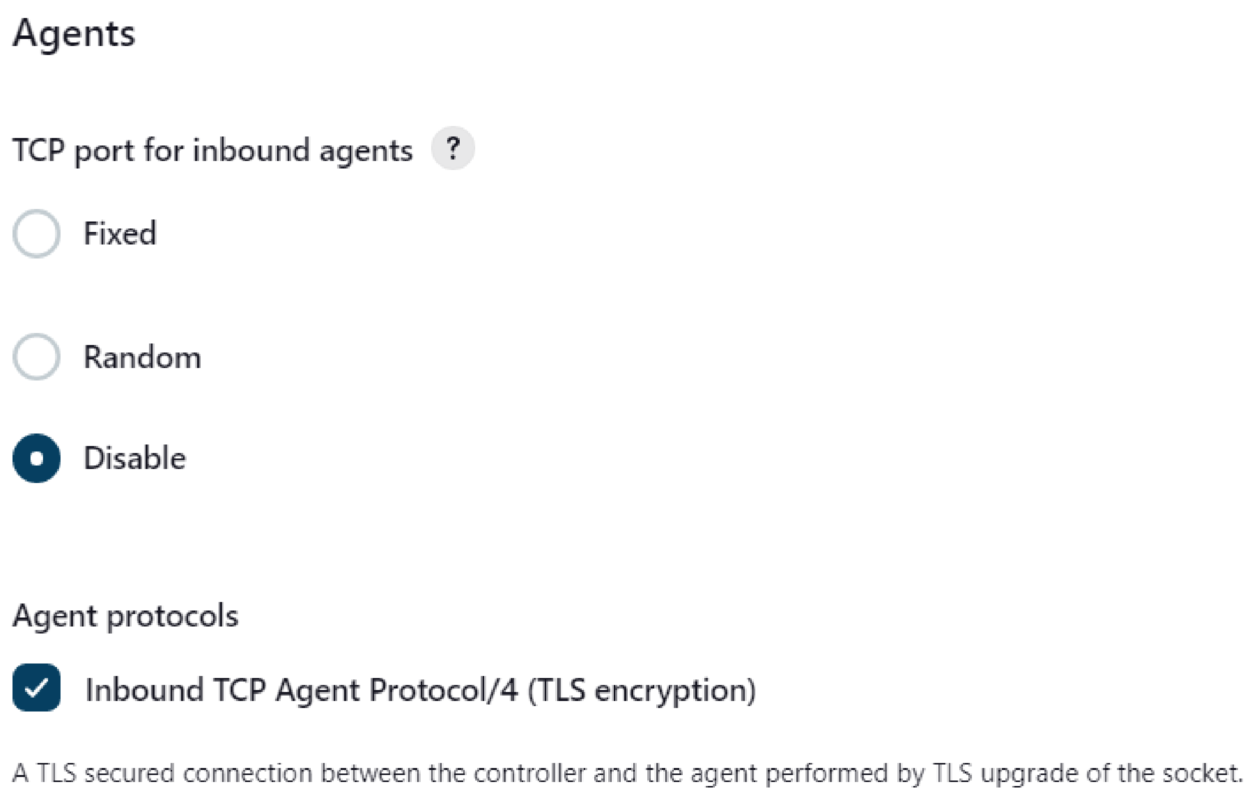
The normal expected behavior of the Jenkins server to agent communication:

1. The server will get into the agent host and will start the jenkins agent.
2. The server will send jobs to the agent via an SSH connection (in our case).

Notice how all these communications are initiated *from* the server and not the agent.

If it were the opposite direction, then this is an *inbound* connection.

And so, we can remove the tcp port for inbound agents.



Agents

TCP port for inbound agents

Fixed [✓ Disable] Random

Agent protocols

✓ Inbound TCP Agent Protocol/4 (TLS encryption)

A TLS secured connection between the controller and the agent performed by TLS upgrade of the socket.

However, because its functions are primarily based on plugins, there is a high chance that these community-based plugins are not thoroughly tested and created with security in mind. And so, there needs to be careful consideration when adding them.

You want to take a minimalistic approach when installing these plugins because of this.

A basic installation that I tested to carry out a nextcloud deployment had these plugins:

1. Build Timeout
2. GitHub Branch Source Plugin
3. LDAP Plugin
4. Matrix Authorization Strategy Plan
5. PAM Authentication plugin
6. Pipeline
7. Pipeline: Stage View Plugin
8. SSH Build Agents Plugin
9. Timestamper
10. Workspace Cleanup Plugin

There are many other default plugins that are installed. So we need to secure those instead as well.

Limit Agent Capabilities

Depending on your use case for Jenkins, you should restrict the user account that Jenkins will use when inside of the agent. A guide is given here