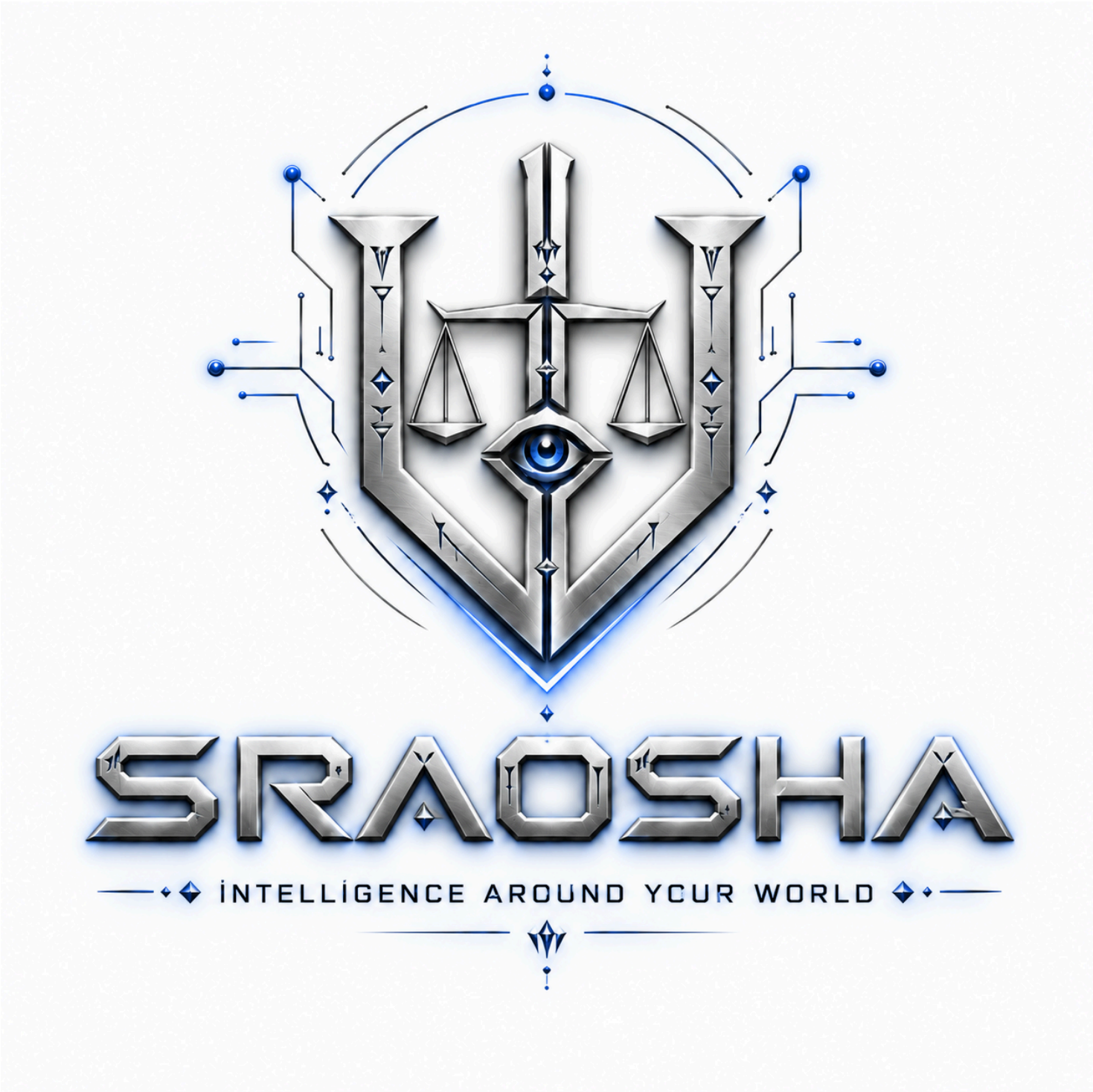


MISSION & BUILD BLUEPRINT

Security-hardening guardian. Adversarial testing counterpart.
All-domain intelligence, built as one Mainely Code system.

A vision with a build path.

Security-led product direction. All-domain scope. Evidence-bound implementation.

THREE DISTINCT LEVELS OF CERTAINTY

REPORTED The supplied work-tree review describes prior results and blockers. These have not been re-run for this document. [S1]

PROPOSED Architecture, feature sets, pass order and acceptance gates in this blueprint are engineering proposals, not completed work.

EXPERIMENTAL Frontier capabilities require evaluation and may be redesigned, deferred or rejected when evidence disagrees.

> The approved names, emblems, visual balance and SRAOSHA tagline are fixed design inputs. Approval of the artwork does not imply approval, delivery or certification of the proposed system. [S2-S3]

> **Revision 1.2 / Mainely Code.** Blue-team guardian and red-team testing direction, the owner's WSL automation callout, and mandatory UI attribution are integrated. The detailed product and interface contracts begin on pages 35–36.

Intelligence around your world.

Security leads the product. It does not limit the intelligence.

“A suit of armor around my world.”

THE OWNER’S DEFINING VISION

SRAOSHA is the security-hardening guardian. Build a persistent, owner-directed intelligence that understands context, coordinates expertise, creates useful work and protects what matters. Its security work should establish baselines, propose reversible hardening, watch for drift and verify outcomes.

DJINN-SRAOSHA is the adversarial counterpart. It brings pentesting and security exercises into selected Kali/WSL workflows. The capability is dual-use: the operator’s purpose and conduct determine whether it helps or harms. Actions, targets, consequences and evidence must remain visible.

All other features remain. Memory, research, legal work, engineering, business, IT operations, creation, devices, family logistics, learning and frontier intelligence stay in scope. Earlier projects are starting material, not the ceiling. Security is the front door—not the limit. [S2]

SRAOSHA hardens. DJINN challenges.
Evidence closes the loop.

Everything is the design horizon, not a blanket grant of access. Ownership, recoverability and inspectable results stay central as capability grows.

Mission and product direction are synthesized from Justin’s instructions in this conversation. [S2]

Power that stays accountable.

The next-next-generation claim must eventually be demonstrated, not announced.

DESIGN PRINCIPLE	WHAT THE OWNER SHOULD NOTICE
01 / Continuity	Goals, preferences, project history and unresolved questions survive beyond one conversation.
02 / Grounding	Important claims lead back to inspectable sources. Observation, inference and speculation remain distinguishable.
03 / Coordination	Specialists contribute through one shared task contract, with dissent and verification preserved.
04 / Initiative	The system can propose the next useful step; action remains bounded by the owner's active permissions.
05 / Recoverability	Work can be paused, resumed, reviewed and, where supported, rolled back without losing its history.
06 / Improvement	New models, skills and strategies enter through controlled evaluation, not self-issued trust.

> **The intended loop:** understand the goal → gather evidence → form a plan → check authority → act within bounds → verify the result → preserve the receipt and the lesson.

Proposed product charter. The historical build contract already denies execution and approval authority to model output. [S1]

Strong local work. Real gaps.

Latest review within the supplied handoff; not a fresh inspection of your machine. [S1]

1,341
reported passing tests

9 / 11
release gates passing / blocked

4,892
reference-only donor skill cards

NOT GA
reported release posture

EVIDENCE LENS	REPORTED POSITION
Described as present	Guided legal, research, coding, knowledge, operations and Growthroom workflows; local review checks; encrypted backups.
Runtime caveat	The review reports a passing doctor run, then a later direct check with an unreachable local model endpoint. Live chat was unavailable at that later check.
Not established	Live Kali execution, broad PC control, runnable generated code, authenticated production adapters, encrypted live storage, or fresh-PC delivery.
Packaging & parity	An unsigned source ZIP; no bundled Python runtime, models or user data. P.S.T. had not received the latest hardening pass.

