Sneakernet: Production Air Gap Implementation

Project: Automated ZFS replication via encrypted removable media
Environment: FreeBSD with ZFS and GELI encryption
Repository: http://svn.dailydata.net/svn/zfs_utils/trunk
License: BSD 2-Clause (FreeBSD License)

Overview

The **sneakernet** project provides a complete automated solution for air gap server replication using ZFS datasets and encrypted transport media. This implementation was developed for a production environment requiring monthly off-site backups with split-key security architecture.

Key Features:

- Automatic source/target mode detection
- Three-drive rotation minimizing site visits
- Multi-layer encryption (GELI + symmetric transport)
- Split-key architecture preventing single-point compromise
- Maintenance mode flag to pause replication
- Monthly cleanup script scheduling (built-in)
- Size tracking and max-delta validation before transfer
- Automated maintenance script execution
- Comprehensive reporting and audit trails
- Full dry-run testing capability

Client Requirements

A production deployment required the following specifications:

Requirement	Implementation
Replication Schedule	Monthly updates from in-house backup server to air gap server (target)
Transport Media	3× 1.9TB SSD drives in rotation
Drive Rotation	One at source, one at target, one in transit — minimizes site visits
Security Model	Multi-layer encryption with split-key architecture
Location	Air gap server in unsecured location (mandatory encryption)
Automation	Fully automated with maintenance script execution

Security Architecture

Encryption Layers:

1. **At Rest (Target):** GELI full disk encryption on air gap server
2. **In Transit:** Symmetric key encryption for all data on transport drives
3. **Maintenance Scripts:** Encrypted with same symmetric key
4. **Split-Key Design:** Target GELI key derived from:
    - Server-resident key component (stored locally)
    - Operator-carried key component (physical transport)
    - Combined via XOR bitwise operation at decrypt time
    - Target GELI key stored securely to facilitate key rotation and recovery

Split-key advantage: Neither component alone can decrypt the air gap server. Compromise of a single key (server or transport) does not expose data.

Installation

Prerequisites

- FreeBSD 13.0 or later (or Linux with ZFS)
- ZFS filesystem
- Perl 5.10 or later
- OpenSSL
- GELI kernel module (for target server encryption)
- Subversion client (for checkout)

Getting the Source

Repository URL: http://svn.dailydata.net/svn/zfs_utils/trunk
Sub-project: sneakernet

Export the project:

```
mkdir -p /usr/local/opt
svn export http://svn.dailydata.net/svn/zfs_utils/trunk
/usr/local/opt/zfs_utils
cd /usr/local/opt/zfs_utils
```

Configuration

The sneakernet script uses YAML configuration files:

1. Main configuration: sneakernet.conf.yaml

2. Default structure: `sneakernet.datastructure`
3. On first run, creates config from datastructure if missing

Key Configuration Sections:

- `source` — Source server settings (hostname, poolname)
- `target` — Target server settings (hostname, poolname, GELI config)
- `transport` — Transport drive settings (label, encryption key, mountpoint)
- `datasets` — Dataset replication mappings
- `source.cleanupScriptSchedule` — Month-based scheduling for maintenance scripts
- `source.oneShotCleanup` — One-time maintenance scripts directory
- `target.maintenanceMode` — Flag files that pause replication
- `target.report.targetDrive` — Optional report drive settings

Example minimal configuration snippet:

```
source:
  hostname: backup-primary
  poolname: tank
  cleanUpScriptsDir: /usr/local/opt/zfs_utils/sneakernet/cleanupScripts
  cleanupScriptSchedule:
    cleanSnaps: [1,2,3,4,5,6,7,8,9,10,11,12]
    scrubZFS: [2,5,8,11]
  oneShotCleanup: /usr/local/opt/zfs_utils/sneakernet/oneShotCleanup
target:
  hostname: airgap-backup
  poolname: backup
  maintenanceMode:
    flags:
      local: /tmp/maintenance
      transport: flags/maint.flag
  report:
    targetDrive:
      label: report
      fstype: msdos
      mountPoint: /mnt/report
transport:
  label: sneakernet
  fstype: ufs
  mountPoint: /mnt/sneakernet
  encryptionKey: your_hex_key_here
datasets:
  dataset1:
    source: tank
    target: backup
    dataset: data
```

