



Software for cracking software. Selecting tools for reverse engineering

Written by [Nik Zerof](#)





Every reverse engineer, malware analyst or simply a researcher eventually collects a set of utility software that they use on a daily basis to analyze, unpack, and crack other software. This article will cover mine. It will be useful to anyone who has not yet collected their own toolset and is just starting to look into the subject. However, an experienced reverse engineer must also be curious about what other crackers are using.

WARNING

This article is for information purposes only. Neither the editorial team nor the author assumes any responsibility for possible harm that may arise from the use of these materials.

Debuggers

Debugging an application is an essential part of studying it, so every reverse engineer needs a debugger at the ready. A modern debugger must support both Intel architectures (x64 and x86), so this is the first prerequisite.

We must also be able to debug kernel-mode code. You will need this every once in a while, especially if you want to look for zero-day vulnerabilities in OS kernels or reverse engineer malware in drivers. The main candidates are x64dbg and WinDbg. The first debugger works in user mode, while the second one can debug kernel-mode code.

x64dbg

x64dbg.com

This is a modern debugger with a good user interface, a worthy successor of OllyDbg. It supports both architectures (x64 and x86), and there are tons of useful plugins.

Project7.exe - PID: 5C - Модуль: project7.exe - Thread: Главный поток E14 - x64dbg

Файл Вид Отладка Трассировка Модуль Избранное Параметры Справка Nov 13 2018

CPU График Журнал Заметки Точки останова Карта памяти стек вызовов SEH Сценарий Отладочные символы Исходный код Ссылки Поток Snowman Декрипторы Трассировка

48:2301 and rdx,rcx
40:8871 FF mov byte ptr ds:[rcx-1],s1
48:8BCB mov rcx,rbx
E8 6ACAFBFF call project7.7FF649063215
B5C0 test eax,ecx
75 00 jmp project7.7FF6490A678C
E8 8F85FBFF call project7.7FF649061D4
8BF3 mov esi,ebp
C700 2A000000 mov dword ptr ds:[rax],2A
48:8BC6 mov rcx,rbx
E8 3BCFBFF call project7.7FF649062E1F
E8 17 call project7.7FF6490A678C
E8 7885FBFF call project7.7FF649061D4
48:83CE FF or esi,esi
C700 16000000 mov dword ptr ds:[rax],16
8BE1 mov ebp,esi
8BF3 mov esi,ebp
8BE1 mov ebp,esi
8BF3 mov esi,ebp
E8 8BCFBFF call project7.7FF649062E1F
8BC6 mov eax,esi
48:8B9C24 80000000 mov rbx,qword ptr ss:[rsp+80]
48:83C4 3D add rsp,30
41:5F pop r15
41:5E pop r14
41:5D pop r13
41:5C pop r12
5F pop rdi
5F pop rsi
5D pop rbp
C3 ret
45:13C9 xor r9d,r9d
48:897424 20 mov qword ptr ss:[rsp+20],rsi
45:13C0 xor r8d,r8d
33D2 xor edx,edx
33C9 xor ecx,ecx
E8 8D00FBFF call project7.7FF649061BA0
int3
int3
ret3

rdx: __enc\$textbss\$end+563
rax: __enc\$textbss\$end+563, 2A: " "
setenv.cpp:151
setenv.cpp:152
setenv.cpp:156
setenv.cpp:177
rax: __enc\$textbss\$end+563
setenv.cpp:356
setenv.cpp:357
setenv.cpp:358
setenv.cpp:344

Скрыть FPU

RAX 00007FF649061217 <project7.EntryPoint>
RBX 0000000000000000
RCX 000000A0B24D6000
RDX 00007FF649061217 <project7.EntryPoint>
RBP 0000000000000000
RSI 000000A0B239FD78
RDI 0000000000000000
R8 000000A0B24D6000
R9 00007FF649061217 <project7.EntryPoint>
R10 0000000000000000
R11 0000000000000000
R12 0000000000000000
R13 0000000000000000
R14 0000000000000000
R15 0000000000000000
RIP 00007FF649061217 <project7.EntryPoint>
RFLAGS 0000000000000244
ZF 1 PF 1 AF 0
OF 0 SF 0 OF 0
CF 0 TF 0 IF 1
LastError 000003E3 (ERROR_IO_PENDING)
LastStatus 00000103 (STATUS_PENDING)
GS 0028 FS 0053
ES 0028 DS 0028
CS 0033 SS 0028
ST(0) 0000000000000000 x87r0 нулю 0.0000000000000000
ST(1) 0000000000000000 x87r1 нулю 0.0000000000000000
ST(2) 0000000000000000 x87r2 нулю 0.0000000000000000

