**Entry Point is from Process.spawn or IO.popen**

**Entry Point in Ruby**  
static rb\_pid\_t  
w32\_spawn(int mode, const char \*cmd, const char \*prog, UINT cp)

ret = child\_result(CreateChild(wcmd, wshell, NULL, NULL, NULL, NULL, 0), mode);

static rb\_pid\_t

w32\_aspawn\_flags(int mode, const char \*prog, char \*const \*argv, DWORD flags, UINT cp)

…  
ret = child\_result(CreateChild(wcmd, wprog, NULL, NULL, NULL, NULL, flags), mode);

Ruby CreateChild(….)  
  
static rb\_pid\_t  
CreateChild(const WCHAR \*cmd, const WCHAR \*prog, SECURITY\_ATTRIBUTES \*psa,  
 HANDLE hInput, HANDLE hOutput, HANDLE hError, DWORD dwCreationFlags)  
{  
 BOOL fRet;  
 STARTUPINFOW aStartupInfo;  
 PROCESS\_INFORMATION aProcessInformation;  
 SECURITY\_ATTRIBUTES sa;  
 struct ChildRecord \*child;  
  
 if (!cmd && !prog) {  
 errno = EFAULT;  
 return NULL;  
 }  
  
 child = FindFreeChildSlot();  
 if (!child) {  
 errno = EAGAIN;  
 return NULL;  
 }  
  
 if (!psa) {  
 sa.nLength = sizeof (SECURITY\_ATTRIBUTES);  
 sa.lpSecurityDescriptor = NULL;  
 sa.bInheritHandle = TRUE;  
 psa = &sa;  
 }  
  
 memset(&aStartupInfo, 0, sizeof(aStartupInfo));  
 memset(&aProcessInformation, 0, sizeof(aProcessInformation));  
 aStartupInfo.cb = sizeof(aStartupInfo);  
 aStartupInfo.dwFlags = STARTF\_USESTDHANDLES;  
 if (hInput) {  
 aStartupInfo.hStdInput = hInput;  
 }  
 else {  
 aStartupInfo.hStdInput = GetStdHandle(STD\_INPUT\_HANDLE);  
 }  
 if (hOutput) {  
 aStartupInfo.hStdOutput = hOutput;  
 }  
 else {  
 aStartupInfo.hStdOutput = GetStdHandle(STD\_OUTPUT\_HANDLE);  
 }  
 if (hError) {  
 aStartupInfo.hStdError = hError;  
 }  
 else {  
 aStartupInfo.hStdError = GetStdHandle(STD\_ERROR\_HANDLE);  
 }  
  
 dwCreationFlags |= NORMAL\_PRIORITY\_CLASS;  
  
 if (lstrlenW(cmd) > 32767) {  
 child->pid = 0; /\* release the slot \*/  
 errno = E2BIG;  
 return NULL;  
 }  
  
 RUBY\_CRITICAL({  
 fRet = CreateProcessW(prog, (WCHAR \*)cmd, psa, psa,  
 ->bInheritHandle, dwCreationFlags, NULL, NULL,  
 &aStartupInfo, &aProcessInformation);  
 errno = map\_errno(GetLastError());  
 });  
  
 if (!fRet) {  
 child->pid = 0; /\* release the slot \*/  
 return NULL;  
 }  
  
 CloseHandle(aProcessInformation.hThread);  
  
 child->hProcess = aProcessInformation.hProcess;  
 child->pid = (rb\_pid\_t)aProcessInformation.dwProcessId;  
  
