Third-Party Software Security Review — SCORPIOX CODE

Field	Value
Software	SCORPIOX CODE
Type	AI-Powered Development Tool / CLI Platform
Language	Pure C (zero external dependencies)
Audit Date	2026-04-29
Codebase Commit	`3ad7dfcfe0c414abda526ac5c747e0e88cd592f5`
Classification	CONFIDENTIAL — For Corporate Review Only

---

1. Executive Summary

This report presents a comprehensive security audit of **SCORPIOX CODE**, an AI-powered development CLI platform written in pure C. The audit was conducted by 13 specialized automated agents covering supply chain, build provenance, network endpoints, TLS security, credentials, file I/O, buffer safety, memory safety, command injection, privilege/access control, Windows deployment, telemetry, and install script security.

Key Metrics

Metric	Value
Total Lines of Code	381,273
Project Code (first-party)	152,747
Vendor Code (third-party)	228,526
Source Files Analyzed	224 (non-vendor)
Vendored Libraries	2 (mbedTLS 3.6.6, yyjson 0.12.0) — both current
Total Findings	286
Critical	2
High	18
Medium	100
Low	60
Info	87

Overall Risk Rating: MEDIUM

The software demonstrates strong engineering practices across most security domains — particularly in buffer safety (258:1 safe-to-unsafe ratio), supply chain hygiene (zero outdated dependencies, no pre-built binaries), and build hardening (stack protector, FORTIFY_SOURCE, full RELRO, NX, CFI). The primary risk areas are:

1. **Memory Safety (MEDIUM)** — 2 critical use-after-free patterns in streaming parsers, 14 high-severity unchecked allocations in network code, and ~80 missing NULL checks
2. **Install Script Integrity (MEDIUM)** — SHA256 checksums are published but not verified during installation
3. **TLS Configuration (MEDIUM)** — Global TLS verification can be disabled via config; silent downgrade on CA bundle failure

For **Windows workstation deployment** (the primary corporate use case), the risk is **LOW-MEDIUM** — most critical findings involve Linux-only code paths (container runtime, MITM proxy, VM manager) that do not compile or ship on Windows.

---

2. Deployment Scope — Windows Workstation

2.1 Windows Binaries (Ship on Windows)

The following binaries compile and deploy on Windows workstations:

Binary	Purpose
`sx.exe`	Main TUI — terminal-based AI chat interface
`scorpiox.exe`	Core CLI entry point
`scorpiox-bash.exe`	Cross-platform shell command executor
`scorpiox-busybox.exe`	Windows-only Unix tool manager (GNU coreutils)
`scorpiox-screenshot.exe`	Screen capture utility
`scorpiox-vi.exe`	Built-in text editor
`scorpiox-config.exe`	Configuration management
`scorpiox-websearch.exe`	Web search integration
`scorpiox-fetch.exe`	HTTP fetch utility
`scorpiox-renderimage.exe`	Image rendering/display
`scorpiox-pwsh.exe`	PowerShell integration
`scorpiox-wsl.exe`	WSL2 container execution (Windows-only)

Additional Windows binaries: `sx.dll`, `scorpiox-sdk.exe`, `scorpiox-gemini.exe`, `scorpiox-openai.exe`, `scorpiox-server.exe`, `scorpiox-dns.exe`, `scorpiox-frp.exe`, `scorpiox-tmux.exe`, `scorpiox-multiplexer.exe`, `scorpiox-voice.exe`, `scorpiox-agent.exe`, `scorpiox-tasks.exe`, `scorpiox-mcp.exe`, plus 20+ additional utilities (token fetchers, logging, search, email).

