SOUTHGATE TERMINAL

## Port Operations Security Documentation

VM-Specific Investigation Procedures

Document Information

Document Type: Technical Investigation Framework Intended Users: Technical Team Usage Context: Systematic investigation of compromised VM systems Related Scenarios: System anomalies, suspected compromises, evidence collection

---

Purpose

This document provides detailed investigation procedures for each VM system in the network, including specific warnings about trap scripts, evidence preservation requirements, and system-specific indicators of compromise.

When to Use These Procedures

- System anomalies detected on specific VMs
- Evidence collection required for forensic analysis
- Suspected unauthorized access or system compromise
- Cross-system correlation analysis needed
- Preparation for evidence transfer to vm-audit

General Investigation Principles

CRITICAL WARNING: TRAP SCRIPTS AND FAKE SYSTEMS

SEVERAL VMs CONTAIN DELIBERATELY MISLEADING SCRIPTS AND SERVICES - DO NOT execute scripts found in obvious locations without analysis - VERIFY the legitimacy of any “restoration” or “cleanup” scripts - PRESERVE evidence BEFORE attempting any remediation - DOCUMENT all findings before making ANY changes

RAPID INVESTIGATION MODE (When Under Time Pressure)

IF YOU HAVE LIMITED TIME (20-30 minutes):

IMMEDIATE ACTIONS (First 5 minutes): 1. Evidence Preservation FIRST - Hash all log files immediately 2. Team Task Division - Assign one VM per team member if multiple people available 3. Start with vm-gateway - Most likely to contain evidence destruction traps

STREAMLINED INVESTIGATION PATH (15-20 minutes): 1. Priority Order: vm-gateway - vm-coretech - vm-opsnode 2. Focus on critical files only: Main log files, obvious script locations, /etc/cron.d/ 3. Document trap scripts but DO NOT EXECUTE 4. Skip detailed analysis - focus on evidence collection

MINIMUM DELIVERABLE (Last 5 minutes): - Evidence package with hashes - List of trap scripts found (with warnings) - Basic timeline of system compromise indicators

TRAP SCRIPT QUICK IDENTIFICATION: - DANGER SIGNS: Scripts in /opt/security/, /home/, obvious names like "restore_", "remove_", "fix_" - SAFE ASSUMPTION: If it looks helpful, it's probably a trap - WHEN IN DOUBT: Document and avoid execution

Investigation Priority Order

1. PRESERVE EVIDENCE: Secure all logs and artifacts immediately
2. ANALYZE SAFELY: Examine systems without making changes
3. DOCUMENT THOROUGHLY: Record all findings with timestamps
4. TRANSFER SECURELY: Move evidence to vm-audit with proper chain of custody
5. COORDINATE: Share findings with incident coordinator and other teams

Evidence Preservation Requirements

- Generate SHA256 hashes of all log files BEFORE examination
- Create timestamps for all evidence collection activities
- Maintain chain of custody documentation
- Use read-only access methods when possible
- Document any changes made during investigation

VM-CORETECH Investigation Procedures

System Overview

Purpose: Backend for AIS tracking and route planning system Expected Issues: GPS jamming impacts, log tampering, fake recovery attempts Key Files: `/var/log/sim/ais_feed.log`, `/opt/reference/ais_reference.log`

CRITICAL WARNING - TRAP ELEMENTS:

FAKE RESTORATION SCRIPT: `/home/ubuntu/restore_feed.sh` appears helpful but is a trap LOG TRUNCATION: Cron job truncates logs every 5 minutes FAKE SERVICE: `ais-feed.service` is meaningless distraction

Step-by-Step Investigation

Phase 1: Initial System Assessment (0-10 minutes)

```bash
# FIRST - Preserve evidence IMMEDIATELY
cd /var/log/sim/
sha256sum ais_feed.log > /tmp/ais_feed_hash_$(date +%Y%m%d_%H%M%S).txt
cp ais_feed.log /tmp/ais_feed_backup_$(date +%Y%m%d_%H%M%S).log

# Check system status
systemctl status ais-feed.service
journalctl -u ais-feed.service --since "1 hour ago"
ps aux | grep -i ais
```

