CONSISTENT PKCS#11

IN OPERATING SYSTEMS

IMPROVING USER EXPERIENCE AND SECURITY IN RHEL AND FEDORA

Jakub Jelen
Software Engineer
Red Hat

jjelen@redhat.com @JakujeCZ Jakuje

PRIVATE KEYS, CERTIFICATES

WHAT ARE THEY USED FOR?

PRIVATE KEYS, CERTIFICATES

WHAT ARE THEY USED FOR?

- Email signatures & decryption

PRIVATE KEYS, CERTIFICATES

WHAT ARE THEY USED FOR?

- Email signatures & decryption
- SSH authentication, remote git

PRIVATE KEYS, CERTIFICATES

WHAT ARE THEY USED FOR?

- Email signatures & decryption
- SSH authentication, remote git
- Git commit/tag signing

PRIVATE KEYS, CERTIFICATES

WHAT ARE THEY USED FOR?

- Email signatures & decryption
- SSH authentication, remote git
- Git commit/tag signing
- TLS client authentication (eGovernment, banking)

PRIVATE KEYS, CERTIFICATES

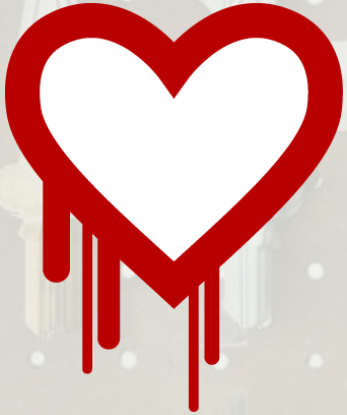
WHAT ARE THEY USED FOR?

- Email signatures & decryption
- SSH authentication, remote git
- Git commit/tag signing
- TLS client authentication (eGovernment, banking)
- More secure password replacement

WHERE ARE THEY STORED?

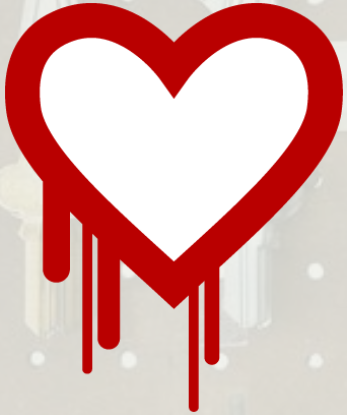
- Hard drive
- Computer memory
- Backup in cloud

ARE THEY SECURE?



DIRTY COW

ARE THEY SECURE?



DIRTY COW

ZERO DAY EXPLOITS?

Last year

Last year

Last year

```
$ pkcs11-tool --read-object --id 01 --type cert \
    --output-file cert.der
```

```
$ pkcs11-tool --sign --id 01 --mechanism RSA-PKCS --login \
    --input-file data --output-file data.sig
```

```
$ ssh-keygen -D /usr/lib64/pkcs11/opensc-pkcs11.so

$ ssh -I /usr/lib64/pkcs11/opensc-pkcs11.so example.com
```

Last year

```
$ pkcs11-tool --read-object --id 01 --type cert \
    --output
```

