Security & Compliance Evidence Index

HashInsight SOC 2 / PCI DSS / GDPR - Control Matrix & Audit Trail

Version: 1.0
Date: October 2025
Document Type: Compliance Evidence Index
Audit Readiness Status: SOC 2 Type II Ready | PCI DSS In Progress | GDPR Compliant

---

Executive Summary

This document provides a **complete index of security controls, evidence locations, and audit procedures** for HashInsight's enterprise platform. It serves as the primary reference for: - SOC 2 Type II auditors - PCI DSS assessors - GDPR compliance officers - Internal security audits

Document Structure: 1. Control Objectives & Evidence Map 2. Security Implementation Details 3. Audit Procedures & Verification 4. Compliance Status Dashboard 5. Evidence Repository Locations

---

1. SOC 2 Type II Control Matrix

1.1 Trust Service Criteria Mapping

TSC	Control Objective	Implementation	Evidence Location	Audit Frequency	Status
CC6.1	Logical Access - Authentication	mTLS + API Keys + Session Management	`/security/mtls/`	Quarterly	Ready
CC6.2	Logical Access - Authorization	Role-Based Access Control (RBAC)	`/models.py` (User roles)	Quarterly	Ready
CC6.6	Logical Access - Encryption at Rest	KMS-based AES-256 encryption	`/security/kms_client.py`	Quarterly	Ready
CC6.7	Logical Access - Encryption in Transit	TLS 1.3, mTLS	Nginx config, SSL Labs report	Quarterly	Ready
CC7.2	System Monitoring - Logging	Comprehensive audit logs	`/monitoring/audit_logger.py`	Monthly	Ready
CC7.3	System Monitoring - Anomaly Detection	Automated alerts (Prometheus)	`/monitoring/prometheus_exporter.py`	Monthly	Ready
A1.2	Availability - SLO Monitoring	99.95% SLA tracking	`/monitoring/slo_manager.py`	Weekly	Ready
A1.3	Availability - Backup & Recovery	Automated backups (RTO≤4h, RPO≤15m)	`/backup/backup_manager.py`	Daily	Ready
PI1.4	Processing Integrity - Data Validation	Dual-algorithm validation	`/mining_calculator.py`	Quarterly	Ready

TSC	Control Objective	Implementation	Evidence Location	Audit Frequency	Status
PI1.5	Processing Integrity - Error Handling	Circuit breakers, retry logic	`/cache_manager.py`	Quarterly	Ready
C1.1	Confidentiality - Data Classification	Sensitive data tagging	`config.py` SECURITY_CLASSIFICATIONS	Semi-annual	Ready
C1.2	Confidentiality - Key Management	KMS with 90-day rotation	`/security/kms_client.py`	Quarterly	Ready

1.2 Security Control Evidence

CC6.1: Authentication Controls

Control Description: Multi-factor authentication and secure session management

Evidence Checklist: - [x] **Code**: `security/mtls/cert_validator.py` - mTLS implementation - [x] **Code**: `models.py` lines 45-78 - User authentication model - [x] **Config**: `config.py` SESSION_CONFIG - Session timeout (30 min) - [x] **Logs**: `/var/log/hashinsight/auth.log` - Authentication events - [x] **Test**: `tests/security/test_mtls.py` - mTLS unit tests - [x] **Documentation**: `OPERATIONS_MANUAL_EN.md` Chapter 6 - Security procedures

Audit Procedure:

```
# 1. Verify mTLS is enforced on production endpoints
curl -X GET https://api.hashinsight.io/admin/users \
  --cert client.crt --key client.key

# 2. Check session timeout configuration
grep SESSION_TIMEOUT config.py
# Expected: 1800 (30 minutes)

# 3. Review authentication logs for anomalies
tail -100 /var/log/hashinsight/auth.log | grep "FAILED_LOGIN"
```

Evidence Retention: 7 years (per SOC 2 requirements)

CC6.6: Encryption at Rest (KMS)

Control Description: All sensitive data encrypted using KMS-managed keys

Evidence Checklist: - [x] **Code**: `security/kms_client.py` - KMS implementation (AWS/GCP/Azure) - [x] **Code**: `models.py` - Encrypted fields (password_hash, api_keys) - [x] **Config**: Environment variables - KMS_KEY_ID, AWS_KMS_REGION - [x] **Policy**: `docs/data_encryption_policy.md` - Encryption standards - [x] **Audit Log**: KMS API calls logged to CloudTrail/GCP Audit Logs - [x] **Certificate**: Encryption algorithm = AES-256-GCM

