IDA Pro 5.4 feature list

HIGHLIGHTS

- **Three new debuggers**

We continue to add more debugger modules to IDA and improve the existing ones. This version introduces three new debuggers: bochs, gdb, and windbg.

Each of these debuggers deserve a separate article, so only the most interesting highlights here:

- **Bochs**: can run any 32bit code, from a few instructions to whole operating systems. Just click F9 and immediately switch to running state. In fact, any code snippet can be executed in a safe and user friendly way. With the bochs debugger, we offer three different worlds: **run-any-code-snippet** facility, **windows-like-environment** for PE files, and **any-bochs-image** bare-bone machine emulation mode. Please read more about Bochs plugin here: http://hexblog.com/2008/11/bochs_plugin_goes_alpha.html
- **GDBServer**: **x86** and **arm** targets are supported. Among other things, it is possible to connect IDA to **QEMU** or debug a virtual machine using **VMWare**.
- **Windbg**: both **user** and **kernel** mode debugging are available. IDA can automatically load required PDB files and populate the listing with meaningful names, types, etc. Speaking of PDB files, IDA imports more information from them: local function variables and types are retrieved too, c++ base classes are handled, etc.

The **GDBServer** and **Windbg** debugger modules support **local** and **remote** debugging. We tried to make the debugger modules as open as possible: target-specific commands can be sent to all backend engines very easyly.

- **Better analysis**

IDA understands GNU-style function arguments (when they are moved onto the stack instead of being pushed). Analysis is more efficient in general and does does fall into the instruction creation/deletion loop. Many FLAIR signatures have been updated and new ones added.

- **Command line**

The command line was present in IDA since very long time but it was not turned on by default. This version activates it and empowers it with the following interpreters:

IDC	native IDA language (we added support for global variables)
Bochs	to send arbitrary commands to the Bochs debugger
GDBServer	to control GDBServer targets

Windbg to access Windbg extensions and kernel information

Python to progr[1]am IDA in the popular language

The new IDA supports Python out of box, thanks to Gergely Erdelyi, who kindly agreed the Python plugin to be included in the official distribution.

PROCESSOR MODULES

+ PC: added new instructions (**AMD SSE4a, Geode LX, XSAVE/XRSTOR, Intel SMX, AMD-V, Intel AES** and some others)
+ PC: added support for **GNU C++ style of preparing function arguments** (moving to stack instead of pushing)
+ PC: added support for **Intel's SSE4.1** and **SSE4.2** instructions
+ PC: added support for **non-Microsoft (Linux/OSX/etc) AMD64 calling convention**
+ PC: **EH_epilog** function is recognized and taken into account in the analysis
+ PC: improved handling of linux syscalls: check os/abi field of elf files to distinguish linux files from other openbsd files
+ PC: slightly better function prolog analysis
+ PC: "xmmword" is used for 16-byte operands instead of "oword"
+ ALPHA: better analyzis of Windows NT PE files
+ ARM: added a processor option to disable detection of BL instructions used for long jumps in Thumb code.
+ ARM: added **UND** pseudo-instruction for the permanently undefined opcode ranges
+ ARM: improved calculation of the size of some jump tables
+ ARM: user can specify whether a Thumb BL instruction is a call or a jump (Edit/Other/Force BL...)
+ C166: bit references to data items of enum type use symbolic constants for bit numbers (ida displays myword.mybit instead of myword.5)
+ IA64: improved analysis
+ PPC: suppport for GCC jump table switch
+ TMS320C6x: added support for **64xx** and **67xx** instructions

FILE FORMATS

+ COFF: arm/thumb switcher symbols ($CODE16/$CODE32) are recognized and properly handled
+ ELF: added support for a few new HP 64bit relocation types; 64-bit hppa processing is slightly better
+ ELF: added support for some IA64 relocations
+ PDB: added import of c++ base classes and unnamed-tag types; some udt types were imported incorrectly
+ PDB: **additional PDB files can be loaded** for DLL files. For that, the "File/Load PDB" command or the popup menu of the Debugger Module Window can be used
+ PDB: **local variable names and types**, static data names are imported into the database
+ PDB: public names starting with __imp__ are converted to dwords
+ PE: IDA does not load discardable segments anymore
+ PE: IDA recognizes the DriverEntry() function prototype
+ PE: IDA parses .pdata segment
+ PE: added support for PE files with 0 sections

