Mar 31

minosgalanakis tf-psa-crypto-… 29160dd Compare

TF-PSA-Crypto 1.1.0

Latest

Description

This release of TF-PSA-Crypto provides new features, bug fixes and minor enhancements. This release includes fixes for security issues.

TF-PSA-Crypto 1.1 is a long-term support (LTS) branch. It will be supported with bug-fixes and security fixes until at least March 2029.

Security Advisories

For full details, please see the following links:

- Entropy on Linux can fall back to `/dev/urandom`
- PSA random generator cloning
- Compiler-induced constant-time violations
- Buffer overflow in FFDH public key export
- FFDH: lack of contributory behaviour due to improper input validation
- CCM multipart finish tag-length validation bypass

Release Notes

Removals

- The undocumented ability to load persistent keys outside the user key ID range has been removed. (This does not affect MBEDTLS_PSA_CRYPTO_BUILTIN_KEYS.)
- The headers no longer define 'inline' as a macro. This was done on Arm Compiler 5 and MSVC. The compiler versions that needed this definition are no longer supported since TF-PSA-Crypto 1.0.

Features

- The automatic computation of MBEDTLS_PSA_STATIC_KEY_SLOT_BUFFER_SIZE has been improved to take into account the following key types: asymmetric keys, ciphers, AEADs, CMAC and HMAC.
- mbedtls_pk_write_pubkey_psa() is introduced to allow exporting the public key from a PK context in a format that can easily be imported into PSA.
- Implement SHAKE (PSA_ALG_SHAKE128, PSA_ALG_SHAKE256).
- The device for reading entropy on platforms without a dedicated system call can now be configured with MBEDTLS_PLATFORM_DEV_RANDOM or mbedtls_platform_dev_random.
- Applications can use the new functions psa_random_reseed() to request an immediate reseed of the PSA random generator, or psa_random_deplete() to force a reseed on the next random generator call.
- Applications can call psa_random_set_prediction_resistance() to toggle prediction resistance in the PSA random generator.

Security

- The default device for reading entropy on platforms without a dedicated system call is now /dev/random instead of /dev/urandom. This is safer on Linux in case the application runs early after the kernel boots, but may block needlessly on Linux <= 5.6. Reported by supers1ngular (BayLibre).
- Fix missing validation of the peer's key in key agreement operations using PSA_ALG_FFDH: low-order elements were not rejected as they should be. This is a problem for protocols using FFDH that expect contributory behaviour, that is, where neither party should be able to force the shared secret into a small set. Reported independently by Eva Crystal (0xiviel) and another reporter.
- Add tag length validation in mbedtls_ccm_finish() to prevent out-of-bounds reads and mitigate potential application buffer overflows where applications relied on the library to enforce

tag length constraints.
Reported by Eva Crystal (0xiviel).

- Fix a buffer overflow in psa_export_public_key() for FFDH keys when the output buffer is too small. Found by Haruto Kimura (Stella).
- If an application called psa_crypto_init() then fork() and continued to use cryptography APIs (possibly indirectly, e.g. for TLS), the random generator states were duplicated. Fix this by forcing a RNG reseed in the child process. CVE-2026-25835
- Applications running in environments where the application state is cloned (for example due to resuming a frozen system state multiple times, or due to cloning a virtual machine image) should arrange to reseed the random generator using one of the new functions psa_random_reseed() or psa_random_deplete(). CVE-2026-25835

Bugfix

- Appease GCC 14.3's array bounds checker by inserting checks in mbedtls_xor that bail before the byte-at-a-time loop when the array size is a constant (using MBEDTLS_HAS_BUILTIN) and an exact multiple of the larger loop size.
- CMake now installs headers to `CMAKE_INSTALL_INCLUDEDIR` instead of the hard-coded `include` directory.
- Fix CMake package version that was inconsistent with the product version. Fixes #553.
- Fix CMake failure on Windows because of a native directory separator. Fixes Mbed-TLS/mbedtls#10502.
- Partially fix a performance regression in RSA operations introduced by a security fix in 1.0, by improving the performance of RSA private key operations when MBEDTLS_RSA_NO_CRT is disabled, which is the default.
- Fix compilation errors in `aesce.c` in some Visual Studio builds. Fixes #548.
- Interruptible operations (ECDH key agreement, ECC key generation) were not actually interruptible (always completed in one go) in builds with ECDSA disabled.
- Built-in SHA3 was included in the build even when SHA3 had a PSA accelerator. Fix this. Fixes #542.
- Fix a bug that caused GCM tag calculations to fail, so that data was correctly encrypted but could not be authenticated. The bug was only observed with GCC 10.0 to 14.2 inclusive, when compiling with -O3, and

running without AESNI or AESCE.
Fixes #665.

