FEDRAMP / FISMA
COMPLIANCE DOCUMENTATION

NIST SP 800-53 Rev. 5

Control Mapping

System Security Plan (SSP) — Technical Controls
LockFlare Engine — In-Memory Application Platform

FedRAMP Impact Level: HIGH

lockflare.com

April 2026

1. Executive Summary

This document maps LockFlare Engine's security architecture to the NIST Special Publication 800-53 Revision 5 control framework, as required for Federal Risk and Authorization Management Program (FedRAMP) authorization and Federal Information Security Modernization Act (FISMA) compliance.

LockFlare Engine is an in-memory application development and runtime platform where all application source code, credentials, and environment variables are encrypted at rest using AES-256-CBC with four independent key domains, and decrypted exclusively in volatile RAM during execution. No source code files exist on the host filesystem at any point during the system lifecycle.

The platform includes Appliance Mode (codename Enclave) — an OS-level server sealing mechanism that disables all interactive login methods (SSH, console, KVM/IPMI) while maintaining full application serving capability. This achieves hardware-enclave-level isolation entirely in software, without specialized hardware.

This architecture inherently satisfies or exceeds the requirements of multiple NIST 800-53 control families, particularly those related to media protection (MP), system and communications protection (SC), access control (AC), physical and environmental protection (PE), audit and accountability (AU), and configuration management (CM).

2. System Description

2.1 System Boundary

- System Name: LockFlare Engine
- Impact Level: HIGH (Confidentiality: HIGH, Integrity: HIGH, Availability: HIGH)
- Deployment Model: On-premise, government-authorized cloud (GovCloud), Azure VM, Amazon EC2, or any Linux environment
- Data Classification: Supports CUI, FOUO, and classified-adjacent workloads (with appropriate overlay)

2.2 System Components

LockFlare Studio: Browser-based IDE for application development. Communicates with the API layer over TLS 1.3. Developers write native Vue, React, Svelte, and Node.js code — no proprietary syntax or custom abstractions.

LockFlare Interpreter: In-memory runtime process. Reads encrypted components from the code repository, decrypts in RAM, compiles, and serves applications. No filesystem artifacts. Runs on bare metal, virtual machines, or any Linux server.

Encrypted Code Repository: All application code stored as AES-256-CBC ciphertext. Supports LockFlare-hosted repository or self-hosted (your own infrastructure) for full data sovereignty.

Encryption Keys: Four independent AES-256-CBC key pairs stored exclusively as server environment variables. Never persisted to disk, config files, or source control.

Appliance Mode (Enclave): OS-level server sealing mechanism. Disables all interactive login methods (SSH, console, KVM/IPMI, hosting panel consoles) while maintaining full application serving. Controlled exclusively from the LockFlare platform with two-factor authentication. Sealed state survives reboot.

2.3 Infrastructure Flexibility

The Interpreter runs on any Linux environment with zero disk I/O in the request path. This eliminates the need for high-IOPS storage tiers on cloud VMs — no SSD throughput charges, no provisioned IOPS, no Ultra Disk premiums. The cheapest storage tier is sufficient, reducing cloud infrastructure costs by 30–50%.

2.4 Native Code & Zero Lock-in

All application code is standard production code. Developers write native Vue, React, Svelte, and Node.js — the same code they would write in any environment. Code can be exported at any time as a standalone Node.js project that runs on any standard server with npm install && npm start. Zero LockFlare dependency after export.

3. Access Control (AC)

LockFlare implements a 3-tier access control model that enforces least-privilege at the global, project, and environment levels. Appliance Mode extends access control to the operating system level.