KERNEL

+ signatures: **MS SDK and Visual Studio** signatures have been updated
+ signatures: **ICL v11.0.066** has been added
+ signatures: added **BDS2008** signatures
+ "unload database to idc" exports all segment register change points, not only user defined ones
+ __thiscall functions without any arguments are forbidden; while they do not make much sense, ida permits such declarations
+ alignment pragmas are printed as part of structure declarations (before they were printed as comments)
+ grouped all **MS Windows window messages** in til files into one big enumeration
+ FLAIR: added support for TMS470 COFF files (used by ARM compilers)
+ idc scripts can be executed from the startup signatures. IDA defines some helper functions for this context. Hopefully this feature will allow us to stop using the 'main hints' that are used by startup signatures and switch to nice IDC scripts

IDC & SDK

+ SDK: ida generates **pre-action** events before modifying the database (please note not all modification have corresponding events because any plugin may modify the database on a very low level)
+ SDK: introduced **command line interpreters**. any plugin may introduce a CLI and the user may switch between them on the fly
+ SDK: added manual memory regions for the debugger module that can not report the memory layout. the user can specify the desired memory layout on the fly
+ SDK: add_chooser_command() is supported in the text version
+ SDK: added callbacks to modify graphs displayed by ida and to display graphs without functions; sample plugins ugraph2/3 illustrate how to use the new functions
+ SDK: added check_bpt() to check the state of a breakpoint
+ SDK: added convenience functions to pack data into bytevec_t
+ SDK: added create_disasm_graph() function
+ SDK: added create_generic_linput() to create inputs from any source
+ SDK: added DBG_FLAG_SMALLBLKS for debugger modules that usually work on slow connections
+ SDK: added debugger_t::set_dbg_options() for debugger specific options
+ SDK: added functions to access IDS files
+ SDK: added functions to work with intervals
+ SDK: added get_dbg_byte() to read data from the debugged process memory
+ SDK: added get_debmod_extensions() to debugger module interface; it can be used by debugger modules to publish additional functionality
+ SDK: added get_nsec_stamp() to get high precision time stamps
+ SDK: added inf.database_change_count. this field is incremented each time a byte is patched or regular segment information is changed (essentially it tracks 'real' program modifications)
+ SDK: added segment base and bitness information to memory_info_t and changed the prototype of get_memory_info in debugger_t; this is an incompatible change, the source code of existing debugger modules must be changed; however, existing debugger modules can be used without recompilation, the kernel will use the correct interface depending on the debugger api version number
+ SDK: added set_process_state() to manually modify the process state from a plugin
+ SDK: added extlang.fileext and convenience functions to work with extlang
+ SDK: colons can be used in form input field labels by escaping them with backslashes.
+ SDK: forms: new field types: F for folder names and f for file names
+ SDK: get_db_byte() to read byte from database / patch_db_byte() to write byte to process memory only

+ SDK: is_valid_typename() to check type names. IDA permits characters encountered in c++ template names in type names.
+ SDK: new flag for debugger modules: DBG_FLAG_DONT_DISTURB. Debugger modules with this flag can not carry out any actions once the application is let to run. They can only wait for the next event or suspend the application.
+ SDK: renamed ua_ana0 -> decode_insn(); ua_code -> create_insn()
+ SDK: Run() function can be used to execute not only compiled functions but also built-in functions and functions defined by plugins
+ SDK: added DBG_FLAG_CLEAN_EXIT to the debugger description: it forces IDA to remove breakpoints before terminating the application. This flag is useful for the platforms where processes share memory
+ IDC: added support for **global variables**. they are declared like this: extern var;
+ IDC: added CheckBpt() to check the state of a breakpoint
+ IDC: added GetFchunkReferer() to enumerate parents of a function chunk
+ IDC: added SetArrayFormat() to specify exact representation for an array in the output listing
+ IDC: IdbByte() to read byte from database / PatchDbgByte() to write byte to process memory only
+ IDC: replaced SegReg() by SetRegEx()
+ IDC: added IDC functions for ARM: ArmForceBLJump(), ArmForceBLCall()

