ScorpioX Security Assessment Report

ScorpioX Security Assessment Report

Comprehensive Security Audit — Unified Report

Project	ScorpioX (clang codebase)
Commit	`b56a30721d22391836e8145ca16fe0e24250a148`
Branch	`main`
Audit Date	2026-04-28
Lines of Code	369,411 (C/H files)
Auditors	Architecture Agent, Network Agent, Data Handling Agent, Vulnerability Agent, Permissions Agent
Classification	CONFIDENTIAL — Internal Use Only

---

1. Executive Summary

This report presents the consolidated findings of a comprehensive security assessment of the ScorpioX C codebase — a TUI-based AI coding assistant platform comprising approximately 369,000 lines of C code, 150 source files, and ~65 statically-linked executables.

Key Security Strengths

- **Zero external package manager dependencies** for the core C build — all third-party code (mbedTLS, yyjson) is vendored from source
- **Single-organization code provenance** — all 987 commits originate from ScorpioX organization; no external contributors
- **Fully static linking** on Linux reduces runtime dependency surface
- **No privilege escalation** — zero `setuid` / `setgid` / `seteuid` / `setegid` calls anywhere in the codebase
- **Strong container isolation** using user namespaces, `pivot_root`, overlayfs, PID/network/mount isolation
- **No executable memory** — no JIT, no `mprotect(PROT_EXEC)`, strong W^X compliance
- **Consistent use of safe string functions** — 2,228 `snprintf` calls, zero `sprintf` calls

Critical Findings Requiring Immediate Attention

#	Finding	Impact
1	**Hardcoded API keys and SSH credentials** in WASM embedded config (`sx_config_embedded.c`)	Credential compromise in compiled binaries
2	**Remote Code Execution** via command injection in HTTP server OTP endpoint (`scorpiox-server.c`)	Unauthenticated RCE on server deployments

3	**OAuth tokens transmitted over plaintext TCP** to `20.53.240.54:9800`	Token interception on network
4	**TLS certificate verification disabled** on all mbedTLS connections	Man-in-the-middle exposure

Overall Risk Rating: MEDIUM-HIGH

The core application architecture is fundamentally sound with excellent supply chain hygiene. However, critical findings around hardcoded credentials, a command injection vulnerability, and disabled TLS verification elevate the overall risk. These are addressable issues that, once remediated, would bring the codebase to a LOW risk posture.

Findings by Severity

Severity	Count
Critical	5
High	15
Medium	22
Low	9
Info	2
Total	**53**

---

2. Application Overview

ScorpioX is a systems-level AI coding assistant platform written entirely in C. It provides:

- **TUI Interface** (`sx`) — A terminal-based interactive AI assistant client
- **Multi-Provider AI Integration** — Supports Anthropic Claude, OpenAI GPT/Codex, Google Gemini, and Z.AI
- **Session Multiplexer** (`sxmux`) — Terminal multiplexer for managing multiple AI sessions
- **Container Runtime** (`scorpiox-unshare`) — Rootless container system using Linux user namespaces
- **Email Server** (`sxmail`) — SMTP server with STARTTLS, Maildir storage, and DKIM signing
- **DNS Server** (`scorpiox-dns`) — Authoritative DNS with upstream forwarding
- **HTTP Server** (`scorpiox-server`) — Embedded web server with JWT authentication and CGI
- **Virtual Machine** (`scorpiox-vm`) — KVM-based micro-VM runner
- **SDK & Agent Framework** — Tools for building and orchestrating AI agents
- **Networking Tools** — FRP reverse proxy client, traffic capture, HTTP relay, WebSocket bridge

The project compiles to approximately 65 statically-linked executables from a single CMake build. The architecture uses three shared static libraries (`libsxutil` , `libsxnet` , `libsxui`) for code reuse across tools.

