



GCSA SECURITY RESEARCH AND PRODUCT WHITEPAPER

HoneyTrace Whitepaper

Honeypot & Attack Intelligence Platform

Turn real attacker behavior into actionable intelligence through high-interaction telemetry, unified analysis, and secure isolation.

Version 1.1 - July 2026

Global Cyber Security Association (GCSA)

01 Executive Summary

From deploying a honeypot to operating a continuous attack observation capability.

HoneyTrace is the GCSA honeypot and attack intelligence platform. It unifies deception nodes, protocol interaction, telemetry intake, behavior analysis, and lifecycle operations. A node can expose a controlled default shell, proxy traffic to a high-interaction target, or use a large language model to generate context-aware SSH, Telnet, and HTTP responses.

Connection, authentication, command, session, file, HTTP request, and probing events enter one normalized data model. Operators can move from a risk overview into sources, attacker profiles, session timelines, port activity, and file evidence, then install, configure, update, or remove nodes from the same console.

3 Shell, proxy, and LLM modes	3 SSH, Telnet, and HTTP surfaces	1 One-click lifecycle control
Multiple deception modes Default shell, proxy, and LLM backends cover predictable simulation, real-system interaction, and adaptive responses.	Unified operations Nodes, configuration, versions, sessions, credentials, scans, files, alerts, and exports share one console.	Cloud or private GCSA SSO provisions isolated cloud workspaces, while private deployment retains a dedicated data and network boundary.

Version 1.1 provides personal cloud workspaces. Organization membership, shared assets, workspace transfer, and team billing are future capabilities.

02 Problem and Design Principles

Internet-facing deception must remain observable, attributable, and isolated over time.

A production honeypot program is more than a sensor. Enrollment, node identity, event intake, search, file handling, authorization, and upgrades often span separate systems. Shared tokens, global event tables, and public dashboards become material risks as the number of users and nodes increases.

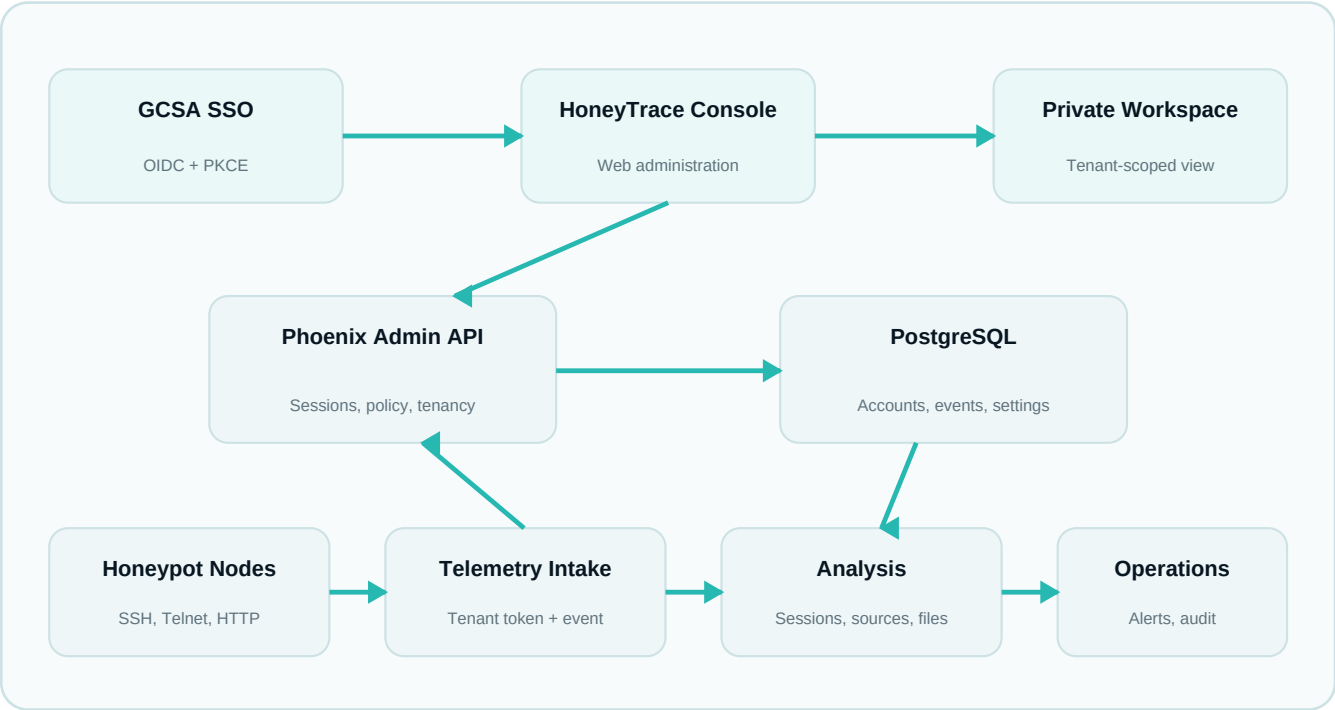
HoneyTrace keeps node onboarding straightforward while treating identity, data ownership, and operational permissions as one coherent security boundary.

