Eburon AI Model Confidentiality Playbook

Purpose This document defines how **Eburon AI** protects proprietary model IP—**architecture, weights, and deployment know-how**—across the full lifecycle: build → store → ship → run → update → revoke.

Audience Product, Security, Platform, ML Engineering, DevOps, Compliance.

Scope Confidentiality mechanisms for:

- Model artifacts (weights, adapters, quantized variants)
- Runtime assets (prompts, system policies, safety layers)
- Distribution pipeline (downloads, updates, caches)
- Edge execution (desktop/mobile)
- Hybrid inference (edge + cloud)

Non-Goals

- This is not a detailed training recipe.
- This does not provide offensive techniques; it focuses on defensive controls.

---

1) Executive Summary

Eburon AI model confidentiality is the set of controls that prevents unauthorized access, copying, reverse engineering, or long-term misuse of proprietary model assets.

We implement a **defense-in-depth** strategy:

1. **Cryptography + key management** (at rest, in transit, and on device)
2. **Moving Target Defense (MTD)** and artifact obfuscation (weights are unusable without reconstruction)
3. **Secure execution** (hardware enclaves / trusted execution environments where possible)
4. **Hybrid inference** and **model decomposition** (ship only what's safe to the edge)
5. **Licensing + anti-fraud** (time-bound entitlements, device binding, and revocation)

Core reality: If the *entire* best model is shipped to an untrusted device, assume a skilled attacker may eventually extract it. The winning strategy is:

- **On-device = small/medium tier** (fast, cheap, offline)
- **Cloud = heavy/premium tier** (protected, scalable)
- **Licensing** gates both

---

2) Threat Model (What We’re Defending Against)

2.1 Primary Threats

- **Model theft:** copying weights/adapters from disk or update cache.
- **Reverse engineering:** analyzing weights, architecture, or loaders to recreate IP.
- **Unauthorized use:** sharing subscriptions, bypassing licensing, running on many devices.
- **Supply chain tampering:** malicious update channels or poisoned downloads.
- **Runtime memory scraping:** extracting weights while loaded for inference.

2.2 Attack Surfaces

- Download endpoints & CDNs
- Local storage (app cache, filesystem, backups)
- Update packages
- Runtime process memory
- Debug tooling hooks
- Compromised OS / rooted devices

2.3 Security Goals

- **Confidentiality:** unauthorized parties cannot access usable weights.
- **Integrity:** artifacts are authentic, untampered.
- **Availability:** licensing and renewals degrade gracefully.
- **Auditability:** we can explain and prove access decisions.

---

3) Design Principles

1. **Least exposure:** ship only what must be on device.
2. **Short-lived access:** temporary keys and renewable entitlements.
3. **Device binding:** artifacts only usable on the authorized device.
4. **Tamper-evident packaging:** detect manipulation early.
5. **Observable + revocable:** strong telemetry and quick kill-switch.
6. **User trust:** fair offline window; clear messaging; no deceptive behavior.

---

4) Lifecycle Overview (Build → Run)

4.1 Artifact Types

- **Base weights** (full model)
- **Adapters / LoRA** (incremental proprietary improvements)
- **Quantized variants** (mobile/CPU)
- **Tokenizer assets**
- **Runtime policy packs** (system + safety + guardrails)

4.2 High-Level Flow

1. **Build:** produce signed artifacts (and optional MTD transforms)
2. **Store:** encrypted in object storage; strict access controls
3. **Distribute:** authenticated download with ephemeral keys
4. **Install:** verify signature; re-encrypt device-bound
5. **Run:** decrypt just-in-time; prefer secure enclaves
6. **Update:** rotate keys; revoke old entitlements
7. **Revoke:** disable token; delete/lock local artifacts

---

5) Multi-Layered Encryption & Key Management

5.1 Encryption at Rest (Backend)

Goal: even if storage is exposed, the model is unusable.

- Store artifacts encrypted (e.g., envelope encryption):
- **DEK** (data encryption key) per artifact
- **KEK** (key encryption key) protected by KMS/HSM
- Separate buckets by sensitivity tier: `public`, `licensed`, `restricted`.