Audit Procedure:

```
# Verify KMS encryption is active
from security.kms_client import KMSClient, KMSProvider

kms = KMSClient(KMSProvider.AWS_KMS, config)
test_data = "sensitive_data"
encrypted = kms.encrypt_secret(test_data, key_id=KMS_KEY_ID)

# Confirm encrypted data is not plaintext
assert test_data not in str(encrypted)
assert len(encrypted) > len(test_data)  # Ciphertext overhead

# Check key rotation status
key_metadata = aws_kms.describe_key(KeyId=KMS_KEY_ID)
assert key_metadata['KeyMetadata']['Enabled'] == True
```

Key Rotation Schedule: - **Frequency**: Every 90 days (automated) - **Last Rotation**: Check `/var/log/kms_rotation.log` - **Next Rotation**: `cron` job scheduled

---

CC7.2: Audit Logging

Control Description: Comprehensive logging of all security-relevant events

Evidence Checklist: - [x] **Code**: `monitoring/audit_logger.py` - Centralized audit logging - [x] **Format**: JSON Lines (JSONL) for structured logs - [x] **Storage**: Immutable S3/GCS bucket with versioning - [x] **Retention**: 7 years (compliance requirement) - [x] **Log Types**: Authentication, Authorization, Data Access, Admin Actions, API Calls - [x] **Monitoring**: Splunk/ELK dashboard for real-time analysis

Logged Events (Examples):

```
{
  "timestamp": "2025-10-03T14:23:45Z",
  "event_type": "USER_LOGIN",
  "user_id": 1234,
  "ip_address": "203.0.113.42",
  "user_agent": "Mozilla/5.0...",
  "status": "SUCCESS",
  "mfa_verified": true
}

{
  "timestamp": "2025-10-03T14:25:12Z",
  "event_type": "API_KEY_CREATED",
  "admin_user_id": 5,
  "target_user_id": 1234,
  "api_key_id": "hsi_prod_key_a7b3c9",
  "permissions": ["read:miners", "write:calculations"]
}

{
  "timestamp": "2025-10-03T14:30:00Z",
  "event_type": "DATA_EXPORT",
  "user_id": 1234,
  "resource": "customer_mining_data",
  "record_count": 5000,
  "export_format": "CSV",
  "compliance_notice": "GDPR_DATA_PORTABILITY"
}
```

Audit Procedure:

```
# 1. Verify audit logs are being written
ls -lh /var/log/hashinsight/audit/*.jsonl
# Expect: Daily rotated files

# 2. Check log integrity (tamper-proof)
sha256sum /var/log/hashinsight/audit/2025-10-03.jsonl
# Compare with S3 stored hash

# 3. Query for sensitive operations (last 24h)
cat audit.jsonl | jq 'select(.event_type == "ADMIN_ACTION" or .event_type == "DATA_DELETION")'
```

1.3 Availability Controls (SLA 99.95%)

A1.2: SLO Monitoring

Control Description: Real-time SLA tracking with error budget management

Evidence Checklist: - [x] **Code**: `monitoring/slo_manager.py` - SLO calculation engine - [x] **Metrics**: Prometheus `/metrics` endpoint - [x] **Dashboard**: Grafana SLO dashboard (`https://grafana.hashinsight.io/slo`) - [x] **Alerts**: PagerDuty integration for SLO violations - [x] **Historical Data**: 12 months of SLO metrics in InfluxDB - [x] **Reports**: Monthly SLA reports sent to customers

SLO Definitions:

```
slo_targets:
  availability:
    target: 99.95%
    measurement_window: 30 days
    error_budget: 21.6 minutes/month

  api_latency:
    target_p95: 250ms
    target_p99: 500ms
    measurement_window: 7 days

  error_rate:
    target: 0.1%
    measurement_window: 24 hours
```

Audit Procedure:

```
# 1. Check current SLO status
curl https://api.hashinsight.io/metrics | grep slo_availability
# slo_availability_percent 99.97

# 2. Calculate error budget remaining
python3 << EOF
import sys
sys.path.append('/opt/hashinsight')
from monitoring.slo_manager import SLOManager

slo = SLOManager()
status = slo.get_current_status()
print(f"Uptime: {status['uptime_percent']}%")
print(f"Error Budget Remaining: {status['error_budget_minutes']} minutes")
```

```
EOF

# 3. Review incident history
SELECT incident_date, downtime_minutes, root_cause
FROM slo_incidents
WHERE incident_date >= NOW() - INTERVAL '90 days';
```