USER INTERFACE

+ gui: command line at the bottom of the main ida window is displayed by default. it can process commands for any registered command line interpreter
+ ui: 'G' hotkey can used to move to the desired offset within the current type in the structure and enum views
+ ui: a reference to a structure type in data items (like myvar mystruct <0>) can be used to rename/jump to the structure type
+ ui: display problematic type sizes in the local types window as "Error"
+ ui: it is possible to rename a structure field staying on a reference to a stack variable of a structure type. For example, [ebp+StartupInfo.dwFlags] can be used to rename "dwFlags"
+ ui: jumping to a structure type definition positions the cursor at the beginning of the definition
+ ui: text mode: added **TVHEADLESS** environment to disable all output
+ gui: added an option to disable hints when the debugger is active
+ gui: added CLOSED_BY_ESC config parameter to specify which windows can be closed by pressing Esc
+ gui: added convenience menu item to save bytes from hex view to a file
+ gui: message window supports copy/delete on single lines as well as saving output to a file
+ gui: added horizontal scrollbar to the log window

DEBUGGER

+ debugger: added **Bochs debugger** back-end. It can execute any code snippet with a single click
+ debugger: added **GDBServer debugger** back-end. ARM and x86 targets are supported.
+ debugger: added **Windbg debugger** back-end. Both user and kernel mode debugging are supported.
+ debugger: added support for segment **(16-bit) debugging** (the debugger module must provide the map_address() function; this function can map seg:off pairs to linear addresses)
+ debugger: added an option to autoload **PDB** files
+ debugger: added the notion of the default debugger, which is autoselected by ida for new databases

+ debugger: right-clicking on the process list refreshes it
+ debugger: stack reconstruction is turned off by default

BUGFIXES

BUGFIX: 'load desktop' was displaying wrong desktop list and could restore some desktops only partially; for example, it would not restore register views if the debugger was not active

BUGFIX: 'search for immediate' command could cause an internal error
BUGFIX: "search for immediate value" could return wrong addresses
BUGFIX: "unload file" command would not generate Patches() function but reference to it

BUGFIX: (arm) BLX Rx is a call, not a jump
BUGFIX: (PE) properly parse fixups of type HIGHADJ (improves analyzis of Alpha PE files)

BUGFIX: .net: pinned elements were not detected
BUGFIX: .net: sometimes the 'case' keyword was missing
BUGFIX: __usercall prototypes were impossible for processors that does not implement the processor_t::get_reg_name callback

BUGFIX: a local structure type could be referenced by name in the result of guess_tinfo(). this could lead to problems later, if the referenced structure was renamed

BUGFIX: a patched a byte in the middle of a data array would not be reflected in the listing until the array was recreated

BUGFIX: a structure member, which is a pointer to a function with some of the argument names specified and some not, would be incorrectly converted into a type string

BUGFIX: a.out: debugging stabs were used as symbol values and errorneous symbols hampered the analysis

BUGFIX: adding a software breakpoints at address 0 would make the whole listing red

BUGFIX: ADSP processor module could not decode references to dmovlay/pmovlay registers

BUGFIX: arm debugger was incorrectly handling the 'step over' command for some BX/BL instructions (it was assuming that they always return to the next instruction)

BUGFIX: arm module could not create some macroinstructions and would leave the code undefined

BUGFIX: arm: some undefined instructions were improperly decoded
BUGFIX: arm: Thumb-2 LDR instructions with long offsets were disassembled incorrectly

BUGFIX: automatic comments were displayed as garbage by generate_disasm_line()

BUGFIX: b2a32() was printing binary numbers without leading zeroes
BUGFIX: C preprocessor could not handle token gluing if the first glued token was a number

BUGFIX: changing the type of a structure member would not lead to reanalysis (required for the creation/deletion of xrefs from offset members)
BUGFIX: cli: ida64 was incorrectly displaying 64bit immediate constants in instructions

BUGFIX: command line arguments were passed incorrectly to the debugger application under WinCE

BUGFIX: debugger modules for the arm processor could miscalculate the target address of jump instructions and lose control of debugged application

BUGFIX: debugger modules for the arm processor improperly handled stepping over a return with Thumb<->ARM mode switch e.g. stepping at LDMFD SP!, {R7,PC} could put breakpoint in wrong place if the popped PC had low bit set.

BUGFIX: deleting a structure type that was referenced from the disassembly could lead to division by zero

BUGFIX: edit segment dialog box: if new segment boundaries were not overlapping with the old segment boundaries and the new segment addresses were higher, ida would crash

BUGFIX: enum width was incorrect in the "edit enum" dialog box
BUGFIX: epoc debugger could not handle breakpoints correctly if we attach to a process and do not suspend it at least once

BUGFIX: function prolog analysis had a logical bug (affects results very rarely)

BUGFIX: gui: the current identifier was not always highlighted if the listing was scrolled to the right

BUGFIX: handling of elf ppc relocation record R_PPC_EMB_SDA21 was not always correct (it seems to be interpreted differently in different files?!)