Total Windows binaries: 54

2.2 Linux-Only Binaries (NOT deployed on Windows)

Binary	Purpose	Why Linux-Only
`scorpiox-unshare`	Rootless container runtime	Linux namespaces (CLONE_NEWUSER)
`scorpiox-vm`	KVM virtual machine manager	/dev/kvm, Linux KVM API
`scorpiox-init`	Container initialization	Linux-specific
`scorpiox-sshpass`	SSH password utility	Unix PTY required
`scorpiox-traffic`	MITM TLS proxy	Linux-specific networking

`scorpiox-podman`	Container management	Linux containers
`scorpiox-docs`	Documentation server	Linux-specific
`scorpiox-runtest`	Test runner	Linux CI
`scorpiox-executecurl`	Curl executor	Linux sandbox
`scorpiox-cron`	Scheduled tasks	Linux cron integration
`scorpiox-whatsapp`	WhatsApp bridge	Linux service

2.3 Windows-Filtered Findings

When filtering findings to Windows-deployed binaries only:

Severity	All Platforms	Windows Only
Critical	2	1
High	18	5
Medium	100	42
Low	60	28
Total (actionable)	**180**	**76**

The reduction is significant because the highest-risk components (container runtime, VM manager, MITM proxy, sshpass) are Linux-only.

---

3. Changes Since Last Audit

Previous audit: 2026-04-28 (commit `f7f14b4`)

Metric	Previous (Apr 28)	Current (Apr 29)	Delta
Critical	5	2	**−3** ✅
High	12	18	**+6** ⚠
Medium	14	100	**+86** ⚠
Low	6	60	**+54**
Info	3	87	**+84**
Total	**40**	**286**	**+246**
Lines of Code	369,495	381,273	+11,778
Audit Agents	5	13	+8
Overall Risk	MEDIUM	MEDIUM	—

Analysis: The increase in findings is primarily due to **expanded audit scope** (13 agents vs. 5), not degraded security. The previous audit used 5 generalized agents; the current audit uses 13 specialized agents with deeper per-domain analysis. Critical findings decreased from 5 to 2, indicating real security improvements. The 80 medium-severity “missing NULL check” findings from the memory-safety agent account for the bulk of the increase — these existed in the prior codebase but were not enumerated.

Notable improvements since last audit: - Bulk migration from `system()` → `fork()+execvp()` (14 files) — major command injection reduction - Commit `089698f` specifically targeted shell injection elimination - Critical findings reduced 60% (5 → 2)

---

4. Supply Chain & Dependencies

Risk: LOW

Aspect	Status
Vendored libraries	2 (mbedTLS 3.6.6 ✅, yyjson 0.12.0 ✅) — both at latest
Pre-built binaries	None in repository ✅
Build-time fetching	None (no FetchContent/ExternalProject) ✅
Package managers	CMake (primary), npm (optional bridge only)
npm dependencies	2 direct, 103 transitive (bridge/ only — not part of Windows deployment)

Key findings: - Zero outdated dependencies - No remote code execution during build - Optional npm bridge component (`@whiskeysockets/baileys`) carries 103 transitive dependencies but is not part of the core C product - Windows GNU tools download script (`download.ps1`) does not verify checksums of downloaded packages

---

5. Build System & Code Provenance

Risk: LOW

Security Hardening Flags

Flag	Status
`-fstack-protector-strong`	✅ Enabled
`-D_FORTIFY_SOURCE=2`	✅ Enabled (Release)
`-fcf-protection`	✅ Enabled (GCC)
`-Wl,-z,noexecstack`	✅ NX stack
`-Wl,-z,relro,-z,now`	✅ Full RELRO
Static linking	✅ Default ON
Strip symbols	✅ Release builds

Build-Time Code Generation

- Model constants generator is **FROZEN by default** — requires explicit `--fetch` to contact network
- Git commit hash embedded at build time (standard practice)
- No code signing of release artifacts (see Section 16)

Missing (recommended):

- `-fstack-clash-protection` (GCC ≥8)
- `-fPIE` / `-pie` (mitigated by static linking)

---

6. Network Endpoints

Risk: LOW

The codebase contains 146 URL references in project code. All production endpoints use HTTPS.

Endpoint Categories

Category	Count	Protocol	Risk
First-party (`*.scorpiox.net`)	20+	HTTPS	Expected
Third-party AI APIs	10+	HTTPS	Expected
Hardcoded DNS resolvers (8.8.8.8, 1.1.1.1)	2	UDP/53	Medium — bypasses org DNS
Hardcoded public IP (token service)	1	TCP/9800	Medium
HTTP localhost endpoints	3	HTTP	Low (local only)
LAN IPs (192.168.x.x)	5	Various	Low — info disclosure

No unauthorized exfiltration endpoints detected. No suspicious domains.