- Fix a build failure with dietlibc.
- Some functions in PK were using large buffers (around 2KB in the default configuration) on the stack, which was a problem in environments with a small stack. Those buffers are now allocated on the heap, except in configurations where ECC is the only supported key type in PK, making PK still independent of the heap in such configurations (if the ECC driver itself is not using the heap). Fixes #476.

Changes

- ChaCha20 size and performance: add a Neon implementation of ChaCha20 for Thumb2 and 32 and 64-bit Arm, for Armv7 onwards. At default settings, this improves performance by around 2x to 2.7x on Aarch64.
- Add a new function, mbedtls_pk_get_key_type(), which returns the PSA key type corresponding to the type of the key represented by the given PK object.
- Running the unit tests now requires a heap (possibly from MBEDTLS_MEMORY_BUFFER_ALLOC_C). They now use less stack (almost 5000 bytes less).
- Static assertions in the library (`MBEDTLS_STATIC_ASSERT`) are now always enabled, using indirect methods in pre-C11 compilers. This change also fixes warnings in pedantic mode with GCC or Clang on some platforms.
- Tweak the detection of Unix-like platforms, which makes more system interfaces (timing, threading) available on Haiku, QNX and Midipix.
- On MinGW, always use a standard-compliant printf function family.
- Non-driver files have been moved out of `drivers/builtin/src` into three new directories at the root of the repository:
  - `extras`: modules implemented on top of the PSA Cryptography API, or providing functionality beyond it (for example, the LMS stateful hash implementation currently).
  - `platform`: modules implementing the platform abstraction layer.
  - `utilities`: utility modules used by the built-in drivers, the PSA Cryptography API implementation, modules in `extras`, and potentially by security protocols such as TLS.
- A new directory `dispatch` has been added at the root of the repository to eventually host all code that dispatches cryptographic operations to

drivers, such as `psa_crypto_driver_wrappers_no_static.c`. For the time being, it only contains `psa_crypto_driver_wrappers_no_static.h`.

Who should update

We recommend all users should update to take advantage of the bug fixes contained in this release at an appropriate point in their development lifecycle.

Note

> ❗ `tf-psa-crypto-1.1.0.tar.bz2` are our official release files. `source.tar.gz` and `source.zip` are automatically generated snapshots that GitHub is generating. They do not include external dependencies, and can't be configured

Checksum

The SHA256 hashes for the archives are:
a0b011b7f2c427cc8ee70116bb2d859543014534ae4d7020a69613aec10dc1b4 tf-psa-crypto-1.1.0.tar.bz2

▼ **Assets** 4

Oct 15, 2025

minosgalanakis tf-psa-crypto-… `76920ed` Compare

TF-PSA-Crypto 1.0.0

Description

The Mbed TLS 4.0.0 and TF-PSA-Crypto 1.0.0 releases introduce a major codebase restructuring:

- PSA Crypto functionality now resides in its own repository Mbed-TLS/TF-PSA-Crypto.

- TLS and X.509 components remain in Mbed TLS.

They also include fixes for two security issues.

API Changes & Migration

The Mbed TLS 4.0.0 and TF-PSA-Crypto 1.0.0 releases introduce significant API changes that break backward compatibility with previous versions. As a result, they may require substantial updates to your codebase. Please refer to the 1.0 migration guide to update your codebase to the new crypto interfaces.

Please note

- Mbed TLS 4.0.0 includes TF-PSA-Crypto 1.0.0 (as a subtree in the tarball, and as a submodule in git) and can only be built with this included version TF-PSA-Crypto.