---

2. PCI DSS Compliance (If Applicable)

2.1 PCI DSS Scope Definition

Important Note: HashInsight **does not process, store, or transmit credit card data** directly. Payment processing is handled by: - **Stripe** (PCI DSS Level 1 Service Provider) - **Chargebee** (Subscription billing)

Our Scope: - **Secure transmission** to payment provider (TLS 1.3) - **Session security** (no card data in logs) - **Access control** to payment admin panel - **NOT in scope**: Card data storage/processing (delegated to Stripe)

2.2 PCI DSS Control Evidence

Requirement	Control	Evidence	Status
Req 2.2	Secure Configuration	Nginx hardening checklist	
Req 4.1	Encryption in Transit	TLS 1.3 (SSL Labs A+ rating)	
Req 6.5	Secure Coding	OWASP Top 10 mitigation	
Req 8.3	Multi-Factor Auth	mTLS for admin access	
Req 10.1	Audit Trails	Payment event logging	

Evidence Location: - SSL/TLS Certificate: `/etc/nginx/ssl/hashinsight.crt` - Nginx Config: `/etc/nginx/sites-enabled/hashinsight.conf` - Payment Logs: `/var/log/hashinsight/payments.log` (no card data, only transaction IDs)

---

3. GDPR Compliance

3.1 Data Privacy Controls

GDPR Article 32: Security of Processing

Control Description: Technical and organizational measures to ensure data security

Evidence Checklist: - [x] **Encryption**: AES-256 (at rest), TLS 1.3 (in transit) - [x] **Pseudonymization**: User IDs hashed for analytics - [x] **Access Control**: Role-based permissions matrix - [x] **Data Minimization**: Only collect essential mining data - [x] **Audit Logging**: All personal data access logged

GDPR Article 17: Right to Erasure

Control Description: User data deletion upon request

Evidence Checklist: - [x] **Code**: `gdpr/data_erasure.py` - Automated deletion workflow - [x] **Procedure**: `/docs/gdpr_data_deletion_procedure.md` - [x] **Verification**: Audit log entry confirms deletion - [x] **Retention Policy**: Marketing data deleted after 30 days of account closure

Audit Procedure:

```
# Test GDPR data deletion
from gdpr.data_erasure import delete_user_data

# 1. Create test user
test_user_id = create_test_user()

# 2. Trigger deletion request
deletion_job_id = delete_user_data(user_id=test_user_id, reason="GDPR_ERASURE_REQUEST")

# 3. Verify all data removed
assert User.query.filter_by(id=test_user_id).first() is None
assert MiningData.query.filter_by(user_id=test_user_id).count() == 0
assert AuditLog.query.filter_by(event_type="USER_DELETED", user_id=test_user_id).count() == 1
```

GDPR Article 20: Data Portability

Control Description: Users can export their data in machine-readable format

Evidence Checklist: - [x] **Code**: `gdpr/data_export.py` - Export functionality - [x] **Format**: JSON (structured, machine-readable) - [x] **Scope**: All user data (account, mining data, calculations) - [x] **Delivery**: Secure download link (expires in 24h)

Export Data Structure:

```
{
  "export_metadata": {
    "export_date": "2025-10-03T14:00:00Z",
    "user_id": 1234,
    "data_version": "1.0"
  },
  "personal_data": {
    "email": "user@example.com",
    "account_created": "2024-01-15T10:00:00Z",
    "last_login": "2025-10-03T09:30:00Z"
  },
  "mining_data": {
    "total_calculations": 523,
    "miners": [...],
    "historical_results": [...]
  }
}
```

---

4. Security Implementation Evidence

4.1 KMS Key Management

Evidence Repository:

```
/security/kms/
├── kms_client.py          # KMS abstraction layer (AWS/GCP/Azure)
├── key_rotation_policy.md  # 90-day rotation schedule
├── encryption_context.py   # Tenant isolation logic
└── tests/
    └── test_kms_integration.py  # Integration tests
```

Audit Trail: - AWS CloudTrail: `kms:Encrypt`, `kms:Decrypt`, `kms:GenerateDataKey` events - GCP Audit Logs: Key usage by service account - Azure Monitor: Key Vault access logs

