### **Security primitives II**

Secure channel Secure storage

...

PA193 – Secure coding

Petr Švenda Zdeněk Říha Faculty of Informatics, Masaryk University, Brno, CZ



Centre for Research on Cryptography and Security

www.fi.muni.cz/crocs

### **Security primitives**

- Secure channel
  - Communication
- Secure envelope

   Data protection
- Secure storage
  - Storage
- Use standard, commonly used mechanisms
  - It is very difficult to create your own mechanisms that will be as secure as the standard ones

#### **Secure channel**

 secure channel is a way of transferring data that is resistant to overhearing and tampering

Source: https://en.wikipedia.org/wiki/Secure\_channel

- Examples
  - Secure Messaging (smartcards)
    - ISO 7816-4
    - Open Platform / Global Platform
  - SSL/TLS
  - IPSEC
  - VPN

#### Secure channel

- Authentication of parties
- Confidentiality of data
- Integrity of data
- Based on:
  - Symmetric crypto
    - E.g. encryption + MAC, 4 symmetric keys shared
      - 2 for each direction
  - Asymmetric crypto

### Case study SSL/TLS

- Let's look at the failure of SSL/TLS in more details – Read more at:
  - http://www.ieee-security.org/TC/SP2013/papers/4977a511.pdf
- Basic knowledge of SSL/TLS expected
  - Mandatory server authentication
  - Optional client authentication
  - Authentication based on X.509 certs and private key
  - PKI infrastructure to validate certs needed
  - Confidentiality and integrity provided
  - Non-repudiation not provided

### Weaknesses in Crypto Primitives

- SSL/TLS started with 40/56 bit symmetric keys
- DES, RC2, RC4
- US export regulation
- Slow changes, backward compatibility
- Still possible to see certs with unsecure parameters
  - Based on RSA-512 (!)
    - By the way Google was using RSA-1024 until Nov 2013
  - Based on md5
    - Collision attack on certs demonstrated

### **PRNG problems**

- Netscape browser prior 1.22 relied on PRNG generating weak keys
- Debian problems with OpenSSL

#### **Remote timing attacks**

- Against SSL servers using optimized RSA decryption based on OpenSSL
  - And that optimized decryption was default in OpenSSL prior 0.9.7b
- the long term secret of the server was leaking during the SSL/TLS handshake

#### CRତCS

#### **Protocol attacks**

- Ciphersuite downgrade
   In SSL 2.0
- SSL Version downgrade
  - If clients misinterpret higher version error and try to continue with a lower protocol version
- Renegotiation attack
  - Renegotiate security related parameters

#### **Trust model - X.509 Certificates**

- Hostname Validation
  - Do not skip the hostname validation
  - Study: over 1000 out of 13500 Android applications do not validate the hostname
  - Read more at:
    - http://www2.dcsec.uni-hannover.de/files/android/p50-fahl.pdf
  - [Analysis of Android SSL (in)security]
- Parsing attacks
  - Binary 0 in CN
  - Something.com0other.com

#### **Trust model - X.509 Certificates**

- Anchoring trust
  - Web browsers include +-150 trust points from +-50 organizations
    - CA Compromise (DigiNotar)
    - The power of goverments over CAs
    - Transitivity of trust (basicConstrains CA:TRUE)
      - This flag must be checked otherwise anybody could be validated as any web site
      - MS CryptoAPI did not
      - Apple iOS did not

### **Trust model - X.509 Certificates**

- Revocation
  - Blocking the certificates
    - OCSP or CRL download

#### http vs. https

- Stripping TLS
  - relay https pages over http
- A tool called SSLstrip exists
  - a man-in-the-middle attack
- HTTP Strict Transport Security (HSTS)
  - web security policy mechanism where a web server declares that only secure HTTPS connections can be used.
  - the policy is communicated via a HTTP response header called "Strict-Transport-Security".

#### CROCS

#### Secure storage: how to keep secrets secret

- Secret data
  - Symmetric encryption keys
  - Asymmetric Private keys
  - Passwords
  - ...
- Storing secrets in SW
  - Completely securely IMPOSSIBLE
    - Debugging
    - Paging memory to files
    - Malicious administrators
    - ...
- Storing secrets in HW (HSM, Smartcards, ...)

