IDA Pro v5.5 feature list

Naturally, after adding Windbg support in v5.4, we had to add support for crash dumps. Just specify a crash dump file as the input file and IDA will create a database from it. The debugger can be 'launched' to enter the familiar debugger environment with module, thread, and stack windows, where information can be retrieved the usual way. The only thing you can not do is to resume the execution, that would be an overkill ;)

The previous addition required a big, hopefully invisible, change in the kernel, because the existing storage method could not handle huge gigabyte segments. Previous versions of IDA had a hard limit on the addressing space of the program: max 256MBytes in the default configuration. Modern programs routinely allocate much bigger memories, so we had to find a solution. Now, if a crash dump segment is bigger than a certain size, IDA automatically chooses the sparse storage method. Instead of storing information about every single byte of the program, IDA remebers only useful information. Thus, a 25MB uninitialized array requires just a few bytes of storage to describe it, not 100MB as before.

We would also explicitly mention and say 'thank you' to the users who contributed to this release of IDA Pro. Bernhard Mueller from SEC Consult GmbH was very kind to investigate why the Symbian debugger was failing on new devices and contributed an improvement. Robert Krkic generously shared his IDS files for Symbian systems with all IDA Pro users. Thank you guys, your contributions make IDA better and easier to use!

Finally! Finally we drop the MDI user interface and switch to dockable windows. They are simpler to use, more flexible, waste less screen space. Well, you know it yourself. The new interface also includes the improved hex viewer and stack view. The new hex view is much easier to use, can display the data in various formats, allows editing in-place.

Probably the fastest thing to do is to visit the **comparison page** for more detail about processor modules. In the endless pursuit to improve the disassembly output, we continue to add new methods, tricks, and heuristic rules to IDA. This time the biggest changes are in the ARM and PC modules. The ARM module handles the stack frame, type information, call/jump instructions better than before.

The PC module knows about more code patterns, like switch and position-independent code idioms, detects more exception handlers, etc

Do you remember that you can add emulated API functions to the Bochs debugger? Just provide an implementation in IDC/Python/C++ and your function is called. For example, you could provide an implementation of the socket() function that would open a socket on the host system or do something else. While this possibility is very useful and remains in place, we added more predefined functions. Now the Visual Studio and Borland C/C++ startup code can be executed without generating exceptions and you can focus on the 'real' code.

In addition, we also added the 'Bochs rc file loader'. It really helps if you already have a bochsrc file and want to debug it with IDA. Just specify it as the input file and IDA will create a nice database for you. No need to create a dummy database, populate it with the code from the boot sector, etc.

Other, probably less visible, improvements include the PDB plugin, the type system, more SDK functions (check out the exec_request_t if you develop multithreaded plugins), etc. The full list is below:

PROCESSOR MODULES

+ PC: added detection of CException destructor; this helps to detect exception handlers and ignore them during function epilog analysis
+ PC: added support for another variation of PIC code by GCC
+ PC: more switches recognized in unoptimized MSVC code
+ ARM: added support for SUB Rx, R11, #fpoff stack variable references
+ ARM: added support for switches implemented using TBB/TBH instruction
+ ARM: better detection of **R7-based** frames
+ ARM: LDMED can be used for return too
+ ARM: type info and argument names are propagated for local variables passed by reference
+ ARM: other unspecified improvements (we removed them to keep the list short and more readable)
+ ARM: LSL Rx, Ry, #0 and ADD Rx, Ry, #0 are simplified to MOV Rx, Ry
+ I51: i/o port names are accepted for all segments (before only FSR definitions were handled)
+ MIPS: **much improved analysis** of ELF files
+ SuperH: simplified display of pc-relative literal loads

