Contents

Cerebion Rivet — User Guide **1**
Table of Contents . . . 1
1. Overview . . . 2
2. Best Practices Before You Scan . . . 3
3. Installation . . . 3
4. License Activation . . . 4
5. Dashboard . . . 6
6. Code Analyzer . . . 7
7. Binary Analyzer . . . 9
8. Certificate Analyzer . . . 12
9. Network Analyzer . . . 13
10. Key Management Analyzer . . . 14
11. Cloud Analyzer . . . 15
12. Risk Scoring . . . 16
13. AI-Powered Fixes . . . 18
14. Reporting & Export . . . 19
15. CI/CD Integration . . . 20
16. Settings Reference . . . 22
17. Troubleshooting . . . 26
18. FAQ . . . 28
Contact & Support . . . 29



Cerebion Rivet — User Guide

Version 1.1.106 | Post-Quantum Security Analysis Platform

Table of Contents

1. Overview
2. Best Practices Before You Scan
3. Installation
4. License Activation
5. Dashboard
6. Code Analyzer
7. Binary Analyzer
8. Certificate Analyzer
9. Network Analyzer
10. Key Management Analyzer
11. Cloud Analyzer

12. Risk Scoring
13. AI-Powered Fixes
14. Reporting & Export
15. CI/CD Integration
16. Settings Reference
17. Troubleshooting
18. FAQ

---

1. Overview

Cerebion Rivet is a post-quantum security analysis platform that identifies cryptographic vulnerabilities in source code, compiled binaries, TLS certificates, and live network infrastructure. It assesses how exposed your systems are to quantum computing threats and provides actionable migration guidance.

The Quantum Threat

Quantum computers running Shor's algorithm will break RSA, ECC (all curves), and Diffie-Hellman key exchange. Grover's algorithm halves the effective security of symmetric algorithms, making AES-128 and SHA-256 significantly weaker. The current best estimates put a cryptographically relevant quantum computer 10–20 years away — but **harvest-now-decrypt-later attacks mean sensitive data encrypted today with RSA or ECC is already at risk**.

What Rivet Does

Analyzer	What it scans	Output
Code Analyzer	Source code (46 languages and config formats)	Findings by severity, AI-generated fixes
Binary Analyzer	Compiled binaries (PE, ELF, Mach-O, .NET, Java)	Quantum Risk Score 0–100
Certificate Analyzer	TLS/SSL certificates and live HTTPS endpoints	Quantum Risk Score 0–100
Network Analyzer	Live hosts — open ports, TLS configuration	Quantum Risk Score 0–100
Key Management Analyzer	Live key/certificate management infrastructure (Vault, cloud KMS, AD CS, and more)	Quantum Risk Score 0–100

All Analysis is Local

All analysis runs entirely on your machine. No files, source code, or binaries are ever sent to Cerebion servers. Rivet is designed to work in air-gapped environments with zero network connectivity.

---

2. Best Practices Before You Scan

Important: Always work on a copy or a dedicated branch — never run scans or apply AI-generated fixes directly on production systems or your main repository branch.

- **Use a branch or backup.** Before applying any AI-generated fixes, create a Git branch or backup copy of your files. This lets you review, test, and roll back changes safely.
- **Review fixes before applying.** AI-suggested fixes are provided as guidance only. Always review and test them in a non-production environment before committing to production.
- **Non-production first.** Run analyses in a development or staging environment where possible, especially when using the AI Fix feature.

---

3. Installation

System Requirements

Platform	Requirement
Windows	Windows 10 or later (64-bit)
macOS	macOS 12 Monterey or later
Linux	Ubuntu 22.04+ recommended
RAM	4 GB minimum, 8 GB recommended
Disk	1 GB for application, additional space for scan history

Installing

1. Download the installer for your platform from cerebion.com/download or from your organization's internal distribution point.
2. Run the installer and follow the on-screen prompts.
3. Launch Cerebion Rivet from the Start Menu (Windows), Applications folder (macOS), or application launcher (Linux).

   Linux note: If your distro is missing required runtime libraries, install them using your package manager before launch. On Ubuntu-based systems, the release notes may include a ready-to-run `apt-get` command. If the archive includes a sandbox helper, you may also need to apply the recommended permissions once during first setup.

   Windows note: Windows may display a SmartScreen warning during install. Click **More info → Run anyway** to proceed. This is expected for newly released software.

On first launch, Rivet will prompt you to accept the End User License Agreement and activate your license.

If you do not have a license yet, Rivet can still open in limited mode so you can start a free trial from the app.

---

4. License Activation

Online Activation

1. Open Cerebion Rivet and go to **Settings → License**.
2. Under the **Online License** tab, enter your license key from your trial signup or purchase confirmation email.
3. Click **Activate License** — Rivet validates against license.cerebion.com and saves the key automatically.

To move your license to a different machine, click **Detach License** on the current machine first, then activate on the new one. If you no longer have access to the old machine, contact support@cerebion.com to free the seat.

Command-Line Activation

On a machine without the desktop app, activate with one command. Rivet asks for the key without showing it on screen:

```
rivet enroll
```

The key can also be piped in (`echo "$KEY" | rivet enroll`) or read from a file for unattended installs (`rivet enroll --key-file /run/secrets/rivet_license`). It is never taken from the command line itself, where it would show up in process listings and shell history.

Rivet checks the key with the license server, then keeps it in your operating system's credential store (Windows Credential Manager, macOS Keychain, or the Linux Secret Service). On a machine that has none, such as a headless server, it keeps the key in a file only your account can read. The key is never written to your environment or to a configuration file. The desktop app and the command line share the activation on the same machine.

To check or manage the license afterwards:

```
rivet license status       # what this machine has, without contacting the license server
rivet license refresh      # check with the license server now
rivet license deactivate   # release this machine's seat and remove the license from it
```

Headless / CI Activation

On CI runners and containers, give each run its key through an environment variable. No activation step is needed:

```
# Linux / macOS
export RIVET_LICENSE_KEY=your-license-key-here

# Windows (cmd)
set RIVET_LICENSE_KEY=your-license-key-here
```

Rivet validates online on first run and caches the result for 7 days in its per-user data folder (`%APPDATA%\CerebionRivet\license` on Windows, `~/Library/Application Support/CerebionRivet/license` on macOS, `~/.local/share/CerebionRivet/license` on Linux). Subsequent runs use the cache — no internet needed until it expires. Store the key in your CI/CD platform's secret store (GitHub Actions Secrets, GitLab CI Variables, Jenkins Credentials) — never hardcode it in pipeline files. A

key given through `RIVET_LICENSE_KEY` takes precedence over one activated with `rivet enroll` or in the desktop app.

