

SOUTHGATE TERMINAL

Port Operations Security Documentation

Evidence Transfer and Chain-of-Custody Procedures

Document Information

Document Type: Forensic Evidence Management Framework

Intended Users: Technical Team, Legal Team, Incident Coordinator

Usage Context: Evidence collection, preservation, and transfer during cyber incidents

Related Scenarios: VM investigations, log analysis, forensic chain-of-custody

Purpose

This document establishes formal procedures for collecting, preserving, transferring, and maintaining chain-of-custody for digital evidence during cybersecurity incidents, ensuring legal admissibility and investigative integrity.

When to Use These Procedures

- Suspected security breaches requiring evidence preservation
 - System anomalies needing forensic investigation
 - Preparation for potential legal proceedings
 - Regulatory compliance requiring evidence documentation
 - Cross-system investigations requiring evidence correlation
-

Legal Framework and Requirements

Chain-of-Custody Definition

Chain-of-custody is the chronological documentation that records the sequence of custody, control, transfer, analysis, and disposition of evidence. It establishes:

- **WHO** had possession of evidence
- **WHAT** was done with the evidence
- **WHEN** actions were performed
- **WHERE** evidence was located
- **WHY** evidence was collected/transferred

Legal Admissibility Requirements

For digital evidence to be legally admissible:

- **Authenticity:** Evidence must be shown to be what it purports to be
- **Reliability:** Evidence collection methods must be sound and scientifically accepted
- **Completeness:** Evidence must not

be altered or selectively edited - **Chain-of-Custody:** Continuous documentation of evidence handling

Evidence Integrity Protection

- **Hash Verification:** SHA256 hashes generated before and after each transfer
 - **Write Protection:** Evidence stored in read-only format when possible
 - **Access Logging:** All access to evidence documented with timestamps
 - **Storage Security:** Evidence stored in secure, access-controlled environment
-

Evidence Collection Standards

Digital Evidence Categories

PRIMARY EVIDENCE (Direct incident artifacts)

- **System Logs:** Authentication, system, application, and security logs
- **Configuration Files:** System configurations at time of incident
- **Memory Dumps:** System memory snapshots (if captured)
- **Network Captures:** Network traffic logs and packet captures
- **File System Artifacts:** Suspicious files, scripts, or modified system files

SECONDARY EVIDENCE (Supporting documentation)

- **Screenshots:** Visual evidence of system states or anomalies
- **Documentation:** Investigation notes, timelines, and analysis
- **Communication Records:** Internal communications about incident
- **Procedure Records:** Documentation of investigation steps taken

METADATA EVIDENCE (Evidence about evidence)

- **Hash Values:** Cryptographic hashes of all evidence files
- **Timestamps:** Collection, modification, and access times
- **File Attributes:** Permissions, ownership, and size information
- **Collection Methods:** Tools and procedures used for evidence collection

Evidence Collection Procedures

Phase 1: Evidence Identification (0-10 minutes)

```
# Document system state immediately
cat > /tmp/evidence_collection_start_$(date +%Y%m%d_%H%M%S).txt <<EOF
EVIDENCE COLLECTION INITIATED
Date/Time: $(date)
Collector: $(whoami)
```

```
System: $(hostname)
Incident ID: [INCIDENT_ID]
```

```
INITIAL SYSTEM STATE:
Uptime: $(uptime)
Current Users: $(who)
Running Processes: $(ps aux | wc -l) processes
Disk Usage: $(df -h / | tail -1)
Network Connections: $(netstat -an | grep ESTABLISHED | wc -l) active
```

```
EVIDENCE TARGETS IDENTIFIED:
- Log files requiring preservation
- Configuration files of interest
- Suspicious files or scripts
- System state information
EOF
```

Phase 2: Evidence Preservation (10-30 minutes)

```
# Create evidence collection directory structure
mkdir -p /tmp/evidence_collection_$(date +%Y%m%d_%H%M%S)/{logs,configs,artifacts,hashes,meta}
EVIDENCE_DIR="/tmp/evidence_collection_$(date +%Y%m%d_%H%M%S)"

# Preserve logs with hash generation
for logfile in /var/log/auth.log /var/log/syslog /var/log/kern.log; do
    if [ -f "$logfile" ]; then
        # Generate hash before copying
        sha256sum "$logfile" >> "$EVIDENCE_DIR/ hashes/pre_collection_hashes.txt"

        # Copy with timestamp preservation
        cp -p "$logfile" "$EVIDENCE_DIR/logs/$(basename $logfile)_$(date +%Y%m%d_%H%M%S)"

        # Generate hash after copying
        sha256sum "$EVIDENCE_DIR/logs/$(basename $logfile)_$(date +%Y%m%d_%H%M%S)" >> "$EVIDENCE_DIR/ hashes/post_collection_hashes.txt"

        # Document collection
        echo "$(date): Collected $logfile" >> "$EVIDENCE_DIR/metadata/collection_log.txt"
    fi
done
```

