Third-Party Software Security Review — ScorpioX Code

Software: ScorpioX Code
Type: AI-Powered Development Tool / CLI Platform
Language: Pure C (zero external dependencies)
Audit Date: 2026-04-28
Codebase Commit: `2b6f3d37e8fc5ffacf967ddcf2682415887cb525`
Classification: CONFIDENTIAL — For Corporate Review Only

---

1. Executive Summary

This report presents the findings of a comprehensive security audit of ScorpioX Code, an AI-powered development tool and CLI platform written in pure C. The audit was conducted by 13 specialized automated agents, each targeting a specific security domain. The codebase comprises **399,383 lines of code** (171,633 first-party, 227,750 vendored) across 492 source files.

Overall Risk Rating: HIGH

The audit identified **240 total findings** across all domains:

Severity	Count
Critical	17
High	28
Medium	68
Low	53
Informational	74
Total	**240**

Key Risk Areas:

1. **TLS/Certificate Verification (CRITICAL):** SSL certificate verification is systematically disabled across most cURL-based HTTP clients and all mbedTLS client connections. Token-fetching modules transmit authentication credentials over plaintext HTTP. Only two modules correctly enforce certificate verification.

2. **Command Injection (HIGH):** Windows `cmd.exe` injection vectors exist in the HTTP server component via unsanitized HTTP query parameters and JWT claims. A web search module on Windows passes user queries into `popen()` without shell escaping.

3. **Install & Distribution (HIGH):** The primary install command lacks an HTTPS scheme prefix. Release builds generate no SHA256 checksums. No code signing exists. SHA256 verification is silently skipped when sidecar files are unavailable.

4. **Memory Safety (MEDIUM):** 46 findings including dangerous realloc patterns that lose pointers on failure, memory leaks on error paths, and 23+ unchecked `strdup` return values in API handling code.

5. **Network Endpoints (CRITICAL):** 149 hardcoded URL references in first-party code, including 4 plaintext HTTP endpoints transmitting authentication tokens and a hardcoded public IP address compiled into the WASM binary.

Key Strengths:

- Zero external package manager dependencies for the core C build
- Static linking reduces runtime dependency attack surface
- No setuid/setgid binaries; no calls to `setuid()` / `setgid()`
- 99.6% safe string operation ratio (2,841 safe vs. 12 unsafe calls)
- Zero `sprintf`, `strcpy`, `gets`, or `scanf` usage in first-party code
- All telemetry disabled by default with explicit opt-in required
- Secure `mkstemp()` used throughout for temp file creation
- Strong compiler hardening flags (`-fstack-protector-strong`, `-D_FORTIFY_SOURCE=2`, `-fcf-protection`)

---

2. Deployment Scope — Windows Workstation

2.1 Primary Deployment: Windows Workstation

The software is evaluated for deployment on **Windows workstations**. Server-side components (email server, container runtime, VM hypervisor) are **not in scope** for this deployment.

2.2 Windows Binaries (56 executables + 1 DLL)

The following binaries compile for and ship on Windows:

Category	Binaries
Core	`sx.exe`, `libsx.dll`, `scorpiox-agent.exe`, `scorpiox-sdk.exe`
Shell & Utilities	`scorpiox-bash.exe`, `scorpiox-busybox.exe`, `scorpiox-vi.exe`, `scorpiox-config.exe`
Search & Web	`scorpiox-websearch.exe`, `scorpiox-fetch.exe`, `scorpiox-search.exe`
Display	`scorpiox-screenshot.exe`, `scorpiox-renderimage.exe`
PowerShell & WSL	`scorpiox-pwsh.exe`, `scorpiox-wsl.exe`
API Providers	`scorpiox-openai.exe`, `scorpiox-gemini.exe`
Token Management	`scorpiox-claudecode-fetchtoken.exe`, `scorpiox-claudecode-refreshtoken.exe`, `scorpiox-claudecode-models.exe`, `scorpiox-codex-fetchtoken.exe`, `scorpiox-codex-refreshtoken.exe`, `scorpiox-gemini-fetchtoken.exe`
Server & Hosting	`scorpiox-server.exe`, `scorpiox-host.exe`, `scorpiox-server-http2tcp.exe`
Infrastructure	`scorpiox-dns.exe`, `scorpiox-frp.exe`, `scorpiox-mcp.exe`, `scorpiox-hook.exe`
Session & Logging	`scorpiox-logger.exe`, `scorpiox-printlogs.exe`, `scorpiox-emit-session.exe`, `scorpiox-transcript.exe`, `scorpiox-rewind.exe`
	`scorpiox-beam.exe`, `scorpiox-conv.exe`, `scorpiox-debug.exe`, `scorpiox-docs.exe`, `scorpiox-`