Operation Workflows

Source Server Workflow

The source server performs the following operations automatically:

1. Auto-detect operating mode (source vs. target) via hostname
2. Mount transport drive using GPT label detection
3. Verify transport drive processed by target (check serial.txt)
4. Securely erase previous data from transport drive
5. Pre-calculate replication sizes and validate against history/maxDelta
6. Calculate incremental ZFS replication stream
7. Encrypt and write replication data to transport drive
8. Record latest snapshots sent (update status file)
9. Select cleanup scripts scheduled for the current month
10. Encrypt and write maintenance scripts to transport drive
11. Copy one-shot cleanup scripts and delete from source
12. Create serial.txt timestamp marker
13. Unmount transport drive
14. Email completion report to administrators

Command:

```
/usr/local/opt/zfs_utils/sneakernet/sneakernet
```

Target Server Workflow

The target server performs the following operations automatically:

1. Check maintenance mode flag(s) and pause if configured
2. Mount transport drive
3. Verify serial.txt exists (indicates unprocessed data)
4. Detect operator-provided secure key (USB/separate media)
5. Combine server key with operator key (XOR operation)
6. Unlock GELI encrypted disks using combined key
7. Import ZFS pool
8. Save current snapshot list to state file (enable rollback if needed)
9. Decrypt and import replication streams from transport
10. Remove serial.txt (marks data as processed)
11. Decrypt and execute maintenance scripts
12. Generate detailed report and write to report drive (if configured)
13. Unmount all media
14. Power off system (if `shutdownAfterReplication` enabled)

Command:

```
/usr/local/opt/zfs_utils/sneakernet/sneakernet
```

The script automatically detects whether it's running on the source or target server by comparing the hostname to the configuration. No mode flags required.

Three-Drive Rotation Strategy

The three-drive rotation minimizes operational overhead:

Normal Operation Cycle:

1. Receive confirmation that the source drive is populated and ready.
2. At the source, remove the populated drive and install the in-transit drive for the next cycle.
3. Transport the populated drive to the target site.
4. At the target, remove the current target drive (last cycle's drive).
5. Install the populated drive into the target, replacing the drive you removed.
6. Verify the target recognizes the newly installed drive.
7. Return the removed target drive to the source on the next cycle.

Benefits:

- Each site visit handles both delivery and pickup
- No waiting time for drive processing
- Reduced frequency of site access (security benefit)
- Built-in offline backup (data exists on multiple drives)

Drive Labeling:

Each transport drive should be labeled with GPT labels:

```
# Label drives for easy identification
gpart add -t freebsd-ufs -l sneakernet /dev/ada0
gpart add -t freebsd-ufs -l sneakernet /dev/ada1
gpart add -t freebsd-ufs -l sneakernet /dev/ada2

# Create filesystems
newfs -U /dev/gpt/sneakernet
newfs -U /dev/gpt/sneakernet
newfs -U /dev/gpt/sneakernet
```

Key Management

Symmetric Transport Key

- Unique key per deployment
- Stored on both source and target servers
- Used to encrypt data and scripts on transport drives
- 256-bit AES encryption (AES-256-CBC mode)

Generate transport key:

```
# Generate 32-byte (256-bit) key in hex format
openssl rand 32 | xxd -p | tr -d '\n' > /secure/path/transport.key
```

```
chmod 400 /secure/path/transport.key
```

Split GELI Key

- Server component: Stored on target server (never leaves facility)
- Operator component: Carried by trusted operator (never stored at target)
- Combined at runtime via XOR: `final_key = server_key ⊕ operator_key`

Generate split keys:

```
# Generate the final GELI key (this will be stored securely off-site)
openssl rand 512 > /secure/offsite/final_geli.key

# Generate operator key
openssl rand 512 > /media/operator/operator.key

# Generate server key (XOR of final and operator keys)
# This requires the makeGeliKey utility from ZFS_Utils
/usr/local/opt/zfs_utils/utilities/makeGeliKey \
  /secure/offsite/final_geli.key \
  /media/operator/operator.key \
  /secure/server/server.key
```

Key Rotation Procedures

If transport drive compromised:

```
# Generate new symmetric key
openssl rand 32 | xxd -p | tr -d '\n' > /secure/path/new_transport.key

# Deploy as maintenance script on next run
# Old data on compromised drive remains encrypted with old key
```

If operator key compromised:

```
# Generate new operator key
openssl rand 512 > /media/operator/new_operator.key

# Retrieve final GELI key from secure off-site storage
# Regenerate server key using makeGeliKey
/usr/local/opt/zfs_utils/utilities/makeGeliKey \
  /secure/offsite/final_geli.key \
  /media/operator/new_operator.key \
  /secure/server/server.key

# Operator must use new key on next visit
```

Key rotation can be automated through maintenance scripts. New keys deployed during normal replication cycles without requiring emergency site visits.

