```
1: {*************************************************************
2:  *
3:  * Project  : SCAPIT - Secure Center API Tester
4:  * Unit Name: 1314_sysinformation_shell, loc's = 131
5:  * Purpose  : Demonstrates SecureCenter imported functions from Windows API
6:  * History  : milo dll, teplas dll, v2.5
7:  *************************************************************}
8:
9: program SecureCenterDLL_Tester;
10:
11: // Example Ressources from SC CD ..\SC_Official_Keylist.zip
12:
13: const PASSPHRASE = '99 Luftballone in Holland$';
14:       KEYFILE = 'Alchenberger_Lorenz_2007.scsekey';   //9F08D2EC....hash
15:       DOCFILE = 'maxbox_starter5.pdf';
16:       CIPHERFILE = 'maxbox_starter5.pdf.scse';
17:       SUBDIR = 'ciphertest\';
18:
19:
20: //DLL Test Generic ------------------------------------------------
21: function  MessageBeep(para: integer): byte;
22:      external 'MessageBeep@user32.dll stdcall';
23:
24: function MyGetTickCount: Longint;
25:   external 'GetTickCount@kernel32.dll stdcall';
26:
27: procedure MySleep(Milliseconds: Cardinal);
28:   external 'Sleep@kernel32.dll stdcall';
29: //-------------------------------------------------------------
30:
31: {function CreateKey(mtype: word; username, keyname, password, validfrom,
32:                   validuntil: PChar; var key: pointer; var keylength: word;
33:                    var keyhash: Pchar): word;
34:   external 'SCCreateNewKey@SecureCiphering.dll stdcall';}
35:
36: function SCCipherTheFile2(handle: word; keyname, password: PChar; actionattrib: word;
          workfilename: pChar; var postfilename: string; MB: boolean): word;
37:   external 'SCCipherTheFile@SecureCiphering.dll stdcall';
38:
39: procedure SCCipherProgress(var PercentsDone: WORD; cancel: boolean);
40:   external 'SCCipherProgress@SecureCiphering.dll stdcall';
41:
42:
43: function SCLoadAndSetKeyStateWithFileName2(Handle: word; SCKeyName, Password: PChar;
44:                         ActId: WORD; KeyHash: pchar): WORD;
45:   external 'SCLoadAndSetKeyStateWithFileName@SecureCiphering.dll stdcall';
46:
47: var
48:   mytimestamp: TDateTime;
49:   E: Extended;
50:   cfile, keyhash: string;
51:   errorCode, progress: word;
52:
53:
54: Procedure DateTimeExample;
55: Var S, S1 : String;
56: begin
57:   {Time, Date, and Now are all basically interchangable.}
58:   {They all return a DateTime value.}
59:   S1:=DateToStr(Now2);
60:   Case DayOfWeek(Date) of
61:      1: S:='Sunday';
62:      2: S:='Monday';
63:      3: S:='Tuesday';
64:      4: S:='Wednesday';
65:      5: S:='Thursday';
66:      6: S:='Friday';
67:      7: S:='Saturday';
68:   Else S:='Noneday';
69:   End; {Case}
70:   writeln(S+' '+S1+' '+TimeToStr(Time));
71: end;
72:
73:
74: procedure SC_Cipher;
```

```
 75:  //example with key and passphrase in parameter, para 1 is cipher
 76: begin
 77:   errorCode:= SCCipherTheFile2(0, ExePath+SUBDIR+KEYFILE, Passphrase,
 78:               1, ExePath+ SUBDIR+DOCFILE, cfile, false)
 79:   if errorCode = 0 then
 80:        SearchAndOpenDoc(ExePath+SUBDIR+CIPHERFILE)
 81:   Writeln(intToStr(errorCode))
 82: end;
 83:
 84:
 85: procedure SC_Decipher;
 86:   //example with key and passphrase in parameter, para 2 is decipher
 87: begin
 88:   errorCode:= SCCipherTheFile2(0, ExePath+SUBDIR+KEYFILE, Passphrase,
 89:               2, ExePath+ SUBDIR+CIPHERFILE, cfile, false)
 90:   if errorcode = 0 then
 91:        SearchAndOpenDoc(ExePath+SUBDIR+DOCFILE)
