Anti-debugging under Linux using vm86

Pierre Bourdon

LSE 2013

December 9, 2011

Anti-debugging

- Reduces a debugger efficiency
- "Security" by ofuscation
- Makes dynamic binary analysis harder/misleading

- Virtual 8086 Mode
- First appearance on the Intel 80386
- Emulates 8086 real mode for compatibility purposes
- Only on x86-32, removed from x86-64

Syscall

- Two syscalls: `vm86, vm86old`
- Not a lot of differences
- Almost no documentation, the few that exists is wrong
- You have to go look at the headers and dosemu source code

Usage

- Initialize a `vm86_struct` structure
- Map some code at an offset below 64k
- Set the EIP in the structure to your entry point
- Call `vm86(VM86_ENTER, &st)`

During 8086 mode

- The processor is in 16 bit mode using a virtual context
- You can do whatever you want
- Returns in case of signal or interrupt
- Macros to get the return cause and interrupt number

How I used it

- Control flow emulation
- Algorithm obfuscation
- Confusing the disassembler
- Confusing debuggers

How can it be debugged?

- `ptrace` does not work!
- When the process get SIGTRAP-ed, it gets out of vm86 mode
- Kernel land hooking

Similar techniques

- `modify_ldt`
- Works on Linux x86-32, x86-64, Win32, Win64
- Confuses the debugger by using a non standard code segment

Conclusion

- Once in a while, look at a random syscall
- Remember that debuggers can miss a lot of stuff
- Static analysis FTW, but can be made very hard