🔥 Linux Crisis Investigation - COMPLETE SOLUTIONS

Network Applications (QHO443) - Unit 07 Crisis Challenge Answer Key

---

📋 OVERVIEW

This document provides **COMPLETE MODEL ANSWERS** for all 5 challenges in the Linux Crisis Investigation activity.

Purpose: For instructors and students to:

- ✅ Understand what constitutes a complete answer
- ✅ See the technical depth expected
- ✅ Learn proper incident response reporting
- ✅ Compare their answers against model responses

🗂️ CHALLENGE #1: The Vanishing Configuration Files

📊 Challenge Details

- **Difficulty:** ROOKIE (20 points)
- **Word Count Required:** 100+ words
- **Scenario:** Root partition 100% full, someone deleted /etc directory

---

✅ MODEL ANSWER (Complete - 100% Score)

INCIDENT REPORT: Critical /etc Directory Deletion

What Happened: A user attempted to free disk space by deleting the /etc directory, which contains all critical system configuration files. This is catastrophic because /etc contains essential files like passwd (user accounts), group (group definitions), shadow (encrypted passwords), hosts (hostname resolution), fstab (filesystem mounts), and network configuration. Without these files, the system cannot authenticate users, resolve hostnames, or boot properly.

Why This Is Critical: The /etc directory is the "brain" of Linux. Deleting it renders the system completely non-functional. Only 3 files remain (likely passwd, group, and shadow backups), meaning most services cannot start, network is down, and user authentication is broken. The root partition being 100% full indicates the system was already under storage stress.

How to Recover:

1. Boot from a live USB/rescue disk to bypass authentication issues
2. Mount the damaged filesystem to a recovery directory
3. Restore /etc from system backups (located in /var/backups/ or external backup solution)
4. If no backups exist, rebuild critical configs manually using package reinstallation: `apt-get install --reinstall $(dpkg --get-selections | grep install | cut -f1)`
5. Free disk space by removing unnecessary files from /var/log and /tmp
6. Verify system integrity before rebooting

Prevention:

- Educate users: NEVER delete system directories (/etc, /var, /usr, /bin, /sbin)
- Implement disk space monitoring alerts
- Maintain regular automated backups of /etc
- Use disk quotas to prevent any user from filling the partition
- Provide proper disk cleanup procedures (apt clean, log rotation, temp file removal)

[182 words]

🎯 KEY POINTS THAT EARN FULL CREDIT

Must Include (Technical Understanding):

1. ✅ Explanation that /etc contains configuration files
2. ✅ Specific examples of critical files (passwd, shadow, hosts, fstab)
3. ✅ Why this causes system failure (no authentication, no services, no boot)
4. ✅ Recognition that only 3 files remain

Must Include (Recovery Process):

1. ✅ Boot from rescue media
2. ✅ Restore from backup (mention /var/backups or external)
3. ✅ Package reinstallation as alternative
4. ✅ Free disk space as part of solution

Must Include (Prevention):

1. ✅ User education
2. ✅ Regular backups
3. ✅ Disk monitoring
4. ✅ Proper cleanup procedures

Keyword Checklist:

- /etc directory ✓
- Configuration files ✓
- passwd/shadow/hosts ✓
- Backup/restore ✓
- Boot from rescue media ✓
- System failure ✓
- Prevention measures ✓

⚠️ COMMON MISTAKES (Lose Points)

❌ Just saying "restore from backup" without explaining HOW ❌ Not explaining WHY /etc is critical ❌ Forgetting to mention freeing disk space ❌ No prevention measures ❌ Too short (less than 100 words) ❌ No specific file examples

---

⚙️ CHALLENGE #2: The Zombie Process Apocalypse

📊 Challenge Details

- **Difficulty:** EXPERT (30 points)
- **Word Count Required:** 150+ words
- **Scenario:** 768 zombie processes, memory exhausted, parent PID 1234 not reaping children

---

✅ MODEL ANSWER (Complete - 100% Score)

INCIDENT REPORT: Zombie Process Outbreak

What Happened: The system has 768 zombie processes (shown as `<defunct>` in ps output) all parented by PID 1234. A zombie process is a child process that has completed execution but whose parent hasn't called wait() to read its exit status. Zombies don't consume CPU or memory resources themselves—they're already dead—but they occupy process table entries. However, the parent process (PID 1234) is consuming excessive memory trying to manage all these dead children, causing the actual memory exhaustion.