Do not convert test count into feature completeness. The handoff explicitly records `all_donor_features_complete: false`. The underlying receipts and linked readiness reports were not independently supplied or inspected here.

Source: Pasted markdown.md, final work-tree review. Historical “ULTRON” naming is retained only where needed for traceability. [S1]

Close the gaps in the right order.

Proposed triage of the source's blockers, not a reconstruction of its unnamed eleven release gates.

ORDER	WORKSTREAM	EXIT INTENT
P0	Runtime truth	Recover the existing model service; expose availability, model identity, failure and cancellation honestly.
P0	Live-data protection	Protect active stores, indexes and temporary artifacts; prove migration, recovery and key rotation.
P0	Authority & isolation	Keep model proposals separate from policy; provision execution boundaries before enabling code or Kali.
P1	Desktop acceptance	Complete keyboard, screen-reader, scaling, focus, interruption and representative-user checks.
P1	Domain credibility	Maintain the legal corpus; deepen research evidence; connect selected production systems through tested adapters.
P1	Delivery & P.S.T.	Build install/update/rollback paths, clean-machine proof, and separately authorized edition parity.
P2	Frontier expansion	Evaluate richer perception, multi-specialist reasoning, persistent goals and measured adaptation after baseline stability.

> **First action in a real build pass:** read the active ledger, reconcile unfinished work, run the present regression and doctor, then freeze an honest baseline. Do not restart an active batch because a new blueprint exists.

Reported blockers and active-work preservation requirements: [S1]. Priority order and exit intents: proposed in this blueprint.

One coherent system.

A proposed shared core with domain workspaces above it and controlled execution beneath it.

01 OWNER EXPERIENCE

Conversation · mission board · evidence explorer · approvals · accessible desktop

02 INTELLIGENCE CORE

Task state · model routing · planning · critique · verification · resource budgets

03 CONTEXT & MEMORY

Source store · claim ledger · project graph · personal context · retention controls

04 AUTHORITY BOUNDARY

Identity · policy decisions · exact-plan approvals · leases · revocation · audit

05 EXECUTION & INTEGRATION

Typed adapters · isolated code jobs · DJINN broker · approved provider connections

06 EVIDENCE & RECOVERY

Artifacts · provenance manifests · restore points · result verification · replay

> **Crucial boundary:** intelligence can recommend an operation. A separate deterministic policy and execution layer decides whether the exact operation is permitted and whether its preconditions still hold.

Architecture proposal, informed by the existing closed-dispatcher contract [S1] and excessive-agency guidance [W2].

A brain, not a prompt wrapper.

Make reasoning persistent, inspectable and useful across changing tasks.

C01

Mission state

Represent the goal, constraints, dependencies, open questions, artifacts and stopping conditions in a durable task record. Resume after interruption without pretending unfinished work succeeded.

C02

Specialist reasoning

Use planner, researcher, builder, critic and verifier roles only where evaluation shows benefit. Preserve disagreement. Shared model weaknesses do not become independent proof.

C03

Adaptive routing

Retain the handoff's 4B chat and small work models; its review names qwen3.5:4b. Select heavier or external routes only after capability, latency, privacy and resource checks, with explicit owner configuration. [S1]

C04

Measured initiative

Surface stale evidence, approaching obligations and blocked dependencies. Let the owner choose notification and automation budgets rather than creating a constant stream of interruptions.

Core task contract: goal + evidence snapshot + allowed tools + budget + verification criteria + owner policy. Every specialist receives that contract; none can enlarge its own authority.

All four capability specifications are proposed. The handoff's small-model baseline and no-unapproved-sub-agent rule remain in force until changed by the owner. [S1]

Memory that can explain itself.

Not a hidden pile of embeddings: a managed record of what is known, inferred and still unresolved.

MEMORY OBJECT	PROPOSED CONTRACT
Source	Original artifact, origin, capture time, content hash, access scope and permitted uses.
Claim	A statement with evidence spans, confidence rationale, source freshness and contradiction links.
Context	Project relationships, owner preferences, task history and unresolved decisions; sensitive context separated.
Revision	An explicit supersession or retirement event. Do not silently rewrite the history of why a decision was made.
Permission	Read, retain, share and export rules bound to the relevant person, workspace and purpose.
Derived index	Search and retrieval structures that inherit permissions and are rebuilt or invalidated when sources change.

REQUIRED OWNER CONTROLS

Inspect a memory. Correct it. See its sources. Restrict its use. Export it. Retire or delete it under a visible retention policy. Show the consequences for derived indexes, caches and backups instead of promising erasure that has not been verified.

> **Acceptance example:** a revoked source no longer supports a new answer; the answer explains what evidence is now missing, while authorized historical audit records preserve the reason for the earlier decision.

Memory model and acceptance example: proposed. Exact-review revocation and history behavior are described in the earlier batch report. [S1]

A command center that feels personal.

One approachable surface; depth appears when the work needs it.

EXPERIENCE

Conversation + mission board

Move from “help me with this” into an inspectable mission: what is happening, what is waiting, what needs approval and where the evidence lives. Keep normal conversation usable during background jobs.

VISIBILITY

No fake progress

Display planned, queued, running, blocked, cancelled, failed and verified states accurately. Show a tool or provider outage as an outage, not a fabricated answer or a forever-spinning indicator.

ACCESSIBILITY

A first-class desktop

Design for keyboard-only use, meaningful focus, screen readers, scaling, contrast and reduced motion. Security and approval states must remain understandable without color alone.

P.S.T. EDITION

Shared core. Separate world.

Retain the son’s distinct emblem and edition identity. Use separate memories, credentials and permissions. Proposed DJINN access begins with an explicitly configured learning lab, not inherited administrator access.

Voice is optional. Final-response playback should be opt-in, cancellable and private by design. The source describes Fish voice as integrated but disabled, without credentials, consent or live playback validation. [S1]

MAINELY CODE / PERSISTENT UI CREDIT

Proudly Built with Mainely Code Buildroom—ADVANCED EDITION

Both modes, footer and status bar. Wrap rather than truncate. Full contract: page 36.

Experience and edition policies are proposals. P.S.T. parity remains a separate delivery obligation; this PDF does not claim it has been updated. [S1-S2]

Understand the world.

Twelve feature families define the intended product breadth. Each family needs its own evidence.

F01 / KNOWLEDGE

A living personal library

Connect approved documents, notes and project records; search by meaning and exact evidence; reconcile contradictions; assemble cited briefs.

Release proof: permission-aware retrieval and a source-backed answer that survives a stale-source test.
Baseline: knowledge workflows reported. [S1]

F02 / RESEARCH

Questions into defensible findings

Plan research, compare sources, record uncertainty and preserve citations. Extend beyond bounded HTTPS reads into validated document and browser workflows.

Release proof: grounded synthesis with source dates, extraction checks and a failed-fetch path.
Baseline: bounded reads reported. [S1]

F03 / LEGAL WORK

Evidence into reviewable work

Organize matters, retrieve jurisdiction-specific authorities, build timelines and draft review-required documents.

Release proof: maintained official corpus, currentness receipts, verified forms and qualified review where required.
Baseline: local Maine workflows, not filing readiness. [S1]

F04 / PERSONAL OPERATIONS

Continuity across responsibilities

Maintain goals, obligations, checklists and shared plans using selected, authorized connections. Explain changes before sending, scheduling or committing anything.

Release proof: a complete planning-to-confirmed-update workflow with cancellation and duplicate prevention.
Baseline: this family is not proven by the handoff.

Feature definitions and release proofs are proposed. Baseline statements are limited to what [S1] reports; “not proven” does not mean no code exists.

Build and operate the world.

Useful work must finish in owned artifacts and verifiable outcomes.

F05 / ENGINEERING

From intent to tested software

Inspect a workspace, plan changes, create patches, execute approved isolated tests, explain differences and prepare a reviewable package.

