Analysis of Yubikey and YubiHSM

(work in progress)

Graham Steel

April 6, 2012

Yubikey

- A low-cost one time password (OTP) generator token
- Connects to USB port
- Acts like a keyboard, no driver required
- Press with your finger to generate an OTP
- Used by Google and Microsoft to secure cloud access

Modes of Operation

Current Version (2.0) has two 'slots', each can store 128 bits
Use these in severval modes:

- Yubikey OTP
- OAUTH-HOTP (RFC 4226)
- Static 128 bit password
- Challenge-response

Software for using Yubikey is mostly open source

Yubikey OTP Protocol

$$\begin{array}{lcll} C & \rightarrow & S & : & ID_c, Nonce, OTP_u \\ S & \rightarrow & C & : & OTP_u, status, Nonce, hmac(K_{cs}, OTP_u, Nonce) \end{array}$$

OTP is:

DeviceID, Session counter, Timestamp, Session use, PR, CRC

Total 16 bytes, encrypted under AES-128 to give OTP

$$\{ID, Sc, T, Su, R, CRC\}_{K_{ID}}$$

Verifying OTP

Server has database, for each device ID: K_{ID}, Last S_u, Last S_c
After decrypting OTP, two cases are possible.
Valid OTP, database is updated with the new values if:

- S_c greater value that the last one received
- S_c same value, but S_u counter is greater

Invalid OTP, if:

- device ID is incorrect
- S_c is smaller than the value stored
- S_c equal to the value stored, S_u equal or smaller

Protecting Keys on the Server

Server must store all K_{ID} which could be used to fake OTPs
In event of server compromise, this is a security risk
Yubico solution: YubiHSM ($500)
Also connects to USB port
Contains a master key $K_M$used to encrypt K_{ID} together with ID

YubiHSM API

YSM_AEAD_GENERATE:

$$h(K_M), K_{ID}, ID \rightarrow \{K_{ID}, ID\}_{K_M}$$

Give AES key and ID for a Yubikey, returns (authenticated) encryption
YSM_AEAD_YUBIKEY_OTP_DECODE:

$$h(K_M), \{K_{ID}, ID\}_{K_M}, \{ID, Sc, T, Su, R, CRC\}_{K_{ID}} \rightarrow Sc, T, Su$$

(+20 other commands)

CCM Mode (RFC 3610)

CTR + CBC-MAC
Proved IND-CCA secure by Jonsson (SAC 2002)

Attack

"The YubiHSM intentionally does not provide any functions that decrypts an AEAD and returns it in clear text, either fully or partial" (Manual, §4.3)

O RLY?

Real command YSM_AEAD_GENERATE:

$$h(K_M), K_{ID}, ID, N \rightarrow \{K_{ID}, ID\}_{K_M}$$

Can be used to decrypt!

Use same nonce, first block of ciphertext in place of K_{ID}

Yubico reaction

http://static.yubico.com/var/uploads/pdfs/SecurityAdvisory%202012-02-13.pdf

"In warranty upgrade" to customers

Suggest that you keep the YubiHSM that generates the passwords separate from the one that verifies them..

A Better Solution?

SIV mode (Rogaway-Shrimpton)

Back to the Yubikey Protocol

Verification of a 'no replay' property: Vamanu Master's thesis on using AIF (Mödersheim CCS '10) tool to try to verify

– successful only on highly simplified version, abstraction of state not suitable

Künnemann now working on this as part of a broader study

Tamarind: new version of Scyther (Basin et al.) handles state but doesn't terminate on Yubikey examples

– accepts invariants, possible approach is to try to infer these

Open Problems

- Provisioning (Keyloading) for Yubikey
- Rest of yubiHSM API
- Interaction of yubiHSM API + kerberos etc.
- Security notions for Yubikey + protocols
- Stateful APIs + protocols more generally

MERCI

prosecco.gforge.inria.fr