---

7. TLS/SSL Security

Risk: MEDIUM

Finding	Severity	Description
Global TLS disable via config	Medium	`SX_TLS_VERIFY=0` disables all cert verification across 10 callsites
Mail relay TLS downgrade	Medium	Silent downgrade to VERIFY_OPTIONAL on CA bundle failure
STARTTLS vulnerable to downgrade	Medium	Falls back to plaintext if STARTTLS rejected
MITM proxy disables TLS	Low	Intentional — proxy tool by design
mbedTLS VERIFY_NONE (server)	Info	Standard SMTP server behavior

Positive findings: - Centralized TLS helper (`sx_tls.h`) enforces verification by default - TLS 1.2 minimum enforced for mbedTLS paths - Strong cipher suites configured (ECDHE, AES-GCM, ChaCha20) - libcurl defaults to system CA bundle

---

8. Hardcoded Credentials

Risk: LOW

No actual secret values (API keys, passwords, tokens) are hardcoded. All credential fields use empty placeholder strings populated at runtime.

Finding	Severity	Description
SSH user+host+port in embedded config	Medium	`root@192.168.1.6:22223` — infrastructure disclosure
Public IP for token service	Medium	`20.53.240.54:9800` in compiled binary

Internal LAN host in embedded config	Medium	`root@192.168.1.3` + SMB share path
Empty credential placeholders compiled	Low	Field names visible in binary strings
Example secret in code comment	Low	`"mysecret"` string literal in frp.h
Podman registry with TLS verify=false	Low	`registry.scorpiox.net` without verification

---

9. File I/O & Data Handling

Risk: LOW-MEDIUM

Finding	Severity	Description
Temp file leak in emit-session fork	High	`/tmp/sx_emit_XXXXXX` never unlinked — leaks conversation data
Inconsistent mkdir permissions	Medium	Some dirs created 0755 instead of 0700
sshpass password in CLI argv	Medium	Visible in `/proc/*/cmdline` (Linux-only)
Traffic logs contain full request bodies	Medium	API responses logged to disk (Linux-only)
Config summary in session files	Medium	Sensitive config keys written to session state
Hardcoded `/tmp` paths	Low	No TMPDIR respect in some paths

Positive findings: - Consistent `mkstemp()` usage for temp files - Atomic write pattern (write-to-tmp + `rename()`) - API key redaction in traffic logs - Path-traversal validation in security-sensitive paths

---

10. Buffer Safety

Risk: LOW

The project demonstrates **exemplary buffer safety discipline**:

Metric	Value
`snprintf` calls	2,213
`strncpy` calls	601
`sprintf` (non-vendor)	1 (proven safe)
`strcpy`	0
`strcat`	11 (all Windows-only, partially bounded)
Safe-to-unsafe ratio	**258:1**
`gets` / uncontrolled `scanf`	0

Findings: - 1 `sprintf` usage proven safe (fixed-length hex encoding) - 11 `strcat` calls in Windows-only code paths — bounded by companion `strncat` but theoretically overflowable with extreme argument counts - All large stack buffers (up to 64KB) have proper bounds checking - No format string vulnerabilities in project code

---

11. Memory Safety

Risk: MEDIUM

This is the highest-risk domain identified.

Severity	Count	Category
Critical	2	Use-after-free patterns
High	14	Missing NULL checks in network/security code
Medium	80	Missing malloc/calloc NULL checks (total)
Low	30	Error-path memory leaks
Info	5	Safe patterns noted

Critical Findings

CRIT-01: Use-After-Free in OpenAI WASM SSE Parser - File: `sx_provider_openai_wasm.c:412-415` - Latent UAF — dead code path that dereferences freed buffers. Currently unreachable but becomes exploitable if allocation logic is refactored.

CRIT-02: vtable→free followed by data access in Provider Cleanup - File: `sx_provider.c:199-207` - Polymorphic `vtable->free(p)` may free `p->data`, which is then accessed. Relies on implicit undocumented convention.