Release proof: a patch-to-test receipt with dependency isolation and an unchanged host.
Baseline: generated code is intentionally inert. [S1]

F06 / IT & INFRASTRUCTURE

Operational intelligence

Unify selected inventory, documentation, network and service workflows. Correlate changes and propose recoverable maintenance.

Release proof: a least-privilege adapter with fresh reads, an approved write and rollback or explicit compensation.
Baseline: deterministic local workflows only. [S1]

F07 / BUSINESS & GROWTH

A coordinated business workspace

Research leads, draft campaigns, manage content and connect CRM or publishing systems under explicit approval and consent rules.

Release proof: truthful preview, scoped publication, duplicate prevention and a receipt.
Baseline: Growthroom UI is reported; live adapters are not. [S1]

F08 / CREATION & DOCUMENTS

Ideas into usable artifacts

Produce structured reports, presentations, interfaces, design briefs and media plans. Preserve source provenance and approved brand assets.

Release proof: an editable artifact plus a rendered quality check and transparent source attribution.
Baseline: complete creative coverage is not established.

Feature definitions and release proofs are proposed. Current-state limitations come from [S1].

Extend reach without losing control.

The broadest capabilities should still feel like one accountable system.

F09 / DEVICE WORKFLOWS

Perceive, explain, then operate

Understand explicitly shared screens and device state; propose a plan; add authorized desktop action only behind a tested execution boundary.

Release proof: focus-safe, cancellable actions that stop when the screen or target changes.
Baseline: observation/focus only. [S1]

F10 / HOME & FAMILY

A helpful shared environment

Support household projects, schedules, maintenance and voluntarily shared information. Separate people's private contexts and show all enabled observation.

Release proof: transparent consent, revocable sharing and no cross-profile leakage.
Baseline: this broader family is not proven.

F11 / LEARNING & ADAPTATION

Grow with the owner

Tutor from selected sources, retain progress, surface knowledge gaps and turn successful workflows into reviewable reusable skills.

Release proof: improved held-out task performance without silently altering permissions or accepted facts.
Baseline: frontier work requires evaluation.

F12 / SECURITY / TWO LENSES

Guardian + adversarial testing

SRAOSHA baselines, hardens and watches drift. DJINN-SRAOSHA tests controls through scoped Kali/WSL workflows and returns evidence for repair.

Release proof: a lab cycle with before/after state, a reversible fix, independent retest and reliable stop.
Baseline: Kali catalog, inventory and planning only. [S1]

Feature definitions and release proofs are proposed. Security-mode naming and broad mission: [S2]. Reported PC/Kali limits: [S1].

Consolidation is not completion.

Bring the work together without importing conflicting authorities or unsupported claims.

A donor feature enters as a candidate, not as a capability. Preserve its source version, license, dependencies, evidence and limitations. Map it onto the shared SRAOSHA identity, memory, policy and audit contracts before exposing it to the owner.

INTAKE STAGE	REQUIRED OUTPUT
1 / Discover	Locate the actual implementation and tests. A skill card alone is reference material.
2 / Admit	Review source, license, dependencies and security; record the exact pinned version.
3 / Adapt	Translate into the shared contracts. Avoid a second identity, credential, approval or policy authority.
4 / Expose	Make the user journey reachable through the dispatcher, CLI or native UI.
5 / Verify	Run targeted and full regression checks, adverse-path tests and provenance checks.
6 / Transfer	Package compatible migrations, artifacts and parity evidence; validate each edition independently.

> **P.S.T. is a separate destination.** Prepare a transfer-ready package while the original write boundary remains active. Do not modify the other workspace or inherit its permissions without fresh authorization. The source explicitly reports missing parity. [S1]

Proposed intake workflow, preserving the source's read-only-donor and edition-transfer requirements. [S1]

The armor protects its own memory.

Privacy is a system property: stores, indexes, logs, credentials, caches and recovery paths all matter.

DATA AT REST

Encrypt the live path

Select and test an encryption design for active records, documents, queues and derived indexes. Cover journals and migration leftovers, not only backups. A locked vault must not break auditability.

KEYS & RECOVERY

Make loss survivable

Separate keys from content; use an appropriate OS-protected secret store. Prove rotation, interrupted migration, recovery and wrong-key behavior before declaring the path usable.

ACCESS & EXPORT

Scope every disclosure

Keep credentials out of prompts and logs. Limit adapters to the necessary account and task. Preview exported evidence, redact deliberately and retain the export decision.

ADVERSARIAL INPUT

Treat content as evidence

Documents, web pages and tool output are not commands or approval. Quarantine untrusted instructions and enforce policy outside model text. Prompt injection remains a risk to test. [W1-W2]

Threat-model limit: storage encryption is not a promise that an already-compromised, unlocked host cannot access session data. Publish the actual trust assumptions and measured recovery guarantees.

Source [S1] reports plaintext live data and encrypted backups, with rotation/recovery/migration still missing. The controls above are proposed.

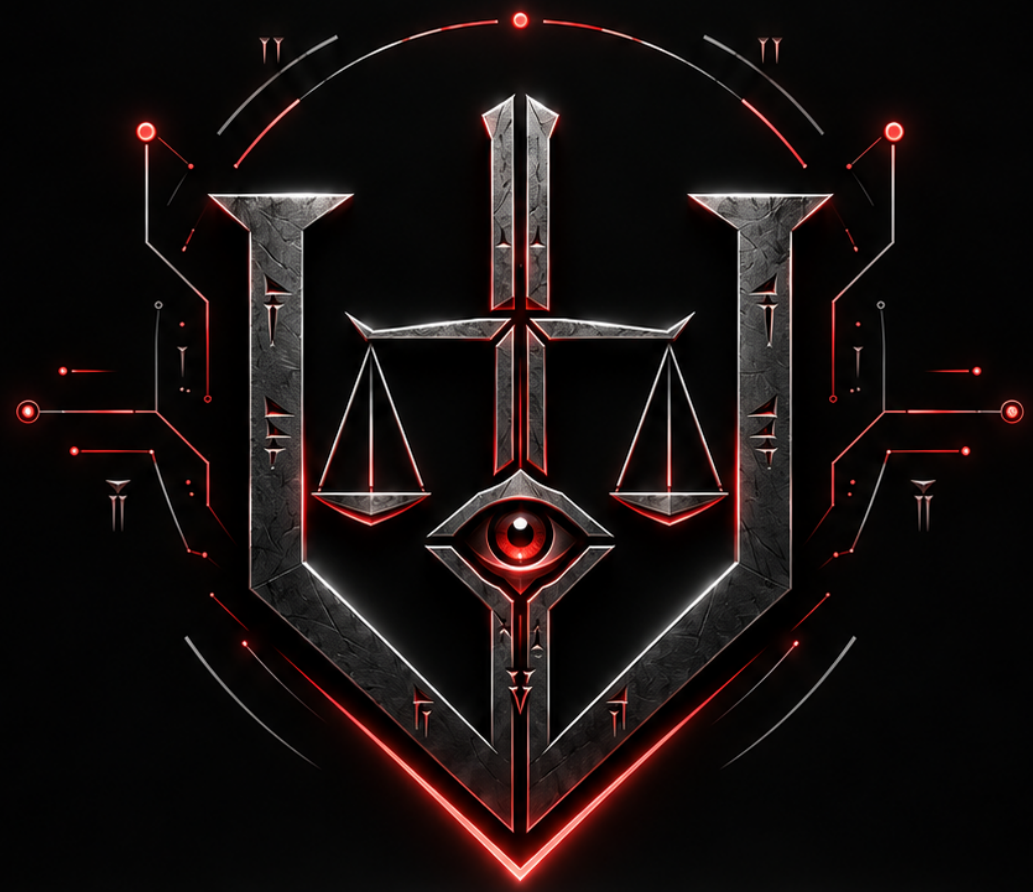
Earn the next-next-generation leap.