Other	`email.exe`, `scorpiox-kql.exe`, `scorpiox-mirror-git.exe`, `scorpiox-multiplexer.exe`, `scorpiox-otp.exe`, `scorpiox-planmode.exe`, `scorpiox-askuserquestion.exe`, `scorpiox-runtest.exe`, `scorpiox-skills.exe`, `scorpiox-systemprompt.exe`, `scorpiox-tasks.exe`, `scorpiox-tmux.exe`, `scorpiox-usage.exe`, `scorpiox-vault-git.exe`, `scorpiox-voice.exe`, `scorpiox-server-email.exe`, `scorpiox-server-fetchtoken.exe`

2.3 Linux-Only Binaries (Not Deployed on Windows)

Binary	Reason
`scorpiox-cron`	Linux cron integration
`scorpiox-imessage`	macOS-only iMessage bridge
`scorpiox-init`	Linux init system integration
`scorpiox-podman`	Linux container management
`scorpiox-sshpass`	Linux SSH utility
`scorpiox-thunderbolt4`	Linux Thunderbolt4 raw interface
`scorpiox-traffic`	Linux traffic capture
`scorpiox-unshare`	Linux user namespace container runtime
`scorpiox-vm`	Linux KVM virtual machine runtime
`scorpiox-whatsapp`	Linux WhatsApp bridge

2.4 Windows-Filtered Findings

When scoped to Windows workstation deployment only (excluding Linux-only binaries and server-only components):

Severity	All Platforms	Windows Only
Critical	17	15
High	28	25
Medium	68	63
Low	53	49
Informational	74	69
Total	**240**	**221**

---

3. Changes Since Last Audit

Previous Audit: `9bfa5b461aa008cd7a1713d37c9aa6c44cecc4ce` (2026-04-28)
Current Audit: `2b6f3d37e8fc5ffacf967ddcf2682415887cb525` (2026-04-28)
Lines of Code Change: 378,558 → 399,383 (+20,825 lines, +5.5%)

Severity	Previous	Current	Delta
Critical	13	17	**+4**
High	33	28	**−5**
Medium	120	68	**−52**

Low	28	53	**+25**
Info	355	74	**−281**
Total	549	240	**−309**

Previous Overall Risk: HIGH
Current Overall Risk: HIGH

Key Changes:

- **Findings reduced by 56%** overall (549 → 240), primarily from reduced informational and medium findings due to improved audit methodology precision
- **Critical findings increased by 4** — new install-script agent identified 3 critical distribution security issues not previously audited; network endpoint findings now scoped more precisely
- **High findings decreased by 5** — improvements in code patterns and refined analysis
- **Codebase grew by ~20,800 lines** — new features added while maintaining similar risk profile
- **New audit agents added:** `telemetry-tracking` and `install-script` agents are new in this audit cycle, covering previously unaudited areas

---

4. Supply Chain & Dependencies

Agent: supply-chain | **Risk:** MEDIUM | **Findings:** 8 (0C, 0H, 2M, 3L, 3I)

4.1 Dependency Overview

Metric	Value
Total lines of code	399,383
First-party code	171,633 (43%)
Vendored/third-party code	227,750 (57%)
Pre-built binaries in repo	2 (Linux ELF — bridge component)
Package managers	2 (CMake for C, npm/Bun for bridge)

4.2 Vendored Libraries

Library	Version	License	Lines	Notes
mbedTLS	3.6.3	Apache-2.0 / GPL-2.0+	207,697	Latest LTS. Includes CVE-2025-27809 fix
yyjson	0.12.0	MIT	19,556	No known CVEs
gnuwin64	N/A	GPL-3.0	476	Download scripts only

