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	`ca3fe9b31756f623e212b011debc7c6d119388f3`
Classification	CONFIDENTIAL — For Corporate Review Only

---

1. Executive Summary

This report presents a comprehensive security audit of **SCORPIOX CODE**, an AI-powered development tool and CLI platform written in pure C with zero external runtime dependencies. The codebase spans **403,876 lines of code** (173,858 first-party, 230,018 vendored) across approximately 149 non-vendor source files. Thirteen specialized audit agents analyzed the software across all major security domains.

Key Findings

Severity	Count
Critical	0
High	7
Medium	28
Low	37
Info	135
Total	**207**

Overall Risk Rating: LOW-MEDIUM

The software demonstrates mature security engineering practices across the majority of its codebase:

- **Zero critical vulnerabilities** were identified across all 13 audit domains.
- **Buffer safety discipline is exceptional** — 99.95% of 2,900+ string operations use bounds-checked functions (`snprintf`, `strncpy`). No uses of `strcpy`, `strcat`, or `gets` were found.
- **Memory safety is strong** — 92%+ of 454 allocation sites have proper NULL checks. All `realloc` patterns use safe temporary variables. The project's `sx_strdup()` wrapper aborts on OOM.
- **Supply chain is clean** — only 2 vendored C libraries (Mbed TLS 3.6.6, yyjson 0.12.0), both well-maintained and properly licensed. No pre-built binaries ship in the repository.
- **All telemetry is disabled by default** — no network calls at startup, no mandatory data

collection. Both tracking subsystems (`USAGE_TRACKING`, `EMIT_SESSION_TRACKING`) default to off.
- **TLS defaults are strong** — certificate verification is enabled by default with centralized configuration across all 9 curl callsites.

The 7 high-severity findings are concentrated in two areas: 1. **Install script integrity** (3 findings): Install scripts download archives without checksum verification, `get.scorpiox.net` accepts HTTP without redirect, and no code signing exists. 2. **Command injection on Windows** (2 findings): The HTTP server's Windows CGI path and git deploy handler use shell interpolation with incomplete sanitization. 3. **Memory safety** (1 finding): A fragile use-after-free contract in the provider vtable cleanup pattern. 4. **Privilege isolation** (1 finding): The `--privileged` container flag bypasses user namespace isolation.

For **Windows workstation deployment** (the primary corporate use case), the risk profile is **LOW** — the most concerning findings (install script integrity, Windows CGI injection) are either mitigable through network controls or limited to server-mode operation not typical of developer workstation use.

---

2. Deployment Scope

Windows Workstation (Primary Deployment Target)

The following binaries compile and deploy on Windows:

Binary	Purpose
`sx.exe`	Main AI-powered TUI — primary developer interface
`scorpiox.exe`	Alias/launcher for sx
`scorpiox-bash.exe`	Shell command execution tool for AI agent
`scorpiox-busybox.exe`	Unix tool manager (MSYS2/BusyBox coreutils)
`scorpiox-screenshot.exe`	Screen capture via Win32 GDI
`scorpiox-vi.exe`	Built-in text editor
`scorpiox-config.exe`	Configuration TUI editor
`scorpiox-websearch.exe`	Web search tool
`scorpiox-fetch.exe`	URL content fetcher (headless browser)
`scorpiox-renderimage.exe`	Image rendering/display
`scorpiox-pwsh.exe`	PowerShell execution wrapper
`scorpiox-wsl.exe`	WSL2 distribution manager (Windows-only)

