Attack Certificate-based Authentication System and Microsoft InfoCard

Xu Hao
windknown@hotmail.com
Agenda

• Background
• Basic Knowledge
• Steal Local Certificate
• Steal Information Card
• Attack Smart Card
• Demo
• Conclusion
Background

• Old password authentication system
  – Advantage: Easy implement, widely used
  – Attack method
    • Weak password, brute force cracking, sniffer
    • Key logger: user mode / kernel mode
      – SetWindowsHookEx / GetAsyncKeyState …
      – Hook KeyboardClassServiceCallback / i8042prt / IDT …
  – Hard to prevent key logger problem totally
    • Soft keyboard, protection plug-in, random screen number
  – Weakness key point: password can be easily copied
Background

• Certificate-based authentication system
  – Based on cryptography algorithms
  – Authentication process includes a series of cryptography operations
  – Instead of password, user private key becomes the trust root of authentication system
  – Become popular
    • Login your computer
    • Online bank transaction
    • Login VPN
Background

• Microsoft InfoCard
  – New feature in Microsoft .Net Framework 3.0, also called CardSpace
  – Four aspects
    • Support for any digital identity system
    • Consistent user control of digital identity
    • Replacement of password-based Web login
    • Improved user confidence in the identity of remote applications
Agenda

• Background
• **Basic Knowledge**
• Steal Local Certificate
• Steal Information Card
• Attack Smart Card
• Demo
• Conclusion
Basic Knowledge

- **Usage of cryptography**
  - **Symmetric (AES, DES, 3DES, RC4)**
    - Use the same key to encrypt and decrypt
    - Operate in blocks, ECB / CBC mode
    - Fast, can be used to encrypt data flow
  - **Asymmetric (RSA, ECC)**
    - Need a pair of key to complete encryption and decryption
    - Slow, is generally used to sign hash value or encrypt key value
    - Private key must be kept by user, public key can be spread
    - Sign: use private key to encrypt Verify: use public key to decrypt
  - **Hash (MD5, SHA1)**
    - No key is needed
    - Any length input data -> certain length output data, unrecoverable
    - Different input -> different output
    - Used to check the integrity of data
Basic Knowledge

• PKI (Public Key Infrastructure)
  – Realize the generation, management, storage, distribution and revocation of key pairs and certificate
  – Key concepts of PKI
    • Certificate
    • Certificate Revocation List (CRL)
    • Certificate Authority (CA)
  – Security service
    • Privacy, Identity verification, Integrity
Basic Knowledge

• Certificate
  – Help to distribute public key in safe
  – Trusted third party signature -> prevent public key from being modified
Basic Knowledge

- **ASN.1**
  - describes a data format for data expressing, encoding, transmission and decoding
    - Identifier octets | Length octets | Contents octets | End-of-contents octets
  - **Length**
    - Less than 0x80, one byte represents the length
    - Over 0x80, bit8 of the first byte is set as 1, bit1-bit7 represents the number of following bytes which shows the length
  - **Important classed**
    - INTEGER/OCTET STRING/OBJECT IDENTIFIER/SEQUENCE
  - **OID encoding**
    - The first two numbers will be combined to one: Num1*40 + Num2
    - If a number is larger than 127, several bytes will be needed to record it. The low 7 bits of every byte is used and bit8 indicates if it is the last byte. For example, 840 (6*128+72) , after encoding: 0x86 0x48
Basic Knowledge

- **X.509**

  - Certificate includes public key, CER files

  Certificate ::= SIGNED { SEQUENCE {
    version [0] Version DEFAULT v1,
    serialNumber CertificateSerialNumber,
    signature AlgorithmIdentifier,
    issuer Name,
    validity Validity,
    subject Name,
    ...
  }

<table>
<thead>
<tr>
<th>Version</th>
<th>Serial Number</th>
<th>Signature</th>
<th>Issuer</th>
<th>Validity</th>
<th>Subject</th>
</tr>
</thead>
<tbody>
<tr>
<td>v3</td>
<td>2f 7d 0b 11 ee 1d d0 86 46 57 f1 e0 82 3d b3 af</td>
<td>1.2.840.113549.1.1.5 (SHA1RSA OID)</td>
<td>2.5.4.3 (Common Name OID)</td>
<td>For POC2009</td>
<td></td>
</tr>
</tbody>
</table>
Basic Knowledge