Technology Stack

Layer	Technology
Language	C (C11)
Build System	CMake 3.16+
TLS/Crypto	mbedTLS (vendored, custom minimal config)
JSON	yyjson (vendored)

HTTP Client	libcurl (system)
Linking	Fully static on Linux
Target Platforms	Linux (primary), macOS, Windows, WASM

---

3. Architecture & Supply Chain Assessment

Source: 01-architecture.md

3.1 Dependency Model

The ScorpioX core build achieves **zero external package manager dependencies** — a significant security advantage. All third-party C code is committed directly into the repository:

Vendored Library	Location	License	Lines
mbedTLS	`scorpiox/vendor/mbedtls/`	Apache-2.0 / GPL-2.0+	206,340
yyjson	`scorpiox/vendor/yyjson/`	MIT	19,556

The mbedTLS vendored copy uses a **custom minimal configuration** (`sx_mbedtls_config.h`) enabling only AES-128-CFB, TLS 1.2, PBKDF2-SHA1, and X.509 — substantially reducing the cryptographic attack surface.

System libraries (libcurl, OpenSSL, zlib) are found at build time via `find_static_library()` and `pkg-config` , not downloaded.

3.2 Build Security

- **No downloaded toolchains** — uses system compiler (GCC/Clang)
- **No** `FetchContent` / `ExternalProject` — no network access during build
- **Compiler hardening flags**: `-Wall -Wextra -ffunction-sections -fdata-sections`
- **Static linking**: `-static` on Linux via `set(CMAKE_EXE_LINKER_FLAGS "-static")`
- **Build-time code generation** disabled by default (no network calls)

3.3 Code Provenance

All 987 commits originate from 6 author identities, all mapping to two related domains (`scorpiox.net` , `scorpioplayer.com`). **No external contributors detected.**

3.4 Supply Chain Findings

Finding	Severity	Status
Core C build: zero package manager dependencies	Info	✅ Strength
Vendored mbedTLS & yyjson: auditable, pinned	Info	✅ Strength
`bridge/package.json` : 2 npm dependencies (WhatsApp bridge only)	Medium	⚠ Acknowledged
2 pre-built ELF binaries in `bridge/` without hash verification	Medium	⚠ Open
Vendored library versions not documented for CVE tracking	Low	⚠ Open

---

4. Network Security Assessment

Source: 02-network.md

4.1 Network Activity Overview

The application is network-intensive with 13 categories of network activity spanning AI provider APIs, OAuth token acquisition, telemetry, software distribution, reverse proxy, SMTP, DNS, web search, and WebSocket bridging.

All AI provider communications use HTTPS (TLS 1.2+) to well-known API endpoints (Anthropic, OpenAI, Google, Z.AI).

4.2 Endpoint Inventory

The audit identified **30+ hardcoded endpoints** across the codebase. Key categories:

Category	Count	Protocol	Risk
AI Provider APIs	8	HTTPS	Low
OAuth/Token endpoints	4	Mixed (HTTP/TCP/SSH)	Critical-High
Telemetry endpoints	2	HTTPS	Medium
Distribution/Update	5	HTTPS	Medium
Internal infrastructure	6	Mixed	High
Services (DNS, SMTP, HTTP)	5	Various	Medium

4.3 Critical Network Findings

OAuth Token over Plaintext TCP: `scorpiox-claudecode-fetchtoken.c` transmits OAuth tokens over unencrypted TCP to `20.53.240.54:9800`. No integrity or confidentiality protection exists.

Token Endpoints over HTTP: `token.scorpiox.net/claude`, `/codex`, `/gemini` use plaintext HTTP for authentication token retrieval.

TLS Certificate Verification Disabled: All mbedTLS connections (FRP, SMTP relay, SMTP server) use `MBEDTLS_SSL_VERIFY_NONE`, enabling man-in-the-middle attacks.

FRP Authentication Uses MD5: The reverse proxy client authenticates using MD5, a cryptographically broken hash. Additionally, PBKDF2 key derivation uses only 64 iterations (recommended minimum: 600,000).

