Security whitepaper

What Palladin protects and where trust boundaries lie.

1.0 25.10.2026

Security model

How does Palladin protect data?

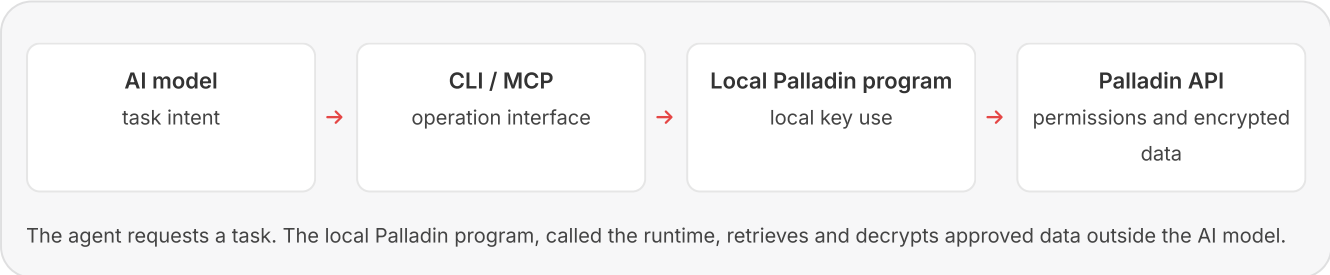
Palladin lets AI agents use passwords and API keys under your control. Data is encrypted on your device. You decide what an agent can access and how it can use that data.

Zero-knowledge guarantee

Zero-knowledge means the server stores encrypted vault content without the keys needed to read it. Encryption and decryption happen in the user app or the agent's local program.

The server still manages accounts, permissions, and operation records. API keys used to connect to Palladin are separate from secrets stored in a vault.

Who can access data?



The agent requests a task. The local Palladin program, called the runtime, retrieves and decrypts approved data outside the AI model.

Participant	Responsibility
User client	Unlock, encryption, and access approval.
Backend	Identity, current authorization, limits, and ciphertext storage.
Native Agent runtime	Material verification, local decryption, and operation execution.
AI model	Requests a task; plaintext access depends on the chosen method.
Target process or site	Receives the data needed to use a credential; part of the disclosure boundary.

Three kinds of control

Encryption protects content. **The server** checks permissions. **The local Palladin program** performs the allowed operation. Each layer has a different job. Blocking another retrieval does not erase data a recipient already copied.

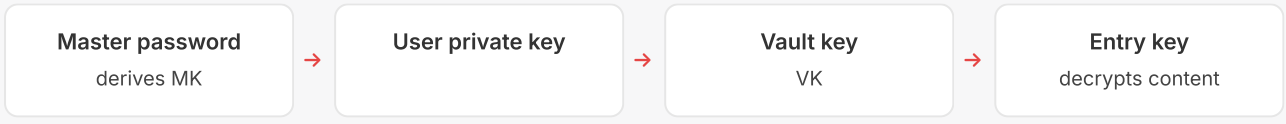
Data and keys

Why use several keys?

Palladin uses separate keys for a user, a vault, and an entry. Sharing one entry therefore does not have to mean sharing the key to the entire vault.

MK — Master Key — is derived from the master password using Argon2id, an algorithm that makes large-scale password guessing more expensive. MK protects the user's private key.

VK — Vault Key — opens the entry keys in a vault. Each authorized member receives their own encrypted copy of VK, which they can open with their private key. We call this protected copy a **key envelope**.



Unlocking takes place on the user's device. The server does not perform these steps.

Other keys

Protocol name	What it protects
EntryDEK — entry encryption key	Entry content and descriptions for vault members. DEK stands for Data Encryption Key.
GrantDEK — access-grant key	A copy of data prepared for a particular approval to access an entry.
VDK — vault discovery key	Metadata an agent can search without receiving the secret's value.
Agent X25519 and Ed25519 keys	Receiving encrypted data and signing agent requests, respectively.

