DNSSEC Keys in SmartcardHSM
OpenSC on Mac OS

Luis D Espinoza Sanchez & Eberhard W Lisse

University of Costa Rica & Namibian Network Information Centre

2015-02-09
Introduction

- DNSSEC is Easy!
  - Is it Secure?

- Secure DNSSEC is Expensive!
  - Is it really?

- So, what were we looking for?
  - An easy, secure and cheap DNSSEC solution
    - for .NA;
    - for demonstration purposes;
    - for fun (see slide 19)
Introduction
Registry System without DNSSEC
Introduction
Registry System with DNSSEC

- Database
- Bind style tables
- Sign files
- Update serial
- Reload signed zone
- SW keys
- HW Keys
Introduction

Hardware Keys

Espinoza & Lisse (NA-NiC)
SmartcardHSM
Introduction

Why Mac?

- **SmartcardHSM**
  - Different Brands
- **Smartcard Readers**
  - Different Brands
- **Open Source Software**
  - **OpenSC**
  - **BIND 9**
    - **Homebrew**
    - **MacPorts**
  - **Virtual Box**
    - **Centos 6**
- **OS X 10.10.2**
  - Native Drivers for the Readers
Proof of Concept
Not Production Grade. Yet.

Why Mac?

- Name Servers usually don’t run on Netbooks
  - BSD
  - Ubuntu
  - Centos
- It’s fun (see slide 19)

No auditing (Ceremony)

- Can be added later from Richard Lamb’s Ceremony documentation

Key in Hardware adds some security

- Physical Access to Server is required
  - Servers are usually in secure data center
Implementation

Bash

- Consolidate Richard Lamb’s Ceremony Scripts
  - into 1 single script
  - 50 lines

- dialog
  - to display/modify Environment Variables
Set Environment Variables

Bash

- `DATE='date -u +%Y%m%d%H%M%S'`
- `DOMAIN=na`
- `PASSWORD=RichardLamb`
- `PATH=~/Downloads/dccom:$PATH`
- `PIN1=123456`
- `PKCS11_LIBRARY_PATH=/Library/OpenSC/lib/opensc-pkcs11.so`
- `SOPIN="3537363231383830"
- `CKALABEL="ksk.""$DOMAIN"".""$DATE"`
Initialization

Prepare the Card

- `sc-hsm-tool --initialize --so-pin $SOPIN --pin $PIN1`
  - Erase the Card

- `sc-hsm-tool --initialize --so-pin $SOPIN --pin $PIN1 --dkek-shares 2`
  - Device Key Encryption Key (DKEK) shares are used to derive the actual keys
Create 2 DKEK Shares

- `sc-hsm-tool –create-dkek-share dkek-share-1.pbe \  
  –so-pin $SOPIN –pin $PIN1 \  
  –password $PASSWORD`

- `sc-hsm-tool –create-dkek-share dkek-share-2.pbe \  
  –so-pin $SOPIN –pin $PIN1 \  
  –password $PASSWORD`
Import the DKEK Shares

- `sc-hsm-tool --import-dkek-share dkek-share-1.pbe` \
  `-so-pin $SOPIN --pin $PIN1` \
  `-password $PASSWORD`

- `sc-hsm-tool --import-dkek-share dkek-share-2.pbe` \
  `-so-pin $SOPIN --pin $PIN1` \
  `-password $PASSWORD`
Generate 2 ZSKs

Why 2?

- `dnssec-keygen -r /dev/random -a 8 -b 1024 \ $DOMAIN.`
- `dnssec-keygen -r /dev/random -a 8 -b 1024 \ $DOMAIN.`
pkcs11-tool -module $PKCS11_LIBRARY_PATH \
 -l -pin $PIN1 -keypairgen -key-type rsa:2048 \
 -read-object -type pubkey \
 -output-file "$CKALABEL".pub \
 -label "$CKALABEL"
Verification

Dump the Card

- pkcs15-tool -D
Wrap the Key

Export and encrypt (wrapped with shares) copy of the private key

- `sc-hsm-tool --wrap-key "$CKALABEL".wrap --key-reference 1 --pin $PIN1`
hcardsign (Bash script)
Generate pre-KSK-signed DNSKEY RRsets for future use

- relies on
  pkcs11-backup -f$CKALABEL:8:257:$DOMAIN. \ 
  -S 0 -P $PIN1

- **Open Source** (Richard Lamb)
  - Doesn’t currently compile on the Mac
  - Will do so RSN

- Not an issue
  - Works on Linux
  - Not required in production
    - Less Safe

- Demonstrated here only to show functionality
Make Backup Card From Wrapped Key
Repeat steps for additional cards

- `sc-hsm-tool --initialize --so-pin $SOPIN --pin $PIN1`
- `sc-hsm-tool --initialize --so-pin $SOPIN --pin $PIN1 --dkek-shares 2`
- `sc-hsm-tool --import-dkek-share dkek-share-1.pbe --so-pin $SOPIN --pin $PIN1 --password $PASSWORD`
- `sc-hsm-tool --import-dkek-share dkek-share-2.pbe --so-pin $SOPIN --pin $PIN1 --password $PASSWORD`
- `sc-hsm-tool --unwrap-key $CKALABEL".wrap" --key-reference 1 --pin $PIN1`
The Real Reason

This is fun!