Additional Windows-compiled binaries (43 total): `scorpiox-agent`, `scorpiox-askuserquestion`, `scorpiox-beam`, `scorpiox-claudecode-fetchtoken`, `scorpiox-claudecode-models`, `scorpiox-claudecode-refreshtoken`, `scorpiox-codex-fetchtoken`, `scorpiox-codex-refreshtoken`, `scorpiox-conv`, `scorpiox-debug`, `scorpiox-dns`, `scorpiox-email`, `scorpiox-emit-session`, `scorpiox-frp`, `scorpiox-gemini`, `scorpiox-gemini-fetchtoken`, `scorpiox-hook`, `scorpiox-host`, `scorpiox-kql`, `scorpiox-logger`, `scorpiox-mcp`, `scorpiox-mirror-git`, `scorpiox-multiplexer`, `scorpiox-openai`, `scorpiox-otp`, `scorpiox-planmode`, `scorpiox-printlogs`, `scorpiox-rewind`, `scorpiox-sdk`, `scorpiox-search`, `scorpiox-server`, `scorpiox-server-email`, `scorpiox-server-fetchtoken`, `scorpiox-server-http2tcp`, `scorpiox-skills`, `scorpiox-systemprompt`, `scorpiox-tasks`, `scorpiox-tmux`, `scorpiox-transcript`, `scorpiox-usage`, `scorpiox-vault-git`, `scorpiox-voice`.

Linux-Only Binaries (Not Deployed on Windows)

Binary	Reason

`scorpiox-unshare`	Linux namespaces / container runtime (`CLONE_NEWNS` , `pivot_root`)
`scorpiox-vm`	KVM-based virtual machine (Linux ioctls)
`scorpiox-init`	Container init process (PID 1 in namespaces)
`scorpiox-sshpass`	SSH password automation (Unix PTY)
`scorpiox-traffic`	Network traffic capture proxy (Unix-only)
`scorpiox-podman`	Podman container integration
`scorpiox-cron`	Cron-style scheduler
`scorpiox-docs`	Documentation generator
`scorpiox-runtest`	Test runner
`scorpiox-executecurl`	Curl execution wrapper
`scorpiox-whatsapp`	WhatsApp bridge (`fork` / `pipe` / `select`)
`scorpiox-thunderbolt4`	Thunderbolt 4 file transfer (raw BPF sockets)
`scorpiox-imessage`	iMessage integration (macOS/Linux)

Windows-Filtered Findings

When excluding findings that apply only to Linux-only components:

Severity	All Platforms	Windows-Relevant
Critical	0	0
High	7	6
Medium	28	23
Low	37	30
Info	135	133
Total	**207**	**192**

The reduction is primarily from Linux-specific findings in privilege management (container `--privileged` flag, thunderbolt4 root requirement, KVM access), file I/O (`O_CLOEXEC` , PID file TOCTOU), and command injection (Linux-only tool paths).

---

3. Changes Since Last Audit

Field	Previous Audit	Current Audit	Delta
Audit Folder	`output/2026-04-29_3ad7dfc`	`output/2026-04-29_ca3fe9b`	—
Commit	`3ad7dfc`	`ca3fe9b`	+21,111 LOC project code
Lines of Code (Total)	381,273	403,876	+22,603
Lines of Code (Project)	152,747	173,858	+21,111
Lines of Code (Vendor)	228,526	230,018	+1,492

Findings Delta by Severity

Severity	Previous	Current	Change
Critical	2	0	**−2** ✅
High	18	7	**−11** ✅
Medium	100	28	**−72** ✅
Low	60	37	**−23** ✅
Info	87	135	+48
Total	**286**	**207**	**−79** ✅

Windows Findings Delta

Severity	Previous	Current	Change
Critical	1	0	**−1** ✅
High	5	6	+1
Medium	42	23	**−19** ✅
Low	28	30	+2
Total (excl info)	**76**	**59**	**−17** ✅

Summary: Significant improvement across all actionable severity levels. Critical findings dropped from 2 to 0, high findings reduced by 61%, and medium findings reduced by 72%. The increase in informational findings reflects more thorough enumeration of network endpoints and positive security controls (not security defects). The codebase grew by 21,111 lines of project code with a net *decrease* in vulnerability count — indicating improved code quality practices.

