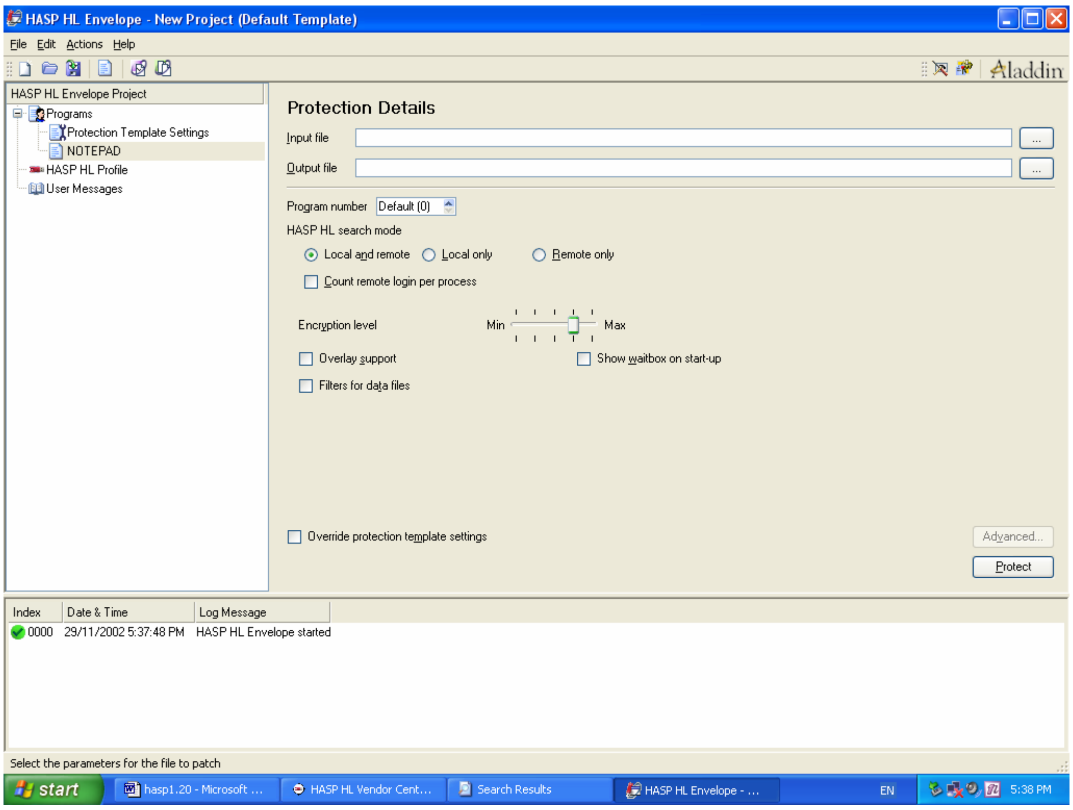
1 The HASP-HL Crack Solution

HASP-HL is the current protection hardware by Aladdin Knowledge Systems. This document explains how to crack and bypass the security of HASP-HL – “their so-called Next Generation of Software Protection”.

2 HASP-HL Envelope

The envelope encryption is done via a graphical user interface as shown in Figure 1. The encryption options are set by default to provide a high level of security. The selection of additional options (more encryption /anti debug modules, detection of user mode and system mode debuggers) has no influence on the analysis and mainly increases the size of the resulting executable and the startup time. Within the analysis there are no noticeable effects of these additional settings.



Program number: Default (0)

HASP HL search mode: [✓ Local and remote] Local only Remote only

☐ Count remote login per process

Encryption level: Min – Max

☐ Overlay support ☐ Show waitbox on start-up

☐ Filters for data files

☐ Override protection template settings

Index	Date & Time	Log Message
0000	29/11/2002 5:37:48 PM	HASP HL Envelope started

Select the parameters for the file to patch

Figure 1 Automatic encryption window of HASP-HL

The Encryption itself encrypts code and data. Resources are not encrypted even if the resource section is included in the list of sections to be encrypted. The Import Address Table (IAT) is encrypted and some APIs are redirected to the security engine (see also the detailed list of the redirected 633 APIs in the appendix). For the runtime check a separate thread is initiated. Anti debugging measures are in place which are mainly active at program startup time. Once the Security engine has decrypted the program it can be dumped to disk. After the IAT of the dump is restored and the program is reset to the Original Entry Point (OEP) the executable can be run without the HASP-HL.

2.1 Locating the Original Entry Point (OEP)

The OEP can be located using the standard hacker tool OllyDbg. Due to anti-debug measures some hardening of the debugger is required.

2.1.1 Installing OllyDbg and the necessary plugins

OllyDbg is copied into a separate directory. There is no dedicated installation necessary. OllyDbg can be obtained from various websites including the homepage of OllyDbg:

http://www.ollydbg.de/

The current version 1.10 is the preferred version.

To analyze HASP-HL it is necessary to install two plugins into OllyDbg: IsDebuggerPresent and OllyDump. These plugins are available from:

http://ollydbg.win32asmcommunity.net/stuph/

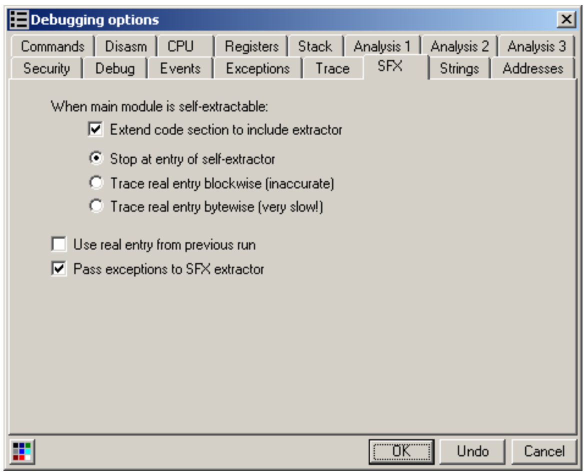
The plugins are simply copied into the same directory as OllyDbg.

2.1.2 Hardening OllyDbg

OllyDbg will be recognized by one of the several text references to OllyDbg inside the code. The usual way to evade this kind of debugger detection is to replace all occurrences of the string OllyDbg inside the executable to something else with the same length. Also the filename of the executable should be changed to this alternate name. Due to the structure of the plugins it is necessary to keep an original copy of OllyDbg in the same directory as the modified version. The string replacement can be done with any usual hex-editor.

2.1.3 Setting the options

Start the hardened OllyDbg (here for simplicity still referenced as OllyDbg) and choose from the menu Options / Debugging Options. In the options dialog select the SFX-tab and select "Extend code section to include extractor", "Stop at entry of self-extractor", and "Pass exceptions to SFX extractor" as shown in Figure 2.