Phase 3: Evidence Documentation (30-45 minutes)

```
# Create comprehensive evidence manifest
cat > "$EVIDENCE_DIR/EVIDENCE_MANIFEST.txt" <<EOF
DIGITAL EVIDENCE MANIFEST
=====
```

```
INCIDENT INFORMATION:
Incident ID: [INCIDENT_ID]
Collection Date: $(date)
Collector: $(whoami)
System: $(hostname)
Collection Tool: Manual collection with standard Linux tools
```

```
EVIDENCE COLLECTED:
EOF
```

```
# Document each piece of evidence
find "$EVIDENCE_DIR" -type f -not -path "*/EVIDENCE_MANIFEST.txt" | while read file; do
    echo "File: $(basename $file)" >> "$EVIDENCE_DIR/EVIDENCE_MANIFEST.txt"
    echo "Path: $file" >> "$EVIDENCE_DIR/EVIDENCE_MANIFEST.txt"
    echo "Size: $(stat -c%s $file) bytes" >> "$EVIDENCE_DIR/EVIDENCE_MANIFEST.txt"
    echo "Hash: $(sha256sum $file | cut -d' ' -f1)" >> "$EVIDENCE_DIR/EVIDENCE_MANIFEST.txt"
    echo "Collected: $(stat -c%y $file)" >> "$EVIDENCE_DIR/EVIDENCE_MANIFEST.txt"
    echo "---" >> "$EVIDENCE_DIR/EVIDENCE_MANIFEST.txt"
done
```

```
# Create chain-of-custody form
cat > "$EVIDENCE_DIR/CHAIN_OF_CUSTODY.txt" <<EOF
CHAIN OF CUSTODY RECORD
=====
```

```
CASE INFORMATION:
Case Number: [INCIDENT_ID]
Incident Type: Cybersecurity Incident
Location: $(hostname)
Date/Time: $(date)
```

```
EVIDENCE DESCRIPTION:
Digital evidence collected from $(hostname) including system logs, configuration files, and
```

```
CHAIN OF CUSTODY:
Date/Time: $(date)
Collected By: $(whoami)
Purpose: Cybersecurity incident investigation
Method: Standard digital forensic collection procedures
Location: $(hostname)
Witness: [TO_BE_FILLED]
Signature: [TO_BE_FILLED]
```

```
TRANSFER LOG:
[Transfers to be documented below]
```

EOF

Evidence Transfer Procedures

VM-to-VM-Audit Transfer Protocol

Secure Transfer Preparation

```
# Prepare evidence package for transfer
EVIDENCE_PACKAGE="evidence_$(hostname)_$(date +%Y%m%d_%H%M%S).tar.gz"

# Create compressed evidence package
cd /tmp/
tar -czf "$EVIDENCE_PACKAGE" evidence_collection_*

# Generate final package hash
sha256sum "$EVIDENCE_PACKAGE" > "${EVIDENCE_PACKAGE}.hash"

# Document transfer preparation
cat > "${EVIDENCE_PACKAGE}.transfer_prep.txt" <<EOF
EVIDENCE TRANSFER PREPARATION
=====

Package: $EVIDENCE_PACKAGE
Created: $(date)
Source System: $(hostname)
Prepared By: $(whoami)
Destination: vm-audit
Transfer Method: Secure Copy (scp)

PACKAGE CONTENTS:
$(tar -tzf "$EVIDENCE_PACKAGE" | head -20)
[... and $(tar -tzf "$EVIDENCE_PACKAGE" | wc -l) total files]

INTEGRITY VERIFICATION:
Package Hash: $(cat "${EVIDENCE_PACKAGE}.hash")
EOF
```