Principle	Implementation	Outcome
Identity before data	Resolve the user or node identity and tenant before any read or write.	Every operation has explicit ownership context.
Private by default	Personal cloud dashboards cannot be made anonymous.	Open registration never means open customer data.
Least privilege	Members manage their assets; platform licensing and service operations require controlled roles.	Product permissions remain separate from infrastructure authority.
Backward compatible	Existing private data and legacy node credentials remain in the system space.	Upgrades preserve historical ownership and continuity.
Continuously verifiable	Two-user isolation and cross-tenant rejection are automated regression cases.	Isolation is tested behavior, not only architecture intent.

The security boundary is not one SQL predicate. It combines identity mapping, request context, token claims, database filters, file paths, and administrative policy.

03 Platform Architecture

Identity, control, telemetry, and storage planes are separated and joined by tenant context.



Layer	Core component	Responsibility
Identity	GCSA SSO / OpenID Connect	Authentication, token issuance, and optional role claims. HoneyTrace does not process user passwords.
Experience	Astro management console	Risk, sessions, honeypots, nodes, configuration, alerts, and file views.
Control	Phoenix Admin API	Session and sensor verification, tenant context, authorization, configuration, and queries.
Data	PostgreSQL + file volumes	Accounts, events, settings, status, and tenant-separated file storage.
Sensor	HoneyTrace honeypot nodes	High-interaction collection across SSH, Telnet, HTTP, commands, sessions, and files.

Official pages, the console, and the API can share one TLS domain behind a reverse proxy while the database and evidence volumes remain independently persistent.

04 Core Honeypot Capabilities

Three runtime modes balance interaction depth, operational cost, and risk boundaries.

HoneyTrace is not limited to one emulator. Each node can use a shell, proxy, or LLM backend and expose SSH, Telnet, or HTTP services as appropriate. Telemetry from every mode enters the same analysis pipeline, so operators can combine layered deception strategies without replacing the management plane.

Runtime mode	Interaction	Primary capabilities	Best fit
Default shell	Medium-interaction simulation	Unix-like files, common commands, authentication, and file transfer capture.	Rapid rollout, many nodes, and a stable controlled baseline.
Proxy	High-interaction backend	Forward SSH or Telnet to one target or a target pool while preserving session observation.	Real system behavior, complex tools, and higher-fidelity research.
LLM	Adaptive contextual response	Use history, identity, directory, and protocol context to generate shell or HTTP responses.	Long-tail commands, dynamic conversation, and unknown web paths.

Surface	Observed evidence	Shared intelligence
SSH	Connections, credentials, public keys, commands, TTY, uploads, and downloads.	Sessions, sources, credential statistics, SHA-256 evidence, and exports.
Telnet	Connections, login attempts, commands, duration, and session close.	Normalized events, command sequences, attacker profiles, and node ownership.
HTTP / HTTPS	Method, path, query, body, Host, User-Agent, cookies, headers, source, and TLS SNI.	Route rules, static responses, LLM fallback, request evidence, and source integrity.

Runtime mode is configured per node. Production design should consider network isolation, compute and model budget, acceptable interaction risk, and research objectives instead of maximizing interaction by default.

05 LLM-Powered SSH, Telnet, and HTTP

Adaptive responses remain observable, constrained, and tied to session context.

HoneyTrace Bridge translates honeypot context into model requests and can connect to an OpenAI-compatible service or Ollama. SSH and Telnet sessions include the client address, username, hostname, current directory, session identifier, and exec mode. Conversation history lets later commands continue earlier state instead of producing unrelated one-shot answers.

