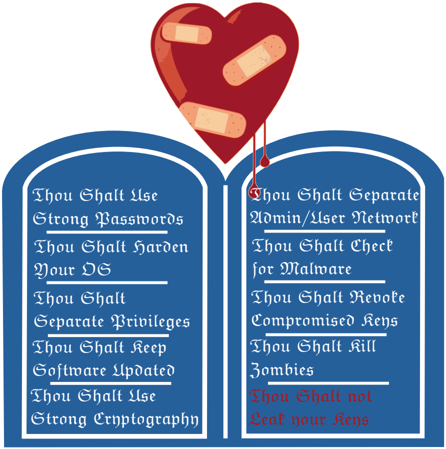
Thou Shalt not Leak your Keys:
Practical Key Privilege Separation Using Caml Crush



FOSDEM'15

R. BENADJILA
T. CALDERON
M. DAUBIGNARD

French Network and Information Security Agency

ANSSI

February 1st, 2015

Context

- Bob hosts a service, wants Alice to access it safely:
  - Hence, TLS is deployed:
    - Bob is authenticated
    - Data integrity and confidentiality
  - Bob is satisfied, Alice is safe

Context

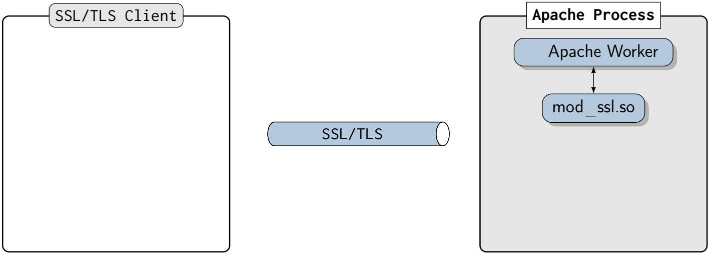
- Bob hosts a service, wants Alice to access it safely:
  - Hence, TLS is deployed:
    - Bob is authenticated
    - Data integrity and confidentiality
  - Bob is satisfied, Alice is safe
- But how safe is she?

Context

- Bob hosts a service, wants Alice to access it safely:
  - Hence, TLS is deployed:
    - Bob is authenticated
    - Data integrity and confidentiality
  - Bob is satisfied, Alice is safe
- But how safe is she?
- Heartbleed was a painful reminder:
  - Using TLS is not enough
  - Vulnerabilities in TLS stack can lead to private key leakage

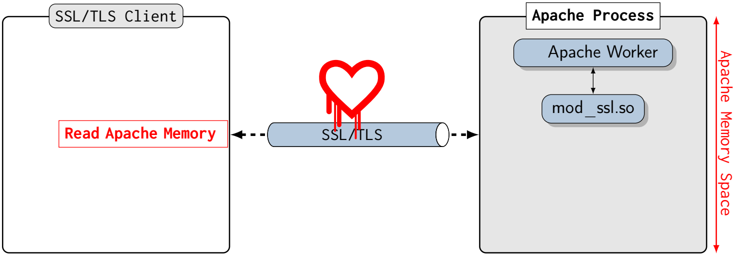
Heartbleed

- Heartbleed is a security bug that affects an implementation of a TLS protocol extension



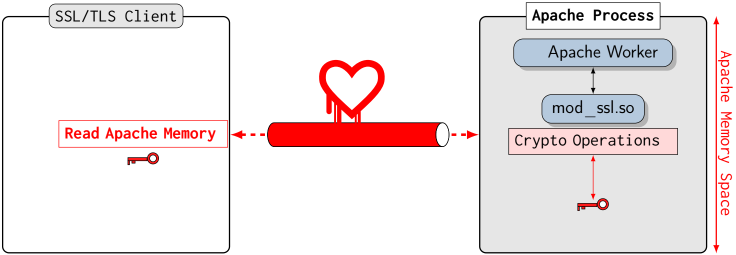
Heartbleed

- Heartbleed is a security bug that affects an implementation of a TLS protocol extension
- Simply put: using a ping feature results in a buffer over-read allowing more data than expected to be read