4.4 Telemetry

Two telemetry channels operate automatically:

1. **Usage Tracking** (`code.scorpiox.net/usage-send`) — sends session_id, hostname, username, OS, architecture, provider, model, token counts
2. **Session Emit** (`code.scorpiox.net/sessions-send`) — when `EMIT_SESSION_TRACKING=1`, sends full conversation content

Both use HTTPS and fire-and-forget POST requests. Usage telemetry includes machine fingerprint data (hostname, username, OS).

4.5 No Covert Exfiltration

The audit confirmed **no patterns of covert data exfiltration**. All network transmissions serve documented purposes and are either user-initiated or controlled by explicit configuration flags.

5. Data Handling & Privacy Assessment

Source: 03-data-handling.md

5.1 Data Flow Architecture

```
User Input → TUI (sx.c)
    ├→ Configuration (sx_config.c) ← scorpiox-env.txt (3 cascade levels)
    ├→ Provider (sx_provider_*.c) → HTTP/TLS → External API
    │     ├→ Traffic Logs (.scorpiox/sessions/<id>/traffic/)
    │     └→ API Keys from config / env vars / embedded config
    ├→ Session Manager (sx_session.c)
    │     ├→ conversation.json, session.log, events.jsonl, trace.jsonl
    │     ├→ config-snapshot.txt (frozen config)
    │     └→ meta.json (metadata)
    ├→ Conversation Store (.scorpiox/conversations/*.json)
    ├→ Email Server (sxmail_*.c) → Maildir format
    ├→ Multiplexer (sxmux_*.c) → Unix domain sockets
    ├→ Usage Tracking → POST to code.scorpiox.net/usage-send
    └→ Session Emit → POST to code.scorpiox.net/sessions-send
```

5.2 Credential Management

The codebase uses a **three-level configuration cascade** for API keys and credentials:

1. **Executable directory** `scorpiox-env.txt` (global)
2. **User home** `~/.claude/scorpiox-env.txt` (user-level override)
3. **Working directory** `.scorpiox/scorpiox-env.txt` (project-level override)

All configuration files store secrets in **plaintext** key-value format. No encryption-at-rest is applied.

59 unique environment variables are read via `getenv()`, including critical authentication variables (`ANTHROPIC_API_KEY`, `GEMINI_API_KEY`, `OPENAI_API_KEY`, `AGENTS_PAT`).

5.3 Encryption

- **In Transit**: All AI provider traffic uses HTTPS (TLS 1.2+ via libcurl). mbedTLS handles TLS for FRP, SMTP, and custom connections.
- **At Rest**: **No encryption** for conversations, sessions, traffic captures, email storage, or configuration files.
- **Password Hashing**: Email server uses **unsalted SHA256** — inadequate for password storage (should use bcrypt/scrypt/Argon2).
- **FRP Encryption**: AES-128-CFB with PBKDF2-SHA1 key derivation (64 iterations — critically low).

5.4 Data Retention

- **Session cleanup**: Automated via `SESSION_RETENTION_DAYS` (default: 7 days) ✅
- **Conversations**: Accumulate indefinitely — no automated cleanup ⚠
- **Traffic captures**: Manual cleanup only (`--clear` flag) ⚠
- **Email/DNS logs**: Grow indefinitely — no rotation configured ⚠
- **Temp files**: Rely on OS cleanup ⚠

5.5 Privacy Findings

Finding	Severity	Details
Hardcoded API keys & SSH	Critical	Compiled into WASM binary; includes Anthropic, Z.AI, Gemini

passwords in embedded config		keys and SSH password “xboxone”
Plaintext API keys in config files	High	Config files at 0755 directory permissions
Unsalted SHA256 password hashing	High	Email auth vulnerable to rainbow table attacks
Config snapshot may contain API keys	High	Session snapshots dump config without redaction
Predictable temp file names	Medium	`/tmp/sx_voice.wav`, `/tmp/sx_emit_*` — symlink attack vector
No encryption at rest for sensitive data	Medium	Conversations, sessions, email stored in plaintext
Usage telemetry includes machine fingerprint	Medium	Hostname, username, OS sent to server

---

6. Code Security & Vulnerability Assessment

Source: 04-vulnerabilities.md

6.1 Vulnerability Summary

The audit identified **23 distinct vulnerability findings** across command injection, buffer overflow, memory safety, and race condition categories.