FILE FORMATS

+ BOCHRC file loader: it is now possible to start IDA with a **bochsrc** file as the input file
+ CRASH DMP file loader: it is now possible to start IDA with an MS Windows Crask dump file
+ COFF: segment permissions are imported for MS object files
+ ELF: accept PPC64 ELF files
+ ELF: handle dynamic symbols in MIPS files
+ ELF: some new SuperH relocations are supported
+ ELF: added support for ARM TLS relocations
+ EPOC: added support for multiple imports with the same ordinal
+ EPOC: user contribution: ids files for epoc6/9 from Robert Krkic
+ EPOC: since AppTRK does not report thread creation/deletion, IDA forcibly refreshes the thread list if an unknown thread id is encountered
+ PDB: added the possibility to manually load a specific PDB file; to load only types from the PDB
+ PDB: added support for anonymous unions. types with bitfields are handled more correctly: we replace them with a corresponding POD type; ida kernel can not handle bitfields yet
+ PDB: added support for undefined enum types
+ PDB: better handling of C++ static methods and functions returning complex types.
+ PDB: better handling of string literals (??_C@...)
+ PDB: information about function argument names is applied, if available in the .pdb file
+ PDB: symbols can be loaded using EXE headers in the database, either from the module list during debugging or via File menu by specifying a valid base address
+ PE: added an option in pe.cfg to force loading of all PE file sections (usually .reloc and similar sections are skipped)
+ PE: if import and/or export tables lie outside .idata segment, they are parsed and formatted

KERNEL

+ Added some common C++ ABI functions to noret.cfg
+ Added support for **__usercall** functions with variable number of arguments (...)
+ Added functions to handle floating point instructions for the decompiler
+ Improved the browsing speed for big databases when autoanalysis is busy: moving around in huge databases is much better
+ Demangler: added support for the latest **gcc4**

+ Updated WinCE ARM ids files to Windows Mobile 6.0

IDC & SDK

+ IDC: added MoveSegm() and RebaseProgram() functions
+ IDC: added OpFloat() function
+ IDC: renamed segment modification functions to start with a verb. Old names continue to be available.
+ SDK: added append_buf() and unpack_buf(), append_obj() and unpack_obj()
+ SDK: added build_anon_type_name()
+ SDK: added build_func_type() to facilitate building of type strings that represent functions
+ SDK: added callbacks to AskUsingForm so that the dialog can be modified on the fly
+ SDK: added change_storage_type() to change the storage method of arbitrary address range. please note that the sparse storage method works well only with uninitialized areas with huge objects
+ SDK: added floating point conversion functions for 64bit values
+ SDK: added functions for working with imports (enum_import_names and others, see nalt.hpp)
+ SDK: added get_zero_areas() to retrieve info about huge zero inited ranges
+ SDK: added **exec_request_t** - class that allows for code execution in the main thread from any other thread
+ SDK: added more convenient areacb_t::for_all_areas2(), which accepts a functor instead of (function,data) pair
+ SDK: added qlist compiler-independent template
+ SDK: added qthread_self()
+ SDK: added set_process_options() to set debugger process options
+ SDK: added ui_enable_input_hotkeys notification to let plugins temporary disable alphanumeric hotkeys which can interfere with user input
+ SDK: added replace_wait_box()
+ SDK: added DBGINV_REDRAW bit to refresh the user interface while invalidating the debugger caches
+ SDK: bitfield definitions in type strings have been changed. since there was no support for bitfields in the parser, this change should not affect anybody
+ SDK: introduced dt_ldbl to represent long double types. long double is different from tbyte and can be 8,10,12, or 16 bytes depending on the compiler
+ SDK: is_sp_based() can now return information about operands which are substracted from stack pointer
+ SDK: patch_byte() and similar functions return success if they succeed in modifying either process memory or idb. before they were returning true only if both process memory and idb were modified
+ SDK: windbg kernel mode: added notion of **virtual threads**

USER INTERFACE

+ UI: it is possible to add several types at once from the local types list (previously only the first one was added)
+ UI: it is possible to delete segment register change points from the "Jump to segment register" dialog or "Segment registers" view
+ UI: it is possible to set the base for array indexes display (decimal, hexadecimal, octal or binary)
+ UI: hex view supports in-place editing, various data formats and unicode strings
+ UI: unicode strings: unprintable characters are grouped into arrays; extra trailing zeroes are omitted by default

DEBUGGER

+ breakpoint condition is evaluated only after the breakpoint is hit

+ Symbian: added support for new AppTRK v3.0.8 (thanks to Bernhard Mueller from SEC Consult GmbH)

