**USB Security System: Unified Technical Specification**

**Executive Summary**

This document specifies a comprehensive USB security system comprising two primary components: a **Gateway** that securely processes and packages USB content, and an **Agent** that provides controlled access to encrypted USB data. The system implements defense-in-depth security through hardware attestation, encrypted storage, policy-based access control, and comprehensive audit logging.

**1. System Architecture Overview**

**1.1 Core Components**

* **Gateway Service**: Windows service for USB device processing, encryption, and packaging
* **Agent Service**: Endpoint service for secure file presentation and user interaction
* **Key Management System (KMS)**: HashiCorp Vault with HSM integration for cryptographic operations
* **Policy Engine**: Open Policy Agent (OPA) for dynamic policy enforcement
* **Attestation Infrastructure**: TPM-based device attestation and MDM compliance verification

**1.2 Security Architecture Principles**

* **Zero Trust**: Every component must authenticate and be authorized for each operation
* **Encryption at Rest**: All USB content encrypted with ephemeral Content Encryption Keys (CEK)
* **Hardware Root of Trust**: TPM-backed attestation for device and service integrity
* **Fail-Secure**: System defaults to denial of access on any security control failure
* **Comprehensive Audit**: Immutable logging of all security-relevant operations

**2. Gateway Flow: USB Processing Pipeline**

**2.1 Device Detection & Hardware Claiming**

**2.1.1 Detection Mechanism**

* Deploy **Windows Filter Manager minifilter driver** (signed) for reliable volume attach/detach event monitoring
* Monitor Win32\_VolumeChangeEvent EventType=2 via WMI for USB insertion events
* Extract unique device identifiers: Serial Number, Volume GUID, Vendor/Product IDs
* Implement rate limiting per device UID within configurable time windows (default: 30 seconds)

**2.1.2 Exclusive Volume Locking**

* **Critical Requirement**: Implement hard enforcement device claiming
* In pre-create callback, execute IRP\_MJ\_CREATE with deny-share flags to prevent auto-mount
* Open device handle with **no sharing flags** (FILE\_SHARE\_NONE) for kernel-level exclusive access
* Maintain exclusive lock throughout entire processing pipeline until job finalization
* This prevents race conditions, malware access, and user interference during processing

**2.1.3 Device Metadata Collection**

Required Metadata Fields:

- device\_serial: USB device serial number

- volume\_guid: Windows volume GUID

- vendor\_id: USB vendor identifier

- product\_id: USB product identifier

- device\_capacity: Total storage capacity

- filesystem\_type: NTFS/FAT32/exFAT detection

- insertion\_timestamp: ISO 8601 UTC timestamp

- hostname: Processing gateway hostname

- gateway\_version: Software version identifier

**2.2 Job Lifecycle Management**

**2.2.1 Job Initialization**

* Generate job\_id using UUID4 for guaranteed uniqueness
* Create job persistence directory: %ProgramData%\SMX\Jobs\<job\_id>\
* Initialize three core job components:
  + metadata.json: Static device and job metadata
  + state.json: Current processing state with transitions
  + logs.jsonl: Append-only timestamped log entries with sequence numbers

**2.2.2 State Management**

Job States:

- INITIALIZED: Job created, metadata collected

- ENUMERATING: File system enumeration in progress

- POLICY\_CHECK: Pre-scan policy validation

- SCANNING: Malware analysis in progress

- PACKAGING: Encryption and manifest creation

- SUCCESS: Job completed successfully

- FAILED\_POLICY: Policy violation detected

- QUARANTINED: Malware or error detected

- FAILED: Unrecoverable error occurred

**2.2.3 Atomic Operations & Recovery**

* Use temporary files with atomic rename for metadata updates
* Implement write-ahead logging for state transitions
* On system restart, automatically resume incomplete jobs by reading state.json
* Implement configurable retention policies for completed jobs (default: 90 days)