Research ambitions become experiments with baselines, failure criteria and promotion gates.

EXPERIMENT	INTENDED ADVANCE	PROMOTION QUESTION
Persistent world model	Track entities, commitments and changes over time.	Can it resolve a later task with fewer corrections than retrieval alone?
Multimodal understanding	Combine owner-shared screen, text, image and optional audio context.	Does it improve held-out tasks without raising privacy or action errors?
Collective reasoning	Route complementary specialists and preserve disagreement.	Does it beat the best single route at matched cost and latency?
Counterfactual planning	Compare possible plans and expected consequences before action.	Do predictions match outcomes in controlled, reversible tasks?
Skill acquisition	Propose reusable workflows from successful work.	Can reviewed skills generalize without widening permissions?
Long-horizon initiative	Maintain goals and propose timely, useful next steps.	Does the owner accept more suggestions than they dismiss or undo?

Promotion rule: experimental capabilities remain opt-in and clearly marked until they improve an agreed benchmark and survive safety, privacy and regression gates. There is no claim here of human-level general intelligence or completed self-improvement.

Original research agenda for the user's stated ambition [S2]. Evaluation discipline is informed by NIST AI RMF [W4]; no certification is claimed.



DJINN-SRAOSHA

PENTESTING / KALI CONTROL MODE

SEE THE EXPOSURE.
PROVE THE RISK.
RESTORE THE BALANCE.

An engagement, not a terminal.

Operator-directed adversarial testing; the broader intelligence remains SRAOSHA.

Power is not the ability to run every command. It is knowing exactly what should run next.

DJINN is the adversarial counterpart to SRAOSHA's hardening guardian: a Windows-facing control experience for selected Kali workflows through WSL. Pentesting capability is dual-use; a product name cannot determine the operator's intent. The design exposes the target, plan, impact, authority and evidence rather than treating a label as proof of good use. [S2]

BEFORE

Clarity of intent

Show the authorized assets, objectives, excluded systems, testing window and stop conditions before an operation begins.

AFTER

Clarity of outcome

Distinguish observations from confirmed findings, propose a remediation plan, and verify the result through a controlled retest.

Current boundary: the supplied review reports only a Kali catalog, WSL inventory and planning. The command experience described in this section is a proposed destination, not an installed live execution capability. [S1]

THE OWNER'S VISION / WSL ON WINDOWS

What's better than a little Blue Team vs. Red Team security exercise with Kali Linux?

Automating the hell out of it through WSL on Windows.

A visible perimeter. A living mission.

Proposed interface concept / synthetic training engagement / no operation executed.

DJINN-SRAOSHA MODE / PLAN ONLY

SCOPE	AUTHORITY	NETWORK
LOCAL-LAB-A	AWAITING OWNER	EGRESS CLOSED

ASSET REGISTER

- **LAB-APP-01** — Allowed / synthetic
- **LAB-DATA-01** — Allowed / synthetic
- **Discovered endpoint** — Out of scope / blocked

PROPOSED WORKFLOW

- **01 / Validate scope** — Ready for owner review
- **02 / Collect inventory** — Queued after approval
- **03 / Review findings** — Waiting for evidence
- **04 / Remediate + retest** — Separate authorization

PLAN HASH / shown at approval REVOKE / STOP

Proudly Built with Mainely Code Buildroom—ADVANCED EDITION

Permanent visibility: environment, scope, policy lease, tool provenance, execution state, evidence progress and stop control. The selected mode must never be inferred from red lighting alone.

Progress means state, not theatre. A step remains “awaiting evidence” until artifacts arrive and checks complete. Interrupted work gets a truthful partial result, not a success badge.

Interface concept created for this blueprint. All labels are illustrative; no target inventory, live session, scan or tool result is represented.

Kali is the tool environment. The owner stays in command.

BOUNDARY	PROPOSED RESPONSIBILITY
1 / Mission shell	Captures the authorized objective and displays the proposed plan, evidence and approval state.
2 / Scope broker	Checks the exact action, environment, asset scope, time window, risk class and remaining budget.
3 / Execution adapter	Builds typed argument lists from reviewed templates; no model-generated shell string is executed by default.
4 / Isolated worker	Runs a pinned tool in an explicitly provisioned environment with restricted mounts, credentials and network paths.
5 / Result normalizer	Retains raw artifacts, parses tool output and identifies ambiguity, tool failure and evidence gaps.
6 / Verifier & receipt	Corroborates conclusions, records limitations and links the result to the authorized plan and execution record.

ENVIRONMENT POLICY

Windows + a dedicated Kali WSL 2 profile is the intended everyday lane. Provision it explicitly; inspect mounts, Windows interoperability, network reachability, privileges and resource limits. Never silently reuse a shared developer distro. The profile and its boundaries need their own acceptance evidence. [W5–W7]

Higher-risk or hostile-code jobs route to a separately hardened, tested VM or remain unavailable. WSL installation alone is not a blanket isolation guarantee. Win-KeX can expose a desktop, but does not replace the scope broker or expand authority. Hardware-specific workflows need separate evidence. [W8]

Execution architecture is proposed. WSL/Win-KeX platform facts: [W5-W8]. No distro registration or environment provisioning is authorized by this PDF.

From visibility to verified resilience.

A capability ladder, not a promise that every Kali tool is supported.

CAPABILITY	INTENDED WORK	CONTROL
D01 / Observe	Review supplied logs, configurations, artifacts and existing inventories.	No target contact required for artifact-only work.
D02 / Discover	Run bounded service and asset discovery within an explicit engagement scope.	Newly discovered assets do not automatically become authorized.
D03 / Assess	Review selected application, host, identity and network controls.	Every tool profile declares supported versions, privileges and expected impact.
D04 / Validate	Corroborate a suspected weakness with separately approved, bounded test methods.	Intrusive validation needs a distinct risk decision and recovery plan.
D05 / Investigate	Organize evidence, correlate events and explain uncertainty.	Do not turn one tool alert into a confirmed compromise claim.
D06 / Guardian fix	Hand verified findings to SRAOSHA for a reviewed, reversible hardening plan.	Separate change approval and recovery plan; return the new state for DJINN retest.
D07 / Retest	Repeat the relevant controlled checks and compare evidence.	A finding closes only when the defined retest supports closure.

Selection over sprawl: admit a small, proven tool set first. Expand only when the next adapter brings a useful end-to-end outcome, not just another command in a catalog.

Proposed DJINN capabilities. Current catalog-only/planning-only boundary: [S1]. No operational exploit procedures are included.

Scope is a contract that can expire.

Authorization is checked before dispatch and throughout the job—not only once at the start.

01 DEFINE

Record assets, ownership/authorization, objectives, exclusions, time window, allowed techniques and recovery contacts.

02 SEAL

Bind the approved plan to an exact revision, tool manifest, environment and budget. Changed scope requires a new review.

03 DISPATCH

Issue a short-lived capability lease. The worker receives only the permissions needed for that one job.

04 OBSERVE

Monitor process state, network destinations, elapsed time and resource budgets. Block unexpected scope changes.

05 VERIFY

Separate raw evidence, parser output, model interpretation and corroborated findings. Preserve missing evidence.

06 CLOSE

Stop workers, revoke leases, preserve artifacts, clean the environment and produce the report plus any retest plan.

> **Revocation must reach the worker.** Cancelling an approval must prevent new dispatch and trigger containment of active work. Do not claim to undo packets already sent or irreversible side effects.

Lifecycle proposal; exact-review revocation in the handoff is a foundation to extend, not evidence that live-process revocation already works. [S1]

The strongest demo is a controlled stop.

A credible control mode proves its boundaries under failure as well as success.