BUGFIXES

BUGFIX: 'move segment' command could affect segments outside of selection
BUGFIX: 'search for immediate' would ignore unexplored dword values (except in some very rare cases)
BUGFIX: "load debug info" command was not available from the module list if the database was created by instant debugger
BUGFIX: 68K: immediate floating-point operands were displayed incorrectly
BUGFIX: 8051: 'x' key didn't work with named bit operands (like P35)
BUGFIX: apply_tinfo() could not apply array types correctly; also applying char[] types was not always producing nice results
BUGFIX: ARM: some Thumb2 instructions were not decoded
BUGFIX: attaching to a process that generates exceptions or any other non-white listed event would still generate an ATTACH event first
BUGFIX: autoanalysis could loop indefinitely in some rare cases
BUGFIX: bochs could hang while parsing a binary file as bochsrc text file
BUGFIX: bochs dbg/rc file selectors were using save dialog and not open dialog
BUGFIX: bochs debugger could not handle some malformed PE files
BUGFIX: bochs debugger PE files with no sections were handled incorrectly
BUGFIX: bochs debugger was not initializing the FPU before running the program
BUGFIX: bochs debugger: CR4.bit9 should be set otherwise an invalid opcode will be thrown if a program attempts to use any of SSE{n} instructions.
BUGFIX: CodeView debug information was applied to wrong addresses
BUGFIX: corrupted epoc executables could not be loaded
BUGFIX: deleting a segment from UI sometimes led to deletion of wrong segment
BUGFIX: demangler: Visual Age was always interpreted as GNU compiler
BUGFIX: ELF: if file had .got.plt section but no .got, IDA did not rename PLT import stubs
BUGFIX: equal_types() was considering floating point and integral types of the same size equal
BUGFIX: esp alignment in 32-bit programs is only 2 bytes, not 4 bytes (push ax/add esp, 2 are perfectly valid but ida was aligning esp to 4 bytes)
BUGFIX: fild/fistp qword operands are marked as floating point doubles
BUGFIX: fpval->integer conversion functions were broken since very long time
BUGFIX: generating html file with inverted colors would hang ida
BUGFIX: GetProcessQty() was failing if called before running the debugger
BUGFIX: huge (>2GB) areas could not be displayed in the navigation band in all zoom levels
BUGFIX: ida could complain about patched bytes at the start of a debugging session while no bytes had been patched
BUGFIX: ida was always asking for a confirmation of array creation if the first byte of the array was unexplored
BUGFIX: ida was not using export information from available DLLs due to a logic error in the code
BUGFIX: ida was silently failing to use dummy prefixes as new location names; now it displays a correct error message
BUGFIX: ida would crash in linux trying to display the debugger specific options dialog box
BUGFIX: IDACall GetProcAddress() was returning wrong addresses for forwarded entries
BUGFIX: idb2scr() was broken. it was impossible to edit comments with 8-bit ascii characters, the dialog box would display them as garbage
BUGFIX: IDC was not comparing floating point with integers correctly
BUGFIX: idc: fixed a typo in INF_SIZEOF_LDBL
BUGFIX: if attaching to multithreaded application, the thread window would not focus on the current thread (no thread would be selected)
BUGFIX: in some cases IDA was still showing the "IDA Pro failed to stop/detach from the debugged program" dialog even if the process was already terminated