Verification Commands:

```
# Check key rotation compliance
aws kms describe-key --key-id $KMS_KEY_ID | jq '.KeyMetadata.CreationDate'
# Calculate days since creation, alert if >90 days

# List all active encryption keys
aws kms list-keys | jq '.Keys[].KeyId' | xargs -I {} aws kms describe-key --key-id {}
```

4.2 mTLS Mutual Authentication

Evidence Repository:

```
/security/mtls/
├── cert_validator.py       # X.509 certificate validation
├── crl_checker.py          # Certificate revocation list (CRL) checking
├── ocsp_client.py          # Online Certificate Status Protocol
├── ca_certificates/
│   ├── root_ca.crt         # Root CA (4096-bit RSA)
│   ├── intermediate_ca.crt # Intermediate CA
│   └── crl.pem             # Certificate Revocation List
└── client_certs/
    └── [issued_certs]/     # Client certificates (rotated every 365 days)
```

Certificate Inventory:

Certificate	Subject CN	Valid From	Valid Until	Key Size	Status
Root CA	HashInsight Root CA	2024-01-01	2034-01-01	4096-bit	Active
Intermediate CA	HashInsight Intermediate CA	2024-01-01	2029-01-01	4096-bit	Active
API Client #1	client-admin-001	2025-01-01	2026-01-01	4096-bit	Active
API Client #2	client-partner-acme	2025-06-01	2026-06-01	4096-bit	Active

Audit Procedure:

```
# 1. Verify certificate chain
openssl verify -CAfile ca_certificates/root_ca.crt \
  -untrusted ca_certificates/intermediate_ca.crt \
  client_certs/client-admin-001.crt
# Output: OK

# 2. Check certificate expiry (alert if <30 days)
openssl x509 -in client_certs/client-admin-001.crt -noout -enddate
```

```
# notAfter=Jan  1 00:00:00 2026 GMT

# 3. Validate CRL is current (updated within 24h)
openssl crl -in ca_certificates/crl.pem -noout -lastupdate -nextupdate
```

4.3 WireGuard Enterprise VPN

Evidence Repository:

```
/security/wireguard/
├── wireguard_config_generator.py  # Auto-generate peer configs
├── hub/
│   └── wg0.conf                   # Hub server config
├── sites/
│   ├── site_gateway_01.conf       # Mining farm gateway 1
│   ├── site_gateway_02.conf       # Mining farm gateway 2
│   └── ...
└── monitoring/
    └── wireguard_metrics.sh       # Connection monitoring
```

Network Topology:

```
                    ┌──────────────┐
                    │  Hub Server  │
                    │ 10.10.0.1/24 │
                    └──────┬───────┘
                           │
            ┌──────────────┼──────────────┐
            │              │              │
      ┌─────▼─────┐  ┌─────▼─────┐  ┌─────▼─────┐
      │ Gateway 1 │  │ Gateway 2 │  │ Gateway 3 │
      │10.10.0.10 │  │10.10.0.20 │  │10.10.0.30 │
      └───────────┘  └───────────┘  └───────────┘

         (Farm A)       (Farm B)       (Farm C)
```

Audit Checklist: - [x] ChaCha20-Poly1305 encryption enabled - [x] Automatic key rotation every 180 days - [x] Connection health monitoring (Prometheus metrics) - [x] Peer authentication via pre-shared keys

Verification:

```
# Check WireGuard tunnel status
sudo wg show wg0
# Expect: All peers connected, recent handshake (<3 min)

# Monitor tunnel traffic
watch -n 1 'sudo wg show wg0 transfer'
```

4.4 API Key Management

Evidence Repository:

```
/security/api_keys/
├── api_key_manager.py      # CRUD operations for API keys
├── key_validator.py        # Real-time validation
├── permissions_matrix.py   # Scope-based permissions
└── tests/
    └── test_api_key_security.py
```

API Key Format: `hsi_{environment}_key_{random}` - `hsi` = HashInsight prefix - `{environment}` = `dev`, `staging`, `prod` - `{random}` = 16-character cryptographically secure random string

Permissions Scopes:

```
PERMISSION_SCOPES = {
    'read:miners': 'View miner specifications',
    'write:miners': 'Add/update miner data',
    'read:calculations': 'View calculation results',
    'write:calculations': 'Run new calculations',
    'admin:users': 'Manage user accounts',
    'admin:billing': 'Access billing data',
}
```