Why Can't We Kill Zombies: You cannot kill zombie processes with `kill -9` because they're already dead. The kill command only works on running processes. Zombies exist in a terminated state waiting for their parent to acknowledge their death by calling wait() or waitpid(). Attempting to kill a zombie directly has no effect.

The Real Culprit: The problem is the parent process (PID 1234), which is poorly coded and fails to reap (clean up) its terminated children. This is a programming bug where the parent creates child processes but doesn't properly handle their termination. Each zombie persists until the parent process either reads its exit status or terminates itself.

Correct Fix:

1. Identify the parent process: `ps -o ppid= -p [zombie_pid]` or `ps aux | grep defunct`
2. Kill the parent process (PID 1234): `sudo kill 1234`
3. Wait 5 seconds; if still running, force kill: `sudo kill -9 1234`
4. When the parent dies, all zombie children are automatically reaped by init (PID 1) and removed from the process table
5. Verify cleanup: `ps aux | grep defunct` should show nothing
6. Monitor system: Check if the process respawns and investigate why

Root Cause Analysis: Contact the developers of the application running as PID 1234. The code needs fixing to properly call wait() or implement signal handlers for SIGCHLD to automatically reap terminated children. This is a code-level bug, not a system configuration issue.

Prevention:

- Code review: Ensure all applications properly handle child process termination
- Implement monitoring for excessive zombie counts
- Use process managers (systemd, supervisord) that handle child reaping automatically
- Set resource limits to prevent any single process from creating excessive children

[354 words]

🎯 KEY POINTS THAT EARN FULL CREDIT

Must Include (Understanding Zombies):

1. ✅ Explain what zombie processes are (terminated but not reaped)
2. ✅ Clarify they DON'T consume CPU/memory (already dead)
3. ✅ Note parent process IS consuming memory
4. ✅ Mention they're shown as `<defunct>`

Must Include (Why Kill Doesn't Work):

1. ✅ Can't kill zombies - already dead
2. ✅ kill only works on running processes
3. ✅ Need to kill parent instead

Must Include (Correct Solution):

1. ✅ Identify parent process (PID 1234)
2. ✅ Kill parent process
3. ✅ Zombies automatically reaped by init
4. ✅ Verification step

Must Include (Root Cause):

1. ✅ Parent not calling wait()
2. ✅ Programming bug
3. ✅ Need code fix

Keyword Checklist:

- Zombie process ✓
- Parent process ✓
- wait() or reaping ✓
- Cannot kill zombies ✓
- Kill parent ✓
- Init/PID 1 ✓
- <defunct> ✓
- Programming bug ✓

⚠️ COMMON MISTAKES (Lose Points)

❌ Trying to kill zombie processes directly ❌ Saying zombies consume memory ❌ Not identifying the parent as the problem ❌ No explanation of what zombies are ❌ No root cause analysis ❌ Missing verification steps

---

📝 CHALLENGE #3: The Authentication Attack

📊 Challenge Details

- **Difficulty:** EXPERT (30 points)
- **Word Count Required:** 200+ words
- **Scenario:** Brute force attack, compromised admin, backdoor user created

---

✅ MODEL ANSWER (Complete - 100% Score)

SECURITY INCIDENT REPORT: Successful Brute Force Attack

Attack Timeline Reconstruction:

1. Initial Attack (14:23:15 - 14:47:33): The attacker began a brute force attack from IP 203.0.113.50 targeting the 'admin' account. The logs show repeated failed password attempts using common passwords. Between 14:23:15 and 14:47:33, there were hundreds of authentication failures for user 'admin' from the same source IP. This indicates an automated brute force tool attempting to guess the admin password.

2. Successful Compromise (14:47:33): At 14:47:33, the attack succeeded with "Accepted password for admin from 203.0.113.50". The attacker successfully guessed or cracked the admin password and gained shell access via SSH. This marks the moment of system compromise.

3. Backdoor Creation (14:48:12): Immediately after gaining access (less than 1 minute), at 14:48:12, a new user account 'backup_svc' was created using the command `useradd -m -G sudo backup_svc`. The -G sudo flag adds this user to the sudo group, giving full root privileges. This is a classic persistence technique—even if the admin password is changed, the attacker maintains access through this backdoor account.

