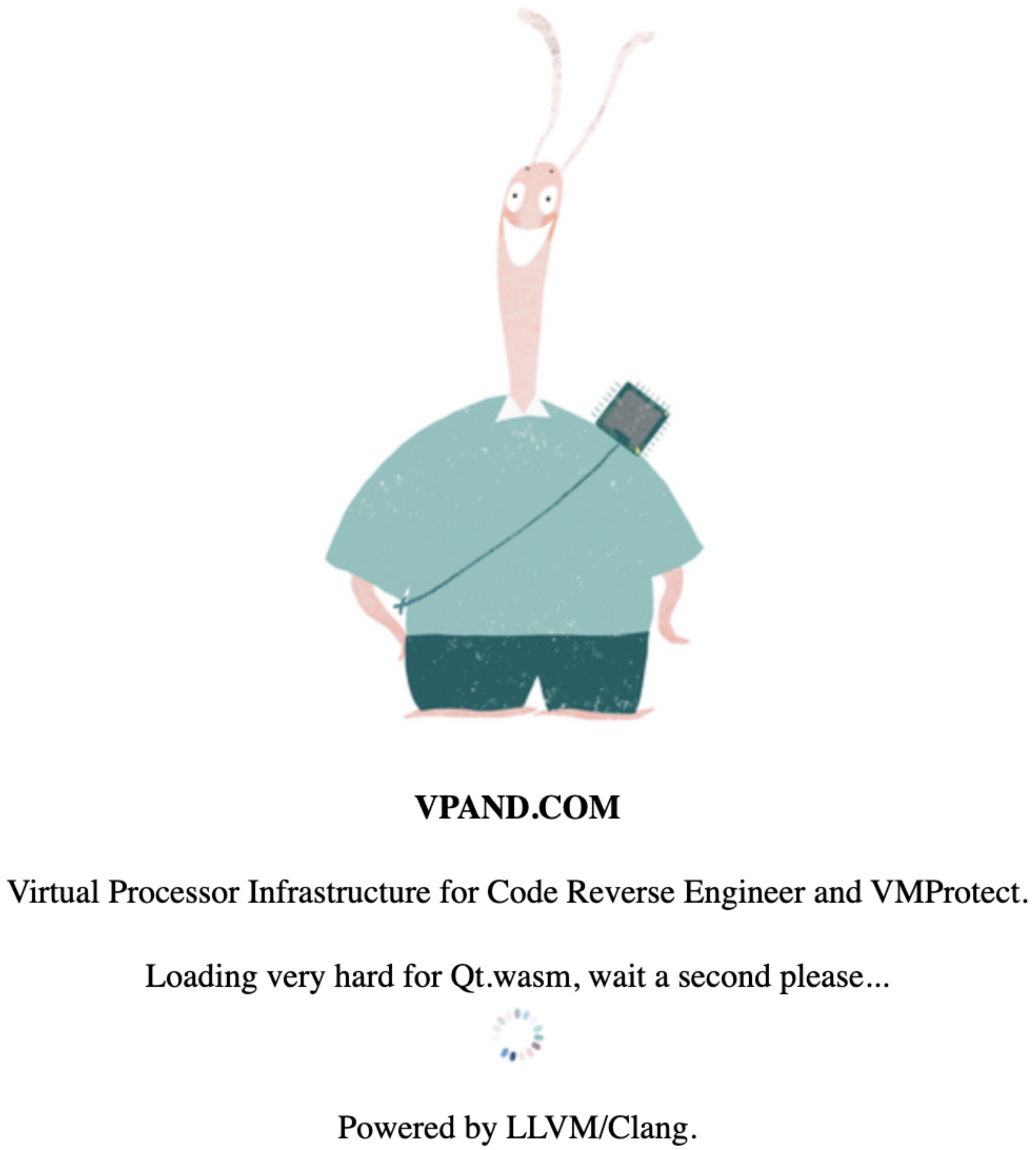
v1.0.0



VPAND.COM

Virtual Processor Infrastructure for Code Reverse Engineer and VMProtect.

Loading very hard for Qt.wasm, wait a second please...

Powered by LLVM/Clang.