5.2 Encryption in Transit

- TLS everywhere; pin or validate cert chains for update endpoints.
- Signed download manifests to prevent downgrade attacks.

5.3 Ephemeral Keys (Per Download / Per Session)

Goal: minimize blast radius.

- Generate an ephemeral key for each download session.
- Key expires quickly (minutes).
- Key is never stored long-term on the client.

5.4 Application-Level Encryption

Goal: encryption close to the source.

- Artifact is encrypted **before** it reaches CDNs.
- Client decrypts only after:
- authentication ✅
- license entitlement ✅
- device attestation (when available) ✅

5.5 Re-Encryption On Device (Device-Bound)

Goal: copying files to another machine is useless.

- After download + verification, re-encrypt with a **device-unique key** stored in:
- Secure Enclave / Keychain (Apple)
- Keystore (Android)
- TPM / OS key store (desktop)
- Optional: bind also to user profile + app install ID.

5.6 Key Rotation Policy

- Rotate KEKs on a schedule (e.g., quarterly) and immediately after incidents.
- Rotate per-artifact DEKs on major model updates.
- Version keys; support rolling migration.

---

6) Moving Target Defense (MTD) & Obfuscation

6.1 Why MTD

Full encryption of multi-GB models can add noticeable latency. MTD aims to:

- Keep artifacts unusable without secret reconstruction
- Reduce download/install overhead
- Make static theft less valuable

6.2 MTD Concept

Instead of shipping usable weights:

- Transform weights into an **obfuscated representation**.
- Keep a **reconstruction map** (or seed + map) that is delivered only after licensing checks.

Outcome: the file looks like a model artifact but cannot be executed without reassembly.

6.3 Implementation Patterns

- **Weight sharding + permutation:** split weights into chunks and reorder.
- **Seeded transforms:** reversible transforms keyed by a secret seed.
- **Architecture/weights separation:** keep the structure and weights mapping decoupled.

6.4 Reconstruction Map Handling

- Treat reconstruction map as a high-sensitivity secret.
- Deliver map via:
- short-lived, device-bound token
- optionally inside an enclave session

- Cache map only in protected storage and only for the offline window.

6.5 Code Obfuscation (Defense Support)

Goal: slow down analysis of model loaders and key flows.

- Obfuscate client code that:
- handles key derivation
- performs reconstruction
- verifies signatures
- Add integrity checks so modified binaries lose access.

> Note: Obfuscation is not a primary security control; it's a time-delay layer.

---

7) Secure Execution Environments

7.1 The Core Problem

Even if a model is encrypted on disk, it must be **usable in memory** during inference. That's where advanced attackers target.

7.2 Trusted Execution Environments (TEEs)

Goal: isolate sensitive operations.

- Keep decryption + reconstruction inside a protected environment.
- Restrict read access even if the OS is compromised.

7.3 Memory Protection & Anti-Scrape Hardening

- Just-in-time decrypt pages; avoid long-lived plaintext buffers.
- Zeroize memory after use.
- Use runtime checks to detect debugging/tampering and restrict access.

7.4 Attestation (When Available)

- Validate device/app state before issuing reconstruction secrets.
- Use attestation results to increase friction on compromised devices.

---

8) Hybrid Inference & Model Decomposition

8.1 Goal

Ship *some* capability to the edge without shipping the whole crown-jewel.

8.2 Decomposition Patterns

- **Front-half on device, back-half in cloud** (or vice-versa)
- **On-device small model** for everyday tasks, **cloud heavy** for premium tasks
- **Specialized head** on device; shared trunk in cloud

8.3 “Safe” Decomposition Property

- The portion on the device should reveal minimal information about:
- full weights
- proprietary training
- premium capabilities

8.4 Practical Policy for Eburon AI

- Put **offline-friendly + low risk** models on device.
- Keep **high value** weights on the server.
- Gate cloud tier with stronger identity checks and billing.

---

9) Licensing & Anti-Fraud (Netflix Model)

9.1 License Pass (Signed Token)

Rule: model runs only with a valid, signed token.

- Token contains entitlements:
- tier (local / hybrid / premium)
- max devices
- offline window
- allowed model versions
- usage limits (optional)
- Token expires (e.g., 24 hours / 7 days) and must renew.