---

4. Supply Chain & Dependencies

Agent Risk Rating: LOW

The project has a minimal and well-controlled supply chain:

Component	Type	Version	License
Mbed TLS	Vendored C library	3.6.6 (LTS)	Apache-2.0 / GPL-2.0+
yyjson	Vendored C library (single-file)	0.12.0	MIT
MSYS2 GNU tools	Download script (Windows dev)	Various	GPL
Bun/npm	Bridge component only	lockfile-pinned	Various

Key Positives: - No `FetchContent`, `ExternalProject`, or runtime dependency downloads in the CMake build - No pre-built binaries in the repository — everything builds from source - Alpine Docker base image is SHA256 digest-pinned - Curl source download in ARM64 Dockerfile is integrity-verified - npm/Bun dependencies are lockfile-pinned with integrity hashes

Findings:

ID	Severity	Finding
M1	Medium	`gnuwin64/download.ps1` downloads MSYS2 packages without SHA256 checksum verification

M2	Medium	`Dockerfile.linux-arm64` uses `ubuntu:22.04` without digest pin
L1	Low	`release_whatsapp.ps1` installs Bun via `curl|bash` without verification
L2	Low	`pip3 install requests` without version pinning in ARM64 Dockerfile

---

5. Build System & Code Provenance

Agent Risk Rating: LOW

The project uses a two-tier CMake build system (C11) with comprehensive security hardening:

Security Flag	Status
`-Wall -Wextra`	✅ Full warnings
`-fstack-protector-strong`	✅ Stack canaries
`-D_FORTIFY_SOURCE=2`	✅ Release mode
`-fcf-protection`	✅ GCC control-flow enforcement
`-Wl,-z,noexecstack`	✅ Non-executable stack
`-Wl,-z,relro,-z,now`	✅ Full RELRO
`-fPIE` / `-pie`	❌ Incompatible with static linking (expected)

Build provenance is maintained via `build-info.json` files with commit hash, branch, date, and platform. Model header code generation (`generate_models`) is frozen by default (`MODELS_FROZEN = True`) and disabled in release builds.

Findings:

ID	Severity	Finding
M1	Medium	Bun runtime downloaded via `curl|bash` at build-time in WhatsApp release script
L1	Low	Bridge `Makefile` missing linker hardening flags (`RELRO` , `noexecstack`)
L2	Low	No reproducible build infrastructure (`SOURCE_DATE_EPOCH` not set)

---

6. Network Endpoints

Agent Risk Rating: LOW

A total of **364 URL references** were found (218 in vendored code, 146+ in project code). All external endpoints serve legitimate purposes:

Category	Count	Examples
First-party infrastructure	19	`dist.scorpiox.net` , `token.scorpiox.net` , `code.scorpiox.net`
AI provider APIs	15+	Anthropic, OpenAI, Google Gemini, Codex endpoints

Search engines	5+	Google, Bing, DuckDuckGo, Brave, Kagi
Public DNS	4	`1.1.1.1`, `8.8.8.8`, `9.9.9.9`, `208.67.222.222`
Localhost bindings	5+	`127.0.0.1` (properly scoped)

Findings:

ID	Severity	Finding
M1	Medium	DNS server defaults to `0.0.0.0` binding (standard but broad)
L1	Low	Some endpoints configurable via env vars without validation
L2	Low	WhatsApp bridge connects to Meta infrastructure (expected)

---

7. TLS/SSL Security

Agent Risk Rating: MEDIUM

The codebase has a **centralized TLS configuration** (`sx_tls.h`) that ensures consistent certificate verification across all 9 curl callsites. Defaults are strong with verification enabled.

Findings:

ID	Severity	Finding
FINDING-01	Medium	TLS verification globally disableable via single `SX_TLS_VERIFY` config flag — affects all 9 curl callsites
FINDING-04	Medium	Missing `mbedtls_ssl_conf_min_tls_version` in 2 of 3 mbedTLS contexts (`sxmail_queue.c`, `sx_frp.c`)
FINDING-03	Low	Opportunistic TLS with `VERIFY_OPTIONAL` for MX mail delivery (industry standard per RFC 3207)
FINDING-06	Low	Non-PFS cipher suite (RSA key exchange) enabled alongside PFS suites
FINDING-08	Low	HTTP credential injection over plaintext localhost IPC

Positive Controls: - Defense-in-depth defaults: `SX_TLS_VERIFY` defaults to `true` with warning when disabled - Fail-closed relay TLS: aborts when CA bundles fail to load - STARTTLS enforcement on port 587 submission - SNI hostname properly set in all mbedTLS contexts - All external API endpoints default to HTTPS - IMAPS (port 993) for IMAP connections

---

8. Hardcoded Credentials

Agent Risk Rating: LOW

No actual secrets, API keys, or passwords are hardcoded in the codebase. All credential configuration fields use empty-string placeholders, deferring injection to runtime.