Secure Transfer Execution

```
# Execute secure transfer to vm-audit
SOURCE_SYSTEM=$(hostname)
TRANSFER_ID="transfer_$(date +%Y%m%d_%H%M%S)"

# Transfer evidence package
scp "$EVIDENCE_PACKAGE" "audituser@vm-audit:/incident/archive/$SOURCE_SYSTEM/"
```

```

scp "${EVIDENCE_PACKAGE}.hash" "audituser@vm-audit:/incident/archive/$SOURCE_SYSTEM/"
scp "${EVIDENCE_PACKAGE}.transfer_prep.txt" "audituser@vm-audit:/incident/archive/$SOURCE_SYSTEM/"

# Verify successful transfer
ssh audituser@vm-audit "sha256sum /incident/archive/$SOURCE_SYSTEM/$EVIDENCE_PACKAGE" > /tmp/transfer_log.txt

# Compare hashes to verify integrity
LOCAL_HASH=$(cat "${EVIDENCE_PACKAGE}.hash" | cut -d' ' -f1)
REMOTE_HASH=$(cat /tmp/remote_hash_verification.txt | cut -d' ' -f1)

if [ "$LOCAL_HASH" = "$REMOTE_HASH" ]; then
    echo "$(date): TRANSFER SUCCESSFUL - Hash verification passed" >> /tmp/transfer_log.txt
    TRANSFER_STATUS="SUCCESS"
else
    echo "$(date): TRANSFER FAILED - Hash mismatch detected" >> /tmp/transfer_log.txt
    TRANSFER_STATUS="FAILED"
fi

# Update chain-of-custody record
cat >> "${EVIDENCE_PACKAGE}.transfer_log.txt" <<EOF

TRANSFER RECORD:
Transfer ID: $TRANSFER_ID
Date/Time: $(date)
From: $(whoami)@$(hostname)
To: audituser@vm-audit
Method: Secure Copy (scp)
Status: $TRANSFER_STATUS
Local Hash: $LOCAL_HASH
Remote Hash: $REMOTE_HASH
Verification: $([ "$LOCAL_HASH" = "$REMOTE_HASH" ] && echo "PASSED" || echo "FAILED")
EOF

```

VM-Audit Reception Procedures

Evidence Reception and Verification

```

# On vm-audit system - receive and verify evidence
EVIDENCE_DIR="/incident/archive"
RECEIVED_PACKAGE="$1" # Package filename passed as parameter
SOURCE_SYSTEM="$2"    # Source system name

# Create reception record
cat > "/incident/hash_records/reception_$(date +%Y%m%d_%H%M%S).txt" <<EOF
EVIDENCE RECEPTION RECORD
=====

```

```

Reception Date/Time: $(date)
Received By: $(whoami)
Source System: $SOURCE_SYSTEM
Package: $RECEIVED_PACKAGE

INTEGRITY VERIFICATION:
EOF

# Verify package integrity
cd "$EVIDENCE_DIR/$SOURCE_SYSTEM"
LOCAL_HASH=$(sha256sum "$RECEIVED_PACKAGE" | cut -d' ' -f1)
EXPECTED_HASH=$(cat "${RECEIVED_PACKAGE}.hash" | cut -d' ' -f1)

echo "Calculated Hash: $LOCAL_HASH" >> "/incident/hash_records/reception_$(date +%Y%m%d_%H%M%S).txt"
echo "Expected Hash: $EXPECTED_HASH" >> "/incident/hash_records/reception_$(date +%Y%m%d_%H%M%S).txt"

if [ "$LOCAL_HASH" = "$EXPECTED_HASH" ]; then
    echo "Verification: PASSED" >> "/incident/hash_records/reception_$(date +%Y%m%d_%H%M%S).txt"
    echo "Status: EVIDENCE RECEIVED AND VERIFIED" >> "/incident/hash_records/reception_$(date +%Y%m%d_%H%M%S).txt"

    # Extract evidence for analysis
    tar -xzf "$RECEIVED_PACKAGE"

    # Generate hashes for all extracted files
    find . -name "evidence_collection_*" -type f | while read file; do
        sha256sum "$file" >> "/incident/hash_records/extracted_file_hashes_$(date +%Y%m%d_%H%M%S).txt"
    done

else
    echo "Verification: FAILED" >> "/incident/hash_records/reception_$(date +%Y%m%d_%H%M%S).txt"
    echo "Status: EVIDENCE INTEGRITY COMPROMISED" >> "/incident/hash_records/reception_$(date +%Y%m%d_%H%M%S).txt"

    # Alert incident coordinator
    echo "CRITICAL: Evidence integrity failure for package $RECEIVED_PACKAGE from $SOURCE_SYSTEM" >> "/incident/alerts/critical_evidence_integrity_failure.txt"
fi