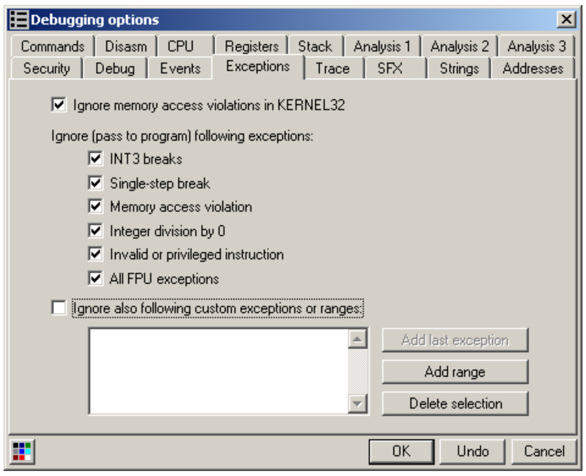


When main module is self-extractable:

- [✓] Extend code section to include extractor
- (●) Stop at entry of self-extractor
- ( ) Trace real entry blockwise (inaccurate)
- ( ) Trace real entry bytewise (very slow!)
- [ ] Use real entry from previous run
- [✓] Pass exceptions to SFX extractor

Figure 2: OllyDbg SFX Dialog

As shown in Figure 3, in the tab Exceptions select mostly everything that can be selected. This will disarm some anti-debugging involving illegal opcodes, illegal memory accesses, and some more.



Debugging options

- [✓] Ignore memory access violations in KERNEL32

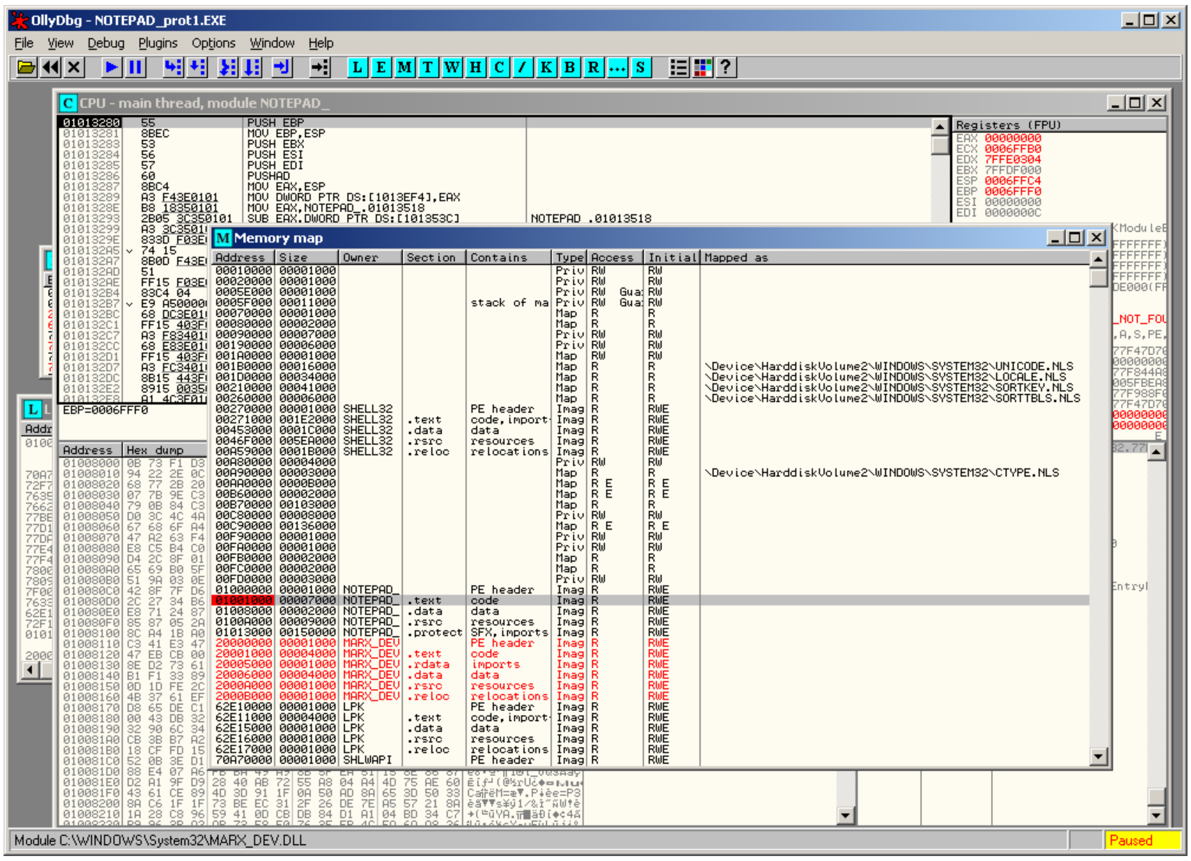
Ignore (pass to program) following exceptions:

- [✓] INT3 breaks
- [✓] Single-step break
- [✓] Memory access violation
- [✓] Integer division by 0
- [✓] Invalid or privileged instruction
- [✓] All FPU exceptions
- [ ] Ignore also following custom exceptions or ranges:

Figure 3: OllyDbg Exceptions Dialog

2.1.4 Tracing the OEP

Now load the protected application into the debugger using the File / Open from the menu. Be sure to have the original HASP-HL dongle attached to the computer when loading the program. After a short analysis and two warnings about encrypted programs etc. the debugger should halt on the entry point of the application. Turn on the debugger hiding by selecting "Plugins / IsDebuggerPresent / Hide". Alternatively you can set the "autohide" in Options Menu of the IsDebuggerPresent plugin. Switch to the hexdump section and go to the address of "ExitProcess" in Kernel32. To go to this Address use "<Ctrl> G" or the context menu. The entered Address is case sensitive. Select left topmost byte with the left mouse button. Then set a hardware breakpoint by clicking right, choosing "Breakpoint / Hardware on write / Byte". Now run the program using <F9> until it hits the breakpoint. Switch to the memory window by hitting "<Alt> M" or selection "View / Memory" from the menu and highlight the code section (usually named "text") of the protected program by left clicking it. Set a breakpoint on the section with <F2>. Your screen should look somewhat like Figure 4.