4. Continued Access: Subsequent logs show successful logins using the 'backup_svc' account from the same IP address, confirming the backdoor is functional and actively being used.

Vulnerabilities Exploited:

1. **Weak Password:** The admin account used a password weak enough to be brute-forced in ~24 minutes
2. **No Rate Limiting:** The system allowed hundreds of authentication attempts without blocking or slowing the attacker
3. **No Fail2Ban:** No automated IP blocking system was in place
4. **Direct Root/Sudo Access:** Admin account had sudo privileges, so compromise immediately gave root access
5. **No Multi-Factor Authentication (MFA):** Password was the only authentication factor
6. **SSH Exposed to Internet:** SSH port 22 was accessible from external IP addresses

Immediate Response Actions:

1. **Isolate the System:** Disconnect from network immediately to prevent further damage or lateral movement
2. **Lock Compromised Accounts:**
   - `sudo passwd -l admin` (lock admin account)
   - `sudo userdel -r backup_svc` (remove backdoor user and home directory)
3. **Kill Active Sessions:** `sudo pkill -u backup_svc` and check for any sessions from 203.0.113.50
4. **Change All Passwords:** Reset passwords for all privileged accounts
5. **Check for Additional Backdoors:**
   - Review /etc/passwd for suspicious accounts

- Check for SSH keys in ~/.ssh/authorized_keys
- Look for cronjobs: `crontab -l` for all users
- Scan for unauthorized SUID binaries

6. **Preserve Evidence:** Make copies of /var/log/auth.log for forensic analysis
7. **Block Attacker IP:** `sudo ufw deny from 203.0.113.50` or add to firewall rules

Long-Term Prevention Measures:

1. **Implement Fail2Ban:** Automatically ban IPs after failed login attempts
   - `sudo apt install fail2ban`
   - Configure to ban after 5 failed attempts for 1 hour
2. **Enforce Strong Password Policy:**
   - Use PAM to require complex passwords (uppercase, lowercase, numbers, symbols)
   - Minimum 12 characters
   - Regular password rotation
3. **Deploy Multi-Factor Authentication (MFA):**
   - Implement Google Authenticator or similar for SSH
   - Require MFA for all sudo accounts
4. **Rate Limiting:** Configure SSH to limit connection attempts
5. **Change SSH Default Port:** Move SSH from port 22 to non-standard port (e.g., 2222)
6. **Disable Password Authentication:** Use SSH keys only: `PasswordAuthentication no` in /etc/ssh/sshd_config
7. **Implement Network Segmentation:** SSH should only be accessible from specific trusted IP ranges
8. **Enable Intrusion Detection:** Install and configure AIDE or Tripwire
9. **Regular Security Audits:** Review auth logs daily, monitor for suspicious activity
10. **Account Monitoring:** Alert on new user creation, sudo group changes
11. **Disable Root SSH:** `PermitRootLogin no` in sshd_config
12. **Use VPN:** Require VPN connection before SSH access is possible

Lessons Learned: This breach was preventable with basic security hardening. The combination of weak passwords, no rate limiting, and no MFA created an easily exploitable attack surface. Organizations must

implement defense-in-depth strategies, assuming password compromise will eventually occur and building multiple layers of protection.

[615 words]

🎯 KEY POINTS THAT EARN FULL CREDIT

Must Include (Attack Timeline):

1. ✅ Initial brute force attempts identified
2. ✅ Successful compromise timestamp (14:47:33)
3. ✅ Backdoor creation timestamp (14:48:12)
4. ✅ Recognition of automated attack pattern

Must Include (Vulnerabilities):

1. ✅ Weak password
2. ✅ No rate limiting
3. ✅ No fail2ban or IP blocking
4. ✅ No MFA
5. ✅ At least 3 vulnerabilities identified

Must Include (Response Actions):

1. ✅ Lock compromised accounts
2. ✅ Remove backdoor user
3. ✅ Change passwords
4. ✅ Block attacker IP
5. ✅ Check for additional backdoors

Must Include (Prevention):

1. ✅ Fail2ban installation
2. ✅ Strong password policy
3. ✅ MFA implementation
4. ✅ SSH hardening

5. ✅ At least 5 prevention measures

Keyword Checklist:

- Brute force ✓
- Failed attempts ✓
- Successful compromise ✓
- Backdoor user ✓
- sudo group ✓
- Weak password ✓
- Fail2ban ✓
- MFA ✓
- SSH hardening ✓
- Prevention measures ✓