Heartbleed

- Heartbleed is a security bug that affects an implementation of a TLS protocol extension
- Simply put: using a ping feature results in a buffer over-read allowing more data than expected to be read
- Memory from the server process can be retrieved
  - Application data
  - TLS symetric session keys
  - Private key of the server



Consequences

- Compromission of private keys
  - MiTM of the server
  - Decryption of past TLS sessions

Consequences

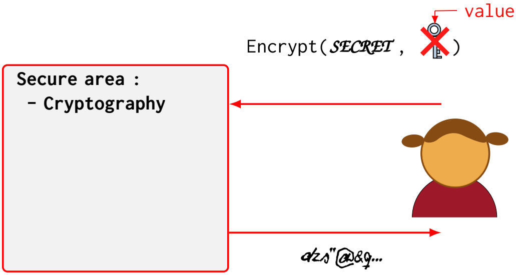
- Compromission of private keys
  - MiTM of the server
  - Decryption of past TLS sessions
- Massive renewal of enterprise and private credentials
  - Costly (think thousands of X.509 certificates to renew)
  - Painful

Did You Say Security API ?

- A security API is a programming interface which allows cryptographic operations and key management

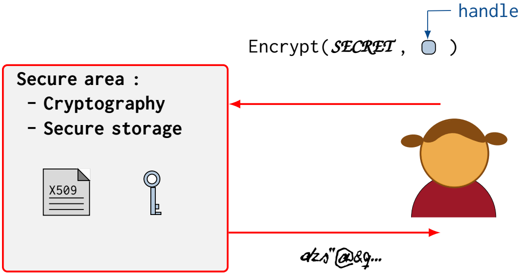
Did You Say Security API ?

- A security API is a programming interface which allows cryptographic operations and key management
- Keys must be usable without needing to know their values



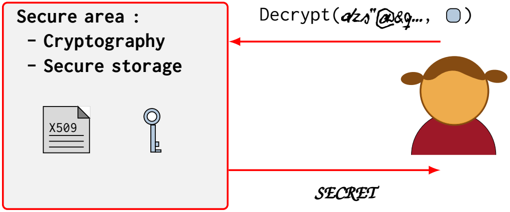
Did You Say Security API ?

- A security API is a programming interface which allows cryptographic operations and key management
- Keys must be usable without needing to know their values
- Keys are refered to using handles (pointers)



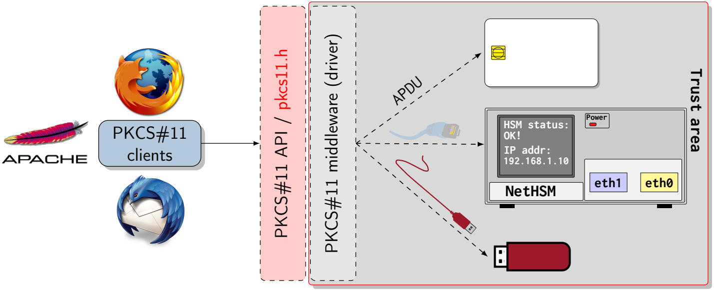
Did You Say Security API ?

- A security API is a programming interface which allows cryptographic operations and key management
- Keys must be usable without needing to know their values
- Keys are refered to using handles (pointers)



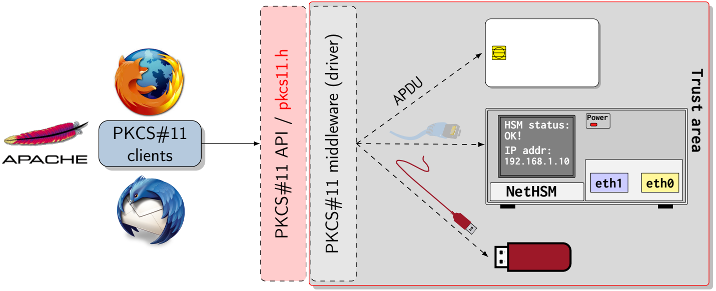
The PKCS#11 security API

- PKCS#11 = subset of Public Key Cryptography Standards initially developed by RSA labs, transfered to OASIS