SSH / Telnet stage	Capability	Security and operational value
Session start	Record protocol, source, username, and session identifier.	Bind model output to the real attack source and full interaction.
Context building	Provide hostname, cwd, exec mode, and command history.	Preserve consistency across directory changes and multi-step probes.
Long-tail commands	Respond without a hand-written handler for every command.	Cover unknown tools, command combinations, and unplanned input.
Files and execution	Recognize SCP uploads and exec requests as session evidence.	Connect command activity, file movement, and follow-on execution.
Interaction guards	Handle vim, top, tail -f, and other full-screen or long-running commands safely.	Prevent unproductive output from consuming model context and resources.

The HTTP honeypot can return deterministic responses for path or regular-expression matches and send unmatched traffic to an LLM plugin. It records method, URI, a size-limited body, Host, User-Agent, cookies, headers, source IP and port, and HTTPS SNI. Trusted-proxy rules prevent untrusted peers from spoofing X-Forwarded-For.

Model API keys remain in node or server configuration. Operators should define rate limits, timeouts, input and output validation, and cost ceilings, and treat model responses as deception content rather than trusted execution results.

06 Default Shell and High-Interaction Proxy

Choose between predictable simulation and interaction with an isolated real backend.

The default shell presents a medium-interaction Debian-style Unix environment. Attacker commands are handled through a virtual file system and controlled command handlers rather than executed directly. Proxy mode forwards a connection to an isolated real or emulated backend when the research objective requires higher fidelity.

Default shell capability	Implementation	Evidence produced
Virtual file system	Customizable filesystem images and text command content simulate paths, permissions, and system files.	Visited paths, file operations, and attacker interests.
Command simulation	Common shell commands, SSH exec, and interactive TTY behavior.	Commands, ordering, exit state, timing, and complete sessions.
File capture	Record wget and curl downloads plus SFTP and SCP uploads.	Filename, origin, direction, SHA-256, and related session.
Replayable logging	Structured events and UML-compatible TTY logs.	Timeline analysis, session replay, and offline research.
Custom appearance	Adjust hostname, prompt, files, and command output.	Industry, device, or asset-specific deception profiles.

Proxy option	Operation	Deployment consideration
Simple	Forward SSH or Telnet to a fixed host and port.	Isolate the target, restrict egress, and prepare snapshots or reset procedures.
Pool	Allocate a session backend from a QEMU or similar emulated target pool.	Improves concurrent isolation and recovery but requires compute and orchestration.
Unified observation	Proxy sessions still enter HoneyTrace event and session pipelines.	Audit escape, lateral movement, and resource consumption at the boundary.

07 One-Click Deployment and Node Lifecycle

Install, configure, update, and remove honeypots from one control plane.

Onboarding stage	Platform action	Node outcome
Select method	Choose a Docker image or native install and generate a one-click command or full script.	Fit the host environment without hand-building configuration.
Download and install	Select the official amd64 or arm64 package for the host.	Reduce architecture mismatches and manual package errors.
Secure enrollment	Use a short-lived bootstrap token; generate a UUID and ECDSA public key.	Later requests use node identity rather than long-lived install credentials.
Automatic activation	Enroll tenant, sensor, ports, and runtime mode.	Start heartbeat and report events to the owning workspace.
Visible status	Show availability, version, architecture, public IP, and last heartbeat.	Identify offline nodes, version drift, and failed deployments.

Remote action	Mechanism	Operator feedback
Live configuration	Persist desired settings and push them through SSE, including mode, ports, proxy, LLM, and advanced JSON.	Desired and applied versions plus step-level progress confirm convergence.
Version update	Compare with the official release and ask the node to pull the image and restart.	Track version, update time, and failure state.
Console deletion	Verify tenant ownership and ask the node to remove its runtime instance.	Inventory, configuration, and node state converge together.
Credential rotation	Regenerate the credential for one sensor without changing others.	Limit the impact of exposure and support regular rotation.

Configuration progress covers database save, event delivery, version check, config apply, restart start, and restart success. Keep a rollback configuration and add validation for proxy and LLM changes.

08 Identity and Workspace Lifecycle

Every SSO-accepted identity can start safely when registration is open.