4.3 Key Findings

- **MEDIUM:** Two pre-built ELF binaries (`scorpiox-ws2tcp` , `scorpiox-ws2tcp-arm64`) committed to the repository without provenance verification
- **MEDIUM:** `MBEDTLS_SSL_VERIFY_NONE` used in 3 locations, disabling certificate chain validation in mail relay and FRP tunnel clients
- **LOW:** No checksum verification for gnuwin64 downloaded packages
- **LOW:** npm dependency uses caret version range permitting minor/patch drift; only `bun.lock` present (no `package-lock.json`)

4.4 Positive Observations

- No `FetchContent` or `ExternalProject` in CMake — all third-party code is vendored locally
- No downloads during the CMake build process
- Clear license files for all vendored libraries
- mbedTLS 3.6.3 is current LTS with latest security patches

---

5. Build System & Code Provenance

Agent: build-provenance | **Risk:** LOW | **Findings:** 12 (0C, 0H, 4M, 2L, 6I)

5.1 Build Configuration

Property	Value
Build system	CMake 3.16+
Language standard	C11
Default build type	Release
Static linking	ON by default
Docker build environment	Alpine-based (musl)

5.2 Compiler Hardening

Flag	Status
`-Wall -Wextra`	✅ Enabled
`-fstack-protector-strong`	✅ Enabled
`-D_FORTIFY_SOURCE=2`	✅ Enabled (Release)
`-fcf-protection`	✅ Enabled (GCC)
`-fPIE` / `-pie`	❌ Missing (MEDIUM)
`-Wl,-z,relro,-z,now`	❌ Missing (MEDIUM)
`-Wl,-z,noexecstack`	❌ Missing (LOW)

5.3 Key Findings

- **MEDIUM:** Position Independent Executable (PIE) not enabled — reduces ASLR effectiveness
- **MEDIUM:** Full RELRO not enabled — GOT overwrite protection absent
- **MEDIUM:** Bridge binary (`Makefile`) lacks `-fstack-protector-strong` and `-D_FORTIFY_SOURCE=2`
- **MEDIUM:** Model header generator can make network calls if `--fetch` flag is passed (currently frozen)
- **INFO:** All 493 C/H source files trace to single-organization authorship (987 commits, `SX_VERSION` macro tracks version)

---

6. Network Endpoints

Agent: network-endpoints | **Risk:** CRITICAL | **Findings:** 66 (4C, 5H, 16M, 14L, 27I)

6.1 Endpoint Summary

- **149** hardcoded URL references in first-party code

- **218** URL references in vendored code (documentation only — no runtime impact)
- **21** distinct active network domains contacted at runtime

6.2 Critical Endpoints

URL	Purpose	Risk
`http://token.scorpiox.net/claude`	Claude token fetch	**CRITICAL** — plaintext HTTP
`http://token.scorpiox.net/codex`	Codex token fetch	**CRITICAL** — plaintext HTTP
`http://token.scorpiox.net/gemini`	Gemini token fetch	**CRITICAL** — plaintext HTTP
`http://scorpiox.net:5176`	WASM router URL	**CRITICAL** — plaintext HTTP

6.3 High-Risk Endpoints

Domain	Purpose	Risk
`console.anthropic.com`	Anthropic OAuth token refresh	HIGH — hardcoded OAuth endpoint
`auth.openai.com`	OpenAI OAuth token refresh	HIGH — hardcoded OAuth endpoint
`chatgpt.com`	ChatGPT Codex backend	HIGH — hardcoded API endpoint
`20.53.240.54` (Azure IP)	TCP token server	HIGH — hardcoded IP, no DNS

6.4 DNS Configuration

Default DNS resolvers are hardcoded to Google (`8.8.8.8`) and Cloudflare (`1.1.1.1`), which leak DNS query data to third-party providers (MEDIUM).

---

7. TLS/SSL Security

Agent: tls-security | **Risk:** CRITICAL | **Findings:** 10 (3C, 2H, 3M, 1L, 1I)

7.1 Critical Findings

ID	Description	CVSS
FINDING-01	SSL certificate verification disabled in token fetch clients (cURL) — `CURLOPT_SSL_VERIFYPEER=0` , `CURLOPT_SSL_VERIFYHOST=0` in 3 token-fetching modules	9.1
FINDING-02	SSL certificate verification disabled in email client (cURL) — SMTP credentials exposed to MITM	8.1
FINDING-03	Hardcoded plaintext HTTP URLs for token retrieval — `http://token.scorpiox.net/codex` , `http://token.scorpiox.net/gemini`	9.1