Findings:

ID	Severity	Finding
MEDIUM-01	Medium	Default SSH username `"root"` hardcoded in embedded config (`CLAUDE_CODE_SSH_USER`)
MEDIUM-02	Medium	Hardcoded private IP addresses (`192.168.1.3` , `192.168.1.6`) in environment template
MEDIUM-03	Medium	Hardcoded internal IP `root@192.168.1.3` in release orchestration scripts
LOW-01	Low	15+ hardcoded internal service URLs compiled into binaries
LOW-02	Low	Non-standard SSH port `22223` reveals infrastructure detail

---

9. File I/O & Data Handling

Agent Risk Rating: LOW-MEDIUM

File I/O practices are **generally strong**: - **Exclusive use of `mkstemp()` / `mkdtemp()`** — zero uses of insecure `tmpnam()` / `tempnam()` - Sensitive config files are `chmod 0600`; directories are `0700` - API keys are redacted from traffic logs - Atomic file writes via rename pattern in critical paths

Findings:

ID	Severity	Finding
FIO-01	Medium	Missing `O_CLOEXEC` on ~48 `open()` calls — FDs may leak to child processes
FIO-02	Medium	PID file TOCTOU race condition (mitigated by socket bind)
FIO-03	Medium	Gemini API key passed in URL query parameters — risk of exposure via intermediaries
FIO-04	Low	Container device mount targets created `0666`
FIO-05	Low	SMTP relay password not zeroed from memory after use
FIO-06	Low	Release scripts pass passwords via command-line arguments
FIO-07	Low	DNS audit log and hook log files world-readable (`0644`)

---

10. Buffer Safety

Agent Risk Rating: LOW

Buffer safety discipline is **exceptional** across the codebase:

Metric	Value
`snprintf` usage	2,252 calls
`strncpy` usage	601 calls

Unsafe `sprintf` (real)	1 (provably safe — bounded hex conversion)
Unsafe `strcpy` / `strcat` / `gets`	**0**
Format string vulnerabilities	**0**
Unbounded `scanf`	**0**
Safety ratio	**>99.95%**

Findings:

ID	Severity	Finding
MEDIUM-01	Medium	`snprintf` return value accumulation without clamping in IMAP FETCH response builder — potential OOB write on truncation
MEDIUM-02	Medium	Same pattern in RFC822 email message builder (`scorpiox-email.c`) — heap buffer overflow writing CRLF terminator
MEDIUM-03	Medium	Same pattern in IMAP SEARCH response builder
LOW-01	Low	234 large stack buffers (≥4096 bytes) — stack exhaustion risk on deep recursion
LOW-02	Low	Integer multiplication in `malloc` without overflow check
LOW-03	Low	`strncpy` null-termination reliance patterns
LOW-04	Low	Static 64KB buffers in WASM HTTP module

---

11. Memory Safety

Agent Risk Rating: LOW

Memory management is **well-disciplined**:

Metric	Value
Total allocation sites	~454
Sites with NULL check	~419 (92%+)
Unsafe realloc patterns	**0**
Confirmed double-free	**0**
`strdup` calls (via `sx_strdup`)	402 (all abort-on-OOM)

Findings:

ID	Severity	Finding
FINDING-01	**High**	Use-after-free risk in `sx_provider_free()` — vtable `free()` call followed by `p->data = NULL` write. Currently safe (all existing providers only free sub-resources), but fragile undocumented contract that breaks if any future provider frees `p` itself