REHEARSAL	ADVERSE CONDITION	EXPECTED PROOF
Scope escape	A redirect, changed address or newly discovered host is outside the sealed scope.	Dispatch is blocked; the discrepancy is recorded.
Stale authority	Approval expires or is withdrawn while work is queued or running.	No new work starts; active-worker containment is attempted and measured.
Tool compromise	A tool output includes instructions, bogus results or an unexpected child process.	Output is untrusted; process and result contracts reject the deviation.
Runtime failure	A worker crashes, stalls, fills its budget or loses the host connection.	Job ends truthfully; evidence is preserved; the environment is quarantined.
Evidence failure	An artifact is missing, changed, duplicated or cannot be parsed.	The finding remains unverified; no fabricated substitute appears.
Edition isolation	A P.S.T. task requests the owner's credentials or engagement records.	Access is denied and the attempted boundary crossing is visible.

Evidence standard: retain the plan revision, policy decision, environment/tool identifiers, start/stop events, raw artifact hashes and verification outcome. Hashes help detect changes; they do not, alone, prove authenticity or prevent a privileged rewrite.

Original failure-rehearsal suite. Prompt-injection and excessive-agency threat categories informed by [W1-W2].

First a lab. Then earned trust.

Execution capability advances only after the preceding boundary is demonstrated.

GATE	PROMOTION	MINIMUM EXIT EVIDENCE
D-G0	Plan-only integrity	Scope editor, catalog provenance and replayable plan export. No live target execution.
D-G1	Broker-only rehearsal	Synthetic workers prove expiry, revocation, quotas, duplicate handling and scope denial.
D-G2	Isolated local lab	An approved environment runs the first low-impact tool profiles against synthetic fixtures.
D-G3	Corroborated assessment	Raw artifacts, normalized findings and independent checks produce an auditable report.
D-G4	Guardian fix + retest	One SRAOSHA lab fix is approved, applied, checked by DJINN and safely reverted where supported.
D-G5	Limited real pilot	A separately authorized engagement proves operations, stop behavior, reporting and recovery.
D-G6	Release profile	Signed packaging, tool/version support matrix, installation proof and scoped product claims.

DJINN should be formidable because its actions are disciplined and its findings withstand inspection—not because its interface hides uncertainty. Every promotion needs a new receipt, not a renamed status.

Proposed gate sequence. The source supports planning/catalog status only; these gates have not been executed for this document. [S1]

A dependency-led path to release.

Sequence gates before calendar promises. Duration depends on the re-baseline, hardware, adapters and accepted scope.

GATE	FOCUS	EXIT INTENT
G0	Re-baseline	Freeze the actual tree; reconcile active work; re-run existing proof; establish a versioned change record.
G1	Usable local core	Restore model access; expose honest state; complete basic desktop, cancellation and offline behavior.
G2	Trusted data	Prove protected live storage, source provenance, scoped memory and recovery on representative data.
G3	Controlled action	Complete deterministic policy, approval revisioning, budgets and the first isolated execution path.
G4	Useful verticals	Deepen research and legal evidence; deliver one real adapter; validate donor and P.S.T. transfer contracts.
G5	DJINN laboratory	Prove the scope broker, isolated lab, tool profiles, evidence normalization, revocation and retest.
G6	Supported pilot	Selected users and authorized systems exercise complete journeys with telemetry and recoverability.
G7	Release candidate	Signed delivery, clean-machine and upgrade proof, accessibility checks, support matrix and known limits.
G8	Measured expansion	Promote additional domains and frontier capabilities only through the same evidence discipline.

Not a replacement for the active ledger. The next pages define twenty proposed delivery slices. Match them against existing work before assigning real batch numbers or marking anything complete.

Proposed sequence; the supplied build contract requires continuous preservation of active work and receipt-backed batch verification. [S1]

Wave A / Establish reality

Re-baseline, runtime and protected data.

01 Reproducible starting point

Build: Expose a baseline report through the existing CLI: source hashes, current tests, doctor, missing dependencies and unfinished ledger entries.
Accept: A changed file invalidates the old receipt; a missing dependency is shown as blocked, not passed.
Receipt: Baseline manifest + regression log. **Depends:** Existing workspace only.

02 Local conversation recovery

Build: Repair access to the already installed model service; display route identity, unavailable state and cancellable responses.
Accept: Prove success, unreachable service, timeout and cancellation without downloading a new model.
Receipt: Live smoke receipt + failure fixtures. **Depends:** 01; local-service permission.

03 Desktop acceptance journey

Build: Complete a keyboard-only path from startup to evidence review, approval denial and export.
Accept: Check focus, screen-reader names, scaling and interrupted-session recovery on the actual desktop.
Receipt: Manual acceptance record + UI tests. **Depends:** 02; interactive desktop access.

04 First encrypted live store

Build: Implement the selected protected-storage path for one bounded record family, including its journal and temporary files.
Accept: Migrate a synthetic fixture, interrupt the migration, recover it and inspect for plaintext residue.
Receipt: Migration/recovery receipt. **Depends:** 01; crypto design decision.

05 Key rotation and restore

Build: Expose rotate, backup and restore for that protected store with explicit operator confirmation.
Accept: Wrong keys fail safely; interrupted rotation recovers; a restored record matches its provenance.
Receipt: Rotation + restore rehearsal. **Depends:** 04; key/recovery policy.

All slices are proposals. Completion requires exposed behavior, synthetic acceptance, full regression, doctor and a tested-byte receipt. [S1]

Wave B / Memory and safe work

Shared contracts before broader authority.

06 Source-backed memory revision

Build: Expose claim/source inspection, correction and retirement with derived-index invalidation.
Accept: A retired source cannot silently support a new answer; prior authorized history remains explainable.
Receipt: Memory lifecycle fixture. **Depends:** 04-05; retention decision.

07 Exact-plan action approval

Build: Bind a typed proposed operation to its parameters, scope, evidence revision and expiring permission.
Accept: An edited plan, revoked approval or duplicate request cannot obtain a fresh execution by accident.
Receipt: Policy and adverse-path receipt. **Depends:** 01; no live target required.

08 Research extraction vertical

Build: Take one permitted document or HTTPS source through fetch, extraction, source-span citation and a reviewable brief.
Accept: Reject unsupported content, malformed input and stale evidence; preserve a useful partial result.
Receipt: Source-to-brief evidence pack. **Depends:** 06; approved access policy.

09 Legal currentness vertical

Build: For a bounded Maine authority set, capture official-source provenance and source-date checks in the draft workflow.
Accept: A missing or stale authority blocks a currentness claim; the document remains review-required.
Receipt: Corpus receipt + draft fixture. **Depends:** 08; corpus stewardship.

10 First runnable code job

Build: Run one generated synthetic program in a provisioned isolated environment through the closed dispatcher.
Accept: Deny unapproved network and host writes; enforce time/resource limits; preserve failure output.
Receipt: Host-isolation + test receipt. **Depends:** 07; provisioning approval.

All slices are proposals. Completion requires exposed behavior, synthetic acceptance, full regression, doctor and a tested-byte receipt. [S1]

Wave C / Useful connected work

Turn integration into end-to-end outcomes.

11 One real read-only adapter

Build: Select a single production system and expose a scoped read with account identity, freshness and evidence.
Accept: Wrong account, expired credentials, pagination and rate limits produce controlled behavior.
Receipt: Adapter contract + live read proof. **Depends:** 07; owner-selected credentials.

12 One recoverable approved write

Build: Extend that adapter with one previewable operation, duplicate protection and supported rollback or compensation.
Accept: A changed preview needs reapproval; retry does not duplicate work; partial failure is visible.
Receipt: Write/recovery receipt. **Depends:** 11; explicit write scope.

13 Optional voice journey

Build: Enable configured, consented final-response playback with cancellation, text fallback and privacy controls.
Accept: Cancel mid-playback; reject absent consent; survive provider outage without blocking chat.
Receipt: Playback smoke + failure proof. **Depends:** 02; voice credential/consent.