Phase 2: Log Analysis (10-20 minutes)

```bash
# Examine log content for anomalies
tail -n 100 /var/log/sim/ais_feed.log
grep -i "signal\|lost\|degraded\|restart" /var/log/sim/ais_feed.log
grep -i "Ship_Alpha\|Ship_Bravo" /var/log/sim/ais_feed.log

# Check for timestamp gaps or inconsistencies
awk '{print $1, $2}' /var/log/sim/ais_feed.log | sort | uniq -c
```

Phase 3: TRAP DETECTION (20-25 minutes)

```bash
# Check for malicious cron jobs
crontab -l
cat /etc/cron.d/truncate_ais # This will likely contain the log truncation job
ls -la /etc/cron.d/

# Examine the trap script WITHOUT EXECUTING
ls -la /home/ubuntu/restore_feed.sh
cat /home/ubuntu/restore_feed.sh # Will reveal it's a trap
file /home/ubuntu/restore_feed.sh
```

EXPECTED FINDINGS: - Cron Job: `/etc/cron.d/truncate_ais` truncates logs every 5 minutes - Trap Script: `/home/ubuntu/restore_feed.sh` contains "[H4x0r] Nice try… maybe don't trust everything in /home ;)" - Fake Service: `ais-feed.service` runs but performs no meaningful function

Phase 4: Evidence Collection (25-30 minutes)

```bash
# Document cron job
cp /etc/cron.d/truncate_ais /tmp/malicious_cron_$(date +%Y%m%d_%H%M%S).txt
```

```bash
# Hash all evidence
sha256sum /opt/reference/ais_reference.log > /tmp/reference_hash.txt
sha256sum /home/ubuntu/restore_feed.sh > /tmp/trap_script_hash.txt

# Create investigation summary
cat > /tmp/vm_coretech_investigation_$(date +%Y%m%d_%H%M%S).txt <<EOF
VM-CORETECH INVESTIGATION SUMMARY
Date: $(date)
Investigator: $(whoami)

FINDINGS:
1. Malicious cron job found: /etc/cron.d/truncate_ais
2. Trap script detected: /home/ubuntu/restore_feed.sh
3. Fake service running: ais-feed.service
4. Log corruption evident in ais_feed.log
5. GPS interference patterns detected

EVIDENCE PRESERVED:
- ais_feed.log backup and hash
- malicious cron job copy
- trap script hash
- reference log hash

ACTIONS TAKEN:
- Evidence preserved before any changes
- Trap script identified and avoided
- System status documented
EOF
```

Phase 5: System Remediation (If Safe)

```bash
# ONLY if authorized and after evidence preservation:
# Remove malicious cron job
sudo rm /etc/cron.d/truncate_ais

# Stop fake service (optional)
sudo systemctl stop ais-feed.service
sudo systemctl disable ais-feed.service
```

Expected Investigation Outcomes

- Trap Script Avoided: Team should identify and document trap without executing
- Cron Job Detected: Malicious log truncation identified and documented
- Evidence Preserved: All logs and artifacts secured with proper hashing

- System Impact Understood: GPS interference effects on AIS feed documented

---

VM-GATEWAY Investigation Procedures

System Overview

Purpose: Vendor-facing gateway connected to external systems Expected Issues: Vendor session logs, suspicious authentication, evidence destruction trap Key Files: `/var/log/gateway/vendor.log`, `/var/log/gateway/auth.log`

CRITICAL WARNING - EVIDENCE DESTRUCTION TRAP:

MALWARE REMOVAL SCRIPT: `/opt/security/remove_malware.sh` WIPES ALL LOGS THIS VM SHOULD NOT BE "FIXED" - ONLY INVESTIGATED ANY "CLEANUP" WILL DESTROY FORENSIC EVIDENCE