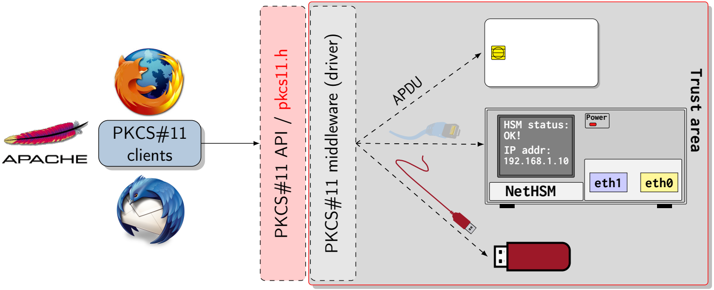
The PKCS#11 security API

- PKCS#11 = subset of Public Key Cryptography Standards initially developed by RSA labs, transfered to OASIS
  - RSA labs provides pkcs11.h
  - Manufacturers provide a shared library (''middleware'')



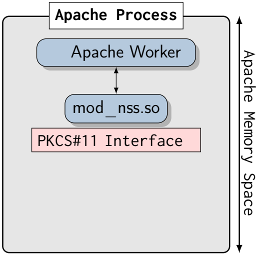
The PKCS#11 security API

- PKCS#11 = subset of Public Key Cryptography Standards initially developed by RSA labs, transfered to OASIS
  - RSA labs provides pkcs11.h
  - Manufacturers provide a shared library (‘‘middleware’’)
- The shared library handles the hardware:
  - Sends APDU sequences to smartcards (via USB, ...)
  - Sends network packets to network HSMs
  - Sends frames to USB dongles



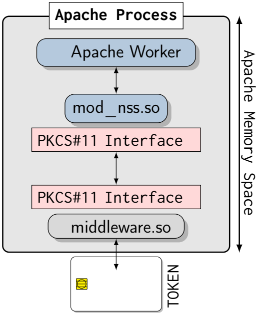
Scenario: a TLS enabled HTTP web server

- Compatible web servers can be configured to use PKCS#11



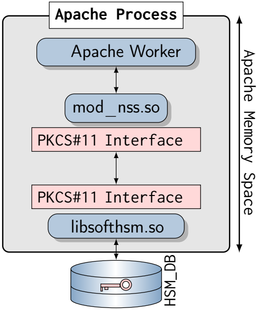
Scenario: a TLS enabled HTTP web server

- Compatible web servers can be configured to use PKCS#11
- Hardware (certified) devices offer:
  - High degree of confidence
  - Inconvenient in production environment and costly



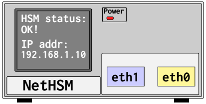
Scenario: a TLS enabled HTTP web server

- Compatible web servers can be configured to use PKCS#11
- Hardware (certified) devices offer:
  - High degree of confidence
  - Inconvenient in production environment and costly
- Software PKCS#11 devices offer:
  - Convenient to deploy and some are open-source
  - Keys are mapped in memory



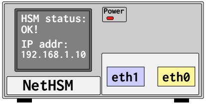
Let's sum up

	Cost	Security	Performance
HSM	✗	✓	✓



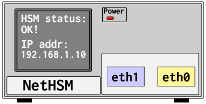
Let's sum up

	Cost	Security	Performance
HSM	✗	✓	✓
Smartcards	✓	✓	✗



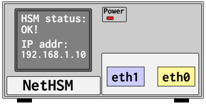
Let's sum up

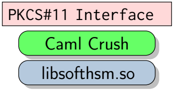
	Cost	Security	Performance
HSM	✗	✓	✓
Smartcards	✓	✓	✗
Software Tokens	✓	✗	✓



