



RIDGE SECURITY

RidgeGen

Agentic AI to find, prove, and fix risk with bounded,
evidence-backed validation

Architecture, feature highlights, and differentiation

TECHNICAL WHITE PAPER

01 EXECUTIVE SUMMARY

RidgeGen is Ridge Security's agentic AI platform for bounded, evidence-backed security validation. It is designed for organizations that need the judgment and depth of human security teams, the repeatability of automation, and the governance required to approve agentic activity against real environments.

RidgeGen is Ridge Security's agentic AI platform for bounded, evidence-backed security validation. It is designed for organizations that need the judgment and depth of human security teams, the repeatability of automation, and the governance required to approve agentic activity against real environments.

Verizon's 2026 DBIR reports that vulnerability exploitation has become the leading initial access vector in breaches, accounting for 31 percent of breaches in the reporting dataset. CISA's Known Exploited Vulnerabilities Catalog reinforces the same operational lesson: teams cannot treat every scanner result equally; they need prioritization based on exploitability, exposure, and evidence. Meanwhile, OWASP and MITRE guidance on LLM and agentic systems makes clear that AI agents introduce their own risk surface, including prompt injection, excessive agency, tool misuse, identity abuse, data leakage, and cascading failures.

RidgeGen addresses both sides of that problem. It uses agentic reasoning to explore multi-step attack paths, business-logic weaknesses, and non-CVE exposure that conventional scanners often miss. It then constrains that reasoning with explicit target authority, policy-controlled tools, guardrails outside model discretion, deterministic run records, and verifier-gated findings.

The result is not a free-form chatbot pointed at production. It is a security validation system that turns operator intent into a frozen launch package, executes bounded work across an explicitly authorized set of FQDNs and IP subnets, and promotes only evidence-supported claims to customer-facing findings.

The latest implementation strengthens four customer-visible promises. First, test authorization is regulated through security zones that define which behaviors are allowed in production, supervised validation, and authorized non-production testing. Second, RidgeGen's governed knowledge loop turns domain expertise and human creativity into shared enterprise memory: the Ridge Knowledge store combines the Knowledge Base and Contextual Findings so trusted insight compounds across runs, while related-job answers remain bounded to product-supplied context for a target cluster. Third, SafeBox keeps sensitive credentials structurally outside the agent's readable context while still allowing authenticated testing. Fourth, remediation has been enhanced so verified findings produce click-away implementation guidance: a concrete fix path, verification step, and handoff-ready action plan for the team that owns the affected system.

RidgeGen is designed to complement RidgeBot, Ridge Security's flagship AI-powered platform for automated and continuous security validation. While RidgeBot provides broad, ongoing testing across the entire enterprise attack surface to see if vulnerabilities are actually exploitable, RidgeGen shifts the focus toward deeper, target-bound investigations.

Optimized for scenarios that demand red-team-grade reasoning, concrete proof, rich remediation context, and strict auditable control, RidgeGen provides deeper target-specific analysis than breadth-oriented automation alone.

Together, they enable a complete transition from periodic, point-in-time assessments to continuous exposure validation grounded in real-world exploitability.

02 MARKET CONTEXT AND DESIGN REQUIREMENT

Security programs face three pressures at once.

1 **Exposure is expanding faster than teams can manually validate it**

Cloud services, SaaS integrations, APIs, identity systems, remote-access infrastructure, IoT and OT systems, and AI-enabled applications create more ways for adversaries to reach business-critical assets. This makes pure enumeration insufficient. A list of possible weaknesses does not tell a CISO which issue can produce business impact.

2 **Attacker behavior has become faster and more exploitation-driven**

The 2026 DBIR highlights this shift, reporting that vulnerability exploitation has overtaken credential abuse as the top initial access vector in the breach dataset. Because attackers increasingly exploit exposed systems directly, prioritization frameworks such as the CISA KEV catalog have become vital: remediation urgency should be driven by exploitability, exposure, and evidence of active weaponization. This makes continuous, real-world exploitability validation more critical than ever.