If a container or server has an OS keyring that prompts or is locked, set `RIVET_SECRET_STORE=file` to keep the key in the protected file instead.

Offline / Air-Gapped Activation

> **Enterprise plan required.** Offline licensing is not available on Trial or Professional tiers.

Offline licenses are cryptographically signed with Ed25519 and hardware-fingerprinted to a specific machine. Each machine requires its own license file.

Step 1 — Get your device fingerprint

In Rivet, go to **Settings → License → Online License** tab and click **Show Device Fingerprint**. Copy the fingerprint string or save it as `device-fingerprint.json`.

Email the fingerprint to support@cerebion.com with your license key or order reference. Support will send back a signed `license-[uuid].json` file.

The fingerprint contains only hardware identifiers (MAC address hash, architecture, hostname) — no personal data or file contents.

Step 2 — Upload the license file

1. Go to **Settings → License → Offline License** tab.
2. Drop the `license-[uuid].json` file onto the upload area or click to browse.
3. Click **Upload License File** — Rivet verifies the signature locally, no internet needed.

Rivet remembers the file, so the license stays active after you restart the application, and the `rivet` command line recognizes it too.

Deploying to headless machines:

Point Rivet at the license file with a configuration setting. It is kept in the Rivet configuration, so it survives restarts and reboots:

```
# For your own user
rivet config set license.offline_file /path/to/license-[uuid].json

# For every user of the machine (run as administrator or root)
rivet config set license.offline_file /etc/rivet/license.json --scope machine
```

An administrator can also prevent users from changing the path: `rivet config set locked "[license.offline_file]" --scope machine`.

The `OFFLINE_LICENSE_FILE` environment variable still works and takes precedence over the setting:

```
# Linux / macOS
export OFFLINE_LICENSE_FILE=/path/to/license-[uuid].json

# Windows (cmd)
set OFFLINE_LICENSE_FILE=C:\path\to\license-[uuid].json
```

Using your own license server:

If your organization runs its own license server, point Rivet at it with a configuration setting. To verify offline license files that server issues, also give Rivet the server's public key (the server publishes it at `<server address>/api/config` as `offline_public_key`):

```
# For your own user
rivet config set license.server_url https://license.example.com
rivet config set license.offline_public_key <public key>

# For every user of the machine (run as administrator or root), and to prevent users from chan
rivet config set license.server_url https://license.example.com --scope machine
rivet config set license.offline_public_key <public key> --scope machine
rivet config set locked "[license.server_url, license.offline_public_key]" --scope machine
```

The `LICENSE_SERVER_URL` and `OFFLINE_PUBLIC_KEY` environment variables still work and take precedence over these settings. Changes apply without restarting Rivet.

Common deployment approaches: - **Bake into machine image** — include the file in your AMI, Docker image, or VM template at a fixed path (e.g. `/etc/rivet/license.json`). One-time setup, survives reboots. - **Deploy via secrets manager** — store the file contents in AWS Secrets Manager, HashiCorp Vault, or similar. Add a provisioning script that writes the secret to disk before Rivet runs. - **Copy manually** — for a small number of fixed machines: `scp license.json user@host:/etc/rivet/license.json`

License Type	Trial Period	Air-Gap Support
Trial	14 days	No
Professional	—	No
Enterprise	—	Yes

The command line and the desktop app share this machine's activation. A license file uploaded in the app, or set with `rivet config set license.offline_file`, is recognized by both.

---

5. Dashboard

The Dashboard provides a snapshot of your security posture across all analyzers.

Overview Panel

Metric	What it shows
Overall Risk Score	Weighted aggregate of all recent scan results
Critical Findings	Count of CRITICAL severity items requiring immediate action
High Risk Assets	Assets (domains, binaries, code repos) scoring above 60
PQC Ready	Percentage of scanned assets already using post-quantum algorithms

Recent Activity

The Recent Activity panel shows the last 10 scans across all analyzers, with timestamp, scan target, score, and status. Click any entry to open the full results.

Quantum Threat Timeline

A visual timeline showing estimated years until quantum computers can break each algorithm class:

- **RSA / ECC / DH** — 10–20 years (Shor's algorithm)
- **AES-128 / SHA-256** — effective security halved by Grover's (plan migration)
- **AES-256 / SHA-384+** — quantum-resistant (no action needed)

---

6. Code Analyzer

Static analysis for quantum-vulnerable cryptography in source code, combining a rule-based scanning engine with custom scanners for languages requiring specialized detection.

Supported Languages

46 languages and configuration formats in total:

Category	Languages
Popular programming languages	Python · JavaScript · TypeScript · Java · C / C++ · C# · Go · Rust · Ruby · PHP · Kotlin · Scala · Swift
Scripting & shell	Bash · Lua · Perl · PowerShell · R · VB / VBScript
Enterprise & legacy	COBOL · Groovy · Apex (Salesforce)
Functional	Elixir · Clojure · OCaml · Scheme
Web & mobile frameworks	Vue · Dart
Blockchain / smart contracts	Solidity · Cairo · Circom · Move
Infrastructure as code	Terraform · HCL · Dockerfile
Config, data & schema formats	JSON · YAML · TOML · XML · Jsonnet · .env files · Protocol Buffers
Database	SQL
Certificates	.pem, .crt, .cer, .p7b, .p7c

> Every language listed above is verified against live-fire test scans as part of Rivet's own release testing, not just claimed — see the release notes for details.

What Gets Detected

Category	Examples
Asymmetric encryption / key exchange	RSA key generation, RSA-OAEP, ECDH, ECDHE, DH, DHE
Digital signatures	ECDSA, DSA, RSA signatures

Category	Examples
Weak hash functions	MD5, SHA-1, SHA-224, SHA-256 (Grover risk)
Weak symmetric encryption	AES-128, DES, 3DES, RC4, Blowfish
Deprecated TLS	TLS 1.0, TLS 1.1, weak cipher suites, SSLv3
Hardcoded secrets	Private keys, API tokens, cryptographic material in source
Insecure random	Math.random(), rand() in cryptographic contexts
Key size issues	RSA $<$ 4096, DH $<$ 3072, ECC below P-384

Running a Scan

1. Click **Code Analysis** in the sidebar.
2. Click **Select Folder** and choose your project directory.
3. Configure file extensions and excluded directories in **Settings → Code Analysis** if needed.
4. Click **Scan**.

Findings stream to the UI in real time as batches complete. The progress bar shows files scanned vs total.

Cancellation: Click **Cancel** during a scan to stop it. Partial results are saved to history and marked as *Partial* — findings collected up to that point are preserved.