OllyDbg - NOTEPAD_prot1.EXE

CPU - main thread, module NOTEPAD_

```
01013280  55               PUSH EBP
01013281  8BEC             MOV EBP,ESP
01013283  53               PUSH EBX
01013284  56               PUSH ESI
01013285  57               PUSH EDI
01013286  60               PUSHAD
01013287  8BC4             MOV EAX,ESP
01013289  A3 F43E0101      MOV DWORD PTR DS:[1013EF4],EAX
0101328E  B8 18350101      MOV EAX,NOTEPAD_.01013518
01013293  2B05 3C350101    SUB EAX,DWORD PTR DS:[101353C]      NOTEPAD .01013518
```

```
Registers (FPU)
EAX 00000000
ECX 0006FFB0
EDX 7FFE0304
EBX 7FFDF000
ESP 0006FFC4
EBP 0006FFF0
ESI 00000000
EDI 0000000C
```

EBP=0006FFF0

Memory map

```
Address  Size     Owner     Section  Contains      Type Access   Initial Mapped as
00010000 00001000                                  Priv RW       RW
00020000 00001000                                  Priv RW       RW
0005E000 00001000                                  Priv RW  Gua: RW
0005F000 00011000                    stack of ma   Priv RW  Gua: RW
00070000 00001000                                  Map  R        R
00080000 00002000                                  Map  R        R
00090000 00007000                                  Priv RW       RW
00190000 00006000                                  Priv RW       RW
001A0000 00001000                                  Map  RW       RW
001B0000 00016000                                  Map  R        R       \Device\HarddiskVolume2\WINDOWS\SYSTEM32\UNICODE.NLS
001D0000 00034000                                  Map  R        R       \Device\HarddiskVolume2\WINDOWS\SYSTEM32\LOCALE.NLS
00210000 00041000                                  Map  R        R       \Device\HarddiskVolume2\WINDOWS\SYSTEM32\SORTKEY.NLS
00260000 00006000                                  Map  R        R       \Device\HarddiskVolume2\WINDOWS\SYSTEM32\SORTTBLS.NLS
00270000 00001000 SHELL32            PE header     Imag R        RWE
00271000 001E2000 SHELL32   .text    code,import   Imag R        RWE
00453000 0001C000 SHELL32   .data    data          Imag R        RWE
0046F000 005EA000 SHELL32   .rsrc    resources     Imag R        RWE
00A59000 0001B000 SHELL32   .reloc   relocations   Imag R        RWE
00A80000 00004000                                  Priv RW       RW
00A90000 00003000                                  Map  R        R       \Device\HarddiskVolume2\WINDOWS\SYSTEM32\CTYPE.NLS
00AA0000 0000B000                                  Map  R E      R E
00B60000 00002000                                  Map  R E      R E
00B70000 00103000                                  Map  R        R
00C80000 00008000                                  Priv RW       RW
00C90000 00136000                                  Map  R E      R E
00F90000 00001000                                  Priv RW       RW
00FA0000 00001000                                  Priv RW       RW
00FB0000 00002000                                  Map  R        R
00FC0000 00002000                                  Map  R        R
00FD0000 00003000                                  Priv RW       RW
01000000 00001000 NOTEPAD_           PE header     Imag R        RWE
01001000 00007000 NOTEPAD_  .text    code          Imag R        RWE
01008000 00002000 NOTEPAD_  .data    data          Imag R        RWE
0100A000 00009000 NOTEPAD_  .rsrc    resources     Imag R        RWE
01013000 00150000 NOTEPAD_  .protect SFX,imports   Imag R        RWE
20000000 00001000 MARX_DEV           PE header     Imag R        RWE
20001000 00004000 MARX_DEV  .text    code          Imag R        RWE
20005000 00001000 MARX_DEV  .rdata   imports       Imag R        RWE
20006000 00004000 MARX_DEV  .data    data          Imag R        RWE
2000A000 00001000 MARX_DEV  .rsrc    resources     Imag R        RWE
2000B000 00001000 MARX_DEV  .reloc   relocations   Imag R        RWE
62E10000 00001000 LPK                PE header     Imag R        RWE
62E11000 00004000 LPK       .text    code,import   Imag R        RWE
62E15000 00001000 LPK       .data    data          Imag R        RWE
62E16000 00001000 LPK       .rsrc    resources     Imag R        RWE
62E17000 00001000 LPK       .reloc   relocations   Imag R        RWE
70A70000 00001000 SHLWAPI            PE header     Imag R        RWE
```

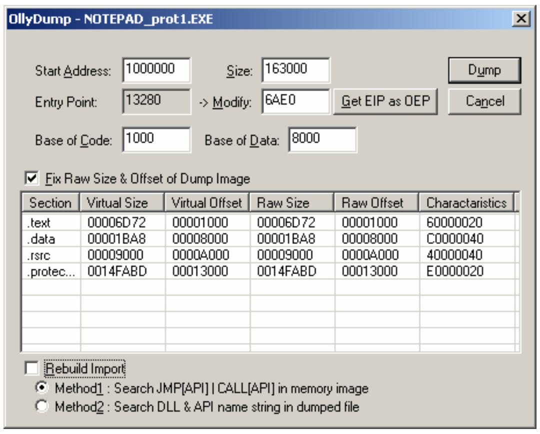
Module C:\WINDOWS\System32\MARX_DEV.DLL   Paused

Figure 4: OllyDbg with Breakpoint on code section

If you <F9> (Run) now the Debugger will put the program at the Original Entry Point (OEP). To get the relative entry point from the module start the module base (address of PE header) from the "Memory" window has to be subtracted. In this case 1,000,000. This OEP is needed in the next steps. Please leave the window open as it is still needed for the dump.

2.2 Dumping the Executable

Once the program is halted at the Original Entry Point (OEP) the program can be dumped to disk. To do this click right into the disassembler window and select "Dump debugged process". A dialog pops up in which you can save the program to disk. Please uncheck the "Rebuild Import" checkbox as this will not work correctly in this case. The entry point should be set automatically to the correct OEP. Otherwise please correct. The other setting should not be changed from the default values.



Start Address: 1000000   Size: 163000
Entry Point: 13280   -> Modify: 6AE0
Base of Code: 1000   Base of Data: 8000

[✓] Fix Raw Size & Offset of Dump Image

Section	Virtual Size	Virtual Offset	Raw Size	Raw Offset	Charactaristics
.text	00006D72	00001000	00006D72	00001000	60000020
.data	00001BA8	00008000	00001BA8	00008000	C0000040
.rsrc	00009000	0000A000	00009000	0000A000	40000040
.protec...	0014FABD	00013000	0014FABD	00013000	E0000020

[ ] Rebuild Import
[✓ Method1 : Search JMP[API] | CALL[API] in memory image]
Method2 : Search DLL & API name string in dumped file

2.3 Rebuilding the Executable

To rebuild the executable it is necessary to create a new Import Address Table (IAT). The original table has been destroyed by the encryptor and the links are rebuilt in real-time by the startup code of the protection engine. The rebuilding process can be managed using another standard hacker tool (Import Reconstructor, ImpRec).