3 **AI agents are powerful but risky when they can plan, call tools, read content, and act across systems**

NIST's AI RMF and Generative AI Profile emphasize governed risk management across the AI lifecycle. OWASP's LLM and Agentic AI guidance highlights direct and indirect prompt injection, excessive agency, tool abuse, identity and privilege abuse, data exposure, and failure cascades. MITRE ATLAS catalogs adversarial techniques against AI-enabled systems, while MITRE

DESIGN REQUIREMENT

These pressures define the RidgeGen design requirement: deliver deeper autonomous security validation without transferring unrestricted agency to the model. The platform must preserve human authorization, maintain hard target boundaries, constrain tool use, record evidence, and make findings explainable after the run.

03 PRODUCT POSITION

RidgeGen is best understood as an autonomous security validation workbench with a strict separation between operator-facing product controls and the runtime that performs the assessment.

RidgeGen Control Plane

The product layer guides setup, collects intent, validates target inputs, records credentials provenance, enforces organization defaults, supports target review and promotion, freezes launch authority, tracks run history, and presents findings, reports, and post-run answers. Operators describe the mission in product language rather than editing raw scenario files.

RidgeGen Engine

The runtime layer executes bounded work. It plans from trusted run state, dispatches task-scoped agents, brokers tools through controlled interfaces, applies guardrails at safety boundaries, records artifacts, verifies candidate findings, and writes trusted results back to the evidence and knowledge layer.

This separation is important for enterprise approval. Commercial authority, operator intent, runtime decisions, target evidence, and final explanations remain distinct and inspectable. The model can reason, but it does not define its own authority.

3.1 RidgeGen and RidgeBot

RidgeBot is Ridge Security's security validation platform for breadth, scale, and continuous validation across enterprise environments. RidgeGen adds a deeper agentic layer for cases where the organization wants autonomous reasoning across an explicitly authorized set of FQDNs and IP subnets, including attack-chain exploration, business-logic testing, non-CVE risk discovery, and evidence-backed remediation.

RidgeBot - Breadth

Helps teams reduce noise and continuously validate exposure across the attack surface.

RidgeGen - Depth

Helps teams investigate a selected target set in depth with bounded agentic reasoning and proof.

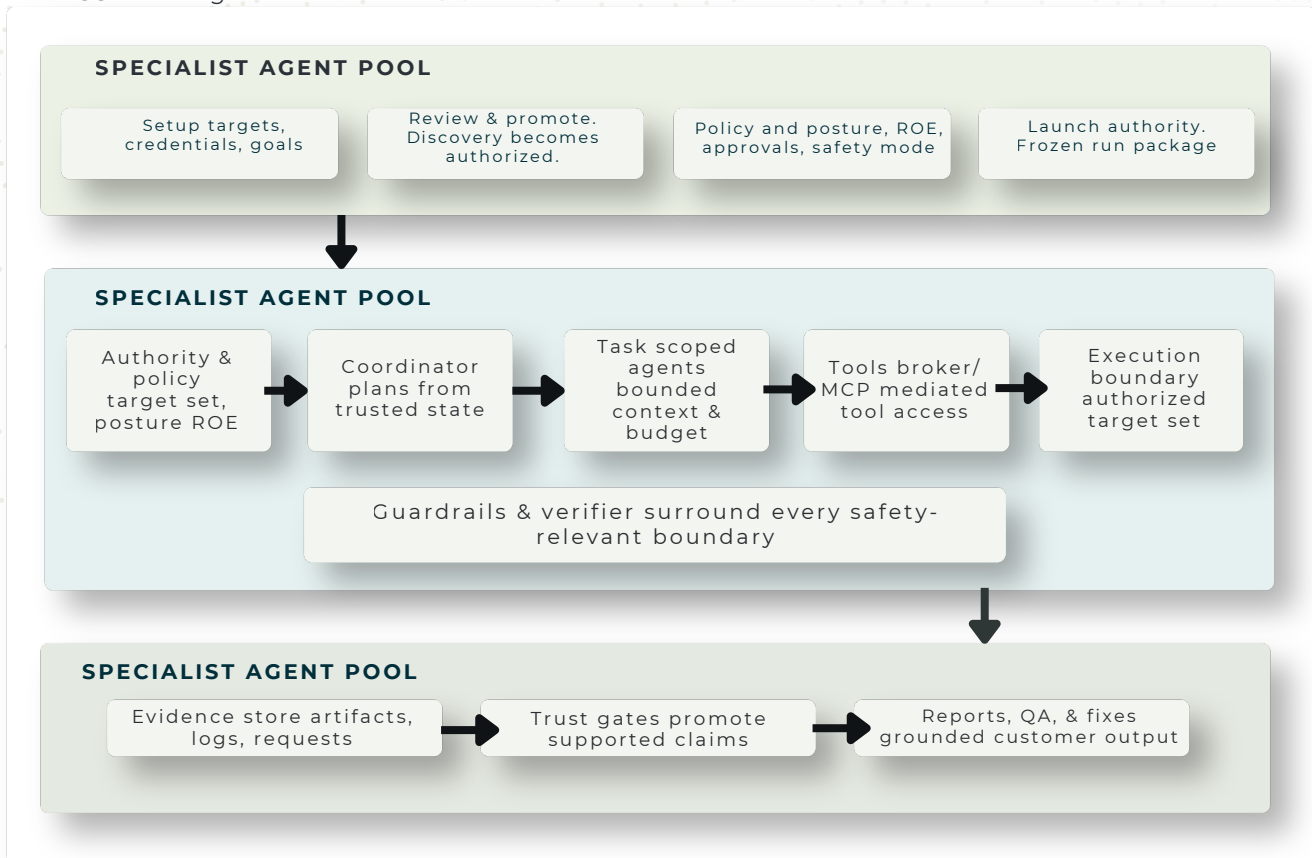
Used together, they let security teams move beyond scanner triage toward verified, auditable validation of exploitable risk.