Control	Name	LockFlare Implementation	Status
AC-2	**Account Management**	Centralized user management with role-based access. Admins create/edit/delete users. Each user has explicit permissions assigned per capability group. Users can be deactivated without deletion. Invitation-based onboarding.	**Implemented**
AC-2(1)	**Automated Management**	Token expiration is embedded in encrypted payload. Expired tokens are rejected automatically. No manual session cleanup required.	**Implemented**
AC-3	**Access Enforcement**	3-tier enforcement: Global permissions (platform operations) → Project permissions (UX/Backend: none/view/edit/create) → Environment permissions (view/edit/pushTo/pushFrom/exportSource). All tiers evaluated on every request. Appliance Mode extends enforcement to the OS level — all interactive login methods disabled when sealed.	**Implemented**
AC-4	**Information Flow**	Code flows: Repository (encrypted) → API (decrypt in RAM) → Browser (TLS). No intermediate files. Environment isolation prevents code flow between Dev/Staging/Production without explicit push permissions.	**Implemented**
AC-6	**Least Privilege**	Users receive minimum permissions needed. Backend developers cannot view frontend code (and vice versa). Environment-level restrictions prevent junior staff from accessing production. Appliance Mode operations require fullAdmin permission plus 2FA verification.	**Implemented**
AC-7	**Unsuccessful Logon**	Failed login attempts tracked per IP with progressive lockout. Account lockout after configurable threshold.	**Implemented**
AC-8	**System Use Notice**	Login page displays configurable system use notification and consent banner.	**Implemented**
AC-10	**Concurrent Sessions**	Configurable single-session enforcement. New login rotates internal user key, invalidating all previous tokens.	**Implemented**

AC-11	**Session Lock**	Token expiration enforces automatic session termination. Configurable duration per account. Inactivity timeout supported.	**Implemented**
AC-12	**Session Termination**	Tokens are stateless and encrypted. Termination is automatic via expiration. No server-side session to orphan.	**Implemented**
AC-17	**Remote Access**	All access is remote via TLS 1.3. IP binding ensures tokens are valid only from the originating IP address. CIDR whitelisting available per server. Appliance Mode disables all remote OS access (SSH, console) while maintaining application-layer access.	**Implemented**

4. Audit & Accountability (AU)

Control	Name	LockFlare Implementation	Status
AU-2	**Event Logging**	Platform logs: user logins, failed attempts, component edits, deploys, environment pushes, AI generations, snapshot operations, Appliance Mode seal/unseal events with actor identity and 2FA method.	**Implemented**
AU-3	**Content of Records**	Every log entry includes: timestamp (UTC), user ID, IP address, action type, target resource, and result. Appliance Mode events additionally log server IP, seal state transition, and 2FA verification method.	**Implemented**
AU-6	**Review/Analysis**	Fleet monitoring dashboard provides real-time audit visibility across all interpreter nodes. Per-server health, memory, CPU, and request metrics. Centralized deployment logs.	**Implemented**
AU-9	**Protection of Info**	Audit logs stored in the encrypted code repository. Same AES-256-CBC protection as application code. Self-hosted option ensures logs remain within organizational boundaries.	**Implemented**
AU-12	**Audit Generation**	All CRUD operations generate audit events. Version history maintains 50 states per component per environment. Project snapshots capture point-in-time state.	**Implemented**

5. Identification & Authentication (IA)

Control	Name	LockFlare Implementation	Status
IA-2	**User ID & Auth**	Username/password with mandatory 2FA. Three 2FA methods: WebAuthn (hardware keys), TOTP (authenticator apps), email PIN. Users without 2FA restricted to account setup only.	**Implemented**
IA-2(1)	**MFA to Privileged**	All administrative actions require active 2FA. Appliance Mode seal/unseal requires explicit 2FA challenge — a stolen session token cannot toggle server state.	**Implemented**
IA-2(2)	**MFA to Non-Privileged**	All users — regardless of privilege level — must complete 2FA to access the platform.	**Implemented**

IA-4	**Identifier Mgmt**	Unique user IDs generated per account. Email-based identity with invitation-only registration. No self-registration.	**Implemented**
IA-5	**Authenticator Mgmt**	Passwords hashed with bcrypt. Tokens encrypted with AES-256-CBC. Token rotation on new login. IP binding prevents token reuse from different addresses.	**Implemented**
IA-8	**Non-Org Users**	Per-environment access controls prevent external collaborators from viewing or modifying production environments. Export permissions controlled independently.	**Implemented**