14 One donor-to-edition transfer

Build: Admit one implemented donor feature into shared contracts and create a P.S.T. parity package.
Accept: Validate exposed workflow and migration in isolated fixtures; leave the other workspace untouched.
Receipt: Pinned donor manifest + parity pack. **Depends:** 06-07; transfer scope only.

15 DJINN scope-to-plan journey

Build: Create, review, seal and export a synthetic engagement plan from the native interface.
Accept: Excluded assets, missing ownership evidence and altered plan revisions prevent promotion.
Receipt: Scope receipt + UI acceptance. **Depends:** 07; no target execution.

All slices are proposals. Completion requires exposed behavior, synthetic acceptance, full regression, doctor and a tested-byte receipt. [S1]

Wave D / DJINN earns execution

A small lab engagement with strong boundaries.

16 DJINN broker and lease

Build: Expose a typed tool job to a synthetic worker with environment binding, expiry, quotas and cancellation.
Accept: Expired or revoked leases fail before dispatch; duplicate submissions stay controlled.
Receipt: Broker adversarial fixture. **Depends:** 15; synthetic execution only.

17 Provision one isolated lab

Build: Add an explicitly approved runtime profile with pinned tools, restricted integration and reset capability.
Accept: Inspect mounts, interop, network policy and cleanup; fail closed when a required boundary is absent.
Receipt: Environment and isolation receipt. **Depends:** 16; new provisioning authority.

18 First scoped inventory run

Build: Use a reviewed low-impact profile against allowlisted synthetic lab assets and normalize its artifacts.
Accept: An out-of-scope destination is denied; tool errors remain errors rather than inferred results.
Receipt: Lab inventory evidence pack. **Depends:** 17; sealed lab engagement.

19 Finding-to-report verification

Build: Link a lab observation to raw artifacts, corroboration, limitations and a redacted engagement report.
Accept: An unsupported finding cannot be promoted; changed artifacts invalidate verification.
Receipt: Finding/report provenance bundle. **Depends:** 18; verification rubric.

20 Revoke, recover and retest

Build: Exercise live revocation plus one approved reversible lab change and a scoped retest.
Accept: Stop/revoke behavior is measured; cleanup is checked; closure requires valid retest evidence.
Receipt: Stop/recovery/retest receipt. **Depends:** 16-19; distinct change approval.

All slices are proposals. Completion requires exposed behavior, synthetic acceptance, full regression, doctor and a tested-byte receipt. [S1]

The first twenty are not the finish line.

Use the initial sequence to establish trustworthy delivery machinery, then widen the product deliberately.

EXPANSION	WHAT REMAINS	PROMOTION EVIDENCE
Data breadth	Expand protected storage beyond the first record family to documents, transcripts, queues, indexes, exports and caches.	Per-store migration, recovery and leakage tests.
Domain depth	Complete the twelve feature families against a published capability matrix rather than “all donor features” marketing.	End-to-end user journeys and domain review.
Research & legal	Broaden official-source maintenance, document extraction, forms checks and professional review paths.	Corpus ownership, freshness and extraction receipts.
Adapters & desktop	Add selected connected systems and supervised desktop operations one contract at a time.	Identity isolation, permission, retry and recovery proof.
DJINN coverage	Add tool profiles and environment types only after scope, containment, output parsing and impact tests.	Adapter-specific support and limitation records.
Edition parity	Apply approved transfer packages to P.S.T. and verify independently.	Migration proof and explicit permission separation.
Frontier intelligence	Promote useful memory, multimodal and coordinated-reasoning experiments.	Held-out improvement with bounded cost and risk.

Completion is a matrix of supported outcomes. A feature can be available in planning mode, supported in a lab, or ready for a limited pilot without being generally released.

Original continuation plan; it deliberately avoids presenting the proposed first twenty slices as complete-product delivery.

Measure outcomes, not mythology.

A larger test suite is useful only when it exercises the behaviors the owner depends on.

LENS	MEASURE	EVIDENCE METHOD
Grounded answers	Citation support, source freshness, contradictions and honest abstention.	Use held-out questions with inspectable sources and human review.
Memory quality	Recall, correction, retirement and permission isolation.	Test stale, revoked and cross-profile material.
Task execution	Completion, correct state, cancellation, replay and recovery.	Inject failures at every external boundary.
Action safety	Out-of-scope denial, approval freshness and privilege separation.	Require every fixed must-deny fixture to pass before release.
DJINN findings	Supported observations, corroboration, false positives and evidence completeness.	Use synthetic known-state labs plus authorized pilot review.
Usability & accessibility	Task success, focus, screen-reader usability and scaling.	Complete representative desktop journeys, not screenshots alone.
Performance & resources	Latency, memory, disk growth, CPU/GPU use and interruption cost.	Publish measured p50/p95 on declared hardware and model routes.

Set budgets after measurement. No universal response time, accuracy percentage or hardware capacity is claimed here. Zero failures in a fixed test suite is a release gate, not proof that real-world failure is impossible.

Proposed evaluation rubric. Risk-based evaluation approach informed by NIST AI RMF [W4].

Ship a system—not a source ZIP.

The release claim must match what a fresh machine and a real operator can actually do.

PACKAGE

Install / update / remove

Choose the supported installer and distribution path. Define runtime/model acquisition, data locations, upgrades, rollback and clean uninstall. Preserve user data intentionally, not accidentally.

PROVENANCE

Build / sign / verify

Produce dependency and license inventory, pinned artifacts, build provenance, malware scanning appropriate to the release, and signed deliverables. A signature proves origin/integrity—not correctness.

ACCEPTANCE

Fresh-machine evidence

Test supported Windows configurations, offline/degraded operation, account isolation, scaling and accessibility. Include interrupted update, missing dependency and recovery rehearsals.

SUPPORT

A truthful capability matrix

Publish supported routes, environments, adapters and tool versions. Distinguish plan-only, lab-only, pilot and released features. Provide diagnostics, known limitations and a rollback/support runbook.

Release decision: all required gates pass on the packaged bytes; unresolved risks and exclusions are explicit; recovery is demonstrated; and the owner signs off on the actual capability scope. P.S.T. and DJINN need their own relevant acceptance records.

Source [S1] describes an unsigned source ZIP with fresh-PC and USB testing outstanding. Proposed release practices align with SSDF [W3].

Keep the receipts. Preserve the tree.

The historical implementation boundary remains binding until explicitly changed by the owner.

Existing workspace contract: work only under `D:\dev\ULTRON`; keep runtime artifacts under `.ultron`; treat other repositories as read-only donors. Do not infer permission for global installs, model downloads, cloud deployment, remote actions, WSL registration, broad PC control or sub-agents from this roadmap. [S1]

EXISTING BATCH COMMANDS / PRESERVED LEGACY NAMES

```
python -B -m unittest discover -s tests -v

python -B -m ultron doctor
```

The public name is SRAOSHA. Do not blindly rename imports, paths, receipts or package identifiers: record a migration plan, preserve compatibility and revalidate the renamed result. The quoted commands are from the supplied contract, not commands executed for this PDF.

PROPOSED RECEIPT SCHEMA

RECEIPT GROUP	REQUIRED RECORD
Identity	Slice ID, source snapshot, donor versions, environment, tool/model route and timestamps.
Authority	Owner decision, exact plan revision, policy version, scope, budgets and revocation state.
Proof	Acceptance assertions, regression/doctor logs, artifact hashes and observed limitations.
Outcome	Complete / blocked / failed / cancelled, recovery result, residual risk and next action.

Integrity distinction: use authenticated records or signed manifests where appropriate, protect keys and anchor history outside the worker's control. An unkeyed hash chain alone is not an immutable trust system.

Historical constraints and commands: [S1]. Receipt schema and integrity enhancements: proposed.

Resolve the decisions that unlock work.