**2.3 File Enumeration & Canonicalization**

**2.3.1 Robust Path Processing**

* Leverage Windows **PathCch API** for Unicode normalization (NFC/NFKC)
* Resolve reparse points safely with circular reference detection
* Sanitize paths: remove ".." traversal attempts, illegal characters, reserved names
* Generate deterministic, sorted file listing for reproducible processing

**2.3.2 NTFS Alternate Data Streams (ADS) Handling**

* **Security Critical**: Use BackupRead API to enumerate all ADS
* Implement ADS whitelist for known legitimate streams (:$DATA, :Zone.Identifier)
* **Default Deny**: Drop all unknown ADS to prevent hidden payload execution
* Log all ADS discoveries with stream names and sizes

**2.3.3 File System Edge Cases**

* Handle maximum path length limitations (260 chars standard, 32K with long path support)
* Process symbolic links and junctions with target validation
* Identify and exclude device nodes, named pipes, and special file types
* Implement size-based filtering for extremely large files (configurable threshold)

**2.4 Policy Enforcement Engine**

**2.4.1 Open Policy Agent (OPA) Integration**

* Deploy OPA as sidecar service with signed Rego policy bundles
* Implement policy signature verification using Ed25519 signatures
* Support live policy updates via GitOps with version-based rollback protection
* Cache policies in memory only; fail-closed on cache miss

**2.4.2 Policy Decision Points**

Evaluated Policies:

- Total file count limits

- Individual file size limits

- Total USB capacity limits

- File extension blacklist/whitelist

- Archive depth restrictions (ZIP, RAR, 7z)

- Device reputation scoring

- User-Based Access Control (UBAC) rules

- Time-based access restrictions

- Content-based classification rules

**2.4.3 Policy Violation Handling**

* On policy failure, immediately update state.json to FAILED\_POLICY
* Log detailed policy violation information including rule name and violation details
* Maintain device lock for potential remediation or VM-based analysis
* Generate user-friendly policy violation notifications

**2.5 Malware Analysis Pipeline**

**2.5.1 Static Analysis**

* **Primary Engine**: Microsoft Defender integration via Windows Defender APIs
* **Secondary Engine**: ClamAV for additional coverage
* Record comprehensive scan results:
  + Verdict: CLEAN/INFECTED/ERROR/TIMEOUT
  + Engine name and signature version
  + Threat names and affected file paths
  + Scan duration and resource usage

**2.5.2 Dynamic Analysis (Advanced Configuration)**

* Submit suspicious files to Cuckoo Sandbox for behavioral analysis
* Integration with VirusTotal API for reputation checking
* Implement file upload size limits and timeout controls
* Support air-gapped environments with offline analysis capabilities

**2.5.3 Threat Response**

* On malware detection, transition to QUARANTINED state
* Preserve device lock for optional VM-based detonation
* Generate detailed incident reports with file hashes and detection metadata
* Support integration with SIEM systems for threat intelligence

**2.6 Cryptographic Packaging**

**2.6.1 Content Encryption Key (CEK) Operations**

* Generate ephemeral AES-256-GCM CEK for each job using cryptographically secure RNG
* Encrypt all file content with generated CEK
* **Immediate Wrapping**: Send CEK to HashiCorp Vault Transit engine for HSM-based wrapping
* Use PKCS#11 HSM integration for Key Encryption Key (KEK) operations
* Securely wipe plaintext CEK from memory immediately after wrapping

**2.6.2 Manifest Creation & Integrity**

* Generate SHA-256 digest for each encrypted file blob
* Create comprehensive manifest including:
  + File paths and original metadata
  + SHA-256 digests of encrypted content
  + File sequence numbers for ordering
  + Encryption parameters and CEK identifiers
  + Job metadata and processing timestamps

**2.6.3 Digital Signing**

* Sign manifest.json using Ed25519 private key stored in HashiCorp Vault PKI engine
* Include signing certificate with OCSP stapling for real-time revocation checking
* Implement certificate auto-renewal (30-day validity) via TPM-based attestation
* Support signature verification downstream with certificate chain validation