7.2 High Findings

ID	Description

FINDING-04	mbedTLS `VERIFY_NONE` in FRP tunnel client and mail relay — certificate chain never validated
FINDING-05	Traffic interception module globally disables TLS verification for child processes via environment variables

7.3 Positive Observations

- Two files (`scorpiox-codex-refreshtoken.c` , `scorpiox-claudecode-refreshtoken.c`) correctly enforce full certificate verification — proves the codebase has the pattern available
- mbedTLS 3.6.3 supports TLS 1.2/1.3 with modern cipher suites
- No deprecated SSL 2.0/3.0 or TLS 1.0/1.1 protocol support

---

8. Hardcoded Credentials

Agent: credential-hardcode | **Risk:** LOW | **Findings:** 5 (0C, 0H, 1M, 2L, 2I)

8.1 Findings

Severity	Finding
MEDIUM	`OPENAI_API_KEY` set to placeholder `"xxx"` in embedded config and environment template — should be empty string
LOW	SSH credentials (host `192.168.1.6` , port `22223` , user `root`) hardcoded in configuration template (password field is empty)
LOW	Multiple private network IPs (`192.168.1.12` , `192.168.1.3` , `192.168.1.6`) in environment template expose internal network topology
INFO	Public IP `20.53.240.54` hardcoded for TCP token server
INFO	All remaining API key fields in embedded config are properly empty strings

8.2 Positive Observations

- Configuration-driven architecture with most credential fields left empty
- No actual API keys, passwords, or tokens found in source code
- The `sx_config_embedded.c` file properly uses empty strings for all credential slots except the placeholder noted above

---

9. File I/O & Data Handling

Agent: file-io | **Risk:** MEDIUM | **Findings:** 16 (0C, 2H, 4M, 4L, 6I)

9.1 High Findings

ID	Description
H-01	Mail username not validated for path traversal — `sxmail_user_path()` interpolates `user` parameter into filesystem paths without checking for `..` or `/`
H-02	Temp file leak in `emit_session_tracking` — session data (full conversation text) persists in world-readable temp files indefinitely

9.2 Medium Findings

ID	Description
M-01	MCP temp files not always cleaned up on abnormal termination
M-02	Slash command temp files never cleaned up (accumulate as 0755 executables in temp directory)
M-03	PID file race condition in session daemon
M-04	World-writable device nodes created in container runtime

9.3 Positive Observations

- Secure `mkstemp()` used consistently — zero `tmpnam()` / `tmpfile()` usage
- Sensitive directories created with 0700 permissions
- File operations use bounded `snprintf` for path construction

---

10. Buffer Safety

Agent: buffer-safety | **Risk:** LOW | **Findings:** 8 (0C, 0H, 2M, 3L, 3I)

10.1 String Operation Safety Ratio

Function	Count	Safety
`snprintf`	2,228	✅ Safe
`strncpy`	593	✅ Safe
`strncat`	20	✅ Safe
`strcat`	11	⚠ Unsafe
`sprintf`	1	⚠ Unsafe (vendor)
`strcpy`	0	✅ Not used
`gets`	0	✅ Not used
`scanf`	0	✅ Not used

Safe-to-unsafe ratio: 99.6%

10.2 Key Findings

Severity	Finding
MEDIUM	`strcat` in Windows command builders without strict length tracking — fragile pattern in 8192-byte buffers
MEDIUM	Unchecked `snprintf` return value used as network send length — truncation can cause out-of-bounds read when sending HTTP headers
LOW	Large stack buffers (up to 128 KB) in several functions — stack exhaustion risk on constrained platforms
LOW	`snprintf` truncation not checked in path construction — truncated paths could target wrong files

---

11. Memory Safety

Agent: memory-safety | **Risk:** MEDIUM | **Findings:** 46 (3C, 8H, 18M, 12L, 5I)

11.1 Allocation Statistics

- **913** dynamic allocation sites in 97 non-vendor C source files
- Mixed NULL-check discipline: critical paths generally checked, but many `strdup` / `malloc` calls omit validation