По умолчанию (x64 fastcall) 5 Разблокировано

1: rcx 000000A0B24D6000
2: rdx 00007FF649061217 <project7.EntryPoint>
3: r8 000000A0B24D6000
4: r9 00007FF649061217 <project7.EntryPoint>

Примечание: проект7.exe:531217 #617 <EntryPoint>

Дан 1 Дан 2 Дан 3 Дан 4 Дан 5 Просмотр 1 Локальные переменные Структура

Адрес Шестнадцатеричное АСII

00007FFD9C8E1000 00 00 00 00 00 00 00 00 F6 41 26 01 0F 83 14 00 B.....0A6...Y
00007FFD9C8E1010 0A 00 F6 41 26 02 75 25 48 88 01 48 88 08 0F B7 ..0A6.u0n...n...
00007FFD9C8E1020 41 18 0F B7 4A 18 48 33 C1 0F B7 00 30 20 15 00 A...3.n3A...D..
00007FFD9C8E1030 48 33 C1 0F B7 4A 18 33 C1 0F B7 00 C2 C3 48 8D B2 n3A...n3A.n.AA...
00007FFD9C8E1040 FF 0F 00 00 48 25 00 F0 FF FF 48 05 00 10 00 00 y...n3L.byu...n...
00007FFD9C8E1050 C3 CC CC CC CC CC CC CC F6 41 26 03 0F 85 FC DC A11111110A6...u
00007FFD9C8E1060 0A 00 0F B7 41 24 C3 CC CC CC CC CC CC CC CC ...A5A111111111
00007FFD9C8E1070 48 83 EC 28 80 B9 82 01 00 00 02 4C 88 D2 75 54 H.t(...)...L.out
00007FFD9C8E1080 4C 88 89 78 01 00 00 4C 89 0A 88 02 80 00 00 66 L...K...l...f
00007FFD9C8E1090 B9 42 12 49 8B CA 48 C7 42 08 C0 00 00 33 02 .B.L.EKb.A...30
00007FFD9C8E10A0 E8 38 00 00 00 41 88 52 11 41 88 43 28 41 28 C1 e:...A.R.A.A(A+A
00007FFD9C8E10B0 41 89 42 18 41 88 41 30 41 28 41 28 41 21 10 41 A.B.A.A0A+A(A!A
00007FFD9C8E10C0 B9 42 1C 4D 89 4A 20 49 88 41 30 49 89 42 28 48 .B.M.3.L.A02.BCh
00007FFD9C8E10D0 B1 C4 28 C3 45 33 C9 E8 AE CC CC CC CC CC CC .A(AE3E*11111111
00007FFD9C8E10E0 F6 41 12 02 88 51 10 75 04 48 89 51 24 C3 CC CC 0A...Q.u.H.Q5A11
00007FFD9C8E10F0 CC CC CC CC 4C 8B CC R1 CC 88 00 00 48 88 1111.Lm.V...n
00007FFD9C8E1100 05 0B 53 16 00 48 33 C4 48 89 44 24 7D 48 8B 84 ..S...H3An.D3n...

Команда: [Принудительно] INT3 точка останова "Останов в точке входа" на <project7.EntryPoint> (00007FF649061217)

Время под отладкой: 0:00:01:17

x64dbg

Project1.exe - PID: 458 - Модуль: project1.exe - Thread: Главный поток 2678 - x64dbg

Файл Вид Отладка Трассировка Модуль Избранное Параметры Справка Nov 13 2018

CPU График Журнал Заметки Точки останова Карта памяти стек вызовов SEH Сценарий Отладочные символы Исходный код Ссылки Поток Snapshot Дескрипторы Трассировка