Understanding Results

Field	What it means
Severity	CRITICAL / HIGH / MEDIUM / LOW / INFO — set by the matching rule
Category	PQC badge shown when the finding involves a quantum-vulnerable algorithm
Line	Line number of the vulnerable code
Message	Description of the issue and why it is quantum-vulnerable, including the recommended migration
Actions	Fix generation buttons — see Section 10

Severity	Quantum meaning
CRITICAL / ERROR	Directly broken by Shor's algorithm (RSA, ECDSA, DH) — immediate migration required
HIGH / WARNING	Significant quantum risk — weak key sizes, AES-128, SHA-256 in critical contexts
MEDIUM	Deprecated but not immediately broken algorithms, weak TLS configurations
LOW / INFO	Best practice violations, lifecycle warnings

Scan History

Every scan is saved to history. Cancelled scans are stored as *Partial*. Clean files (no findings) are stored separately and lazy-loaded on demand.

---

7. Binary Analyzer

Scans compiled binaries for quantum-vulnerable cryptographic patterns without requiring source code. Combines YARA pattern matching, binary structure analysis, and disassembly.

Cross-platform analysis: Rivet can analyze binaries from any platform regardless of your OS. A Windows user can analyze Linux ELF binaries and macOS Mach-O libraries. Rivet does not unpack container images: extract the binaries from an image first, then scan them.

Supported File Formats

Format	Platform	Notes
PE (.exe, .dll, .sys)	Windows	Includes .NET assemblies
ELF	Linux	Shared libraries (.so) and executables
Mach-O (.dylib, .app, .framework)	macOS	Universal binaries supported
Java .class / JAR	Cross-platform	JAR contents scanned individually
DEX	Android	Basic string extraction

Scan Options

Analysis Type:

Option	What it does
Crypto (default)	Cryptographic algorithm detection — imports, symbols, YARA patterns, API calls
Full	Crypto + hardcoded secrets, security flags, binary metadata

Analysis Depth:

Option	What it does
Quick	YARA pattern matching and import table scan only. Fastest — suitable for triage.
Standard (default)	YARA + full binary structure analysis + disassembly of code sections
Deep	All of the above plus extended disassembly and crypto constant detection

Business Criticality: Low, Medium (default), High or Critical. How critical the system that uses the binary is; it sets part of the Business Impact component of the score, so the same binary scores higher for a critical system than for a low one. On the command line: `rivet scan-binary app.exe --business-criticality critical --compliance fips_140,fedramp` (`--compliance` takes a comma-separated list of the frameworks that apply).

How Analysis Works

1. **YARA pattern matching** — scans raw binary bytes against quantum-specific YARA rules. Fast and format-agnostic. Runs on every scan.
2. **Binary structure parsing** — parses PE/ELF/Mach-O, extracts imported symbols, exported functions, and section metadata.
3. **Disassembly** — disassembles code sections to identify cryptographic operations at the instruction level.
4. **.NET metadata extraction** — for PE binaries, extracts managed code metadata to detect .NET crypto API usage.
5. **Risk scoring** — findings from all sources feed into the unified 4-component Quantum Risk Score.

Q-Score Reference

Each YARA rule match carries a Q-Score (1–5) indicating quantum severity:

Q-Score	Severity	Examples
5	CRITICAL	RSA (any key size), ECDSA, ECDH, DH, AES-128, MD5, SHA-1
4	HIGH	Kyber-512, Dilithium-2, FALCON-512, PRNG in quantum contexts
3	MEDIUM	RSA-1024/2048 key size constants, legacy crypto in quantum-aware systems
2	LOW	PQC performance-over-security modes, debug logging of quantum state
1	INFO	Migration recommendations, algorithm lifecycle warnings

Post-Quantum Migration Reference

Vulnerable Algorithm	Post-Quantum Alternative	NIST Standard
RSA (signatures)	ML-DSA (CRYSTALS-Dilithium)	FIPS 204
ECDSA	ML-DSA or SLH-DSA (SPHINCS+)	FIPS 204 / 205
RSA-OAEP / ECDH (encryption)	ML-KEM (CRYSTALS-Kyber)	FIPS 203

Vulnerable Algorithm	Post-Quantum Alternative	NIST Standard
DH / DHE	ML-KEM (CRYSTALS-Kyber)	FIPS 203
AES-128	AES-256, ChaCha20-Poly1305	—
SHA-1 / MD5	SHA-384, SHA-512, SHAKE-256	—

Resource Limits

All configurable in **Settings → Binary Analysis**:

Limit	Default	Range
Max file size	500 MB	1–5000 MB
Analysis timeout	600 seconds	30–3600 s
Max memory	2048 MB	512–8192 MB
Max concurrent scans	3	1–10
Max archive size	500 MB	1–2000 MB
Max files per archive	100	1–1000

Stripped and Packed Binaries

- **Stripped binaries** — YARA pattern matching still works on raw bytes. Binary analysis may find fewer named crypto functions but can still detect crypto constants and instruction patterns. Confidence level will be *YARA Primary* or *Medium*. The score may be understated — a low score on a stripped binary should be interpreted with caution.
- **Packed binaries** — the outer packer layer may prevent crypto detection in packed sections.
- **Very small binaries** (< 100 KB) — may return UNKNOWN if no analyzable functions are found.

Library-Linked Algorithms

When a binary links against a crypto library (e.g. OpenSSL), the library's symbols for all supported algorithms appear in the binary even if the application only calls a subset of them. Rivet cross-references YARA matches against confirmed algorithm usage detected by the binary parser. Matches that are present in the linked library but not confirmed as actually called by the application are shown in a separate **Library-Linked Algorithms** section rather than as active vulnerabilities. This prevents false-positive CRITICAL scores on binaries that link OpenSSL but only use AES.

Limitations

The Binary Analyzer performs **static analysis only** — it does not execute code. The following scenarios are outside its detection scope:

- **Runtime-only crypto** — algorithms loaded or assembled at runtime are not visible to static analysis.
- **Custom or proprietary implementations** — crypto that does not use known constants, OIDs, or library function names will not be detected.
- **Encrypted or heavily obfuscated payloads** — if the code section is encrypted and unpacked only at runtime, detection coverage is reduced.

- **Key size inference** — when a key size cannot be extracted, conservative worst-case assumptions are applied (e.g. RSA without a detected key size is treated as RSA-2048).
- **Unsupported formats** — file types other than those listed above return no findings, not a clean result.

  A clean scan result means no known vulnerable cryptographic patterns were detected. It does not constitute a guarantee of quantum safety.

---

8. Certificate Analyzer

Analyzes SSL/TLS certificates and live HTTPS endpoints for quantum cryptography vulnerabilities.