FINDING-02	Medium	Missing NULL checks in `scorpiox-thunderbolt4.c` buffer allocations — memory leak on OOM in error path
FINDING-04	Low	Missing NULL check on `sx_term.c` capture buffer
FINDING-05–07	Low	Redundant dead-code NULL check patterns (cosmetic)

---

12. Command Injection

Agent Risk Rating: MEDIUM

Most code paths use `fork()+execvp()` (argv-based, no shell) — the safe approach. Where `system()` / `popen()` is used, input validation is generally present (`is_safe_shell_arg()` , `SX_SANITIZE_CMD` , character whitelisting, single-quote escaping).

Findings:

ID	Severity	CVSS	Finding
CJ-01	**High**	8.1	Windows HTTP server CGI path builds `cmd /C` with HTTP request fields via `SX_SANITIZE_CMD` — strips rather than rejects dangerous chars; does not filter `\t` , `\0` , `@`
CJ-09	**High**	7.5	Git deploy handler in HTTP server single-quote-wraps paths but does not escape embedded single quotes
CJ-02	Medium	6.3	Image file paths interpolated into `ffmpeg` / `convert` / `ffprobe` shell commands — filenames with metacharacters could escape double-quoting
CJ-03	Medium	5.3	`sx_tool_exists()` passes unsanitized tool name to `which` via `popen()`
CJ-05	Medium	5.9	Windows agent tool-wait path uses `system()` with incomplete `is_safe_shell_arg()` validation
CJ-06	Medium	5.6	Windows SSH command in `scorpiox-tmux.c` uses `system()` with unsanitized hostname
CJ-08	Medium	5.4	Windows `sx_rmrf()` uses `system("rd /s /q ...")` with path interpolation

---

13. Privilege & Access Control

Agent Risk Rating: MEDIUM

No setuid/setgid bits are set on any binary. The container runtime (`scorpiox-unshare`) implements comprehensive Linux namespace isolation with `pivot_root`.

Findings:

ID	Severity	Finding
H-1	**High**	`--privileged` flag bypasses user namespace isolation in `scorpiox-unshare` (Linux-only, requires root)
M-1	Medium	`scorpiox-thunderbolt4` requires unconditional root — could use `CAP_NET_RAW` instead
M-2	Medium	DNS server does not drop privileges after binding to port 53
M-3	Medium	`sx_system_safe()` passes commands through `/bin/sh -c` despite its name
L-1	Low	Chroot paths in unshare container built via string formatting
L-2	Low	Agent metacharacter filtering for `is_safe_shell_arg()` is incomplete
L-3	Low	KVM VM requires `/dev/kvm` group membership

Positive Controls: - Container runtime implements `CLONE_NEWNS | CLONE_NEWPID | CLONE_NEWUTS | CLONE_NEWUSER | CLONE_NEWNET` - Filesystem isolation via `pivot_root()` with old root unmounted - Mount hardening: `MS_NOSUID | MS_NODEV | MS_NOEXEC` on proc, sysfs, devpts - IPC exclusively via pipes (minimal attack surface) - No shared memory, Unix domain sockets, or message queues for IPC

---

14. Windows Deployment Analysis

Agent Risk Rating: LOW

The Windows deployment surface is **clean and well-structured**:

- **53 binaries** compile on Windows; **13 are correctly excluded** as Linux-only
- Platform-specific code is properly guarded with `#ifdef SX_WINDOWS` / `if(WIN32)` CMake guards
- No registry access, service installation, UAC elevation, or DLL injection patterns found
- Standard Win32 API usage: WinINet, `CreateProcessA`, GDI (screenshot), Winsock2, `bcrypt`
- Self-update mechanism (`scorpiox-wsl`) verifies SHA256 checksums of downloaded binaries
- Uses `_spawnvp` and `CreateProcessA` (safe) rather than `system()` for WSL invocation

Observation	Detail
Self-update integrity	SHA256 verified via `.sha256` sidecar file (falls through if unavailable)
Network surface	Server binaries bind to configurable addresses; none require elevation
Optional DLL	`libsx.dll` (C# P/Invoke) — standard export pattern, no elevated concerns

---

15. Telemetry and Tracking

Agent Risk Rating: LOW

Verdict: No tracking by default.