**2.7 Job Finalization**

* Write final manifest and wrapped CEK data to job directory
* Update state.json to SUCCESS with completion timestamp
* **Critical**: Release exclusive device lock to allow user access
* Archive job data according to retention policies
* Generate job completion notifications with summary statistics

**3. Agent Flow: Secure File Presentation**

**3.1 Secure Execution Environment**

**3.1.1 Enclave-Based Architecture**

* **Split Architecture**: Untrusted shim process + trusted enclave core
* Use **Open Enclave SDK** for cross-platform enclave development
* Shim handles I/O operations and user interface
* Enclave performs all cryptographic operations and policy enforcement
* Implement secure channel between shim and enclave for data transfer

**3.1.2 Hardware Attestation**

* On startup, generate TPM Quote for platform integrity measurement
* Obtain MDM compliance token for device policy compliance verification
* Present attestation evidence to HashiCorp Vault for authentication
* Implement periodic re-attestation (configurable interval, default: 24 hours)

**3.2 Key Management & Unwrapping**

**3.2.1 Vault Authentication**

* Use TPM-bound AppRole authentication for initial service authentication
* Implement short-lived token rotation (default: 1 hour TTL)
* **No Persistent Storage**: Enforce -cache=false flag to prevent token caching
* Support offline operation with limited cached policy decisions

**3.2.2 Secure CEK Unwrapping**

* **Mutual ECDH Key Exchange**: Establish ephemeral shared secret between agent enclave and Vault
* Agent requests CEK unwrap with job context: job\_id, manifest\_hash, user\_identity, timestamp
* Vault returns CEK encrypted with ECDH-derived key
* **Enclave-Only Unwrapping**: Perform final CEK decryption entirely within enclave memory
* Immediately discard wrapped CEK data after successful unwrap

**3.2.3 User-Based Access Control (UBAC)**

* Issue unwrap tokens scoped to user identity via Vault OIDC integration
* Agent queries Active Directory LDAP for user group membership
* Cache group membership with short TTL (default: 15 minutes)
* Enclave verifies user claims against manifest access policies before unwrap

**3.3 Controlled File Presentation**

**3.3.1 Windows Projected File System (ProjFS) Integration**

* **Memory-Only Decryption**: Files remain encrypted at rest, decrypted on-demand in memory
* Implement ProjFS provider for seamless user experience
* Integrate with Windows Data Loss Prevention (DLP) and Endpoint Detection & Response (EDR)
* Restrict memory paging for decrypted content to prevent disk-based recovery

**3.3.2 File Access Controls**

* Enforce read-only access by default with explicit write permissions
* Implement file-level access controls based on user group membership
* Support watermarking and classification labeling integration
* Log all file access attempts with user identity and access patterns

**3.4 Secure Write Operations**

**3.4.1 Write Intent Processing**

* Intercept write operations through ProjFS write hooks
* Validate write permissions against current user policy
* Generate new CEK for modified content within enclave
* Re-encrypt modified files with new CEK immediately

**3.4.2 Policy Re-validation**

* Submit modified content to OPA for policy compliance checking
* Abort write operations on policy violations with detailed error messages
* Maintain audit trail of all modification attempts and policy decisions

**3.4.3 Two-Phase Commit Protocol**

* **Staging Phase**: Store encrypted modifications in memory-based staging area
* **Commit Phase**: Atomically write new manifest and encrypted blobs
* Implement conflict detection via manifest version comparison
* Provide user resolution interface for concurrent modification conflicts

**3.5 Audit & Logging**

**3.5.1 Immutable Log Chain**

* Generate hash-linked audit logs within enclave: prev\_hash || entry\_data → SHA-256
* Include sequence numbers and cryptographic timestamps
* Write tamper-evident logs to logs.jsonl with regular integrity verification

**3.5.2 External Audit Integration**

* Push log digests to ImmuDB or WORM storage for external tamper evidence
* Support integration with SIEM systems (Splunk, ELK Stack, Azure Sentinel)
* Generate compliance reports for regulatory requirements (GDPR, HIPAA, SOX)