```

Cross-System Evidence Correlation

Evidence Correlation Framework

Timeline Correlation Procedures

```

# Create cross-system timeline correlation
cat > "/incident/analysis/timeline_correlation_$(date +%Y%m%d_%H%M%S).txt" <<EOF

```

CROSS-SYSTEM EVIDENCE CORRELATION ANALYSIS

=====

Analysis Date: \$(date)
Analyst: \$(whoami)
Incident ID: [INCIDENT_ID]

EVIDENCE SOURCES:

```
$(find /incident/archive -name "EVIDENCE_MANIFEST.txt" -exec echo "Source: {}" \; -exec head -n 1 {} \;)
```

TIMELINE ANALYSIS:

EOF

Extract timestamps from all evidence sources

```
find /incident/archive -name "*.log" -exec grep -H "$(date +%Y-%m-%d)" {} \; | sort > "/tmp/timeline_raw.txt"
```

Create structured timeline

```
awk -F: '{print $3:"$4":"$5, $1}' /tmp/timeline_raw.txt | sort >> "/incident/analysis/timeline.txt"
```

Evidence Cross-Reference Matrix

Create evidence cross-reference analysis

```
cat > "/incident/analysis/cross_reference_$(date +%Y%m%d_%H%M%S).txt" <<EOF
```

EVIDENCE CROSS-REFERENCE ANALYSIS

=====

AUTHENTICATION FAILURES:

```
$(find /incident/archive -name "*auth*" -exec grep -H "authentication failure" {} \;)
```

SYSTEM ANOMALIES:

```
$(find /incident/archive -name "*syslog*" -exec grep -H -i "error\|warning\|failed" {} \; | sort)
```

NETWORK CORRELATIONS:

```
$(find /incident/archive -name "*log*" -exec grep -H -E "[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+" {} \; | sort)
```

TEMPORAL CORRELATIONS:

[Analysis of events occurring at similar times across systems]

EOF

Evidence Quality Assessment

Evidence Completeness Check

Assess evidence completeness across all systems

```
cat > "/incident/analysis/evidence_completeness_$(date +%Y%m%d_%H%M%S).txt" <<EOF
```

EVIDENCE COMPLETENESS ASSESSMENT

=====

EXPECTED EVIDENCE SOURCES:

- vm-coretech: AIS logs, GPS data, system logs
- vm-gateway: Vendor logs, authentication logs, access logs
- vm-opsnode: CCTV logs, stream data, archive files
- vm-audit: Reception records, hash verifications

RECEIVED EVIDENCE:

```
$(find /incident/archive -type f -name "*.log" | wc -l) log files received
$(find /incident/archive -type f -name "*.hash" | wc -l) hash files received
$(find /incident/archive -type f -name "*manifest*" | wc -l) manifest files received
```

EVIDENCE GAPS IDENTIFIED:

EOF

Check for missing expected evidence

EXPECTED_SYSTEMS=("vm-coretech" "vm-gateway" "vm-opsnode")

for system in "\${EXPECTED_SYSTEMS[@]}; do

if [! -d "/incident/archive/\$system"]; then

echo "MISSING: No evidence received from \$system" >> "/incident/analysis/evidence_co

else

echo "RECEIVED: Evidence package from \$system" >> "/incident/analysis/evidence_comp

fi

done

Chain-of-Custody Documentation

Formal Chain-of-Custody Record

Chain-of-Custody Form Template

CHAIN OF CUSTODY RECORD

CASE: [INCIDENT_ID] | PAGE: ___ of ___

EVIDENCE DESCRIPTION:

Item #: _____

Description: _____

Source Location: _____

Collection Date/Time: _____

COLLECTED BY:

Name: _____ Badge/ID: _____

Agency: _____ Phone: _____

Signature: _____ Date/Time: _____

EVIDENCE SEALED/INITIALED: _____ WITNESS: _____

RECEIVED BY:

Name: _____ Badge/ID: _____
Agency: _____ Phone: _____
Purpose: _____
Signature: _____ Date/Time: _____

RELEASED BY:

Name: _____ Badge/ID: _____
Agency: _____ Phone: _____
Reason: _____
Signature: _____ Date/Time: _____

[Additional transfer entries...]

