The Reverse Engineers Toolkit

by R4ndom on May.31, 2012, under Reverse Engineering, Tools, Tutorials

I remember when I first got started in reverse engineering. Well, let's be honest. It was cracking commercial software. but I digress. Anyway, when one first starts getting introduced to this world, it doesn't take long to learn a very important tenet: Tools are King. Most budding young crackers start getting the *'tool-bug'* and start amassing as many tools as possible; it doesn't matter that you don't know what most of them do (or that half are actually viruses), just that it felt like the more tools one had, the better potential cracker they would be.

After the initial *tool-buzz* wears off you soon learn that most of those tools (and viruses) you amassed really aren't that pivotal in day-to-day cracking (if there is such a thing). Hell, most of them you never really learn what the heck they do anyway! In fact, after you've been reverse engineering as long as I have (much longer than I like to admit) you eventually learn that you really only use a small subset of all the tools out there. Some you use regularly, some semi-regularly, and some rarely, tho all of the ones you use have a purpose.

It can be hard for a beginner to learn even what tools are out there, much less the tools that are really important and which aren't. In order to help those who are interested in getting into reverse engineering (and yes, *sigh*, cracking) I have put together a list of what I consider the most important tools for really getting involved in RCE. I have arranged the list by several qualities, first of which is how often you would use them, followed by their importance and experience needed to use them, and finally where you can get them. If they are tough to find, I am hosting on this site so you can download them here. I have also included a description of what the tool does. And just because everything sounds cooler as an anagram, I am calling it

R4ndom's "Reverse Engineering And Cracking Tools Of Note" or R.E.A.C.T.I.O.N.

I know it doesn't change anything, but damn, it sounds cool!!!!

Now, before you begin flaming me with your "How could you not include tool X!!!" and "Tool Y SUCKS!!!", please keep in mind that these are tools ordered by importance for *ME*. I know that everyone will not share my same viewpoints, but I hope to at least get the beginner started. And my ratings may be a little 'loose'.

And lastly, don't get me wrong, I still get excited when I learn of a new tool, even if it was programmed in 1997, Norton won't even let me open it, it's packed with Themida, and it's called BackOrifice. Just the idea that it might be that long lost 'secret' program that let's you unpack, un-protect, disassemble into proper English, debug and remove all copy-protection with the click of a single button, well, you never know. Some dreams you just don't get over.

DAILY

OllyDBG

Importance: 10
Experience Needed: 6
Where can you get it: www.ollydbg.de/

If reverse engineers were fish, Olly Debugger pronounced Ollydebug and always spelled OllyDBG) would be the water, the tank, the miniature sunken ship, and the fish food. Every reverse engineer at some point (usually first) get's acquainted with Olly. If I was on a deserted island and could only bring one tool (besides matches, flint, a knife, a flare gun, a stove, a cell-phone, or a yacht) it would be Olly.

Olly Plugins

Importance: 10
Experience Needed: 3
Where can you get it: On my tools page.

The second most important thing behind OllyDBG are the plugins for OllDBG. There are many, and without them, the reverser's life would be miserable. Anything from automatically finding the Original Entry Point in a packed binary, to thwarting pretty much every anti-debugging trick invented, the plugins are a must have. (See my guide to Olly plugins for specific information on some of the most important one).

Packer/Protector Identifiers

Importance: 9.5
Experience Needed: 2
Where can you get it: On my tools page.

Almost all binaries now-a-days are packed with some sort of protection software, sometimes to make the binary much smaller, but, more importantly, to keep people like us doing what we do. The first offense in overcoming these hurdles are knowing which of the dozens of packers the binary in question is packed with. These programs automatically identify not only a packer or protector, but also what language they were written in, and some even suggest an appropriate script or un-packer to use to get rid of it! Also, in this field, you can never have enough when it comes to identifiers. The most stable (and the ones I use religiously) are PEid, DIE, eXeInfoPE and RDG Packer Detector. Get them on my tools page.

ImpREC

Importance: 9.5
Experience Needed: 8
Where can you get it: on my tools page

Import REConstructor is an invaluable tool for rebuilding binaries that have been mangled by packers and protectors. Generally, you start using this tool after you have gained some experience in reverse engineering, as it involves some pretty detailed areas. But for what it does, there's nothing else like it.

A Good Text Editor

Importance: 9
Experience Needed: 8
Where can you get it: notepad-plus-plus.org/

Sometimes it's the small things that make the biggest differences. A good text editor is one of them. You will use it often, between reading readme's, taking notes, quickly looking at source code, you name it. I use Notepad++, but any text editor developed with programmers in mind would work.