Config Key	Default	Meaning
`USAGE_TRACKING`	`"0"` (off)	Token usage reporting
`EMIT_SESSION_TRACKING`	`"0"` (off)	Session event telemetry
`WA_BRIDGE_AUTO_UPDATE`	`"on-demand"`	WhatsApp bridge updates

- **No network calls at startup** in the main `sx` binary
- **No mandatory data collection** exists that cannot be disabled
- When tracking is explicitly enabled, data includes: session metadata, token counts, machine fingerprint (hostname, username, OS), and project context
- `EMIT_SESSION_TRACKING` (when enabled) sends **full conversation content** — users should be aware of scope

For zero-tracking corporate deployment, no configuration changes are needed — defaults are already off.

Findings:

ID	Severity	Finding
TEL-01	Low	Usage tracking guard uses opt-out pattern (`if value == "0"`) instead of opt-in (`if value == "1"`) — NULL would bypass, mitigated by config always providing default `"0"`

---

16. Install Script & Distribution Security

Agent Risk Rating: MEDIUM

The installation mechanism uses `curl|bash` / `iwr|iex` pattern. While internal components (`scorpiox-wsl` , `libsxbridge`) implement SHA256 verification, the user-facing install scripts do not.

Findings:

ID	Severity	Finding
F-01	**High**	Install scripts download archives without checksum verification — SHA256 sidecars exist but are not consumed
F-02	**High**	`get.scorpiox.net` serves install scripts over HTTP without redirect to HTTPS
F-03	**High**	No code signing or GPG signature verification on downloaded binaries
F-04	Medium	Non-atomic installation — interrupted downloads leave partial state
F-05	Medium	Embedded `scorpiox-update` mechanism has no checksum verification
F-06	Medium	Linux installer writes to `/usr/local/bin` with sudo
F-07	Medium	Windows updater `.cmd` wrapper bypasses PowerShell execution policy

F-08	Low	No HSTS header on `dist.scorpiox.net`
F-09	Low	No uninstall mechanism or documentation
F-10	Low	macOS installer modifies shell rc files without backup
F-11	Low	Branch-based distribution allows `index.txt` hijack to redirect to arbitrary branch

Positive Controls: - SHA256 checksums are generated and published alongside archives during release - Per-commit build provenance (`build-info.json`) is published - `dist.scorpiox.net` redirects HTTP→HTTPS (308 Permanent Redirect) - C-level components implement SHA256 verification (WSL: open, Bridge: fail-closed)

---

17. Consolidated Risk Matrix

#	Area	Finding	Severity	Status	Recommendation
1	Install Script	Archives downloaded without checksum verification	**HIGH**	Open	Consume existing `.sha256` sidecar files in install scripts
2	Install Script	`get.scorpiox.net` serves over HTTP	**HIGH**	Open	Enforce HTTPS redirect on install script server
3	Install Script	No code signing on binaries	**HIGH**	Open	Implement GPG or Sigstore signing for release artifacts
4	Command Injection	Windows CGI `cmd /C` with incomplete sanitization	**HIGH**	Partial	Use `CreateProcess()` with environment block instead of `cmd /C`
5	Command Injection	Git deploy handler missing single-quote escaping	**HIGH**	Partial	Use `fork()+execvp()` with argv array
6	Memory Safety	Use-after-free risk in provider vtable cleanup	**HIGH**	Latent	Document invariant or restructure to set `p->data = NULL` before `vtable->free()`
7	Privilege Access	`--privileged` bypasses namespace isolation	**HIGH**	By Design	Add explicit documentation and `--yes-i-know` confirmation
8	TLS Security	Global TLS verification toggle affects all endpoints	Medium	Mitigated	Make per-module or add persistent visual warning
9	TLS Security	Missing minimum TLS version in 2 of 3 mbedTLS contexts	Medium	Open	Add `mbedtls_ssl_conf_min_tls_version(TLS1_2)` to all contexts
10	Credential Hardcode	Default SSH user `"root"` in embedded config	Medium	Open	Change default to non-privileged service account
11	Credential Hardcode	Hardcoded private IPs in environment template	Medium	Open	Replace with placeholder values

