

Post-Quantum Vulnerable Documentation: Post- Quantum Vulnerable SSL and Recommended Countermeasures

Overview

The rise of quantum computing poses a direct threat to traditional cryptographic algorithms used in SSL/TLS, such as **RSA**, **ECDSA**, and **Elliptic Curve Diffie-Hellman (ECDH)**. These algorithms rely on problems (like integer factorization and discrete logarithms) that quantum computers can solve efficiently using **Shor's algorithm**, rendering current SSL/TLS protocols **post-quantum vulnerable**.

This document outlines how SSL/TLS can be vulnerable in a post-quantum context and provides recommendations for post-quantum secure alternatives.

⚠ Post-Quantum Vulnerabilities in SSL/TLS

1. Key Exchange Vulnerabilities

Traditional key exchange mechanisms such as:

- RSA key exchange
- Elliptic Curve Diffie-Hellman (ECDH)

These are vulnerable because quantum computers can derive the private keys from intercepted public data using Shor’s algorithm.

2. Digital Signature Vulnerabilities

- RSA and ECDSA signatures can be forged with quantum algorithms.
- Intercepted SSL traffic today can be decrypted in the future using **harvest-now, decrypt-later** attacks.

Modern Post-Quantum Safe Ciphers (Hybrid and Pure)

To mitigate quantum risks, TLS implementations must transition to **quantum-resistant cryptographic algorithms**. These can be:

- **Hybrid schemes:** Combine classical and post-quantum algorithms (interoperable and backward compatible)
- **Pure PQC schemes:** Use only post-quantum algorithms (ideal for future-proofing)

NIST-Recommended Post-Quantum Algorithms (2024)

Type	Algorithm	Description
Key Encapsulation	Kyber (Kyber-512, Kyber-768, Kyber-1024)	Lattice-based; standardized as ML-KEM
Signatures	Dilithium (Dilithium2, Dilithium3, Dilithium5)	Lattice-based digital signature
	Falcon	Compact, lattice-based signature
	SPHINCS+	Hash-based signature; conservative fallback

☐ Recommended Cipher Suites (TLS 1.3 with PQC)

As of today, post-quantum cryptography is often integrated through **TLS hybrid key exchanges** via extensions like [x25519+kyber512] or via **Open Quantum Safe (OQS)** libraries.

Example Hybrid Cipher Suites (using NIST PQC algorithms):

1. **TLS_AES_256_GCM_SHA384 + X25519+Kyber768**
2. **TLS_CHACHA20_POLY1305_SHA256 + Kyber512**
3. **TLS_AES_128_GCM_SHA256 + ECDHE + Kyber1024**

“△ These are not yet formal cipher suite names; they represent the **key exchange combinations** used in hybrid TLS libraries like [BoringSSL](#), [OpenSSL](#) with OQS, and [AWS s2n-tls](#).

☐ Implementation Options

Libraries supporting Post-Quantum TLS:

- **Open Quantum Safe (OQS)**
 - Extensions for **OpenSSL**, **BoringSSL**, and **liboqs**
 - **Google BoringSSL**
 - Early experimental support for hybrid schemes
 - **AWS s2n-tls**
 - Supports hybrid post-quantum TLS 1.2 and TLS 1.3
 - **Cloudflare** and **Google Chrome** (experimental support in QUIC)
-

Migration Strategy

- Evaluate inventory of TLS endpoints**
 - Identify services using RSA/ECDSA and vulnerable ciphers.
- Use hybrid schemes first**
 - Interoperable with legacy clients while offering PQC protection.
- Enable PQC cipher suites in controlled environments**
 - Internal APIs, backend services, etc.
- Monitor NIST and IETF standards**
 - Stay aligned with official PQC cipher standardization efforts.
- Implement TLS 1.3**
 - Required for most PQC-ready key exchange implementations.

Ciphers to Deprecate Immediate

Avoid these in future-proof systems:

- TLS_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA

These are based on RSA/ECDSA and vulnerable to quantum attacks.

NIST PQC Standardization Milestones

Year	Milestone
2022	NIST announced finalists (Kyber, Dilithium, etc.)
2024	Final standards (ML-KEM for Kyber, ML-DSA for Dilithium)
2025	Adoption into commercial systems and TLS libraries

Summary

- SSL/TLS systems that rely on RSA/ECDSA are **vulnerable to quantum attacks**.
- Transitioning to **post-quantum key exchange and signature schemes** is critical.
- Adopt hybrid schemes like **X25519+Kyber** during the transition phase.
- Monitor evolving standards from **NIST**, **IETF**, and **TLS WG**.

Revision #1

Created 23 July 2025 08:26:49 by Admin

Updated 23 July 2025 08:29:42 by Admin