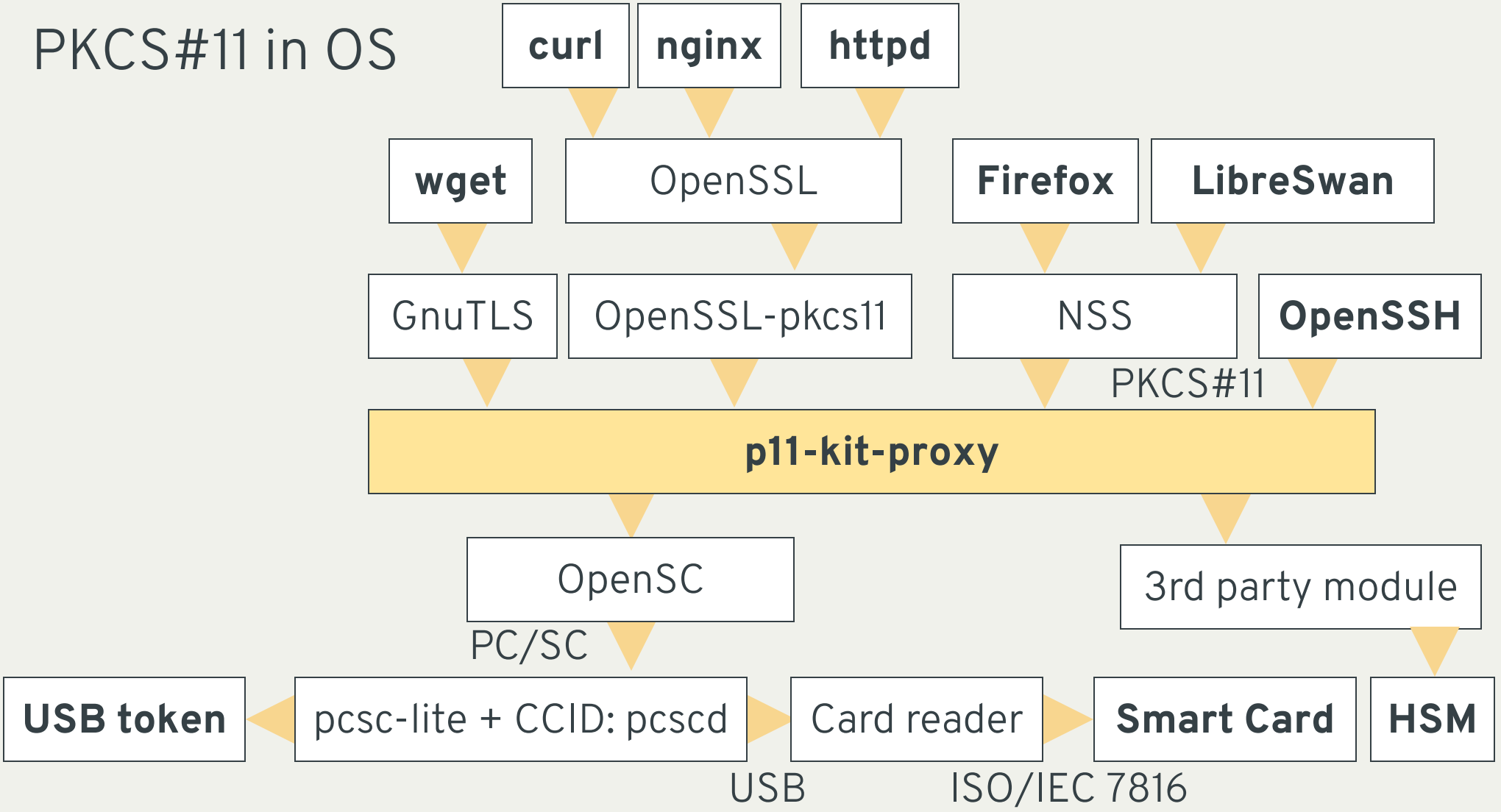
```
kcs --login \
```

DEDICATED HARDWARE IN OS IS NOT EASY

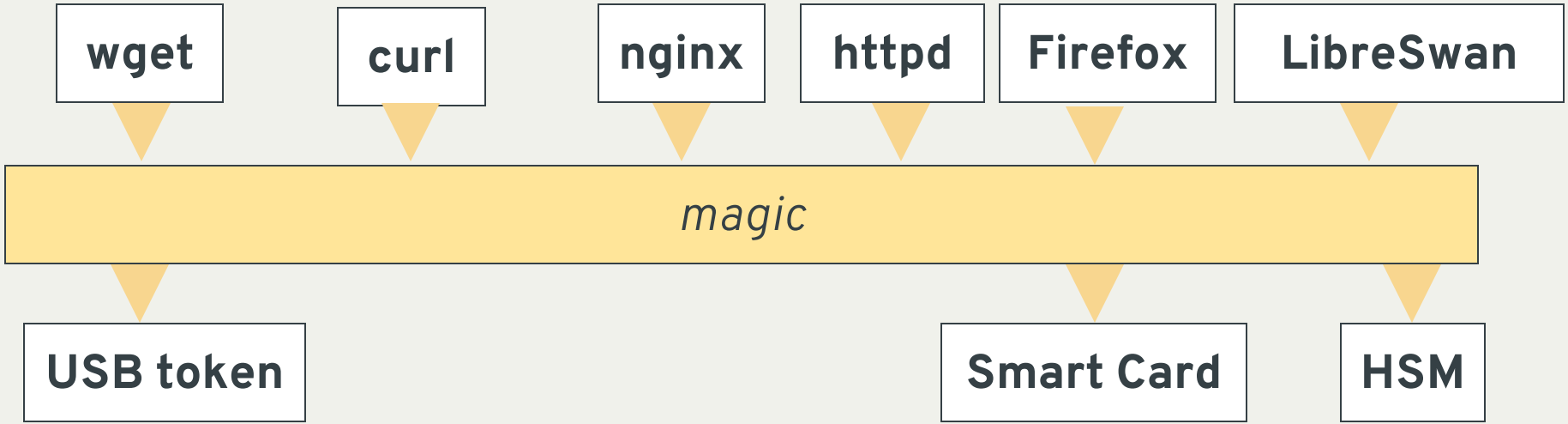
```
$ ssh-keygen -D /usr/lib64/pkcs11/opensc-pkcs11.so

$ ssh -I /usr/lib64/pkcs11/opensc-pkcs11.so example.com
```