Input Methods

- **Hostname scan** — connects to a live host on port 443 (or custom port) and retrieves the full certificate chain.
- **File upload** — analyze `.pem`, `.crt`, `.cer`, or `.p7b` files directly.
- **Bulk scan** — scan a list of hostnames from a CSV or newline-delimited text file.
- **Identity provider scan**: analyze the token-signing keys an identity provider publishes, along with its TLS certificate. See Identity Provider Keys below.

Running a Scan

1. Click **Certificate Analysis** in the sidebar.
2. Enter a hostname (e.g. `example.com`) or upload a certificate file.
3. Click **Scan**.

For bulk scans, click **Bulk Scan**, upload your CSV/text file, and click **Start Bulk Scan**. Results for each host are saved individually to history.

Result Fields

Field	Range	What it means
Quantum Risk Score	0–100	Higher = more vulnerable
Security Grade	A–F / ?	A (< 20), B (20–39), C (40–59), D (60–79), F (80+). Shows **?** when certificate details could not be retrieved.
Threat Level	minimal / low / medium / high / critical	Highest quantum threat within the certificate's remaining lifetime
Migration	monitor / plan / prepare / urgent / immediate	Recommended urgency
PQC Ready	Ready / Not Ready	Whether the certificate uses a NIST-approved PQC algorithm
Days to Expiry	integer / —	Days until notAfter. Short-lived certificates carry lower quantum risk. Shows — when the expiry date is unavailable.

Risk Score Components

Component	Weight	What drives it
Algorithm Risk	40%	Key algorithm and size vulnerability to Shor's algorithm
Timeline Risk	25%	Whether a quantum computer will exist before the certificate expires
Business Impact	20%	Criticality and compliance requirements
PQC Readiness	15%	Whether PQC algorithms are already in use

Identity Provider Keys

Single sign-on depends on signing keys that an identity provider (such as Microsoft Entra ID, Okta, Ping, ADFS or Keycloak) publishes for every application to trust. These keys are almost always RSA or elliptic-curve keys, which a large quantum computer could break, and a regular certificate scan of the same host does not see them.

1. In the Certificate Analyzer, choose **Identity Provider**.
2. Enter one of the following:
   - an OpenID Connect issuer or discovery URL (for example `https://login.example.com/oauth2/defaul`
   - a JWKS (published key set) URL
   - a SAML or WS-Federation metadata URL
3. Click **Analyze Identity Provider**.

The report adds an **Identity Provider Signing Keys** section to the usual certificate results. It shows each published key with its algorithm and whether it is quantum-vulnerable, and the signing algorithms the provider advertises. It also flags serious misconfigurations, such as private key material in the published key set or a metadata signature that uses SHA-1. These results are saved to history and included in the dashboard, reports, CBOM and exports like any other certificate scan.

From the command line:

```
rivet scan-idp https://login.example.com/oauth2/default
rivet scan-idp https://idp.example.com/saml/metadata --format json -o idp.json
```

The command exits with code 1 when a quantum-vulnerable key is found (use `--no-fail` to always exit 0), and with code 3 when the document lists no keys, so nothing could be analyzed (see Exit Codes). Only `https://` URLs are accepted.

Identity documents committed to a code repository, such as SAML metadata XML and JWKS JSON files, are covered by the Code Analyzer. So are SSH keys, `authorized_keys` and `known_hosts` files.

---

9. Network Analyzer

Scans live hosts for open ports and assesses the quantum security of detected TLS services.

Running a Scan

1. Click **Network Analysis** in the sidebar.
2. Enter a hostname or IP address.
3. Optionally configure port range and scan depth in Settings.
4. Click **Scan**.

   Note: Port scanning without authorization is illegal in many jurisdictions. Only scan hosts you own or have explicit written permission to test.

What Gets Assessed

- Open ports and detected services
- TLS protocol version (TLS 1.0/1.1/1.2/1.3)
- Cipher suite quantum vulnerability
- Certificate chain quantum risk (same as Certificate Analyzer)
- Weak key exchange (DHE $<$ 2048, ECDHE with weak curves)

Port Restrictions

System ports (below 1024) are restricted by default. Use **Settings → Network Analysis** to configure allowed port ranges.

---

10. Key Management Analyzer

Scans live key and certificate management infrastructure for quantum-vulnerable algorithms — a different approach from the other four analyzers, which read a file or probe a network endpoint. Key Management Analyzer connects directly to your key management systems using credentials you issue, and enumerates key and certificate metadata for analysis.

Supported Backends

Backend	What it scans
HashiCorp Vault	Transit encryption keys, PKI certificate authorities
AWS KMS	Customer-managed keys
AWS Certificate Manager	Issued and imported certificates
Azure Key Vault	Keys and certificates
Azure Managed HSM	Hardware-backed keys
GCP Cloud KMS	Key rings and crypto keys
Kubernetes Certificate Manager (cert-manager)	Certificate custom resources
Microsoft AD CS	Certificate authorities and certificate templates

Credential Handling

Key Management Analyzer requires a credential you issue and scope yourself — a Vault AppRole, an AWS IAM key, an Azure Service Principal, a GCP Service Account key, a Kubernetes ServiceAccount

token, or an AD domain bind account. Rivet never asks for broader access than you grant, and this analyzer is on-premise only — it is not offered through any hosted or cloud deployment.

- Your credential is stored only in your own machine's operating system credential manager (Windows Credential Manager, macOS Keychain, or Linux Secret Service) — never in Rivet's own database, and never transmitted to Cerebion.
- Rivet cannot decrypt, sign, or export anything using this credential — every connection is read-only by design, not just by policy.
- You can revoke access at any time from your own infrastructure, instantly, without contacting Cerebion.

Running a Scan

1. Click **Key Management** in the sidebar.
2. Click **Add Connection**, choose a backend, and enter the connection details and credential.
3. Click **Test Connection** to verify access before saving — or use **Quick Scan** to run a one-time scan without saving a connection.
4. Click **Scan** to enumerate keys and certificates and assess their quantum risk.

Saved connections can be re-scanned at any time from the Key Management page, and results appear in Scan History, the Dashboard, and Reports/Export like every other analyzer.

What Gets Detected

The same algorithm categories as the other analyzers — RSA, ECC, and DH/DSA key generation, weak hash and cipher usage — evaluated against the metadata your key management system reports for each key or certificate, and scored on the same 0–100 Quantum Risk Score scale used throughout Rivet.