• PKCS#12
  – Certificate includes public / private key, PFX files
  – PFX can be protected by password
  – Some important information

  OID: 1.2.840.113549.1.9.20
  Name: d7782fa9-25cc-4014-95d1-91d4d72ff7ce

  OID: 1.3.6.1.4.1.311.17.1
  Name: Microsoft Enhanced Cryptographic Provider v1.0
Agenda

• Background
• Basic Knowledge
• Steal Local Certificate
• Steal Information Card
• Attack Smart Card
• Demo
• Conclusion
Steal Local Certificate

- **CryptoAPI**
  - Microsoft defined API for crypto operation and key management
  - CryptXXXX APIs

- **CSP (Cryptographic Service Providers)**
  - Real service providers, implement key functions
  - Developed by third party, register to OS, export CryptoSPI
  - Container -> Signature Key & Exchange Key
Steal Local Certificate

• CNG (Cryptography Next Generation)
  – Appear in Vista, used to replace CryptoAPI
  – For compatible, Vista supports CNG & CryptoAPI
  – Feature
    • Support more algorithms
    • Support for adding algorithm developed by third party
    • Security improvement, isolation process operation
  – Key concepts
    • Cryptographic Primitives Provider
      – Symmetric, asymmetric, hash, signature, RNG
      – Not support key management
    • Key Storage Provider (KSP)
      – Support key management, run in isolation process
      – Support for third party developed KSP module
Steal Local Certificate

- Public key certificate storage
  - Public key is used to be spread -> no security consideration
  - Stored in system registry
    - HKEY_CURRENT_USER\Software\Microsoft\SystemCertificates
    - Stored by certificate store
    - Store certificate value in Blob key
Steal Local Certificate

• Public key certificate storage Cont.
  – Windows saves public key certificate by storing a group of properties

typedef struct
{
    DWORD dwType,
    DWORD dwEffective,
    DWORD dwDataLen,
    BYTE bData[1]
};

– The definition of dwType can be found in reference of CertGetCertificateContextProperty function
– Type 0x20 represents the X.509 certificate value

2009/11/05 For POC2009
Steal Local Certificate

• Private key and certificate storage
  – Key and certificate management are implemented by certain CSP module
  – We discuss Microsoft CSP (rsaenh.dll) here
  – Write three associated files
    • Private key file
      – %APPDATA%\Microsoft\Crypto\RSA\<user SID>\n      – We will discuss the file format later
    • Public key certificate file
      – %APPDATA%\Microsoft\SystemCertificates\My\Certificates\n      – The same format as Blob key mentioned above
      – Must include CERT_KEY_PROV_INFO_PROP_ID property
    • Key file
      – %APPDATA%\Microsoft\SystemCertificates\My\Keys\n      – Just store CERT_KEY_PROV_INFO_PROP_ID property
Steal Local Certificate

• Private key and certificate storage Cont.
  – CERT_KEY_PROV_INFO_PROP_ID help OS to find the private key associated with a certificate

```
typedef struct _CRYPT_KEY_PROV_INFO {
    LPWSTR pwszContainerName; // Container Name
    LPWSTR pwszProvName; // CSP Name
    DWORD dwProvType; // CSP Type
    DWORD dwFlags;
    DWORD cProvParam;
    PCRYPT_KEY_PROV_PARAM rgProvParam;
    DWORD dwKeySpec; // Key Type
} CRYPT_KEY_PROV_INFO, *PCRYPT_KEY_PROV_INFO;
```

2009/11/05

For POC2009
Steal Local Certificate

• Protections for private key
  – DPAPI encrypt sensitive data
    • CryptProtectData / CryptUnprotectData
    • RPC local call -> lsass.exe
    • The encryption and decryption must be done in the same user context

BOOL WINAPI CryptProtectData(
    DATA_BLOB* pDataIn,
    LPCWSTR szDataDescr,
    DATA_BLOB* pOptionalEntropy,
    PVOID pvReserved,
    CRYPTPROTECT_PROMPTSTRUCT* pPromptStruct,
    DWORD dwFlags,
    DATA_BLOB* pDataOut);
Steal Local Certificate