PKCS#11 in OS


curl
nginx
httpd
wget
OpenSSL
Firefox
LibreSwan
GnuTLS
OpenSSL-pkcs11
NSS
OpenSSH
PKCS#11
p11-kit-proxy
OpenSC
3rd party module
PC/SC
USB token
pcsc-lite + CCID: pcscd
Card reader
Smart Card
HSM
USB
ISO/IEC 7816

User expectations



AGENDA

- PKCS#11
- Usability improvements
  - PKCS#11 URI
  - p11-kit-proxy
- Application support
  - OpenSSH
  - HTTPS clients & servers
  - Firefox
  - Your application?
- Further work

PKCS#11

- *Open Standard for cryptographic tokens controlling authentication information (personal identity, cryptographic keys, certificates, digital signatures, ...)*
- PKCS#11 module: implementation of PKCS#11 interface providing access to cryptographic tokens
- low-level C API

CONSISTENT PKCS#11

system-wide consistency for usage and configuration

CONSISTENT PKCS#11

system-wide consistency for usage and configuration

```
$ p11tool --list-all "pkcs11:manufacturer=piv_II;token=SSH%20key"
```

```
$ pkcs11-tool --read-object --id 01 --type cert \
    --output-file cert.der
```

```
$ ssh -I /usr/lib64/pkcs11/opensc-pkcs11.so example.com
```

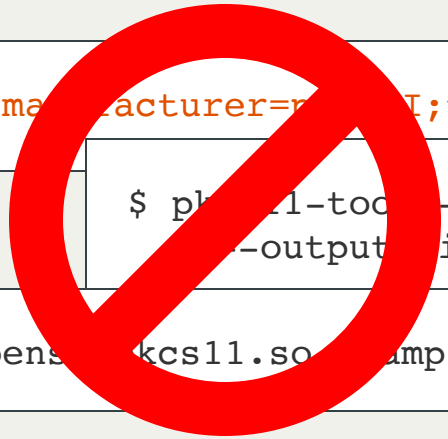
CONSISTENT PKCS#11

system-wide consistency for usage and configuration

```
$ p11tool --list-all "pkcs11:ma   facturer=n    I;token=SSH%20key"
```

```
$ p   l-too   --read-object --id 01 --type cert \
   -output   ile cert.der
```

```
$ ssh -I /usr/lib64/pkcs11/opens   kcs11.so   ample.com
```



PKCS#11 URI (RFC 7512)

- Strongest and simplest expression
- **pkcs11:** uri scheme -- distinguishable from filenames
- Uniquely identifies each object in the system
- Non-mandatory filtering by PKCS#11 attributes
- Can provide also PIN or pkcs11 module

pkcs11:

pkcs11:manufacturer=piv_II;token=SSH%20key;**id=%04**;object=PIV%20AUTH%20pubkey;**type=private**

P11-KIT

- PKCS#11 modules exposed to users
- System-wide registry of PKCS#11 modules
- Automatically loaded by applications
- PKCS#11 modules registered in one place:
  - System and 3rd party

```
$ cat /usr/share/p11-kit/modules/opensc.module
module: opensc-pkcs11.so
```

APPLICATION SUPPORT

How does it work?