Let's sum up

	Cost	Security	Performance
HSM	✗	✓	✓
Smartcards	✓	✓	✗
Software Tokens with Caml Crush	✓	✓	✓



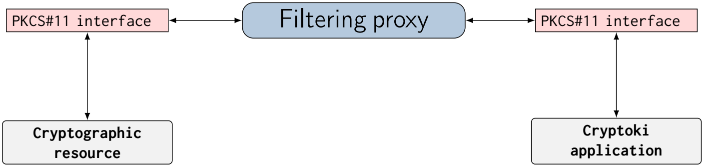


PKCS#11 API through a Proxy

- Can we use a low-cost solution such as SoftHSM?
- What if we leverage process isolation?

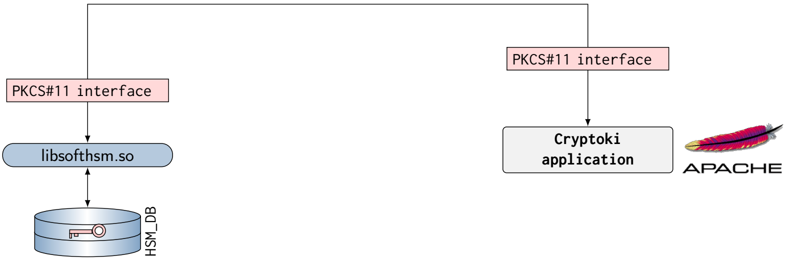
PKCS#11 API through a Proxy

- Can we use a low-cost solution such as SoftHSM?
- What if we leverage process isolation?
- Caml Crush is a PKCS#11 filtering proxy

