Sentinel SuperPro v praxi

Úvod

V dnešnej dobe je termín hardwarový kľúč pomerne známy. O jeho popularizáciu sa pričinili predovšetkým tisícky softwarových pirátov, ktorí prispeli k nelegálnemu šíreniu komerčného softwaru. Na rozdiel od analógových záznamových médií je z diskety (a teda aj softwaru) možné vytvoriť prakticky neobmedzené množstvo kópií. Navyše pravdepodobnosť usvedčenia z porušenia autorských práv je minimálna, tak prečo nekopírovať?

Pochopiteľne, softwarové firmy mali na celý problém iný názor a vyvinuli niekoľko techník, ktoré mali zabrániť nelegálnemu šíreniu programov. Spočiatku sa používali špeciálne „nekopírovateľné“ diskety. Toto riešenie bolo síce lacné, avšak vzhľadom na to, že používalo náchylné médium, nie príliš spoľahlivé. Crackerom netrvalo dlho, a z „nekopírovateľných“ diskiet urobili diskety „kopírovateľné“. Nebolo tiež výnimkou, že originálny program upravili tak, aby ochrannú disketu nevyžadoval.

Potom prišli na scénu hardwarové kľúče. Na rozdiel od diskety sa nedali prekopírovať a boli podstatne spoľahlivejšie. Hardwarové kľúče preto „disketovú“ metódu úplne nahradili. Treba však podotknúť, že zavedením hardwarových kľúčov počet pirátskych programov príliš neklesol. Crackeri sa ani tentokrát nedali zahanbiť, a tak sa na Internete čoraz častejšie stretávame s upravenými („cracknutými“) komerčnými programami, ktorým neprítomnosť hardwarového kľúča vôbec neprekáža...

Základným princípom práce s hardwarovým kľúčom je vzájomná interakcia medzi kľúčom a aplikáciou. Vnútorný obsah HW kľúča sa člení na bunky (cells). Aplikácia zadá číslo bunky, z ktorej sa potom prečíta hodnota. Aplikácia potom zisťuje, či je táto hodnota správna. Bunky v HW kľúči môžu obsahovať napríklad heslá, sériové čísla a pod. Takýto kľúč sa nazýva pamäťový. V súčasnosti sa používajú kombinácie pamäťového a tzv. algoritmického kľúča. Pamäťový kľúč nám pri požiadavke na prečítanie obsahu určitej bunky vráti vždy tú istú hodnotu. Algoritmickému kľúču musíme dodať aj vstupný údaj, na základe ktorého nám potom poskytne výstupnú údaj. Ak napríklad dáme kľúču vstupnú hodnotu 8FA31B4B, kľúč nám vráti hodntou 1BC235B1. Môžeme si vytvoriť niekoľko desiatok takýchto párov čísel, pomocou ktorých budeme vykonávať kontrolu prítomnosti HW kľúča.

Hneď na úvod musím upozorniť, že tento článok si nekladie za cieľ suplovať príručku k Sentinelu SuperPro. Nebudeme sa zaoberať Advanced Editorom, ani pravidlami, ktoré je potrebné dodržiavať pri konfigurácii jednotlivých typov buniek. Cieľom článku je priblížiť záujemcom použitie tohto kľúča formou príkladov napísaných v Delphi 3. Je síce pravdou, že inštalačné CD obsahuje demoprogram aj pre Delphi, ten je však podľa môjho názoru relatívne komplikovaný.

Práca so Sentinelom

V prvej fáze je potrebné HW kľúč najskôr inicializovať. Program, ktorý si napíšeme bude v nemalej miere podobný programu Sentinel SuperPro Evaluation Program. Vytvoríme teda v Delphi nový projekt, do ktorého ihneď zaradíme jednotku (unit) SPROMEPS.PAS.

Ukážkový program nám inicializuje HW kľúč a vypíše základné údaje o verzii ovládača a prostredia, pre ktoré je ovládač určený.

```
procedure TForm1.InitBtnClick(Sender: TObject);
var error:integer;
majVer,minVer,revision,OSDriver: byte;
ffUnit:integer;
drv:string;
begin
  error := SproInitialize(@thePacket);
  ffUnit:=SproFindFirstUnit(@thePacket,DevId);
  SproGetVersion(@thePacket,majVer,minVer,revision,OSDriver);
  drv:=GetDriverString(OSDriver);
  Application.MessageBox(PChar('SproInitialize Status: ' +
StatusDesc(error)+
  chr(13)+ 'SproFindFirstUnit: '+StatusDesc(ffUnit)+
  chr(13)+ 'Major version: '+Inttostr(majVer)+
  chr(13)+ 'Minor version: '+Inttostr(minVer)+
  chr(13)+ 'Revision: '+Inttostr(revision)+
  chr(13)+ 'Driver type: '+drv),
  'SuperPro Initialization',MB_OK OR MB_ICONINFORMATION);
end;
```