6.2 Critical & High Vulnerabilities

CRITICAL: Remote Code Execution — OTP Command Injection

File: `scorpiox-server.c`, Lines 2480-2486
CVSS: 9.8
Unsanitized query parameters from the OTP endpoint are injected directly into a `system()` shell command, allowing unauthenticated remote code execution.

HIGH: Command Injection via URL/Path in Container Runtime

File: `scorpiox-unshare.c`, Lines 309-312
CVSS: 8.1
URL/path values wrapped in single quotes and passed to `system()` via `curl` / `wget`. Single-quote injection possible.

HIGH: Command Injection via Maildir Deletion

File: `sxmail_maildir.c`, Lines 704-711
CVSS: 7.5
User-derived path injected into `rm -rf '%s'` / `rmdir /s /q` shell command.

HIGH: Path Traversal in HTTP Server

File: `scorpiox-server.c`, Lines 1192-1222
CVSS: 7.5
`url_decode()` applied before path validation without post-decode `..` check.

HIGH: Command Injection via `EDITOR`

File: `sx.c`, Lines 655-659
CVSS: 7.2
`EDITOR` environment variable injected into `system()` without sanitization.

6.3 Medium Vulnerabilities (12 findings)

Category	Count	Key Issues
Memory leaks	2	Null-check logic errors in `sx_term.c`, `sxmux_vt.c`
Missing NULL checks	1	50+ `malloc()` calls without NULL validation
Return type mismatches	2	`return -1` in void functions, int/pointer confusion
Integer overflow	3	Unchecked multiplication before `malloc()` in multiple files
Unsafe string operations	2	28 `strcpy()` calls, weak `..` check in `ws2tcp.c`
Stack overflow risk	1	Stack buffers ≥64KB (5 locations with `char[65536]`)
Diagnostic alloc	1	5 `malloc()` calls without NULL checks in heap diagnostic

6.4 Low Vulnerabilities (5 findings)

Category	Count	Key Issues
Memory leak on error path	1	Scrollback alloc failure in `sxmux_vt.c`
Race condition	1	Volatile flag access without mutex in `sx_dll.c`
Logic error	1	Always-false comparison in `sxmux_session.c`
Signal handling	1	`signal()` instead of `sigaction()`
TOCTOU	1	`stat()` then `open()` patterns

6.5 Positive Security Observations

- **Zero `sprintf()` usage** — exclusively `snprintf()` (2,228 calls)
- **No format string vulnerabilities** — all format strings are literals
- **No `gets()` usage** — all input functions are bounded
- **Consistent `-Wall -Wextra`** compiler warnings enabled

---

7. Permissions & System Access Assessment

Source: 05-permissions.md

7.1 Privilege Model

The vast majority of ScorpioX tools (~60+) run as **fully unprivileged processes**. Only 3-4 tools require or benefit from root access:

Component	Root Required	Reason
`scorpiox-thunderbolt4`	**Yes** (enforced)	BPF access for raw Ethernet frames
`scorpiox-dns`	Practical yes	Port 53 binding

`sxmail_smtp`	Practical yes	Port 25 binding
`scorpiox-vm`	No (KVM group)	`/dev/kvm` access

No privilege escalation: Zero `setuid` / `setgid` / `seteuid` / `setegid` calls exist in the entire codebase. The code never escalates its own privileges.

7.2 Process Spawning

42 source files spawn child processes. The codebase uses 164 total `system()` / `popen()` calls — a systemic risk surface for command injection.