What can the server see?

The server stores encrypted data and key envelopes. It knows organization, vault, entry, and agent identifiers, along with permissions, versions, and audit events. Encryption does not hide every fact about service usage.

Search data is encrypted separately from secrets. An agent can find an entry without receiving its password or API key.

Preventing substitution

Before decrypting, the application checks that data belongs to the correct vault, recipient, and access grant, and has the correct version. It compares these with trusted connection and authorization state, not just the description attached to the received package.

Where are keys stored?

The web app keeps decrypted keys in the unlocked session's memory. Locking clears them from that session. Locally saved encrypted data still requires a key to read it.

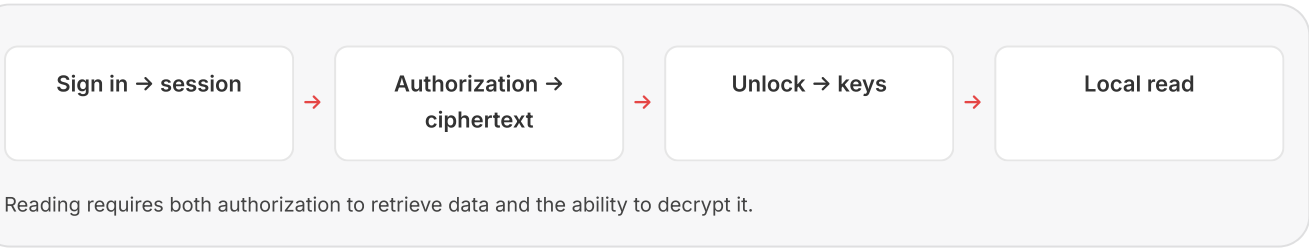
The mobile app is not yet released to stores. Its quick-unlock implementation can persist the master key under system protection. This behavior is under security review and blocks the mobile release; it is not an approved exception to the in-memory key model.

Agent identity keys are held in system key storage or by a separate protective service called a broker. The installation guide explains `Hardened` and `Convenience`.

Identity and unlock

Signing in and unlocking a vault

Signing in establishes who you are. Unlocking a vault gives the app the keys to read it. These are separate steps: a login session alone cannot decrypt your data.



Reading requires both authorization to retrieve data and the ability to decrypt it.

Recovery

Your recovery key restores access when you lose your master password. The client opens the protected material locally, lets you set a new password, and creates a new recovery key. Store the new key in a safe place.

Restoring a login session alone does not decrypt a vault. Palladin cannot recover data without the material needed to decrypt it.

Identity changes

Changing a password can preserve an existing private key and protect it with the new password. Replacing the keypair is a different operation: access to shared vaults must be prepared again.

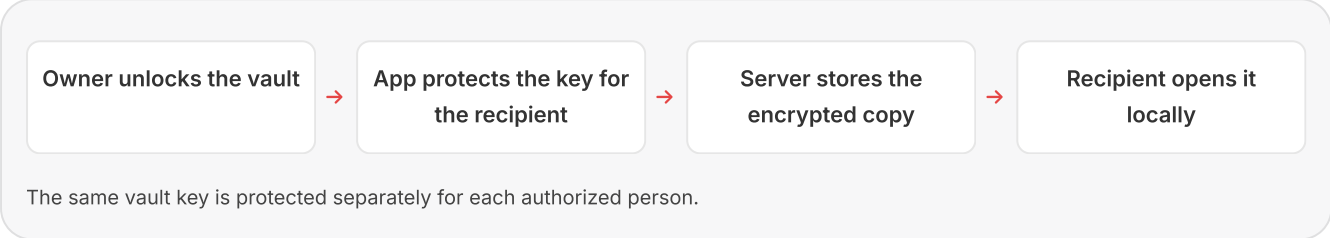
Client boundary

Your browser downloads the app code from a server. Someone who replaces that code may try to capture data when you unlock. Vault encryption therefore does not replace application and device security.