BUGFIX: hex-view could stop reacting to navigation hotkeys after a while
BUGFIX: hppa: ida was trying to continue to decode instructions after some conditional instructions with 'always' as the condition (movib, cmpib, ...)

BUGFIX: IDA could corrupt its state file (in ~/.idapro) under linux/mac if multiple instances were launched simultaneously

BUGFIX: IDA could crash on some .net files (because of too long user-defined strings)

BUGFIX: IDA could hang trying to load empty files for remote debugging
BUGFIX: IDC: substr() function with wrong parameters could crash
BUGFIX: identifier highlight would be unset after dragging a graph the second time

BUGFIX: if the debugger that was selected in a previous session was not available anymore, the debugger menu would be absent and the user could not switch to another debugger module

BUGFIX: if the single step exception was masked from the application, stepping over an instruction that itself would generate a single step exception would let the application run freely

BUGFIX: in some very rare cases the same very long name could be used for multiple locations (btree search failure)
BUGFIX: in txt-ui when input filename exceed visible limit and filename does not contain any path's (e.g. library module) ida crash by null-pointer dereference

BUGFIX: interr could occur if a switch idiom without an input register was manually specified

BUGFIX: linux-tvision: buffer overflow when 2 unrecognized esc-sequence are received from the keyboard

BUGFIX: list windows were displayed incorrectly on dual monitor systems if the second monitor was on the left

BUGFIX: Mac OS X version of ida could not be run on older systems because of libiconv incompatibility

BUGFIX: Mach-O: don't skip loading of sections which lie ouside of segment's boundaries (apparently OSX loader accepts such files)

BUGFIX: modification of a structure member type was not generating idb_event::ti_changed event; it had to generate it

BUGFIX: nagivation band could not represent the memory correctly if the address space was bigger than 2GBs

BUGFIX: one line hints were truncated
BUGFIX: opcode bytes were not visible for tms320c6 listings
BUGFIX: PC: fixed decoding of movhpd and movlpd instructions (operand size modifiers were wrong)

BUGFIX: pc: function arguments in partial registers (like al/ah) were not properly handled at the call sites

BUGFIX: pc: IDA was considering "lock cmpxchg" as an insane instruction
BUGFIX: pc: some linux syscalls had wrong prototypes
BUGFIX: ppc: in some case analyzer could enter an infinite loop
BUGFIX: pro.h could not be compiled with visual studio c++ v6.0
BUGFIX: sdk: removed a reference to unexisting function named intseq_t::del()

BUGFIX: some equal type were considered incompatible
BUGFIX: some IA64 auto comments were wrong
BUGFIX: some mach-o files could lead to internal error
BUGFIX: sparc relocations were not parsed in a.out files under MS Windows
BUGFIX: the debugger was not refreshing segmentation information properly after system calls like VirtualAlloc

BUGFIX: the error message about database open errors was incorrect
BUGFIX: the last decoded instruction was not always refreshed after suspending the debugged process

BUGFIX: TMS320C6x coff object files were loaded with insufficient alignment (must be at least 32 bytes to ensure correct execution packet boundaries)

BUGFIX: tricore memory addressing modes with displacement could be displayed incorrectly

BUGFIX: txt: deleting a menu item that was the last selected one would lead to a crash when the user tried to open the menu once more

BUGFIX: txt: warning() and info() dialog boxes could truncate the message by making the dialog box too small

BUGFIX: uiswitch plugin could randomly crash before displaying a dialog box
BUGFIX: user-defined menu items with printable hotkeys were interfering with the built-in notepad

BUGFIX: viewer_set_titlebar_height() was broken
BUGFIX: wince debugger could hide the process pages that were not yet present in the memory but would be loaded upon a page fault

BUGFIX: PIC: numbers with leading zeroes were displayed with too many leading zeroes

BUGFIX: ad218x: some ALU/MAC instructions were dissasembled incorrectly
BUGFIX: text version could momentarily display some garbage characters at the start

BUGFIX: the cursor was not positioned on the last selected xref in xref selection dialog box; this happened if the address was present multiple times in the list

29/01/2009