**3.5.3 Fail-Safe Behavior**

* **Fail-Closed Policy**: If Vault/OPA unreachable, deny all unwrap and write operations
* Maintain clear error state indicators for users and administrators
* Implement graceful degradation with cached policy decisions when appropriate

**4. Key Management Infrastructure**

**4.1 HashiCorp Vault Configuration**

**4.1.1 Transit Engine (Cryptographic Operations)**

* **HSM Integration**: Configure PKCS#11 integration with certified HSM (Luna SA, nShield)
* **Key Hierarchy**: Root KEK in HSM, derived KEKs for different security domains
* Support key rotation with backward compatibility for existing encrypted content
* Implement key escrow capabilities for regulatory compliance requirements

**4.1.2 PKI Engine (Certificate Management)**

* Issue short-lived certificates (30-day validity) for gateway and agent signing
* Implement automated certificate renewal via TPM-based attestation
* Support certificate revocation with OCSP and CRL distribution
* Maintain certificate transparency logging for audit requirements

**4.1.3 Identity & Access Management**

* **OIDC Integration**: Connect with enterprise identity providers (Azure AD, Okta)
* **AppRole Authentication**: For service-to-service authentication
* **User-Based Policies**: Dynamic policy assignment based on user attributes
* Implement just-in-time access provisioning for administrative operations

**4.2 Open Policy Agent (OPA) Configuration**

**4.2.1 Policy Bundle Management**

* Implement signed policy bundles with Ed25519 signature verification
* Support GitOps-based policy deployment with version control
* Enforce policy rollback protection to prevent downgrade attacks
* Maintain policy decision audit logs with full request/response context

**4.2.2 Decision Caching & Performance**

* **Memory-Only Caching**: Sidecar cache permitted only in volatile memory
* Implement policy decision memoization for repeated queries
* **Fail-Closed Behavior**: Deny operations on cache miss or policy engine unavailability
* Support policy simulation and testing modes for safe policy updates

**4.3 Hardware Security Module (HSM) Integration**

**4.3.1 HSM Requirements**

* **Certification**: FIPS 140-2 Level 3 or Common Criteria EAL4+ certified
* **High Availability**: Clustered HSM deployment with automatic failover
* **Key Management**: Support for key generation, wrapping, and secure key storage
* **Audit Logging**: Comprehensive HSM operation logging with tamper detection

**4.3.2 Key Lifecycle Management**

* Automated key generation with cryptographically secure entropy sources
* Key backup and recovery procedures with split-knowledge controls
* Regular key rotation schedules based on cryptographic best practices
* Secure key destruction with certified media sanitization

**5. Security & Compliance Framework**

**5.1 Threat Model & Mitigations**

**5.1.1 Insider Threats**

* **Mitigation**: Privileged access management with just-in-time elevation
* Comprehensive audit logging of all administrative actions
* Separation of duties for critical operations
* Regular access reviews and automated deprovisioning

**5.1.2 Malware & Advanced Persistent Threats**

* **Mitigation**: Multi-layered malware detection with static and dynamic analysis
* Application whitelisting and code signing enforcement
* Network segmentation with micro-segmentation policies
* Behavioral analysis and anomaly detection

**5.1.3 Cryptographic Attacks**

* **Mitigation**: Use of established cryptographic algorithms (AES-256, Ed25519, SHA-256)
* Regular cryptographic library updates and vulnerability management
* Quantum-resistant algorithm evaluation and migration planning
* Hardware-based key protection with HSM integration

**5.2 Compliance Requirements**

**5.2.1 Data Protection Regulations**

* **GDPR Compliance**: Data minimization, purpose limitation, and user consent management
* **CCPA Compliance**: Consumer rights management and data disclosure tracking
* **HIPAA Compliance**: Administrative, physical, and technical safeguards for PHI