12	Credential Hardcode	Hardcoded internal IP in release scripts	Medium	Open	Use configurable `$BUILD_HOST` variable
13	File I/O	Missing `O_CLOEXEC` on ~48 file descriptors	Medium	Open	Add `O_CLOEXEC` to all `open()` calls
14	File I/O	Gemini API key in URL query parameters	Medium	Open	Move to request header or POST body
15	File I/O	PID file TOCTOU race condition	Medium	Mitigated	Socket bind provides secondary protection
16	Buffer Safety	`snprintf` return value accumulation (3 sites)	Medium	Open	Clamp `rl` after each `snprintf` to prevent unsigned underflow
17	Command Injection	Image path injection via ffmpeg/convert shell commands	Medium	Open	Use `fork()+execvp()` with discrete argv arguments
18	Command Injection	`sx_tool_exists()` unsanitized tool name in `popen()`	Medium	Open	Use `fork()+execvp()` for `which` lookup
19	Command Injection	Windows agent tool-wait via `system()`	Medium	Partial	Migrate to `CreateProcess()` with argv
20	Command Injection	Windows SSH command via `system()`	Medium	Open	Use `CreateProcess()` or `_spawnvp()`
21	Command Injection	Windows `sx_rmrf()` via `system("rd /s /q")`	Medium	Open	Use `RemoveDirectoryW()` Win32 API
22	Privilege Access	DNS server doesn't drop privileges after port 53 bind	Medium	Open	Implement `setuid(nobody)` after bind
23	Privilege Access	`sx_system_safe()` misleading name — uses `/bin/sh -c`	Medium	Open	Rename or refactor to use `execvp()`
24	Privilege Access	Thunderbolt4 requires full root (could use CAP_NET_RAW)	Medium	Open	Support Linux capabilities as alternative
25	Install Script	Non-atomic installation	Medium	Open	Implement staging directory + atomic rename
26	Install Script	`scorpiox-update` lacks checksum verification	Medium	Open	Port `verify_sha256` from C code to update scripts
27	Install Script	Linux installer uses sudo for `/usr/local/bin`	Medium	Open	Offer user-local install option (`~/.local/bin`)
28	Install Script	Windows .cmd wrapper bypasses execution policy	Medium	Open	Document security implication; offer signed script alternative
		MSYS2 downloads			

29	Supply Chain	without integrity verification	Medium	Open	Add SHA256 checksum verification to `download.ps1`
30	Supply Chain	Unpinned Docker base image (ubuntu:22.04)	Medium	Open	Pin to specific SHA256 digest
31	Build Provenance	Bun runtime via `curl|bash` in release script	Medium	Open	Pin Bun version and verify install script hash
32	Network Endpoints	DNS server default `0.0.0.0` binding	Medium	Open	Document and/or default to `127.0.0.1`
33	Memory Safety	Thunderbolt4 buffer allocation leak on OOM	Medium	Open	Free all buffers individually in error path

---

18. Conclusion & Recommendations

Overall Assessment

SCORPIOX CODE demonstrates a **mature security posture** for a large C codebase. The software exhibits:

- **Architectural security awareness**: centralized TLS configuration, `fork()+execvp()` preference over `system()`, abort-on-OOM wrappers, `mkstemp()` for all temp files, comprehensive namespace isolation in the container runtime
- **Zero critical vulnerabilities** across 13 audit domains and 403,876 lines of code
- **Significant improvement** over the previous audit: critical findings eliminated (2→0), high findings reduced by 61%, medium findings reduced by 72%
- **Privacy-respecting defaults**: all telemetry disabled by default, no startup network activity