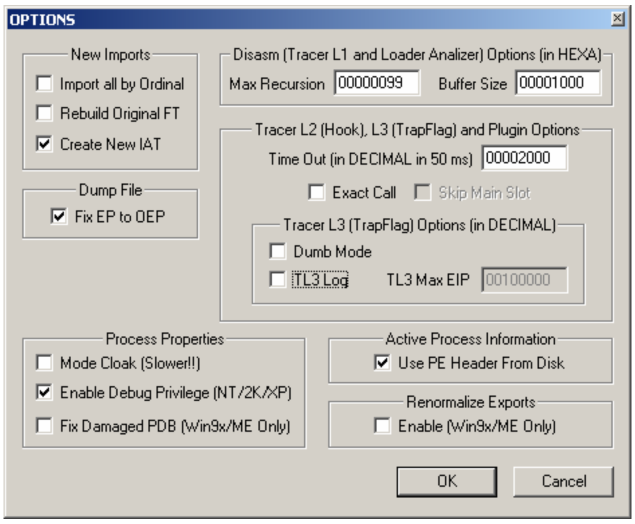
2.3.1 Installing ImpRec

ImpRec can be obtained from a number of pages including

http://wave.prohosting.com/mackt/projects/imprec/ucfir16f.zip

ImpRec does not require a dedicated installation. To install it copy it into an appropriate directory.

Be sure to select the Option "Fix EP to OEP" in the options menu (Figure 5)



```
OPTIONS
New Imports
[ ] Import all by Ordinal
[ ] Rebuild Original FT
[✓] Create New IAT
Dump File
[✓] Fix EP to OEP
Disasm (Tracer L1 and Loader Analizer) Options (in HEXA)
Max Recursion 00000099   Buffer Size 00001000
Tracer L2 (Hook), L3 (TrapFlag) and Plugin Options
Time Out (in DECIMAL in 50 ms) 00002000
[ ] Exact Call   [ ] Skip Main Slot
Tracer L3 (TrapFlag) Options (in DECIMAL)
[ ] Dumb Mode
[ ] TL3 Log   TL3 Max EIP 00100000
Process Properties
[ ] Mode Cloak (Slower!!)
[✓] Enable Debug Privilege (NT/2K/XP)
[ ] Fix Damaged PDB (Win9x/ME Only)
Active Process Information
[✓] Use PE Header From Disk
Renormalize Exports
[ ] Enable (Win9x/ME Only)
OK   Cancel
```

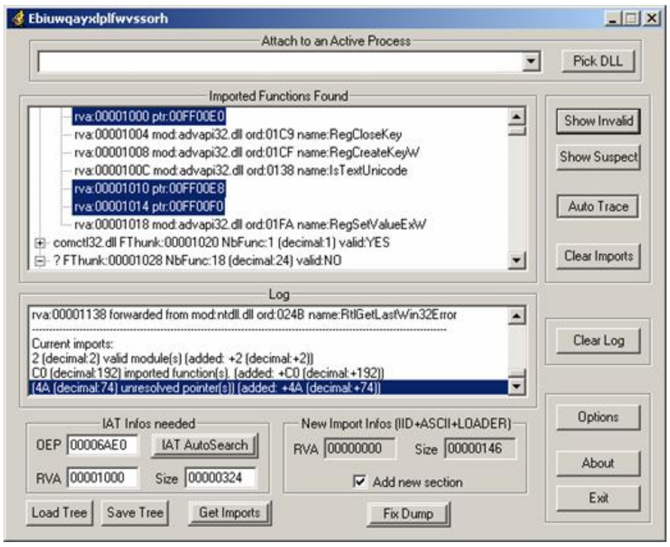
Figure 5: ImRec Options Menu

2.3.2 Rebuilding the IAT

Start the protected application. Once it is running start ImpRec. If ImpRec is running at program startup it will be detected as a Debugger. Do not forget to connect the dongle.

Attach ImpRec to the running process by selecting the correct entry in the drop down list.

Enter the OEP calculated above and hit the button “IAT AutoSearch”. The button “Get Imports” delivers the list of imported APIs with still some entries invalid. To see the invalid entries hit the button “Show Invalid”. This should result in a list similar to the list below (Figure 6) with some entries (in this case 74) are highlighted. As the following process might involve some iterations hit the “Save Tree” and store the results found so far in a text file. After restarting ImpRec and re-attaching to the executable this file can be loaded using “Load Tree”.



```
Ebiuwqayxlplfwvssorh
Attach to an Active Process
Imported Functions Found
    rva:00001000 ptr:00FF00E0
    rva:00001004 mod:advapi32.dll ord:01C9 name:RegCloseKey
    rva:00001008 mod:advapi32.dll ord:01CF name:RegCreateKeyW
    rva:0000100C mod:advapi32.dll ord:0138 name:IsTextUnicode
    rva:00001010 ptr:00FF00E8
    rva:00001014 ptr:00FF00F0
    rva:00001018 mod:advapi32.dll ord:01FA name:RegSetValueExW
+ comctl32.dll FThunk:00001020 NbFunc:1 (decimal:1) valid:YES
- ? FThunk:00001028 NbFunc:18 (decimal:24) valid:NO
Log
rva:00001138 forwarded from mod:ntdll.dll ord:024B name:RtlGetLastWin32Error
--------------------------------------------------------------
Current imports:
2 (decimal:2) valid module(s) (added: +2 (decimal:+2))
C0 (decimal:192) imported function(s). (added: +C0 (decimal:+192))
(4A (decimal:74) unresolved pointer(s)) (added: +4A (decimal:+74))
IAT Infos needed
OEP 00006AE0   RVA 00001000   Size 00000324
New Import Infos (IID+ASCII+LOADER)
RVA 00000000   Size 00000146
[✓] Add new section
```

Figure 6: ImpRec invalid entries

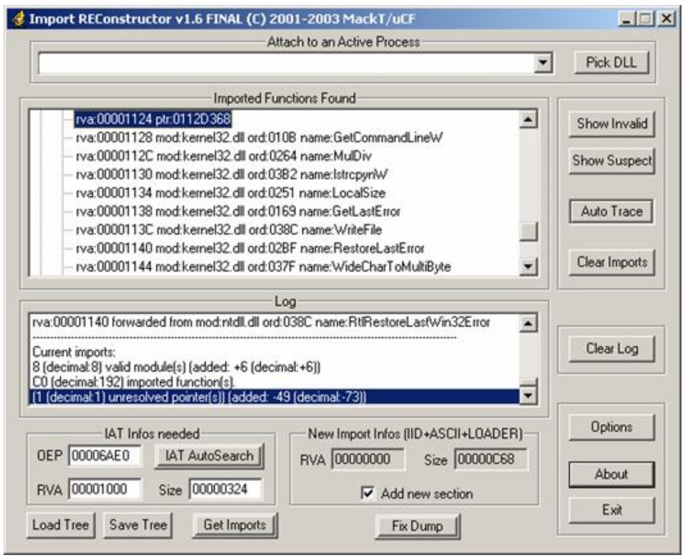
Now right click on an invalid API and select “Trace Level3 (Trap Flag)”. ImpRec should now resolve the invalid APIs one by one. Eventually the protected application will terminate or ImpRec is “unable to initialize the tracer”. In this case save the tree again, exit ImpRec, restart the application, restart ImpRec, Attach to the process, load the tree, hit “Show Invalid”, and manually deselect the first invalid entry (<Ctrl> <left click>) to stop ImpRec from resolving this entry again. If you have to restart multiply, you should successively deselect all problematic entries. With HASP-HL 1.20 there should be a maximum of 3 different problematic entries (out of several hundred redirected entries). The problematic entries are:

1. GetVersion
2. GetProcAddress

3. ExitProcess

2.3.3 Resolving the problematic 3 entries

Resolving these three entries requires either a little trial and error or a short debugging session. For the invalid entries ImpRec offers the Address to which the API has been redirected. In the example below (Figure 7) it would be the address "011D368". In a new debugger session (the old one will be detected as a debugger session because of the long delay) the address should be entered as a hardware breakpoint by calling <Ctrl G> in the disassembler window and than setting a hardware breakpoint on execution using a right click. A hardware breakpoint is needed here as the code has still to be decrypted. If a breakpoint occurs a short inspection reveals the correct entry. "GetVersion" is usually called at the beginning. "GetProcAddress" has the Address of an API and a module handle on the stack (lower right in the CPU windows, see Figure 8, were the entry "RegisterPenApp" and a reference to kerner32 is passed on the stack). "ExitProcess" is called to terminate the program. In Delphi-applications some of these APIs can be referenced multiply. If all APIs have been identified they can be entered in ImpRec by double clicking the respective invalid API. A list of possible APIs pops up and the correct one can be selected (see Figure 9). This leads to a completed import tree.



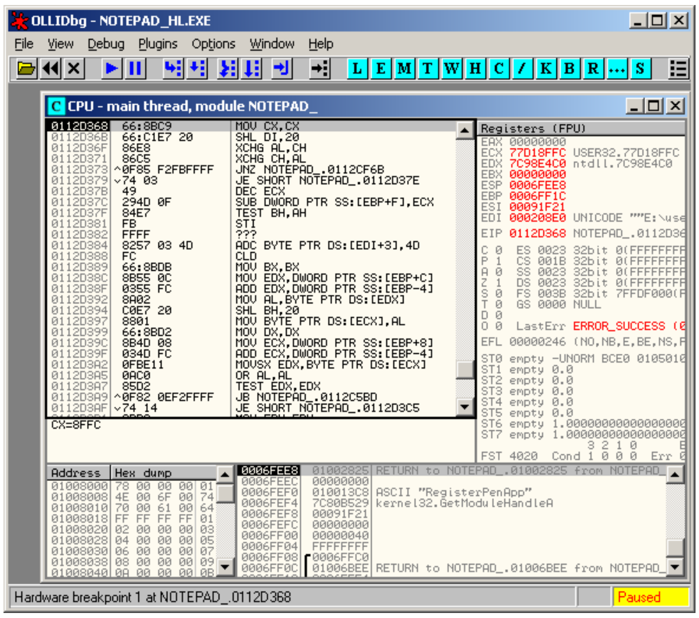
```
rva:00001124 ptr:0112D368
rva:00001128 mod:kernel32.dll ord:010B name:GetCommandLineW
rva:0000112C mod:kernel32.dll ord:0264 name:MulDiv
rva:00001130 mod:kernel32.dll ord:03B2 name:lstrcpynW
rva:00001134 mod:kernel32.dll ord:0251 name:LocalSize
rva:00001138 mod:kernel32.dll ord:0169 name:GetLastError
rva:0000113C mod:kernel32.dll ord:038C name:WriteFile
rva:00001140 mod:kernel32.dll ord:02BF name:RestoreLastError
rva:00001144 mod:kernel32.dll ord:037F name:WideCharToMultiByte
```

```
rva:00001140 forwarded from mod:ntdll.dll ord:038C name:RtlRestoreLastWin32Error
------------------------------------------------------------------------
Current imports:
8 (decimal:8) valid module(s) (added: +6 (decimal:+6))
C0 (decimal:192) imported function(s).
(1 (decimal:1) unresolved pointer(s)) (added: -49 (decimal:-73))
```

OEP 00006AE0 RVA 00001000 Size 00000324

RVA 00000000 Size 00000C68 [✓ Add new section]

Figure 7: Address of redirected function



```
0112D368   66:8BC9          MOV CX,CX
0112D36B   66:C1E7 20       SHL DI,20
0112D36F   86E8             XCHG AL,CH
0112D371   86C5             XCHG CH,AL
0112D373  ^0F85 F2FBFFFF    JNZ NOTEPAD_.0112CF6B
0112D379  v74 03            JE SHORT NOTEPAD_.0112D37E
0112D37B   49               DEC ECX
0112D37C   294D 0F          SUB DWORD PTR SS:[EBP+F],ECX
0112D37F   84E7             TEST BH,AH
0112D381   FB               STI
0112D382   FFFF             ???
0112D384   8257 03 4D       ADC BYTE PTR DS:[EDI+3],4D
0112D388   FC               CLD
0112D389   66:8BDB          MOV BX,BX
0112D38C   8B55 0C          MOV EDX,DWORD PTR SS:[EBP+C]
0112D38F   0355 FC          ADD EDX,DWORD PTR SS:[EBP-4]
0112D392   8A02             MOV AL,BYTE PTR DS:[EDX]
0112D394   C0E7 20          SHL BH,20
0112D397   8801             MOV BYTE PTR DS:[ECX],AL
0112D399   66:8BD2          MOV DX,DX
0112D39C   8B4D 08          MOV ECX,DWORD PTR SS:[EBP+8]
0112D39F   034D FC          ADD ECX,DWORD PTR SS:[EBP-4]
0112D3A2   0FBE11           MOVSX EDX,BYTE PTR DS:[ECX]
0112D3A5   0AC0             OR AL,AL
0112D3A7   85D2             TEST EDX,EDX
0112D3A9  ^0F82 0EF2FFFF    JB NOTEPAD_.0112C5BD
0112D3AF  v74 14            JE SHORT NOTEPAD_.0112D3C5
CX=8FFC
```