**5.2.2 Industry Standards**

* **NIST Cybersecurity Framework**: Implementation of Identify, Protect, Detect, Respond, Recover
* **ISO 27001**: Information security management system controls
* **SOC 2 Type II**: Security, availability, and confidentiality controls

**5.3 Monitoring & Incident Response**

**5.3.1 Security Monitoring**

* **SIEM Integration**: Real-time log analysis and correlation
* **Metrics Collection**: Prometheus metrics with Grafana dashboards
* **Alerting**: Automated incident detection and notification
* **Threat Intelligence**: Integration with external threat feeds

**5.3.2 Incident Response**

* **Automated Response**: Immediate containment actions for detected threats
* **Forensic Capabilities**: Comprehensive audit trails and evidence preservation
* **Recovery Procedures**: System restoration and business continuity planning
* **Lessons Learned**: Post-incident analysis and process improvement

**6. Implementation Considerations**

**6.1 Development Requirements**

**6.1.1 Technology Stack**

* **Gateway**: Windows Service (C# .NET Framework 4.8 or .NET 6+)
* **Filter Driver**: Windows Filter Manager (C/C++ with WDK)
* **Agent**: Windows Service with ProjFS integration
* **Enclave**: Open Enclave SDK with Intel SGX or ARM TrustZone support
* **Database**: SQLite for local job storage, PostgreSQL for centralized logging

**6.1.2 Testing Strategy**

* **Unit Testing**: Comprehensive test coverage with mocking for external dependencies
* **Integration Testing**: End-to-end workflow testing with real USB devices
* **Security Testing**: Penetration testing, fuzzing, and vulnerability assessment
* **Performance Testing**: Load testing with concurrent USB operations
* **Compatibility Testing**: Multi-platform and multi-device testing

**6.1.3 Quality Assurance**

* **Code Reviews**: Mandatory peer review for all security-critical code
* **Static Analysis**: Automated SAST scanning with SonarQube or similar
* **Dynamic Analysis**: Runtime security testing with tools like Veracode
* **Dependency Scanning**: Regular scanning for vulnerable third-party libraries

**6.2 Deployment Architecture**

**6.2.1 Enterprise Deployment**

* **Group Policy Integration**: Automated deployment via Active Directory GPO
* **Configuration Management**: Centralized configuration with Ansible/Puppet
* **Update Management**: Automated updates with rollback capabilities
* **Monitoring Integration**: Enterprise monitoring system integration

**6.2.2 Cloud Integration**

* **Hybrid Architecture**: Support for cloud-hosted Vault and OPA instances
* **Multi-Region Deployment**: Geographic distribution for performance and resilience
* **Disaster Recovery**: Cross-region backup and recovery capabilities
* **Compliance**: Cloud security posture management and compliance monitoring

**6.3 Performance & Scalability**

**6.3.1 Performance Requirements**

* **USB Processing**: Complete processing within 2 minutes for typical 32GB device
* **File Access**: Sub-100ms latency for file access through ProjFS
* **Concurrent Operations**: Support 10+ simultaneous USB device processing
* **Memory Usage**: Maximum 512MB per gateway instance, 256MB per agent

**6.3.2 Scalability Considerations**

* **Horizontal Scaling**: Support multiple gateway instances per organization
* **Load Balancing**: Distribute Vault and OPA requests across multiple instances
* **Caching Strategy**: Intelligent caching to reduce backend load
* **Resource Management**: Automatic resource cleanup and garbage collection

**7. Risk Assessment & Mitigation**

**7.1 Technical Risks**

**7.1.1 Single Points of Failure**

* **Risk**: Vault or HSM unavailability prevents all operations
* **Mitigation**: High-availability clustering with automatic failover
* **Contingency**: Offline emergency access procedures for critical situations

**7.1.2 Performance Bottlenecks**

* **Risk**: Large file processing causes system performance degradation
* **Mitigation**: Streaming encryption, chunked processing, and resource throttling
* **Monitoring**: Real-time performance metrics with automatic alerts