11.2 Critical Findings

ID	Description
CRIT-01	Memory leak in VT scrollback allocation failure — `vt` and `vt->cells` leaked when `scrollback` calloc fails
CRIT-02	Leak on early return in `sx_term_resize_internal` — `new_front` leaked when `new_back` is NULL due to unreachable combined check
CRIT-03	Dangerous realloc pattern in MCP module — `buf = realloc(buf, cap)` loses original pointer on failure, causing unrecoverable memory leak

11.3 High Findings

ID	Description
HIGH-01	23 unchecked `strdup` return values in `sx_api.c` — NULL dereference under memory pressure
HIGH-02	Unchecked allocations in provider modules (Claude, OpenAI, Gemini, Codex)
HIGH-03	`realloc` without temp variable in conversation history growth
HIGH-04	Missing NULL checks in config parser allocation chains
HIGH-05–08	Various unchecked allocations in server, multiplexer, and utility modules

---

12. Command Injection

Agent: command-injection | **Risk:** HIGH | **Findings:** 21 (3C, 5H, 6M, 3L, 4I)

12.1 Execution Function Usage

Function	Count
`system()`	88
`popen()`	52
`exec*()`	92

12.2 Critical Findings

ID	Description	CVSS	Network
CJ-01	Windows `cmd.exe` injection via HTTP query parameters in `scorpiox-server.c` — `SX_SANITIZE_CMD` not applied to `env_pairs[].key` / `.value`	9.8	Yes
	Windows `cmd.exe` injection via JWT		

CJ-02	`user_id` / `user_email` in `scorpiox-server.c` — unsanitized JWT claims embedded in `set` commands	8.1	Yes
CJ-03	Windows `popen()` URL injection in `scorpiox-websearch.c` — user search query passed into shell command without escaping	8.6	Indirect

12.3 High Findings

ID	Description
CJ-04	`scorpiox-otp.c` — OTP secret from user config passed to `system()` via `snprintf` without sanitization
CJ-05	`scorpiox-vault-git.c` — `run_cmd()` wraps `system()` with `vsnprintf`, accepting arbitrary format strings
CJ-06	`scorpiox-agent.c` — Multiple `system()` calls construct commands from agent task parameters
CJ-07	`scorpiox-executecurl.c` — cURL command built from user-provided parameters passed to `system()`
CJ-08	`scorpiox-kql.c` — KQL queries embedded in shell commands on Windows

12.4 Positive Observations

- Many Unix code paths intentionally use `fork()+execvp()` to avoid shell injection, with explicit comments noting this choice
- `SX_SANITIZE_CMD` macro exists and is applied in many Windows code paths — the critical findings are cases where it was missed
- Windows `CreateProcessA` used in several modules instead of `system()`

---

13. Privilege & Access Control

Agent: privilege-access | **Risk:** MEDIUM | **Findings:** 15 (1C, 2H, 4M, 3L, 5I)

13.1 Key Findings

Severity	Finding
CRITICAL	`scorpiox-thunderbolt4` requires root (`geteuid() == 0`) — Linux-only, not deployed on Windows
HIGH	`scorpiox-unshare --privileged` mode skips user namespace isolation — Linux-only container runtime
HIGH	`scorpiox-server` HTTP server binds to `0.0.0.0` by default — exposes to all network interfaces
MEDIUM	Auto-creation of `/etc/subuid` / `/etc/subgid` without user consent (Linux-only)
MEDIUM	Container device nodes created as world-writable (Linux-only)
MEDIUM	Over 50 `system()` call sites with `snprintf()`-constructed commands
MEDIUM	Environment variable injection possible via unsanitized config values

13.2 Windows-Specific Access

- No setuid/setgid equivalent requested
- No Windows service installation
- No Windows registry modifications
- No UAC elevation requests
- Process spawning primarily uses `CreateProcessA` on Windows

---

14. Windows Deployment Analysis

Agent: windows-deployment | **Risk:** LOW | **Findings:** 8 (0C, 0H, 3M, 2L, 3I)

14.1 Build Configuration

- MinGW (GCC) with static linking via CMake+Ninja
- 56 executables + 1 shared library (`libsx.dll`)
- 2 Windows-only binaries: `scorpiox-wsl.exe` , `scorpiox-busybox.exe`

