MOKSLAS – LIETUVOS ATEITIS
SCIENCE – FUTURE OF LITHUANIA

ISSN 2029-2341 print / ISSN 2029-2252 online
2011 3(1): 15–19 doi: 10.3846/mla.2011.003

Sistemų inžinerija, kompiuterių technologija T 120
Systems Engineering, Computer Technology T 120

Elektronika ir elektrotechnika
Electronics and Electrical Engineering

WINDOWS API FUNKCIJŲ SEKŲ PERĖMIMO BIBLIOTEKŲ TYRIMAS

Lukas Radvilavičius[1], **Dainius Čeponis**[2]

Vilniaus Gedimino technikos universitetas
El. paštas: [1]*lukas@fmf.vgtu.lt;* [2]*vgtu.dainius@gmail.com*

Santrauka. Nagrinėjamos bibliotekos, skirtos Windows API funkcijų sekų perėmimui. Sekų perėmimas gali būti panaudojamas įvairiose srityse: siekiant išsiaiškinti operacinės sistemos veikimo principus, norint atlikti programos derinimą arba pridėti papildomą funkcionalumą prie jau esamo. Straipsnyje apžvelgiamos Windows API ir pateikiamos trečiųjų šalių priemonės funkcijų sekoms perimti. Aptariamos jų teikiamos galimybės, panaudojimo specifika. Taip pat atlikti testai, siekiant išsiaiškinti, kuri biblioteka su užduotimis susitvarko greičiausiai. Tyrimui pasirinkti du nemokami gaminiai: Microsoft Detours ir EasyHook bibliotekos.

Reikšminiai žodžiai: Windows API, funkcijų sekų perėmimas, įsiskverbimo bibliotekos, palyginimas, C++, C#, .NET, Detours, EasyHook, SetWindowsHookEx, atviras kodas.

Įvadas

Windows API (angl. *Application Programming Interface*) funkcijų perėmimas reikalauja itin gero operacinės sistemos supratimo. Kadangi tai susiję su sistemoje veikiančių procesų atminties modifikavimu, reikia būti labai atidžiam ir tiksliam. Žinoma, visa tai galioja, jeigu programuotojas funkcijų perėmimo biblioteką rašo nuo pradžių. Kai naudojamos trečiųjų šalių siūlomos bibliotekos, rizika, jog sistema bus apgadinta arba veiks nekorektiškai, smarkiai sumažėja.

Bibliotekų pasirinkimą nulemia jų panaudojimo reikalavimai. Šiuo atveju pasirinktas metodas bus naudojamas realaus laiko failų skenavimui (angl. *On-Access Scan*).

Metodas privalo turėti galimybę perimti funkcijas, kurias operacinė sistema naudoja darbui su failais. Windows operacinėje sistemoje tai atlieka *kernel32.dll* bibliotekos funkcijos *CreateFile*, *OpenFile*, *ReadFile* ir kitos.

Antras reikalavimas. Kai funkcijos jau perimtos, sistemos veikimo spartos pakitimai turi būti minimalūs. Tai galioja ir pačios funkcijos kvietimui, ir paties metodo įdiegimui į norimą procesą.

Trečias reikalavimas. Metodai (biblioteka) turi būti platinami pagal atviro kodo (AK) (angl. *GPL – General Public License*) licenciją, kadangi failų skenavimas bus atliekamas panaudojant *ClamAv* duomenų bazę.

SetWindowsHookEx funkcija

Šį metodą suteikia Windows API funkcionalumas. Pagal aprašymą jis skirtas įterpti funkciją, esančią mūsų bibliotekoje, į sistemos funkcijų grandinę. Tokiu būdu galima stebėti tam tikrus sistemos įvykius. Vienintelis būtinas reikalavimas naudojant šį mechanizmą – funkcijos pabaigoje iškviesti *CallNextHookEx*. Taip užtikrinsime, kad funkcijos kvietimas nepasibaigs pas mus, jį gaus ir kitos programos, kurios naudoja *SetWindowsHookEx* (Čeponis *et al.* 2008). Galime stebėti vieną pasirinktą arba visus šiuo metu sistemoje vykdomus procesus. Galimi stebėti įvykiai:

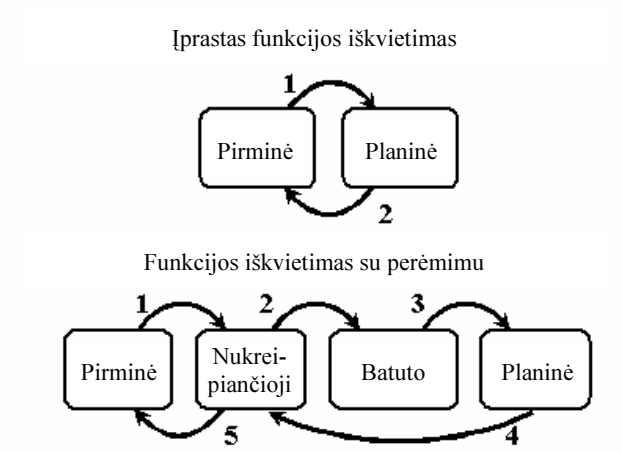
- klaviatūros veiksmai;
- pelės veiksmai;
- dialogų veiksmai;
- žinutės (angl. *Messages*) po jų apdorojimo;
- žinutės, siunčiamos ekrano langams.

SetWindowsHookEx metodo funkcionalumo pakanka stebėti tik sistemoje naudojamas žinutes. Naudojant jas, neįmanoma perimti informacijos, susijusios su darbu failinėje sistemoje.

Microsoft Detours biblioteka

Detours yra biblioteka, skirta perimti Win32 funkcijas. Autoriai – Microsoft tyrimų laboratorijos darbuotojai. Veikimo principas: ji pakeičia pačius pirmuosius funkcijos vykdymo baitus į vartotojo sukurtą funkciją, kuri pakeičia originalą. Originalios funkcijos kvietimas ap-

saugotas su nukreipiančiaja funkcija (angl. *Trampoline*). Iškvietus perimtą funkciją, kvietimas perduodamas į perrašytą jos variantą. Kai atliekamos papildomos operacijos, dėl kurių buvo perimta funkcija, kviečiama originali funkcija. Tuomet ir panaudojamas nukreipiančios funkcijos išsaugotos originalios funkcijos kvietimas (tai atliekama dėl to, kad vėl neiškviestume perrašytos funkcijos ir nepapultume į amžiną ciklą). 1 pav. parodo kaip vyksta perrašytos funkcijos kvietimas ir sąsaja su originaliąja prieš ir po perėmimo (Hunt *et al.* 1999).



1 pav. Kreipimasis prieš ir po perėmimo
Fig. 1. Invocation with and without interception

Detours biblioteka funkcijas perima perrašydama procesuose esančias jų kvietimo lenteles. Kiekvienai funkcijai biblioteka iš tikrųjų perrašo dvi funkcijas. Viena kviečiama vietoj tikrosios, o kita – aktyvuojama nukreipiančiosios.

```
;; Target Function                ;; Target Function
            ...                             ...
TargetFunction:                 TargetFunction:
  push  ebp                       jmp   DetourFunction
  mov   ebp,esp
  push  ebx                     TargetFunction+5:
  push  esi                       push  edi
  push  edi                               ...
            ...
                                ;; Trampoline
;; Trampoline                             ...
            ...                 TrampolineFunction:
TrampolineFunction:               push  ebp
  jmp  TargetFunction             mov   ebp,esp
            ...                   push  ebx
                                  push  esi
                                  jmp   TargetFunction+5
                                            ...
```

2 pav. Batuto ir originalios funkcijos programinis kodas prieš ir po Detours bibliotekos įterpimo (kairė ir dešinė)

Fig. 2. Trampoline and target functions, before and after insertion of the detour (left and right)

2 pav. pateiktas pavyzdys, kaip atrodo originalus ir perimtos funkcijos kodas atmintyje (Hunt *et al.* 1999). Visų pirma Detours išskiria atminties nukreipiančiai funkcijai. Tuomet į jos vietą nukopijuojami mažiausiai 5 baitai originalios funkcijos (tai minimalus kiekis baitų, kai galima atlikti besąlygine *jmp* komandą). Jei funkcija-taikinys yra trumpesnė nei 5 baitai, tai Detours biblioteka gražina klaidą.

Detours biblioteka leidžia perimti bet kokios bibliotekos bet kokią funkciją. Versija 32 bitų sistemoms platinama nemokamai (skirta nekomerciniam naudojimui), o 64 bitų – mokama.

EasyHook biblioteka

Šis projektas palaiko nekontroliuojamo (angl. *Unmanaged*) programinio kodo perėmimą naudodamas kontroliuojamą (angl. *Managed*) kodą. Funkcionalumas pasiekiamas naudojant C# aplinką bei naujas operacines sistemas (Windows 2000 SP4, Windows XP x64, Windows Vista x64 ir Windows Server 2008 x64) (Allaeys 2009).