⚠️ COMMON MISTAKES (Lose Points)

❌ Not reconstructing the timeline ❌ Missing the backdoor user creation ❌ No vulnerability analysis ❌ Only giving response OR prevention, not both ❌ No mention of fail2ban ❌ No mention of MFA ❌ Too short (less than 200 words) ❌ Not explaining why each step matters

---

📦 CHALLENGE #4: The Broken Package Manager

📊 Challenge Details

- **Difficulty:** ROOKIE (15 points)
- **Word Count Required:** 120+ words
- **Scenario:** APT locked, zombie apt-get process holding locks

---

✅ MODEL ANSWER (Complete - 100% Score)

INCIDENT REPORT: APT Lock File Deadlock

What Happened: The APT package manager is locked, preventing any software installation or updates. This occurs because APT uses lock files to prevent multiple package operations from running simultaneously, which could corrupt the package database. The error indicates another package manager process is running or a previous apt operation crashed without releasing its locks.

Root Cause: A zombie apt-get process (PID 3847) is holding the lock files. This process started hours ago but never completed, either due to network timeout, system interruption, or a bug. The lock files (/var/lib/dpkg/lock, /var/lib/apt/lists/lock, /var/cache/apt/archives/lock) remain claimed by this dead process.

Why Lock Files Exist: Lock files prevent database corruption. If two apt processes ran simultaneously, they could conflict when modifying /var/lib/dpkg/status (the package database), leading to inconsistent package states, missing dependencies, or broken installations. Locks ensure only one package operation occurs at a time.

Safe Resolution Procedure:

Step 1: Verify No Active Operations

```bash
ps aux | grep -E 'apt|dpkg'
```

Check if any legitimate apt/dpkg processes are actually running. If you see processes consuming CPU or with recent start times, wait for them to complete naturally.

Step 2: Kill Zombie Process

```bash
sudo kill 3847        # Try graceful termination first
sleep 5
ps aux | grep 3847    # Check if still running
sudo kill -9 3847     # Force kill if necessary
```

Step 3: Remove Stale Lock Files

```bash
```

```
sudo rm /var/lib/dpkg/lock-frontend
sudo rm /var/lib/dpkg/lock
sudo rm /var/lib/apt/lists/lock
sudo rm /var/cache/apt/archives/lock
```

Only remove locks after confirming no processes are using them!

Step 4: Reconfigure Package System

```
bash

sudo dpkg --configure -a
```

This completes any interrupted package configurations and ensures database consistency.

Step 5: Update Package Lists

```
bash

sudo apt update
```

Refresh the package database to verify APT is working properly.

Verification: Test by installing a small package:

```
bash

sudo apt install tree
```

If installation succeeds without lock errors, the problem is resolved.

Prevention:

- Don't force-quit apt operations (let them complete or cancel gracefully)
- Ensure stable network connections during updates
- Use `screen` or `tmux` for long-running updates over SSH
- If system crashes during apt operations, run `sudo dpkg --configure -a` on next boot
- Regular system maintenance to prevent unfinished package operations

[369 words]

🎯 KEY POINTS THAT EARN FULL CREDIT

Must Include (Understanding):

1. ✅ Explain what lock files are
2. ✅ Why they prevent simultaneous operations
3. ✅ Prevent database corruption
4. ✅ Zombie process is holding locks

Must Include (Resolution Steps):

1. ✅ Check for active processes first
2. ✅ Kill zombie process
3. ✅ Remove lock files (with caution warning)
4. ✅ Run dpkg --configure -a
5. ✅ Verification step

Must Include (Safety):

1. ✅ Warning to check for active processes
2. ✅ Only remove locks if safe
3. ✅ Try graceful kill before force kill

Keyword Checklist:

- Lock files ✓
- Database corruption ✓
- Zombie process ✓
- dpkg --configure -a ✓
- Kill process ✓
- Remove locks ✓
- Safety verification ✓

⚠️ COMMON MISTAKES (Lose Points)

❌ Immediately deleting lock files without checking for active processes ❌ Not explaining WHY locks exist ❌ No mention of dpkg --configure -a ❌ No verification step ❌ Not killing the zombie process ❌ Too short (less than 120 words)

🌐 CHALLENGE #5: The DNS Mystery

📊 Challenge Details

- **Difficulty:** NIGHTMARE (35 points)
- **Word Count Required:** 250+ words
- **Scenario:** Can ping IPs but not domain names, TWO layered problems

---

✅ MODEL ANSWER (Complete - 100% Score)

CRITICAL INCIDENT REPORT: Multi-Layer DNS Failure

Symptom Analysis:

The system can ping IP addresses (8.8.8.8 works) but cannot resolve domain names (google.com fails). This definitively indicates DNS resolution is broken, not the network connection itself. Network connectivity is functional (verified by successful IP pings), but the Domain Name System translation from names to IPs has failed.