Command-Line Options

```
Usage: sneakernet [OPTIONS]

Options:
  -n, --dryrun              Run in dry-run mode (no writes, shows what would
happen)
  -v, --verbosity LEVEL     Set verbosity level (0-5)
  -d, --debug LEVEL         Debug breakpoint level (integer)
  -V, --version             Display version number
  -h, --help                Display this help message

Verbosity Levels:
  0 = Errors and critical messages only
  1 = Standard operations
  2 = Detailed operations
  3 = Debugging information
  4 = Snapshot lists
  5 = Full detailed output
```

Examples:

```
# Test configuration without making changes
sneakernet --dryrun

# Run with detailed logging
sneakernet -v 2

# Maximum verbosity for troubleshooting
sneakernet -v 5
```

Maintenance Scripts

Maintenance scripts are Perl scripts that execute on the target server after replication completes.

Script Requirements:

- Must be valid Perl code
- Return **two strings**: `($resultsString, $errorsString)`
  - `$resultsString` is optional informational output (newline-delimited)
  - `$errorsString` is empty on success, or newline-delimited errors
- Encrypted with transport symmetric key
- Stored in configured `cleanUpScriptsDir` on transport
- Optional one-shot scripts can be transferred once and then deleted at source

- See `cleanupScripts/helloWorld` for the authoritative template

Example maintenance script (based on `cleanupScripts/helloWorld`):

```
#!/usr/bin/env perl
# helloWorld-style template

use strict;
use warnings;

my @result;
my @errorMessages = ();
my $caller = caller(); # Check if called from another script and, if not,
prints updates to STDOUT
my $verbosityLevel = ($caller && defined $ZFS_Utils::verboseLoggingLevel) ?
$ZFS_Utils::verboseLoggingLevel : 5;


# Add messages to result array
push @result, "hello";
push @result, "world" if $verbosityLevel > 2;

# Add test error message (remove for production)
push @errorMessages, "This is a test error message";

# Return two strings: results and errors
return (
  join("\n", @result) . "\n",
  @errorMessages ? join("\n", @errorMessages) : ""
);
```

Month-based scheduling (monthly updates assumed):

Configure the scripts that should be transferred on the current month:

```
source:
  cleanupScriptSchedule:
    zpoolStats: [1,2,3,4,5,6,7,8,9,10,11,12]
    cleanSnaps: [1,2,3,4,5,6,7,8,9,10,11,12]
    scrubZFS: [2,5,8,11]
    trimZFS: [3,6,9,12]
    runSmart: [1,7]
```

Scripts not listed are not copied. This aligns with monthly updates to the air gap target.

Deploy maintenance script:

```
# On source server: encrypt and copy to cleanup directory
openssl enc -aes-256-cbc -salt -in cleanup_script.pl \
  -out /configured/cleanup/dir/cleanup_script.pl.enc \
```

```
  -pass file:/secure/transport.key
```

Size Tracking and Validation

Before replication, the source pre-calculates stream sizes and compares them to historical averages stored in the source history file. If a dataset exceeds its `maxDelta` threshold, replication is aborted to prevent unexpected large transfers (e.g., ransomware). The transfer is also aborted if the estimated total size exceeds transport capacity.

Monitoring and Reports

Reports send a brief summary, followed by the logs generated. The degree of detail in the logs are controlled by the verbosity level, so can vary based on this setting. Most mainteannce scripts will change the amount of output based on the verbosity level also.

Note: While it is assumed the output of the reports is e-mail for the source machine and a file written to disk on the target, both options are available in both modes, and are not exclusive. Source can send e-mail **and** write to a disk file, for example.