Milovníci češtiny a slovenčiny mi dúfam odpustia anglické výrazy, ktoré budem vo svojom ukážkovom programe používať.
Premenná `thePacket` je typu `RB_SPRO_APIPACKET`. Používa ju väčšina funkcii pracujúcich s HW kľúčom. Aby bola „viditeľná“ aj pre ostatné jednotky, mala by byť deklarovaná ešte pred sekciou `implementation`. Súčasťou inicializácie je aj funkcia `SproFindFirstUnit`, do ktorej si musíte dosadiť svoje Developer ID. O zistenie informácií týkajúcich sa verzie sa postará funkcia `SproGetVersion`. Keďže pri práci s funkciami používame iba nič nehovoriace čísla, použil som procedúru `StatusDesc`, ktorá nám „ničnehovoriacu“ návratovú hodnotu nahradí veľavravným hlásením, ktoré som kvôli jednoduchosti ponechal v pôvodnej podobe, teda v angličtine. V angličtine sa nám budú zobrazovať aj reťazce popisujúce prostredie (operačný systém), pod ktorým ovládač HW kľúča pracuje.
Jednou s najzákladnejších operácií je čítanie. Pomocou Advanced Editoru kľúč naprogramujeme tak, aby bunka číslo 08h obsahovala hodnotu 00FFh. Realizácia samotného čítania je jednoduchá:

```
procedure TForm1.ReadBtnClick(Sender: TObject);
var status, data:word;
begin
  status := SproRead(@thePacket, $08, data);
  Application.MessageBox(PChar(StatusDesc(status)+
  chr(13) + 'Value: ' + IntToHex(data,2)), 'SuperPro Read',
  MB_OK OR MB_ICONINFORMATION);
end;
```

Samozrejme, najskôr je potrebné stlačiť tlačidlo, ktoré vykonáva inicializáciu. Aby sme neostali iba pri čítaní, ukážeme si teraz, ako prebieha zápis. Do tej istej bunky (t.j. 08h) zapíšeme hodnotu FAh:

```
procedure TForm1.WriteBtnClick(Sender: TObject);
var status:word;
tmp:string;
msg:array[0..80] of char;
begin
  status := SproWrite(@thePacket, WritePW, $08, $FA,0);
  tmp:=StatusDesc(status);
  StrPCopy(msg,tmp);
Application.MessageBox(Msg, 'SuperPro Write', MB_OK OR MB_ICONINFORMATION);
end;
```

V tomto prípade sme nepoužili klasické pretypvanie, pretože to nastavenie prekladača nedovoľuje. Nemusím pripomínať, že premenná `WritePW` má obsahovať vaše Write Password . Keďže je bunka 08h typu data word, nemusíme uvádzať prístupový kód, a tak má posledný parameter funkcie `SproWrite` hodnotu nula.
Keďže Sentinel SuperPro je aj kľúčom algoritmickým, ukážeme si teraz, ako pracovať s algoritmami. Ešte predtým však musím zdôrazniť, že program si budete musieť upraviť závislosti na hodnotách, ktoré vám HW kľúč vráti. V bunke číslo 20h vytvoríme aktívny algoritmus so statickým typom aktivácie a s aktivovaným Enhanced Algorithm Engine. Potom pravým tlačítko klikneme na túto bunku a z popup menu vyberieme položku Evaluate API. Objaví sa ďalšie menu, z ktorého vyberieme Query. Objaví sa dialógové okno Query API Parameters. Do položky Query String napíšeme AAFF a stlačíme OK. Následne sa otvorí dialógové okno zobrazujúce výsledky operácie. Pre nás je najdôležitejšia položka `Response`, ktorá má v mojom prípade hodnotu F9BF. Vrátime sa späť do Delphi a napíšeme si krátky kód:

```
procedure TForm1.QryBtnClick(Sender: TObject);
var
  realQueryStr,ResponseStr,disResponseStr : string[80];
  status : WORD;
  response32 : longint;
  tmp:array[0..50] of char;
begin
  ConvertQueryStr('AAFF', realQueryStr);
  status := SproQuery (@thePacket, $20, @realQueryStr[1],
                           @responseStr[1], response32,
                           Length(realQueryStr));
  SetLength(ResponseStr,Length(realQueryStr));
```

```
  ConvertResponseStr (responseStr, disResponseStr);
  StrPcopy(tmp,StatusDesc(status) + chr(13)+ 'Query Results:
'+disResponseStr);
  Application.MessageBox(tmp,'SuperPro Query', MB_OK OR
MB_ICONINFORMATION);
end;
```