---

11. Cloud Analyzer

Scans cloud infrastructure for quantum-vulnerable cryptography across AWS, Azure, and GCP. Unlike the other analyzers, Cloud Analyzer connects to your cloud accounts using credentials you supply and enumerates managed services — load balancers, CDN distributions, API gateways, storage, databases, secrets, and HSMs — assessing each for quantum-vulnerable algorithms.

Supported Cloud Providers

Provider	Services Scanned
AWS	Elastic Load Balancers (ALB/NLB/Classic), CloudFront distributions, API Gateway stages, S3 bucket encryption, RDS instances, EBS volumes, Secrets Manager secrets, CloudHSM clusters
Azure	Application Gateway SSL policies, Storage Account TLS settings, SQL Server minimum TLS version
GCP	Cloud Load Balancing SSL policies (including whether post-quantum key exchange is enabled), Cloud Storage encryption, Cloud SQL TLS enforcement, Secret Manager

AWS Certificate Manager and AWS KMS are covered by the **Key Management Analyzer**, not the Cloud Analyzer. From a terminal, the Cloud Analyzer runs as `rivet scan-aws-account`, `rivet scan-azure-account` and `rivet scan-gcp-account`.

Credential Handling

Cloud Analyzer requires a read-only credential you issue and scope yourself — an AWS IAM access key, an Azure Service Principal, or a GCP Service Account key. Credentials are stored only in your machine's operating system credential manager (Windows Credential Manager, macOS Keychain, or Linux Secret Service) — never in Rivet's database, and never transmitted to Cerebion.

For AWS, the minimum required IAM permissions are:

- `elasticloadbalancing:Describe*`
- `cloudfront:List*`, `cloudfront:Get*`
- `apigateway:GET`
- `s3:ListAllMyBuckets`, `s3:GetBucketEncryption`
- `rds:DescribeDBInstances`
- `secretsmanager:ListSecrets`, `secretsmanager:DescribeSecret`
- `cloudhsm:DescribeClusters`
- `sts:GetCallerIdentity` (used for connection verification)

Running a Scan

1. Click **Cloud Analyzer** in the sidebar.
2. Click **Add Connection**, choose a cloud provider, and enter the connection details and credential.
3. Click **Test Connection** to verify access before saving.
4. Click **Scan** to enumerate cloud resources and assess their quantum risk.

Saved connections can be edited (pencil icon) or deleted (trash icon with confirmation) at any time. Results appear in Scan History with a Risk Score column and status indicators.

What Gets Detected

The same algorithm categories as the other analyzers — RSA, ECC, and weak symmetric encryption — evaluated against the metadata your cloud provider reports for each resource. Resources using quantum-vulnerable algorithms are flagged with their algorithm, key size, and recommended migration path. Resources with no TLS or encryption configuration are flagged separately.

Scan History

Each scan result is saved to history with a timestamp, resource count, risk score, and connection status indicator. A shield icon shows whether the connection was reachable at scan time. Results can be exported from the Reports page.

---

12. Risk Scoring

All Binary, Certificate, Network, and Key Management scans produce a unified **Quantum Risk Score from 0 to 100**.

Score Ranges

Range	Level	Recommended Action
0–14	Minimal	Quantum-resistant or very low risk
15–39	Low	Monitor; no immediate action needed
40–59	Medium	Plan migration within 3–5 years
60–79	High	Prioritize migration
80–100	Critical	Begin migration immediately

These are the default cut-offs of the risk engine. An administrator can change them (see `QR_CRITICAL_THRESHOLD`, `QR_HIGH_THRESHOLD`, `QR_MEDIUM_THRESHOLD` and `QR_LOW_THRESHOLD`). Some screens also show their own labels, such as the certificate Security Grade (A–F) described in the Certificate Analyzer section.

Score Components

Component	Weight	Description
Algorithm Risk	40%	Vulnerability of detected algorithms to Shor's/Grover's. RSA/ECC/DH score highest.
Timeline Risk	25%	Whether a quantum computer capable of breaking the detected algorithms is expected before the asset's replacement cycle.
Business Impact	20%	Business criticality and compliance requirements (FedRAMP, HIPAA, PCI, SOX).
PQC Readiness	15%	Whether post-quantum algorithms (ML-KEM, ML-DSA, SLH-DSA) are already present.

Algorithm Risk Reference

Algorithm	Risk Score	Notes
RSA-1024	95	Practically breakable today, and the first to fall to a quantum computer
RSA-2048	88	Broken by Shor's algorithm; the most common vulnerable RSA

Algorithm	Risk Score	Notes
RSA-3072	82	A larger key buys a few years, not a different outcome
RSA-4096	78	Still fully broken by Shor's algorithm; it only needs more qubits
RSA-8192	72	Same, with the longest timeline
ECDSA / ECDH P-256 (also secp256k1, Ed25519)	90	Fewer qubits needed than RSA
ECDSA / ECDH P-384	80	Larger curve, slightly longer timeline
ECDSA / ECDH P-521 (also Ed448)	72	Largest curve, longest timeline
DSA / DH	85–95	Broken by Shor's algorithm. 95 in certificates and network scans, 85 when found in code or binaries
MD5 / SHA-1	100	Classically broken
SHA-224	70	Weak under Grover's algorithm
SHA-256	50	Grover reduces the effective strength
SHA-384 / SHA-512	25 / 20	Considered safe; a small residual score
DES / RC4 / 3DES	100 / 95 / 90	Classically broken or deprecated
AES-128	30	Grover halves effective key size to 64 bits
AES-192 / AES-256	20 / 15	Quantum-resistant; a small residual score

---

13. AI-Powered Fixes

The Code Analyzer can generate and apply fixes for detected vulnerabilities using an LLM.

Requirements

- An LLM API key configured in **Settings → AI Configuration** (cloud provider by default; Enterprise air-gapped licenses can point to a custom or on-premise LLM endpoint instead)
- The file must be inside a Git repository for Apply and Rollback to work

Fix Modes

Two modes are available, selectable in **Settings → AI Configuration**:

Mode	Button	What it generates
AI Recommendation (default)	Generate AI Recommendation	Inserts a single-line `@TODO-pq` comment above the vulnerable code describing the issue and the recommended PQC migration. Does not modify existing code.
AI Code Fix	Generate AI Code Fix	Replaces the vulnerable code with a working PQC-safe implementation and adds a `@TODO-pq REPLACED:` comment. Review carefully — the LLM works from ~30 lines of context.

If AI is disabled or no LLM key is configured, a **Generate Recommendation** button appears instead. This uses the rule autofix field where available, or a rule-based recommender — no LLM required.