These are explicit decision points, not excuses to keep producing scaffolding.

ACCOUNTABLE ROLE	DECISION TO RECORD
Owner / project lead	Confirm the present write boundary, pilot scope, P.S.T. permissions and any expansion of automation authority.
Core engineering	Select the task/policy contracts, routing strategy, concurrency budgets and migration/versioning approach.
Security engineering	Choose the protected-storage design, key recovery model, isolated runtime and DJINN threat model.
Domain stewardship	Assign corpus maintenance, qualified legal review and acceptance responsibility for domain-specific claims.
Integration owner	Select the first real system, test account, least-privilege scopes and rollback/compensation contract.
Release / quality	Choose distribution, signing, support targets, manual accessibility checks and independent acceptance review.

TOP RISKS TO WATCH

Scope dilution: breadth grows faster than reliable verticals. **False readiness:** catalog entries and tests are mistaken for live capability. **Resource pressure:** model and build work compete with other projects. **Authority drift:** convenience bypasses the broker. **Recovery debt:** new stores or adapters escape the restore plan.

> **Operating response:** keep one visible dependency queue, limit concurrent expensive work, retire abandoned scaffolding, and stop promotion when critical evidence is missing. Several roles may be held by one person, but the decisions must still be recorded.

Proposed governance. The historical handoff explicitly warns against expensive parallel work and unsupported completion claims. [S1]

Guard. Challenge. Harden. Repeat.

A Mainely Code project. Security-led, all-domain, and designed as one intelligence.

SRAOSHA

SECURITY-HARDENING GUARDIAN

Understand the environment. Reduce exposure. Preserve recoverability. Make protection an ongoing, evidence-backed workflow—not a one-time checklist.

DJINN-SRAOSHA

ADVERSARIAL TESTING COUNTERPART

Challenge the controls. Investigate weaknesses. Coordinate selected Kali/WSL exercises. Return findings the guardian can harden against and independently retest.

Dual-use by nature; operator-directed in practice. DJINN can be used constructively or misused. Its branding is not a moral guarantee. Named scope, impact visibility, revocation and durable evidence stay in the engineering contract. [S2]

LOOP	SRAOSHA / BLUE-TEAM RESPONSIBILITY	DJINN / ADVERSARIAL CONTRIBUTION
B01 Baseline	Version approved asset, identity, policy and service state.	Use the same snapshot to define the test scope.
B02 Harden	Preview reversible configuration and privilege changes.	Challenge the specific control after approval.
B03 Watch	Detect drift; explain severity, source and confidence.	Run bounded regression checks with stated budgets.
B04 Recover	Protect active data, secrets and restore paths.	Exercise recovery assumptions inside the lab.
B05 Respond	Triage evidence; propose containment and repair.	Corroborate findings and preserve raw artifacts.
B06 Learn	Retain lessons across the full AI feature set.	Feed verified outcomes into shared context and memory.

BASELINE / CHALLENGE / EVIDENCE / HARDEN / RETEST

Automate repeatable missions with budgets, scheduling, pause/resume and drift-triggered checks—not blind command loops.

Proposed product contract, not live capability evidence. All twelve feature families and the frontier program remain in scope. [S1–S2]

Mainely Code. In every mode.

Permanent product credit. Visible operating state. Approved identities.

REQUIRED ATTRIBUTION / EXACT CASE AND EM DASH

Proudly Built with Mainely Code Buildroom—ADVANCED EDITION

SRAOSHA MAINELY CODE / CONCEPT ONLY

Mission Security Evidence Knowledge Build

Baseline: not captured / Hardening: preview only / Evidence: none

GUARDIAN | OBSERVE ONLY | DATA: LOCKED | NO JOB RUNNING

Proudly Built with Mainely Code Buildroom—ADVANCED EDITION

DJINN-SRAOSHA MAINELY CODE / CONCEPT ONLY

Mission Security Evidence Knowledge Build

Scope: LOCAL-LAB-A / Approval: pending / Evidence: none

ADVERSARIAL | PLAN ONLY | WSL: NOT VERIFIED | STOP: IDLE

Proudly Built with Mainely Code Buildroom—ADVANCED EDITION

IMPLEMENTATION & ACCEPTANCE REQUIREMENTS

Persistent placement. Show the full credit in UI footers and persistent status bars in both modes and P.S.T. A combined footer/status bar may share one credit row. Keep job state, scope and stop controls separately visible.

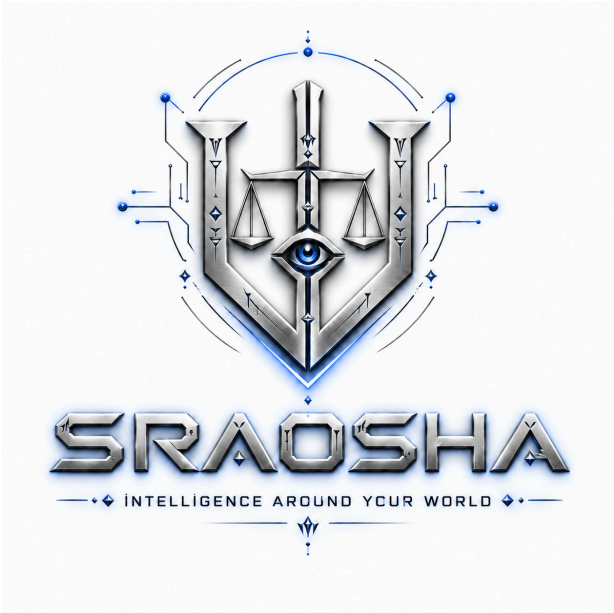
Responsive & accessible. Use a shared branding component. At narrow widths or high scaling, wrap into a dedicated credit row; never truncate, hide, animate or use color alone for operating state.

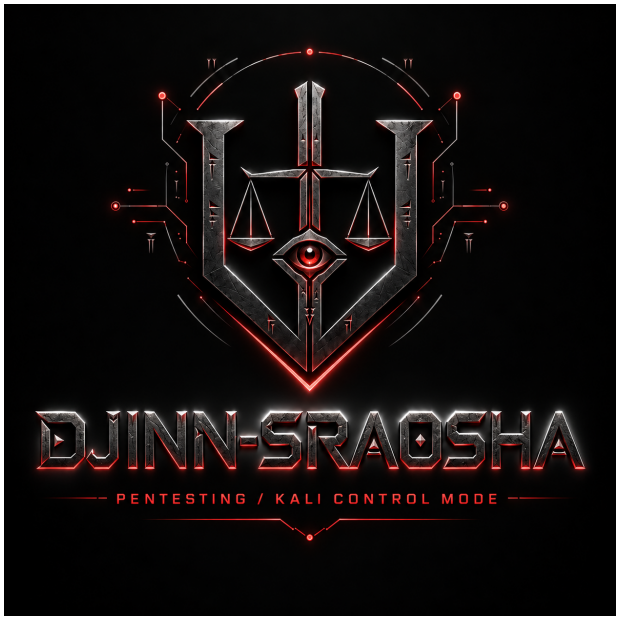
Release evidence. Verify exact text, blue/red theme fidelity, keyboard and screen-reader behavior, and readable layouts at supported scaling. Capture both modes and the P.S.T. edition independently.

Illustrative interface requirements, not screenshots of an implemented app. No native UI or repository was changed by this PDF revision. [S2]

One emblem. Complementary expressions.

A Mainely Code product family. Preserve the approved masters and mode identities.





ELEMENT	APPROVED DIRECTION
SRAOSHA	White field; metallic silver; blue guardian highlights and dark beveled edges; restrained blue light.
DJINN-SRAOSHA	Black field; dark metal; fine white edge/backlight for legibility; warm-red accents rather than a harsh blood-red wash.
Names & taglines	SRAOSHA / INTELLIGENCE AROUND YOUR WORLD DJINN-SRAOSHA / PENTESTING / KALI CONTROL MODE
Shape & symbolism	Preserve the recognizable shield, scales and eye. Blue guardian / red assessment; balance through finish and color, not a new silhouette or literal yin-yang mark.
Cuneiform treatment	Use the approved wedge-inspired details and restrained ornamental accents. These are decorative, not claimed translations or authenticated sacred inscriptions.