#### **Secure storage**

- MS Windows: Data Protection API (DPAPI)
- Mac OS X: Keychain Services API
- Linux GNOME: gnome-keyring-manager
- Linux KDE: kwallet

#### **Need to store secrets?**

- Password hashing
  - Crypt (old)
  - Md5 (old)
  - Sha1,2
- Salting
  - Attackers cannot store pre-computed hash values
  - 64 bit salt means that an attacker would have to potentially prepare 2<sup>64</sup> more hash values

#### CRତCS

## **Deriving keys from passwords**

- Make it slow
  - Avoid dictionary attacks!
- PKCS#5
  - Password-Based Key Derivation Function #1 (PBKDF1)
  - Password-Based Key Derivation Function #2 (PBKDF2)
  - Hash the password 100x 1000x
- In MS Crypto API use CryptDeriveKey
  - With similar functionality

[CryptoWorld 11-12/2013]

#### CRତCS

#### **Protecting secrets in Windows**

- Data Protection API (DPAPI)
  - CryptProtectData, CryptUnprotectData
- Data available to user
  - Bound with user account, available on multiple machines but not on other accounts
- Data available to machine
  - Available to any user at the machine, not available at other machines
  - Use CRYPTPROTECT\_LOCAL\_MACHINE flag

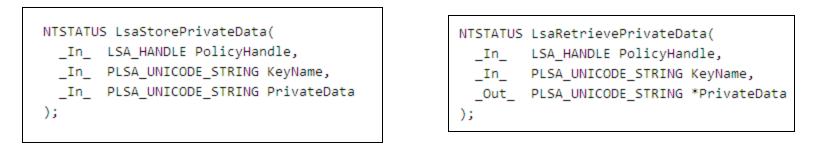
#### CROCS

#### **Protecting secrets on Windows**

- DAPI does not provide storage (only encryption/decryption)
- You have to manage storage yourself
  - Be careful to protect the encrypted data with correct ACLs in files/registry
- Any application running on the USER can decrypt the secrets!
- If you do not like this, use pOptionalEntropy field
   To protect your secrets with another secret <sup>(i)</sup>

### LSA interface (old stuff)

- "The Local Security Authority (LSA) is a protected subsystem of Windows that maintains information about all aspects of local security on a system."
- LSA secrets:



NTSTATUS	LsaOpenPolicy(
_In_	PLSA_UNICODE_STRING SystemName,
_In_	PLSA_OBJECT_ATTRIBUTES ObjectAttributes,
_In_	ACCESS_MASK DesiredAccess,
_Inout_	PLSA_HANDLE PolicyHandle
);	
1	

#### LSA interface

- LSA secrets:
  - Local data
    - Can be read only at the machine storing data (L\$)
  - Global data
    - Created on domain controller and replicated (G\$)
  - Machine data
    - Can be accessed only by OS (M\$)
  - Private data
    - Can be used by your application

### LSA vs. DPAPI

#### The Differences Between LSA Secrets and DPAPI

You should be aware of a number of differences between these two data protection technologies. They include the following:

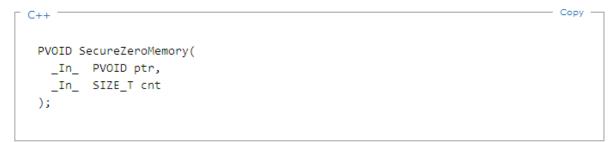
- LSA secrets are limited to 4096 objects; DPAPI is unlimited.
- LSA code is complex; DPAPI code is simple!
- DPAPI adds an integrity check to the data; LSA does not.
- LSA stores the data on behalf of the application; DPAPI returns an encrypted blob to the application, and the application must store the data.
- To use LSA, the calling application must execute in the context of an administrator. Any user—ACLs on the encrypted data aside—can use DPAPI.