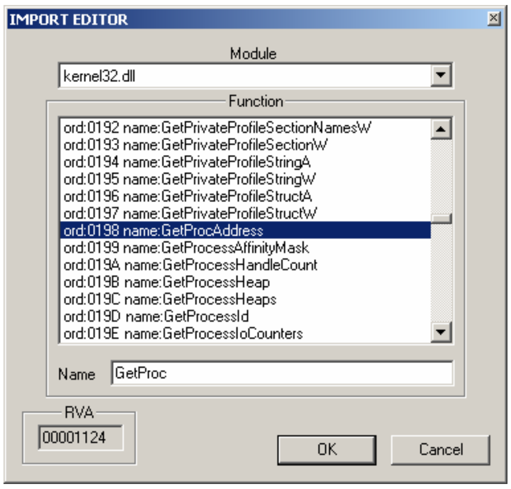
```
Registers (FPU)
EAX 00000000
ECX 77D18FFC USER32.77D18FFC
EDX 7C98E4C0 ntdll.7C98E4C0
EBX 00000000
ESP 0006FEE8
EBP 0006FF1C
ESI 00091F21
EDI 000208E0 UNICODE ""E:\use
EIP 0112D368 NOTEPAD_.0112D36
C 0  ES 0023 32bit 0(FFFFFFFF
P 1  CS 001B 32bit 0(FFFFFFFF
A 0  SS 0023 32bit 0(FFFFFFFF
Z 1  DS 0023 32bit 0(FFFFFFFF
S 0  FS 003B 32bit 7FFDF000(F
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (0
EFL 00000246 (NO,NB,E,BE,NS,P
ST0 empty -UNORM BCE0 0105010
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 1.000000000000000
ST7 empty 1.000000000000000
               3 2 1 0      E
FST 4020  Cond 1 0 0 0  Err 0
```

```
Address   Hex dump
01008000  78 00 00 00 01
01008008  4E 00 6F 00 74
01008010  70 00 61 00 64
01008018  FF FF FF FF 01
01008020  02 00 00 00 03
01008028  04 00 00 00 05
01008030  06 00 00 00 07
01008038  08 00 00 00 09
01008040  0A 00 00 00 0B
```

```
0006FEE8   01002825  RETURN to NOTEPAD_.01002825 from NOTEPAD_
0006FEEC   00000000
0006FEF0   010013C8  ASCII "RegisterPenApp"
0006FEF4   7C80B529  kernel32.GetModuleHandleA
0006FEF8   00091F21
0006FEFC   00000000
0006FF00   00000040
0006FF04   FFFFFFFF
0006FF08   0006FFC0
0006FF0C   01006BEE  RETURN to NOTEPAD_.01006BEE from NOTEPAD_
```

Hardware breakpoint 1 at NOTEPAD_.0112D368   Paused

Figure 8: Resolving additional APIs


IMPORT EDITOR

Module: kernel32.dll

Function

```
ord:0192 name:GetPrivateProfileSectionNamesW
ord:0193 name:GetPrivateProfileSectionW
ord:0194 name:GetPrivateProfileStringA
ord:0195 name:GetPrivateProfileStringW
ord:0196 name:GetPrivateProfileStructA
ord:0197 name:GetPrivateProfileStructW
ord:0198 name:GetProcAddress
ord:0199 name:GetProcessAffinityMask
ord:019A name:GetProcessHandleCount
ord:019B name:GetProcessHeap
ord:019C name:GetProcessHeaps
ord:019D name:GetProcessId
ord:019E name:GetProcessIoCounters
```

Name: GetProc

RVA: 00001124

Figure 9: Selecting a resolved API

2.3.4 Fixing the executable

After all invalid entries have been fixed the dump can be adjusted to contain the correct OEP and a correct IAT. To fix the executable it is positively necessary to have ImpRec attached to the running program. Again due to debugger detection ImpRec has to be started after the protected program. If necessary load the completed tree file. Now hit the button “FixDump” and select the dump created in step 2.2 using OllyDebug. The new executable will contain an appended underscore and should be able to be run without a HASP-HL dongle.

3 Conclusion

The automatic protection by HASP-HL from Aladdin knowledge systems provides a limited protection that can be removed with some experience and prior knowledge within a few minutes of time. If no dongle is available a restoration is not possible due to the encryption. The large number of redirected APIs (up to several hundred) and the considerable number of anti-debugging modules (50) seems to provide absolutely no security. Even after a closer look the purpose of the anti-debugging modules is unclear as they show no effect at all. The large number of redirects can be reduced to 3 by use of automatic tools. Here too the reason for the large number remains unclear as it has no effect on the security.

4 Appendices

4.1 List of redirected APIs

Some APIs that are redirected by HASP-HL 1.20 are identified and are listed here for completeness