Vaults and sharing

How do you share a vault?

Sharing a vault gives the recipient permission and an encrypted copy of its **Vault Key (VK)**. The recipient opens that copy with their private key. They do not receive your private key or master password.



The correct recipient

The app checks the recipient's identity, key, and vault membership. Signing in alone does not let someone substitute a key or gain access to data.

Updating an entry

Changing an entry creates a new data version. Copies shared with agents must match it. An authorized user app prepares them: the server does not hold the keys needed to decrypt and rebuild the content.

Removing access

Removing a member blocks further retrievals. Protecting new data from someone holding an old key requires changing the key and protecting data for the remaining recipients. It does not erase copies the removed member has already read. See revocation and rotation.

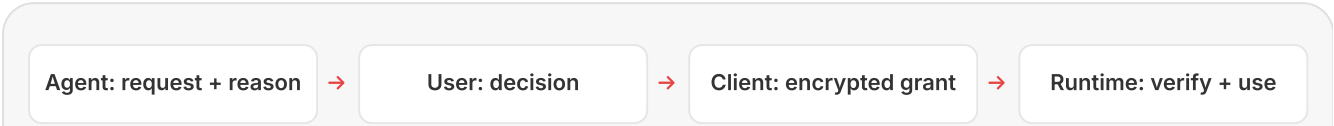
Agent identity and grants

Agent identity

Each agent has its own identifier and keys: one for receiving encrypted data, another for signing requests. A profile is the local name you use to select that agent. An API key connects it to an organization, but does not replace its own identity.

Browser pairing allows approval without copying a secret into the model. An internal Agent-bound credential remains under the logical organization key and creates no independent permissions.

Approving access



Approval covers a specific scope and method. The approving client prepares cryptographic material.

An approval to access data is called a **grant**. It defines the data scope, permitted use, duration, and usage limit. The server checks these conditions, while the local Palladin program checks that received data belongs to the correct agent and task.

GRANULAR and FULL

Model	Material	Meaning
GRANULAR	A separate key and copy of data for a specific entry	The agent receives only the approved data, without the key to the entire vault.
FULL	Encrypted Vault Key (VK)	The agent receives the key to the vault. This requires greater trust than sharing a single entry.

Retrieval limits control server responses. They do not prevent a vault-key holder from decrypting data they already have.

Methods and limits

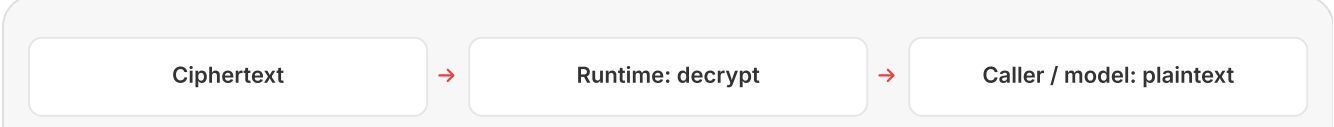
A grant specifies permitted use methods. `Exec` permission does not authorize `Get`. Retrieving encrypted data may consume a use even if the program later fails. Retrying may count as another use.

Denial and deliberate revocation are user decisions. An Agent should not bypass them using a new profile, another method, or repeated requests.

