

# Master Guide: Secure Multi-Server AD Sync with OpenBao

This system provides a high-security framework for synchronizing Active Directory users, security groups, and credentials across disconnected (air-gapped) environments. It utilizes a sophisticated "Pull" architecture to minimize the attack surface of the source environment, ensuring that high-trust domains remain isolated from inbound network requests and potential lateral movement from lower-security zones.

## 1. System Architecture

The workflow is divided into two distinct roles, strategically separated to maintain a strictly one-way flow of information and control:

- **Source Server (The Exporter):** Resides in the primary (source) domain. It performs a comprehensive discovery scan of a designated Organizational Unit (OU), manages user passwords within a local, hardened OpenBao Vault instance, encrypts those secrets via RSA-4096, and generates a cryptographically signed transport package ready for secure retrieval.
- **Target Server (The Receiver):** Resides in the secondary (target) domain. It initiates an outbound "Pull" request via SFTP to retrieve the package from the Source. Upon arrival, it restores the cryptographic twin-context, verifies the payload's structural and cryptographic integrity against the digital signature, and reconciles the Target AD to mirror the source state with precision.

## 2. Detailed Script Breakdown

### [1] Initialize-ADSyncEnvironment.ps1 (The Architect)

This script serves as the foundational deployment tool and must be executed with Local Administrator privileges on both the Source and Target servers to establish the necessary infrastructure and security guardrails.

- **Directory Provisioning:** Creates a hardened folder structure (C:\ADSync) and applies strict NTFS inheritance rules. By limiting access exclusively to local Administrators and the SYSTEM account, it prevents unauthorized users or service accounts from tampering with binaries, configuration files, or sensitive cryptographic metadata.
- **Service Management:** Configures the OpenBao binary as a persistent Windows Service. This setup includes a loopback listener restricted to 127.0.0.1, ensuring the Vault's REST API is physically unreachable from the external network, thereby neutralizing entire classes of remote web-based attacks.
- **Network Hardening:** Automatically provisions granular Windows Firewall rules. It opens inbound Port 22 for the SFTP service and Port 8200 for local API traffic, while

simultaneously authorizing outbound ports for critical Active Directory communication (LDAP 389, LDAPS 636, SMB 445 for RPC, and Kerberos 88) to ensure the engine can communicate reliably with Domain Controllers.

- **Vault Bootstrapping:** Orchestrates the first-time "Operator Init" process. It generates the master unseal keys and the root token, which are exported to vault\_keys.json. This file effectively becomes the "keys to the kingdom" and must be protected with the highest level of offline scrutiny or stored in a secure hardware security module (HSM) if available.

## [2] Invoke-BaoAutomation.ps1 (The Gatekeeper)

This script manages the daily operational health and initial security provisioning of the OpenBao Vault, ensuring the cryptographic environment is always primed for synchronization.

- **Auto-Unsealing:** Since OpenBao starts in a "sealed" state by default for maximum security, this script reads the vault\_keys.json to programmatically unlock the Vault. This allows the synchronization engine to continue functioning without manual human intervention after a server reboot, power failure, or service restart.
- **Engine Lifecycle:** Dynamically provisions and monitors the required storage engines. It sets up the KV-V2 (Key-Value) engine for persistent administrative secrets and the Transit engine to provide high-performance "Encryption-as-a-Service" for user passwords and sensitive attributes.
- **Burn-After-Reading Ingestion:** Provides a secure bridge for the initial system configuration. It safely imports administrative credentials from a temporary JSON file into the Vault's internal, encrypted storage. Once ingestion is confirmed, it performs a secure deletion of the plaintext file, overwriting the disk sectors to ensure the password cannot be recovered via forensic data recovery tools.

## [3] Sync-AD-Transport.ps1 (The Engine)

The core logic engine of the project. It uses sophisticated environment detection to determine its operational mode, allowing a single, auditable codebase to maintain consistency across different server roles.

- **v5.2 Logic Enhancements:**
  1. **Conditional Password Logging:** The engine now optimizes the audit trail in C:\ADSync\Users. A reference .txt file is generated **only when a new password is created** for a user. Existing users synchronized via the Vault do not trigger a new file write, reducing data clutter and exposure.
  2. **Expanded Attribute Fidelity:** Synchronizes an enterprise-grade attribute set including Title, Department, EmployeeID, StreetAddress, and MobilePhone alongside core identity data.
  3. **Purge-on-Completion:** On the Target server, the script automatically deletes the transport payload, signature, and key backup immediately after a successful import to prevent stale sensitive data from persisting on disk.
- **Export Mode Logic:**
  1. **Deep Discovery:** Performs a recursive scan of the Source OU to identify all user

- objects, their metadata, and their corresponding security group hierarchies.
2. **Cryptographic Protection:** For every user identified, it retrieves an existing password or generates a new one. This secret is then encrypted using **RSA-4096** (Asymmetric Encryption). This ensures that even if the transport file is intercepted or stolen, the passwords remain mathematically unreadable without the private key held inside the Target Vault.
  3. **Serialization:** Bundles all discovered user attributes, encrypted secrets, and group memberships into a single, compact JSON payload.
  4. **Signing:** Generates an **HMAC-SHA256** signature of the entire payload. This acts as a digital tamper-evident seal; if a single bit of the file is modified during transit, the seal breaks and the import is aborted.
- **Import Mode Logic:**
    1. **Integrity Validation:** Before processing any data, it verifies the HMAC signature. If the signature is invalid, the script terminates immediately to prevent the injection of malicious accounts or corrupt data into the Target AD.
    2. **Cryptographic Twin Restoration:** Bridges the air-gap by restoring the Transit Key from the backup blob. This allows the Target Vault to act as a cryptographic twin of the Source, enabling it to decrypt the RSA-4096 ciphertext.
    3. **AD Reconciliation:** Decrypts the passwords and applies a full CRUD (Create, Update, Delete) cycle. It creates missing users, synchronizes modified attributes (like job titles or departments), and removes stale accounts that have been deleted from the Source OU to maintain strict parity.
    4. **Membership Parity:** Iterates through security groups to ensure user memberships in the Target domain exactly mirror the Source, including the dynamic creation of missing groups within the Target OU.