EasyHook biblioteka pateikia keletą naujų dalykų. Visų pirma ji užtikrina, kad perimtoje programoje neliktų jokių resursų ir atminties šiukšlių. Biblioteka taip pat leidžia naudoti kontroliuojamas dorokles (angl. *Handler*), skirtas nekontroliuojamom API funkcijom perimti. Tai leidžia naudotis kontroliuojamo kodo suteikiamomis galimybėmis: .NET nutolusių procesų bendravimo funkcionalumu (angl. *.NET Remoting*), WCF (funkcionalumas skirtas kurti sujungtas, orientuotas į servisus programas), ir WPF (vartotojo sąsajos atvaizdavimui). Taip pat svarbu paminėti, kad EasyHook leidžia su ta pačia programa (angl. *Assemblie*) perimti 32 ir 64 bitų procesuose esančias funkcijas naudojant 32 ir 64 bitų procesus (Allaeys 2009).

EasyHook biblioteka stabilią versiją pasiekė 2009 m. kovo 8 dieną. Bet kol kas turi keletą smulkių trūkumų, susijusių su kontroliuojamu kodu, kurie bus ištaisyti kitoje versijoje.

Bibliotekų testavimas

Bibliotekų testavimas atliekamas norint išsiaiškinti, kuri biblioteka turi mažiausią poveikį perimtų procesų veikimo greičiui. Tai yra labai svarbu, kadangi pasirinkta biblioteka bus naudojama realaus laiko failų skenavimo sistemoje. Tai reiškia, kad poveikis turi būti minimalus tiek kuriant naujus procesus, tiek kviečiant perimtas funkcijas.

Norint nustatyti perimtos programos startavimo pakitimus, visų pirma išmatuojamas jos įprastinio startavimo laikas. Tam pasirinktas versijų kontrolės sistemos Git teikiamas funkcionalumas – komanda *time*. Ji įvykdo pateiktą komandų grandinę (angl. *Pipeline*) ir atspausdina ekrane sugaištą realų bei procesoriaus laiką.

Kiekvienai iš bibliotekų parašytos programos-serveriai. Jie palaiko komandinės eilutės formatą. Kviečiant abi programas, nurodoma, ką reikia atlikti, ir su kokia aplikacija. Jei duodama komanda *run*, nurodyta aplikacija yra tik paleidžiama ir laukiama, kol naujas procesas baigs savo darbą. Jei *hook* – į nurodytą aplikaciją įterpiama viena iš bibliotekų (priklausomai nuo serverio) ir perima funkcijos *CreateFile* kvietimą. 3 pav. pateiktas aplikacijos, kuri naudojama bibliotekų įsiskverbimo greičiui nustatyti, programinis kodas.

```
#include <tchar.h>

int _tmain(int argc, _TCHAR* argv[])
{
    return 0;
}
```

3 pav. Aplikacijos, kuri naudojama bibliotekų įsiskverbimo greičiui nustatyti, programinis kodas
Fig. 3. Code of application, which used to determine libraries injection speed

Kaip matome, programa neatlieka jokių užduočių, tiesiog startuoja ir iškart baigia darbą. Tai užtikrina, kad jos veikimo laiko neįtakos vykdomų komandų trukmė.

4 pav. pateiktos komandos, kurių reikėjo nustatyti programos startavimo, veikimo ir užsidarymo greitį naudojant Detours (Abramov 2008) biblioteką.

```
clear
echo Detours bibliotekos paleidimo testas.
echo Testas bus kartojamas 10 kartu.

for ((  i = 1 ;  i <= 10;  i++  ))
do
    echo
    echo
    echo Bandymas Nr.: $i
    echo RUN
    time DetoursHookCenter.exe run HookDllLoadingTest
    echo
    echo HOOK
    time DetoursHookCenter.exe hook HookDllLoadingTest
done
```

4 pav. Detours bibliotekos įsiskverbimo testavimo komandos
Fig. 4. Batch file code for Detours library injection test

Komandos kartojamos 10 kartų, kad būtų gauti kuo įvairesni rezultatai. Identiškos komandos naudojamos EasyHook (Husse 2008) testavimui, tik vietoje *DetoursHookCenter.exe* naudojama *FileMon.exe* (EasyHook serveris) programa.