04 REFERENCE ARCHITECTURE

RidgeGen uses a three-layer architecture: an operator control plane that prepares authority, a policy-bounded engine that executes work, and an evidence/read layer that makes results reviewable. Authority flows into the run from the top. Evidence, verification decisions, findings, and explanations flow back out.

Figure 1 summarizes the separation between operator authority, policy-bounded execution, evidence collection, verification, and customer-facing readout.

FIGURE 1 RidgeGen Reference Architecture



4.1 Operator control plane

The control plane translates intent into a safe, reviewable launch package. It gathers the target input, credentials and secret provenance, allowed posture, model alias, organization policy, run objective, and rules of engagement. It also separates discovery from authorization: a discovered host, domain, route, or service is an observation until an operator promotes it into the scope of an authorized run.

The control plane is also where buyer trust is created. It provides the artifacts a CISO or risk owner needs before approval: who authorized the work, which target set is allowed, which credentials may be used, which posture applies, which actions require approval, and how the resulting evidence will be retained or masked.

4.2 RidgeGen Engine

The engine is a coordinated runtime, not a hard-coded collection of named specialists. A coordinator plans from trusted run state and instantiates task-scoped agents with only the context, budget, target component, tools, and stop conditions they need. This keeps the benefits of specialization while avoiding a brittle architecture that depends on fixed agent identities.

Task-scoped agents can focus on discovery, authentication flows, application logic, API behavior, identity and authorization controls, data exposure, exploit validation, remediation analysis, or other mission-specific work. Tool access is mediated through a tool broker and execution boundary. The engine records what was attempted, what was observed, what artifacts were produced, which guardrail decisions were made, and why a candidate claim should or should not be trusted.

4.3 Evidence, verification, and read layer

RidgeGen treats evidence as a first-class product object. Requests, responses, screenshots, logs, payload metadata, target observations, policy decisions, and provenance are preserved so findings can be reviewed after the run. The verifier checks whether artifacts support the affected target, severity, exploitability, business impact, and remediation story. Customer-facing output is grounded in promoted findings and run digests, not in vague model memory.

METHODOLOGY

This approach aligns with established security testing methodology. NIST SP 800-115, OWASP WSTG, and PTES all emphasize planning, scoped execution, evidence collection, analysis, and reporting. RidgeGen modernizes that workflow for autonomous execution while preserving the parts that make an assessment defensible.