Investigation Findings - TWO Separate Problems:

PROBLEM #1: Configuration File Permission Error

Discovery:

```
bash

ls -l /etc/systemd/resolved.conf
-rw------- 1 root root 847 Dec 3 17:45 /etc/systemd/resolved.conf
```

The resolved.conf file has permissions 600 (rw-------), meaning only root can read it. The systemd-resolved service runs as user 'systemd-resolve' (not root), so it cannot read its own configuration file.

Technical Details: systemd-resolved is the DNS resolution service in modern Linux. It reads /etc/systemd/resolved.conf on startup to determine DNS servers and settings. With 600 permissions, when systemd-resolved tries to read the config file as the 'systemd-resolve' user, it gets "Permission denied" and falls back to no DNS configuration, causing complete DNS failure.

Fix for Problem #1:

```bash
sudo chmod 644 /etc/systemd/resolved.conf
```

This sets permissions to rw-r--r--, allowing the service to read its configuration.

PROBLEM #2: Unreachable Custom DNS Server

Discovery: The custom configuration file (/etc/systemd/resolved.conf.d/custom-dns.conf) specifies DNS=192.168.1.1. This is typically a router's IP address. However, investigation reveals:

```bash
ping 192.168.1.1
ping: connect: Network is unreachable
```

The DNS server at 192.168.1.1 is not accessible. This could be because:

- The router is offline or has been replaced
- The router's IP changed to a different address
- Network routing to that subnet is broken
- The router's DNS service is disabled

Technical Details: Even if systemd-resolved can now read the config file (after fixing permissions), it will try to use 192.168.1.1 as the DNS server. When queries are sent to an unreachable DNS server, they timeout after several seconds with no response, effectively preventing all DNS resolution.

Fix for Problem #2 (Option A - Quick Fix):

```bash
sudo rm /etc/systemd/resolved.conf.d/custom-dns.conf
```

Remove the custom DNS configuration. System will fall back to DNS servers provided by DHCP or the FallbackDNS servers in the main resolved.conf.

Fix for Problem #2 (Option B - Proper Fix):

```
bash

sudo nano /etc/systemd/resolved.conf.d/custom-dns.conf
```

Change DNS=192.168.1.1 to working DNS servers:

```
[Resolve]
DNS=8.8.8.8 8.8.4.4
FallbackDNS=1.1.1.1 1.0.0.1
```

Complete Resolution Procedure:

Step 1: Fix Permissions

```
bash

sudo chmod 644 /etc/systemd/resolved.conf
```

Step 2: Fix DNS Server Configuration

```
bash

sudo nano /etc/systemd/resolved.conf.d/custom-dns.conf
```

Change DNS=192.168.1.1 to DNS=8.8.8.8 8.8.4.4

Step 3: Restart systemd-resolved

```
bash

sudo systemctl restart systemd-resolved
```

Step 4: Verify Service is Running

```
bash

sudo systemctl status systemd-resolved
```

Should show "active (running)" in green.

Step 5: Check DNS Configuration

```bash
systemd-resolve --status
```

Verify it shows the correct DNS servers (8.8.8.8, not 192.168.1.1).

Step 6: Test DNS Resolution

```bash
nslookup google.com
```

Should return IP addresses, confirming DNS works.

Step 7: Test End-to-End

```bash
ping google.com
```

Should successfully ping google.com, proving both network and DNS work.

Step 8: Verify Persistent

```bash
sudo reboot
# After reboot:
ping google.com
```

Ensure fix persists across reboots.

Root Cause Analysis:

Problem 1 Origin: Someone (likely during "DNS improvements") incorrectly changed the permissions on resolved.conf using `chmod 600` thinking it would make the file more secure. However, this broke the service's ability to read its own config. Proper security is chmod 644 - readable by all, writable only by root.