OPENSSH CLIENTS (WAS)

- Listing keys

```
$ ssh-keygen -D /usr/lib64/pkcs11/opensc-pkcs11.so
ssh-rsa AAAAB3Nza[...]751SVdOhUaBiTXGiClQ==
  /usr/lib64/pkcs11/opensc-pkcs11.so
```

- Public key authentication

```
$ ssh -i /usr/lib64/pkcs11/opensc-pkcs11.so example.com
Enter PIN for 'SSH key':
```

- Filtering keys
  - N/A

OPENSSH CLIENTS (WAS)

- Listing keys

```
$ ssh-keygen -D /usr/lib64/pkcs11/opensc-pkcs11.so
ssh-rsa AAAAB3Nza[...]751SVdOhUaBiTXGiClQ==
  /usr/lib64/pkcs11/opensc-pkcs11.so
```

- Public key authentication

```
$ ssh -i /usr/lib64/pkcs11/opensc-pkcs11.so example.com
Enter PIN for 'SSH key':
```

- Filtering keys
  - N/A

OPENSSH CLIENTS (WAS)

- Listing keys

```
$ ssh-keygen -D /usr/lib64/pkcs11/opensc-pkcs11.so
ssh-rsa AAAAB3Nza[...]751SVdOhUaBiTXGiClQ==
  /usr/lib64/pkcs11/opensc-pkcs11.so
```

- Public key authentication

```
$ ssh -i /usr/lib64/pkcs11/opensc-pkcs11.so example.com
Enter PIN for 'SSH key':
```

- Filtering keys
  - N/A

OPENSSH CLIENTS (WAS)

- Listing keys

```
$ ssh-keygen -D /usr/lib64/pkcs11/opensc-pkcs11.so ☹
ssh-rsa AAAAB3Nza[...]751SVdOhUaBiTXGiClQ==
  /usr/lib64/pkcs11/opensc-pkcs11.so
```

- Public key authentication

```
$ ssh -i /usr/lib64/pkcs11/opensc-pkcs11.so example.com ☹
Enter PIN for 'SSH key':
```

- Filtering keys
  - N/A ☹

OPENSSH CLIENTS

- Listing keys

```
$ ssh-keygen -D pkcs11:
ssh-rsa AAAAB3Nza[...]751SVdOhUaBiTXGiClQ==
  pkcs11:id=%03;[...]?module-path=/usr/lib64/p11-kit-proxy.so
ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzd[...]SStk3J0hkYnnsM=
```

- Public key authentication

```
$ ssh -i pkcs11: example.com
Enter PIN for 'SSH key':
```

- Filtering keys

```
$ ssh -i pkcs11:id=%02 localhost
Enter PIN for 'SSH key':
```

OPENSSH CLIENTS 2

- Using ssh-agent

```
$ ssh-add pkcs11:id=%02
Enter passphrase for PKCS#11:
Card added: pkcs11:id=%02
$ ssh-add -l
521 SHA256:5BrE5wevULd[...]+kF5hA9X8 ECDSA jjelen (ECDSA)
$ ssh example.com
```

- Configuration

```
$ cat ~/.ssh/config
IdentityFile "pkcs11:id=%01?module-path=/usr/lib64/opensc-pkcs11.so"
```

HTTPS CLIENTS

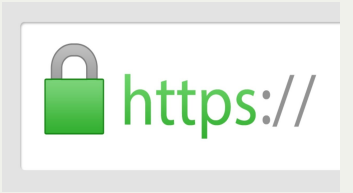
- wget

```
$ wget --certificate 'pkcs11:id=%01;type=cert' \
  --private-key 'pkcs11:id=%01;type=private' https://example.com/
```

- curl

```
$ curl --cert 'pkcs11:id=%01;type=cert' \
  --key 'pkcs11:id=%01;type=private' https://example.com/
```