 return child;  
}

Ruby createChild(..) disassembly may be partial  
00000000649BA1BA 66:0F1F4400 00 nop word ptr ds:[rax+rax],ax   
00000000649BA1C0 55 push rbp   
00000000649BA1C1 41:57 push r15   
00000000649BA1C3 41:56 push r14   
00000000649BA1C5 41:55 push r13 `r13:L"C:\\Ruby30-x64\\bin\\rubyw.exe"  
00000000649BA1C7 41:54 push r12 r12:L"rubyw C:/Users/User/Desktop/popen\_client.rb \"Other Args\""  
00000000649BA1C9 57 push rdi   
00000000649BA1CA 56 push rsi   
00000000649BA1CB 53 push rbx   
00000000649BA1CC 48:81EC 08010000 sub rsp,108   
00000000649BA1D3 48:8DAC24 80000000 lea rbp,qword ptr ss:[rsp+80]   
00000000649BA1DB 48:85D2 test rdx,rdx rdx:L"C:\\Ruby30-x64\\bin\\rubyw.exe"  
00000000649BA1DE 49:89CC mov r12,rcx r12:L"rubyw C:/Users/User/Desktop/popen\_client.rb \"Other Args\""  
00000000649BA1E1 49:89D5 mov r13,rd r13:L"C:\\Ruby30-x64\\bin\\rubyw.exe", rdx:L"C:\\Ruby30-x64\\bin\\rubyw.exe"  
00000000649BA1E4 45:89C6 mov r14d,r8d   
00000000649BA1E7 0F84 B8010000 je x64-msvcrt-ruby220.649BA3A5   
00000000649BA1ED 48:8D1D 0C130D00 lea rbx,qword ptr ds:[64A8B500]   
00000000649BA1F4 48:8D05 05230D00 lea rax,qword ptr ds:[64A8C500]   
00000000649BA1FB EB 10 jmp x64-msvcrt-ruby220.649BA20D   
00000000649BA1FD 0F1F00 nop dword ptr ds:[rax],eax   
00000000649BA200 48:83C3 10 add rbx,10   
00000000649BA204 48:39C3 cmp rbx,rax   
00000000649BA207 0F83 64010000 jae x64-msvcrt-ruby220.649BA371   
00000000649BA20D 48:837B 08 00 cmp qword ptr ds:[rbx+8],0   
00000000649BA212 75 EC jne x64-msvcrt-ruby220.649BA200   
00000000649BA214 48:8D75 10 lea rsi,qword ptr ss:[rbp+10]   
00000000649BA218 31C0 xor eax,eax   
00000000649BA21A B9 0D000000 mov ecx,D D:'\r'  
00000000649BA21F 48:C743 08 FFFFFFFF mov qword ptr ds:[rbx+8],FFFFFFFFFFFFFFFF   
00000000649BA227 48:C703 00000000 mov qword ptr ds:[rbx],0   
00000000649BA22E 41:83CE 20 or r14d,20   
00000000649BA232 48:89F7 mov rdi,rsi   
00000000649BA235 C745 F0 18000000 mov dword ptr ss:[rbp-10],18   
00000000649BA23C 48:C745 F8 00000000 mov qword ptr ss:[rbp-8],0   
00000000649BA244 F348:AB rep stosq   
00000000649BA247 C745 00 01000000 mov dword ptr ss:[rbp],1   
00000000649BA24E 48:8B3D 1F3B0E00 mov rdi,qword ptr ds:[<&GetStdHandle>]   
00000000649BA255 B9 F6FFFFFF mov ecx,FFFFFFF6 #STDIN   
00000000649BA25A 48:C745 D0 00000000 mov qword ptr ss:[rbp-30],0   
00000000649BA262 48:C745 D8 00000000 mov qword ptr ss:[rbp-28],0   
00000000649BA26A 4C:8D7D D0 lea r15,qword ptr ss:[rbp-30]   
00000000649BA26E 48:C745 E0 00000000 mov qword ptr ss:[rbp-20],0   
00000000649BA276 C745 10 68000000 mov dword ptr ss:[rbp+10],68 68:'h'  
00000000649BA27D C745 4C 00010000 mov dword ptr ss:[rbp+4C],100   
00000000649BA284 FFD7 call rdi   
00000000649BA286 B9 F5FFFFFF mov ecx,FFFFFFF5 # STDOUT   
00000000649BA28B 48:8945 60 mov qword ptr ss:[rbp+60],rax   
00000000649BA28F FFD7 call rdi   
00000000649BA291 B9 F4FFFFFF mov ecx,FFFFFFF4 #STDERR   
00000000649BA296 48:8945 68 mov qword ptr ss:[rbp+68],rax   
00000000649BA29A FFD7 call rdi   
00000000649BA29C 4C:89E1 mov rcx,r12 r12:L"rubyw C:/Users/User/Desktop/popen\_client.rb \"Other Args\""  
00000000649BA29F 48:8945 70 mov qword ptr ss:[rbp+70],rax   
00000000649BA2A3 FF15 EB3C0E00 call qword ptr ds:[<&lstrlenW>]   
00000000649BA2A9 3D FF7F0000 cmp eax,7FFF   
  
<kernelbase.GetStdHandle>   
00007FF9E6FABC90 40:53 push rbx   
00007FF9E6FABC92 48:83EC 20 sub rsp,20   
00007FF9E6FABC96 83F9 F5 cmp ecx,FFFFFFF5   
00007FF9E6FABC99 75 3E jne kernelbase.7FF9E6FABCD9   
00007FF9E6FABC9B 6548:8B0425 60000000 mov rax,qword ptr gs:[60]   
00007FF9E6FABCA4 48:8B48 20 mov rcx,qword ptr ds:[rax+20]   
00007FF9E6FABCA8 F781 A4000000 0004000 test dword ptr ds:[rcx+A4],400   
00007FF9E6FABCB2 75 77 jne kernelbase.7FF9E6FABD2B   
00007FF9E6FABCB4 6548:8B0425 60000000 mov rax,qword ptr gs:[60]   
00007FF9E6FABCBD 48:8B48 20 mov rcx,qword ptr ds:[rax+20]   
00007FF9E6FABCC1 48:8B59 28 mov rbx,qword ptr ds:[rcx+28]   
00007FF9E6FABCC5 48:83FB FF cmp rbx,FFFFFFFFFFFFFFFF   
00007FF9E6FABCC9 0F84 37680400 je kernelbase.7FF9E6FF2506   
00007FF9E6FABCCF 48:8BC3 mov rax,rbx   
00007FF9E6FABCD2 48:83C4 20 add rsp,20   
00007FF9E6FABCD6 5B pop rbx   
00007FF9E6FABCD7 C3 ret   
00007FF9E6FABCD8 CC int3   
00007FF9E6FABCD9 83F9 F4 cmp ecx,FFFFFFF4   
00007FF9E6FABCDC 75 13 jne kernelbase.7FF9E6FABCF1   
00007FF9E6FABCDE 6548:8B0425 60000000 mov rax,qword ptr gs:[60]  
  
