OPENSSL 1.1.1 DEPRECATION AND UPGRADE GUIDE TO OPENSSL 3.X

Audience: security leads, SREs, platform engineers, compliance, and developers maintaining software linked against OpenSSL 1.1.1.

EXECUTIVE SUMMARY

OpenSSL 1.1.1 reached End of Life (EOL) on September 11, 2023. EOL means it no longer receives security updates or bug fixes from the OpenSSL project. Remaining on 1.1.1 exposes organizations to unpatched vulnerabilities, compliance findings, and vendor support issues. Migrate to OpenSSL 3.x—preferably 3.0 LTS for stability and FIPS options, or a newer supported 3.x release for features and performance.

WHAT EOL/DEPRECATION MEANS

- **No security fixes**: New vulnerabilities discovered after EOL will not be patched in 1.1.1.
- **No bug fixes**: Functional defects and regression issues remain unresolved.
- **No official support**: Upstream maintainers will direct you to upgrade; many vendors follow suit.
- **Compliance risk**: EOL crypto components commonly trigger audit and policy findings.

RISKS OF REMAINING ON OPENSSL 1.1.1

- **Unpatched CVEs**: Any newly discovered weaknesses post-EOL remain exploitable on 1.1.1-based systems.
- **Supply chain exposure**: Images and dependencies pinned to 1.1.1 increase SBOM and vulnerability scanner alerts.
- **Weak/legacy crypto usage**: 1.1.1 may still allow legacy algorithms and ciphers that are disabled or isolated in 3.x.
- **Compliance and certification gaps**: Modern standards prefer actively maintained crypto libraries; FIPS validation pathways center on OpenSSL 3.x.

- **Vendor ecosystem**: Tooling and SDKs increasingly assume 3.x APIs and features (Providers, FIPS module).

KNOWN ISSUES AND SECURITY GAPS IN THE 1.1.1 BRANCH

- **EOL status**: No backported fixes after Sept 2023; any future protocol or implementation weaknesses will persist.
- **Legacy mechanisms**: Engines and certain deprecated algorithms remain in use; 3.x moves them to the *legacy provider* to discourage accidental reliance.
- **FIPS limitations**: Practical FIPS pathways for modern OpenSSL are in the 3.x series (via the 3.x FIPS provider), not 1.1.1.
- **Hardening gaps**: Ongoing side-channel mitigations, stricter defaults, and policy controls land in 3.x; 1.1.1 will not receive these improvements.

OPENSSL SUPPORT LIFECYCLE SNAPSHOT

Series	Status	Security Fixes End	Notes
1.1.1	**EOL**	Sept 11, 2023	No further updates
3.0 (LTS)	Supported	Planned through Sept 2026 (LTS)	FIPS provider available
3.1/3.2/3.3	Supported (non-LTS)	Per-series window	Features, performance, QUIC (from 3.1)

BENEFITS OF UPGRADING TO OPENSSL 3.X

- **Active security maintenance**: Timely patches for vulnerabilities.
- **Provider architecture**: Clean separation of approved and legacy algorithms via providers (default, legacy, FIPS, etc.).
- **FIPS module**: FIPS 140-3 validated module pathway for compliance-sensitive environments (3.0+).
- **Stronger defaults**: Tighter algorithm policies and easier enforcement of crypto standards.
- **Modern features**: QUIC support (3.1+), improved algorithm implementations, ongoing performance and hardening work.

UPGRADE PLANNING CHECKLIST

1. **Inventory**: Identify all binaries and libraries linked with OpenSSL 1.1.1 (apps, proxies, agents, containers, OS images).
2. **Select target**: Choose 3.0 LTS for stability/FIPS; or 3.2/3.3 if you need newer features and are comfortable with non-LTS cadence.
3. **Assess crypto policy**: Decide whether to enable the *legacy* provider temporarily; define acceptable algorithms and key sizes.
4. **FIPS needs**: Determine if your environment requires the 3.x FIPS provider and plan validation and configuration steps.
5. **Map APIs**: Replace deprecated low-level crypto calls with **EVP** APIs; migrate ENGINE usage to Providers.
6. **Build system**: Update toolchains, pkg-config, CI images, and cross-compile targets to link against 3.x.
7. **Testing**: Run unit, integration, TLS interoperability, and performance tests; enable strict cipher and cert policy tests.
8. **Rollout**: Stage deployments with canaries and feature flags; implement rollback and monitoring.