05 RUN LIFECYCLE: SEED TO EVIDENCE

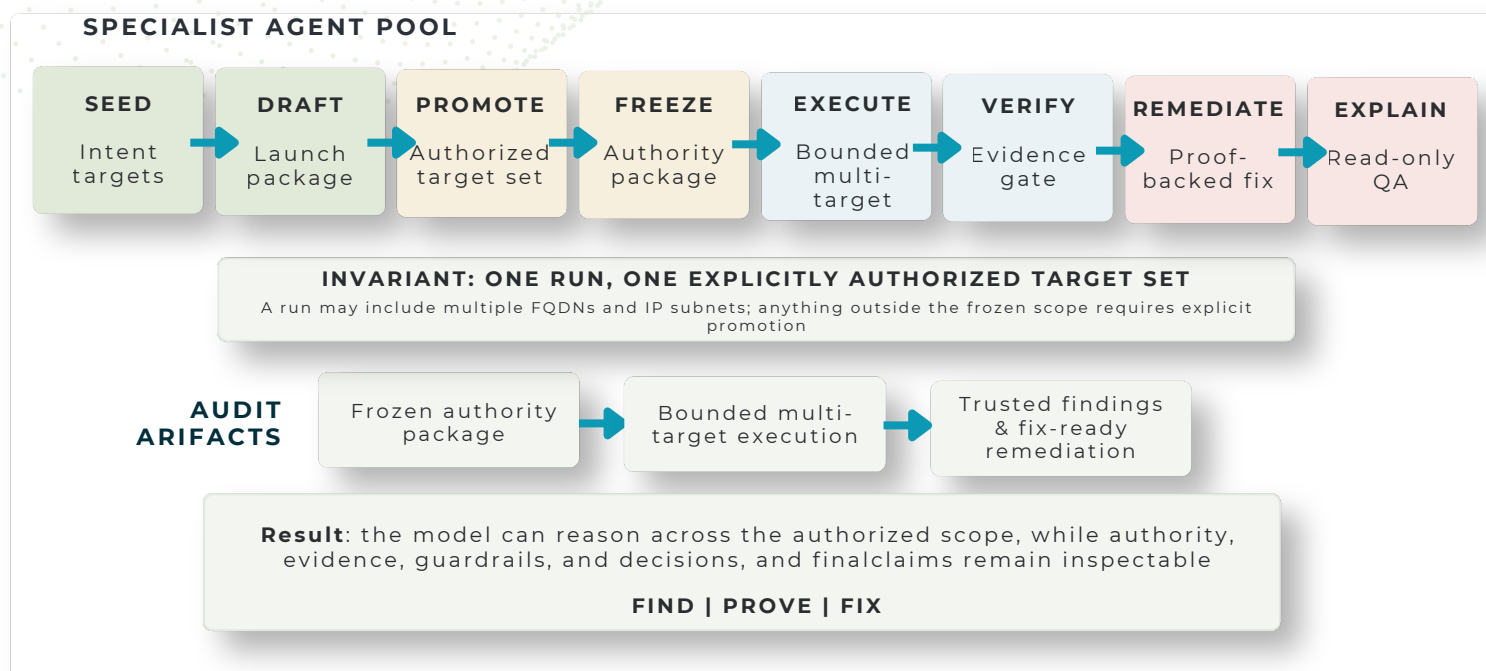
RidgeGen deliberately turns an open-ended conversation into a sequence of authority-bearing steps. The workflow is not "chat, then agent." It is seed, draft, promote, freeze, execute, verify, remediate, and explain.

Each transition is a trust boundary. Seed and Draft collect operator intent without granting execution authority. Promote and Freeze convert candidate scope into explicit authority. Execute and Verify separate bounded action from claim promotion. Remediate and Explain turn proof into usable output without allowing post-run answers to invent facts outside the evidence record.

Operationally, the lifecycle creates two audit artifacts: a frozen launch package before execution and an evidence-backed run digest after execution. These artifacts give approvers a concrete record of scope, policy, posture, credential provenance, tool activity, guardrail decisions, verifier rationale, and customer-facing claims.