Applying a Fix

1. Click the generate button on any finding — a loading indicator appears while the fix is generated.
2. Click **View AI Recommendation** or **View AI Code Fix** to review the diff.
3. Click **Apply Fix** — Rivet applies the patch as an atomic Git commit.
4. If the fix causes issues, click **Rollback** — the commit is reverted.

Safety Notes

- Fixes are applied via Git patch operations — no changes are made without a reviewable diff.
- Apply and Rollback are atomic — either the full patch applies or nothing changes.
- Rivet detects staleness: if the file has changed since the fix was generated, it prompts you to re-generate before applying.

---

14. Reporting & Export

Exporting Individual Scan Results

From any scan result view, click **Export** to download results in your preferred format:

Format	Contents	Best for
CSV	Findings table with severity, algorithm, file location, and recommendations	Spreadsheet analysis, ticketing system import
JSON	Full structured output including all metadata and raw findings	Programmatic processing, custom tooling

Format	Contents	Best for
SARIF	Static Analysis Results Interchange Format — standard findings format	GitHub Advanced Security, Azure DevOps, IDEs

AI Executive Report

From the **Reports** page, select one or more scans, add optional context, and click **Generate Report** to produce an AI-powered executive summary. Generated reports are saved to the Reports history and can be downloaded at any time.

> *Results are based on static pattern analysis. A clean result does not constitute a guarantee of quantum safety.*

Bulk Export

From the **Reports** page, click **Export CSV** or **Export JSON** to download all scan records in one file. Records are sorted by date, newest first (up to 500 most recent).

Code Analysis Export

From the Code Analysis results view, the **Export** button offers CSV, JSON, and SARIF formats for the current scan's findings.

---

15. CI/CD Integration

Rivet can be integrated into CI/CD pipelines for automated quantum security scanning on every build.

Install Rivet on the runner (or bake it into your own build image) and give it the license key through the `RIVET_LICENSE_KEY` environment variable. The scan takes the folder as an argument, writes its report with `-o`, and exits with one of the codes in the table below, so a pipeline step fails on findings by itself. A GitHub Action and a ready-made CI image are planned; they are not available yet, so the examples below call `rivet` directly.

GitHub Actions

```yaml
- name: Quantum Security Scan
  env:
    RIVET_LICENSE_KEY: ${{ secrets.RIVET_LICENSE_KEY }}
  run: rivet scan ./src --format sarif -o rivet-report.sarif   # exits 1 on findings
- name: Upload report
  if: always()
  uses: github/codeql-action/upload-sarif@v3
  with:
    sarif_file: rivet-report.sarif
```

GitLab CI

```yaml
quantum-scan:
  script:
    - rivet scan ./src --format json -o rivet-report.json
  artifacts:
    when: always
    paths:
      - rivet-report.json
  variables:
    RIVET_LICENSE_KEY: $RIVET_LICENSE_KEY
```

Jenkins

```groovy
stage('Quantum Security') {
    environment {
        RIVET_LICENSE_KEY = credentials('rivet-license-key')
    }
    steps {
        sh 'rivet scan ./src --format json -o rivet-report.json'
    }
    post {
        always {
            archiveArtifacts artifacts: 'rivet-report.json'
        }
    }
}
```

To let a pipeline pass while you work through existing findings, add `--no-fail` (exit 1 becomes 0). Exit code 3 is not changed by it; see below.

Exit Codes

Every scan command (`scan`, `scan-binary`, `scan-cert`, `scan-network`, `scan-idp`, and all the Key Management and Cloud Analyzer commands) uses the same four codes, and shows this table in its `--help`.

Code	Meaning
0	Clean: nothing was found, and everything was read
1	Findings: something quantum-vulnerable was found
2	Error: the scan could not run (bad options or credentials, license, an unreadable configuration file)
3	Incomplete: nothing was found, but part of the target could not be read, so this is **not** a clean result

If a scan finds something and also could not read part of its target, it exits 1: the pipeline fails either way and the findings are what to act on. The warning about what was not read is still printed

(and is in the JSON output as `status` and `warnings`).

Code 3 means, for example: an AWS, Azure or GCP credential that could sign in but read nothing, a Vault policy that denied the Transit engine, an Azure Key Vault that answered 403, a host that did not answer on any port, a certificate whose handshake failed, a binary that could not be parsed, a language that timed out, or configuration files that could not be scanned because the rule registry was unreachable.

Two options accept a result you have chosen to live with:

- `--no-fail` turns code 1 into 0. It does not touch code 3.
- `--allow-incomplete` turns code 3 into 0. The warnings are still printed, so the gap is in the log even when the job is green.

For an air-gapped pipeline, set `code.online_config_rules: false` once (see Settings Reference) rather than passing `--allow-incomplete` on every run: the scan is then not incomplete, because nothing was skipped by accident.

The Key Management and Cloud Analyzer commands' `--format json` output is an object, `{"status": ..., "warnings": [...], "findings": [...]}`, where it used to be the bare list of findings. `status` is `complete`, `partial` (some objects could not be read: findings are what was read) or `incomplete` (nothing could be read).

---

16. Settings Reference

Configuration Files

The `rivet` command and the desktop app read their scan settings from the same YAML files, so a team can keep them with the code and an administrator can enforce them. Settings are applied in this order, and a later item overrides an earlier one:

1. Built-in defaults. The desktop app adds one more default: it skips dependency, build and cache folders (`node_modules`, `.git`, `venv`, `dist`, `build` and similar) unless you change the list. The command line skips nothing by default.
2. The **machine file** (shared by everyone on the computer, written by an administrator)
3. Your **user file**
4. The **project file**, `.rivet.yml`, found by searching upward from the folder you scan
5. Environment variables (only the existing `LOG_LEVEL`, `LICENSE_SERVER_URL` and `OFFLINE_LICENSE_FILE`)
6. Command-line flags

File	Windows	macOS	Linux
Machine	`%ProgramData%\CerebionRivet\config.yml`	`/Library/Application Support/CerebionRivet/config.yml`	`/etc/cerebion-rivet/config.yml`
User	`%APPDATA%\CerebionRivet\config.yml`	`~/Library/Application Support/CerebionRivet/config.yml`	`~/.config/cerebion-rivet/config`
Project	`.rivet.yml` in or above the scanned folder		

Example `.rivet.yml`:

```yaml
code:
  mode: pqc                  # all | pqc
  exclude_dirs: [tests, vendor, "*.egg-info"]
  include_extensions: [py, js, java]
  timeout_seconds: 600
  fail_on: any               # any | none  (none = always exit 0, like --no-fail)
  online_config_rules: true # false = do not scan configuration files with OpenGrep registry r
binary:
  depth: deep                # quick | standard | deep
  max_memory_mb: 4096
network:
  ports: [443, 8443]
  targets_file: hosts.txt    # used by `rivet scan-network` when no host is given
certificate:
  port: 443
  targets_file: domains.txt # used by `rivet scan-cert` when no domain is given
```