Source Server Reporting

Source server sends email reports via configured SMTP:

Report Contents:

- Datasets replicated
- Snapshot names and sizes
- Total data transferred
- Any errors or warnings

Configuration:

```
source:
  report:
    subject: "Sneakernet Replication Report"
    emailTo: admin@example.com
    emailFrom: backup@example.com
```

Target Server Reporting

Target server writes reports to removable media:

Report Drive Configuration:

```
target:
```

```
  report:
    targetDrive:
      label: report_drive
      fstype: ufs
      mountPoint: /mnt/report
```

Report Contents:

- Timestamp of operation
- Datasets imported
- Maintenance script results
- Any errors or warnings

Operational Benefits

This implementation balances security with operational efficiency:

Security Advantages:

- No single point of key compromise
- Lost transport drive: data remains encrypted
- Lost operator key: server data still protected
- Automated key rotation capability
- Audit trail via detailed reports
- Decryption failure = automatic abort

Operational Advantages:

- Minimal site visits (monthly vs. weekly)
- No waiting time for processing
- Fully automated operation (no manual commands)
- Email reports from source (connected)
- Physical reports from target (air-gapped)
- Automated maintenance without network access
- Dry-run mode for testing changes

Troubleshooting

Common Issues

Problem	Solution
Transport drive not detected	Verify GPT label matches config, check `dmesg` for device
Decryption fails on target	Verify transport key matches on both servers
Serial.txt already exists	Previous run not completed on target, investigate
Serial.txt missing on target	Drive already processed or not created on source
GELI disks won't unlock	Verify operator key present, check XOR key generation

Problem	Solution
Server won't shutdown	Check `shutdownAfterReplication` config, review logs

Debug Mode

```
# Run with maximum verbosity and dry-run
sneakernet --dryrun -vvvvv 2>&1 | tee debug.log

# Check ZFS_Utils log file
tail -f /tmp/zfs_utils.log # location overwritten by sneakernet.conf.yaml

# Check sneakernet log file (if configured)
tail -f /path/to/sneakernet.log
```

Log Files

- Main log: Configured via `logFile` in YAML (default: `sneakernet.log`)
- ZFS_Utils log: `/tmp/zfs_utils.log` unless defined in sneakernet.conf.yaml
- Status file: Configured via `statusFile` (tracks last replicated snapshots)
- State file: Target only, records pre-update snapshot state

Version History

- **v1.5.3** (2026-02-05) - Added CLI debug option to set config debug level
- **v1.5.2** (2026-02-05) - Capture and report cleanup script errors on target; documentation updates
- **v1.5.1** (2026-02-04) - Always log cleanup script output (even when empty)
- **v1.5.0** (2026-01-29) - Maintenance mode flag to pause replication
- **v1.4.x** (2026-01-22 to 2026-01-28) - Month-based cleanup scheduling, one-shot cleanup, size estimation/validation, improved reporting and diagnostics
- **v1.3.x** (2026-01-18 to 2026-01-21) - Configuration/logging improvements, GELI refactor, report drive notifications, modular cleanup stats
- **v1.2.x** (2026-01-11 to 2026-01-17) - Serial.txt mechanism, cleanup script support, modular config, improved security and error handling
- **v1.1.x** (2025-12-17 to 2025-12-21) - Snapshot filtering, rollback capability, config validation, logging improvements
- **v1.0.1** (2025-12-15) - Added verbose logging control
- **v1.0** (2025-12-15) - Initial release
- **v0.1** (2025-12-10) - Development version

See `CHANGELOG.md` in the repository for complete revision history.

References

- Air Gap Server Concepts — Theoretical background and best practices
- FreeBSD GELI Documentation

- FreeBSD ZFS Handbook
- OpenSSL Documentation

Support

Repository: `http://svn.dailydata.net/svn/zfs_utils/trunk`
Author: R. W. Rodolico
License: BSD 2-Clause (FreeBSD License)
Company: Daily Data Inc.

For bug reports, feature requests, or questions, contact the repository maintainer via web form at https://dailydata.net/contact-us/.

Disclaimer

This document was edited for clarity and formatting by an AI agent (GitHub Copilot), and the content was reviewed for accuracy afterward.

From:
https://kb.unixservertech.com/ - **Unix Server Tech Knowledge Base**

Permanent link:
https://kb.unixservertech.com/unix/freebsd/system_builds/airgap/sneakernet_implementation

Last update: **2026/02/06 06:37**