Digital Chain-of-Custody Automation

Automate chain-of-custody record creation

```
create_custody_record() {  
    local evidence_file="$1"  
    local custodian="$2"  
    local purpose="$3"  
  
    cat > "/incident/custody/custody_$(basename $evidence_file)_$(date +%Y%m%d_%H%M%S).txt"  
DIGITAL CHAIN OF CUSTODY RECORD  
=====
```

EVIDENCE DETAILS:

File: \$(basename \$evidence_file)
Full Path: \$evidence_file
Hash: \$(sha256sum \$evidence_file | cut -d' ' -f1)
Size: \$(stat -c%s \$evidence_file) bytes
Created: \$(stat -c%y \$evidence_file)

CUSTODY TRANSFER:

Date/Time: \$(date)
From: \$(whoami)@\$(hostname)
To: \$custodian
Purpose: \$purpose
Method: Digital transfer with hash verification

INTEGRITY VERIFICATION:

Original Hash: [TO_BE_VERIFIED]
Transfer Hash: \$(sha256sum \$evidence_file | cut -d' ' -f1)
Verification: [TO_BE_COMPLETED]

CUSTODIAN CERTIFICATION:

I hereby acknowledge receipt of the above-described evidence and certify that it was received

```
Custodian: $custodian
Date/Time: $(date)
Signature: [DIGITAL_SIGNATURE_REQUIRED]
EOF
}
```

Audit Trail Maintenance

Evidence Access Logging

```
# Log all access to evidence files
log_evidence_access() {
    local evidence_file="$1"
    local access_type="$2" # READ, WRITE, TRANSFER, ANALYSIS
    local purpose="$3"

    echo "$(date)|$(whoami)|$(hostname)|$access_type|$(basename $evidence_file)|$purpose" >> /incident/audit/evidence_access_log.txt
}

# Example usage:
# log_evidence_access "/incident/archive/vm-coretech/auth.log" "READ" "Timeline analysis"
# log_evidence_access "/incident/archive/vm-gateway/vendor.log" "ANALYSIS" "Malware investigation"
```

Periodic Integrity Verification

```
# Verify evidence integrity periodically
verify_evidence_integrity() {
    echo "EVIDENCE INTEGRITY CHECK - $(date)" >> "/incident/audit/integrity_check_log.txt"

    find /incident/archive -name "*.hash" | while read hashfile; do
        evidence_file="${hashfile%.hash}"
        if [ -f "$evidence_file" ]; then
            expected_hash=$(cat "$hashfile" | cut -d' ' -f1)
            current_hash=$(sha256sum "$evidence_file" | cut -d' ' -f1)

            if [ "$expected_hash" = "$current_hash" ]; then
                echo "PASS: $(basename $evidence_file)" >> "/incident/audit/integrity_check_log.txt"
            else
                echo "FAIL: $(basename $evidence_file) - INTEGRITY COMPROMISED" >> "/incident/audit/integrity_check_log.txt"
                # Alert incident coordinator immediately
                echo "CRITICAL: Evidence integrity failure detected for $evidence_file" | mail -s "Evidence Integrity Failure" coordinator@incident.com
            fi
        fi
    done
}
```

```

    done
}

# Schedule periodic integrity checks
echo "0 */6 * * * /usr/local/bin/verify_evidence_integrity" >> /etc/cron.d/evidence-integrity

```

Legal Compliance Requirements

Regulatory Evidence Standards

Evidence Retention Requirements

- **Cybersecurity Incidents:** Minimum 7 years retention
- **Financial Impact:** Extended retention per regulatory requirements
- **Legal Proceedings:** Litigation hold until case resolution
- **Insurance Claims:** Retention until claim resolution + 2 years

Evidence Disclosure Obligations

```

# Prepare evidence for legal disclosure
prepare_legal_disclosure() {
    local case_number="$1"
    local requesting_party="$2"

    mkdir -p "/incident/legal_disclosure/$case_number"

    # Create disclosure package with proper legal formatting
    cat > "/incident/legal_disclosure/$case_number/DISCLOSURE_PACKAGE.txt" <<EOF
LEGAL EVIDENCE DISCLOSURE PACKAGE
=====

Case Number: $case_number
Requesting Party: $requesting_party
Disclosure Date: $(date)
Prepared By: $(whoami)

EVIDENCE INCLUDED:
[List of evidence files being disclosed]

CHAIN OF CUSTODY:
[Complete chain-of-custody documentation]

INTEGRITY VERIFICATION:
[Hash verification records]

LEGAL CERTIFICATION:

```

I hereby certify that the evidence included in this disclosure package represents a true and

Prepared By: \$(whoami)
Title: [TITLE]
Date: \$(date)
Signature: [REQUIRED]
EOF
}

Privacy and Privilege Protection

Evidence Sanitization Procedures

```
# Remove privileged or private information before disclosure
sanitize_evidence() {
    local input_file="$1"
    local output_file="$2"

    # Remove attorney-client privileged communications
    grep -v -i "attorney\|lawyer\|counsel\|privileged" "$input_file" > "$output_file.temp1"

    # Remove personal identifying information (basic pattern matching)
    sed -E 's/[0-9]{3}-[0-9]{2}-[0-9]{4}/[SSN_REDACTED]/g' "$output_file.temp1" > "$output_file.temp2"
    sed -E 's/[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}/[EMAIL_REDACTED]/g' "$output_file.temp2" > "$output_file"

    # Document sanitization
    echo "$(date): Sanitized $input_file -> $output_file" >> "/incident/audit/sanitization_log.txt"

    # Clean up temporary files
    rm "$output_file.temp1" "$output_file.temp2"

    # Generate hash of sanitized file
    sha256sum "$output_file" > "$output_file.hash"
}
```

Quality Assurance and Validation

Evidence Collection Validation Checklist

Pre-Transfer Validation

- ☐ **Hash Generation:** SHA256 hash generated for all evidence files
- ☐ **Timestamp Verification:** File timestamps preserved and documented
- ☐ **Completeness Check:** All relevant evidence identified and collected
- ☐ **Chain-of-Custody:** Initial custody record completed
- ☐ **Legal Review:** Evidence collection methods legally sound

Transfer Validation

- ☐ **Secure Transfer:** Evidence transferred using secure, authenticated method
- ☐ **Integrity Verification:** Hash verification completed post-transfer
- ☐ **Receipt Confirmation:** Transfer receipt confirmed by receiving party
- ☐ **Documentation Update:** Chain-of-custody updated with transfer details
- ☐ **Access Control:** Evidence secured with appropriate access controls

Post-Transfer Validation

- ☐ **Storage Security:** Evidence stored in secure, controlled environment
- ☐ **Access Logging:** All evidence access properly logged
- ☐ **Periodic Verification:** Regular integrity checks scheduled
- ☐ **Retention Compliance:** Evidence retention schedule established
- ☐ **Disposal Planning:** Secure disposal procedures planned for end-of-retention

Common Evidence Handling Errors to Avoid

CRITICAL ERRORS:

- **No Hash Generation:** Failing to generate cryptographic hashes
- **Chain-of-Custody Gaps:** Missing or incomplete custody documentation
- **Integrity Compromise:** Evidence modified after collection
- **Insecure Transfer:** Evidence transferred without encryption or verification
- **Access Control Failure:** Unauthorized access to evidence

BEST PRACTICES:

- **Immediate Hashing:** Generate hashes before any analysis or transfer
- **Continuous Documentation:** Document every action taken with evidence
- **Secure Storage:** Use encrypted, access-controlled storage systems
- **Regular Verification:** Perform periodic integrity checks
- **Legal Consultation:** Involve legal team in evidence handling decisions

Success Criteria for Evidence Management

Technical Success Criteria

- All evidence collected with proper cryptographic hash verification
- Chain-of-custody documentation complete and unbroken
- Evidence transferred securely with verified integrity
- No unauthorized modification or access to evidence
- All evidence properly correlated across systems

Legal Success Criteria

- Evidence collection methods legally defensible
- Chain-of-custody meets legal admissibility standards
- Privacy and privilege protection maintained throughout
- Regulatory compliance requirements satisfied
- Evidence ready for legal proceedings if required

Operational Success Criteria

- Evidence collection completed without disrupting ongoing operations
- Technical investigation supported by quality evidence
- Cross-team coordination effective throughout evidence handling
- Documentation sufficient for comprehensive incident analysis
- Lessons learned captured for future evidence handling improvement

Owner: Technical Team Lead / Legal Counsel

Reference: TECH-EVIDENCE-01

Version: 1.0

Approved by: Cyber-Ops Coordination Cell