Audit Log (API Key Events):

```
SELECT * FROM api_key_audit_log
WHERE event_type IN ('KEY_CREATED', 'KEY_REVOKED', 'KEY_ROTATION')
  AND created_at >= NOW() - INTERVAL '90 days'
ORDER BY created_at DESC;
```

5. Compliance Audit Calendar

5.1 Recurring Audit Schedule

Audit Type	Frequency	Next Scheduled	Owner	Deliverable
SOC 2 Type II	Annual	March 2026	CTO + External Auditor	SOC 2 Report
PCI DSS Self-Assessment	Annual	January 2026	Security Team	SAQ-A Form
GDPR Data Audit	Semi-annual	April 2026	DPO	Compliance Report
Penetration Testing	Quarterly	Q1 2026	External Firm	Pen Test Report
Vulnerability Scanning	Weekly	Automated	DevSecOps	Scan Results
Access Review	Quarterly	January 2026	Security Team	Access Matrix
Backup Testing	Monthly	November 2025	Operations	Restore Report
Key Rotation Verification	Quarterly	Q1 2026	Security Team	Rotation Log

5.2 Evidence Collection Timeline

30 Days Before Audit: - [ ] Collect all code evidence (security/, monitoring/, backup/) - [ ] Export audit logs (7-year retention verified) - [ ] Generate compliance reports (SLO, uptime, incidents) - [ ] Review and update policies (encryption, access control)

14 Days Before Audit: - [ ] Run security scans (OWASP ZAP, Nessus) - [ ] Test disaster recovery procedures - [ ] Verify all certificates are valid (SSL/TLS, mTLS) - [ ] Prepare evidence binder (digital + physical)

7 Days Before Audit: - [ ] Internal dry-run audit with checklist - [ ] Fix any identified gaps - [ ] Brief team on audit procedures - [ ] Confirm auditor access (VPN, documentation portal)

---

6. Evidence Repository Map

6.1 Primary Evidence Locations

Source Code Evidence:

```
/opt/hashinsight/
├── security/
│   ├── kms_client.py           # [CC6.6] Encryption at rest
│   ├── mtls/cert_validator.py  # [CC6.1] mTLS authentication
│   ├── wireguard/              # [C1.1] Network isolation
│   └── api_keys/               # [CC6.2] API authorization
├── monitoring/
│   ├── slo_manager.py          # [A1.2] Availability SLO
│   ├── audit_logger.py         # [CC7.2] Audit logging
│   └── prometheus_exporter.py  # [CC7.3] Metrics collection
├── backup/
│   └── backup_manager.py       # [A1.3] Backup & recovery
├── gdpr/
│   ├── data_erasure.py         # [GDPR Art.17] Right to erasure
│   └── data_export.py          # [GDPR Art.20] Data portability
└── config.py                   # [All] Security configurations
```

Operational Evidence:

```
/var/log/hashinsight/
├── audit/
│   ├── 2025-10-01.jsonl        # Daily audit logs
│   ├── 2025-10-02.jsonl
│   └── ...
├── auth.log                    # Authentication events
├── api_access.log              # API usage logs
├── kms_rotation.log            # Key rotation events
└── backup_verification.log     # Backup test results
```

Compliance Documentation:

```
/docs/compliance/
├── SOC2_Readiness_Checklist.pdf
├── PCI_DSS_SAQ_A.pdf
├── GDPR_Data_Processing_Agreement.pdf
├── Incident_Response_Plan.md
├── Data_Retention_Policy.md
└── Security_Training_Records.xlsx
```

6.2 External Evidence (Third-party Reports)

Evidence Type	Provider	Location	Refresh Frequency
SSL/TLS Certificate	Let's Encrypt	Nginx `/etc/ssl/`	90 days (auto-renew)
SSL Labs Report	Qualys	`https://ssllabs.com/ssltest/analyze.html?d=hashinsight.io`	Monthly
Pen Test Report	CyberSec Inc.	`/docs/compliance/pentest_2025_Q3.pdf`	Quarterly
SOC 2 Report (Stripe)	Stripe	Stripe Dashboard → Compliance	Annual
AWS Compliance Certs	AWS	AWS Artifact	On-demand

7. Incident Response & Breach Notification

7.1 Security Incident Evidence Chain

Incident Classification: - **Level 1 (Low)**: Failed login attempts, rate limit hits - **Level 2 (Medium)**: Unauthorized access attempts, DDoS - **Level 3 (High)**: Data breach, system compromise - **Level 4 (Critical)**: Ransomware, nation-state attack