PE Viewers/Editors

Importance: 8.5
Experience Needed: 8
Where can you get it:www.ntcore.com/

At some point or another, no matter how much you push it off, as a reverse engineer you are going to have to tackle the PE header. Sure, you can create your own PE viewer in assembly language 'just for the heck of it', but fortunately, there are also a lot of progs out there that make it a lot easier. My first choice is CFFExplorer Suite. I think it has a nice set of options and allows a lot of editing. PEBrowse is also good in that it is almost a full disassembler. There is also PEditor and PEView, just to name a few. It doesn't really matter which tool, just as long as you can view and edit PE information. For CFFExplorer, follow the link above. All of the others are on my tools page.

zip/rar file packers/unpackers

Importance: 8
Experience Needed: 1
Where can you get it: www.7-zip.org/

This is one of those tools you take for granted. Try and see how many of the tools in this list you can run without a zip/rar un-packer…Need I say more?

Notes Software

Importance: 6
Experience Needed: 3
Where can you get it: www.mytreedb.com/

This one is more for the 'professional' reverse engineer, though I think it should be used by everyone. I can't tell you how many times I had to search and search for a specific piece of information before I started using a note-taking app. TreeDBNotes is more than simple note-taking, it also allows you to embed pictures, media files, clippings and link between the various pages. Here's an example; I have folders for every packer I have dealt with. In each folder is a link to the packer, tutorials I have collected for that packer, lists of scripts and plugins that I have had success with, options that can be selected on that packer and how to defeat them, as well as code snippets to show how it does it. It's hard to start, but after using it for a while, it will become invaluable.

WEEKLY

API guide

Importance: 8
Experience Needed: 4

Let's face it- API's are hard to remember (pop quiz: what is the fourth parameter passed to CreateWindowEx?) Add to that that there are literally thousands of them and you got a serious problem. Finally, how are you supposed to reverse engineer a program when, half way through a very important function, the app calls SetImageConfigInformation? Answer: you can't. But there is help. There are downloadable API Help files, books, references, and the most elegant solution, adding an API help file

directly into Olly 😃

Hex Editors

Importance: 6
Experience Needed: 6
Where can you get it: www.hhdsoftware.com/hex-editor

Hex editors are extremely important in reverse engineering. The options for using a hex editor are vast (patches, PE header viewers, finding code caves…). The choice is really a personal one. I prefer HHD Hex Editor, and not just because it's free. It's just simple and gets the job done. I would list other options, but frankly, there are hundreds, so some experimentation would help. I think that once you get to the point that you need a hex editor, it will be more obvious which one is for you.

DEDE : The Delphi Disassembler

Importance: 5
Experience Needed: 5
Where can you get it: www.softpedia.com/

Delphi creates its own set of challenges when reverse engineering and DEDE is a lifesaver. It allows you to take a binary compiled in Delphi, disassemble it, view all of the forms, buttons, function calls and so forth, and even export a MAP file that can be imported into Olly to help with the naming of functions. I don't even consider attempting to disassemble a Delphi binary without it.

P32DASM

Importance: 5
Experience Needed: 5
Where can you get it: www.woodmann.com/

What I said above can also be said for binaries compiled with Visual Basic. VB can be even worse due to the fact that 99% of the code is run in the VB run-time library. It is a complete mess when trying to reverse engineer. P32Dasm is similar to DEDE in that it allows you to see the binary in a much more user-friendly view. It also allows MAP files to be exported (and imported into Olly), and also includes a complete disassembler. You can technically reverse engineer (and patch) the program from start to finish in P32Dasm if so inclined.

IDA Pro

Importance: 4
Experience Needed: 8

I already know I'm going to take major hell over this, but before you start inundating me with hate mail, let me say this: Generally, reverse engineers use a static disassembler or a dynamic disassembler, but not both, and it comes down to personal choice and what you learned on. For years I have used Olly and I rarely use IDA, and seeing as this is *my* toolbox, I put IDA down on the list. That being said, IDA is the best static disassembler out there, and if you are one of the reverse engineers who use it primarily as your disassembly tool, the importance of this tool would be a 10. Even if Olly is your -go-to- tool, IDA can help immensely in de-compiling binaries. It is a hugely intricate program with a staggering amount of options, and if the time is spent really learning it, it's awe inspiring to watch. Even if you start with Olly, take the time to at least mess with IDA and see what it has to offer.

dUP2 Universal Patcher

Importance: 4
Experience Needed: 7
Where can you get it: on my tools page.

You've just figured out what line to patch in that super-duper app and can make it run without that horribly annoying nag screen! Now what? Run Olly every time you want to run the app, patch that line, and hit run? No. You can go into a hex editor and change it in the binary, but what if you want to share it? Now you have to send the huge exe file with the installer. And what happens when there's a version upgrade? Enter the patcher. It allows you to create a stand-alone program that will dynamically apply patches to a binary. dUP2 offers many ways of doing this (as well as making a 'loader') and the new one (v2) is even more powerful. Definitely a must for any reverse engineer that wishes to share his 'booty'.