Mechanism	Usage	Risk
`clone()` with namespace flags	Container creation	Core functionality; well-isolated
`fork()` + `exec*()`	Standard command execution	Safe pattern
`system()`	Shell command execution (91 calls)	⚠ Injection risk if inputs not sanitized
`popen()`	Capture command output (73 calls)	⚠ Same injection risk as `system()`

7.3 Container Security

The `scorpiox-unshare` container runtime implements industry-standard isolation:

- **User namespaces** — rootless operation by default
- **PID namespace** — process isolation
- **Network namespace** — network isolation (optional)
- **Mount namespace** — filesystem isolation via `pivot_root` + overlayfs
- **UTS namespace** — hostname isolation
- **Read-only host bind mounts** — host binaries mounted read-only

Missing hardening: - No `seccomp` filtering - No `PR_SET_NO_NEW_PRIVS` - No capability dropping

7.4 Environment Modification

The `scorpiox-traffic` tool intentionally modifies TLS-related environment variables (`HTTP_PROXY`, `NODE_TLS_REJECT_UNAUTHORIZED=0`, etc.) for its MITM capture functionality. These modifications are correctly scoped to the child process only, not applied globally.

No `LD_PRELOAD` or `LD_LIBRARY_PATH` modification exists anywhere — no dynamic library injection vectors.

7.5 Filesystem Access

The codebase accesses a broad set of filesystem paths:

Path Category	Examples	Purpose
Application data	`~/.scorpiox/` , `~/.claude/`	Config, sessions, conversations
Temporary files	`/tmp/sxmux/` , `/tmp/sx_*`	IPC, temp storage
System paths	`/proc/` , `/sys/` , `/dev/`	Container setup, KVM, diagnostics
Container rootfs	`/var/lib/scorpiox/`	Container images and overlays

---

8. Consolidated Risk Matrix

8.1 Critical Findings (5)

#	Area	Finding	Severity	Status	Recommendation
C1	Data/Network	Hardcoded API keys (Anthropic, Z.AI, Gemini) and SSH password (“xboxone”) in `sx_config_embedded.c`	**CRITICAL**	Open	Remove all secrets from source code; use runtime-only configuration; rotate all exposed keys immediately
C2	Vulnerability	OTP endpoint command injection in `scorpiox-server.c` (CVSS 9.8) — unauthenticated RCE	**CRITICAL**	Open	Sanitize query parameters; restrict to `[a-zA-Z0-9@._-]` ; replace `system()` with `execvp()`
C3	Network	OAuth tokens transmitted over plaintext TCP to `20.53.240.54:9800`	**CRITICAL**	Open	Require TLS for all token transmission
C4	Network	SSH credentials hardcoded — root password for `192.168.1.6:22223`	**CRITICAL**	Open	Remove from source; use SSH key-based auth
C5	Network	Token endpoints (`token.scorpiox.net`) use HTTP not HTTPS	**CRITICAL**	Open	Upgrade all authentication endpoints to HTTPS

8.2 High Findings (15)

#	Area	Finding	Severity	Status	Recommendation
H1	Network	TLS certificate verification disabled on all mbedTLS connections	**HIGH**	Open	Replace `MBEDTLS_SSL_VERIFY_NONE` with `MBEDTLS_SSL_VERIFY_REQUIRED` ; load CA certs
H2	Vulnerability	Command injection via URL/path in `scorpiox-unshare.c` (CVSS 8.1)	**HIGH**	Open	Use `fork()` + `exec()` instead of `system()` with quoted strings
H3	Vulnerability	Command injection in maildir deletion `sxmail_maildir.c` (CVSS 7.5)	**HIGH**	Open	Use `nftw()` recursive delete or `fork()` + `exec("rm")`
H4	Vulnerability	Path traversal in HTTP server `scorpiox-server.c` (CVSS 7.5)	**HIGH**	Open	Add `realpath()` canonicalization with webroot prefix check
H5	Vulnerability	Command injection via `EDITOR` in `sx.c` (CVSS 7.2)	**HIGH**	Open	Sanitize editor path or use `execlp()` directly
		FRP auth uses MD5 —			