Evidence Collection Checklist (Level 3+): 1. [ ] Preserve system state (memory dump, disk snapshot) 2. [ ] Collect all relevant logs (auth, audit, network) 3. [ ] Document timeline of events 4. [ ] Identify affected user accounts/data 5. [ ] Notify stakeholders within 72 hours (GDPR requirement) 6. [ ] Engage forensic investigation team 7. [ ] File incident report with authorities (if required)

Breach Notification Timeline (GDPR Art. 33/34): - **T+0h**: Incident detected, containment initiated - **T+2h**: Internal security team briefed - **T+8h**: CTO/CISO notified - **T+24h**: Preliminary impact assessment - **T+72h**: DPA notification (if personal data breach) - **T+7d**: Individual notification (if high risk to rights)

---

8. Continuous Compliance Monitoring

8.1 Automated Compliance Checks

Daily Automated Checks:

```
#!/bin/bash
# /opt/hashinsight/compliance/daily_checks.sh

# 1. Certificate expiry check
for cert in /etc/ssl/hashinsight/*.crt; do
  expiry=$(openssl x509 -in $cert -noout -enddate | cut -d= -f2)
  days_left=$(( ($(date -d "$expiry" +%s) - $(date +%s)) / 86400 ))
  if [ $days_left -lt 30 ]; then
    alert "CRITICAL: Certificate $cert expires in $days_left days"
  fi
done

# 2. Encryption status check
python3 -c "
from security.kms_client import KMSClient
kms = KMSClient.get_instance()
assert kms.is_key_enabled(), 'KMS key is disabled'
"

# 3. Audit log integrity check
sha256sum /var/log/hashinsight/audit/$(date +%Y-%m-%d).jsonl > /tmp/audit_hash.txt
aws s3 cp /tmp/audit_hash.txt s3://hashinsight-compliance/audit_hashes/

# 4. Access review (detect new admin accounts)
psql $DATABASE_URL -c "
```

```
  SELECT username, created_at
  FROM users
  WHERE role = 'admin' AND created_at >= NOW() - INTERVAL '1 day';
" | mail -s "New Admin Accounts" security@hashinsight.io
```

Weekly Compliance Dashboard: - 127/127 security controls passing - 0 critical vulnerabilities - ⚠ 2 SSL certificates expire in 45 days - All backups verified (last 7 days) - SLO: 99.97% (target: 99.95%)

---

9. Appendix

9.1 Contact Information

Security Team: - **CISO (Chief Information Security Officer)**: ciso@hashinsight.io - **Security Engineers**: security-team@hashinsight.io - **24/7 Security Hotline**: +1-800-HASHSEC

Compliance Officers: - **Data Protection Officer (DPO)**: dpo@hashinsight.io - **Compliance Manager**: compliance@hashinsight.io

External Auditors: - **SOC 2 Auditor**: [Audit Firm Name] - **Pen Testing**: CyberSec Inc. (contact@cybersec.example)

9.2 Audit Request Process

For External Auditors: 1. Submit audit request to `compliance@hashinsight.io` 2. Sign NDA (template provided) 3. Receive secure access to evidence portal 4. Schedule kickoff meeting (2 weeks notice) 5. Conduct audit (on-site or remote) 6. Deliver preliminary findings 7. Address remediation items (if any) 8. Receive final audit report

9.3 Evidence Request Form

Template:

```
Subject: SOC 2 Evidence Request - [Control ID]

Auditor: [Name]
Audit Firm: [Company]
Control Reference: [e.g., CC6.1 - Authentication]
```

```
Evidence Needed:
  - [ ] Source code (specific files)
  - [ ] Configuration files
  - [ ] Audit logs (date range)
  - [ ] Policies/procedures
  - [ ] Test results
  - [ ] Incident reports

Delivery Method:
  [ ] Secure portal upload
  [ ] Encrypted email
  [ ] In-person review

Deadline: [Date]
```

Document Control: - **Version**: 1.0 - **Last Updated**: October 3, 2025 - **Owner**: HashInsight Security & Compliance Team - **Review Cycle**: Quarterly (or upon significant security changes) - **Next Review**: January 1, 2026 - **Approvals**: - CISO: [Signature] - DPO: [Signature] - CTO: [Signature]

Change Log: | Date | Version | Changes | Approver | |------|---------|---------|----------| | 2025-10-03 | 1.0 | Initial compliance evidence index | CISO |

Classification: Internal Use Only (Auditor Access Permitted)