4.1.1 Kernel32.dll

kernel32.dll, WriteTapemark, WriteProcessMemory, WritePrivateProfileStructW, WritePrivateProfileStructA, WritePrivateProfileStringW, WritePrivateProfileStringA, WriteFileEx, WriteFile, WriteConsoleOutputW, WriteConsoleOutputCharacter, WriteConsoleOutputAttribute, WinExec, WideCharToMultiByte, WaitNamedPipeW, WaitNamedPipeA, WaitForDebugEvent, WaitCommEvent, VirtualUnlock, VirtualQueryEx, VirtualQuery, VirtualProtectEx, VirtualProtect, VirtualLock, VirtualFreeEx, VirtualFree, VirtualAlloc, VerifyVersionInfoA, VerSetConditionMask, VerLanguageNameA, UpdateResourceW, UpdateResourceA, UnlockFileEx, UnlockFile, TransmitCommChar, SystemTimeToFileTime, SuspendThread, SetupComm, SetWaitableTimer, SetVolumeLabelA, SetThreadPriorityBoost, SetThreadPriority, SetThreadExecutionState, SetThreadAffinityMask, SetTapePosition, SetTapeParameters, SetSystemTimeAdjustment, SetSystemTime, SetStdHandle, SetProcessWorkingSetSize, SetProcessPriorityBoost, SetPriorityClass, SetNamedPipeHandleState, SetMailslotInfo, SetLocaleInfoA, SetLocalTime, RestoreLastError, SetHandleInformation, SetFirmwareEnvironmentVaria, SetFileValidData, SetFileTime, SetFileShortNameW, SetFilePointerEx, SetFilePointer, SetFileAttributesW, SetErrorMode, SetEnvironmentVariableW, SetEnvironmentVariableA, SetEndOfFile, SetCurrentDirectoryW, SetComputerNameW, SetComputerNameA, SetCommTimeouts, SetCommState, SetCommMask, SetCommConfig, SearchPathW, SearchPathA, ScrollConsoleScreenBufferW, ScrollConsoleScreenBufferA, ResumeThread, ResetWriteWatch, RemoveDirectoryW, ReadProcessMemory, ReadFileEx, ReadDirectoryChangesW, ReadConsoleW, ReadConsoleOutputW, ReadConsoleOutputCharacterW, ReadConsoleOutputAttribute, ReadConsoleOutputA, ReadConsoleA, RaiseException, QueueUserAPC, QueryPerformanceFrequency, QueryPerformanceCounter, QueryDosDeviceA, QueryDepthSList, PurgeComm, ProcessIdToSessionId, PrepareTape, OutputDebugStringW, OpenWaitableTimerW, OpenWaitableTimerA, OpenThread, OpenSemaphoreW, OpenSemaphoreA, OpenProcess, OpenMutexW, OpenMutexA, OpenFileMappingW, OpenFileMappingA, OpenEventW, OpenEventA, MultiByteToWideChar, MapViewOfFileEx, MapViewOfFile, LockFileEx, LockFile, LocalReAlloc, LocalLock, LocalFree, LocalAlloc, LCMapStringW, LCMapStringA, IsValidLocale, HeapWalk, HeapValidate, HeapUnlock, HeapSize, HeapSetInformation, HeapReAlloc, HeapQueryInformation, HeapLock, HeapFree, HeapDestroy, HeapCreate, HeapCompact, HeapAlloc, GlobalWire, GlobalUnfix, GlobalUnWire, GlobalMemoryStatusEx, GlobalMemoryStatus, GlobalFix, GetWriteWatch, GetVersionExA, GetVersion, GetUserGeoID, GetUserDefaultLCID, GetTimeFormatW, GetTimeFormatA, GetThreadTimes, GetThreadSelectorEntry, GetThreadPriorityBoost, GetThreadPriority, _hwrite, GetTempPathA, GetTempFileNameW, GetTempFileNameA, GetTapePosition, GetTapeParameters, GetSystemWindowsDirectoryA, GetSystemTimeAdjustment, GetSystemTime, GetSystemPowerStatus, GetSystemInfo, GetSystemDirectoryA, GetStringTypeExW, GetStringTypeExA, GetStringTypeA, GetQueuedCompletionStatus, GetProfileStringW, GetProfileStringA, GetProcessWorkingSetSize, GetProcessTimes, GetProcessPriorityBoost, GetProcessId, GetProcessHeaps, GetProcessAffinityMask, GetProcAddress, GetPrivateProfileStructW, GetPrivateProfileStructA, GetPrivateProfileStringW, GetPrivateProfileStringA, GetPrivateProfileSectionW, GetPrivateProfileSectionA, GetPrivateProfileIntW, GetPrivateProfileIntA, GetPriorityClass, GetNumberFormatA, GetNumaHighestNodeNumber, GetNamedPipeInfo, GetNamedPipeHandleStateW, GetNamedPipeHandleStateA, GetModuleHandleW, GetModuleFileNameA, GetMailslotInfo, GetLogicalDrives, GetLogicalDriveStringsW, GetLogicalDriveStringsA, GetLocaleInfoA, GetLocalTime, GetHandleInformation, GetFullPathNameA, GetFirmwareEnvironmentVaria, GetFileType, GetFileTime, GetFileSizeEx, GetFileSize, GetFileAttributesW, GetFileAttributesExW, GetExitCodeThread, GetExitCodeProcess, GetEnvironmentVariableW, GetEnvironmentVariableA, GetEnvironmentStrings, GetDriveTypeW, GetDiskFreeSpaceW, GetDiskFreeSpaceExW, GetDiskFreeSpaceExA, GetDiskFreeSpaceA, GetDevicePowerState, GetDateFormatW, GetDateFormatA, GetCurrencyFormatW, GetCurrencyFormatA, GetComputerNameW, GetComputerNameA, GetCompressedFileSizeW, GetCommTimeouts, GetCommState, GetCommProperties, GetCommModemStatus, GetCommMask, GetCommConfig, GetCPInfoExA, GetBinaryType, FormatMessageW, FormatMessageA, FoldStringW, FoldStringA, FlushViewOfFile, FlushInstructionCache, FlushFileBuffers, FindNextFileA, FindFirstFileExW, FindFirstFileExA, FindFirstFileA, FindFirstChangeNotification, FillConsoleOutputCharacterW, FillConsoleOutputAttribute, FileTimeToSystemTime, FileTimeToDosDateTime, FatalAppExitW, FatalAppExitA, ExpandEnvironmentStringsW, ExpandEnvironmentStringsA, ExitProcess, EscapeCommFunction, EraseTape, EndUpdateResourceW, DosDateTimeToFileTime, DisconnectNamedPipe, DeleteFileW, DeleteFiber, DefineDosDeviceW, DefineDosDeviceA, DebugSetProcessKillOnExit, CreateWaitableTimerA, CreateThread, CreateTapePartition, CreateSemaphoreW, CreateSemaphoreA, CreateProcessW, CreateProcessA, CreatePipe, CreateNamedPipeW, CreateNamedPipeA, CreateMutexW, CreateMutexA, CreateMailslotW, CreateMailslotA, CreateIoCompletionPort, CreateFileW, CreateFileMappingW, CreateFileMappingA, CreateFileA, CreateFiberEx, CreateEventW, CreateEventA, CreateDirectoryW, CreateDirectoryExA, CopyFileExA, CopyFileA, ContinueDebugEvent, ConnectNamedPipe, CompareStringW, CompareStringA, ClearCommError, CancelIo, CallNamedPipeA, BuildCommDCBW, BuildCommDCBAndTimeoutsW, BuildCommDCBA, BeginUpdateResourceW, BeginUpdateResourceA, Beep, BackupWrite, BackupSeek, BackupRead, _hread, _hwrite, lstrcmpi, CreateWaitableTimerW