Using a secret

Get: deliberate disclosure

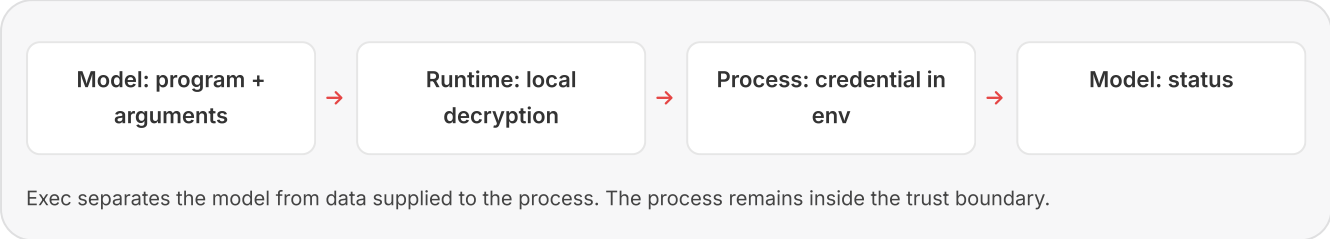
The runtime decrypts approved data and returns it to the caller. If the caller is a model, plaintext enters its context. This path must not be described as hiding secrets from AI.



Get discloses permitted data to its recipient.

Exec: use in a process

The runtime supplies required fields through the target process environment, not command arguments. The standard model response is limited to status; child output is withheld. The target process nevertheless receives the secret and must be trusted for the task.

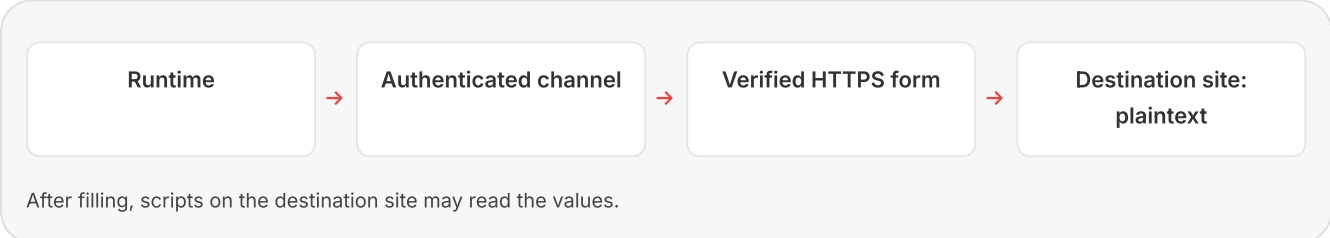


Exec separates the model from data supplied to the process. The process remains inside the trust boundary.

For Script entries, returning the script result to the agent is a separate option. The result may disclose data, including encoded or transformed values. Enable it only for a trusted script and a deliberately selected result.

Inject: use in a browser

The local Palladin program sends the secret to the browser integration over a protected connection. Before filling the form, it checks the HTTPS address, tab, current page, and selected entry. The model receives the operation result without field values.



After filling, scripts on the destination site may read the values.

Inject requires a compatible browser integration and an authenticated channel to the runtime. An extension ID alone does not prove store provenance. An unavailable provider ends the operation; it never automatically enables Get.

Data lifetime

The runtime limits plaintext lifetime and copies. It does not cache delivered credentials between calls. This does not protect against privileged memory inspection. Unix Script execution has a controlled exception: a private temporary source file removed during cleanup. Details depend on the platform and artifact.

Revocation and rotation

What does revocation do?

After approval is revoked, expires, or reaches its usage limit, the server refuses further retrievals. It checks current permissions each time it delivers data. An agent's previously saved status is not enough.

What cannot be revoked?

You cannot make a recipient forget a password or erase a key they already copied. Revoking access in Palladin does not change the password at the destination service either. If a password or API key has leaked, change it with its issuer.

Changing keys

Rotation means replacing an old key with a new one. A new **Vault Key (VK)** protects future data only after entry keys and data are appropriately protected again and access is provided to the remaining recipients. Material copied before the change remains associated with the old keys.

Devices without internet

An offline device cannot immediately receive a revocation notice. The server blocks further retrievals, but does not erase data or keys already saved on the device.