Figure 2 shows how operator intent becomes a bounded multi-target run: scope is drafted, promoted, frozen, executed, verified, remediated, and explained from the resulting evidence record.

FIGURE 2 *Seed-to-evidence Lifecycle*



5.1 The eight-step lifecycle

1. Seed

The operator supplies a URL, FQDN, IP, IP subnet, credentials, constraints, and goals. These are inputs to configuration and discovery, not yet permission to test every related system.

2. Draft

The assistant produces a structured launch configuration: target-discovery settings, model selection, posture, policy, credentials provenance, mission focus, and preliminary constraints. Invalid, ambiguous, or under-authorized inputs can be rejected before execution.

3. Promote

Discovery may identify candidate FQDNs, IPs, routes, services, or applications. Candidates remain observations until the operator promotes the intended FQDNs and IP subnets into the run's authorized scope.

4. Freeze

The system freezes the launch package so the engine receives concrete authority: authorized target set, posture, policy, model alias, credentials provenance, rules of engagement, and run settings. This prevents the runtime from silently widening the mission.

5. Execute

The engine performs bounded work across the explicitly authorized target set. Guardrails apply at safety boundaries, tooling is mediated, artifacts are recorded, and any FQDN or IP outside that set is logged rather than automatically tested.

6. Verify

Candidate findings, coverage claims, and impact statements are promoted only when evidence and provenance support them.

7. Remediate

Verified findings include proof and remediation context so security teams can fix without replaying the entire assessment by hand.

8. Explain

Post-run QA is read-only. It answers from artifacts, trusted findings, run digests, timelines, policy decisions, and related-job context supplied by the product.

TWO AUDIT ARTIFACTS

Operationally, the lifecycle creates two audit artifacts: a frozen launch package before execution and an evidence-backed run digest after execution. These give approvers a concrete record of scope, policy, posture, credential provenance, tool activity, guardrail decisions, verifier rationale, and customer-facing claims.

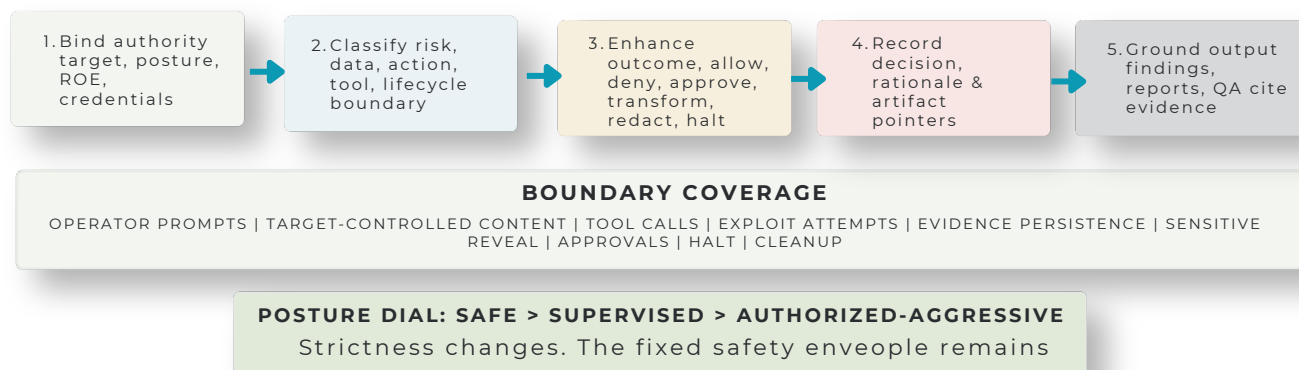
06 GUARDRAILS AS THE SAFETY CONTRACT

Autonomous security validation needs more than prompt instructions. Prompting can shape behavior, but customer trust depends on controls outside model discretion. RidgeGen guardrails are the safety contract that binds authority, classifies risk, enforces an outcome, and records the decision.

Those boundaries keep authority explicit: targets, posture, approvers, credential use, reveal rights, and launch authority come from configuration and rules of engagement, not from the agent's interpretation of what it observes.