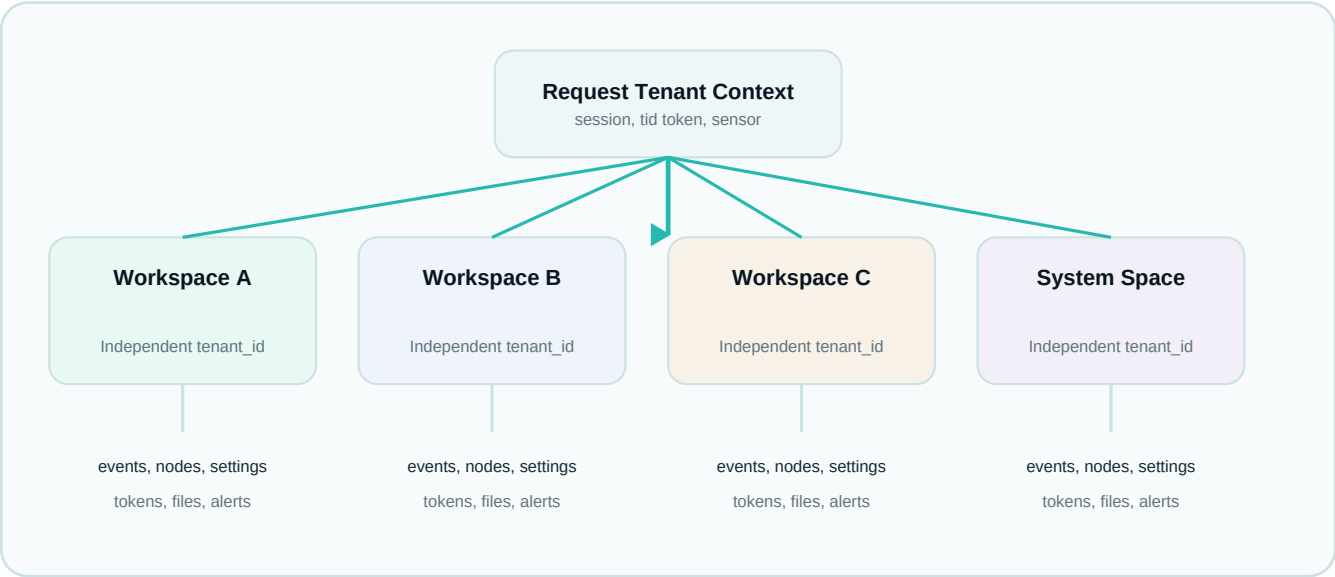
The cloud edition uses OpenID Connect Authorization Code Flow with PKCE. Login creates state, nonce, and code_verifier values. The callback validates this context, exchanges the code, verifies the ID Token, and maps the immutable issuer + sub pair to one stable workspace. Email and display name can change without moving the workspace.

Stage	System action	Security property
Start login	Redirect to GCSA SSO with openid, profile, and email scopes.	state and nonce bind context; PKCE protects an intercepted code.
Authenticate	Exchange code and validate issuer, signature, audience, nonce, and time.	The client secret remains only in the API environment.
Authorize app	Apply required role, subject, or email rules when configured.	With no allowlist, every SSO-accepted identity may use the cloud service.
Provision	Upsert by issuer + sub and assign UUID tenant_id.	Repeat logins reuse one space; different subjects remain separated.
Establish session	Store user and workspace context for subsequent requests.	State-changing browser requests require same-origin context.
Log out	Clear the local session and enter the provider end-session flow.	Shared devices retain less session residue.

Platform operations are separate from product membership. A normal cloud user cannot activate or clear platform licenses or perform infrastructure administration.

09 Tenant Isolation Model

One ownership decision follows the request through data, configuration, tokens, and files.