OllyDBG 2.0

Importance: 4
Experience Needed: 6
Where can you get it: www.ollydbg.de/

I put this under weekly instead of daily because 1) it's hard to give up tools you know so well (Olly V1.10) and 2) not all plugins have been ported to ver.2 yet. That being said, this version rocks. There have been a lot of major enhancements, it works on x64 machines, and there have been a lot of bug fixes. I'm sure, over time, this version will replace the old one and will become the new tool of choice.

LordPE

Importance: 3
Experience Needed: 6
Where can you get it: On my tools page.

I put LordPE as an importance of 3 only because I don't personally use it that often, but I know of a lot of engineers out there who do, so for them I'm sure it would be much higher. Personally, I have other tools that I'm just more familiar with that do the same thing. That being said, LordPE allows for editing PE Headers, dumping processes from memory, analyzing binaries, viewing tasks, and a PE builder. The one thing I do use this tool for regularly is it's PE file checker: after you rebuild IATs and ditch unused thunks, it's good to validate them.

Olly Scripts

Importance: 3
Experience Needed: 3
Where can you get it: Everywhere.

At a certain point, it becomes a little monotonous performing the same actions over and over. Fortunately, Olly has a complete scripting language, and the RE community has certainly used it!. Scripts allow every conceivable version of every possible packer to be automatically unpacked, OEP's found, code caves populated, stolen bytes un-stolen, you name it. Of course the hard part is finding the right one, what with so many to choose from…

Sandboxie

Importance: 3
Experience Needed: 2
Where can you get it: www.sandboxie.com/

You can read my blog on using Sandboxie in cracking, but to re-iterate, it's a lifesaver with finding apps to learn to reverse engineer; you can install the app and find out what packer it is packed in, what language it was compiled in, even check it for viruses, all without making a single change to your system.

ShowString

Importance: 2
Experience Needed: 1
Where can you get it: On my tools page.

I am really sorry to the author of this app, but I cannot remember where I got it. It is a simple app with a simple objective: show me all the ASCII strings in an app. Now, many tools on this list will do just that, some even better than ShowString, but for ease of use and when you don't feel like running a big tool just to check strings, it comes in handy. You can also port the output, which I like.

Unpackers

Importance: 2
Experience Needed: 5
Where can you get it: Many places

For those reverse engineers who didn't pay attention in these tutorials (or any other tutorials for that matter), there are unpackers out there that will unpack packed binaries. Of course, lazy engineer's are not the only one's who use these, I use them sometimes because it's faster than going through the whole rigamarole of unpacking a binary with a packer you have unpacked 10,000 times. Ok, it's because I'm lazy. But if you're just starting out, please, please, please skip these and learn how to unpack binaries manually. You'll be happy you did.

MONTHLY

Kernel Debuggers

Importance: 9
Experience Needed: 10
Where can you get it: Windows SDK

Though used more for malware research, sometimes a ring 0 kernel debugger can come in handy. If the protection scheme uses multiple mutexes spawning multiple threads, a kernel debugger is the tool to use. Among the choices are WinDBG from Microsoft and SoftIce. SoftIce is a little old but allows you to debug on the same computer you are debugging on. WinDBG requires a different computer (or you can run it in a VM).

Handle

Importance: 6
Experience Needed: 5
Where can you get it: Sysintenrnals

Sometimes a protection scheme uses some sort of file open or registry key mechanism to keep you from terminating a thread. Usually this thread is keeping you from changing the main thread that needs to be patched. Handle shows all of the handles a program is currently holding, so you can track down exactly which process is keeping you from terminating a thread.

WinASM

Importance: 5
Experience Needed: 3
Where can you get it: www.winasm.net/

Occasionally you need to write assembly code to inject as a DLL or to hook an API. There are several assembler IDE's out there, and WinAsm is a little dated, but it just works (and it's free). Though you need to download MASM, it is still a very easy install.

Denabler

Importance: 3
Experience Needed: 3
Where can you get it: www.secforce.com/

Every once in a while you have a feature disabled in some way as part of the protection scheme, for instance the "Save" button is greyed out until you purchase the product. Denabler can help in this case by allowing you to attach to a window and enable any resource in it. It also allows you to do more such as adding menu options and changing resources, but mostly I use it to enable disabled buttons.

Dependency Walker

Importance: 2
Experience Needed: 3
Where can you get it:www.dependencywalker.com/

Sometimes it gets a little confusing trying to figure out which DLL requires which function from another DLL (confused yet?) Dependecy Walker helps by showing all of the imported (and exported) functions in a binary and exactly which ones depend on others.

System Monitors

Importance: 1.5
Experience Needed: 1
Where can you get it: Many places

Very occasionally (mostly when I'm being lazy) I will run an app and I don't feel like tracing through all of the code to see what registry key or init file was modified. There are several system monitor apps out there that can keep track of this: RegShot takes a system snapshot before and after running an app and tells you every change that was made, Process Monitor tells you every process that was started and stopped while the binary ran, and API monitor tells you every API that was called. Again, not used very often, but helpful none the less.