14.2 Findings

Severity	Finding
MEDIUM	Plaintext HTTP token fetching defaults — token URLs default to `http://` scheme
MEDIUM	`system()` calls in `scorpiox-agent` pass through `cmd.exe` on Windows
MEDIUM	`scorpiox-server` HTTP server binds to `0.0.0.0` — Windows Firewall should mitigate but not ideal
LOW	Windows Defender may flag static-linked executables as suspicious due to uncommon PE characteristics
LOW	No Windows code signing (Authenticode) — SmartScreen warnings for end users
INFO	`scorpiox-wsl` has self-update capability with rollback mechanism
INFO	`scorpiox-busybox` performs local-only file operations, no network
INFO	Proper `#ifdef SX_WINDOWS` / `#ifdef _WIN32` platform abstraction throughout

14.3 Positive Observations

- Sound platform abstraction with proper `#ifdef` guards
- Correct Winsock initialization patterns (`WSAStartup`)
- `CreateProcessA` preferred over `system()` in most Windows paths
- No Windows registry modifications
- No UAC elevation requests
- No persistent Windows services installed

---

15. Telemetry and Tracking

Agent: telemetry-tracking | **Risk:** LOW | **Findings:** 9 (0C, 0H, 0M, 2L, 7I)

15.1 Telemetry Status: Disabled by Default

All telemetry features are disabled by default in compiled defaults, shipped config, and WASM embedded config. No network calls related to telemetry or analytics occur unless explicitly enabled by the user.

15.2 Telemetry Features

Feature	Default	Data Collected	Sends Content
`USAGE_TRACKING`	OFF (`0`)	Session ID, provider, model, hostname, username, OS, token counts	No
`EMIT_SESSION_TRACKING`	OFF (`0`)	All above + full message text (up to 1MB)	**Yes**

15.3 Key Findings

Severity	Finding
LOW	When `EMIT_SESSION_TRACKING` is enabled, full conversation content (including user prompts, assistant responses, tool calls) is transmitted to the server — users should be clearly informed
LOW	Misleading documentation in `scorpiox-usage.c` states “default: 1” but actual compiled default is `0`
INFO	No auto-update at startup; no network calls at startup
INFO	No launchd/systemd/cron auto-update services installed
INFO	`WA_BRIDGE_AUTO_UPDATE` is “on-demand” — only when WhatsApp feature is used

15.4 Positive Observations

- Explicit opt-in required for all telemetry
- No phone-home behavior at application startup
- No background update checks
- Usage tracking does NOT collect conversation content
- Clear separation between metadata telemetry and content telemetry

---

16. Install Script & Distribution Security

Agent: install-script | **Risk:** HIGH | **Findings:** 16 (3C, 4H, 5M, 2L, 2I)

16.1 Critical Findings

ID	Description
CRIT-01	Install URL in `scorpiox-sdk.c` missing `https://` scheme — `curl -fsSL "get.scorpiox.net?platform=linux" | bash` defaults to HTTP on older curl versions, enabling trivial MITM of install script
CRIT-02	Release scripts (`release*.ps1`) generate no SHA256 checksums — despite client-side verification code existing, sidecar `.sha256` files are never produced
CRIT-03	No code signing (GPG, cosign, Sigstore) on any release artifact

16.2 High Findings

ID	Description
HIGH-01	SHA256 verification in `sx_bridge.c` and `scorpiox-wsl.c` silently skips when `.sha256` file unavailable — "graceful degradation" defeats the purpose of integrity checking
HIGH-02	Container image downloads (`scorpiox-unshare`) have zero integrity verification
HIGH-03	VM image/firmware downloads (`scorpiox-vm`) have zero integrity verification
HIGH-04	Container install snippet in `scorpiox-tmux` downloads without checksum

16.3 Medium Findings

ID	Description
MED-01	`scorpiox-wsl.c` WinINet `InternetOpenUrlA` missing `INTERNET_FLAG_SECURE`
MED-02	Documentation uses `curl -k` (disables TLS verification) in `podman.txt`
MED-03	`TMUX_PODMAN_TLS_VERIFY` defaults to `false` in config template
MED-04	Self-update uses file size comparison instead of cryptographic hash
MED-05	Non-atomic install to `/usr/local/bin` — partial state on interrupt