6. System & Communications Protection (SC)

Control	Name	LockFlare Implementation	Status
SC-4	**Shared Resources**	Each domain runs in its own V8 isolate with independent memory limits. No shared memory between application domains. Worker process isolation prevents cross-contamination.	**Implemented**
SC-7	**Boundary Protection**	Application boundary: WAF middleware inspects all requests (SQLi, XSS, CMDI, path traversal, prototype pollution). OS boundary: Appliance Mode disables all interactive access methods, creating a sealed execution perimeter. The server accepts only application traffic — no management plane access.	**Implemented**
SC-8	**Transmission Confidentiality**	TLS 1.3 for all communications. Optional AES-256-CBC transport encryption between frontend and backend (4 configurable modes). Encrypted payloads between platform and interpreter.	**Implemented**
SC-12	**Key Establishment**	Four independent AES-256-CBC key domains. Keys stored exclusively as environment variables. Appliance Mode seal encryption uses a key derived from the server license via SHA-256, ensuring each server has a unique encryption context.	**Implemented**
SC-13	**Cryptographic Protection**	AES-256-CBC with independent IVs per domain. HMAC-SHA256 for mutual authentication between platform and interpreter. Constant-time signature comparison prevents timing attacks.	**Implemented**
SC-28	**Protection at Rest**	All source code, credentials, and environment variables encrypted at rest using AES-256-CBC. Four independent key domains limit blast radius of any single key compromise. Self-hosted repository option ensures ciphertext remains within organizational boundaries.	**Implemented**

7. Media Protection (MP)

Control	Name	LockFlare Implementation	Status
MP-2	**Media Access**	No application code exists on any media (disk, SSD, NVMe). Code is decrypted exclusively in volatile RAM.	**Implemented**

		Disk cloning, imaging, or physical removal yields no application data.	
MP-4	**Media Storage**	The filesystem contains only the encrypted engine binary and encrypted interpreter payload. No readable application files exist in storage at any point in the system lifecycle.	**Implemented**
MP-5	**Media Transport**	Code transport is encrypted end-to-end: AES-256-CBC from repository to interpreter via TLS 1.3. Decryption occurs only in RAM. No intermediate plaintext files.	**Implemented**
MP-6	**Media Sanitization**	No media sanitization required for application code — it was never written to persistent media. Server decommissioning requires only OS reinstallation. Appliance Mode ensures no unauthorized access to the sealed system during the decommissioning process.	**Implemented**

8. Physical & Environmental Protection (PE)

Appliance Mode provides software-enforced physical access controls that complement traditional facility security.

Control	Name	LockFlare Implementation	Status
PE-2	**Physical Access Auth**	Appliance Mode disables physical console login. Even with direct hardware access (KVM, IPMI, keyboard/monitor), authentication is rejected. Physical access to the machine yields a sealed appliance with no interactive login capability.	**Implemented**
PE-3	**Physical Access Control**	The server is sealed at the OS level. Physical access, including console access via hosting provider panels, keyboard/monitor access, and rescue mode access, does not grant any ability to read, copy, or modify application code or configuration. The sealed state survives reboot.	**Implemented**
PE-5	**Access to Output Devices**	Console output on a sealed server shows only the boot sequence and LockFlare Engine status. No interactive shell is available. No application data is accessible through output devices.	**Implemented**

9. Configuration Management (CM)

Control	Name	LockFlare Implementation	Status
CM-2	**Baseline Config**	Application configuration stored as encrypted environment variables per project per environment. Changes require platform-level authentication and appropriate permissions. Appliance Mode prevents local configuration changes on sealed servers.	**Implemented**
CM-3	**Change Control**	All code changes tracked via 50-version history per component. Project snapshots provide point-in-time recovery. Compare & Push validates changes before promotion between environments.	**Implemented**
CM-5	**Access Restrictions**	Code modifications restricted to authorized users via 3-tier permissions. No filesystem access to application code —	**Implemented**

		changes are only possible through the authenticated platform interface. Appliance Mode prevents OS-level configuration changes.	
CM-6	**Config Settings**	Environment variables are encrypted per-environment with independent AES-256-CBC keys. Configuration is immutable on the interpreter server — settings are pulled from the platform at boot and stored only in RAM.	**Implemented**
CM-7	**Least Functionality**	Appliance Mode disables unnecessary OS services for application serving: SSH daemon connections are blocked, console login is disabled, all interactive access methods are removed. The server runs only the LockFlare Engine and essential system services.	**Implemented**
CM-8	**Component Inventory**	Platform maintains complete inventory of all components, endpoints, environments, servers, and user permissions. Fleet monitoring provides real-time visibility into every interpreter node.	**Implemented**