Keďže funkcia `SproQuery` nepracuje s hodnotami typu `string`, musíme reťazec najskôr skonvertovať do natívneho dátového typu používaného touto funkciou. Na to nám slúži funkcia `ConvertQueryStr`. Samozrejme, po prijatí odpovede z HW kľúča je potrebná spätná konverzia na typ `string`, ktorú zabezpečí funkcia `ConvertResponseStr`. Ani v tomto prípade nie je možné použiť klasické pretypovanie, keďže to nastavenie prekladača nedovoľuje.
Základný prehľad o práci s kľúčom teda máme. Teraz si ukážeme, ako v praxi využiť údaje, ktoré sme s HW kľúča získali. V prvom rade je potrebné povedať, že tieto údaje by sme v programe nemali uvádzať priamo. Inými slovami, mali by sme sa vyvarovať konštrukciám typu

```
if x<>$ F9BF then
```

Hacker totiž môže relatívne jednoducho vyhľadať potrebnú assebmlerskú inštrukciu skoku a modifikovať ju tak, aby program pracoval aj vtedy, keď nedostane správnu hodnotu. Oveľa výhodnejším riešením je použiť túto hodnotu pri výpočtoch, ktorých výsledok ovplyvní beh programu. Výsledok výpočtov totiž nie je na prvý pohľad jasný, môže to byť prakticky ľubovoľné číslo. Ja som sa rozhodol použiť nasledovný postup: vytvoril som program, ktorý cieľovú adresu procedúry vypočíta pomocou konštanty a údaju získaného z HW kľúča. Počas ladenia programu bude k dispozícii kontrola pamäťových adries: najskôr sa prečíta hodnota s HW kľúča a potom sa táto hodnota porovná so skutočnou adresou procedúry. Pokiaľ čísla nie sú rovnaké, vyvolá sa výnimka. Nemusím pripomínať, že táto kontrolná rutina sa vo finálnej verzii vášho programu nesmie vyskytovať. Procedúru, ktorej adresu budeme vypočítavať, som nazval `KillerProc`. Obsahuje jednoduchý výpis „Hello world!“. Keďže jej adresa bude v rôznych fázach vývoja programu rôzna, je potrebné pridať ladiace funkcie, o ktorých som hovoril v predchádzajúcich odstavcoch. Najskôr si však musíme napísať aplikáciu, ktorá nám tieto adresy vypočíta. Keďže program bude pracovať s adresou procedúr, nazval som ho Proc Calc. Obsahuje tri okienka typu TEdit, z ktorých dve vypĺňa používateľ a tretia je „read-only“. Do prvej je potrebné zadať adresu procedúry, ktorú získame pomocou operátora `@`. Do druhej treba zadať hodnotu, ktorú sme obdržali od HW kľúča. Výsledné číslo sa nám zobrazí v treťom okienku. Kód asociovaný s tlačítkom vypočítaj vyzerá nasledovne:

```
procedure TProcCalcWndClass.VypocClick(Sender: TObject);
var ProcAddr:pointer;
tmp:integer;
subtract:integer;
begin
  try
    tmp:=StrToInt('$'+ PAddr.Text);
  except
   on EConvertError do
    Application.MessageBox('Neplatná adresa!','Chyba',MB_OK OR
MB_ICONERROR);
  end;//except
  try
    subtract:=StrToInt('$'+Mno.Text);
  except
   on EConvertError do
    Application.MessageBox('Neplatná hodnota!','Chyba',MB_OK OR
MB_ICONERROR);
  end;//except
 ProcAddr:=Pointer(tmp);
 asm
   mov eax,ProcAddr
   sub eax,subtract
   mov tmp,eax
 end;
```

```
 vysled.Text:=IntToHex(tmp,2);
end;
```

Potom sa vrátime k nášmu pôvodnému programu. Vytvoríme novú procedúru, ktorá bude volať procedúru `KillerProc` na základe informácií získamých z HW kľúča. Výpis tejto procedúry budem uvádzať viackrát, aby bol výklad čo najzrozumiteľnejší. V prvom rade je potrebné zistiť adresu procedúry `KillerProc`:

```
procedure TForm1.ProcTestClick(Sender: TObject);
var
  realQueryStr,ResponseStr,disResponseStr : string[80];
  status : WORD;
  response32 : longint;
  Magic:integer;
  tmp:pointer;
begin
  InitBtnClick(Sender);
  ConvertQueryStr('AAFF', realQueryStr);
  status := SproQuery (@thePacket, $20, @realQueryStr[1],
                           @responseStr[1], response32,
                           Length(realQueryStr));
  SetLength(ResponseStr,Length(realQueryStr));
  ConvertResponseStr (responseStr, disResponseStr);
  Magic:=StrToInt('$'+ disResponseStr);

  tmp:=@KillerProc;

end;
```