**7.1.3 Compatibility Issues**

* **Risk**: Filter driver incompatibility with security software
* **Mitigation**: Extensive compatibility testing and whitelisting procedures
* **Support**: Technical support processes for compatibility resolution

**7.2 Security Risks**

**7.2.1 Key Compromise**

* **Risk**: CEK or KEK compromise leads to data exposure
* **Mitigation**: Hardware-based key protection, regular rotation, and compromise detection
* **Response**: Automated key revocation and re-encryption procedures

**7.2.2 Policy Bypass**

* **Risk**: Users find ways to bypass security controls
* **Mitigation**: Defense-in-depth architecture with multiple policy enforcement points
* **Detection**: Behavioral analysis and anomaly detection for bypass attempts

**7.3 Operational Risks**

**7.3.1 Administrative Errors**

* **Risk**: Misconfiguration leads to security gaps or service outages
* **Mitigation**: Configuration validation, change management processes, and rollback capabilities
* **Training**: Comprehensive administrator training and certification

**7.3.2 Vendor Dependencies**

* **Risk**: Third-party component vulnerabilities or discontinuation
* **Mitigation**: Vendor security assessment, alternative vendor evaluation, and escape planning
* **Monitoring**: Continuous vulnerability monitoring and update management

**USB Security System: Unified Technical Specification**

*[Previous content remains unchanged through Section 6.3]*

**8. Revocation Flow & Access Termination**

**8.1 Revocation Scenarios**

**8.1.1 Immediate Revocation Triggers**

* User termination or role change
* Device compromise or theft
* Policy violation detection
* Certificate expiration or revocation
* Administrative security incident response

**8.1.2 Revocation Flow Diagram**

Revocation Event Detected

↓

[Administrative Action Required?]

↓ YES ↓ NO

Manual Revocation Automatic Revocation

↓ ↓

┌─────────────────────────────────────┐

│ Vault Operations │

│ 1. Revoke AppRole credentials │

│ 2. Revoke issued leases/tokens │

│ 3. Add user/device to deny list │

│ 4. Increment policy version │

└─────────────────────────────────────┘

↓

┌─────────────────────────────────────┐

│ OPA Policy Update │

│ 1. Push updated deny list │

│ 2. Increment policy bundle version │

│ 3. Distribute to all agents │

└─────────────────────────────────────┘

↓

┌─────────────────────────────────────┐

│ Agent Enforcement │

│ 1. Policy cache invalidation │

│ 2. Force re-authentication │

│ 3. Terminate active sessions │

│ 4. Clear in-memory CEKs │

└─────────────────────────────────────┘

↓

[Verification & Audit Logging]

↓

[Revocation Complete]

**8.2 Technical Implementation**

**8.2.1 Short-Lived Token Strategy**

* **CEK Unwrap Tokens**: Maximum TTL of 15 minutes (configurable: 5-30 minutes)
* **AppRole Tokens**: Maximum TTL of 1 hour with automatic renewal
* **Policy Tokens**: Maximum TTL of 2 hours with forced refresh on policy version mismatch

**8.2.2 Vault Lease Revocation**

Revocation API Call:

POST /v1/sys/leases/revoke-prefix

{

"lease\_id": "auth/approle/login/<user\_identifier>/\*"

}

Response Processing:

- HTTP 200: Revocation successful

- HTTP 204: No active leases found

- HTTP 500: Partial revocation (requires retry)

**8.2.3 Policy Version Enforcement**

* **Version Bump**: Increment global policy version number
* **Agent Validation**: Agents verify policy version on each unwrap request
* **Forced Re-wrap**: Version mismatch triggers complete session termination and re-authentication

**8.3 Revocation Verification**

**8.3.1 Immediate Effects (< 30 seconds)**

* Active Agent sessions terminated
* In-memory CEKs cleared from all endpoints
* New authentication attempts denied