 92:   Writeln(intToStr(errorCode))
 93: end;
 94:
 95: procedure SC_LoadandSetKeyState;
 96: begin
 97:   errorCode:= SCLoadAndSetKeyStateWithFileName2(0, ExePath+SUBDIR+KEYFILE,
 98:                  PASSPHRASE, 2, keyhash)
 99:   if errorCode = 0 then
100:     Writeln(' this is hash of ' +keyhash)
101:     Writeln(inttostr(errorCode))
102: end;
103:
104:
105: begin
106:   writeln(Format('Pi starts off as %.15f', [PI]));
107:   dateTimeExample;
108:   writeln('machine name is: '+getHostName)
109:   writeln('user name is: '+getUserName)
110:   mytimestamp:= GetFileCreationTime(exepath+'maxbox3.exe')
111:   writeln(datetimetoStr(mytimestamp)+' Creation Date of maXbox3')
112:   writeln('tick count from func ' +IntToStr(GetTickCount));
113:   writeln('tick count from api   ' +IntToStr(MyGetTickCount));
114:   MySleep(10);
115: //maxburaut TEST  ("CE7899D61F05331B20E321814D1CF0726268)
116:
117:     errorCode:= 0;
118:     cfile:= '';  //just to init the pChar no return available
119:     keyhash:= '';
120:     progress:= 0;
121:
122:     SC_Cipher;
123:     //SCCipherProgress(progress, true)
124:
125:     SC_Decipher;
126:     //SC_LoadAndSetKeyState;
127:
128:    {if errorCode = 0 then
129:     writeln('well done cipher: '+inttoStr(errorcode)) else
130:     writeln('error from sc api: '+inttoStr(errorcode))}
131: End.
132:
133: ********************************************************************************
134: There is no place like 127.0.0.1
135:
136: function SCCipherTheFile(Handle: HWND; SCKeyName: PChar;
137: Password: PChar;
138: ActionAttrib: WORD;
139: WorkFileName: PChar;
140: var PostFileName: PChar
141: ): WORD; export; stdcall;
142:
143: function SCLoadAndSetKeyStateWithFileName(Handle: HWND; SCKeyName: PChar;
144: Password: PChar;
145: ActId: WORD;
146: var KeyHash: PChar
147: ): WORD; stdcall;
148:
149: The structure of a script is as follows (keywords are shown in ALLCAPS bold):
```

```
150:
151: PROGRAM
152:
153: CONST
154:
155: <Constant declarations>
156: TYPE
157:
158: <Type declarations>
159: VAR
160:
161: <Variable declarations>
162: BEGIN
163:
164: <Executable statements>
165: END.
166:
167: Note that:
168:
169: The main code must be within the begin and end. keywords.
170: All statements in the script use the semicolon ";" as terminator. Only the last statement
     (END.) uses a dot as terminator.
171:
172: function BinToOct(BinStr: string): string;
173: var
174:   i: Integer;
175:   OctPart: single;
176:   LastPart, OctStr: string;
177:   Error: Boolean;
178: begin
179:   Error:=False;
180:   OctStr:='';
181:   case Length(BinStr) mod 3 of
182:     1: BinStr:='00'+BinStr;
183:     2: BinStr:='0'+BinStr;
184:   end;
185:
186:   while Length(BinStr)>0 do begin
187:     LastPart:=Copy(BinStr, Length(BinStr)-2, 3);
188:     Delete(BinStr, Length(BinStr)-2, 3);
189:     OctPart:=0;
190:     for i:=1 to 3 do
191:       if not ((LastPart[i] ='1') or (LastPart[i] ='0')) then begin
192:         ShowMessage('This is not binary number');
193:         Error:=True;
194:         Break;
195:       end
196:       else
197:         OctPart:=OctPart+StrToInt(LastPart[i])*Power(2, 3-i);
198:     OctStr:=OctStr+FloatToStr(OctPart);
199:   end;
200:   Result:='';
201:   if Error<>True then begin
202:     for i:=1 to Length(OctStr) do
203:       Result:=Result+OctStr[Length(OctStr)-i+1];
204:     while (Result[1]='0') and (Length(Result)>1) do
205:       Delete(Result, 1, 1);
206:   end;
207: end;
```