Security Advisories

Compared to Mbed TLS 3.6.4 (the latest LTS release before the split), TF-PSA-Crypto 1.0.0 includes fixes for the following vulnerabilities:

- Padding oracle through timing of cipher error reporting
- Side channel in RSA key generation and operations (SSBleed, M-Step)

Compared to Mbed TLS 3.6.0 (the latest feature release before the split), TF-PSA-Crypto 1.0.0 includes fixes for the following vulnerabilities, which were already fixed in Mbed TLS 3.6.4:

- Race condition in AESNI support detection
- Heap buffer under-read when parsing PEM-encrypted material
- Unchecked return value in LMS verification allows signature bypass
- Out-of-bounds read in mbedtls_lms_import_public_key()
- Timing side-channel in block cipher decryption with PKCS#7 padding

Release Notes

API changes

- The experimental functions psa_generate_key_ext() and psa_key_derivation_output_key_ext() have been replaced by psa_generate_key_custom() and psa_key_derivation_output_key_custom(). They have almost exactly the same interface, but the variable-length

data is passed in a separate parameter instead of a flexible array member. This resolves a build failure under C++ compilers that do not support flexible array members (a C99 feature not adopted by C++). Fixes #9020.

- The PSA and Mbed TLS error spaces are now unified. mbedtls_xxx() functions can now return PSA_ERROR_xxx values.
  This will not affect most applications since the error values are between -32767 and -1 as before.
- Remove MBEDTLS_PK_RSA_ALT from the PK module.
- MBEDTLS_ERR_PK_SIG_LEN_MISMATCH is no longer a distinct error code. A valid signature with trailing garbage is now reported as an invalid signature with all algorithms.
- All API functions now use the PSA random generator psa_generate_random() internally. As a consequence, functions no longer take RNG parameters.
  Please refer to the migration guide at : docs/4.0-migration-guide.md.
- Privatize the functions mbedtls_ecc_group_to_psa and mbedtls_ecc_group_from_psa.
- Remove the functions mbedtls_ecc_group_to_psa() and mbedtls_ecc_group_from_psa(), which are no longer meaningful since ECC groups are no longer exposed directly in the API.
- mbedtls_pk_verify_ext() ignores the options parameter when an MBEDTLS_PK_RSASSA_PSS context type is used. The function assumes that salt length is any and that the hash algorithm used for message, encoding and MGF1 is the same. An error will be returned if any of these assumptions is false.
- Align the mbedtls_nist_kw_wrap() and mbedtls_nist_kw_unwrap() functions with the PSA Crypto API. The functions mbedtls_nist_kw_wrap() and mbedtls_nist_kw_unwrap() now take a PSA key identifier instead of a plain-text key via a custom context.
- Remove mbedtls_pk_encrypt() and mbedtls_pk_decrypt(). Convert the key to PSA and use the PSA functions instead, see the migration guide for details.
- Change MBEDTLS_ERR_ECP_IN_PROGRESS to be an alias of PSA_OPERATION_INCOMPLETE and MBEDTLS_ERR_RSA_VERIFY_FAILED to be an alias
  of PSA_ERROR_INVALID_SIGNATURE.
- Rename mbedtls_pk_setup_opaque to mbedtls_pk_wrap_psa.

- The custom entropy collector callback mbedtls_hardware_poll() (enabled by MBEDTLS_ENTROPY_HARDWARE_ALT) has been replaced by a new callback mbedtls_platform_get_entropy() with different parameters (enabled by MBEDTLS_PSA_DRIVER_GET_ENTROPY). See the new function's documentation and “Custom entropy collector” in the migration guide.
- To build the library with only a nonvolatile seed and no actual entropy source, you now need to enable the new option MBEDTLS_ENTROPY_NO_SOURCES_OK.
- Due to the entropy configuration changes, if you write a configuration file from scratch, the default entropy source MBEDTLS_PSA_BUILTIN_GET_ENTROPY now needs to be enabled explicitly.
- The configuration options MBEDTLS_CTR_DRBG_RESEED_INTERVAL and MBEDTLS_HMAC_DRBG_RESEED_INTERVAL have been replaced by a unified setting of MBEDTLS_PSA_RNG_RESEED_INTERVAL.
- The configuration option MBEDTLS_ENTROPY_FORCE_SHA256 has been removed. MBEDTLS_PSA_CRYPTO_RNG_HASH can now be used to select the entropy module's hashing algorithm.
- The mutex functions provided by platforms where MBEDTLS_THREADING_ALT is enabled have changed in minor ways:
  - The type of mutex objects provided by the platform functions is now called mbedtls_platform_mutex_t, distinct from the API type mbedtls_threading_mutex_t.
  - The mutex_init function now returns an error code.
  - Mutex functions other than mutex_init can now assume that the mutex has been successfully initialized.
- The Random Number Generator configuration options have been refactored.
  The following options have been removed:
  MBEDTLS_ENTROPY_C, MBEDTLS_ENTROPY_FORCE_SHA256, MBEDTLS_ENTROPY_MAX_GATHER, MBEDTLS_ENTROPY_MAX_SOURCES, MBEDTLS_CTR_DRBG_ENTROPY_LEN, MBEDTLS_CTR_DRBG_MAX_INPUT, MBEDTLS_CTR_DRBG_MAX_REQUEST, MBEDTLS_CTR_DRBG_MAX_SEED_INPUT, MBEDTLS_CTR_DRBG_USE_128_BIT_KEY, MBEDTLS_HMAC_DRBG_MAX_INPUT, MBEDTLS_HMAC_DRBG_MAX_REQUEST, MBEDTLS_HMAC_DRBG_MAX_SEED_INPUT and MBEDTLS_PSA_HMAC_DRBG_MD_TYPE.
  The following options have been introduced:
  MBEDTLS_PSA_CRYPTO_RNG_HASH and MBEDTLS_PSA_CRYPTO_RNG_STRENGTH.