Source: Writing secure code, 2<sup>nd</sup> edition

#### CROCS

### Managing secrets in memory

- ZeroMemory()
  - Macro using memset
- Compiler optimizations can remove the call of the function!!!
- Use SecureZeroMemory() instead



#### Parameters

ptr [in]

A pointer to the starting address of the block of memory to fill with zeros.

cnt [in]

The size of the block of memory to fill with zeros, in bytes.

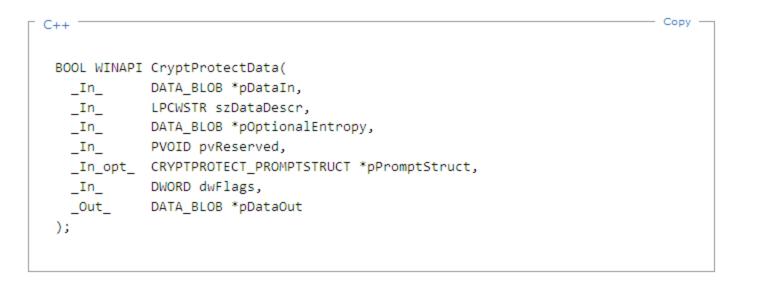
#### CRତCS

### **Managing secrets in Memory**

#### CryptProtectMemory() and CryptUnprotectMemory()

The **CryptProtectData** function performs encryption on the data in a **DATA\_BLOB** structure. Typically, only a user with the same logon credential as the user who encrypted the data can decrypt the data. In addition, the encryption and decryption usually must be done on the same computer. For information about exceptions, see Remarks.

#### Syntax



Source: MSDN

www.fi.muni.cz/crocs

#### CRତCS

## **Locking Memory to Prevent Paging**

- To keep your sensitive data in memory only (and not in a paging file)
- Affects performance
- Does not prevent dumping memory to disk when hibernating (or crash dump file)
- Does not prevent a debugger to read the memory
- Lock memory before storing the secrets in the memory
- VirtualLock()
- AllocateUserPhysicalPages()

#### CROCS

### Windows credentials manager

- Asking the user for credentials
- CredUIPromptForCredentials()
- CredUIPromptForWindowsCredentials()
  - From Vista up

Connect to Cr	edentialsDialogTestHarn ? 🔀
Welcome back to User name:	CredentialsDialogTestHarness
Password:	•••

Windows Security		
Please enter your username and password. Since this is a sample the credentials won't be used for anything, so you can enter anything you like.		
User name Password		
Remember my credentials		
OK Cancel		

#### www.fi.muni.cz/crocs

#### **Protect data**

- For secret keys use
  - Pkcs#8
  - Pkcs#12 (pfx)
- For digital signatures use
  - CMS (PKCS#7)
  - Secure email
    - S/MIME
      - Based on X.509 certificates
      - Transparent vs. opaque signing
    - PGP

#### PKCS#8

- Format for storing private key
- Independent on private key algorithm
- Key can be encrypted
- File suffix ".pkcs8"

PrivateKeyInfo ::= SEQUENCE { version Version, privateKeyAlgorithm AlgorithmIdentifier {{PrivateKeyAlgorithms}}, privateKey PrivateKey, attributes [0] Attributes OPTIONAL }

### **PKCS#12**

- Collection of cryptographic objects
- Privacy/confidentiality
  - Public key privacy mode: encrypted by a public key
  - Password privacy mode: encrypted by a symmetric key derived from username and password
- Integrity modes
  - Public key integrity mode: digital signature
  - Password protection mode: MAC based on password
- File suffix ".p12", ".pfx".

