CCDC Playbook

1. Identify the Issue

- **Gather Information:**
  - Check logs in `/var/log/` for unusual activity (e.g., `auth.log`, `syslog`, `messages`, `secure`).
  - Identify running processes using `ps aux` or `top` for unexpected activity.
  - Verify active network connections using `netstat -tulnp` or `ss -tulnp`.

  Inspect failed login attempts or account activity:
  ```
  grep "Failed password" /var/log/auth.log
  ```
  Check for suspicious cron jobs:
  ```
  crontab -l; ls -la /etc/cron* /var/spool/cron
  ```
- **Verify Scope:**
  - Determine affected users, services, or files.
  - Validate integrity of critical files using checksums (e.g., `sha256sum` or `md5sum`).

---

2. Mitigate the Risk

- **Contain the Issue:**

  Disable suspicious accounts:
  ```
  usermod -L <username>
  ```
  Stop and disable malicious processes or services:
  ```
  kill -9 <pid>
  systemctl stop <service>
  systemctl disable <service>
  ```
  -

  Block network access for affected systems or connections:
  ```
  iptables -A INPUT -s <suspicious_IP> -j DROP
  ```
  Change permissions to prevent further access:
  ```
  chmod 000 <compromised_file>
  ```
- **Notify Stakeholders:**
  - Inform the incident response team or team lead.
  - Escalate to compliance or legal teams if data breaches are suspected.

---

3. Recover and Restore Service

- **Account Lockout or Compromise:**

  Reset the password for locked or compromised accounts securely:
  ```
  passwd <username>
  ```

  Force logouts for compromised accounts:
  ```
  pkill -u <username>
  ```
- **System or Service Restoration:**
  - Check for system integrity using file verification tools like `Tripwire` or `AIDE` if implemented.

Restore from backups if system integrity is in question:
```
rsync -av /path/to/backup /destination
```

Reinstall affected software packages:
```
apt-get --reinstall install <package>
```

- **Validate Restoration:**
  - Test restored services for functionality and ensure there is no remaining compromise.

---

4. Post-Incident Review and Logging

- **Document Actions Taken:**
  - Record details of the incident, including commands used and findings.

  Save relevant logs:
  ```
  cp /var/log/auth.log /incident_logs/auth.log
  ```
  - Capture a timeline of events using log timestamps.

- **Conduct a Root Cause Analysis:**
  - Review compromised files or binaries using tools like `file` or `strings`.

  Check for unauthorized configuration changes (e.g., SSH keys, `sudoers` modifications):
  ```
  cat /etc/sudoers
  ```
- **Implement Security Improvements:**

Apply patches:
```
apt-get update && apt-get upgrade
```
  - Enable SELinux/AppArmor and verify rules.
  - Deploy or update monitoring tools like `fail2ban`, `iptables`, or centralized logging systems.

Incident-Specific Instructions on Linux

Account Lockout

1. Check for failed login attempts:
   ```
   grep "Failed password" /var/log/auth.log
   ```
2. Unlock the account:
   ```
   usermod -U <username>
   ```
3. Reset the password securely:
   ```
   passwd <username>
   ```

Password Reset

1. Validate user identity using an alternate communication channel.
2. Reset the password:
   ```
   passwd <username>
   ```
3. Require the user to change the password upon next login:
   ```
   chage -d 0 <username>
   ```

System or Network Intrusion

1. Identify suspicious processes:
   ```
   ps aux | grep <suspicious_name>
   ```

2. Stop and quarantine the process:
   `kill -9 <pid>`
3. Investigate listening ports and active connections:
   `netstat -tulnp | grep <suspicious_port>`
4. Isolate the system by removing it from the network:
   `ifdown <interface>`
1.