H6	Network	cryptographically broken	**HIGH**	Open	Replace with SHA-256 or better
H7	Network	FRP PBKDF2 uses only 64 iterations (min recommended: 600,000)	**HIGH**	Open	Increase to ≥600,000 iterations
H8	Network	Session telemetry can send full conversation content	**HIGH**	Acknowledged	Document; ensure opt-in only with user consent
H9	Network	Private network IPs hardcoded in WASM binary	**HIGH**	Open	Remove internal network topology from published binaries
H10	Data	Plaintext API keys in config files with 0755 directory permissions	**HIGH**	Open	Encrypt sensitive values or use OS keychain; restrict to 0600
H11	Data	Unsalted SHA256 password hashing for email auth	**HIGH**	Open	Replace with bcrypt/scrypt/Argon2 with per-user salt
H12	Data	Config snapshot may write API keys to session directory	**HIGH**	Open	Redact sensitive config keys before writing snapshot
H13	Data	Raw TCP token fetch without TLS	**HIGH**	Open	Require TLS for all credential transmission
H14	Permissions	Excessive `system()` usage (91 calls) — systemic injection risk	**HIGH**	Open	Create shared `sx_safe_exec()` utility using `fork()` + `execvp()`
H15	Permissions	TLS bypass environment variables in `scorpiox-traffic`	**HIGH**	Mitigated	Scoped to child process only; by-design for MITM tool

8.3 Medium Findings (22)

#	Area	Finding	Severity	Status	Recommendation
M1	Architecture	`bridge/package.json` — 2 npm dependencies for WhatsApp bridge	**MEDIUM**	Acknowledged	Isolate bridge; consider separate repo
M2	Architecture	2 pre-built ELF binaries without hash verification	**MEDIUM**	Open	Add CI step to verify checksums against source
M3	Network	Usage telemetry sends machine fingerprint	**MEDIUM**	Acknowledged	Minimize PII; add opt-out mechanism
M4	Network	Auto-update downloads executables without signature verification	**MEDIUM**	Open	Add binary signature verification
M5	Network	Git PAT tokens embedded in clone URLs	**MEDIUM**	Acknowledged	Use credential helpers instead
		HTTP relay uses custom			Add authentication to relay

M6	Network	binary protocol without authentication	**MEDIUM**	Open	protocol
M7	Network	SMTP AUTH PLAIN sends credentials in base64	**MEDIUM**	Acknowledged	Standard SMTP behavior over STARTTLS — acceptable
M8	Network	Router URL over plaintext HTTP	**MEDIUM**	Open	Upgrade to HTTPS
M9	Network	Cache keepalive generates ongoing API traffic	**MEDIUM**	Acknowledged	Inform users of background API costs
M10	Vulnerability	Memory leak in `sx_term.c` null-check logic error	**MEDIUM**	Open	Fix null-check ordering
M11	Vulnerability	50+ `malloc()` calls without NULL validation	**MEDIUM**	Open	Add NULL checks after all allocations
M12	Vulnerability	`return -1` in void functions (`scorpiox-server.c`)	**MEDIUM**	Open	Fix return types; add cleanup on error paths
M13	Vulnerability	Type confusion in `scorpiox-mcp.c` — `int` from `char*` function	**MEDIUM**	Open	Fix return type
M14	Vulnerability	Integer overflow before `malloc()` in multiple files	**MEDIUM**	Open	Add overflow guards before multiplication
M15	Vulnerability	`strcpy()` used 28 times — some with computed offsets	**MEDIUM**	Open	Replace with `snprintf()` or bounded copies
M16	Vulnerability	Weak `..` path traversal check in `ws2tcp.c`	**MEDIUM**	Open	Use `realpath()` canonicalization
M17	Vulnerability	Stack buffers ≥64KB (5 locations)	**MEDIUM**	Open	Replace with heap allocations
M18	Data	Predictable temp file names	**MEDIUM**	Open	Use `mkstemp()` for all temporary files
M19	Data	No encryption at rest for conversations/sessions/email	**MEDIUM**	Acknowledged	Consider encrypted storage for sensitive data
M20	Permissions	No seccomp in containers	**MEDIUM**	Open	Add seccomp filter for container child processes
M21	Permissions	No `PR_SET_NO_NEW_PRIVS` in containers	**MEDIUM**	Open	Call `prctl(PR_SET_NO_NEW_PRIVS, 1)` before exec
M22	Permissions	Unbounded CGI process spawning in HTTP server	**MEDIUM**	Open	Add connection limits and child process caps