TECHNICAL MIGRATION GUIDE

API AND ARCHITECTURAL CHANGES

- **Providers replace ENGINE**: The ENGINE framework is deprecated; use providers to load algorithms (default, legacy, FIPS).
- **Prefer EVP layer**: Low-level crypto (e.g., direct RSA/MD APIs) is deprecated; use `EVP_PKEY`, `EVP_MD`, `EVP_CIPHER`, `EVP_KDF`, `EVP_MAC`.
- **Algorithm availability**: Legacy algorithms (e.g., MD4/MD5, RC4, DES, some SHA-1 uses) may require enabling the *legacy* provider explicitly.
- **Randomness**: Use `EVP_RAND` abstractions; avoid direct RAND_bytes if your design relies on specific DRBG instances.
- **Version checks**: Use `OPENSSL_version_major()` or feature detection to branch behavior when necessary.

CONFIGURATION EXAMPLES

Enable the legacy provider temporarily (system-wide openssl.cnf):

```
# openssl.cnf (OpenSSL 3.x)
config_diagnostics = 1
[openssl_init]
providers = provider_sect
alg_section = algorithm_sect
[provider_sect]
default = default_sect
legacy = legacy_sect
[default_sect]
activate = 1
[legacy_sect]
activate = 1
[algorithm_sect]
# Optional: raise security level, tune ciphers, etc.
```

Recommendation: Use the legacy provider only as a transitional aid; remove it once dependencies are updated.

Programmatically load providers (C):

```
OSSL_PROVIDER *def = OSSL_PROVIDER_load(NULL, "default");
OSSL_PROVIDER *leg = OSSL_PROVIDER_load(NULL, "legacy");
/* ... use EVP APIs ... */
OSSL_PROVIDER_unload(leg);
OSSL_PROVIDER_unload(def);
```

EVP-based signing (replacing low-level RSA):

```
EVP_PKEY *pkey = /* load key */;
EVP_MD_CTX *ctx = EVP_MD_CTX_new();
EVP_DigestSignInit(ctx, NULL, EVP_sha256(), NULL, pkey);
EVP_DigestSignUpdate(ctx, data, data_len);
EVP_DigestSignFinal(ctx, sig, &sig_len);
EVP_MD_CTX_free(ctx);
```

TLS AND CIPHER POLICY

- **Prefer TLS 1.2+**; disable TLS 1.0/1.1 if still enabled in your stack.
- **Strong suites**: Prefer ECDHE and AEAD (e.g., TLS 1.2 with ECDHE-ECDSA-AES128-GCM-SHA256; TLS 1.3 suites are predefined and strong by default).
- **Certificate policy**: Avoid SHA-1 signatures; use SHA-256+ and 2048-bit+ RSA or ECDSA with P-256/P-384.
- **Security level**: Consider raising OpenSSL security level (e.g., to 2) where interoperability permits.

FIPS 140-3 CONSIDERATIONS (3.X)

- OpenSSL 3.x supports a FIPS provider that, when properly installed and configured, enforces FIPS-approved algorithms.
- Follow project/vendor guidance to install the validated module and lock configuration.
- Audit that applications do not require legacy algorithms once FIPS mode is enabled.

TESTING AND ROLLOUT STRATEGY

1. **Build matrix**: Test across your supported OS/arch combinations.
2. **Interop tests**: Validate TLS handshakes with major peers (web servers, proxies, databases, message brokers).
3. **Negative tests**: Ensure disallowed algorithms/ciphers correctly fail.
4. **Performance**: Benchmark CPU profiles and latency; some algorithms may differ in performance.
5. **Staged deploy**: Canary, observe, then gradually expand. Keep rollback images with 3.x-linked binaries.

COMMON PITFALLS

- **Hidden dependencies**: Third-party libraries may dynamically load 1.1.1; scan containers and LD paths.
- **ENGINE reliance**: HSM/KMS integrations via ENGINE need provider-based alternatives or vendor updates.

- **Algorithm availability**: Builds fail or runtime errors occur if legacy algorithms are assumed but not enabled.
- **Policy drift**: Inconsistent openssl.cnf across hosts leads to surprising behavior; standardize config.

MIGRATION DECISION TABLE

Requirement	Recommendation
Long-term stability, FIPS option	OpenSSL 3.0 LTS
Newest features/perf (willing to track updates)	OpenSSL 3.2/3.3
Legacy algorithms temporarily needed	Enable legacy provider during migration only
Strict compliance	Use FIPS provider, disable legacy provider

ACTION ITEMS

- Set an organizational deadline to remove OpenSSL 1.1.1 from all production systems.
- Adopt OpenSSL 3.0 LTS (or newer supported 3.x) across builds and base images.
- Replace deprecated APIs with EVP and remove ENGINE usage.
- Establish crypto policy baselines and continuous compliance checks.

REFERENCES

- OpenSSL Project: Release and support policies (EOL dates)
- OpenSSL 3.x Migration Guide and Provider documentation
- FIPS 140-3 guidance for OpenSSL 3.x

ASSUMPTIONS

- You require a maintained, supported crypto stack and either LTS stability (3.0) or newer features (3.2/3.3).

- Some environments may temporarily need legacy algorithms; you will phase them out post-migration.

David Keith Jr "mrdj" ✔