Priority Recommendations

Immediate (High Priority): 1. **Implement checksum verification in install scripts** — the infrastructure already generates SHA256 sidecars; install scripts need to consume them 2. **Enforce HTTPS on `get.scorpiox.net`** — add HTTP→HTTPS redirect for the install script server 3. **Migrate Windows `cmd /C` CGI path to `CreateProcess()`** — eliminate shell interpolation for HTTP-received inputs 4. **Document the provider vtable memory contract** — add explicit comment that `vtable->free()` must not free the provider struct itself, or restructure the cleanup order

Short-Term (Medium Priority): 5. Add `O_CLOEXEC` to all `open()` calls across the codebase 6. Clamp `snprintf` return values in accumulation patterns (IMAP, email builders) 7. Migrate remaining `popen()` / `system()` calls on Windows to `CreateProcess()` with argv arrays 8. Add `mbedtls_ssl_conf_min_tls_version()` to all mbedTLS contexts 9. Replace hardcoded internal IPs with configurable variables 10. Implement code signing for release artifacts (Sigstore/GPG)

Long-Term (Hardening): 11. Implement atomic installation with staging directory 12. Add privilege dropping after binding privileged ports 13. Consider DANE/TLSA or MTA-STS for outbound mail TLS 14. Add per-module TLS verification controls instead of global toggle 15. Provide uninstall mechanism across all platforms

Windows Workstation Risk Rating

For the primary deployment scenario of Windows developer workstations using the CLI tools (not running server components):

Risk: LOW — The software is safe for Windows workstation deployment. The highest-risk findings

(install script integrity, Windows CGI injection) are mitigated by: - Corporate deployment can distribute binaries via internal package management, bypassing the install script entirely - CGI/server functionality is not part of typical developer workstation use - No registry modification, UAC elevation, or service installation patterns - All telemetry disabled by default - Self-update mechanism (WSL helper) implements SHA256 verification

---

19. Appendix: Audit Methodology

Audit Agents

#	Agent	Scope
01	`supply-chain`	Package managers, vendored libraries, pre-built binaries, Docker base images
02	`build-provenance`	CMake configuration, compiler flags, security hardening, linking, release scripts
03	`network-endpoints`	Hardcoded URLs, IP addresses, domain names, ports, binding behavior
04	`tls-security`	Certificate verification, TLS versions, cipher suites, plaintext protocols
05	`credential-hardcode`	API keys, passwords, tokens, secrets, infrastructure details in source
06	`file-io`	Temp files, directory permissions, data at rest, logging, path traversal
07	`buffer-safety`	String operations, `snprintf` / `sprintf`, format strings, `scanf`, stack buffers
08	`memory-safety`	`malloc` NULL checks, use-after-free, double-free, realloc patterns, leaks
09	`command-injection`	`system()`, `popen()`, `exec*()` calls, shell interpolation, input sanitization
10	`privilege-access`	setuid/setgid, root requirements, namespace isolation, process spawning, IPC
11	`windows-deployment`	Windows binary compilation, Win32 API usage, network surface, self-update
12	`telemetry-tracking`	Data collection inventory, opt-in/opt-out behavior, startup network activity
13	`install-script`	Install scripts, update mechanism, distribution integrity, release pipeline

Tools & Techniques

- Static analysis via `grep`, `awk`, and pattern matching across all non-vendor source files
- Manual code review of all flagged patterns with context analysis
- CWE classification for all confirmed findings
- CVSS scoring for command injection findings
- Cross-reference between audit domains (e.g., TLS findings validated against network endpoint inventory)
- Comparison against industry best practices (Homebrew, rustup, nvm for install scripts)

Scope Exclusions

- Vendored library internals (Mbed TLS, yyjson) — assessed at version/CVE level only

- Runtime behavior / dynamic analysis — this is a static source code audit
- Third-party API security (Anthropic, OpenAI, Google) — out of scope
- Infrastructure security of distribution servers — limited to observable behavior

---

Report generated by automated security audit pipeline — 13 agents, unified analysis Classification: CONFIDENTIAL — For Corporate Review Only