#### **PKCS#12**

- "SafeContents" is made up of "SafeBags"
- SafeBag types:
  - KeyBag: PKCS#8 private key
  - PKCS8ShroudedKeyBag: private key, which has been shrouded (encrypted) in accordance with PKCS #8
  - CertBag: certificate (X.509, SDSI Simple Distributed Security Infrastructure)
  - CRLBag: CRL (X.509)
  - SecretBag: any other secret of a user

### PKCS#7 / CMS

- Encapsulated content
- Content types:
  - Data (any plaintext)
  - Signed Data (digital signature based on X.509 certs)
  - Enveloped Data (encrypted data)
    - key transport: symmetric key encrypted by the recipient's public key;
    - key agreement: pairwise symmetric key created using the recipient's public key and the sender's private key
    - symmetric key-encryption keys: using a previously distributed key
    - passwords: key is derived from a password
  - Authenticated data (MAC + MAC key)

# **REDUNDANCY/DUPLICATION**

I

#### CRତCS

## **Redundancy/duplication**

- Repeating elements
  - Sometimes useful e.g. RAID 1
- In database structures and programming redundancy is seen as negative aspect
- Database systems
  - Database normalization (1NF, 2NF, 3NF, ...)
- Programming
  - Avoid repeating code or data

CROCS

#### Redundancy

- Redundant can be:
  - Data
  - Туре
  - Algorithm

I

### **Data duplication**

```
public Position WalkNorth()
 var player = GetPlayer();
 player.Move("N");
 return player.NewPosition;
}
public Position WalkSouth()
 var player = GetPlayer();
 player.Move("S");
 return player.NewPosition;
public Position WalkEast()
 var player = GetPlayer();
 player.Move("E");
 return player.NewPosition;
public Position WalkWest()
 var player = GetPlayer();
 player.Move("W");
 return player.NewPosition;
```



public Position Walk(string direction)

```
var player = GetPlayer();
player.Move(direction);
return player.NewPosition;
```

Source: http://simpleprogrammer.com/2012/05/27/types-of-duplication-in-code/

## **Type duplication (C#)**

```
public int FindIntMatch(int i)
{
  var match = (int)container.Get(i);
  return match;
}
public string
FindStringMatch(string s)
{
  var match =
  (string)container.Get(s);
  return match;
}
```



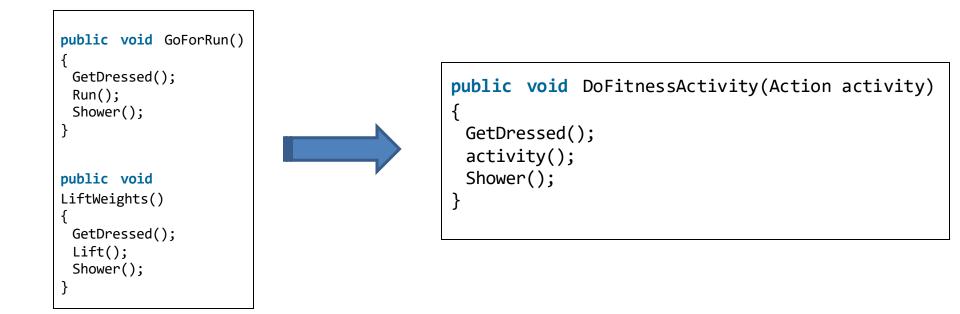
public T FindMatch(T t)
{

}

var match = (T)container.Get(t);
return match;

Source: http://simpleprogrammer.com/2012/05/27/types-of-duplication-in-code/

## **Algorithm duplication**



Source: http://simpleprogrammer.com/2012/05/27/types-of-duplication-in-code/

### **Code Duplication**

- Code duplication = sequence of source code that occurs more than once, either within a program or across different programs owned or maintained by the same entity.
- Also known as (code) clones.
- Solution
  - Clone detection + Code refactoring

Source: https://en.wikipedia.org/wiki/Duplicate\_code

#### Example

**extern** int array\_a[]; **extern** int array\_b[];

int sum\_a = 0;

```
for (int i = 0; i < 4; i++)
sum_a += array_a[i];</pre>
```

```
int average_a = sum_a / 4;
```

int sum\_b = 0;

```
for (int i = 0; i < 4; i++)
  sum_b += array_b[i];</pre>
```

```
int average_b = sum_b / 4;
```

int calc\_average\_of\_four(int\* array) int sum = 0; for (int i = 0; i < 4; i++) sum += array[i];return sum / 4; **extern** int array1[]; extern int array2[]; int average1 = calcAverage(array1); int average2 = calcAverage(array2);

Source: https://en.wikipedia.org/wiki/Duplicate\_code