Figure 3 illustrates how RidgeGen turns each safety-relevant boundary into an enforceable decision path, from authority binding through risk classification, enforcement, audit recording, and evidence-grounded output.

FIGURE 3 *Guardrail Decision Flow - Safety controls outside model discretion govern actions, data and output*



Guardrails apply wherever safety-relevant behavior, data, authority, or lifecycle state crosses a boundary between the operator, the agent, the tools, the target, and the evidence layer. Each decision produces a policy-defined outcome, for example, permit, block, require approval, or transform the action into a safer form, and the outcome is recorded for audit.

In practice, this gives customers one safety model across the surfaces they actually use: the operator assistant, the autonomous runtime, the evidence and reporting layer, and the approval, reveal, halt, and cleanup controls. The same decision pattern applies whether the risk starts as a prompt, a tool call, a target-controlled response, a saved artifact, or a human approval request.

6.1 Posture model

RidgeGen lets customers choose strictness without letting them disable the fixed safety envelope. This is implemented as three security zones that regulate test authorization and permitted behavior.

Safe - production default

The run is limited to non-intrusive validation and non-destructive evidence. Write, modify, delete, disruptive, and off-target actions are denied or transformed into evidenced-risk findings. The agent can still use its own workspace for scratch files and tooling, but it cannot mutate customer target state.

Supervised

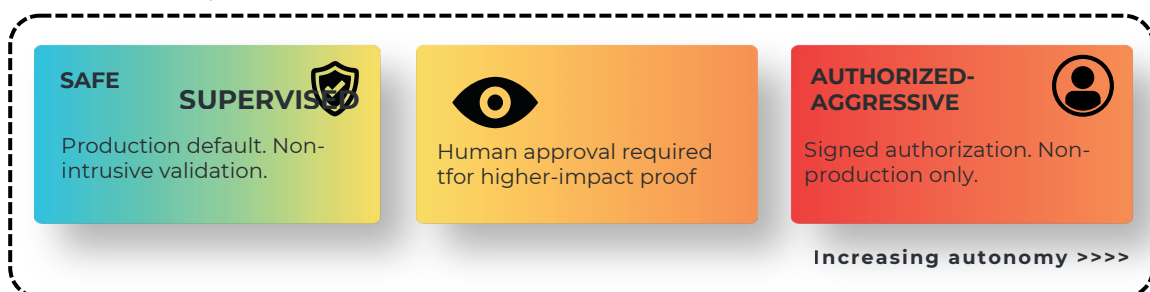
Used when the customer wants stronger proof without allowing uncontrolled impact. The platform may permit creation of new test data to prove a finding, but modification, deletion, disruption of existing target state, and sensitive reveal still require explicit control. Risky actions are routed through approval gates and cleanup obligations are recorded.

Authorized-Aggressive

Reserved for signed rules of engagement in lab or non-production environments. The zone can permit stronger actions against the authorized target, including write, modify, delete, or disruption where the customer has accepted the risk. Off-target confinement still holds, and the fixed safety envelope still records authority, approvals, evidence handling, and cleanup.

Figure 4 shows the three security zones as increasing autonomy inside a fixed safety envelope. The key point is that customers can choose strictness for proof, but cannot disable target confinement, auditability, or guardrail enforcement.

FIGURE 4 Security Zones



Guardrails regulate authorization behaviour; autonomy can increase inside the fixed safety envelope.



6.2 Evidence without unsafe exposure

A blocked action should not mean a missing finding. RidgeGen can report evidenced risk by preserving the minimum convincing proof: schema, object names, record counts, versions, masked samples, metadata, response headers, control-plane decisions, and safe indicators before sensitive content. Credentials, tokens, cookies, API keys, and PII should be masked by default. Reveal should be exceptional, per-value, authorized, reasoned, and audited.

6.3 Your domain knowledge becomes shared enterprise memory

RidgeGen's setup interface is open to domain knowledge and human creativity. Rather than translating keywords into wizard settings, it is a window for an operator to teach the platform about their specific environment — and that human knowledge is documented into shared enterprise memory. Inside the RidgeGen Engine, the Ridge Knowledge store (Knowledge Base plus Contextual Findings) holds the trusted state that planning, reporting, and follow-up QA rely on, so insight compounds across runs instead of evaporating when a session ends. Cross-run answers draw on related-job context the product supplies for a target cluster — bounded, defensible memory rather than claims from an unlimited and unverifiable history.