---

17. Consolidated Risk Matrix

#	Area	Finding	Severity	Status	Recommendation
1	TLS	SSL verification disabled in token fetch clients (3 modules)	**CRITICAL**	Open	Enable `CURLOPT_SSL_VERIFYPEER=1L` , `CURLOPT_SSL_VERIFYHOST=2L`
2	TLS	SSL verification disabled in email client	**CRITICAL**	Open	Enable certificate verification; allow per-server CA config
3	TLS	Hardcoded plaintext HTTP URLs for token retrieval	**CRITICAL**	Open	Migrate to `https://` for all token endpoints
4	Network	Plaintext HTTP token endpoints (4 URLs)	**CRITICAL**	Open	Enforce HTTPS for all credential transport
5	Install	Install URL missing `https://` scheme	**CRITICAL**	Open	Add explicit `https://` prefix
6	Install	No SHA256 checksums generated during release	**CRITICAL**	Open	Add checksum generation to release scripts
7	Install	No code signing on release artifacts	**CRITICAL**	Open	Implement GPG or Sigstore signing
		Windows `cmd.exe`			Apply `SX_SANITIZE_CMD` to

8	CmdInj	injection via HTTP query params (CVSS 9.8)	**CRITICAL**	Open	`env_pairs`
9	CmdInj	Windows `cmd.exe` injection via JWT claims	**CRITICAL**	Open	Sanitize JWT fields before shell embedding
10	CmdInj	Windows `popen()` URL injection in websearch	**CRITICAL**	Open	Use `CreateProcessA` or escape shell metacharacters
11	Memory	Dangerous realloc pattern loses pointer on failure	**CRITICAL**	Open	Use temporary variable for realloc return
12	Memory	Memory leak in VT scrollback allocation failure	**CRITICAL**	Open	Free previously allocated resources on error
13	Memory	Leak on early return in term resize	**CRITICAL**	Open	Remove redundant guard; use combined check
14	Privilege	`scorpiox-thunderbolt4` requires root	**CRITICAL**	N/A (Linux)	Not deployed on Windows
15	Network	Hardcoded public IP in WASM embedded config	**HIGH**	Open	Use DNS hostname; implement certificate pinning
16	Network	Hardcoded OAuth endpoints for Anthropic/OpenAI	**HIGH**	Open	Make endpoints configurable
17	TLS	mbedTLS `VERIFY_NONE` in FRP and mail relay	**HIGH**	Open	Enable certificate verification
18	TLS	Traffic module disables TLS globally for child processes	**HIGH**	Open	Scope TLS bypass to specific connections only
19	Install	SHA256 verification silently skipped when sidecar missing	**HIGH**	Open	Fail closed when checksum unavailable
20	Install	Container/VM image downloads lack integrity verification	**HIGH**	Open	Add mandatory checksum verification
21	CmdInj	OTP secret passed to `system()` unsanitized	**HIGH**	Open	Use `execvp()` or sanitize input
22	CmdInj	`run_cmd()` in vault-git wraps `system()` with `vsnprintf`	**HIGH**	Open	Replace with `execvp()` array-based execution

23	CmdInj	Agent task parameters passed to `system()`	**HIGH**	Open	Use `CreateProcessA` on Windows
24	Memory	23+ unchecked `strdup` returns in API module	**HIGH**	Open	Add NULL checks after all `strdup` calls
25	Memory	Unchecked allocations in provider modules	**HIGH**	Open	Add NULL checks; handle allocation failure
26	FileIO	Mail username not validated for path traversal	**HIGH**	Open	Reject `..`, `/`, `\` in usernames
27	FileIO	Session temp files leak conversation content	**HIGH**	Open	Unlink temp files after consumption
28	Privilege	HTTP server binds to `0.0.0.0` by default	**HIGH**	Open	Default to `127.0.0.1`; require explicit flag for all-interface binding
29	Build	PIE not enabled — reduces ASLR effectiveness	**MEDIUM**	Open	Add `-fPIE` / `-pie` to CMake flags
30	Build	Full RELRO not enabled	**MEDIUM**	Open	Add `-Wl,-z,relro,-z,now`
31	Buffer	Unchecked `snprintf` return used as send length	**MEDIUM**	Open	Clamp return value to buffer size
32	Supply	Pre-built ELF binaries without provenance	**MEDIUM**	Open	Build from source in CI; sign artifacts

---

18. Conclusion & Recommendations

18.1 Overall Assessment

ScorpioX Code demonstrates strong foundational security practices in its C codebase — particularly in buffer safety (99.6% safe), absence of privilege escalation mechanisms, and disabled-by-default telemetry. However, significant security gaps exist in **TLS certificate verification**, **distribution integrity**, and **Windows command injection** vectors that warrant remediation before enterprise deployment.