• Protections for private key Cont.
  – Export Flag
    • DWORD value (0/1), stored together with private key
    • Encrypted by DPAPI
  – Notice dialog displaying
    • DPAPI parameter CRYPTPROTECT_PROMPTSTRUCT
      – CRYPTPROTECT_PROMPT_ON_PROTECT
        » Display notice dialog when encrypting
      – CRYPTPROTECT_PROMPT_ON_UNPROTECT
        » Display notice dialog when decrypting
      – CRYPTPROTECT_STRONG
        » Force to use password protection
Steal Local Certificate

- Private key format
  - Container Name
  - Public Key BLOB / Private Key Blob
  - Export Flag

```
Header (40 Bytes)
Container Name
RSA1
Public Key BLOB

Private Key BLOB (Encrypted)

Export Flag (0/1)
CryptProtectData

Data Description: Export Flag
Optional Entropy: HjidiQ6kpUx7VC4m

Version
Protect Flag
Container Name size
Public Key BLOB size
Private Key BLOB size
Export Flag size
```

For POC2009
Steal Local Certificate

- CNG Security Improvement
  - Isolation process
    - CryptoAPI -> DPAPI -> lsass.exe
    - CNG -> lsass.exe
    - ncrypt.dll calls the image of itself in lsass.exe by keyiso.dll transfer
Steal Local Certificate

• CNG Security Improvement Cont.
  – Safety storage
    • Private key file location
      – %APPDATA%\Microsoft\Crypto\Keys
    • Use lsasrv.dll!LsaICryptProtectDataEx to encrypt
      – The same code as lsasrv.dll!s_SSCryptProtectData
    • Store a group of properties associated with a private key

typedef struct
{
    DWORD    dwLen,           // length of structure
    DWORD    dwType,          // property type
    DWORD    dwUnknown,
    DWORD    dwNameLen,       // byte count of property name
    DWORD    dwPropertyLen,   // byte count of property value
    WCAHR    szPropertyName[dwNameLen / 2],  // property name
    BYTE     bPropertyData[dwPropertyLen]      // property value
};
Steal Local Certificate

• CNG Security Improvement Cont.
  – Safety storage
    • General Properties: “Modified”, “Export Policy”
    • Private key file format
Steal Local Certificate

• Export PFX certificate
  – Open destination certificate store, such as “MY”
    • CertOpenStore
  – Search certificate, obtain CERT_CONTEXT
    • CertFindCertificateInStore
  – Open a temp certificate store, add CERTCONTEXT to the store
    • CertAddCertificateContextToStore
  – Export the certificate in the temp store, set the flag as EXPORT_PRIVATE_KEYS
    • PFXExportCertStoreEx
Steal Local Certificate

- Break Export Flag protection
  - Export Flag: DWORD (0 / 1)
  - CryptProtectData
    - Description: Export Flag
    - OptionalEntropy: Hj1diQ6kpUx7VC4m
    - 0xA8 length encrypted data, as follows
Steal Local Certificate

• Break Export Flag protection Cont.
  – hook CryptUnprotectData
    • Easy to hook in own process space
    • Determine if it is going to unprotect Export Flag through OptionalEntropy parameter and input length
    • Modify the 4 byte result to 0x00000001
  – Rewrite export flag
    • Find the file offset of storing Export Flag
    • Use CryptProtectData to encrypt DWORD = 1 and rewrite the data
Steal Local Certificate

• Break notice displaying protection
  – Protection flag is store at the offset 0x28
    CRYPTPROTECT_PROMPT_ON_PROTECT / CRYPTPROTECT_PROMPT_ON_UNPROTECT
  – CryptUnprotectData test the flag -> I_CryptUIProtect -> cryptui!I_CryptUIProtect
  – hook cryptui!I_CryptUIProtect
    xor eax, eax
    ret 18h
Steal Local Certificate

• Break password protection (CRYPTPROTECT_STRONG)
  – Password is set by user at the first time to protect data
  – I_CryptUIProtect compute SHA1 value of password -> s_SSCrypt ProtectData
    