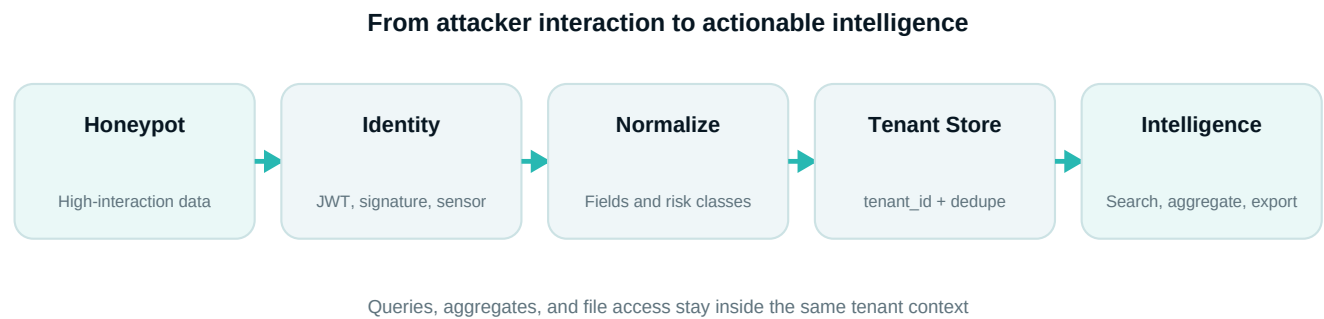


Domain	Isolation key or mechanism	Cross-tenant behavior
Accounts	saas_accounts UUID	issuer + sub is unique; profile changes do not move ownership.
Events	ingest_events.tenant_id	Counts, pages, sessions, ports, and file associations always filter by tenant.
Honeypots	registry tenantId + token tid	List, update, heartbeat, delete, and rotation only see owned sensors.
Settings	tenant:{uuid};setting_key	Control credentials, dashboard preferences, and config status stay independent.
Files	base_dir/{tenant_id}/{sha256}	Preview and download require both visible events and the tenant path.
Alerts	tenant-namespaced state	Acknowledgement in one space cannot affect another.
Private legacy	reserved system space	Historical events and old credentials retain compatible ownership.

Personal cloud workspaces are always private. Opening SSO self-registration creates an empty isolated workspace; it does not reveal existing system or customer data.

10 Telemetry and Data Pipeline

Ownership is verified before an event becomes part of the intelligence record.



A HoneyTrace node reports with a verified tenant-aware token or a registered sensor signature. The API resolves tenant context, confirms that the sensor belongs to it, normalizes the payload, and stores both the analytical fields and the original document for later investigation.

Every event carries tenant_id and import_hash. The composite uniqueness rule prevents duplicate ingestion within one workspace while allowing different workspaces to retain independent copies of equivalent telemetry.

Telemetry	Representative fields	Operational value
Connections and probes	src_ip, dst_ip, protocol, sensor, timestamp	Exposure scanning, source distribution, and node health trends.
Credential attempts	username, password, success, session	Credential dictionaries, successful paths, and related activity.
Commands and sessions	input, ttylog, duration, session	Attacker intent, tool chains, and interaction sequence.
Files and samples	sha256, filename, URL, direction	Traceability from a session to transferred content and indicators.
Node status	heartbeat, version, architecture, public IP	Availability, version posture, and remote configuration progress.

11 Analysis and Operations

Organize evidence around attack paths instead of isolated log lines.

Risk overview Aggregate events, sessions, sources, protocols, and node status across operational time ranges.	Attack sources Explore source IP, location, timing, target ports, sessions, and repeated behavior.	Attacker profiles Combine credentials, commands, interaction depth, and file activity into behavioral context.
Session timelines Reconstruct connection, authentication, commands, files, and close events by session.	Node operations Create, enroll, monitor, update, remove, rotate credentials, and track configuration state.	Files and exports Manage files by SHA-256 with tenant-scoped preview, download, and table export.

The analysis model preserves relationships among source, session, behavior, file, and sensor. An operator can move from a macro trend into one interaction and then pivot to related sources or artifacts.

Scenario	Recommended view	Operational output
New service receives scans	Port activity + attack sources	Confirm origin, target ports, cadence, and exposure controls.
Possible credential success	Credentials + session timeline	Review follow-on commands and files; extract detection indicators.
Unknown file appears	Session + file intelligence	Identify origin, SHA-256, related sessions, and subsequent activity.
Node offline or inconsistent	Honeypot inventory + config progress	Check heartbeat and desired state, then update or restart.

HoneyTrace provides observation and operational support. It does not replace an organization's incident response, evidence preservation, or risk decision process.

12 Management Console Capabilities

Bring discovery, investigation, node operations, and evidence output into one workspace.

Risk overview Summarize events, sessions, sources, protocols, trends, and node state across 12H, 24H, and 7D ranges.	Source analysis Correlate source IP, location, timing, target ports, and sessions to find repeated and concentrated activity.	Attacker profiles Combine credential lists, commands, depth, file activity, and session behavior into reusable context.
Live screen Continuously display recent attacks, trends, and sources for operations, monitoring, and rapid anomaly discovery.	Session timeline Reconstruct connection, authentication, commands, files, duration, and close events by session.	Threat and scan logs Review intrusion events, port scans, ping probes, and credentials with search and pagination.