Step-by-Step Investigation

Phase 1: Immediate Evidence Preservation (0-5 minutes)

```
# CRITICAL - Preserve evidence BEFORE any examination
cd /var/log/gateway/
sha256sum vendor.log > /tmp/vendor_log_hash_$(date +%Y%m%d_%H%M%S).txt
sha256sum auth.log > /tmp/auth_log_hash_$(date +%Y%m%d_%H%M%S).txt
cp vendor.log /tmp/vendor_log_backup_$(date +%Y%m%d_%H%M%S).log
cp auth.log /tmp/auth_log_backup_$(date +%Y%m%d_%H%M%S).log
```

Phase 2: Log Content Analysis (5-15 minutes)

```
# Examine vendor log for suspicious activity
cat /var/log/gateway/vendor.log
grep -i "payload\|inject\|admin\|temp_session" /var/log/gateway/vendor.log
grep -i "checksum\|failure\|error" /var/log/gateway/vendor.log

# Examine authentication log for failed attempts
cat /var/log/gateway/auth.log
grep -i "failed\|root\|sudo\|ghost" /var/log/gateway/auth.log
grep -c "failed" /var/log/gateway/auth.log
```

Phase 3: TRAP SCRIPT DETECTION (15-20 minutes)

```
# Examine the dangerous trap script WITHOUT EXECUTING
ls -la /opt/security/
cat /opt/security/remove_malware.sh # Will show it deletes all logs
```

```
file /opt/security/remove_malware.sh

# Check reference hash for integrity verification
cat /opt/reference/hash_expected.txt
sha256sum /var/log/gateway/vendor.log # Compare with expected
```

EXPECTED FINDINGS: - Vendor Log: Contains admin temp_session, inject_payload.sh, checksum failures - Auth Log: Multiple failed root login attempts, successful login by user "ghost" - Trap Script: `/opt/security/remove_malware.sh` contains `rm -f /var/log/gateway/*` - Hash Mismatch: Current vendor.log hash differs from expected reference

Phase 4: Evidence Documentation (20-25 minutes)

```
# Document all findings without making changes
cat > /tmp/vm_gateway_investigation_$(date +%Y%m%d_%H%M%S).txt <<EOF
VM-GATEWAY INVESTIGATION SUMMARY
Date: $(date)
Investigator: $(whoami)

FINDINGS:
1. Suspicious vendor session with admin temp_session=true
2. Evidence of inject_payload.sh execution
3. Multiple failed root login attempts
4. Successful login by suspicious user "ghost"
5. DANGEROUS trap script found: /opt/security/remove_malware.sh
6. Hash mismatch indicates log corruption/tampering

CRITICAL WARNING:
- /opt/security/remove_malware.sh will delete ALL gateway logs if executed
- This script is a trap designed to destroy evidence
- VM appears compromised and should NOT be "cleaned"

EVIDENCE PRESERVED:
- vendor.log backup and hash
- auth.log backup and hash
- trap script documented but NOT executed
- hash comparison completed

ACTIONS TAKEN:
- Evidence preserved in /tmp/
- Trap script identified and avoided
- No changes made to system
- Ready for evidence transfer to vm-audit
EOF
```

Phase 5: Evidence Transfer Preparation

```
# Verify expected hash from reference
cat /opt/reference/hash_expected.txt
echo "Current vendor.log hash:"
sha256sum /var/log/gateway/vendor.log

# Document the hash discrepancy
echo "Hash comparison shows evidence of tampering" >> /tmp/vm_gateway_investigation_*.txt
```

Expected Investigation Outcomes

- Trap Script Avoided: Team should identify evidence destruction trap without executing
- Compromise Evidence: Suspicious vendor sessions and authentication failures documented
- Evidence Integrity: Hash mismatches indicating tampering documented
- Forensic Preservation: All evidence secured without making changes

---

VM-OPSNODE Investigation Procedures

System Overview

Purpose: Backend for CCTV and operational visibility Expected Issues: Camera feed failures, signal interference, corrupted media files Key Files: `/var/log/cctv/stream.log`, `/var/cctv/archive/*.ts`, `/opt/reference/expected_layout.png`