qword ptr gs:[0000002D4577E060]=0000002D4577D000  
Address=0000002D4577D000 Size=000000000000B000 Page Information=PEB Allocation Type=PRV Current Protection=-RW-- Allocation Protection=-RW--  
<https://en.wikipedia.org/wiki/Process_Environment_Block>  
  
  
00007FF9E6FABCE7 48:8B48 20 mov rcx,qword ptr ds:[rax+20]   
   
00007FF9E6FABCEB 48:8B59 30 mov rbx,qword ptr ds:[rcx+30]   
00007FF9E6FABCEF EB D4 jmp kernelbase.7FF9E6FABCC5   
00007FF9E6FABCF1 83F9 F6 cmp ecx,FFFFFFF6   
00007FF9E6FABCF4 75 2C jne kernelbase.7FF9E6FABD22   
00007FF9E6FABCF6 6548:8B0425 60000000 mov rax,qword ptr gs:[60]   
00007FF9E6FABCFF 48:8B48 20 mov rcx,qword ptr ds:[rax+20]   
00007FF9E6FABD03 F781 A4000000 0002000 test dword ptr ds:[rcx+A4],200   
00007FF9E6FABD0D 75 1C jne kernelbase.7FF9E6FABD2B   
00007FF9E6FABD0F 6548:8B0425 60000000 mov rax,qword ptr gs:[60]   
00007FF9E6FABD18 48:8B48 20 mov rcx,qword ptr ds:[rax+20]   
00007FF9E6FABD1C 48:8B59 20 mov rbx,qword ptr ds:[rcx+20]   
00007FF9E6FABD20 EB A3 jmp kernelbase.7FF9E6FABCC5   
00007FF9E6FABD22 48:83CB FF or rbx,FFFFFFFFFFFFFFFF   
00007FF9E6FABD26 E9 DB670400 jmp kernelbase.7FF9E6FF2506   
00007FF9E6FABD2B 33DB xor ebx,ebx   
00007FF9E6FABD2D EB A0 jmp kernelbase.7FF9E6FABCCF

**Ruby get process parent pid**  
rb\_pid\_t  
rb\_w32\_getppid(void)  
{  
 typedef long (WINAPI query\_func)(HANDLE, int, void \*, ULONG, ULONG \*);  
 static query\_func \*pNtQueryInformationProcess = NULL;  
 rb\_pid\_t ppid = 0;  
  
 if (rb\_w32\_osver() >= 5) {  
 if (!pNtQueryInformationProcess)  
 pNtQueryInformationProcess = (query\_func \*)get\_proc\_address("ntdll.dll", "NtQueryInformationProcess", NULL);  
 if (pNtQueryInformationProcess) {  
 struct {  
 long ExitStatus;  
 void\* PebBaseAddress;  
 uintptr\_t AffinityMask;  
 uintptr\_t BasePriority;  
 uintptr\_t UniqueProcessId;  
 uintptr\_t ParentProcessId;  
 } pbi;  
 ULONG len;  
 long ret = pNtQueryInformationProcess(GetCurrentProcess(), 0, &pbi, sizeof(pbi), &len);  
 if (!ret) {  
 ppid = pbi.ParentProcessId;  
 }  
 }  
 }  
  
 return ppid;  
}

**Edit the Sketchup.exe Process Environment Header**

import sys

import struct

if len(sys.argv) < 4:

print "Change Exe Run Mode Application by burlachenkok@gmail.com\nNot sufficient parametrs. 'exe\_src\_name.exe' 'exe\_dest\_name.exe' 'to\_console' or 'to\_windows'"

sys.exit(-1)

source = open(sys.argv[1], "rb")

dest = open(sys.argv[2], "w+b")

dest.write(source.read())

dest.seek(0x3c)

(PeHeaderOffset,)=struct.unpack("H", dest.read(2))

dest.seek(PeHeaderOffset)

(PeSignature,)=struct.unpack("I", dest.read(4))

if PeSignature != 0x4550:

print "Error in Find PE header"

dest.seek(PeHeaderOffset + 0x5C)

if sys.argv[3].strip() == "to\_console":

# console mode

dest.write(struct.pack("H", 0x03))

elif sys.argv[3].strip() == "to\_windows":

# window mode

dest.write(struct.pack("H", 0x02))

else:

print "Wrong Format: '" + sys.argv[3] + "'"

source.close()

dest.close()

print "Completed succesfully.."