YubiHSM 1.0 security advisory 2012-01

Abstract

A security review has revealed two methods to decrypt AES-CCM encrypted data (so called AEADs) if an attacker has live (remote) access to an unlocked YubiHSM, under some valid configurations. The methods have not been documented and may be unexpected for those unfamiliar with how AES-CCM works. This document describes the two methods and suggests workarounds.

Credits

The security review was performed by Robert Künnemann and Graham Steel of the Laboratoire Spécification et Vérification, a joint CNRS/ENS-Cachan/INRIA research group.

Decrypt using raw AES (AES_ECB_BLOCK_ENCRYPT)

Section 4.3 of the reference manual (version 2011-09-14) says : "The YubiHSM intentionally does not provide any functions that decrypts an AEAD and returns it in clear text, either fully or partial.". To clarify, this refers to the command set of the YubiHSM, not necessarily to the cryptographic properties of the generated AEADs.

The YubiHSM 1.0 provide basic AES ECB block encryption/decryption primitives. If the keyhandle which is used to generate AEADs also has the AES-ECB encryption operation, the user can use the AES_ECB_BLOCK_ENCRYPT function to decrypt AES-CCM encrypted data. This is a natural consequence of the design of AES CCM, which generates a stream that is used to XOR the plaintext. The stream is generated by AES encrypting certain data. By using the ECB encryption operation, the same stream is reproduced and data can be decrypted.

This threat is avoided by removing the AES_ECB_BLOCK_ENCRYPT permission for the particular key handle. In general, we strongly recommend that only the permission flags necessary for operation are enabled for each key.

Decrypt using AEAD generate (YSM_AEAD_GENERATE)

The YSM_AEAD_GENERATE command can be used to effectively decrypt a previously generated AEAD. This also follows from the nature of AES-CCM which generates an encryption stream to which the data is XORed to. If an attacker

1. can acquire a previously generated AEAD, together with the nonce value used when generating the AEAD, and
2. can use a YubiHSM with the same keyhandle that generated the first AEAD to generate a second AEAD with arbitrary nonce and plaintext

then the attacker will be able to decrypt the plaintext from the first AEAD using the YubiHSM. This threat is mitigated by observing that a YubiHSM used to generate AEADs is guarded closely to not permit maliciously crafted input.