10. System & Information Integrity (SI)

Control	Name	LockFlare Implementation	Status
SI-3	**Malicious Code**	No executable files on the filesystem. Application code exists only in RAM as compiled artifacts. Traditional malware, trojans, and webshells have no filesystem to target. The attack surface for malicious code injection is architecturally eliminated.	**Implemented**
SI-4	**Monitoring**	Built-in fleet monitoring: CPU, memory, uptime, request rates per server. Interpreter health-check process with cryptographic verification. WAF logs all blocked requests with classification (SQLi, XSS, CMDI, path traversal).	**Implemented**
SI-7	**Integrity Verification**	Health-check process periodically re-fetches encrypted bundle from the repository. If in-memory code differs from the master, it is overwritten immediately. Tampered code survives no longer than one check interval. Appliance Mode seal state is verified on every engine startup.	**Implemented**
SI-7(1)	**Integrity Checks**	Interpreter startup includes SHA-256 verification of the worker shim before execution. HMAC-SHA256 mutual authentication between platform and interpreter for all Appliance Mode operations. Nonce-based replay protection with 30-second TTL.	**Implemented**
SI-10	**Input Validation**	Express.js middleware validates all input. WAF inspects request bodies, query strings, headers, and paths. Document ID validation on all references. Permission checks on every API call. Encrypted tokens reject any modification.	**Implemented**
SI-16	**Memory Protection**	Worker process isolation (one per allocated core). V8 isolate sandboxing per domain with independent memory limits. Crash containment ensures single-process failure does not affect others. Automatic respawn in ~2 seconds.	**Implemented**

11. Contingency Planning (CP)

Control	Name	LockFlare Implementation	Status
CP-2	**Contingency Plan**	Architecture designed for zero-data-loss recovery. All code persists in the encrypted repository. Any interpreter node can be rebuilt from the repository in <60 seconds. Sealed servers can be replaced by provisioning a new server and installing the LockFlare Engine in under 2 minutes.	**Implemented**
CP-6	**Alternate Storage**	Repository supports multi-region replication. Self-hosted deployments support standard replica sets. Project snapshots provide point-in-time recovery independent of database backups.	**Implemented**
CP-7	**Alternate Processing**	Multi-server interpreter architecture. Server loss triggers automatic traffic redistribution. New nodes join the pool in <60 seconds by pulling the encrypted bundle from the repository. Works on any Linux environment — bare metal, Azure VM, Amazon EC2, or GCP.	**Implemented**
CP-9	**System Backup**	Auto-versioning saves every component state before modification (50 versions retained). Project snapshots capture full application state on demand. Source export generates standalone Node.js archives deployable outside LockFlare.	**Implemented**
CP-10	**Recovery**	Interpreter node recovery: connect to repository → decrypt in RAM → compile → serve. Total time: <60 seconds. Full project restoration from snapshot: delete + reinsert with automatic safety backup. Sealed servers: provision new hardware, install Engine (2 minutes), platform pushes entire application stack automatically.	**Implemented**

12. Supply Chain Risk Management (SA)

Control	Name	LockFlare Implementation	Status
SA-9	**External Services**	Application code never leaves the encrypted pipeline. Self-hosted code storage option eliminates reliance on third-party infrastructure for code persistence. Appliance Mode ensures code cannot be extracted from client infrastructure.	**Implemented**
SA-11	**Developer Testing**	Multi-environment architecture (Dev/Staging/Production) enables isolated testing. Live Preview provides real-time compilation without deployment. Compare & Push validates code before promotion.	**Implemented**
SA-12	**Supply Chain Protection**	No node_modules directory on interpreter servers. NPM packages are installed once on the host and bridged through controlled V8 isolate proxy. Application code has no direct filesystem access to resolve dependencies.	**Implemented**

13. Risk Assessment Summary

13.1 Threats Eliminated by Architecture

- **Ransomware** — No files on disk to encrypt. Attack vector eliminated entirely.
- **Source Code Exfiltration** — No files to copy. SSH access reveals no application code. Appliance Mode disables SSH entirely on sealed servers.
- **Local File Inclusion (LFI)** — No application files to include. Directory traversal has no targets.
- **Supply Chain (node_modules)** — No node_modules on the server. Application code has no require() calls resolved from disk.
- **Insider Threat (Priviledged Access Risk)** — In traditional application environments, privileged users (e.g., system administrators, infrastructure operators) typically have the ability to access application source code and configuration data through filesystem access, process inspection, or administrative interfaces.

