Author: rupp,scd
Project: Michigan Game Machine
Date: 01/06/13

REVERSE ENGINEERING

Index

Screens

1. Introduction

This document describes the process achieved for retrieving the terminal key, card key, PIN codes for IC and ACS1.

2. Interception DLLs

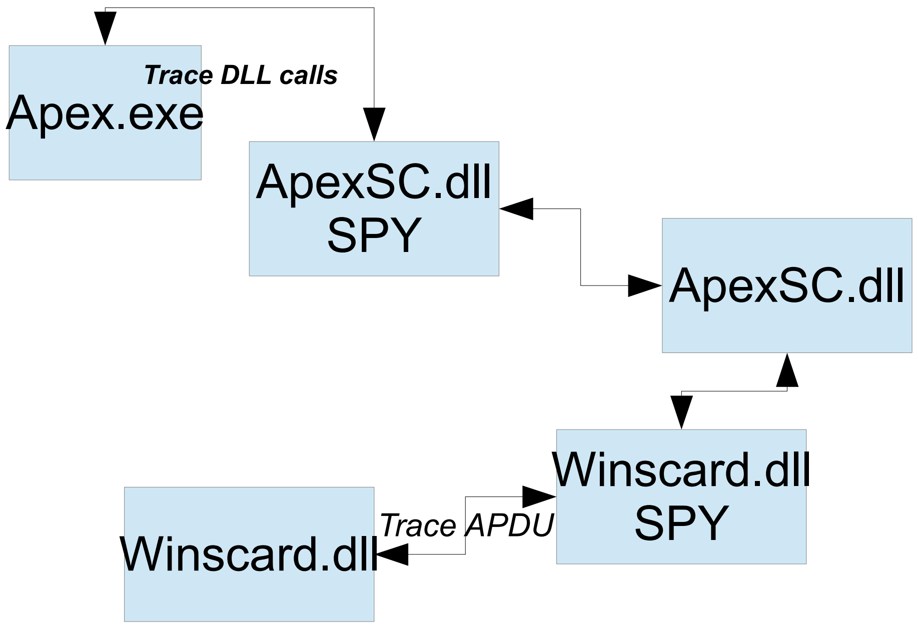
We first use a Linux boot drive to access the hard drive of the game machine but due to NTFS problems, the hard drive could not be mounted despite loading the 3G NTFS driver in our Linux boot system.

The content of the hard drive of the terminal is finally copied by using a USB to SATA adapter.

We discover that most of the code is to be found inside the “APEX” folder. We discover that a main binary named “APEX CE DLL” , APEX.EXE is the executable system and use several DLL for accessing serial, database, and smartcards.

We find that ApexSC.dll is the DLL component responsible for dealing with the SIM cards.

We proceed with a DLL proxy/spy/interceptor system . We duplicate ApexSC.dll using a stub dll generation system and we trace the calls between APEX.EXE and ApexSC.dll



We discover that APEX.exe has a built-in file protection system. The resources used by APEX.exe are compared with a database of file signatures and if the signature does not match the file is rejected. If the signature match the file is loaded in a RAM drive.

We cannot then load our SPY DLL using the existing APEX binary so we must disable the file protection system.

3. OllyDbg

Olly debug is a free GNU assembly-level debugger.

By using that tool, we can edit the assembly code. We replace the code that throw exception when file signature do not match by inoperant code so we can bypass the file protectioon.

We then discover, by comparing our loging DLL call trace log with the APDU trace log that full card authentication (Acos3 Start session, mutual authentication and verify) is done by call of the following ApexSC.dll functions:

1) *initialize()*
2) *connect()*

3) *verify()*

4. IDA+Hex-Ray

We use IDA and the dissasembler Hex-Ray to reverse the ApexSC.dll to C code.
It appears that ApexSC.dll is not obfuscated and that Hex-Ray can reverse 99% of the DLL to C code. The code is tokenized so that deep reverse analysis must be done in order to reconstitute human-readable code and understanding the functionnement.