## [4] Receive-ADSyncPayload.ps1 (The Courier)

A specialized transport utility that resides exclusively on the **Target Server**, acting as the secure bridge between the two isolated domains.

- **SFTP Pull Client:** Initiates a secure connection to the Source server via SSH/SFTP. By "Pulling" data into the Target domain rather than "Pushing" out of the Source, the high-security source environment remains completely invisible to inbound network requests, significantly hardening the primary domain against external scanning and lateral movement.
- **Transfer Verification:** Rigorously evaluates the successful arrival of the three-part sync package: the JSON user data, the HMAC integrity signature, and the encrypted Transit Key backup blob. It performs file-size and checksum comparisons to ensure the files were not truncated or corrupted during the transfer process.
- **Workflow Orchestrator:** Acts as the automation trigger for the entire import sequence. Once the transfer is 100% verified, it automatically launches the local instance of Sync-AD-Transport.ps1, ensuring the absolute minimum latency between data arrival and the update of Active Directory records.

## [5] Update-VaultPassword.ps1 (The Override)

An interactive administrative utility designed for emergency scenarios, helpdesk requests, or manual password rotations that fall outside the standard automated sync schedule.

- **Secure Manual Rotation:** Provides a protected interface for an authorized administrator to set a specific password for a user within the Vault. This ensures that the administrator never has to touch the Active Directory console directly, maintaining the Vault as the single source of truth for passwords.
- **Comprehensive Audit Trail:** Every manual override is logged with a high-resolution timestamp and the administrator's identity to the Windows Event Log. Additionally, it generates a local reference file in C:\ADSync\Users to provide a human-readable audit log of manual interventions outside of the automated flow.

## 3. Security & Air-Gap Cryptography

### Asymmetric Transport Engine

The system utilizes an **Exportable Transit Engine** configuration to solve the "Key Paradox" often found in disconnected environments:

- **RSA-4096 Military-Grade Encryption:** This provides a massive security margin. Unlike symmetric encryption (like AES), where a shared key must be pre-shared over an insecure channel, this asymmetric approach allows for much more secure and complex trust relationships between isolated servers.
- **Key Wrapping & Restoration:** The transport-key is backed up as a "wrapped" encrypted blob. This blob can only be successfully unwrapped and used by a Vault instance that shares the same master initialization parameters. Sync-AD-Transport.ps1 handles this restoration automatically, maintaining cryptographic continuity without the need for manual, insecure key hand-offs.
- **HMAC-SHA256 Integrity Seals:** By signing every payload, we ensure that the transport medium (the SFTP server or the network path) does not need to be trusted. Even in a "Total Compromise" scenario of the SFTP server, the Target server will identify that the "seal" on the payload is broken and will refuse to ingest any data.

## 4. Account & Permission Requirements

### Scheduled Task Service Account

For the system to operate autonomously and reliably, the service account (e.g., SVC\_ADSync) must be granted a specific, audited set of high-privilege rights:

1. **Local Server Permissions:**
  - Must be granted the **"Log on as a batch job"** right within the Local Security Policy to allow execution via Task Scheduler.
  - Requires **"Full Control"** over the C:\ADSync directory tree to manage the Vault keys, binary logs, and transport payloads.

## 2. Active Directory Delegation:

The account must be delegated the following specific rights on the Target OU to perform object reconciliation:

- **Create/Delete User & Group Objects:** To manage account lifecycles automatically.
- **Write all properties:** To synchronize crucial metadata like DisplayName, Office location, and Departmental codes.
- **Reset Password:** An absolute requirement to push the decrypted passwords from the Vault into the AD objects.
- **Modify Membership:** Required to manage the member attribute of security groups and ensure group-based access controls are synced.

# 5. Implementation Steps

## Step 1: Initialization

1. Ensure the bao.exe binary is correctly placed in C:\ADSync\OpenBao.
2. Execute Initialize-ADSyncEnvironment.ps1 as a Local Administrator to build the system infrastructure and register the services.

## Step 2: Security Provisioning

1. Place the administrative credentials in ad\_creds\_temp.json within the Sync folder for initial ingestion.
2. Execute Invoke-BaoAutomation.ps1. This unseals the vault, prepares the cryptographic engines, and securely ingests the admin credentials before purging the plaintext file.

## Step 3: Automation Setup

Configure the **Windows Task Scheduler** to ensure hands-off, consistent operation:

- **Source Server:** Schedule Sync-AD-Transport.ps1 (Daily at 01:00 AM) to generate the signed export package.
- **Target Server:** Schedule Receive-ADSyncPayload.ps1 (Daily at 02:00 AM) to pull the export package and automatically trigger the import engine.

# 6. Audit & Troubleshooting

All system operations, errors, and successes are audited in the **Windows Event Viewer** under the specialized log **Applications and Services Logs > ADSync**. This provides a centralized, standard location for monitoring the health and security of the synchronization process.

- **Event ID 1000:** Detailed logs for Core Engine Sync actions (User creates, attribute updates, account deletions).
- **Event ID 2000:** Logs for manual password overrides initiated via the Update-VaultPassword tool.
- **Event ID 3001:** Detailed status of SFTP transport operations, including connection failures, credential errors, or successful package downloads.