High Findings (Network Code)

- 14 missing NULL checks on `malloc` in `libsxnet/` (HTTP handlers, streaming parsers)
- Could cause crashes under memory pressure during active network connections

Positive findings: - Realloc patterns consistently use temp variables (no realloc-in-place) - 91 `calloc` usages (zero-initialized memory) - Large buffers use `calloc` for clarity

---

12. Command Injection

Risk: LOW

The codebase underwent a significant bulk migration (commit `089698f`) converting most `system()` calls to `fork()+execvp()`, eliminating shell interpretation. Remaining findings:

Finding	Severity	Description	Platform
`IMAGE_BASE_URL` in `popen()`	Medium	Environment variable reaches shell via single-quoted interpolation	Linux
`SCORPIOX_INDEX_URL` in `popen()`	Medium	Config value reaches shell without quoting	Linux
`INIT_TOOLS` env in `system()`	Low	Tool names from env reach `which` command	Linux
Windows `system()` fallbacks	Low	6 remaining `system()` calls behind `#ifdef _WIN32`	Windows
`scorpiox-server` OTP	Low	Network-reachable but has `is_safe_shell_arg()` sanitization	Both

Post-migration metrics: - `system()` calls remaining: 8 (6 Windows-only, 2 Linux) - `popen()` calls remaining: 14 (most with controlled inputs) - `fork()+execvp()` calls: 45+ (safe argument vector pattern)

---

13. Privilege & Access Control

Risk: MEDIUM

Finding	Severity	Description	Platform
Container runtime no seccomp	High	`scorpiox-unshare` containers run with full syscall surface	Linux-only
Hard root requirement (thunderbolt4)	Medium	Requires `geteuid() == 0`	Linux-only
Root-adaptive container runtime	Medium	`--privileged` bypasses user namespace isolation	Linux-only
No SUID/SGID binaries	Info	Positive — no setuid bits set	All
No `prctl` / `capset` / `seccomp`	Info	No capability dropping in containers	Linux-only

Windows impact: Minimal. The high-severity container runtime finding is Linux-only. Windows binaries do not require elevated privileges for normal operation. `scorpiox-wsl.exe` uses standard Windows APIs (WinInet, CreateProcess).

---

14. Windows Deployment Analysis

Risk: LOW

Windows-Specific Security Findings

Finding	Severity	Description
`system()` fallback on Windows	Medium	`sx_system_safe()` maps to `system()` on `_WIN32` — no shell escape
Windows Defender SmartScreen	Low	Unsigned binaries trigger SmartScreen warnings
`CreateProcess` without full path	Low	Some process spawning without absolute paths
ConPTY usage	Info	Modern Windows pseudoconsole API used correctly
Static MinGW linking	Info	No DLL dependencies — self-contained deployment
SHA256 self-update verification	Info	`scorpiox-wsl.exe` verifies update integrity ✓

Windows Security Architecture

- **No Windows services installed** — all binaries are user-space CLI tools
- **No registry modifications** — configuration stored in flat files
- **No kernel drivers** — pure user-mode operation
- **Static linking** — no DLL side-loading risk

- **No admin rights required** — installs to user-writable paths

---

15. Telemetry and Tracking

Risk: LOW

Verdict: No tracking by default.

Feature	Default	Opt-In Required	Data Collected
Usage Tracking	**OFF**	Yes (`USAGE_TRACKING=1`)	Token counts, model name, hostname, username, project
Session Tracking	**OFF**	Yes (`EMIT_SESSION_TRACKING=1`)	Full conversation content (messages, tool calls)
Auto-Update	None	User-initiated only	—

Key findings: - Both telemetry features disabled by default (`0`) in all config sources - No phone-home, no heartbeat, no background network activity when disabled - When enabled, usage tracking collects machine hostname + OS username (PII) - Session tracking (when enabled) transmits full conversation content — significant data sensitivity - No unsubscribe/delete mechanism documented

For corporate deployment: Ensure `USAGE_TRACKING=0` and `EMIT_SESSION_TRACKING=0` remain in deployed configuration.