See "Random number generation configuration" in the migration guide for more information.

- The following PK interfaces are now private and should no longer be used.
  mbedtls_pk_type_t
  mbedtls_pk_debug_type
  mbedtls_pk_debug_item
  MBEDTLS_PK_DEBUG_MAX_ITEMS
  mbedtls_pk_info_from_type()
  mbedtls_pk_setup()
  mbedtls_pk_get_len()
  mbedtls_pk_can_do()
  mbedtls_pk_can_do_ext()
  mbedtls_pk_debug()
  mbedtls_pk_get_name()
  mbedtls_pk_get_type()
  mbedtls_pk_rsa()
  mbedtls_pk_ec()
  mbedtls_pk_parse_subpubkey()
  mbedtls_pk_write_pubkey()
  mbedtls_pk_verify_new()
- The hmac parameter of the mbedtls_md_setup() function must now always be set to 0 as HMAC is no longer supported by MD. To use HMAC, please use the psa_mac_ API.
- Make the following error codes aliases of their PSA equivalents, where xxx is a module, e.g. ASN1 or PK.
  MBEDTLS_ERR_xxx_BAD_INPUT[*DATA] -> PSA_ERROR_INVALID_ARGUMENT*
  MBEDTLS_ERR_xxx_ALLOC_FAILED -> PSA_ERROR_INSUFFICIENT_MEMORY
  MBEDTLS_ERR_xxx[AUTH/VERIFY]_FAILED -> PSA_ERROR_INVALID_SIGNATURE
  MBEDTLS_ERR_xxx_BUFFER_TOO_SMALL -> PSA_ERROR_BUFFER_TOO_SMALL
  MBEDTLS_ERR_xxx_OUTPUT_TOO_LARGE -> PSA_ERROR_BUFFER_TOO_SMALL
  MBEDTLS_ERR_xxx_INVALID_PADDING -> PSA_ERROR_INVALID_PADDING
- A PK context no longer associates specific algorithms with the key, except when wrapping a PSA key. In particular, after mbedtls_pk_copy_from_psa() or mbedtls_pk_copy_public_from_psa() on an RSA key, the functions mbedtls_pk_get_psa_attributes(), mbedtls_pk_sign() and mbedtls_pk_verify() will use PKCS#1v1.5 signature or encryption, regardless of the original key's policy.

Default behavior changes

- In a PSA-client-only build (i.e. MBEDTLS_PSA_CRYPTO_CLIENT && !MBEDTLS_PSA_CRYPTO_C), do not automatically enable local crypto when the corresponding PSA mechanism is enabled, since the server provides the crypto. Fixes #9126.
- The PK module now always uses the PSA subsystem to perform cryptographic operations, with a few exceptions documented in docs/architecture/psa-migration/psa-limitations.md. This corresponds to the behavior of Mbed TLS 3.x when MBEDTLS_USE_PSA_CRYPTO is enabled. In effect, MBEDTLS_USE_PSA_CRYPTO is now always enabled.
- psa_crypto_init() must be called before performing any cryptographic operation, including indirect requests such as parsing a key or certificate or starting a TLS handshake.
- The `PSA_WANT_XXX` symbols as defined in tf-psa-crypto/in...