File Analyse View Help

```
7ff701254677 struct s0 {
7ff701254679     int8_t[95] pad95;
7ff70125467c     uint32_t f95;
7ff70125467f };
7ff701254681
7ff701254684
7ff701254687 int64_t __guard_dispatch_icall_fptr = 0x7ff701249c4c;
7ff70125468c
7ff701254690
7ff701254694 int64_t GetProcessHeap = 0x7ff701249f20;
7ff701254699
7ff70125469c int64_t HeapFree = 0x7ff701249f40;
7ff70125469f
7ff7012546a4 void fun_7ff701254677(uint32_t ecx) {
7ff7012546a9     int64_t rcx1;
7ff7012546ac     int1_t cf2;
7ff7012546af     int32_t r13d3;
7ff7012546b1     struct s0* rbp4;
7ff7012546b6     int64_t rax5;
7ff7012546be     uint32_t* r15_6;
7ff7012546c0     void* r14_7;
7ff7012546c4     int8_t al8;
7ff7012546c8     int64_t rax9;
7ff7012546cf
7ff7012546d2     *(uint32_t*)&rcx1 = ecx;
7ff7012546d5     if (cf2 || (*(uint32_t*)&rcx1 = *(uint32_t*)&rcx1 + r13d3, *(int32_t*)((int64_t)&rcx1 + 4) - 0, *(uint32_t*)&rcx1 >= rbp4->f95)) {
7ff7012546dc         *(int32_t*)&rax5 = static_cast<int32_t>(rcx1 - 1);
7ff7012546df         *(int32_t*)((int64_t)&rax5 + 4) = 0;
7ff7012546e2         *r15_6 = *(uint32_t*)((int64_t)r14_7 + rax5 * 8 + 4) & 0xffffffff;
7ff7012546e8         al8 = (int8_t)__guard_dispatch_icall_fptr();
7ff7012546ec         if (al8) {
7ff7012546ef             }
7ff7012546f6         rax9 = (int64_t)GetProcessHeap();
7ff701254700         HeapFree(rax9);
7ff701254703         __guard_dispatch_icall_fptr();
7ff701254707         __guard_dispatch_icall_fptr();
7ff70125470d         __guard_dispatch_icall_fptr();
7ff701254711         __guard_dispatch_icall_fptr();
7ff701254714     }
7ff701254718 }
7ff70125471e
7ff701254720
7ff701254725
```

[Info] Decompilation completed.

Команда: project1.exe: 00007ff701254677 -> 00007ff701254724 (0x0000000AE bytes)

По умолчанию

Принести в буфер

Время под отладкой: 0:00:03:23

Built-in decompiler

Granted, it has its downsides as there are a number of annoying bugs. But it is actively developed and supported. Since the debugger works in user mode, it is of course vulnerable to a wide range of anti-debugging techniques. This is, however, in part offset by the availability of many different debugger hiding plugins.

x64dbg has a built-in decompiler and imports reconstructor (both x64 and x86), supports code graph visualization and read/write/execute/access breakpoints. This debugger has enabled some hackers to break down the infamous Denuvo DRM system!