Threats and limitations

Threat model

Threat	Protection	Boundary
Theft of the ciphertext database	Client encryption and key separation	Structural metadata may remain visible; password and KDF strength matter.
Substitution across vaults or Agents	Signatures and independent binding verification	Does not protect against executing a malicious client.
Agent requests excessive access	User decision, GRANULAR, method and time limits	Approving an unsafe scope remains a trust decision.
Prompt injection induces credential use	Authorization outside the model and minimal scope	Palladin cannot reliably infer model intent or detect every harmful task.
Malicious Exec target process	Model separation and controlled delivery	The process receives the secret and can use or disclose it.
Site script reads a filled form	Exact host and document checks before writing	The destination site can observe data in its DOM.
Malicious same-user process	Tier- and OS-dependent protection	Convenience does not isolate the entire same-user environment.
Administrator/root reads memory	Lifetime minimization and platform controls	A compromised device is outside the server-encryption guarantee.

Threat	Protection	Boundary
Revoked recipient retained data	Block new delivery and rotate	Earlier knowledge cannot be revoked.

FULL requires trust

FULL delivers the Vault Key (VK) to the agent's local program. Single-entry API responses, expiry, and usage counters are online controls. They do not establish cryptographic isolation from the rest of a vault against a malicious VK holder.

Distribution is part of security

A modified launcher, runtime, web client, or provider can change system behavior. Signatures, artifact provenance, and version compatibility belong to the security model, not just installation. Missing required evidence should block sensitive operations.

What "no secret in the model" means

It applies to a particular delivery path. It does not cover Get, explicitly returned Script results, or data later read from the target process or site. Hiding a value from the model does not remove the risk of an unwanted operation using that value.

Evidence and releases

Technical sources

Source	Scope
palladin-agent	CLI, MCP, native runtime, and platform contracts.
palladin-crypto	Shared cryptographic primitives and implementations.
palladin-protocol	Public protocol contracts and fixtures.
palladin-security	Security-assurance documentation.

Reading this document

This whitepaper describes Palladin's architecture, key flows, and protection boundaries. Protocol code and compatibility tests supplement the description. An audit report is a separate document: its conclusions apply to the specified version and assessment scope.

Document history

Version	Date	Change
1.0	2026-10-25	Security model, keys, agent access, and protection limits.

PL and EN versions and the HTML, Markdown, and PDF formats describe the same model. The whitepaper version identifies the document independently of the CLI version.

Appendix: cryptography

This appendix contains technical details. See Data and keys for an introduction. VK means Vault Key and DEK means Data Encryption Key.

Cryptographic suite

Operation	Mechanism
Password derivation	Argon2id: 32 MiB memory, 2 iterations, 1 lane, 32-byte account root; profile `identity-argon2id-password-v1`.
Separate authentication and Master Key	HKDF-SHA256 derives two keys from the account root, using distinct purpose labels and the account ID.
Account private-key protection	XSalsa20-Poly1305. Recovery uses a separate Argon2id profile with 19 MiB memory.
Vault content encryption and symmetric key wrapping	XChaCha20-Poly1305, suite `palladin-vault-xchacha-v1`.
Asymmetric key delivery	X25519 sealed package, suite `palladin-x25519-sealed-box-v1`.
Agent identity signatures	Ed25519.
Symmetric VK / DEK	Random 32-byte keys generated client-side.

Algorithm names alone do not specify a protocol. Implementations must use the exact format, signature domains, encoding, nonce rules, and canonicalization from the versioned contract. Do not implement cryptography from this summary alone.

Verify before use

1. Read the explicit suite identifier and protocol version; reject unsupported variants.
2. Check structure and size bounds before processing.
3. Verify signatures and required commitments.
4. Compare organization, vault, principal, grant, revisions, epochs, and key versions against independent authoritative context.

5. Open material only within the permitted boundary and stop on authentication failure.

Implementation compatibility

Shared fixtures and test vectors should verify client/backend compatibility. Negative tests must cover substituted recipients, different vaults, stale revisions, wrong epochs, and modified ciphertext. A library decrypting its own output is insufficient evidence.