Console module	Capabilities	Typical use
File intelligence	Archive by SHA-256 with controlled preview, download, and session links.	Extract indicators and hand content to downstream sandbox analysis.
Honeypots and nodes	Inventory, heartbeat, version, architecture, IP, config progress, update, delete, and credential rotation.	Asset control, health review, and remote operations.
Alerts	Record important state changes and tenant-scoped read status.	Reduce missed offline nodes, config failures, or new activity.
Search and export	Filter and paginate events, sessions, sources, credentials, and files, then export tables.	Investigation records, offline statistics, and external workflows.
Experience and access	Multiple languages, light and dark themes, responsive UI, SSO, and controlled public dashboards.	Desktop or mobile operations with explicit private and public boundaries.

All queries, downloads, exports, and mutations inherit the current workspace context. UI visibility is not the authorization boundary; the API still validates tenant ownership for every resource read and change.

13 Security and Privacy

Open product access and open customer data are fundamentally different decisions.

Control	Current implementation	Operational guidance
Authentication	OIDC Authorization Code + PKCE; state, nonce, issuer, signature, and time checks.	Enable MFA, anomaly detection, and identity lifecycle controls at the provider.
Sessions	Encrypted server-side session cookie and coordinated provider logout.	Require HTTPS, set appropriate lifetime, and avoid shared terminals.
Writes	Same-origin browser writes; bearer or signature validation for nodes.	Preserve Host and Origin at the proxy and reject abnormal cross-origin traffic.
Isolation	Request tenant_id linked to SQL filters, settings, token tid, and file paths.	Run two-user isolation and authorization regression tests before release.
Secrets	client_secret, administrative tokens, and signing keys stay server-side.	Inject through secret management, rotate, and exclude from code and logs.
Minimization	The UI receives only required fields; public views restrict sensitive values.	Define retention and deletion rules for raw events and files.
Platform authority	License changes require a management bearer or platform role.	Separate service operators from ordinary members and review regularly.

Privacy obligations depend on deployment location, collected content, and intended use. Operators should document scope, retention, and access, then assess applicable laws and internal policy. HoneyTrace must not be used for unauthorized monitoring.

Recommended prerequisites for open SSO: migrate the database; enforce private personal workspaces; use production TLS; rotate demo credentials; validate issuer and JWKS; run cross-tenant tests; establish backup, monitoring, retention, and deletion workflows.

14 Reliability, Operations, and Upgrades

A managed security service depends on repeatable operational discipline.

Area	Runtime mechanism	Recommended indicators
Database	PostgreSQL persists accounts, events, and settings; migrations run during upgrades.	Pool use, query latency, disk growth, migration success, and restore drills.
File storage	Persistent volume with tenant directories outside the system space.	Capacity, count, write failures, access audit, and cleanup jobs.
API	Phoenix health, identity, intake, configuration, and query endpoints.	Traffic, P95/P99 latency, 4xx/5xx, OIDC failures, and token rejection.
Nodes	Heartbeat, version, architecture, public address, and config progress.	Availability, heartbeat delay, version drift, update, and restart failure.
Frontend	Static Astro applications use same-domain API routes.	Build integrity, asset delivery, key page availability, and SSO redirects.
Release	Container images, Compose, persistent volumes, and reverse proxy evolve together.	Image digest, rollback time, migration window, and smoke-test outcome.

Recommended sequence: back up data and configuration; build and verify images; apply backward-compatible migrations; start API and site; check health; verify SSO; enroll a test node; ingest one event; prove that two users cannot see each other; then expand access.

The tenant migration is additive: ingest_events receives tenant_id, existing rows become system-owned, import_hash uniqueness becomes tenant_id + import_hash, and saas_accounts stores stable identity mappings.

15 Deployment Models and Boundaries

One product core supports managed personal workspaces and controlled private infrastructure.

Dimension	HoneyTrace Cloud	HoneyTrace Private
Identity	GCSA SSO; registration may accept any provider-authorized user.	GCSA SSO, with an optional controlled break-glass local login.
Data space	One personal tenant_id per issuer + sub identity.	Existing data remains in system within the customer deployment boundary.
Anonymous access	Always disabled for personal spaces.	A system dashboard may be explicitly published by an administrator.
Licensing	Provided as a service capability; members do not manage machine codes.	Machine code, signed license, sensor limit, and maintenance period.
Infrastructure	Platform operator manages API, database, files, TLS, and upgrades.	Customer manages containers, PostgreSQL, volumes, proxy, and backups.
Best fit	Rapid evaluation, individual research, distributed nodes, and no control-plane buildout.	Data residency, internal networks, custom integration, and dedicated operations.