Data Breach

1. Identify files accessed or modified:
   `find / -mtime -<days> -ls`
2. Check for data exfiltration using logs or network monitoring tools like `tcpdump` for Wireshark.
3. Disable access to sensitive files:
   `chmod 000 <file>`

---

1. Identify Open Ports

Use `netstat` or `ss` to identify listening ports:

```
netstat -tulnp or  ss -tulnp
```

Check firewall rules for currently allowed ports:

```
iptables -L -n -v or ufw status verbose
```

2. Determine Services Using Open Ports

Find which service is using a specific port:

```
lsof -i :<port_number>
```

3. Close Ports

Stop the associated service:

```
systemctl stop <service_name>
systemctl disable <service_name>
```

Block the port using `iptables`:

```
iptables -A INPUT -p tcp --dport <port_number> -j DROP
```

For persistent rules, save changes:

```
iptables-save > /etc/iptables/rules.v4
```

For `ufw` (Uncomplicated Firewall):

```
ufw deny <port_number>/tcp
```

Reload the firewall to apply changes:

```
ufw reload
```

Common Ports to Evaluate and Close

Unnecessary or Commonly Exploited Ports

1. **21 (FTP)**
   - Use SFTP (via SSH) or disable FTP if not needed.

   Close FTP port:
   ```
   systemctl stop vsftpd
   systemctl disable vsftpd
   ```
2. **23 (Telnet)**
   - Replace with SSH (port 22) as Telnet is insecure.

   Disable Telnet:
   ```
   systemctl stop telnet
   systemctl disable telnet
   ```
3. **25 (SMTP)**
   - Used for sending emails. Close if the server is not a mail server.

   Block SMTP:
   ```
   ufw deny 25/tcp
   ```
4. **69 (TFTP)**
   - Trivial File Transfer Protocol is insecure. Close unless explicitly needed.
5. **80 (HTTP)**
   - Close if you are not hosting a web server.
   - Redirect traffic to HTTPS (port 443).
6. **110 (POP3)** and **143 (IMAP)**
   - Replace with secure versions: POP3S (995) or IMAPS (993).
7. **135, 137-139, 445 (NetBIOS/SMB)**
   - Used for Windows file sharing. Disable if not needed.
8. **161-162 (SNMP)**
   - Simple Network Management Protocol can expose sensitive network information. Disable if not required.
9. **2049 (NFS)**
   - Disable Network File System unless it is necessary for file sharing.
10. **3306 (MySQL)**

    If the database server is local only, bind it to 127.0.0.1:
    ```
    bind-address = 127.0.0.1
    ```
11. **3389 (RDP)**
    - Used for Remote Desktop. If not needed, close this port.

12. **5000+ (Debugging/Development Ports)**
    - Many development applications open high-numbered ports (e.g., Flask, Node.js). Ensure these are closed in production.

For Public-Facing Systems

- Ensure only essential ports (e.g., 22 for SSH, 443 for HTTPS) are open to the internet.
  Use tools like `nmap` to scan your system from an external perspective to ensure no unnecessary ports are open:
  ```
  nmap -v <your_ip>
  ```
- **Use Firewalls for Tightened Control**
  For ufw:
  ```
  ufw default deny incoming
  ufw allow 22/tcp
  ufw allow 443/tcp
  ufw enable
  ```
- For `iptables`:
  Deny all incoming traffic by default:
  ```
  iptables -P INPUT DROP
  iptables -P FORWARD DROP
  iptables -P OUTPUT ACCEPT
  ```

Best Practices

1. **Audit Ports Regularly**:
    - Periodically review open ports and associated services.
2. **Limit SSH Access**:
   Restrict SSH (port 22) to specific IPs:
   ```
   iptables -A INPUT -p tcp --dport 22 -s <trusted_IP> -j ACCEPT
   ```
3. **Implement Intrusion Detection**:
    - Use tools like `fail2ban` to monitor and block suspicious login attempts.
4. **Enable Logging**:
   Ensure firewall rules are logged to review unauthorized attempts:
   ```
   iptables -A INPUT -j LOG --log-prefix "IPTables-Dropped: "
   ```
    -

By systematically evaluating open ports and closing unnecessary ones, you can significantly reduce the attack surface of a Linux system.