9.2 Device-Bound Activation

- On first activation, bind token to:
- device key (secure store)
- app install ID
- user account ID
- Prevent file copying from enabling another machine.

9.3 Graceful Offline Window

- Clear offline policy:
- works offline up to **X days**
- then requires a quick check-in
- If offline window expires:

- degrade to limited features, or pause local inference until renewal

9.4 Risk Scoring & Enforcement

Monitor for suspicious patterns:

- frequent device ID changes
- repeated reinstalls
- abnormal download frequency
- token replay attempts

Actions:

- step-up verification
- limit renewals
- require online verification
- revoke entitlement
- lock local artifacts (re-encrypt with new key) or request deletion

9.5 Revocation & Kill Switch

- Token revocation list (server-side)
- “Deny renewals” immediately for flagged accounts
- Force update policy for security incidents

---

10) Artifact Authenticity & Supply Chain Security

10.1 Signed Manifests

- Each artifact has:
- hash
- version
- minimum client version
- signature

10.2 Update Channel Hardening

- Prevent downgrade to vulnerable clients.
- Separate stable vs. canary.
- Rollback plan with integrity.

10.3 Secure Logging (No IP Leakage)

- Avoid logging:
- raw keys
- reconstruction maps
- decrypted weight fragments

- Use redaction and structured events.

---

11) Operational Controls

11.1 Access Control (Internal)

- Strict RBAC for:
- artifact storage
- KMS/HSM operations
- release approvals

11.2 Monitoring

Alert on:

- unusual artifact downloads
- repeated failed decrypt attempts
- mass token renewals from new devices
- signature failures

11.3 Incident Response

Runbook:

1. Freeze token renewals for affected model versions
2. Rotate keys / invalidate reconstruction seeds
3. Force update clients
4. Publish advisory + internal postmortem

---

12) Performance, UX, and Tradeoffs

12.1 Security vs. Speed

- Full encryption is strongest but may increase install latency.
- MTD can reduce overhead but increases engineering complexity.

12.2 UX Guardrails

- Keep licensing messaging simple:
- “Offline available until: DATE”
- “Quick check-in needed to continue”
- Never fake connectivity or hide enforcement.

---

13) Recommended Default Policy (Eburon AI Baseline)

13.1 Tiers

- **Edge Lite:** on-device small model + offline up to 3 days
- **Hybrid Pro:** on-device medium + cloud heavy for premium tasks
- **Cloud Ultra:** cloud only; strict usage metering

13.2 Token Defaults

- Expiry: 24 hours (renew when online)
- Offline window: 72 hours
- Max devices: 1–2 (plan dependent)
- Model versions: allow current + previous for rollback

13.3 Key Policy

- Ephemeral download keys: expire in minutes
- Device key: non-exportable where possible
- Rotate artifact keys on major updates

---

14) Implementation Checklist

Client (Edge)

-

Server (Control Plane)

-

Release & Ops

-

---

15) Appendix: Reference Architecture (Text Diagram)

```
                  ┌───────────────────────────────┐
                  │        Eburon Control Plane     │
                  │  Auth • Entitlements • Billing  │
                  │  License Tokens • Revocation    │
                  └───────────────────────────────┘
                                  │
                                  │ signed token
                                  │ + ephemeral keys
```

```
                                ▼
┌─────────────────────────────────────────────────────────┐
│                    Client Device (Edge)                 │
│  1) Login → receive License Pass (expires)              │
│  2) Download encrypted/MTD artifact                      │
│  3) Verify signature + hash                              │
│  4) Re-encrypt device-bound                              │
│  5) JIT decrypt + run (prefer enclave)                   │
│  6) Offline window (X days) → renew token when online    │
└─────────────────────────────────────────────────────────┘

(Optional)
    Hybrid Heavy Tier: Edge calls cloud inference for premium workloads
```

16) Appendix: Glossary

- **Weights:** learned parameters; primary IP asset.
- **Architecture:** model structure; also sensitive.
- **DEK/KEK:** data encryption key / key encryption key.
- **MTD:** moving target defense; makes artifacts unusable without reconstruction.
- **TEE/Enclave:** isolated execution environment.
- **License pass:** signed entitlement token with expiry.