Problem 2 Origin: The custom DNS configuration was created pointing to a router (192.168.1.1) that either failed, was reconfigured, or was replaced. No monitoring was in place to detect when this DNS server became unreachable. The FallbackDNS servers should have activated but didn't because of the permission problem preventing the entire config from being read.

Why Both Problems Together Caused Complete Failure:

If only Problem 1 existed (permission error), the FallbackDNS servers in the default config might have worked. If only Problem 2 existed (unreachable DNS), systemd-resolved could have read the config and fallen back to FallbackDNS. But with BOTH problems, systemd-resolved couldn't read any configuration AND the primary DNS was unreachable, causing total DNS failure.

Prevention Measures:

1. **Configuration Management:**
   - Use version control (git) for /etc configuration changes
   - Document all configuration changes in a change log
   - Require peer review before modifying critical services
2. **Permission Auditing:**
   - Regular audits of critical file permissions
   - Automated alerts for permission changes on system files
   - Training on proper Linux file permissions
3. **DNS Monitoring:**
   - Monitor DNS resolution health (automated checks every 5 minutes)
   - Alert if DNS queries fail
   - Monitor DNS server reachability
   - Configure multiple fallback DNS servers
4. **Testing:**
   - Test DNS after any network changes: `nslookup google.com`
   - Verify services can read their config files
   - Implement canary tests for critical services
5. **Documentation:**
   - Document why custom DNS servers are configured
   - Document correct file permissions for system configs
   - Create runbooks for common DNS troubleshooting
6. **Redundancy:**
   - Always configure multiple DNS servers (primary + fallback)
   - Use reliable public DNS servers (8.8.8.8, 1.1.1.1) as fallbacks

- Don't rely on single points of failure (one router)

Technical Learning Points:

1. systemd-resolved runs as unprivileged user for security
2. Config files need appropriate permissions (644 for readable, 600 for sensitive)
3. DNS troubleshooting: if IP works but names don't = DNS problem
4. Multiple layers of failure can compound to total outage
5. Always test configurations after changes
6. Monitoring and alerting prevent extended outages

[1084 words]

🎯 KEY POINTS THAT EARN FULL CREDIT

Must Include (Problem Identification):

1. ✅ TWO separate problems identified
2. ✅ Problem 1: Permission error (600 should be 644)
3. ✅ Problem 2: Unreachable DNS server (192.168.1.1)
4. ✅ Explanation of why each causes DNS failure

Must Include (Technical Understanding):

1. ✅ systemd-resolved can't read config with 600 permissions
2. ✅ Service runs as unprivileged user
3. ✅ Unreachable DNS server causes timeouts
4. ✅ Why both together cause complete failure

Must Include (Complete Fix):

1. ✅ chmod 644 /etc/systemd/resolved.conf
2. ✅ Fix or remove custom-dns.conf
3. ✅ Restart systemd-resolved
4. ✅ Verification with nslookup

5. ✅ All 8 resolution steps

Must Include (Prevention):

1. ✅ Monitor DNS health
2. ✅ Multiple fallback DNS servers
3. ✅ Permission auditing
4. ✅ Configuration management
5. ✅ At least 3 prevention measures

Keyword Checklist:

- TWO problems ✓
- Permission 600 vs 644 ✓
- systemd-resolved ✓
- Unreachable DNS ✓
- 192.168.1.1 ✓
- chmod 644 ✓
- Restart service ✓
- nslookup verification ✓
- Multiple DNS servers ✓
- Monitoring ✓

⚠️ COMMON MISTAKES (Lose Major Points)

❌ Identifying only ONE problem (major point loss) ❌ Not explaining the permission issue ❌ Not explaining the unreachable DNS server ❌ Incomplete fix (missing restart or verification) ❌ No explanation of WHY each problem causes failure ❌ No prevention measures ❌ Too short (less than 250 words) ❌ Not explaining how both problems compound

---

📊 SCORING SUMMARY

Maximum Points by Challenge

Challenge	Points	Difficulty
#1: Vanishing Configuration	20	ROOKIE
#2: Zombie Apocalypse	30	EXPERT
#3: Authentication Attack	30	EXPERT
#4: Broken Package Manager	15	ROOKIE
#5: DNS Mystery	35	NIGHTMARE
TOTAL	**130**	