HTTPS SERVERS



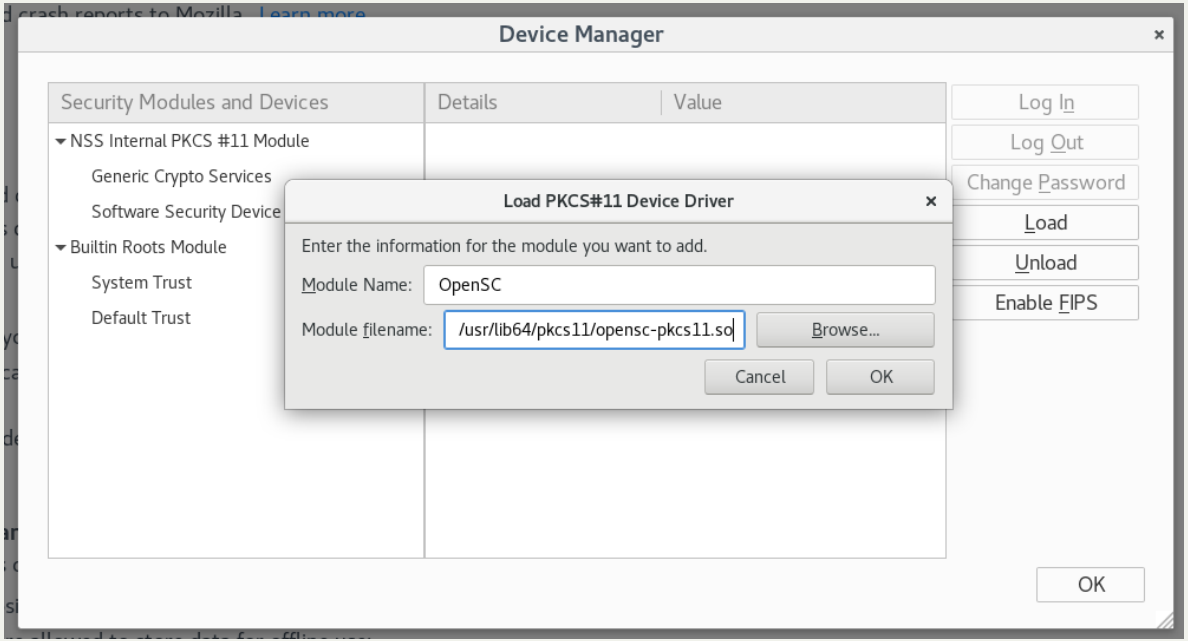
- httpd configuration file

```
SSLCertificateFile pkcs11:id=%01;type=cert
SSLCertificateKeyFile pkcs11:id=%01;type=private
```

- nginx configuration file

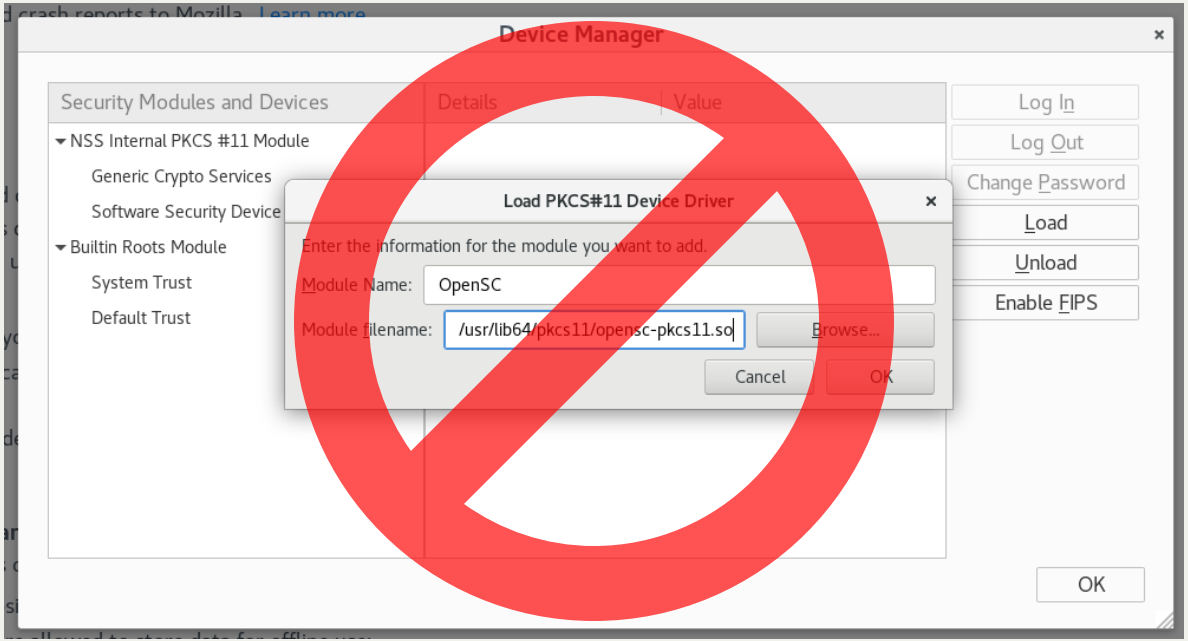
```
# ssl_certificate # does not work
ssl_certificate_key "engine:pkcs11:id=%01;type=private";
```