BUGFIX: in some rare cases IDA could display an exception if "full stack pointer analysis" was enabled.
BUGFIX: it was impossible to refuse to import types from local type window to enum/structure windows
BUGFIX: it was impossible to debug a dll when two or more dlls of the same name are loaded
BUGFIX: it was impossible to suspend a process that keeps on generating masked exceptions.
BUGFIX: java module could crash trying to display an automatic comment for a long (multi-line) output
BUGFIX: linux debugger could interr if the input file name was incorrect in the process options
BUGFIX: long descriptions of local void and typedef types were lacking the 'typedef' keyword
BUGFIX: LX: some LE files were not completely loaded (if the object's virtual size was less than the actual size)
BUGFIX: mac executables could not be used under Mac OS X 10.3
BUGFIX: mac os x debugger could exit with "can’t find dll name" error
BUGFIX: mips dmfc1/dmtc1 instructions were not supported
BUGFIX: mips module was requiring even fp register numbers for ldc1/sdc1 instructions; removed this limitation
BUGFIX: mips: clz/clo/plzcw/movf/movt instructions could not be disassembled in delay slots
BUGFIX: move_segm() would rebase offset expressions regardless of the MSF_NOFIX flag
BUGFIX: multiple problems with the linux debugger module have been fixed
BUGFIX: nlm loader was not populating the imports window
BUGFIX: noType() called on a tail byte could corrupt the database
BUGFIX: pausing execution in VMWare was reported as an exception
BUGFIX: PC: (x64) movq instruction was decoded as movd
BUGFIX: PC: fixed occasional inifinite loop during analysis
BUGFIX: PC: fpu instructions had wrong values in the dtyp field (fpu reg types are still set to be dt_double, ideally it should be dt_tbyte)
BUGFIX: PC: functions with _SEH_prolog were not analyzed properly when using PDB symbols
BUGFIX: PC: LAHF/SAHF instructions could not be disassembled in 64-bit mode
BUGFIX: PC: operands of the fcompp insn were displayed in reverse order; they are not displayed anymore if they are regular st, st(1)
BUGFIX: PC: some one-byte opcodes were decoded incorrectly in 64-bit mode. REX.B was not ignored for opcodes 05, 25 etc
BUGFIX: PDB plugin could create a circular dependency in type, which would later crash IDA
BUGFIX: PE: imports in files with zero sections were parsed incorrectly
BUGFIX: PE: several fixes to handle unusual alignment values (thanks to Ivan Teblin)
BUGFIX: PIC files were loaded with 8-bit 'bytes' by default
BUGFIX: pressing and holding F8 in bochs debugger could lead to a racing condition and deadlock
BUGFIX: SDK: c2ustr was returning a string with an extra terminating zero which was included in the string length
BUGFIX: SDK: for 16bit segmented memory, ida would set incorrect value for eip after a breakpoint (visible with third-party 16bit debugger plugins)
BUGFIX: SDK: open_hexdump_window() didn't work in GUI version
BUGFIX: SDK: patch_word()/patch_long() did not work properly if the new value had some bytes matching the database
BUGFIX: set_offset_ex() with RVAOFF could fail randomly
BUGFIX: some symbian sisx files could not be loaded (ida crashed)
BUGFIX: some undefined types referenced by name were imported as 'void'
BUGFIX: structure alignment was sometimes wrong in types created by the pdb plugin
BUGFIX: structures with arrays of structures should not be displayed in terse form
BUGFIX: Symbian: file names in embedded sis files were displayed incorrectly
BUGFIX: the horizontal scrollbar in disassembly views was handled incorrectly
BUGFIX: tms6 module was incorrectly commenting branch/call targets under linux
BUGFIX: type system was not adding comments for arguments that were handled from generic code (gen_use_arg_types())

BUGFIX: u2cstr() was returning a string with an extra terminating zero which was included in the string length
BUGFIX: UI: ida could display "Access denied" message when running in a remote desktop session and the session was minimized or closed
BUGFIX: unicode support in the mac version of ida was broken
BUGFIX: unix implementation of qsem() functions could crash on NULL pointer
BUGFIX: unpatched PIC GOT32/GOTOFF relocations were handled incorrectly
BUGFIX: validate_tofs() could not handle variable sized structures. modifying a byte of a variable sized structure could lead to interrs.
BUGFIX: win32 debugger would freeze trying to attach to a crippled program that would generate exceptions during attaching
BUGFIX: win7 rc has a hardcoded int3 instruction in LdrpDoDebuggerBreak() instead of a call to DbgBreakPoint() and IDA stops on it
BUGFIX: wince debugger could not attach to an already running process if the "debugger->process options->input file" was specified
BUGFIX: WinCE debugger module could hang in some cases if ida_kdstub.dll was missing from IDA directory and the PocketPC
BUGFIX: wince debugger was causing system wide memory leaks on the PDA
BUGFIX: windbg debugger module (kernel mode) generates duplicate debug names each single step
BUGFIX: windbg debugger module was not enabled for drivers
BUGFIX: windbg was reverting the debugging tools path to the default at each process start
BUGFIX: wrong types with circular dependencies could crash ida (like typedef x x;)