The settings that take effect, and where:

Section	Settings	Used by
`code`	`mode`, `include_extensions`, `exclude_dirs`, `timeout_seconds`, `fail_on`, `online_config_rules`	`rivet scan` and the desktop app (the desktop app has no screen for `mode`, `timeout_seconds`, `fail_on` or `online_config_rules`; set them in a configuration file)
`binary`	`depth`, `max_binary_size_mb`, `analysis_timeout_seconds`, `max_memory_mb`, `max_concurrent_scans`, `max_archive_size_mb`, `max_files_per_archive`	`rivet scan-binary` and the desktop app (`depth` is command line only)
`network`	`ports`, `max_bulk_targets`, `targets_file`	`rivet scan-network` and the desktop app (`targets_file` is command line only)
`certificate`	`port`, `targets_file`	`rivet scan-cert`

Configuration files and the rule registry. Configuration files such as `web.config`, `nginx.conf`, `Dockerfile.prod` and `.env` are scanned with rule sets that OpenGrep downloads from its public registry (semgrep.dev), so that pass needs network access. Before it runs, Rivet checks for a connection (a few seconds at most). If the registry cannot be reached, the pass is skipped and the scan reports "online config rules unavailable" as an error and `rivet scan` exits 3 (incomplete, see Exit Codes) unless it also found something (exit 1), so the scan is never reported as clean when those files were not looked at. If nothing may be sent to the registry from your environment, set `code.online_config_rules` to `false` (or pass `--no-online-config-rules`). Rivet then makes

no connection, skips those files, and prints a note saying so; this is not an error. The pass stops after at most 60 seconds, whatever `timeout_seconds` is.

`rivet config show` also lists other sections. They are reserved for later releases and have no effect yet.

Commands:

Command	What it does
`rivet config show`	Shows every setting in effect and which file, variable or flag each value comes from
`rivet config path`	Shows where the machine, user and project files are on this computer
`rivet config validate [FILE]`	Checks a file (or every file that applies) for mistakes
`rivet config set KEY VALUE [--scope user|project|machine]`	Changes a setting, for example `rivet config set code.exclude_dirs "[node_modules]"`
`rivet config unset KEY [--scope ...]`	Removes a setting so the next layer applies
`rivet scan PATH --explain`	Shows the settings in effect and exactly which files would be scanned, which are left out and why. Does not scan

Notes:

- Exclusions apply to folders **inside** the scanned folder. A folder above it never excludes the scan.
- Folder names in the file types and exclusions lists may contain letters, digits and the characters `. _ - *` (for example `*.egg-info`). Entries with other characters, such as spaces, are ignored with a warning.
- The desktop app uses these files too. What you choose in **Settings** (Code Analysis, Network Analysis, Binary Analysis) is saved to your user file, so the command line sees it, and a `.rivet.yml` in the folder you scan applies to the desktop app's code scans as well.
- Flags win over the files: `rivet scan-network --ports 22,443` uses those ports whatever the files say, unless an administrator locked `network.ports`, in which case the command stops with `RIV-CFG-004`.
- Binary limits and network ports you saved in an earlier version of the desktop app keep applying. Rivet reads them as your own settings, and anything you save now goes to your user file.
- If a configuration file cannot be read, scans stop with a message that names the file and the code `RIV-CFG-003`; `rivet config validate` shows the mistake.
- Unknown settings are ignored with a warning, so a file written for a newer Rivet still loads.
- `rivet config set` rewrites the whole file, so comments you added by hand are not kept.
- **Locked settings.** An administrator can list settings under `locked:` in the machine file. Locked settings cannot be changed by your files, environment variables or flags; a flag that conflicts with one stops the scan with an error (`RIV-CFG-004`). Only the machine file can lock settings, so a `.rivet.yml` in a repository cannot weaken an enforced policy. Because a project file is part of the code being scanned, use the machine file (or, later, the management console) for settings your organization must enforce.

- When you contact support, `rivet doctor` and `rivet diagnose` include the effective configuration.

License

Setting	Description
License Key	Your Cerebion Rivet license key
Device Fingerprint	Hardware fingerprint for offline license generation
Offline License File	Path to air-gapped license file

Code Analysis

Setting	Description
File Extensions	Which file types to include in scans (default: every supported type)
Excluded Directories	Folders to skip (default: `node_modules`, `.git`, `venv`, `dist`, `build` and other dependency, build and cache folders)

These are saved to your user configuration file, so the command line uses the same choices. Once you change one, a **Reset to defaults** button appears under it. A setting your administrator has locked is shown as *Managed by your administrator* and cannot be edited.

Binary Analysis

Setting	Default	Range	Description
Max binary size	500 MB	1 to 5000	Largest binary accepted
Analysis timeout	600 s	30 to 3600	Per-scan time limit
Max memory	2048 MB	512 to 8192	Memory limit per scan
Max concurrent scans	3	1 to 10	Parallel scan limit
Max archive size	500 MB	1 to 2000	Largest ZIP or ISO archive accepted
Max files per archive	100	1 to 1000	Files examined inside one archive

These are shared with `rivet scan-binary` (see Configuration Files above). A limit your administrator locked is shown as *Managed by your administrator* and cannot be edited.

AI Configuration

Setting	Description
LLM Provider	Cloud (default) or custom/on-premise endpoint (Enterprise air-gapped)

Setting	Description
API Key	Your AI provider API key. It is kept in your operating system's credential store (Windows Credential Manager, macOS Keychain or the Linux Secret Service), or, where there is none, in a file only your user can read. It is not kept in the app's settings file and is never shown again after saving; leave the field blank to keep using the saved key, or choose **Remove it** to delete it. For the command line, CI and MCP, set the `RIVET_AI_API_KEY` environment variable instead.
Model	Model to use for fix generation

Network Analysis

Setting	Default	Description
Default scan ports	443, 465, 587, 636, 993, 995, 5671, 8443	Ports scanned when a scan does not name its own (1 to 65535, at most 100)
Max bulk targets	50	Hosts allowed in one bulk scan (1 to 500)

These are shared with `rivet scan-network`, which uses the same default ports. Locked settings are shown as *Managed by your administrator*.

Key Management

Connections are managed from the Key Management page itself rather than Settings — see Section 10 for adding, testing, and scanning a connection.

Application

Setting	Default	Description
Log Level	Info	Controls verbosity of application logs written to `app.log`. **Debug** — detailed diagnostics; **Info** — normal operation (default); **Warning** — warnings and errors only; **Error** — errors only. Changes apply immediately and persist across restarts.