18.2 Priority Recommendations

Immediate (Critical — Before Deployment):

1. **Enable TLS certificate verification** across all cURL and mbedTLS client connections. The codebase already demonstrates correct verification in `*-refreshtoken.c` files — replicate this pattern across all token-fetch and email modules.
2. **Migrate all token endpoints to HTTPS.** Replace `http://token.scorpiox.net/*` URLs with `https://` equivalents in `sx_config_embedded.c`, `scorpiox-codex-fetchtoken.c`, and `scorpiox-gemini-fetchtoken.c`.

3. **Fix Windows command injection vectors** in `scorpiox-server.c` by applying `SX_SANITIZE_CMD` to HTTP query parameters and JWT claims, and in `scorpiox-websearch.c` by using `CreateProcessA` instead of `popen()`.

4. **Implement distribution integrity:** Generate SHA256 checksums in release scripts, enforce verification (fail-closed), and add code signing (Authenticode for Windows, GPG for Linux).

Short-Term (High — Within 30 Days):

5. Fix dangerous `realloc` patterns (use temporary variable).
6. Add NULL checks for all `strdup` / `malloc` return values in API and provider modules.
7. Add username validation in mail subsystem to prevent path traversal.
8. Default HTTP server to `127.0.0.1` binding.
9. Enable PIE and Full RELRO compiler flags.

Medium-Term (Medium — Within 90 Days):

10. Clean up temp file management — ensure all `mkstemp` files are unlinked by the consumer.
11. Replace all `system()` calls with `CreateProcessA` (Windows) / `execvp()` (Linux) where feasible.
12. Make OAuth endpoints and DNS resolvers configurable rather than hardcoded.
13. Add `-fPIE` / `-pie` and `-Wl,-z,relro,-z,now` to the bridge Makefile.

18.3 Risk Acceptance Notes

For **Windows workstation deployment specifically**, the risk profile is somewhat improved: - Linux-only critical finding (root requirement in Thunderbolt4) does not apply - Container runtime privilege escalation findings do not apply - Windows Firewall provides default inbound protection for the `0.0.0.0` binding issue - The primary concerns remain TLS verification, command injection, and distribution integrity

---

19. Appendix: Audit Methodology

19.1 Agents Deployed

#	Agent	Scope
01	`supply-chain`	Dependencies, vendored libraries, package managers, pre-built binaries
02	`build-provenance`	Build system, compiler flags, code generation, reproducibility
03	`network-endpoints`	Hardcoded URLs, IP addresses, DNS configuration, active network domains
04	`tls-security`	TLS/SSL verification, protocol versions, cipher suites, plaintext usage
05	`credential-hardcode`	API keys, passwords, tokens, secrets in source and config files
06	`file-io`	File operations, temp files, directory permissions, data at rest
07	`buffer-safety`	Buffer overflows, string operations, stack buffers, format strings
08	`memory-safety`	malloc NULL checks, use-after-free, double-free, realloc safety, leaks
09	`command-injection`	`system()`, `popen()`, `exec*()` with unsanitized input

10	`privilege-access`	Privilege escalation, setuid/setgid, process spawning, access control
11	`windows-deployment`	Windows-specific binaries, build config, platform risks
12	`telemetry-tracking`	Data collection, phone-home behavior, tracking defaults
13	`install-script`	Installation, distribution, update mechanisms, integrity verification

19.2 Scope

- **In scope:** All 492 C/H source files in first-party code; vendored library configuration; build scripts; installation scripts; documentation
- **Out of scope:** Vendored library internals (mbedTLS, yyjson) except for configuration; runtime behavior testing; penetration testing; third-party API security

19.3 Limitations

- Static analysis only — no dynamic testing or runtime behavior verification
- Findings are based on source code review; actual exploitability may differ
- Vendor library security relies on upstream CVE tracking (mbedTLS 3.6.3 is current LTS)
- Windows-specific behavior inferred from `#ifdef` guards and platform abstraction patterns

---

Report generated by 13 automated security audit agents. All findings should be validated by human security reviewers before remediation prioritization.

End of Report