6.4 SafeBox credential handling

SafeBox is the credential-safety mechanism behind RidgeGen's default restraint promise. Some attack paths can only be proven with real customer-provided credentials, but those credentials should not become model-readable text, report content, logs, telemetry, or durable evidence. SafeBox gives the agent a usable stand-in while keeping the real value out of the agent's prompt, context, memory, and narration path.

SafeBox closes the places credentials most often leak in AI-driven testing: while at rest, while in use during an assessment, and when results are captured for evidence and reporting. Stored credentials remain protected on the appliance, and evidence describes structure, scale, and masked samples rather than live secrets.

The important technical distinction is that SafeBox is not merely a masking instruction. The real credential is applied at the tool or action boundary outside the model's view. If target-controlled content attempts prompt injection, such as asking the agent to print credentials in context, the agent has no readable credential to reveal. Tool results that could otherwise echo credential material are masked before reasoning, storage, or display.

SafeBox also supports customer governance. Credentials remain tied to the appliance and the specific run, so a copied store or stolen backup is not equivalent to credential theft. Use can be bounded by operator-defined limits, and any authorized reveal is exceptional, per-value, reasoned, time-bounded, and audited. For customers, this means authenticated testing can happen without placing live secrets into vendor-side logs, model memory, evidence artifacts, or final reports.

07 FINDINGS, REMEDIATION, & EXPLAINABILITY

RidgeGen's trust model answers four questions.

AUTHORITY TRUST

was this allowed? The target, posture, policy, credentials provenance, launch actor, approvals, and rules of engagement must come from explicit configuration.

BEHAVIOR TRUST

Tool use, target access, sensitive reveal, and lifecycle state transitions must pass guardrails that are enforced outside model discretion.

FUNDING TRUST

A candidate finding becomes trusted only when artifacts support the affected target, exploitability, severity, impact, and remediation.

EXPLANATION TRUST

Is the answer grounded? Reports and post-run QA answer from run digests, trusted findings, timelines, evidence, artifacts, and policy decisions.

7.1 Click-away remediation implementation

RidgeGen's remediation layer is designed for the team that has to ship the fix. A verified finding should not end with a generic recommendation. It should name the affected host, endpoint, path, parameter, account, or configuration item; explain the change to make; identify immediate containment; and provide a verification step that confirms the original issue no longer works.

The remediation plan is generated from evidence captured during the run, not from a generic vulnerability template alone. For example, when a client-side JavaScript bundle exposes recoverable credentials, RidgeGen can tell the customer to remove the secret from the shipped asset, rebuild the bundle, rotate the affected account, invalidate sessions or tokens created with the leaked secret, move the secret to server-side configuration or a managed secret store, search adjacent assets and repository history for similar exposure, verify the fix by confirming the rebuilt asset no longer contains the secret and the old credential no longer authenticates, and add build-time detection for credential-like patterns so similar exposures are caught before release.