Read more

▶ **Assets** 4

Jul 4, 2025

minosgalanakis tf-psa-crypto-… `0cc6306` Compare

TF-PSA-Crypto v1.0.0-beta

Pre-release

Description

The Mbed TLS 4.0.0-beta and TF-PSA-Crypto 1.0.0-beta releases introduce a major codebase restructuring:

- PSA Crypto functionality now resides in its own repository Mbed-TLS/TF-PSA-Crypto.
- TLS and X.509 components remain in Mbed TLS.

API Changes & Migration

These beta releases introduce significant API changes that break backward compatibility with previous versions. As a result, they may require substantial updates to your codebase. Please refer to the 4.0-migration-guide for guidance.

Please note

- These beta versions **must be used together** to validate your integration against the newly split interfaces.
- The 4.0 Migration Guide is currently under construction and will be continuously updated as we prepare for the full, non-beta release.
- Any mention of “previous releases” in this document refers to Mbed TLS 3.6.3 or Mbed TLS 3.6.4 , the latest pre-split versions at the time of these beta releases.

Intended audience for this beta

- Integrators: To evaluate the impact of the Mbed TLS split and API changes on your codebase.
- Early adopters: Anyone who wants to experiment with the upcoming interfaces and provide feedback before the formal release.

These betas are not production-ready. For deployments requiring stability and the latest security fixes, please continue using the LTS release Mbed TLS 3.6.4:

Security Advisories

For full details, please see the following links:

- Race condition in AESNI support detection
- Heap buffer under-read when parsing PEM-encrypted material
- Unchecked return value in LMS verification allows signature bypass
- Out-of-bounds read in mbedtls_lms_import_public_key()
- Timing side-channel in block cipher decryption with PKCS#7 padding

Release Notes

API changes

- The PSA and Mbed TLS error spaces are now unified. mbedtls_xxx() functions can now return PSA_ERROR_xxx values.

This will not affect most applications since the error values are between -32767 and -1 as before.

- Remove MBEDTLS_PK_RSA_ALT from the PK module.
- Replaced MBEDTLS_ERR_PK_SIG_LEN_MISMATCH and MBEDTLS_ERR_ECP_SIG_LEN_MISMATCH error codes with MBEDTLS_ERR_ECP_VERIFY_FAILED and MBEDTLS_ERR_RSA_VERIFY_FAILED
- All API functions now use the PSA random generator psa_get_random() internally. As a consequence, functions no longer take RNG parameters. Please refer to the migration guide at : tf-psa-crypto/docs/4.0-migration-guide.md.
- Privatize the functions mbedtls_ecc_group_to_psa and mbedtls_ecc_group_from_psa.
- mbedtls_pk_verify_ext() ignores the options parameter when an MBEDTLS_PK_RSASSA_PSS context type is used. The function assumes that salt length is any and that the hash algorithm used for message, encoding and MGF1 is the same. An error will be returned if any of these assumptions is false.
- Align the mbedtls_nist_kw_wrap() and mbedtls_nist_kw_unwrap() functions with the PSA Crypto API. The functions mbedtls_nist_kw_wrap() and mbedtls_nist_kw_unwrap() now take a PSA key identifier instead of a plain-text key via a custom context.
- Remove mbedtls_pk_encrypt() and mbedtls_pk_decrypt(). Convert the key to PSA and use the PSA functions instead, see the migration guide for details.
- Change MBEDTLS_ERR_ECP_IN_PROGRESS to be an alias of PSA_OPERATION_INCOMPLETE and MBEDTLS_ERR_RSA_VERIFY_FAILED to be an alias of PSA_ERROR_INVALID_SIGNATURE.

Removals

- Remove many MBEDTLS_ERR_xxx error codes, superseded by PSA_ERROR_xxx. See the 4.0 migration guide for details.
- Support for dynamic secure elements (i.e. MBEDTLS_PSA_CRYPTO_SE_C) was already marked as deprecated and it has been removed.
- Removed the MBEDTLS_PSA_INJECT_ENTROPY configuration option from crypto_config.h. The functionality that this option was enabling will be reintroduced as part of the work on issue #8150.