  The LockFlare architecture is designed to reduce the exposure of application code to privileged insiders by removing persistent code artifacts from the system and restricting access pathways.

  Application code is not stored as plaintext on the host filesystem at any point in the system lifecycle. All code is maintained in encrypted form within the repository and is decrypted only in volatile memory during execution. Execution occurs within isolated V8 contexts, where application logic exists as transformed and compiled artifacts rather than directly readable source code.

  As a result, standard administrative access to the underlying operating system (including root-level access) does not provide direct access to application source code through conventional mechanisms such as file access or process-level inspection.

  Additional access controls are enforced at the platform level, including role-based permissions, environment isolation, and multi-factor authentication for all administrative actions. These controls limit the ability of users to access or modify application components outside of authorized workflows.
-
- In deployments where Appliance Mode (Enclave) is enabled, interactive system access (including SSH, console, and out-of-band management interfaces) is disabled. This further restricts the ability of privileged insiders to interact directly with the operating system, reducing the available attack surface for insider-driven code access.
-
- While no system can fully eliminate insider risk, the LockFlare design significantly reduces the likelihood of source code exposure through privileged system access by removing persistent storage vectors, limiting runtime visibility, and enforcing strict access controls at both the platform and system levels.
-
- Residual risk remains in scenarios involving compromise of the application platform, encryption key material, or authorized user accounts. These risks are addressed through layered controls including encryption, authentication, audit logging, and environment segmentation, consistent with NIST SP 800-53 control families for access control, system integrity, and audit accountability.

- **Physical Access** — Appliance Mode disables console login. Physical access to the hardware yields a sealed appliance with no interactive capability. Disk cloning yields only encrypted engine artifacts.
- **Session Hijacking** — Tokens are IP-bound. Stolen tokens fail from any other IP address.
- **Credential Stuffing** — Progressive lockout per IP. No user enumeration via login response.

13.2 Residual Risks

- **Repository Compromise + Key Theft** — An attacker cannot access encrypted source code through the LockFlare platform. All access is protected by IP-bound tokens, two-factor authentication, and 3-tier permissions. Even with valid credentials, login can be restricted to specific IP addresses and working hours. The only theoretical path to decryption is through direct access to the client's own database infrastructure, where encrypted ciphertext is stored, combined with theft of the AES-256 encryption keys from the server's environment variables — two independent systems that must both be compromised simultaneously. Even in this scenario, the attacker gains read access to source code but has no mechanism to modify or deploy code through LockFlare, as all write operations require authenticated platform access with full permission validation. Four independent key domains limit the blast radius of any single key compromise. Ultimately, the security of the encrypted repository is governed by the client's own database security practices — network access controls, authentication, and infrastructure hardening — which is within their operational responsibility.

- **Memory Dump Attack** — Memory Dump Attack - An attacker with root access over a non-sealed server could theoretically dump process memory. However, application code executes inside V8 isolates where the original source has already been heavily obfuscated (variable renaming, control flow flattening, string array encoding) and then compiled to bytecode via Bytenode before execution. A memory dump yields V8 bytecode derived from obfuscated source - not the original application logic. Reconstructing readable code from this requires defeating both the bytecode format and the obfuscation layer independently, significantly increases complexity of reconstruction. Additionally, the LockFlare Engine scans for tracing and debugging tools (strace, ltrace, gdb, perf, etc.) every 5 seconds - if any are detected, the Engine shuts down immediately, terminating all application processes and clearing memory. These tools must be fully uninstalled from the server before the LockFlare Engine will activate. Appliance Mode eliminates this vector entirely by preventing root access on sealed servers.

13.3 Compensating Controls

The in-memory architecture combined with Appliance Mode provides compensating controls that exceed the requirements of traditional file-based systems. The complete elimination of the filesystem attack surface, combined with OS-level access lockdown, compensates for any residual risk by removing both the most common exploit vector and the most common threat actor (privileged insiders) from the system's attack surface.

14. Document Control

Document Title	LockFlare Engine — FedRAMP/FISMA Compliance Documentation
Document Version	3.0
Date	April 2026
NIST Framework	SP 800-53 Revision 5
FedRAMP Impact Level	HIGH
Prepared By	LockFlare Engineering
Contact	engineering@lockflare.com

lockflare.com