8.4 Low Findings (9)

#	Area	Finding	Severity	Status	Recommendation
L1	Architecture	Vendored library versions not documented for CVE tracking	**LOW**	Open	Create `VENDOR_VERSIONS.md`
L2	Network	DNS server defaults to Google/Cloudflare upstream	**LOW**	Acknowledged	Privacy consideration; configurable
L3	Network	WebSocket bridge has no authentication	**LOW**	Open	Add authentication mechanism

L4	Network	Traffic logging writes raw HTTP bodies to disk	**LOW**	Acknowledged	By-design for debugging
L5	Vulnerability	Memory leak on scrollback alloc failure	**LOW**	Open	Free `vt->cells` before returning NULL
L6	Vulnerability	Race condition on volatile flag in `sx_dll.c`	**LOW**	Open	Add mutex protection
L7	Vulnerability	Always-false comparison in `sxmux_session.c`	**LOW**	Open	Fix data type or comparison
L8	Vulnerability	`signal()` instead of `sigaction()`	**LOW**	Open	Migrate to `sigaction()`
L9	Permissions	Predictable Unix socket paths at `/tmp/sxmux/`	**LOW**	Open	Check socket ownership; use per-user subdirs

8.5 Informational (2)

#	Area	Finding	Severity	Status	Recommendation
I1	Architecture	Core C build has zero package manager dependencies	**INFO**	✅ Strength	Maintain this approach
I2	Architecture	Single-organization code provenance (987 commits)	**INFO**	✅ Strength	Continue contributor governance

---

9. Conclusion & Recommendations

9.1 Overall Assessment

The ScorpioX codebase demonstrates a **mature, security-conscious architecture** with notable strengths in supply chain security, dependency management, and memory safety practices. The decision to vendor all third-party code, use static linking, and maintain single-organization provenance places the project well ahead of typical open-source C projects in supply chain security.

However, the audit identified **5 critical and 15 high-severity findings** that require attention. The most impactful are:

1. **Hardcoded credentials in source code** — This is the single most urgent issue. API keys and SSH passwords compiled into binaries must be rotated immediately.
2. **Remote Code Execution** — The OTP endpoint command injection in the HTTP server could allow unauthenticated remote attackers to execute arbitrary commands.
3. **TLS security gaps** — Disabled certificate verification, plaintext token transmission, and weak cryptographic primitives (MD5, 64-iteration PBKDF2) undermine the confidentiality and integrity of network communications.

9.2 Prioritized Remediation Roadmap

Immediate (Week 1) — Critical

1. **Rotate all embedded credentials** — API keys, SSH passwords in `sx_config_embedded.c` are compromised
2. **Fix OTP command injection** — Sanitize inputs in `scorpiox-server.c:2480`; replace `system()` with `execvp()`
3. **Enable TLS for token fetch** — Replace raw TCP with HTTPS for OAuth token acquisition
4. **Upgrade token endpoints to HTTPS** — `token.scorpiox.net` must use TLS