---

16. Install Script & Distribution Security

Risk: MEDIUM

Finding	Severity	Description
No checksum verification at install	High	SHA256 sidecar files published but install scripts don't verify them
No code signing	High	No GPG/cosign signatures on release artifacts
`curl | bash` pattern	Medium	Remote code execution without preview
Non-atomic install	Medium	Partial state possible on interrupted download
No HSTS headers	Medium	First HTTP request vulnerable to MITM before redirect
`-ExecutionPolicy Bypass` in updater	Medium	Windows PowerShell policy circumvented
System-wide install on Linux	Low	`/usr/local/bin` with sudo
No uninstall mechanism	Low	No documented cleanup procedure
RC file modification without backup	Low	`.zshrc` / `.bash_profile` modified in-place
Unvalidated branch names in updater	Low	Arbitrary branch names accepted

Positive findings: - HTTPS enforced on both distribution domains (Caddy 308 redirect) - `scorpiox-wsl.exe` self-update **does** verify SHA256 + fails closed ✅ - No auto-update without user consent - No cron/launchd/systemd persistence

---

17. Consolidated Risk Matrix

#	Area	Finding	Severity	Status	Platform	Recommendation
1	Memory	Use-after-free in WASM SSE parser	Critical	Open	Both	Remove dead code path or fix dereference ordering
2	Memory	vtable→free contract ambiguity	Critical	Open	Both	Document contract; set `p->data = NULL` after vtable→free
3	Install	No checksum verification at install	High	Open	Both	Add `sha256sum` verification to install scripts
4	Install	No code signing	High	Open	Both	Implement cosign or GPG signing
5	Memory	14 unchecked allocs in network code	High	Open	Both	Add NULL checks in `libsxnet/`
6	Privilege	Container runtime no seccomp	High	Open	Linux	Add default seccomp profile
7	File I/O	Temp file leak (emit-session)	High	Open	Both	Unlink temp file in child process
8	TLS	Global TLS disable via config	Medium	Open	Both	Remove global disable or restrict to debug builds
9	TLS	Mail relay silent TLS downgrade	Medium	Open	Linux	Fail closed on CA bundle failure
10	TLS	STARTTLS downgrade to plaintext	Medium	Open	Linux	Enforce STARTTLS when configured
11	Credential	Infrastructure IPs in compiled binary	Medium	Open	Both	Move to runtime-only config
12	Network	Hardcoded DNS resolvers	Medium	Open	Both	Use system resolver or make configurable
13	Network	Public IP for token service	Medium	Open	Both	Use DNS hostname
14	Command Inj	`IMAGE_BASE_URL` in `popen()`	Medium	Open	Linux	Use `fork()+execvp()`
15	Command Inj	`SCORPIOX_INDEX_URL` in `popen()`	Medium	Open	Linux	Use `fork()+execvp()`
16	File I/O	Inconsistent mkdir 0755	Medium	Open	Linux	Use 0700 consistently
17	File I/O	sshpass password in argv	Medium	Open	Linux	Remove `-p` option; use env var only
18	Install	`curl | bash` pattern	Medium	Open	Both	Provide two-step install option
19	Install	No HSTS headers	Medium	Open	Both	Add Strict-Transport-Security
20	Install	`-ExecutionPolicy Bypass`	Medium	Open	Windows	Use signed PowerShell scripts
21	Windows	`system()` fallback	Medium	Open	Windows	Migrate to `CreateProcess` with

		on Windows				arg vector
22	Memory	80 missing NULL checks	Medium	Open	Both	Add systematic NULL check audit
23	Privilege	Root-adaptive container	Medium	Open	Linux	Document security implications
24	Buffer	`strcat` in Windows code	Low	Open	Windows	Replace with bounded alternatives
25	Network	HTTP localhost endpoints	Low	Open	Both	Document as local-only
26	Telemetry	Hostname + username collected	Low	Open	Both	Document in privacy policy
27	File I/O	Hardcoded /tmp paths	Low	Open	Linux	Respect TMPDIR
28	Memory	30 error-path memory leaks	Low	Open	Both	Fix error-path cleanup