- MBEDTLS_NO_PLATFORM_ENTROPY and the previously deprecated MBEDTLS_ENTROPY_HARDWARE_ALT are removed. See the documentation of MBEDTLS_PLATFORM_GET_ENTROPY_ALT for a description on how the entropy module gathers entropy data.
- MBEDTLS_ENTROPY_MIN_HARDWARE is also removed following the removal of MBEDTLS_ENTROPY_HARDWARE_ALT.
- TF-PSA-Crypto does not provide an OID API. A subset of the OID interfaces of Mbed TLS 3.x are now in the X.509 library in Mbed TLS 4.x.
- Removed the MBEDTLS_SHA3_C configuration option from crypto_config.h. SHA3 can now be configured with the PSA_WANT_SHA3_* options.
- The library no longer offers interfaces to look up values by OID or OID by enum values. The compilation option MBEDTLS_OID_C no longer exists. OID tables are included in the build automatically as needed. OIDs that are not relevant to TF-PSA-Crypto have been removed.
- Remove the function mbedtls_asn1_get_mpi() from the public interface. It is replaced by mbedtls_asn1_get_integer().

Features

- Add an interruptible version of export public-key to the PSA interface. See psa_export_public_key_iop_setup() and related functions.
- MD module can now perform PSA dispatching also when `MBEDTLS_PSA_CRYPTO_CLIENT && !MBEDTLS_PSA_CRYPTO_C`, even though this configuration is not officially supported. This requires that a PSA Crypto provider library which:
  - supports the required `PSA_WANT_ALG_xxx` and
  - implements `psa_can_do_hash()` on the client interface is linked against Mbed TLS and that `psa_crypto_init()` is called before performing any PSA call.
- Add a program (which_aes) that uses an internal function to print out the current implementation of AES, i.e. software, AESCE, AESNI assembly, or AESNI intrinsics.
- To supply a platform-specific entropy source, define the compilation option MBEDTLS_PLATFORM_GET_ENTROPY_ALT and provide the callback function mbedtls_platform_get_entropy_alt(). This function should typically access a TRNG ("true hardware random generator") device on bare-metal platforms, or call an operating system function to obtain cryptographic-quality random data. Mbed TLS requires that a minimum of 32 bytes (not configurable) are returned from this function for a successful entropy gathering round.

- The new function mbedtls_asn1_get_integer() parses an integer into a byte array. It replaces mbedtls_asn1_get_mpi().

Security

- Zeroize a temporary heap buffer used in psa_key_derivation_output_key() when deriving an ECC key pair.
- Zeroize temporary heap buffers used in PSA operations.
- Fix a buffer overread in mbedtls_lms_import_public_key() when the input is less than 3 bytes. Reported by Linh Le and Ngan Nguyen from Calif. CVE-2025-49601
- Fix a vulnerability in LMS verification through which an adversary could get an invalid signature accepted if they could cause a hash accelerator to fail. Found and reported by Linh Le and Ngan Nguyen from Calif. CVE-2025-49600
- On x86/amd64 platforms, with some compilers, when the library is compiled with support for both AESNI and software AES and AESNI is available in hardware, an adversary with fine control over which threads make progress in a multithreaded program could force software AES to be used for some time when the program starts. This could allow the adversary to conduct timing attacks and potentially recover the key. In particular, this attacker model may be possible against an SGX enclave.
  The same vulnerability affects GCM acceleration, which could allow a similarly powerful adversary to craft GCM forgeries.
  CVE-2025-52496
- Fix a bug in mbedtls_asn1_store_named_data() where it would sometimes leave an item in the output list in an inconsistent state with val.p == NULL but val.len > 0. Functions using the structure after that, including mbedtls_asn1_store_named_data() itself (and functions from the X.509 library) would then dereference a NULL pointer. Applications that do not call this function (directly, or indirectly through X.509 writing) are not affected. Found by Linh Le and Ngan Nguyen from Calif. CVE-2025-48965
- Fix an integer underflow that could occur when parsing malformed PEM keys, which could be used by an attacker capable of feeding encrypted PEM keys to a user. This could cause a crash or information disclosure. Found and reported by Linh Le and Ngan Nguyen from Calif. CVE-2025-52497

- Fix a timing side channel in the implementation of PKCS#7 padding which would allow an attacker who can request decryption of arbitrary ciphertexts to recover the plaintext through a timing oracle attack. Reported by Ka Lok Wu from Stony Brook University and Doria Tang from The Chinese University of Hong Kong. CVE-2025-4908...

Read more

Assets 4

🚀 1 1 person reacted