Short Term (Weeks 2-4) — High

5. **Enable TLS certificate verification** on all mbedTLS connections
6. **Fix remaining command injection** vulnerabilities (unshare, maildir, `EDITOR`)
7. **Fix path traversal** in HTTP server with `realpath()` validation
8. **Replace MD5 in FRP** with SHA-256; increase PBKDF2 to ≥600,000 iterations
9. **Create `sx_safe_exec()` utility** — centralized safe process execution
10. **Upgrade password hashing** to bcrypt/Argon2 with per-user salts
11. **Redact config snapshots** — strip sensitive keys before writing

Medium Term (Months 2-3) — Medium

12. **Add NULL checks** after all allocation calls
13. **Fix integer overflow guards** before allocation size multiplication
14. **Add seccomp +** `PR_SET_NO_NEW_PRIVS` to container runtime
15. **Add binary signature verification** for auto-updates
16. **Replace `strcpy()` with bounded alternatives**
17. **Move large stack buffers to heap**
18. **Use `mkstemp()` for temporary files**

Long Term (Quarter 2) — Low/Hardening

19. **Migrate from `signal()` to `sigaction()`**
20. **Document vendored library versions** for CVE tracking
21. **Add `-Wformat-security -Werror=return-type`** to CI build flags
22. **Implement TOCTOU-safe patterns** with `openat()` / `fstatat()`
23. **Add data retention policies** for conversations, traffic captures, logs

9.3 Risk Forecast

Once the Critical and High findings are remediated, the overall risk rating is expected to drop from **MEDIUM-HIGH** to **LOW**. The fundamental architecture is sound, and the identified issues are implementation-level fixes rather than architectural redesigns.

---

10. Appendix: Audit Methodology

10.1 Scope

Parameter	Value
Codebase	ScorpioX clang repository
Commit	`b56a30721d22391836e8145ca16fe0e24250a148`
Branch	`main`
Total files analyzed	493 (.c/.h), 224 non-vendor
Lines of code	369,411 total; ~143,515 non-vendor

Exclusions	`scorpiox/vendor/mbedtls/` , `scorpiox/vendor/yyjson/` (audited as vendored dependencies only)

10.2 Audit Agents

Agent	Scope	Report
Architecture Agent	Dependencies, build process, supply chain, code provenance, binary analysis	01-architecture.md
Network Agent	Endpoints, protocols, TLS, telemetry, data-in-transit, DNS	02-network.md
Data Handling Agent	File I/O, credentials, encryption, PII, data retention, privacy	03-data-handling.md
Vulnerability Agent	Buffer overflow, command injection, memory safety, race conditions, format strings	04-vulnerabilities.md
Permissions Agent	Privilege model, process spawning, filesystem access, sandboxing, environment	05-permissions.md

10.3 Techniques

- **Static analysis**: grep-based pattern matching for unsafe functions (`system` , `popen` , `sprintf` , `strcpy` , `gets` , `strcat`)
- **Compiler analysis**: GCC 13 with `-Wall -Wextra -Wformat-security -Wformat=2`
- **Endpoint enumeration**: Manual extraction of all hardcoded URLs, IPs, and ports
- **Dependency analysis**: Recursive search for package manager files across all known ecosystems
- **Binary analysis**: `file` , `readelf` , `strings` on pre-built binaries
- **Git forensics**: Author analysis across all 987 commits
- **Configuration review**: Analysis of all config file formats, cascade logic, and secret handling
- **Process trace**: Enumeration of all `fork()` , `exec*()` , `system()` , `popen()` , `clone()` call sites
- **Risk scoring**: CVSS 3.1 base scores adapted for application context

10.4 Limitations

- This is a **source code audit only** — no dynamic testing, fuzzing, or penetration testing was performed
- Vendored libraries (mbedTLS, yyjson) were assessed for integration risk but not subjected to full vulnerability analysis
- The WASM build target was not tested in a runtime environment
- Network endpoints were identified from source; actual runtime behavior was not captured

---

Report compiled 2026-04-28 by Security Audit Agent — ScorpioX Security Assessment