FIREFOX


Device Manager
Security Modules and Devices
Details
Value
NSS Internal PKCS #11 Module
Generic Crypto Services
Software Security Device
Builtin Roots Module
System Trust
Default Trust
Log In
Log Out
Change Password
Load
Unload
Enable FIPS
OK
Load PKCS#11 Device Driver
Enter the information for the module you want to add.
Module Name:
OpenSC
Module filename:
/usr/lib64/pkcs11/opensc-pkcs11.so
Browse...
Cancel
OK

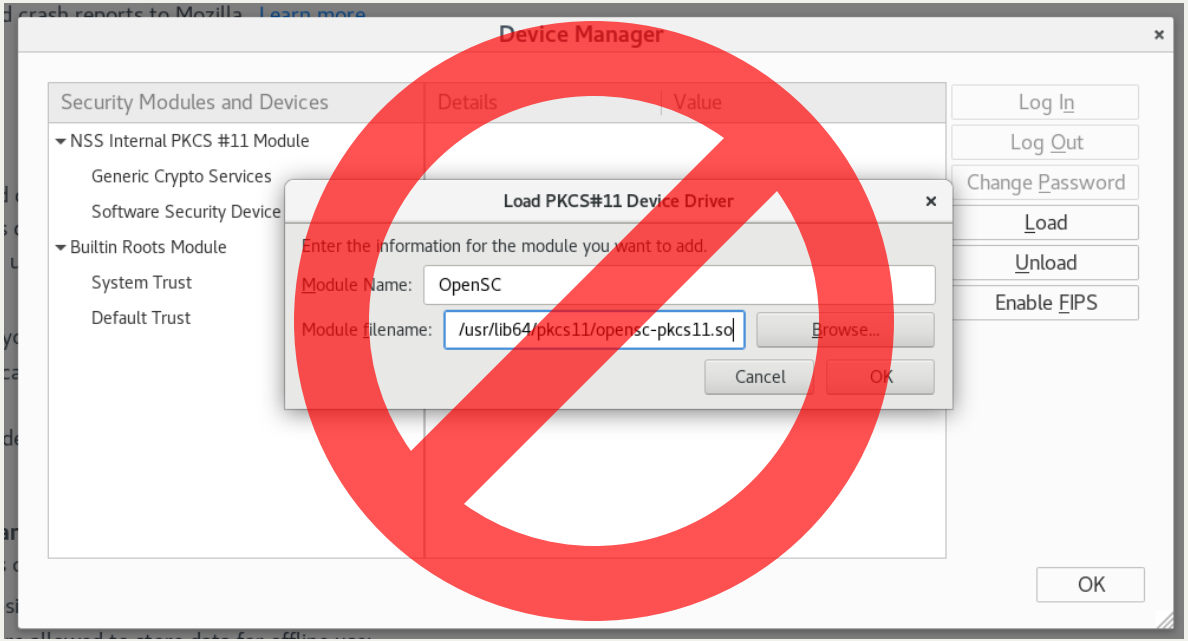
FIREFOX

- No more adding PKCS#11 modules



FIREFOX

- No more adding PKCS#11 modules



- Just works

YOUR OTHER APPLICATION?

- might already work
- high-level crypto applications
- p11-kit
  - Available tokens
- PKCS#11 URI
  - Identify objects
  - Handled p11-kit

TRY IT AT HOME

- TPM2.0
  - any computer from last years
  - alternative to storing private keys on hard drive
  - tied to specific machine
  - TCG provides PKCS#11 module (tpm2-pkcs11)
- SoftHSM
  - PKCS#11 module
  - data stored on filesystem
  - integrated in p11-kit

SUMMARY

- Security bugs in processors, OS, software
- Smart cards and HSMs to store secrets in HW
- Consistent identification using PKCS#11 URI
- System-wide registration in p11-kit-proxy
- Support for most important system applications

QUESTIONS?

jjelen@redhat.com @JakujeCZ Jakuje