4.1.2 User32.dll

IsCharAlphaW, IsCharAlphaNumericW, IsCharAlphaNumericA, IsCharAlphaA, InsertMenuW, InsertMenuItemW, InsertMenuItemA, InsertMenuA, GrayStringW, GrayStringA, GetWindowThreadProcessId, GetUserObjectSecurity, GetUserObjectInformationA, GetThreadDesktop, GetTabbedTextExtentW, GetTabbedTextExtentA, GetMessageW, GetMessageA, GetMenuStringW, GetMenuStringA, GetMenuItemInfoW, GetMenuItemInfoA, GetKeyboardType, GetKeyboardLayoutNameW, GetDlgItemTextW, GetClipboardData, GetClassNameW, GetClassInfoW, GetClassInfoExW, GetClassInfoExA, GetClassInfoA, FillRect, ExitWindowsEx, EnableMenuItem, DrawTextW, DrawTextExW, DrawTextExA, DrawTextA, DrawStateW, DrawStateA, DrawIconEx, DrawEdge, DlgDirSelectExA, DlgDirSelectComboBoxExA, DlgDirListW, DlgDirListComboBoxW, DialogBoxParamW, DialogBoxParamA, DialogBoxIndirectParamW, DialogBoxIndirectParamA, DefFrameProcW, DefFrameProcA, CreateWindowStationW, CreateWindowStationA, CreateWindowExW, CreateWindowExA, CreateMDIWindowW, CreateMDIWindowA, CreateIconFromResourceEx, CreateIcon, CreateDialogParamW, CreateDialogParamA, CreateDialogIndirectParamW, CreateDialogIndirectParamA, CreateDesktopW, CreateDesktopA, CreateCursor, CopyImage, CopyIcon, CheckMenuRadioItem, CheckMenuItem, CharUpperW, CharUpperBuffA, CharUpperA, CharToOemBuffW, IsCharUpperA, CharPrevW, CharPrevExA, CharPrevA, CharNextW, CharLowerW, CharLowerBuffA, CharLowerA, ChangeMenuW, ChangeMenuA, IsCharLowerA, ChangeDisplaySettingsExA, CascadeWindows, CallWindowProcW, CallWindowProcA, CallNextHookEx, CallMsgFilterW, CallMsgFilter, BroadcastSystemMessageW, BroadcastSystemMessage, AppendMenuW, AppendMenuA, MessageBoxExW, MessageBoxIndirectA, MessageBoxIndirectW, ModifyMenuA, ModifyMenuW, MsgWaitForMultipleObjects, MsgWaitForMultipleObjectsEx, OemToCharBuffW, OpenClipboard, OpenDesktopA, OpenDesktopW, OpenWindowStationA, OpenWindowStationW, PeekMessageA, PeekMessageW, PostMessageA, PostMessageW, PostThreadMessageA, RegisterClassA, RegisterClassW, RegisterClipboardFormatA, RegisterClipboardFormatW, RegisterClipboardFormatA, RegisterClipboardFormatW, RemovePropA, RemovePropW, ScrollDC, ScrollWindow, ScrollWindowEx, SendDlgItemMessageA, SendDlgItemMessageW, SendMessageA, SendMessageCallbackA, SendMessageCallbackW, SendMessageTimeoutA, SendMessageTimeoutW, SendMessageW, SendNotifyMessageA, SendNotifyMessageW, SetClassLongA, SetClassLongW, SetClipboardData, SetDlgItemInt, SetMenuItemBitmaps, SetMenuItemInfoA, SetMenuItemInfoW, SetPropA, SetPropW, SetRect, SetScrollPos, SetScrollRange, SubtractRect, TabbedTextOutA, TabbedTextOutW, TileWindows, ToAscii, ToAsciiEx, ToUnicode, ToUnicodeEx, TrackPopupMenu, UnionRect, UnregisterClassA, UnregisterClassW, VkKeyScanA, VkKeyScanExA, WaitForInputIdle, WinHelpA, WinHelpW, keybd_event, mouse_event, wvsprintfA, wvsprintfW, IsCharUpperW, IsDialogMessageW, LoadBitmapA, LoadCursorA, LoadCursorFromFileA, LoadIconA, LoadImageA, LoadImageW, LoadKeyboardLayoutA, LoadMenuIndirectA, LoadMenuIndirectA, ChangeDisplaySettingsExW, IsCharLowerW, LoadStringA, LookupIconIdFromDirectoryEx, MapVirtualKeyExA, MessageBoxExA

4.1.3 Advapi32.dll

SetFileSecurityW, SetFileSecurityA, RevertToSelf, RegisterEventSourceW, RegisterEventSourceA, RegUnLoadKeyW, RegUnLoadKeyA, RegSetValueW, RegSetValueA, RegSetKeySecurity, RegSaveKeyW, RegSaveKeyExW, RegSaveKeyExA, RegSaveKeyA, RegRestoreKeyW, RegRestoreKeyA, RegReplaceKeyW, RegReplaceKeyA, RegQueryValueW, RegQueryValueExW, RegQueryValueExA, RegQueryValueA, RegQueryMultipleValuesW, RegQueryMultipleValuesA, RegQueryInfoKeyW, RegQueryInfoKeyA, RegOpenUserClassesRoot, RegOpenKeyExW, RegOpenKeyExA, RegNotifyChangeKeyValue, RegLoadKeyW, RegLoadKeyA, RegGetKeySecurity, RegFlushKey, RegEnumValueW, RegEnumValueA, RegEnumKeyW, RegEnumKeyExW, RegEnumKeyExA, RegEnumKeyA, RegDeleteValueW, RegDeleteValueA, RegDeleteKeyW, RegDeleteKeyA, RegCreateKeyExW, RegCreateKeyExA, RegCreateKeyA, RegConnectRegistryW, RegConnectRegistryA, ReadEventLogW, ReadEventLogA, PrivilegedServiceAuditAlarm, PrivilegeCheck, OpenEventLogW, OpenEventLogA, OpenEncryptedFileRawA, OpenBackupEventLogW, OpenBackupEventLogA, ObjectPrivilegeAuditAlarmW, ObjectPrivilegeAuditAlarmA, ObjectOpenAuditAlarmW, ObjectOpenAuditAlarmA, ObjectDeleteAuditAlarmW, ObjectDeleteAuditAlarmA, ObjectCloseAuditAlarmW, ObjectCloseAuditAlarmA, MakeAbsoluteSD, LookupPrivilegeValueW, LookupPrivilegeValueA, LookupPrivilegeNameW, LookupPrivilegeNameA, LookupPrivilegeDisplayNameW, LookupAccountSidW, LookupAccountSidA, LookupAccountNameW, LookupAccountNameA, LogonUserW, LogonUserExW, LogonUserExA, LogonUserA, IsTokenUntrusted, IsTokenRestricted, InitiateSystemShutdownW, InitiateSystemShutdownExW, InitiateSystemShutdownExA, InitiateSystemShutdownA, ImpersonateNamedPipeClient, ImpersonateLoggedOnUser, GetTokenInformation, GetSecurityDescriptorSacl, GetSecurityDescriptorOwner, GetSecurityDescriptorGroup, GetSecurityDescriptorDacl, GetPrivateObjectSecurity, GetKernelObjectSecurity, GetFileSecurityW, GetFileSecurityA, GetEventLogInformation, AbortSystemShutdownA, AccessCheck, AccessCheckAndAuditAlarmA, AccessCheckAndAuditAlarmW, AddAce, AddAuditAccessAce, AdjustTokenGroups, AdjustTokenPrivileges, AllocateAndInitializeSid, BackupEventLogA, BackupEventLogW, CheckTokenMembership, ClearEventLogA, ClearEventLogW, CreatePrivateObjectSecurity, CreateProcessAsUserW, FileEncryptionStatusA, EncryptFileA, DuplicateTokenEx, DecryptFileA, CreateRestrictedToken