Testuojant perimtų funkcijų veikimo spartos pokyčius, naudojama kita aplikacija. Ji taip pat 10 kartų atlieka failo kūrimo testą: kuriamas objektas *CFile* su nuoroda, kad failas būtų sukurtas iš naujo. Tokiu būdu iškviečiama *CreateFile* funkcija. Visa tai atliekama 20 000 kartų, laikas sumuojamas ir dalinamas iš 20 000. Taip gaunamas laikas, sugaištas sukurti *CFile* objektą (kartu ir iškviesti *CreateFile* funkciją). 5 pav. pateiktas šios aplikacijos programinis kodas.

```
for(int k = 0; k < 10; k++)
{
    time_t start, stop;
    int nCount = 20000;

    time(&start);

    for(int i = 0; i < nCount; ++i)
    {
        CFile file(_T("Test.txt"), CFile::modeCreate);
    }

    time(&stop);

    double dDiff = difftime(stop, start);
    printf("Bendras laikas     %.5f s. \n", dDiff);
    double dOneFile = dDiff / nCount;
    printf("Vieno failo laikas %.5f s. \n\n", dOneFile);
}
```

5 pav. *CreateFile* kvietimo testas
Fig. 5. *CreateFile* call test

Funkcijos kvietimo testui taip pat naudoti bibliotekų serveriai su komandomis *run* ir *hook.* 6 pav. pateiktos komandos, naudotos EasyHook bibliotekos testavimui.

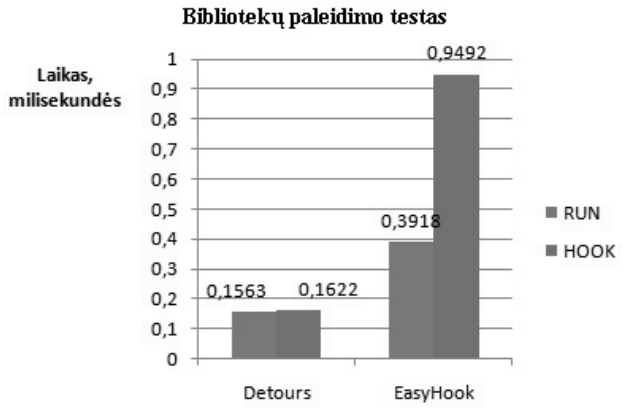
```
clear
echo EasyHook bibliotekos veikimo testas.
echo RUN
FileMon.exe run FileUsage.exe
echo
echo HOOK
FileMon.exe hook FileUsage.exe
```

6 pav. EasyHook bibliotekos *CreateFile* kvietimo testo komandos
Fig. 6. Batch file for EasyHook library *CreateFile* call test

Testavimo rezultatai

Testai atlikti Windows XP sistemoje. Kompiuterio duomenys: Intel® T2250 procesorius, operatyvioji atmintis – 3 GB.

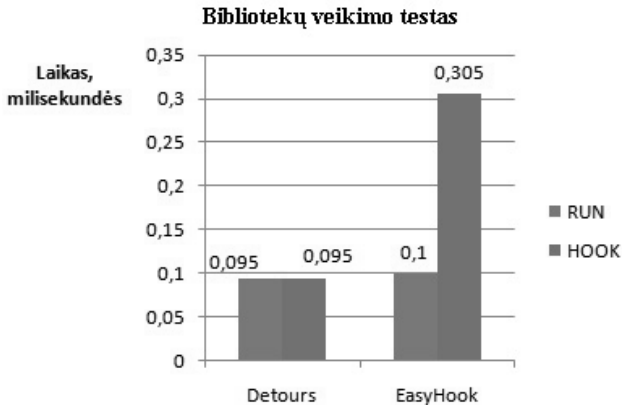
7 pav. pateiktas grafikas, vaizduojantis bibliotekų įtaką, kai programa startuoja.



7 pav. Bibliotekų paleidimo testas
Fig. 7. Libraries run test

Naudojant Detours biblioteką, aplikacijos startavimas pasikeičia labai nežymiai. Nuo 0,1563 iki 0,1622 ms. Tai įtakoja tik 3,77 % sulėtėjimą. Naudojant EasyHook biblioteką, rezultatai skiriasi net keletą kartų. Startuojant be bibliotekos – 0,3918 ms, o startuojant su EasyHook – 0,9492 ms. Tai sudaro 142,27 % sulėtėjimą ir reiškia, kad EasyHook biblioteka programos startavimą prailgino šiek tiek daugiau nei dvigubai.

8 pav. pateiktas funkcijų veikimas įprastu režimu ir perėmus jas su testuojamomis bibliotekomis.



8 pav. Bibliotekų veikimo testas
Fig. 8. Libraries work test

Kaip matome, Detours biblioteka neturėjo jokios įtakos funkcijos vykdymo laikui, o EasyHook biblioteka pateikė kitokius rezultatus. Funkcijos vykdymo laikas nuo 0,1 ms pailgėjo iki 0,305 ms. Tai sudaro 305 % pradinio laiko arba 3 kartų sulėtėjimą.