---

17. Troubleshooting

Check Your Installation and Collect Diagnostics

Rivet includes two commands that help you and support find problems quickly. Both run from any terminal (the installer adds `rivet` to your PATH), work even when your license is not valid, and

change nothing on your system.

- `rivet doctor` checks the installation: the data, database, log and license locations, the scanner (including a quick test scan) and its rules, your license state, whether the license server is reachable, and whether your operating system offers a secure credential store. Each problem is shown with a code such as `RIV-LIC-004` and a suggested next step. Add `--offline` on an air-gapped machine to skip network checks, or `--format json` for scripts.
- `rivet diagnose -o support.zip` creates a file for support containing the doctor report, version and build details, recent logs, and the audit log tail. Passwords, tokens and keys are masked automatically, and your license cache and credential files are never included. Look it over, then email it to support@cerebion.com. It works on air-gapped machines.

When you contact support, quote any error code (`RIV-...`) shown in the message.

Application Won't Start

- Verify your system meets the minimum requirements (64-bit OS, 4 GB RAM).
- On macOS: if blocked by Gatekeeper, go to **System Preferences → Security & Privacy** and click **Open Anyway**.
- On Linux: after extracting the archive, ensure the launcher is executable: `chmod +x CerebionRivet`.
- Check the application log at (`rivet doctor` prints the exact location on your machine):
  - Windows: `%APPDATA%\CerebionRivet\logs\app.log`
  - macOS: `~/Library/Application Support/CerebionRivet/logs/app.log`
  - Linux: `~/.local/share/CerebionRivet/logs/app.log`

License Activation Fails

- Verify you have internet access and that `license.cerebion.com` is reachable.
- Check that your license key was entered correctly (no extra spaces).
- If your license is already active on another machine, click **Detach License** there first, or contact support@cerebion.com.

Code Scan Returns No Results

- Verify the selected folder contains files with supported extensions.
- Check **Settings → Code Analysis → File Extensions** — ensure the relevant languages are enabled.
- Check **Settings → Code Analysis → Excluded Directories**, and look for a `.rivet.yml` in the folder or one of its parents: either can leave folders out. `rivet scan PATH --explain` lists what would be scanned and what is skipped, and why.
- Try scanning a single file first to confirm the scanner is working.
- Ensure the bundled rules are present in the installation directory. On a clean install, rules are included automatically. If rules appear missing, reinstall the application.

Binary Scan Returns No Findings

- Confirm the file is a supported format (PE, ELF, Mach-O, Java .class/JAR).
- Increase **Analysis Depth** to **Deep** for a more thorough scan.

- Stripped or heavily packed binaries may return fewer findings — see the Stripped and Packed Binaries section.
- Check file size is within limits (**Settings → Binary Analysis → Max file size**).

Certificate Scan Fails

- Verify the host is reachable and has a valid TLS certificate on the specified port.
- For certificate file uploads, ensure the file is in PEM, DER, or P7B format.
- Check that port 443 (or your specified port) is not blocked by a firewall.

Key Management Connection Fails

- Click **Test Connection** before saving to isolate whether the problem is credentials or network reachability.
- Confirm the credential's attached policy/role grants at least read access to the keys or certificates you expect to see — Rivet only ever requests read-only access, so an overly narrow policy on your end is the most common cause.
- For on-premise backends (Vault, AD CS, Kubernetes Certificate Manager), confirm the endpoint is reachable from the machine running Rivet, not just from your browser or another host.

AI Fix Generation Fails

- Verify your Google API key is configured in **Settings → AI Configuration**.
- Ensure the file is inside a Git repository (required for Apply and Rollback).
- If the fix was generated for an older version of the file, click **Re-validate** to regenerate with the current content.

Log Locations

Platform	Log path
Windows	`%APPDATA%\CerebionRivet\logs\main.log`
macOS	`~/Library/Logs/CerebionRivet/main.log`
Linux	`~/.config/CerebionRivet/logs/main.log`
Backend (Windows)	`%APPDATA%\CerebionRivet\logs\app.log`
Backend (macOS)	`~/Library/Application Support/CerebionRivet/logs/app.log`
Backend (Linux)	`~/.local/share/CerebionRivet/logs/app.log`
Backend errors	Same directory, `error.log`

---

18. FAQ

When will quantum computers break RSA? Current estimates range from 10–20 years for cryptographically relevant quantum computers. However, harvest-now-decrypt-later attacks mean sensitive data encrypted today is already at risk.

Does Rivet upload my files to the cloud? No. All analysis runs locally on your machine. No files, binaries, or source code are ever sent to Cerebion servers.

Does Rivet store my Key Management Analyzer credentials? Your credential is stored only in your own machine's operating system credential manager (Windows Credential Manager, macOS Keychain, or Linux Secret Service) — never in Rivet's database, and never sent to Cerebion. Rivet's connector code has no ability to decrypt, sign, or export anything with it; it can only read key and certificate metadata. You can revoke the credential at any time from your own infrastructure.

Does a clean Binary Analyzer result mean my binary is quantum-safe? No. A clean result means no known vulnerable cryptographic patterns were detected using static analysis. Rivet cannot detect runtime-only crypto, custom implementations that do not match known patterns, or algorithms hidden inside encrypted/packed binary sections.

What is the difference between Desktop and Server editions? Desktop is a single-seat license for individual use. Server supports unlimited seats and includes CI/CD integrations, air-gap support, and priority support.

Can I use Rivet in an air-gapped environment? Yes. Rivet supports fully offline operation with Ed25519-signed offline license files. No internet connection is required after activation. All analysis rules are bundled with the application.

What NIST PQC algorithms does Rivet recommend? ML-KEM (CRYSTALS-Kyber) for key encapsulation — NIST FIPS 203. ML-DSA (CRYSTALS-Dilithium) for digital signatures — NIST FIPS 204. SLH-DSA (SPHINCS+) as a stateless hash-based signature alternative — NIST FIPS 205.

How do I move my license to a new machine? Go to **Settings → License → Deactivate License** on the old machine. Then activate on the new machine with the same key. If you no longer have access to the old machine, contact support@cerebion.com.

How do I cancel my subscription? Cancel anytime from your account at cerebion.com/portal or by contacting support. Your license remains active until the end of the billing period.

Contact & Support

- **Documentation:** cerebion.com/docs
- **Support:** support@cerebion.com
- **License portal:** cerebion.com/portal
- **Security disclosures:** security@cerebion.com

Cerebion Rivet is © Cerebion. All rights reserved. See cerebion.com/terms for license terms.