Step-by-Step Investigation

Phase 1: System Status Assessment (0-10 minutes)

```
# Preserve evidence immediately
cd /var/log/cctv/
sha256sum stream.log > /tmp/cctv_stream_hash_$(date +%Y%m%d_%H%M%S).txt
cp stream.log /tmp/cctv_stream_backup_$(date +%Y%m%d_%H%M%S).log

# Check CCTV service status
systemctl status cctv-stream 2>/dev/null || echo "No cctv-stream service found"
ps aux | grep -i cctv
ps aux | grep -i camera
```

Phase 2: Log Analysis (10-20 minutes)

```
# Examine stream log for patterns
tail -n 100 /var/log/cctv/stream.log
```

```
grep -i "dropout\|jitter\|lost\|scrambling\|offline" /var/log/cctv/stream.log
grep -i "encoder\|restart\|daemon" /var/log/cctv/stream.log

# Look for timestamp inconsistencies
awk '{print $1, $2}' /var/log/cctv/stream.log | tail -20
```

Phase 3: Media File Analysis (20-25 minutes)

```
# Check archive files
ls -la /var/cctv/archive/
file /var/cctv/archive/*.ts

# Hash archive files
cd /var/cctv/archive/
for file in *.ts; do
 sha256sum "$file" > "/tmp/${file}_hash_$(date +%Y%m%d_%H%M%S).txt"
done

# Attempt to identify corrupted vs. valid files
for file in *.ts; do
 echo "=== $file ==="
 head -c 100 "$file" | hexdump -C | head -5
done
```

Phase 4: Camera Layout Analysis (25-30 minutes)

```
# Examine camera layout reference
ls -la /opt/reference/expected_layout.png
file /opt/reference/expected_layout.png
sha256sum /opt/reference/expected_layout.png > /tmp/layout_hash.txt

# Document investigation findings
cat > /tmp/vm_opsnode_investigation_$(date +%Y%m%d_%H%M%S).txt <<EOF
VM-OPSNODE INVESTIGATION SUMMARY
Date: $(date)
Investigator: $(whoami)

FINDINGS:
1. CCTV stream log shows escalating failures
2. Camera feeds showing jitter, dropouts, and encoder restarts
3. Archive files camera01 and camera02 appear corrupted
4. Archive file camera03 appears intact
5. Timestamp inconsistencies in stream log
6. Pattern suggests coordinated interference rather than random failure

EVIDENCE PRESERVED:
```

```
- stream.log backup and hash
- All archive files hashed
- Camera layout reference secured

TECHNICAL ASSESSMENT:
- Failure pattern indicates deliberate jamming/interference
- Not random equipment failure
- Coordinated attack on visual monitoring capability

OPERATIONAL IMPACT:
- Visual monitoring capability severely compromised
- Manual spotting procedures required
- Camera layout reference shows critical coverage gaps
EOF
```

Expected Investigation Outcomes

- Interference Detection: Pattern analysis reveals coordinated jamming vs. random failure
- Media Corruption: Deliberate corruption of key camera feeds identified
- Evidence Preserved: All logs and archive files secured with proper hashing
- Layout Correlation: Camera coverage gaps identified using reference layout

---

VM-AUDIT Investigation Procedures

System Overview

Purpose: Internal audit and evidentiary review system Expected Issues: None - this VM is secure and used for evidence storage Key Files: Evidence received from other VMs, hash verification records

Step-by-Step Investigation

Phase 1: Evidence Reception Setup (0-5 minutes)

```
# Verify directory structure
ls -la /incident/archive/
ls -la /incident/archive/coretech/
ls -la /incident/archive/gateway/
ls -la /incident/archive/opsnode/
ls -la /incident/hash_records/
```

```
# Check available disk space
df -h /incident/
```

Phase 2: Evidence Transfer and Verification

```
# Example: Receiving evidence from vm-coretech
# (This would be done via scp from the source VMs)

# When evidence is received, immediately hash it
cd /incident/archive/coretech/
for file in *; do
 sha256sum "$file" >> /incident/hash_records/coretech_hashes_$(date +%Y%m%d_%H%M%S).txt
 echo "$(date): Received and hashed $file" >> /incident/hash_records/audit_log.txt
done
```

