

Secret Server - Hardened Android Admin Platform

"We may be virtual, but we have your back"

A secure, local-first admin platform for Android/Termux that provides encrypted remote administration

Overview

Secret Server is a hardened Android userspace admin system that:

Runs entirely in Termux (no root required)

Provides secure admin access from Linux laptops

Uses two-factor authentication (hotspot + human approval)

Establishes encrypted SSH tunnels for all traffic

Mounts phone filesystem for remote editing via SSHFS

Auto-reconnects if connection drops

Currently hosts a secret manager with autovivification-based storage

Key Features

Security

Two-factor authentication: WiFi hotspot password + manual approval

SSH key-based auth: Automatic key generation and deployment

Encrypted tunnels: All traffic over SSH (no plaintext HTTP)

Human-in-the-loop pairing: Physical confirmation required for new devices

No cloud dependencies: Everything runs locally

User Experience

One-command connection: Single script handles everything

Auto-reconnection: Monitors connection and recovers from drops

Terminal browser UI: Uses w3m for approval (no app switching)

SSHFS integration: Edit phone files from laptop IDE

Clear error messages: Helpful guidance when things go wrong

Architecture

Hardened Android userspace: Works within Android's restrictions

Modular design: Auth layer separate from applications

Extensible: Other apps can use the same auth infrastructure

Local-first: No external servers or dependencies

Project Status

First Generation (Current)

- Admin pairing and connection system
- SSH tunnel and SSHFS mounting
- Secret manager application (autovivification storage)
- Auto-start on phone boot
- Documentation and install scripts (in progress)

Second Generation (Planned)

Semantic edge applications

User-tier connections (non-admin)

Additional security hardening

Enhanced install automation

Components

Phone Side (Android/Termux)

auth_server.py - Pairing and admin endpoint (port 8080)

web_server.py - Secret manager Flask app (port 5001, localhost only)

Boot scripts for auto-start

Laptop Side (Linux)

ssh_admin_connection.py - Complete connection manager
Auto-generates SSH keys if needed
Handles pairing, tunneling, mounting, monitoring

Repository Structure

```
secret-server/  
■■■■ README.md  
■■■■ INSTALL.md  
■■■■ LLM_CONTEXT.md  
■■■■ admin-ssh-connection/  
■ ■■■■ ssh_admin_connection.py  
■■■■ android_mnt/                # SSHFS mount point  
    ■■■■ auth-server/            # Phone code (when mounted)  
        ■■■■ auth_server.py  
        ■■■■ web_server.py  
        ■■■■ lib/
```

Quick Start

Prerequisites

Phone: Android device with Termux + Termux:Boot

Laptop: Linux with nmcli, sshfs, Python 3

Installation

See INSTALL.md for complete instructions.

Basic Usage

On phone:

```
bashcd ~/auth-server
```

```
./auth_server.py
```

On laptop:

```
bashexport SSH_PHONE_PASSWORD="your-hotspot-password"
```

```
cd ~/secret-server/admin-ssh-connection
```

```
./ssh_admin_connection.py
```

First run triggers pairing flow. Subsequent runs connect automatically.

Technology Stack

Python 3 - Core application logic

Flask - Web server framework

SSH/OpenSSH - Encryption and authentication

SSHFS - Filesystem mounting

w3m - Terminal web browser

NetworkManager (nmcli) - WiFi management

Termux - Android Linux environment

Use Cases

Current

Secure secret/password management

Remote Android administration

Local-first encrypted storage

Development/debugging on phone from laptop

Future

Semantic search and AI-powered applications

Collaborative document editing (peer-to-peer)

Personal knowledge base management
Edge computing applications

Design Philosophy

Security by architecture - Not security theater
Local-first - Your data stays on your device
Minimal dependencies - Fewer moving parts, less to break
Human-centric - Clear UX, helpful errors
Platform agnostic - Works within Android's constraints
Extensible - Auth platform for multiple applications

Development History

Developed iteratively with AI assistance (Claude, NotebookLM, Gemini, Copilot) while working a

Android's permission restrictions
Termux environment limitations
Dynamic IP addressing in hotspots
Boot script reliability issues
LLM hallucinations and context scrambling

The project evolved from a simple secret manager into a general-purpose hardened admin platform

Contributing

This is currently a personal project, but feedback and suggestions are welcome. See installation

License

[To be determined]

Acknowledgments

Built with assistance from Claude (Anthropic), exploring the intersection of:

Android userspace hardening

Local-first software
Autovivification-based storage (inspired by Perl's approach)
Human-in-the-loop security

Note: android_mnt/ directory contains stale code in git. When SSHFS mounts, it overlays with 1