Most of the code can be reconstituted manually from the hex-ray C code after a deep analysis. The following code that manage the mutual authentication can be found:

```
int __cdecl _card_authentication_workflow_()
{
 int v0; // eax@1
 int _session_key_poss_ptr_; // esi@1
 int result_auth; // eax@9
 int v3; // eax@11
 char v5; // zf@1
 int  s; // [sp+58h] [bp+0h]@1
 unsigned int v7; // [sp+44h] [bp-14h]@1
 int v8; // [sp+4Ch] [bp-Ch]@1
 int v9; // [sp-4h] [bp-5Ch]@1
 int *v10; // [sp+48h] [bp-10h]@1
 int v11; // [sp+54h] [bp-4h]@1
 void *v12; // [sp+24h] [bp-34h]@1
 signed int v13; // [sp+20h] [bp-38h]@2
 signed int v14; // [sp+10h] [bp-48h]@4
 char Dst; // [sp+28h] [bp-30h]@5
 signed int v16; // [sp+14h] [bp-44h]@6
 signed int v17; // [sp+18h] [bp-40h]@8
 char v18; // [sp+3Bh] [bp-1Dh]@9
 char v19; // [sp+3Eh] [bp-1Ah]@9
 char _3DES_KEY1_1; // [sp+34h] [bp-24h]@9
 char _3DES_KEY1_2; // [sp+35h] [bp-23h]@9
 char _3DES_KEY1_3; // [sp+36h] [bp-22h]@9
 char _3DES_KEY1_4; // [sp+37h] [bp-21h]@9
 char _3DES_KEY1_5; // [sp+38h] [bp-20h]@9
 char _3DES_KEY1_6; // [sp+39h] [bp-1Fh]@9
 char _3DES_KEY1_7; // [sp+3Ah] [bp-1Eh]@9
 char _3DES_KEY2_1; // [sp+3Ch] [bp-1Ch]@9
 char _3DES_KEY2_2; // [sp+3Dh] [bp-1Bh]@9
 char _3DES_KEY2_3; // [sp+3Fh] [bp-19h]@9
 char _3DES_KEY2_4; // [sp+40h] [bp-18h]@9
 char _3DES_KEY2_5; // [sp+41h] [bp-17h]@9
 char _3DES_KEY2_6; // [sp+42h] [bp-16h]@9
 char _3DES_KEY2_7; // [sp+43h] [bp-15h]@9
 signed int v34; // [sp+30h] [bp-28h]@10
 signed int v35; // [sp+1Ch] [bp-3Ch]@12
 signed int v36; // [sp+2Ch] [bp-2Ch]@14

 v7 = (unsigned int)& s ^ dword_10007018;
 v0 = (int)&v8;
 v10 = &v9;
 _session_key_poss_ptr_ = (int)dword_10007708;
 v11 = 0;
 v5 = *((_DWORD *)dword_10007708 + 135) == 0;
 v12 = dword_10007708;
 if ( v5 )
 {
  v13 = -5; //0xFB
  v0 = CxxThrowException(&v13, dword_10005A28);
 }
 //read the main file record
 if ( !_ACOS_card_read_record_NS(v0, _session_key_poss_ptr_, -254 /*0x01*/) )
 {
   v14 = -6;
   CxxThrowException(&v14, dword_10005A28);
 }
 //not sure what it is

 if ( !sub_10002990(4u, &Dst, 1) )
 {
   v16 = -7;
   CxxThrowException(&v16, dword_10005A28);
 }

 //this is the content of File id=0xFF02
 //must contain "WTGL"
 if ( strncmp(&Dst, "WTGL", 4u) )
 {
   v17 = -8;
   CxxThrowException(&v17, dword_10005A28);
 }

 //could be the DES KEY?? Which one? Terminal Key
 //16 bytes ?

 v18 = 69;              //0x45
 v19 = 69;              //0x45

 // card terminal and card key ???


 _3DES_KEY1_1 = 13;         //0x0D
 _3DES_KEY1_2 = 79;         //0x4F
 _3DES_KEY1_3 = 17;         //0x11
 _3DES_KEY1_4 = -51;        //0xCD
 _3DES_KEY1_5 = 83;         //0x53
 _3DES_KEY1_6 = -63;        //0xC1
 _3DES_KEY1_7 = -69;        //0xBB

 _3DES_KEY2_1 = -29;        //0xE3
 _3DES_KEY2_2 = -44;        //0xD4
 _3DES_KEY2_3 = -80;        //0xB0
 _3DES_KEY2_4 = 41;         //0x29
 _3DES_KEY2_5 = 70;         //0x46
 _3DES_KEY2_6 = 51;         //0x33
 _3DES_KEY2_7 = 108;        //0x6C

 //if the keys are passed there,  in which parameter?
 //third parameter ??
 //but 3rd parameters has 7 bytes not 8
 //or should we put v19 at rank #3 but why??
 result_auth = _mutual_auth_get_response_((int)&_3DES_KEY1_1, _session_key_poss_ptr_, (int)&_3DES_KEY2_1,
(int)&_3DES_KEY1_1);
 if ( !result_auth )
 {
   v34 = -2;
   result_auth = CxxThrowException(&v34, dword_10005A28);
 }

 //now the SESSION KEY K_S MUST HAVE BEEN COMPUTED
 //must be in _session_key_poss_ptr_
 //new keys??


// secret CODE  1 ?
//will be send ciphered by K_S
//they are using that buffer to store statically the des keys and the secrets??

 _3DES_KEY2_1 = 72; //0x48
 _3DES_KEY2_2 = 63; //0x3F
   v19 = 88;
 _3DES_KEY2_3 = 14; //0x0E
 _3DES_KEY2_4 = -21;//0xEB
 _3DES_KEY2_5 = 15; //0x0F
 _3DES_KEY2_6 = -92;//0xA4
 _3DES_KEY2_7 = -28;//0xE4
 //secret type=7
 v3 = _card_send_code_(v2, _session_key_poss_ptr_, (int)&_3DES_KEY2_1, 7);
 if ( !v3 )
 {
   v35 = -3;
   v3 = CxxThrowException(&v35, dword_10005A28);
 }


 //secret code 2

 _3DES_KEY2_1 = -18;
 _3DES_KEY2_2 = 62;
   v19 = 80;
 _3DES_KEY2_3 = 94;
 _3DES_KEY2_4 = 53;
 _3DES_KEY2_5 = -81;
 _3DES_KEY2_6 = -6;
 _3DES_KEY2_7 = -42;
 //secret type=1
 //CODE REFERENCE ACS1
 if ( !_card_send_code_(v3, _session_key_poss_ptr_, (int)&_3DES_KEY2_1, 1) )
 {
   v36 = -4;
   CxxThrowException(&v36, dword_10005A28);
 }
 return 0;
}
```

5. Final analysis

The analysis finally shows that the keys are made with 7 bytes constants and with a special byte data that is inserted to create the 64 bits DES keys.

We finally find the following keys and PIN code that we verify:

terminal key:	*E3 D4 45 B0 29 46 33 6C (VERIFIED)*
card key:	*0D 4F 11 CD 53 C1 BB 45 (VERIFIED)*
secret code / PIN for IC:	*48 3F 58 0E EB 0F A4 E4 (VERIFIED)*
secret code / PIN for ACS1:	*EE 3E 50 5E 35 AF FA D6 (VERIFIED)*