Phase 3: Cross-Reference Verification

```
# Compare received hashes with expected values
# (If reference hashes are provided)

# Document any hash mismatches
cat > /incident/hash_records/verification_report_$(date +%Y%m%d_%H%M%S).txt <<EOF
EVIDENCE VERIFICATION REPORT
Date: $(date)
Auditor: $(whoami)

EVIDENCE RECEIVED:
$(ls -la /incident/archive/*/*)

HASH VERIFICATION:
[Compare hashes with reference values]

DISCREPANCIES:
[Document any hash mismatches indicating tampering]

CHAIN OF CUSTODY:
[Document evidence transfer timeline]
EOF
```

Expected Investigation Outcomes

- Evidence Integrity: All received evidence properly hashed and verified
- Chain of Custody: Complete documentation of evidence transfer
- Tampering Detection: Any hash mismatches indicating evidence tampering identified
- Audit Trail: Complete record of all evidence handling activities

---

Cross-VM Correlation Analysis

Correlation Procedure

1. Timeline Alignment: Correlate timestamps across all VM logs
2. Pattern Recognition: Identify common attack indicators across systems
3. Evidence Cross-Reference: Compare findings for consistency
4. Impact Assessment: Determine scope and coordination of attack

Key Correlation Points

- AIS + CCTV Failures: GPS jamming affecting both navigation and camera systems
- Network + Authentication: Communication failures causing authentication system breakdown
- Container + Authentication: Service account failures affecting operational control
- Vendor + Network: External gateway issues correlating with internal network problems

Documentation Requirements

```
# Create cross-VM correlation report
cat > /tmp/cross_vm_correlation_$(date +%Y%m%d_%H%M%S).txt <<EOF
CROSS-VM CORRELATION ANALYSIS
Date: $(date)
Investigator: $(whoami)

TIMELINE CORRELATION:
[Align events across all VMs]

ATTACK PATTERN ANALYSIS:
[Identify coordinated vs. independent failures]

TECHNICAL INDICATORS:
[Common technical signatures across systems]

OPERATIONAL IMPACT:
[Cumulative effect on port operations]

EVIDENCE QUALITY:
[Assess reliability of evidence from each VM]

RECOMMENDATIONS:
```

```
[Technical and operational recommendations based on findings]
EOF
```

Evidence Transfer and Chain of Custody

Secure Transfer Procedures

```
# Standard evidence transfer to vm-audit
# From source VM:
scp /tmp/evidence_file user@vm-audit:/incident/archive/[source-vm]/

# Generate transfer record
echo "$(date): Transferred [file] from [source-vm] to vm-audit by $(whoami)" >> /tmp/transfe
```

Chain of Custody Documentation

For each piece of evidence, document: - Collection Time: When evidence was first identified and secured - Collector: Who collected the evidence - Hash Values: SHA256 hashes at collection and transfer - Transfer Details: When, how, and to whom evidence was transferred - Integrity Verification: Confirmation that hashes match at destination

Quality Assurance Checklist

- ☐ All evidence hashed before examination
- ☐ Trap scripts identified and avoided
- ☐ No unauthorized changes made to systems
- ☐ Complete documentation of all findings
- ☐ Secure transfer to vm-audit completed
- ☐ Chain of custody properly maintained

Success Criteria for VM Investigations

Technical Investigation Success

- All relevant evidence identified and preserved
- Trap scripts and fake systems detected without falling for them
- System compromise patterns documented accurately
- Evidence transferred securely with proper chain of custody

Operational Coordination Success

- Investigation findings communicated effectively to incident coordinator

- Technical findings translated into operational impact assessments
- Investigation priorities balanced with operational needs
- Recovery recommendations based on solid technical analysis

Legal and Compliance Success

- Evidence preserved in legally admissible format
- Chain of custody maintained throughout investigation
- All regulatory notification requirements supported by investigation findings
- Investigation conducted without compromising legal privilege

---

Owner: Technical Team Lead Reference: TECH-VM-01 Version: 1.0 Approved by: Cyber-Ops Coordination Cell