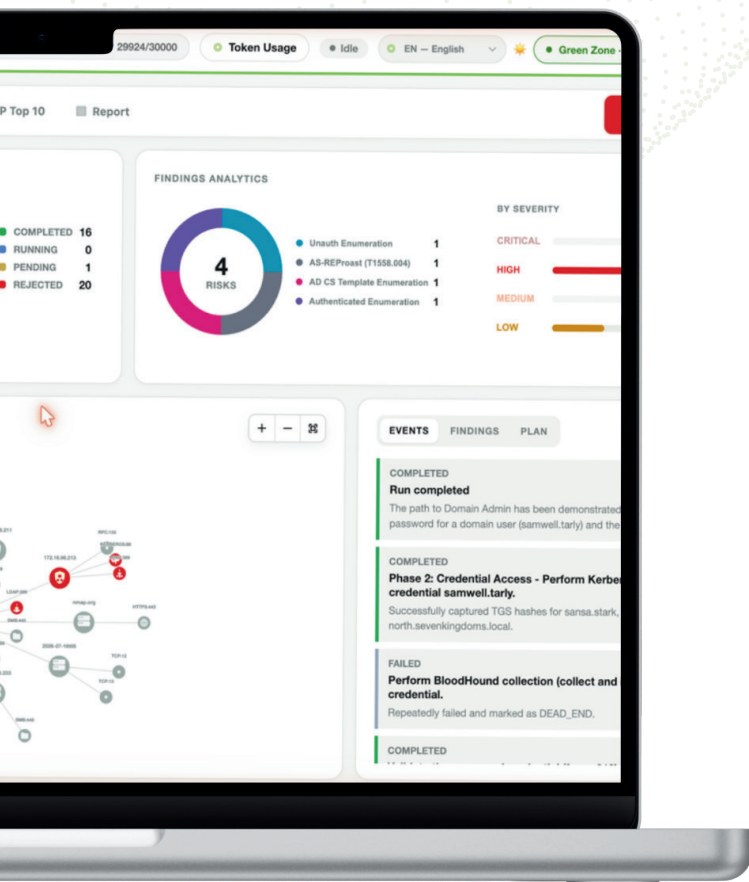
This is what Ridge Security means by click-away implementation: the finding already contains enough context for engineering, AppSec, or platform teams to start the fix without reconstructing the assessment. The output is specific, executable, verifiable, forward-looking, and consistent across reports and follow-up reviews.

08 DEPLOYMENT MODEL AND EXTENSIBILITY

RidgeGen is designed for enterprise environments where targets, credentials, and evidence may need to remain inside the customer boundary. An on-premises or customer-controlled deployment model supports sensitive assessment workflows, reduces data-governance friction, and gives buyers a clearer path to approve autonomous testing.

The architecture is also model-agnostic. Models can change as the market changes, while target authority, tool mediation, evidence storage, guardrails, and verification remain product controls. That separation reduces lock-in and makes model performance an implementation choice rather than the root of the trust model.

Extensibility should follow the same pattern. Bring-your-own-agent capabilities are valuable only if external or customer-authored agents inherit the platform's authority model, tool boundary, evidence requirements, and verifier gates. A custom agent should extend capability, not bypass governance.



09 DIFFERENTIATION

RidgeGen is differentiated by how it combines depth, autonomy, and governance.

Against vulnerability scanners, RidgeGen is deeper.

Scanners are useful for coverage and discovery, but they often stop at signals: versions, banners, patterns, and signatures. RidgeGen is designed to reason through exploit paths, business logic, chained conditions, and non-CVE exposure, then support findings with evidence.

Against unconstrained general-purpose LLM agents, RidgeGen is safer and more auditable.

General-purpose agents may plan and act quickly, but they are difficult to approve when target boundaries, tool authority, data handling, and evidence standards are left primarily to prompts. RidgeGen treats those concerns as product controls.

Against traditional human-only testing, RidgeGen is more repeatable and scalable.

Human expertise remains essential for strategy, judgment, and novel attack hypotheses, but the platform can execute bounded technical work, preserve artifacts, and provide repeatable evidence-backed findings at machine speed.

Against cloud-only agentic services, RidgeGen is better aligned to sensitive enterprise use cases.

On-premises deployment, model-agnostic operation, BYOA extensibility, and evidence governance help organizations adopt agentic security validation without giving up control of targets, credentials, and proof.

10 CONCLUSION

RidgeGen's thesis is that autonomous security validation can be both powerful and governable if the product architecture separates model reasoning from authority, enforcement, evidence, and trust. The model can help plan, investigate, and explain. It should not define or expand its own target scope, grant its own tool rights, expose secrets by default, or promote claims without proof.

By combining task-scoped agentic execution with explicitly authorized multi-target scope, policy-controlled tooling, guardrails outside model discretion, durable artifacts, verifier-gated findings, and grounded read-only QA, RidgeGen gives security teams a practical path to deeper validation at machine speed. Paired with RidgeBot's broad continuous validation, Ridge Security can help customers move from noisy exposure discovery to credible proof and click-away remediation of exploitable risk.