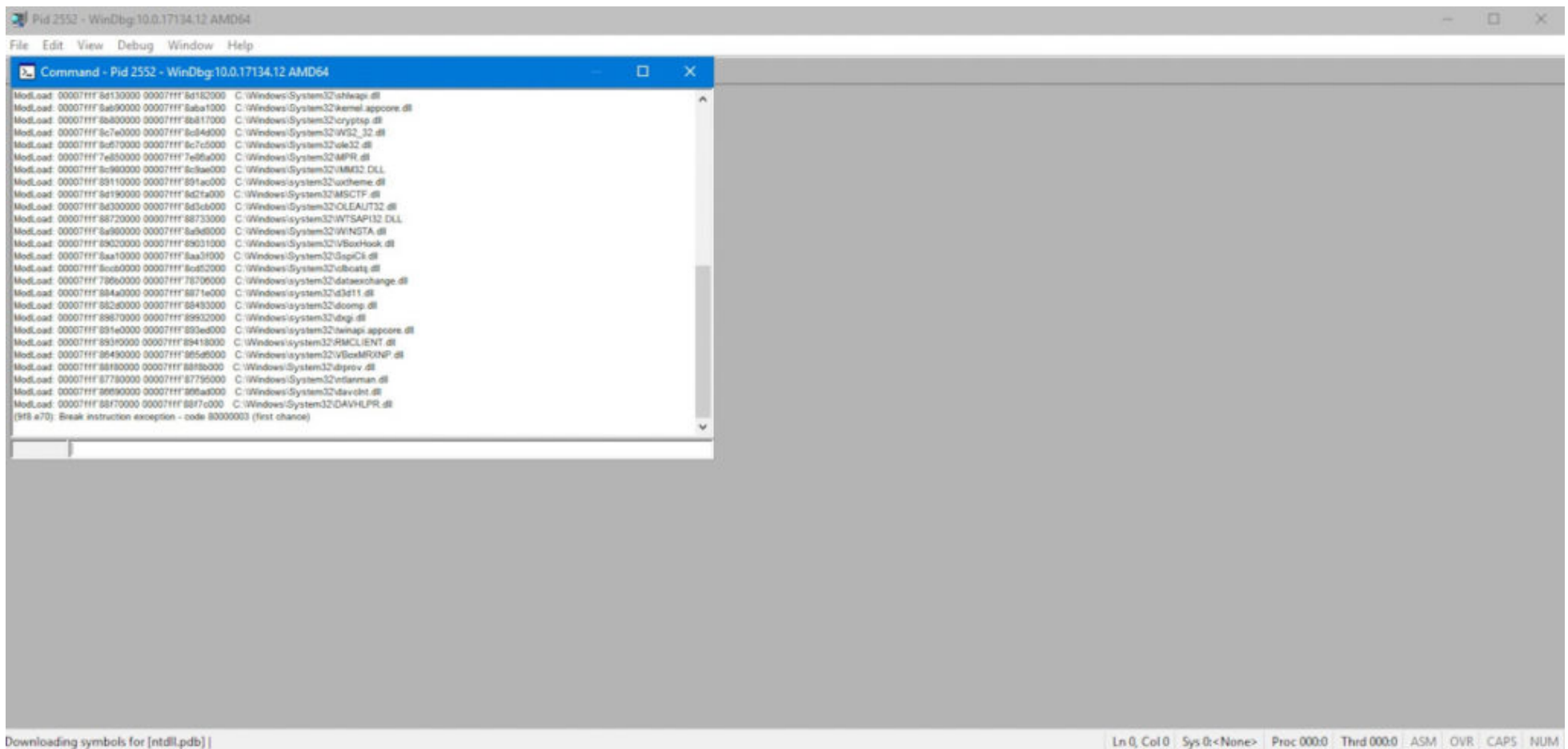
Why not OllyDbg

We haven't included OllyDbg here because it is very outdated. It does not support the latest operating systems or x64. The app's official website announced a x64 version and even reported some development progress, but the site itself has not been updated since 2014. OllyDbg is undoubtedly a milestone piece of software, but now it seems that its time has passed. There have also been fewer kernel mode debuggers since Syser Kernel Debugger, a successor to SoftICE, was abandoned.

WinDbg

[Official homepage](#)

WinDbg is one of the best kernel or driver debugging tools. This debugger is supported by Microsoft and included in the Windows Driver Kit (WDK). This is currently the most up-to-date and powerful kernel code debugger. It does not feature the user-friendly interface of x64dbg, but there are not many other options, as other debuggers don't support kernel-mode code.



WinDbg supports remote debugging and can download debug symbols directly from Microsoft servers. The VirtualKD booster exists to speed up the WinDbg setup for debugging kernel-mode code in a VM. WinDbg is definitely not for beginners, but as you gain experience in reverse engineering and start testing various interesting options, you won't be able to avoid it.

WinDbg enables you to view various system structures and easily disassemble NTAPI functions. Of course it can also be used to debug “regular” applications, but I prefer to unleash this powerful weapon only when it is really needed!

PLEASE SUBSCRIBE TO READ FULL ARTICLE

[Already subscribed?](#)

1 YEAR

for only **\$29**

Pay with Card

With subscription you are free to read all of the materials of Hackmag.com.

[Read more about the project](#)



Please subscribe to view comments

Only subscribers can participate in the discussions. You may [login in to your account](#) or [sign up to Hackmag](#) and [pay a subscription](#) to access the discussions.

Full access subscription
for only \$29 per year!

Pay with Card

Search

Search

Recent Posts

- [Software für das Cracken von Software. Auswahl von Tools für das Reverse Engineering](#)
- [Software for cracking software. Selecting tools for reverse engineering](#)
- [Attacking a car alarm. How does a car alarm security system work?](#)
- [What to See on the Darkweb: A Travel Guide to Hidden Services](#)
- [Tips&tricks: Android's hidden capabilities that everyone should know](#)

Recent Comments

- aquageek on [Automation for OS X: the JavaScript way](#)
- Vx00001 on [Dive into exceptions: caution, this may be hard](#)
- gangaprabha on [A saga about a packer which can prepare similar OS images for development and production scenes in various environments](#)
- try harder-Legend's BLog on [TOP-10 ways to boost your privileges in Windows systems](#)
- dxon on [Building weather station with STM32F3DISCOVERY and WizFi220 Wi-Fi module](#)

HackMag.com © 2019

[HackMag.com](#) publishes high-quality translated content about information security, cyber security, hacking, malware and devops.