**8.3.2 Propagation Timeline**

* **Vault revocation**: Immediate (< 5 seconds)
* **OPA policy distribution**: Within 1 minute
* **Agent policy refresh**: Within 2 minutes
* **Full system consistency**: Within 5 minutes

**9. Offline Mode & Grace Periods**

**9.1 Offline Grace Configuration**

**9.1.1 Policy Cache Validity**

* **Default Grace Period**: 2 hours for cached policy decisions
* **Maximum Grace Period**: 8 hours (configurable per security policy)
* **Grace Period Scope**: Read-only operations only; write operations require online validation

**9.1.2 Token Caching Strategy**

Cached Components (Memory Only):

✓ AppRole authentication tokens (1 hour TTL)

✓ Policy bundles with signatures (2 hour TTL)

✓ User group memberships (15 minute TTL)

✓ Device attestation status (4 hour TTL)

Never Cached:

✗ Content Encryption Keys (CEKs)

✗ Key Encryption Keys (KEKs)

✗ Private signing keys

✗ Unwrap tokens beyond active session

**9.2 Offline Operation Modes**

**9.2.1 Connected Mode (Default)**

* Real-time policy validation for all operations
* Immediate revocation enforcement
* Full audit logging to central systems
* Dynamic policy updates

**9.2.2 Degraded Mode (Network Issues)**

* **Duration**: Up to 2 hours with cached policies
* **Restrictions**: Read-only access to previously validated content
* **Logging**: Local logging with sync on reconnection
* **User Notification**: "Operating in offline mode" indicator

**9.2.3 Offline Mode (Extended Outage)**

* **Duration**: Beyond 2-hour grace period
* **Behavior**: **Fail-closed** - Deny all unwrap requests
* **Exception**: Emergency access procedures (if configured)
* **Recovery**: Full re-authentication required on reconnection

**9.3 Reconnection Procedures**

**9.3.1 Automatic Recovery**

Network Connectivity Restored

↓

[Verify Vault/OPA Connectivity]

↓

[Validate Cached Token Status]

↓

[Sync Pending Audit Logs]

↓

[Refresh Policy Bundles]

↓

[Resume Normal Operation]

**9.3.2 Manual Recovery (Post-Incident)**

* Force full re-authentication for all users
* Invalidate all cached credentials and policies
* Require fresh device attestation
* Complete audit log reconciliation

**10. Enclave Strategy & Fallback Architecture**

**10.1 Primary Enclave Technologies**

**10.1.1 Intel SGX (Preferred)**

* **Requirements**: Intel processors with SGX support (6th gen and later)
* **Benefits**: Hardware-enforced memory encryption and attestation
* **Limitations**: Limited EPC memory, driver dependencies
* **Use Case**: High-security environments with compatible hardware

**10.1.2 ARM TrustZone (Alternative)**

* **Requirements**: ARM processors with TrustZone technology
* **Benefits**: Secure world isolation, lower overhead than SGX
* **Limitations**: Platform-specific implementation
* **Use Case**: ARM-based Windows devices (Surface Pro X, etc.)

**10.2 Fallback Architecture**

**10.2.1 Windows Virtualization-Based Security (VBS)**

Fallback Decision Tree:

SGX Available? → YES → Use Intel SGX

↓ NO

TrustZone Available? → YES → Use ARM TrustZone

↓ NO

VBS Compatible? → YES → Use VBS Enclaves

↓ NO

Use Process Isolation + DLP

**10.2.2 VBS Enclave Implementation**

* **Secure Kernel**: Leverage Windows Hypervisor Code Integrity (HVCI)
* **Memory Protection**: Use Virtual Secure Mode (VSM) for crypto operations
* **Attestation**: TPM-based attestation with Platform Configuration Registers (PCR)
* **Performance**: Higher overhead than SGX but broadly compatible

**10.2.3 Process Isolation + DLP (Minimum Security)**

* **Process Hardening**: Run agent in sandboxed process with limited privileges
* **Memory Protection**: Use VirtualLock to prevent memory paging
* **DLP Integration**: Enforce Windows Information Protection (WIP) policies
* **Monitoring**: Enhanced logging and behavioral analysis for anomaly detection