    ```
    int s_SSCryptProtectData( 
        HANDLE BindingHandle, PBYTE pDataOut.pData, PDWORD &pDataOut.cbData, 
        PBYTE pDataIn.pData, DWORD pDataIn.cdData, LPWSTR* ppszDataDescr, 
        PBYTE pOptionalEntropy.pData, DWORD pOptionalEntropy.cdData, 
        PVOID pvReserved, PVOID pPromptFlags, DWORD dwFlags, 
        PBYTE pPasswordHash, // SHA1 hash value of password 
        DWORD pPasswordHashLen // length of hash value 
    );
    ```
  – Hash value is used to derive the symmetric key
  – Prerequisite: known the password
    - hook cryptui!I_CryptUIProtect
    - Compute SHA1 hash: UNICODE string exclude the NULL terminator
    - Copy hash value to the buffer address in sixth parameter
Steal Local Certificate

• CNG Problems
  – Few CA require users to use CNG
  – Key point of export certificate -> be able to export private key -> “Export Policy” property
  – Hook method -> inject into lsass.exe -> administrator priority required
  – Rewrite properties data -> normal user priority
Steal Local Certificate

• Real case
  – Alipay.com use certificate
  – Two certificates: user cert and machine cert
    • Microsoft Base Cryptographic Provider v1.0
      – Inside: machine cert can’t be exported
      – We can successfully export it using the methods above
      – Not safe, suggest for using USB key
Steal Local Certificate

• Security improvement suggestion
  – Password protection is relatively safer than other protection, so we suggest to add some UI for user setting and changing the password
  – In Vista or Win7, key pair import and export operations only need normal user priority. If these operations need administrator priority, UAC mechanism can help to protect users’ private key
Agenda

• Background
• Basic Knowledge
• Steal Local Certificate
• Steal Information Card
• Attack Smart Card
• Demo
• Conclusion
Steal Information Card

• CardSpace Components
  – infocardapi.dll
    • Export some useful API: ManageCardSpace, ImportInformationCard, GetBrowserToken
      – rundll32.exe infocardapi.dll,ManageCardSpace
    • This dll is used like a shell. It will establish RPC connection to call real function in infocard service
  – icardagt.exe
    • It is responsible for displaying CardSpace UI
    • The process is created by infocard service
    • User can create, backup, import and modify information card through UI
    • Also establish RPC connection to send command
Steal Information Card

• CardSpace Components Cont.
  – infocard.exe

  • Microsoft .Net Framework Windows Service. Service name is “idsvc”
  • Implements all the functions of managing information cards
  • Accept RPC connections and receive commands
  • Reflector can be used to disassemble it
  • %LOCALAPPDATA%\Microsoft\CardSpace\CardSpaceSP2.db
Steal Information Card

- Protection for CardSpace
  - Separate desktop
    - CardSpace UI is displayed in a separate new created desktop like UAC
    - CreateDesktop, SwitchDesktop, GdiplusStartup
  - Safety storage
    - CardSpaceSP2.db is encrypted
    - Microsoft.InfoCards.FileDataSource.LoadFrom
    - Only system account can access the file
Steal Information Card

- Protection for CardSpace Cont.
  - PIN code
    - User can set PIN code for certain information card
    - Before using this card, user must block it using PIN code
    - Actually private data in information card will be encrypted by key derived from PIN code
  - Protection of icardagt.exe
    - Microsoft signature check
    - Exception catch
Steal Information Card

- Function Call Protocol
  - Icardagt.exe use NdrClientCall2 to call functions in infocard service, different pFormat indicates different function call
  - RPCRequestDispatch is used to send various request command to infocard service
  - Three parameters
    - HANDLE, REQUEST *, RESPONSE **
  - Some request command
    - GetIsBrowserClientRequest, SendAgentStatusRequest, InfoCardListRequest, ExportFileRequest
Steal Information Card

