# WinAppDbg Documentation

*Release 1.6*

**Mario Vilas**

**Aug 28, 2019**

# Contents

CHAPTER 1

# Introduction

The *WinAppDbg* python module allows developers to quickly code instrumentation scripts in **Python** under a **Windows** environment.

It uses **ctypes** to wrap many Win32 API calls related to debugging, and provides a powerful abstraction layer to manipulate threads, libraries and processes, attach your script as a debugger, trace execution, hook API calls, handle events in your debugee and set breakpoints of different kinds (code, hardware and memory). Additionally it has no native code at all, making it easier to maintain or modify than other debuggers on Windows.

The intended audience are QA engineers and software security auditors wishing to test or fuzz Windows applications with quickly coded Python scripts. Several *ready to use tools* are shipped and can be used for this purposes.

Current features also include disassembling x86/x64 native code, debugging multiple processes simultaneously and produce a detailed log of application crashes, useful for fuzzing and automated testing.

Here is a list of software projects that use *WinAppDbg* in alphabetical order:

- Heappie! is a heap analysis tool geared towards exploit writing. It allows you to visualize the heap layout during the heap spray or heap massaging stage in your exploits. The original version uses vtrace but here's a patch to use WinAppDbg instead. The patch also adds 64 bit support.
- PyPeElf is an open source GUI executable file analyzer for Windows and Linux released under the BSD license.
- python-haystack is a heap analysis framework, focused on classic C structure matching. The basic functionality is to search in a process' memory maps for a specific C Structures. The extended reverse engineering functionality aims at reversing structures from memory/heap analysis.
- SRS is a tool to spy on registry API calls made by the program of your choice.
- Tracer.py is a "small and cute" execution tracer, in the words of it's author :) to aid in differential debugging.
- unpack.py is a script using WinAppDbg to automatically unpack malware, written by Karl Denton.

And this is a list of some alternatives to *WinAppDbg* in case it doesn't suit your needs, also in alphabetical order:

- InfinityHook lets you hook system calls, context switches, page faults, DPCs and more. *InfinityHook* works along side Patchguard and VBS/Hyperguard to subtly hook various kernel events. It works in Windows 7 and above. Sadly it does not have a Python wrapper at the time of writing this but if you write one and combine this tool with *WinAppDbg* please let me know! :)

- ImmLib is a Python library to integrate your custom scripts into *Immunity Debugger*. It can only function inside the debugger, but it's the best solution if you aim at writing plugins for that debugger instead of standalone tools.
- OllyPython is an *OllyDbg* plugin that integrates a Python debugger. Naturally it only works within OllyDbg and is not suitable for standalone projects.
- PyDbg is another debugging library for Python that is part of the *Paimei* framework, but may work separately as well. It works on Windows and OSX. It predates *WinAppDbg* by quite some time but it's also been unmaintained for long, and it only works in Python versions 2.4 and 2.5. A newer branch called PyDbg64 implements 64 bit support for both platforms.
- PyDbgEng is a similar project to *WinAppDbg*, but it uses the Microsoft Debug Engine as a back end while *WinAppDbg* uses only bare Win32 API calls. The advantage of this approach is the ability to support kernel debugging, which is not allowed by the Win32 API alone. The disadvantage is having to install the Windows SDK/WDK to the machine where you run your scripts (or at least the components needed for debugging). See also the Buggery project which is based on *PyDbgEng*.
- PyDbgExt is the reverse of *PyDbgEng*: instead of instancing the *Microsoft Debug Engine* from a Python interpreter, it embeds a Python interpreter inside the Microsoft debugger *WinDbg*.
- pygdb is a simple wrapper on the GNU debugger that provides a GTK interface to it. Works in Linux and OSX.
- PyKd is like *PyDbgEng* and *PyDbgExt* combined into one - it can be both used from within the debugger and a standalone Python interpreter. Being a younger project it's still in alpha state, but looks very promising!
- PyMem is a memory instrumentation library written in Python for Windows. It provides a subset of the functionality found in *WinAppDbg*, but if you're developing a tool that only needs to manipulate a process memory you may find it convenient to support both backends and leave the choice to the user.
- python-ptrace is another debugger library for Python with the same goals as *WinAppDbg*. Here the approach used was to call the ptrace syscall, so naturally it works only on POSIX systems (BSD, Linux, maybe OSX). If Kenshoto's vtrace is not an option you could try combining this with *WinAppDbg* to implement a multiplatform tool.
- PythonGdb is an embedded Python interpreter for the GNU debugger. It's already included in GDB 7.
- Radare is a console based multiplatform disassembler, debugger and reverse engineering framework. Python is among the languages supported for plugins and scripting.
- Universal Hooker (uhooker) is a Python library to implement function hooks in other processes. While its functionality overlaps with some of *WinAppDbg*, the hooks implementation of *uhooker* is superior. Unfortunately the last update was in 2007. :(
- Vivisect (previously known as Kenshoto's vtrace debugger) is a full fledged multiplatform debugger written in Python, and a personal favorite of mine. I took a few ideas from it when designing *WinAppDbg* and, while I feel mine is more complete when it comes to Windows-specific features, this is what I'd definitely recommend for cross-platform projects.

See also the wonderful Python Arsenal for RE for another reference of security related Python tools.

CHAPTER 2

# Programming Guide