Tokių testų rezultatų buvo galima tikėtis, nes EasyHook bibliotekos dokumentacijoje neaptariamas jos greitis, o Detours dokumentacijoje bibliotekos greitis yra pabrėžiamas. 1 lentelėje pateiktas Detours bibliotekos spartos palyginimas su kitais API perėmimo mechanizmais (Hunt *et al.* 1999).

1 lentelė. Įsiskverbimo metodų palyginimas
Table 1. Comparison of interception techniques

Įsiskverbimo būdas	Funkcija	
	Tuščia	*CoCreateInstance*
Tiesioginis	0,113 µs	14,836 µs
Keičiant kreipimąsi	0,143 µs	15,193 µs
Peradresuojant biblioteką	0,143 µs	15,193 µs
Taikant Detours	0,145 µs	15,194 µs
Stabdos taško spąstais	229,564 µs	256,851 µs

Šiame teste buvo palygintas tuščios funkcijos ir *CoCreateInstance* veikimas naudojant įvairias įsiskverbimo technikas. Testas atliktas kompiuteryje su 200 MHz taktinio dažnio Pentium Pro procesoriumi (Hunt *et al.* 1999). Ir, kaip matome, Detours naudojimas turi labai mažą laiko skirtumą prieš metodus, kurie pripažinti kaip greičiausi.

Tyrimai šioje srityje

Beieškant informacijos apie minėtas bibliotekas paaiškėjo, jog tyrimai šioje srityje nėra stipriai plėtojami. Windows API perėmimo metodų ir bibliotekų palyginimų yra labai mažai. Daugiausia – pačių gaminių pristatomieji straipsniai, kuriuose apžvelgiamos gerosios savybės.

Išvados

1. Atlikus metodų apžvalgą nustatyta, kad darbui su failine sistema skirtas funkcijas perimti gali tik Detours ir EasyHook bibliotekos. Windows API pateikiama SetWindowsHookEx funkcija tinkama tik sistemoje perduodamoms žinutėms ir jų grandinėms perimti.
2. Atlikus EasyHook ir Detours bibliotekų testavimą paaiškėjo, kad EasyHook labai stipriai sulėtina tiek programos startavimą, tiek perimtų funkcijų vykdymo laiką, o Detours biblioteka turi labai nežymią arba išvis jokios įtakos minėtiems procesams.

Literatūra

Abramov, A. 2008. *API Hooking with MS Detours* [interaktyvus], [žiūrėta 2010 m. kovo 10 d.]. Prieiga per internetą: <http://www.codeproject.com/KB/DLL/funapihook.aspx>.

Allaeys, S. 2009. *Design and Implementation of a Portable Virtualization System*: bachelor thesis. Helsinki Metropolia University of Applied Sciences.

Čeponis, D.; Radvilavičius, L. 2008. Windows API funkcijų stebėjimas, procesų perėmimas ir stebėjimas, iš 11-*osios Lietuvos jaunųjų mokslininkų konferencijos „Mokslas – Lie-*

tuvos ateitis“, įvykusios Vilniuje 2008 m. balandžio mėn. 11 d., pranešimų medžiaga.

Hunt, G.; Brubacher, D. 1999. Detours: Binary Interception of Win32 Functions, in *USENIX Windows NT Symposium, Seattle*, 9.

Husse, C. 2008. *EasyHook – The reinvention of Windows API hooking* [interaktyvus], [žiūrėta 2010 m. kovo 15 d.]. Prieiga per internetą: <http://www.codeproject.com/KB/DLL/EasyHook64.aspx>.

Detours – Microsoft Research. 2010 [interaktyvus], [žiūrėta 2010 m. kovo 10 d.]. Prieiga per internetą: <http://research.microsoft.com/en-us/projects/detours>.

Husse, C. 2010. *EasyHook – The reinvention of Windows API Hooking* [interaktyvus], [žiūrėta 2010 m. kovo 15 d.]. Prieiga per internetą: < http://easyhook.codeplex.com>.

WINDOWS API HOOKING LIBRARIES RESEARCH

D. Čeponis, L. Radvilavičius

Abstract

The paper describes methods how to apply Windows API hooking with third party libraries and solutions. In this research were used Windows API function SetWindowsHookEx, Detours and EasyHook libraries. Libraries methods, features and advantages were discussed in this paper. The practical part contains libraries tests. In analysis we tested target program start with hooking library and injected function call.

Keywords: Windows API, functions interception, hooking library, research, C++, C#, .NET, Detours, EasyHook, SetWindowsHookEx, Open Source.