• Function Call Protocol Cont.
  – Structure definition
    ```
    typedef struct _REQUEST
    {
        PWCHAR pwszCommandName;       // name of command to send
        DWORD dwSize;                 // buffer size of command parameter
        PVOID pParam;                // command parameter
    } REQUEST;
    ```
    ```
    typedef struct _RESPONSE
    {
        DWORD dwSize;                // buffer size
        PVOID pBuf;                 // buffer to store command result
    } RESPONSE;
    ```
  – InfoCardListRequest
    • Enumerate all information cards stored in CardSpace
    • REQUEST.dwSize = 0, REQUEST.pParam = NULL
    • RESPONSE.pBuf holds data of information cards in XML format
    • Uuid property is used to represent a card
Steal Information Card

• Function Call Protocol Cont.
  – ExportFileRequest
    • Make infocard service to backup information cards
    • REQUEST.pParam points structure:

```c
typedef struct _EXPORT_PARAM
{
    DWORD dwPathCount;         // character count of backup file path
    WCHAR wszPath[dwPathCount]; // backup file path
    DWORD dwPassCount;         // character count of password
    WCHAR wszPass[dwPassCount]; // password used to protect backup file
    DWORD dwCardCount;         // count of cards to be exported
    DWORD dwFirstCardUuidCount; // character count of card uuid
    WCHAR wszFirstCardUuid[dwFirstCardUuidCount];    // card uuid
    ...
} EXPORT_PARAM;
```
Steal Information Card

• Backup Information Card
  – It is quite complicated to communication with infocard service by ourselves
  – Assume that our code is already injected into icardagt.exe, we can reuse values needed to execute NdrClientCall2
  – Two steps to backup information card
    • Send InfoCardListRequest command
      – Gain card uuid by searching “urn:uuid:” in RESPONSE.pBuf
    • Send ExportFileRequest command
Steal Information Card

• Looking into card backup file
  – The backup file is an XML file. Information card data is stored in CipherData property.
  – Encrypt process can be found:
    Microsoft.InfoCards.RoamingStoreFileUtility.Encrypt
  – Example

```xml
<?xml version="1.0" encoding="utf-8" ?>
<EncryptedStore xmlns="http://schemas.xmlsoap.org/ws/2005/05/identity">
  <StoreSalt>+t6us2rvZ03tcsGWLc0cL6A==</StoreSalt>
  <EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#">
    <CipherData>
      <CipherValue>wEhVTjBXtkINP0xKeMgOQTwN/3fB88A0PhM/j+uTVslQZm</CipherValue>
    </CipherData>
  </EncryptedData>
</EncryptedStore>
```
Steal Information Card

• Information card example
Steal Information Card

• Break CardSpace Protection
  – Inject steps
    • Inject code to infocard.exe and hook CreateProcess
    • Copy our dll to <directory>\<system dll>
      – System dll is a dll name will be used by icardagt.exe, eg: uxtheme.dll
    • Change lpApplicationName from “C:\windows\system32\icardagt.exe” to “<directory>\icardagt.exe”
  – Make process background
    • Hook SwitchDesktop
    • User cannot see CardSpace UI
  – Record values
    • Hook NdrClientCall2
Steal Information Card

• Break CardSpace Protection Cont.
  – PIN code protection
    • Private data of card (e.g., master key) will be encrypted by key derived by PIN code
  • Hook NdrClientCall2
  • Search REQUEST.pParam of UnlockCardRequest
    – After “urn:oasis:names” property and near the end of buffer
Agenda

• Background
• Basic Knowledge
• Steal Local Certificate
• Steal Information Card

• **Attack Smart Card**
• Demo
• Conclusion
• Smart card CSP
  – Cryptography operation supported by hardware, completely isolated security storage environment
  – Compared with Microsoft CSP
    • Cryptography operations become faster, and key need to be loaded into OS memory
    • Support to generate asymmetric key on card, so the private key will never leave card
    • PIN verification is needed before using private key
    • Can be used in different computers
Attack Smart Card

• Components of smart card CSP
  – CSP module
    • Be registered to OS, exports CryptoSPI
    • Implement: container, key, hash, symmetric, signature, verification, RNG, derive key, generate key
    • CPXXXX functions
  – Intermediate interface
    • RSA PKCS#11, Cryptoki
      – Compared with CSP: more flexible, cross-platform
    • Implement CSP based on PKCS#11
    • Some open source software use PKCS#11: FireFox
    • C_XXXX functions
Components of smart card CSP Cont.