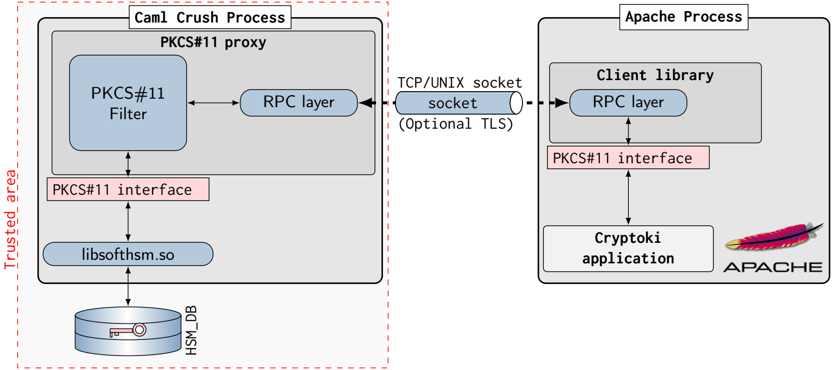


APACHE

PKCS#11 API through a Proxy

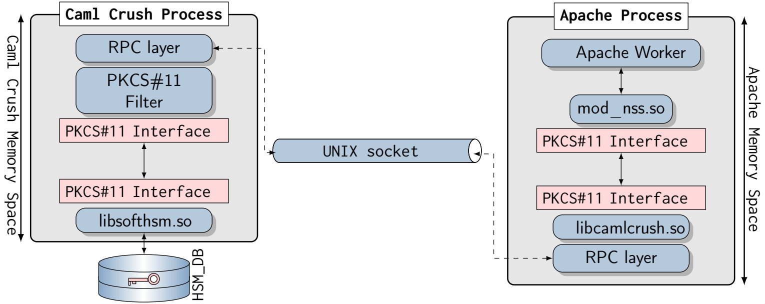


PKCS#11 API through a Proxy



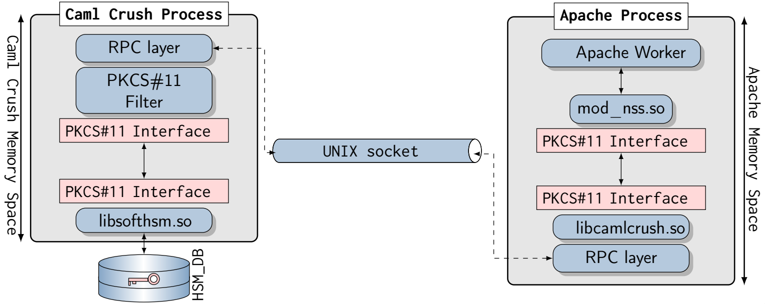
Scenario: a TLS enabled HTTP web server

- Caml Crush combined with a software PKCS#11 token



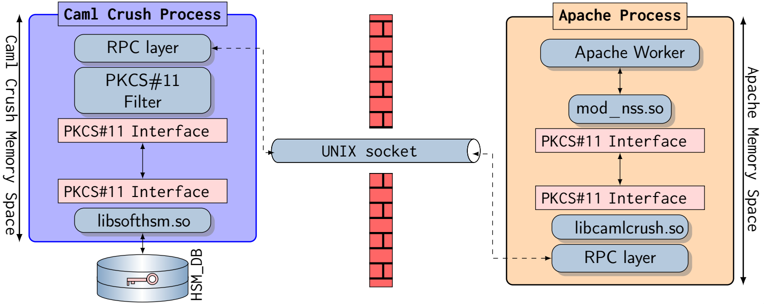
Scenario: a TLS enabled HTTP web server

- Caml Crush combined with a software PKCS#11 token
- Private key leak is avoided



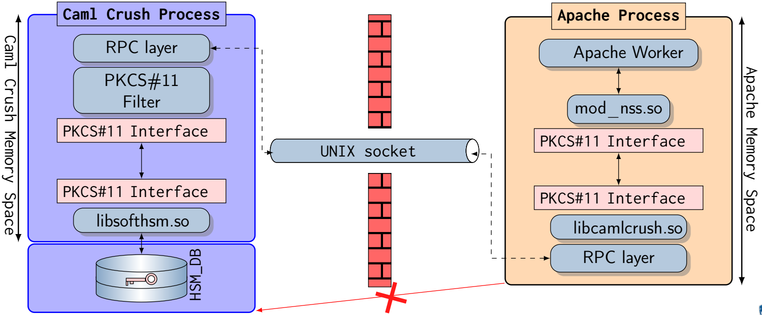
Scenario: a TLS enabled HTTP web server

- Caml Crush combined with a software PKCS#11 token
- Private key leak is avoided
- Minimal OS-level hardening required
  - ‘‘Dedicated uid/gid’’ for Apache and proxy



Scenario: a TLS enabled HTTP web server

- Caml Crush combined with a software PKCS#11 token
- Private key leak is avoided
- Minimal OS-level hardening required
  - ''Dedicated uid/gid'' for Apache and proxy
  - Coherent file permission on object database



Why use Caml Crush?

- I heard about other PKCS#11 proxies, why use yours ?

Why use Caml Crush?

- I heard about other PKCS#11 proxies, why use yours ?
- Caml Crush is security oriented

Why use Caml Crush?

- I heard about other PKCS#11 proxies, why use yours ?
- Caml Crush is security oriented
  - OCaml programming language
  - Able to sandbox itself
  - Blocks known cryptographic attacks
  - Restricts cryptographic mechanisms
  - Object filtering capabilities
  - Token read-only mode
  - . . .

Beyond Heartbleed

- Other threats?

Beyond Heartbleed

- Other threats?
- Think remote code execution

Beyond Heartbleed

- Other threats?
- Think remote code execution
  - Process memory inspection (we've seen and addressed that)
  - Use the PKCS#11 stack as an oracle
  - Could lead to private key leak