P.S.T.: retain its distinct original icon geometry; its updated master artwork still needs a separate approval. Do not substitute the owner's emblem and call the edition finished.

Brand decisions and approval: conversation [S2]. Approved masters: [S3]. No trademark or design clearance is established by this document.

What this blueprint is grounded in.

A traceable distinction between project evidence, owner direction and proposed engineering.

[S1] SUPPLIED BUILD HANDOFF

Pasted markdown.md — the attached conversation/work-tree review. It reports prior batches, exact-review revocation, the later 1,341-test result, an unreachable model endpoint in the latest direct check, the 9-passing/11-blocked release score, and remaining runtime, storage, domain, execution, integration, packaging and parity gaps.

The file is a pasted narrative, not the live repository or the underlying verification artifacts. Its relative time labels are not converted into a new audit date. The later reviewed snapshot is used for baseline statements; earlier batch counts are not combined into a new total.

[S2] OWNER DIRECTION AND BRAND APPROVAL

This conversation establishes SRAOSHA as a Mainely Code, all-domain AI with a security-hardening guardian lead identity, and DJINN-SRAOSHA as its dual-use adversarial/pentesting counterpart. The broader feature set, P.S.T. edition and “suit of armor around my world” mission remain. Blue/silver-on-white and red/dark branding are approved. [S2]

The tagline, WSL-on-Windows promotional callout and exact Buildroom—ADVANCED EDITION footer/status-bar credit are owner-supplied direction. Other project IP is not co-branded. Product intent is not proof of implementation.

[S3] APPROVED ARTWORK

SRAOSHA_Guardian_Blue_Original_Emblem.png
djinn_sraosha_cyber_sentinel_crest.png

The approved image objects are retained from the blue/red revision without redesigning their emblems. Earlier rejected artwork is not substituted.

Authorship boundary: architecture, feature families, delivery slices, interface concepts and acceptance contracts are blueprint proposals. The Mainely Code credit is a required brand treatment, not a new receipt proving a Buildroom or application run.

Revision 1.2 / 07 September 2026. No repository, model service, native UI, account, Kali runtime or release package was executed or inspected during this PDF update. The supplied handoff remains the historical baseline.

Engineering references—not endorsements.

Primary guidance consulted 07 September 2026. Source titles below are clickable.

W1 **OWASP — LLM01:2025 Prompt Injection**

Threat categories for untrusted documents, web content and tool output.

W2 **OWASP — LLM06:2025 Excessive Agency**

Rationale for separating model proposals from scoped permissions and action approval.

W3 **NIST — SP 800-218, SSDF Version 1.1**

Secure-development practices used as guidance for build, release and supplier evidence.

W4 **NIST — AI Risk Management Framework**

Risk and trustworthiness considerations for evaluation and product governance.

W5 **Kali Linux — Kali WSL**

Official WSL deployment context; not proof of a configured SRAOSHA environment.

W6 **Microsoft — Advanced settings configuration in WSL**

Mount and interoperability controls relevant to reviewing a proposed WSL profile.

W7 **Microsoft — Accessing network applications with WSL**

Networking behavior that must be considered when defining the target boundary.

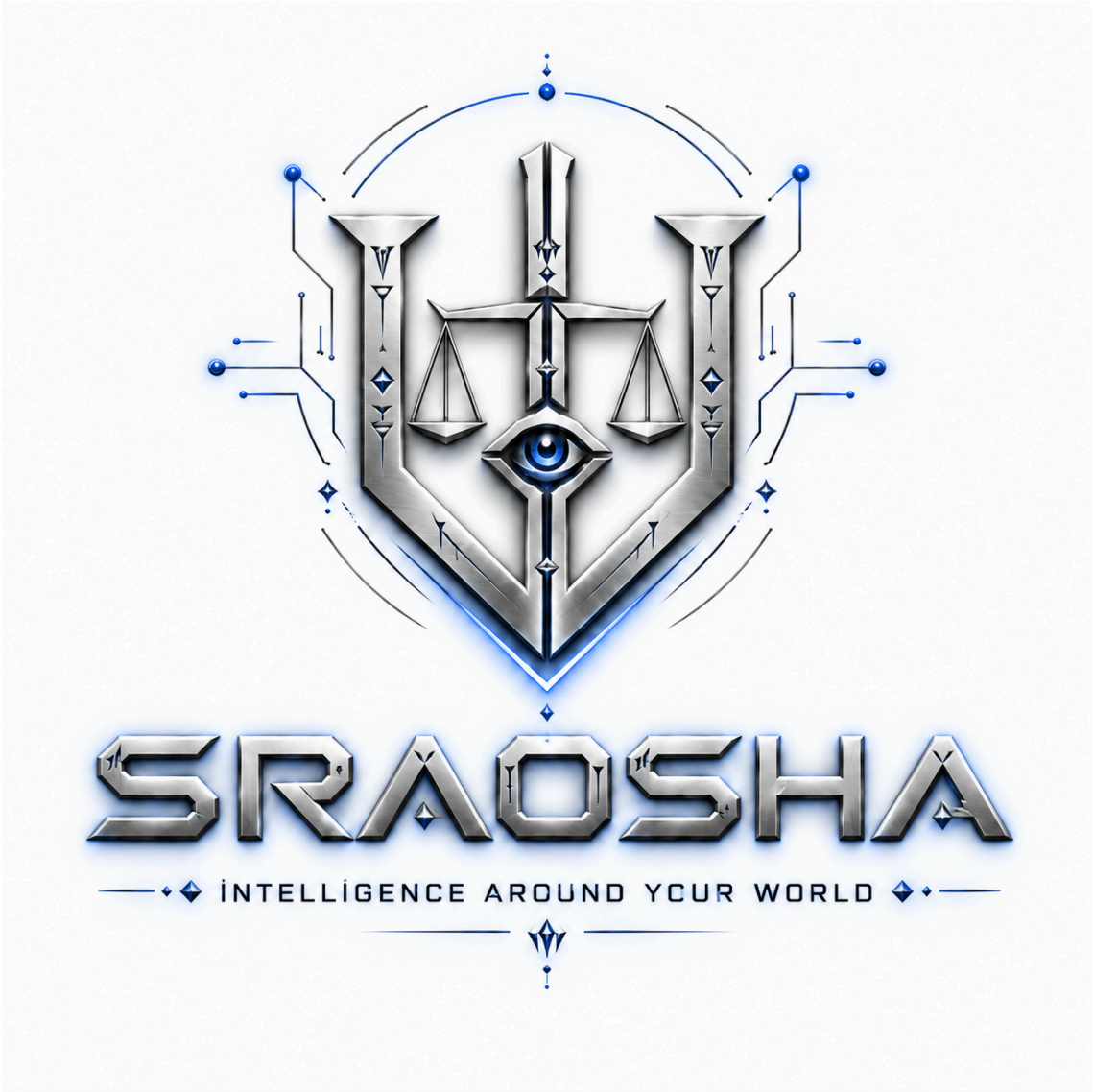
W8 **Kali Linux — Win-KeX**

The Kali desktop experience under WSL 2; separate from application authorization.

W9 **Microsoft — Windows Sandbox**

Candidate Windows code-test environment; default networking requires explicit review.

These references inform design choices. They do not certify this product, validate its current implementation, establish legal authorization for an engagement or clear the product names for trademark use.



Build the intelligence.
Earn the authority.
Keep the human in command.

A suit of armor around your world—
made real through working capabilities and evidence.