Both modes share telemetry, sensor onboarding, and the analysis experience, but their responsibility boundaries differ. Cloud users should evaluate retention and service terms; private operators own database, secret, network, and upgrade controls.

The current personal workspace model is not an enterprise organization model. Teams that require shared honeypots, member roles, consolidated billing, or team audit should use a dedicated private deployment or wait for organization workspaces.

16 Roadmap and Conclusion

Strengthen deception, isolation, and operability first, then expand collaboration and ecosystem reach.

Phase	Priority capabilities	Position
Version 1.1	Automatic personal workspace provisioning; isolation for events, nodes, settings, tokens, files; regression tests.	Current technical baseline.
Near term	Open-registration policy, quotas, rate limits, retention, tenant metrics, account suspension, and deletion.	Operational readiness for broader access.
Mid term	Organization workspaces, invitations, RBAC, shared honeypots, transfer, billing, and team audit.	Expansion from individual to team workflows.
Later	Threat intelligence exchange, detection rule ecosystem, webhooks, APIs, tiered storage, and advanced reports.	Cross-system integration and long-term research value.

HoneyTrace places the default shell, proxy, and LLM honeypots on one control and analysis plane. Nodes observe real attacker interaction, the normalized event model preserves investigative context, the console connects sources, sessions, commands, credentials, scans, and files, and one-click lifecycle controls keep a distributed fleet deployable and current.

For cloud access, verified tenant isolation makes open SSO practical. Unified identity establishes who the user is, a workspace establishes what belongs to that user, sensor credentials establish who receives telemetry, and end-to-end filtering determines what the user can ultimately see. This baseline also leaves a clear path to organization workspaces, quotas, audit, and intelligence exchange.

Learn more

honeytrace.gcsa.org | [Contact GCSA](https://www.gcsa.org) | [gcsa.org](https://www.gcsa.org)

This document describes the Version 1.1 technical design and product boundary. It is not a legal, compliance, or commercial commitment. Actual capabilities depend on the deployed version, service terms, and runtime configuration.

APPENDIX Key Configuration and Interfaces

A concise reference for technical evaluation and release readiness.

OpenID Connect configuration

Variable	Example or purpose
GCSA_SSO_ISSUER	https://sso.gcsa.org/oidc
GCSA_SSO_CLIENT_ID	Application identifier assigned by GCSA SSO.
GCSA_SSO_CLIENT_SECRET	Backend runtime only; never expose to the browser or repository.
GCSA_SSO_REDIRECT_URI	https://honeytrace.gcsa.org/admin/auth/callback
GCSA_SSO_POST_LOGOUT_REDIRECT_URI	https://honeytrace.gcsa.org/admin/
GCSA_SSO_SCOPES	openid profile email, plus any required role scope.
GCSA_SSO_REQUIRED_ROLE / ALLOWED_*	Optional application policy. Empty values accept every SSO-authorized identity.
GCSA_SSO_PLATFORM_ADMIN_ROLES	Service operator roles only; default honeytrace_admin,gcsa_admin.

Key endpoints

Endpoint	Purpose
GET /api/auth/sso/login	Create OIDC context and redirect to authorization.
POST /api/auth/sso/callback	Exchange code, verify identity, provision workspace, and create session.
GET /api/session	Return the current user, workspace, and management session.
POST /api/honeypots	Create a honeypot and issue a tenant-aware sensor token.
POST /api/honeypots/install	Generate a Docker or native one-click install command with short-lived enrollment credentials.
PATCH /api/sensor-config	Persist desired configuration and notify the node through the live event channel.
POST /api/honeypots/:sensor/update	Ask a node to pull the current official release and restart.
DELETE /api/honeypots/:sensor	Verify ownership, remove the honeypot, and request runtime cleanup.
POST /api/ingest	Validate sensor ownership and persist tenant events.
GET /api/dashboard	Run tenant-scoped aggregation, paging, and capability checks.
GET /api/health	Deployment health check.

Release checklist: migration complete; TLS active; SSO metadata reachable; client secret rotated; system history preserved; new workspace empty; user A cannot see user B; sensor ownership correct; backup and restore verified.