Grading Scale

Score	Grade	Performance
110-130	A+	Exceptional - Professional level
90-109	A	Excellent - Deep understanding
70-89	B	Good - Solid grasp
50-69	C	Satisfactory - Meets requirements
30-49	D	Pass - Needs improvement
0-29	F	Fail - Requires review

Points Breakdown by Category

Understanding (40%):

- Explain what happened
- Why it's critical
- Technical details
- Root cause analysis

Solution (35%):

- Correct fix procedure
- Step-by-step commands
- Verification steps
- Safety considerations

Prevention (25%):

- Long-term measures
- Monitoring and alerts
- Best practices
- Lessons learned

🎓 ASSESSMENT RUBRIC

For Each Challenge

EXCELLENT (90-100% of points):

- ✅ All technical details correct
- ✅ Complete step-by-step solution
- ✅ Thorough prevention measures
- ✅ Proper incident report format
- ✅ Exceeds minimum word count
- ✅ Professional quality

GOOD (70-89% of points):

- ✅ Most technical details correct
- ✅ Solution mostly complete
- ✅ Some prevention measures
- ✅ Meets minimum word count
- ⚠️ Minor gaps or inaccuracies

SATISFACTORY (50-69% of points):

- ✅ Basic understanding shown
- ✅ Partial solution provided
- ⚠️ Limited prevention measures
- ⚠️ Barely meets word count
- ⚠️ Some technical errors

NEEDS IMPROVEMENT (<50% of points):

- ❌ Major technical misunderstandings
- ❌ Incomplete solution
- ❌ No prevention measures
- ❌ Below minimum word count
- ❌ Significant errors

💡 TIPS FOR STUDENTS

How to Write Excellent Answers

1. Structure Your Response:

> 1. What Happened (situation analysis)
> 2. Why It's Critical (impact)
> 3. Root Cause (technical explanation)
> 4. Fix Procedure (step-by-step)
> 5. Verification (how to confirm it's fixed)
> 6. Prevention (how to avoid in future)

2. Use Technical Terms Correctly:

- Don't say "the computer" → say "the system" or "the server"
- Don't say "stuff" → be specific
- Use proper command names
- Explain technical terms

3. Show Your Thinking:

- Explain WHY each step is needed
- Show cause and effect
- Demonstrate understanding, not just memorization

4. Be Thorough:

- Better to write too much than too little
- Include verification steps
- Think about prevention
- Consider edge cases

5. Format Matters:

- Use paragraphs (not walls of text)
- Commands in code format where possible
- Clear section headings
- Professional tone

📚 LEARNING OUTCOMES

After Completing These Challenges

Students should be able to:

✅ **Analyze system failures** from log files and error messages ✅ **Identify root causes** vs. symptoms ✅ **Develop systematic solutions** with verification ✅ **Think about prevention** not just fixing ✅ **Write professional incident reports** ✅ **Apply Linux knowledge** to real scenarios ✅ **Troubleshoot under pressure** ✅ **Understand security implications**

🔗 RELATED CONCEPTS

Additional Topics to Study

For Challenge #1 (Filesystem):

- File system hierarchy
- Backup strategies
- Recovery procedures
- Disk space management

For Challenge #2 (Processes):

- Process lifecycle
- Parent-child relationships
- Signal handling
- Process states

For Challenge #3 (Security):

- SSH hardening
- Intrusion detection
- Log analysis
- Security best practices

For Challenge #4 (Packages):

- Package management
- Lock mechanisms
- Database integrity
- System recovery

For Challenge #5 (DNS):

- DNS architecture
- systemd-resolved
- File permissions

- Network troubleshooting

Solutions Created by: Babashaheer
Module: QHO443 Network Applications
Institution: Southampton Solent University (QA Partnership)
Purpose: Model answers for Linux Crisis Investigation Challenge
Date: December 2024

📝 INSTRUCTOR NOTES

Using These Solutions:

1. **Don't give to students before the challenge!**
2. Use for grading reference
3. Share after completion for learning
4. Discuss in post-activity debrief
5. Highlight different approaches that work
6. Use to calibrate expectations

Grading Tips:

- Award partial credit for correct concepts even if incomplete
- Value understanding over perfect wording
- Look for key technical terms in keyword checklist
- Consider word count as minimum, not maximum
- Professional format and clarity matter

Good luck assessing! 🚀