Beyond Heartbleed

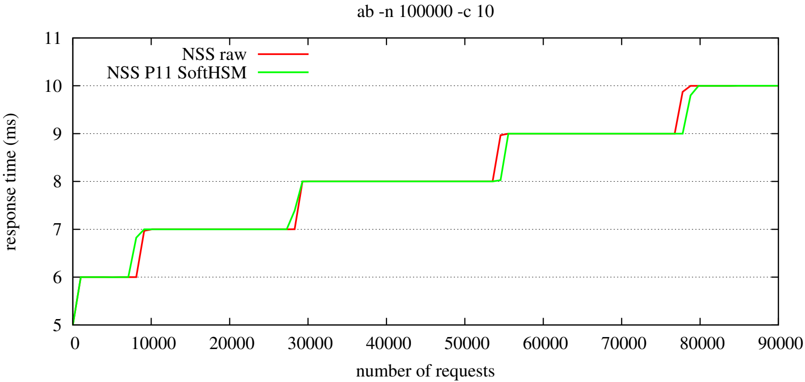
- Other threats?
- Think remote code execution
  - Process memory inspection (we've seen and addressed that)
  - Use the PKCS#11 stack as an oracle
  - Could lead to private key leak
- Caml Crush filtering engine protects from such attacks

Beyond Heartbleed

- Other threats?
- Think remote code execution
  - Process memory inspection (we've seen and addressed that)
  - Use the PKCS#11 stack as an oracle
  - Could lead to private key leak
- Caml Crush filtering engine protects from such attacks
- Other deployments
  - Transform local cryptographic tokens (PCI HSM, smartcard) into network devices
  - . . .

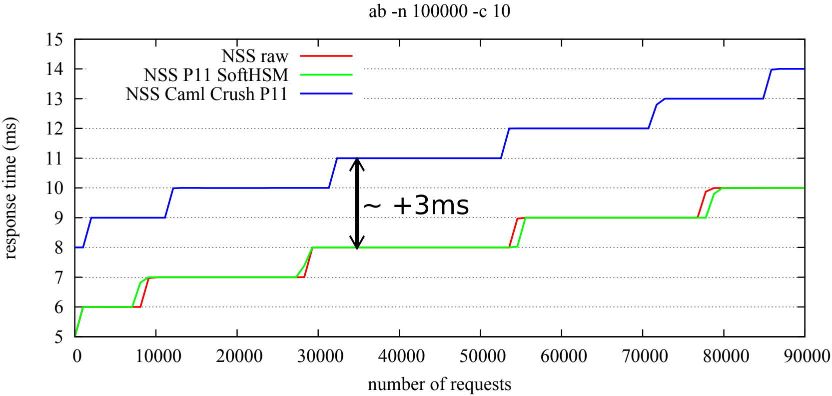
Performances

- No overhead when using plain SoftHSM



Performances

- **Reasonable** overhead with Caml Crush



Server compatibility

- Web server:
  - Apache (mod_nss[1], mod_gnutls[2])
  - NGINX (since 1.7.9[3])
- Other server applications:
  - Ex: LDAPS for OpenLDAP
  - Should work transparently if linked to GnuTLS

[1] PFS is not supported
[2] requires a patch from Nikos
[3] using OpenSC engine_pkcs11

Conclusion

- Caml Crush has benefits applicable to TLS stacks
- Caml Crush is also useful in a variety of other scenarios
- Soon in Debian Sid
- Caml Crush is open source:
  - https://github.com/ANSSI-FR/caml-crush

Conclusion

- Caml Crush has benefits applicable to TLS stacks
- Caml Crush is also useful in a variety of other scenarios
- Soon in Debian Sid
- Caml Crush is open source:
  - https://github.com/ANSSI-FR/caml-crush

Thou Shalt Ask Questions!

Compatibility Matrix

	C client		OCaml client		pkcs11proxyd		SSL/TLS
	Unix	TCP	Unix	TCP	Unix	TCP	
Linux	✓	✓	✓	✓	✓	✓	✓
FreeBSD	✓	✓	✓	✓	✓	✓	✓
Mac OS X	✗	✓	✓	✓	✓	✓	✓
Win32 (native)	✗	✓	✗	✗	✗	✗	
Win32 (cygwin)							

- Caml Crush works on Little/Big Endian platforms (even with hybrid architectures between client and server)