**10.3 Enclave Detection & Selection**

**10.3.1 Runtime Capability Detection**

Agent Startup Sequence:

1. Query CPU capabilities (CPUID for SGX, ARM features)

2. Test enclave creation (handle failures gracefully)

3. Validate TPM availability and attestation capability

4. Select highest security mode available

5. Log selected security mode for audit

**10.3.2 Security Mode Indicators**

* **High Security**: SGX or TrustZone with hardware attestation
* **Medium Security**: VBS enclaves with TPM attestation
* **Basic Security**: Process isolation with DLP controls
* **Insufficient Security**: Fail deployment, require remediation

**11. Operational Model & Ownership**

**11.1 Organizational Responsibilities**

**11.1.1 Security Operations Team**

**Primary Owner**: Chief Information Security Officer (CISO) organization

**Responsibilities**:

* HashiCorp Vault cluster management and monitoring
* HSM lifecycle management (key rotation, backup, disaster recovery)
* Security policy definition and governance
* Incident response and forensic analysis
* Compliance reporting and audit coordination

**Staffing Requirements**:

* 2+ Vault administrators with HashiCorp certifications
* 1+ HSM specialist with vendor certifications
* 24/7 on-call rotation for security incidents

**11.1.2 Platform Engineering Team**

**Primary Owner**: Chief Technology Officer (CTO) organization

**Responsibilities**:

* Open Policy Agent (OPA) cluster deployment and scaling
* Gateway and Agent software deployment automation
* Infrastructure monitoring and performance optimization
* Configuration management via GitOps pipelines
* Integration with enterprise systems (AD, MDM, SIEM)

**Staffing Requirements**:

* 2+ Platform engineers with Kubernetes/container expertise
* 1+ Windows systems specialist for filter driver support
* DevOps engineers for CI/CD pipeline maintenance

**11.1.3 End-User Computing Team**

**Primary Owner**: Chief Information Officer (CIO) organization

**Responsibilities**:

* Endpoint deployment via Group Policy or MDM
* End-user support and training programs
* Hardware compatibility testing and validation
* Performance monitoring and capacity planning
* Change management and rollout coordination

**Staffing Requirements**:

* Desktop support technicians for L1/L2 support
* System administrators for Group Policy management
* Training specialists for user education programs

**11.2 Service Level Agreements (SLAs)**

**11.2.1 Availability Targets**

* **Vault/HSM Availability**: 99.9% uptime (8.76 hours downtime/year)
* **OPA Policy Engine**: 99.95% uptime (4.38 hours downtime/year)
* **Gateway Processing**: 99.5% success rate for standard USB operations
* **Agent File Access**: 99.9% availability during business hours

**11.2.2 Performance Targets**

* **USB Processing Time**: 95th percentile under 3 minutes for 32GB device
* **File Access Latency**: 95th percentile under 200ms via ProjFS
* **Policy Decision Time**: 95th percentile under 100ms for OPA queries
* **Key Unwrap Operations**: 95th percentile under 500ms end-to-end

**11.3 Escalation Matrix**

**11.3.1 Incident Severity Levels**

**Severity 1 (Critical)**:

* Complete system outage affecting all users
* Security breach or malware detection failure
* HSM failure or key compromise
* **Response Time**: 15 minutes, 24/7 coverage

**Severity 2 (High)**:

* Degraded performance affecting >50% of users
* Single component failure with workaround
* Policy enforcement bypass detection
* **Response Time**: 2 hours during business hours

**Severity 3 (Medium)**:

* Individual user access issues
* Non-critical feature failures
* Performance degradation <25% of users
* **Response Time**: 8 hours during business hours

**11.2.2 Communication Channels**

* **Critical Incidents**: PagerDuty → Security team → Executive notification
* **Status Updates**: ServiceNow incidents → Automated status page
* **User Communication**: Email notifications → Internal portal updates