- Driver module

  - Provide functions for open, close, transmit
  - Microsoft PC/SC standard, SCardXXXX functions

    - `SCardEstablishContext` // establish a context to use
    - `SCardListReaders` // enumerate usable reader
    - `SCardConnect` // connect to card in reader
    - `SCardStatus` // get information of card: reader name, transmit protocol, card status, ATR
    - `SCardBeginTransaction` // tell driver to begin a transaction
    - `SCardTransmit` // send and receive APDU
    - `SCardEndTransaction` // close a transaction
    - `SCardDisconnect` // close connection of card
    - `SCardReleaseContext` // release the context
Components of smart card CSP Cont.

- Smart card COS (Chipset Operating System)
  - Upper level API control card by sending APDU
  - COS receive command and execute, then return the response
  - COS manage key and certificate file by small file system
  - ISO7816 standard describes APDU format

<table>
<thead>
<tr>
<th>Command APDU</th>
<th>Response APDU</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLA</td>
<td>INS</td>
</tr>
<tr>
<td>Le</td>
<td>Response Data</td>
</tr>
</tbody>
</table>

CLA: command transmit type
INS: instruction  P1/P2: parameters
Lc: length of Data Filed  Le: length of Response Data
SW1/SW2: status word 1/2 , 90 00 indicates the command execute successfully
Attack Smart Card

- Components of smart card CSP Cont.
  - Architecture

For POC2009
Attack Smart Card

• Attack Method
  – Private key is stored in card, we cannot read it through software methods
  – In a specific verification process, server relies on matching signature data with signature result, therefore it can still pass the verification if the signature data is matched with signature result
  – Three key points
    • Simulating a smart card which makes the upper level believe the existence of smart card
    • Obtain public key certificate
    • Obtain signature data and matching signature result
Attack Smart Card

• Simulating smart card
  – Applications may use different interfaces, for example, smart card management tool may use driver directly to obtain key serial, so we need to simulate in several levels
  – CSP Proxy
    • Rewrite Image Path & Signature key in registry
    • Export CryptoSPI to cheat upper level applications
    • Call original CSP module and record useful data
      CPHashData
      CPSSetHashParam
      CPGetUserKey
      CPSSignHash
Attack Smart Card

• Simulating smart card Cont.
  – PKCS#11 Proxy
    • No signature protection, just replace module file
    • Similar usage with CSP Proxy
      C_Login
      C_DigestInit / C_DigestUpdate / C_Digest / C_DigestFinal
      C_SignInit / C_SignUpdate / C_Sign / C_SignFinal
  – Driver cheating
    • Implement some PC/SC interface
    • Record some useful APDU which is needed for cheating
      SCardListReaders
      SCardStatus
      SCardTransmit
Attack Smart Card

• Obtain public key certificate
  – Applications (IE) will find certificates in store and locate the associated private key
  – Just copy related files from remote to local

• Obtain signature result
  – Compute signature result on remote computer
  – Is signature data predictable?
  – Use CSP to compute signature
    • CryptAcquireContext -> CryptCreateHash -> CryptHashData -> CryptSignHash

2009/11/05 For POC2009
Attack Smart Card

- Architecture
Attack Smart Card

• Real case
  – Online bank (USB key)
    ……
  – Security improvement suggestion
    • Three key points: transaction information, PIN code, signature result
    • Assume the computer is not trusted
      – Transaction information may be modified, PIN code may be stolen
    • Suggestion
      – Smart card shows the transaction information and requires user’s confirm
      – Directly input PIN code on smart card
Agenda

• Background
• Basic Knowledge
• Steal Local Certificate
• Steal Information Card
• Attack Smart Card

• **Demo**

• Conclusion
Demo

- Local certificate
- Information card
 Agenda

• Background
• Basic Knowledge
• Steal Local Certificate
• Steal Information Card
• Attack Smart Card
• Demo

• Conclusion
Conclusion

- People are aware of the serious security problems of password authentication system and begin to use certificate-base authentication system instead.
- For sake of cost, local certificate is more popular although it is not so safe as we image.
- Microsoft CardSpace is very convenient but not as safe as we think.
- Smart card is much safer.
- No absolutely safety, smart card developed by third party may have security problems.
Thank you for your time

Q & A