---

18. Conclusion & Recommendations

Overall Assessment

SCORPIOX CODE demonstrates a **mature security posture** for a C-based project of this scale. The codebase shows clear evidence of ongoing security investment:

- **Excellent** buffer safety (258:1 safe-to-unsafe ratio, zero `strcpy` / `gets`)
- **Strong** build hardening (stack protector, FORTIFY_SOURCE, full RELRO, CFI)
- **Clean** supply chain (2 current vendored libraries, no pre-built binaries)
- **No tracking by default** — opt-in telemetry only
- **Recent security migration** — bulk `system()` → `fork()+execvp()` conversion

Priority Recommendations

Immediate (Critical/High): 1. Fix use-after-free patterns in `sx_provider_openai_wasm.c` and `sx_provider.c` 2. Add SHA256 verification to install scripts (infrastructure exists, just not wired up) 3. Implement code signing for release artifacts 4. Add NULL checks to 14 network-code allocations in `libsxnet/` 5. Fix temp file leak in `emit_session_external()`

Short-term (Medium): 6. Remove or restrict global TLS verification disable 7. Externalize infrastructure IPs from compiled embedded config 8. Migrate remaining 2 Linux `popen()` calls to `fork()+execvp()` 9. Add HSTS headers to distribution domains 10. Migrate Windows `sx_system_safe()` to `CreateProcess`

Long-term (Low): 11. Systematic NULL-check audit across all 80 sites 12. Error-path memory leak cleanup (30 sites) 13. Document privacy policy for opt-in telemetry data 14. Add seccomp profile to container runtime

Windows Deployment Clearance

For **Windows workstation deployment**, the software presents **LOW-MEDIUM risk**: - No admin rights required - No Windows services or registry modifications - No DLL dependencies (static linking) - No kernel drivers - No tracking by default - Most high-severity findings are Linux-only code paths - Primary concern: unsigned binaries (SmartScreen warnings) and missing install-time checksum verification

Recommendation: Approve for Windows workstation deployment with the following conditions: 1. Ensure telemetry remains disabled in corporate config 2. Monitor for code signing implementation in future releases 3. Deploy via managed software distribution (bypasses install script concerns)

---

19. Appendix: Audit Methodology

Agents Deployed

#	Agent	Scope	Technique
1	supply-chain	Package managers, vendored libs, lock files	Static analysis of manifests + version comparison
2	build-provenance	CMake config, compiler flags, build scripts	Build system configuration review
3	network-endpoints	URLs, IPs, ports, domains in source	Regex + semantic analysis of all `.c` / `.h` files
4	tls-security	Certificate verification, protocols, ciphers	Code path analysis of TLS callsites
5	credential-hardcode	API keys, passwords, tokens, secrets	Pattern matching + entropy analysis
6	file-io	Temp files, permissions, data at rest	File operation audit across all sources
7	buffer-safety	sprintf/strcpy/strcat, stack buffers, bounds	Function call census + bounds verification
8	memory-safety	malloc/free patterns, UAF, double-free, leaks	Allocation tracking + lifetime analysis
9	command-injection	system/popen/exec, shell interpretation	Taint analysis from input to shell sinks
10	privilege-access	setuid, root checks, namespaces, IPC	Privilege escalation path analysis
11	windows-deployment	Win32 API usage, Windows-specific risks	Platform-conditional code review
12	telemetry-tracking	Data collection, phone-home, tracking	Network callsite + payload analysis
13	install-script	Install/update scripts, distribution integrity	Script review + transport security verification

Scope & Limitations

- **In scope:** All non-vendor C source code (152,747 LOC), build configuration, install scripts, embedded configuration
- **Out of scope:** Vendor library internals (mbedTLS, yyjson) except for configuration; runtime behavior testing; penetration testing
- **Methodology:** Automated static analysis with manual finding verification
- **False positive rate:** Findings verified against code context; theoretical/latent issues marked as such
- **Commit analyzed:** `3ad7dfcfe0c414abda526ac5c747e0e88cd592f5` (main branch, 2026-04-29)

End of Report