Hodnotu získanú z premennej `tmp` a tiež aj hodnotu F9BF (ktorú sme získali s HW kľúča) použijeme ako vstupné údaje pre program Proc Calc. Výsledné číslo potom použijeme ako deklaráciu konštanty, ktorú pridáme do procedúry. Taktiež pridáme premennú, ktorá bude obsahovať finálnu adresu procedúry `KillerProc`, získanú sčítaním konštanty `StaticAddr` a hodnoty získanej s HW kľúča:

```
procedure TForm1.ProcTestClick(Sender: TObject);
var
  realQueryStr,ResponseStr,disResponseStr : string[80];
  status : WORD;
  response32 : longint;
  Magic:integer;
  tmp:pointer;
  FinalAddr:Pointer;
  const StaticAddr=$41D4ED;
begin
 (nasleduje kód procedúry)
```

Nemusím pripomínať, že vám s veľkou pravdepodobnosťou budú vychádzať iné adresy, preto si obsah konštanty `StaticAddr` budete musieť upraviť. Záverečnou fázou bude pridanie série príkazov, ktoré ku konštante `StaticAddr` pripočítajú číslo získané z HW kľúča (v mojom prípade číslo F9BFh). Takto „zrekonštruovaná“ adresa bude uložená do premennej `FinalAddr`. Pomocou inštrukcie CALL potom bude predané riadenie na adresu definovanú premennou `FinalAddr`. Počas vývoja programu sa však bude adresa procedúry `KillerProc` pomerne často meniť, preto som pridal kontrolný kód, ktorý vyvolá výnimku v prípade, že sa „zrekonštruovaná“ a skutočná adresa nezhodujú. Tento kód však bude obsiahnutý v programe iba vtedy, ak je v programe definovaný symbol `kontrola`. Konečná podoba procedúry je nasledovná:

```
procedure TForm1.ProcTestClick(Sender: TObject);
var
  realQueryStr,ResponseStr,disResponseStr : string[80];
  status : WORD;
  response32 : longint;
  Magic:integer;
```

```
  tmp:pointer;
  FinalAddr:Pointer;
  const StaticAddr=$41D53D;
begin
  InitBtnClick(Sender);
  ConvertQueryStr('AAFF', realQueryStr);
  status := SproQuery (@thePacket, $20, @realQueryStr[1],
                          @responseStr[1], response32,
                          Length(realQueryStr));
  SetLength(ResponseStr,Length(realQueryStr));
  ConvertResponseStr (responseStr, disResponseStr);
  Magic:=StrToInt('$'+ disResponseStr);
  tmp:=@KillerProc;

  asm
    mov eax,StaticAddr
    mov ebx,Magic
    add eax,ebx
    mov FinalAddr,eax
  end;
 {$ifdef kontrola}
   if tmp<>FinalAddr then raise EAddrMismatch.Create('Address mismatch!');
 {$endif}
 asm
   call FinalAddr
 end;
end;
```

V praxi sa jedná o presne opačný postup, než aký sme použili v programe Proc Calc. V ňom sme od adresy procedúry `KillerProc` odčítali číslo F9BFh, v procedúre `ProcTestClick` sme ho zasa pripočítali. Samozrejme, pred vytvorením finálnej verzie programu je potrebné zrušiť symbol `kontrola`. Po zostavení programu by sme však pre istotu mali túto dôležitú procedúru opäť prekrokovať, aby sme sa presvedčili, že sa adresa premennej `KillerProc` nezmenila. Je veľmi dôležité, aby ste si boli stopercentne istí obsahom premennej `FinalAddr`, nakoľko v prípade chyby môže inštrukcia CALL odovzdať riadenie „do neznáma“. Náš krátky výlet do sveta Sentinelu SuperPro je na konci. Je jasné, že takýto krátky príspevok nemohol vyčerpať všetky možnosti, ktoré tento hardwarový kľúč vývojárom poskytuje. Dúfam však, že bude užitočným pomocníkom pre tých, ktorí sa s možnosťami tohto HW kľúča zatiaľ iba oboznamujú.

Ivan Zernovác ml, (ivan@gratis.sk)