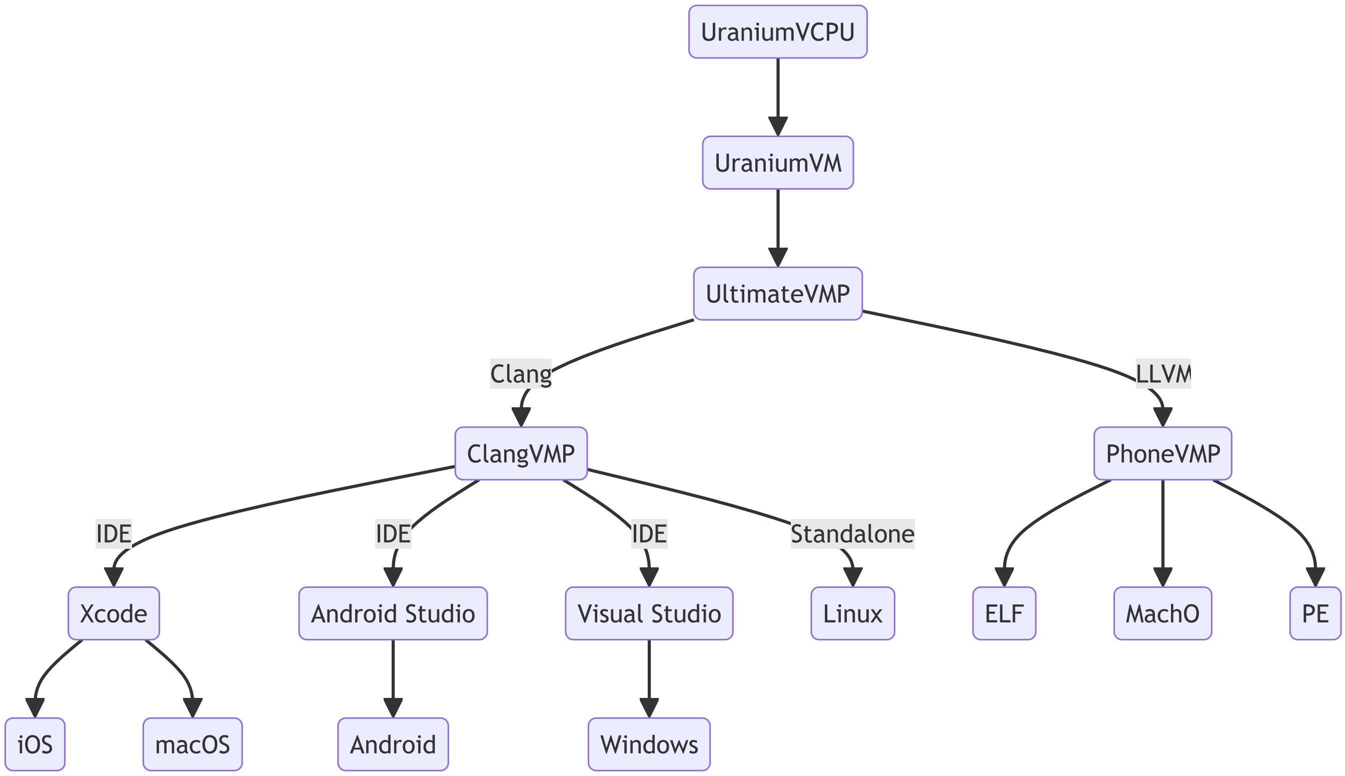
vpand.com

PhoneVMP User Manual

UltimateVMP

UltimateVMP (Ultimate Virtual Machine Protection), is an **arm/arm64/x86/x86_64** assembly level code virtualization encryption software for Darwin (**macOS/iOS**), **Linux** (Ubuntu/**Android**), **Windows** and other operating systems. It re-encodes the binary instructions of the target function into a private instruction format and then interprets the encoded instruction directly at run time. Unlike low-intensity code encryption such as obfuscation and shell, VMP code will not restore the original instruction throughout the whole execution process, so it can achieve high code encryption strength, greatly raising the threshold of reverse engineering, so as to achieve the purpose of protecting software assets. UltimateVMP currently supports platforms including **macOS, iOS, Linux, Android, Windows**, and supported architectures including **x86, x86_64, arm, and arm64**.

UltimateVMP includes two products. One is ClangVMP(based on the Clang compiler), it encodes **C/C++/ObjC/ObjC++/Swift**/etc.(static native programming language) function into VMP data during compilation process. The other is PhoneVMP, it encodes assembly instructions into VMP data from binary file(like **MachO/ELF/PE**). All of them are based on virtual processor infrastructure UraniumVM from vpand.com. Their relationship is as follows:



This manual focuses on PhoneVMP.

PhoneVMP

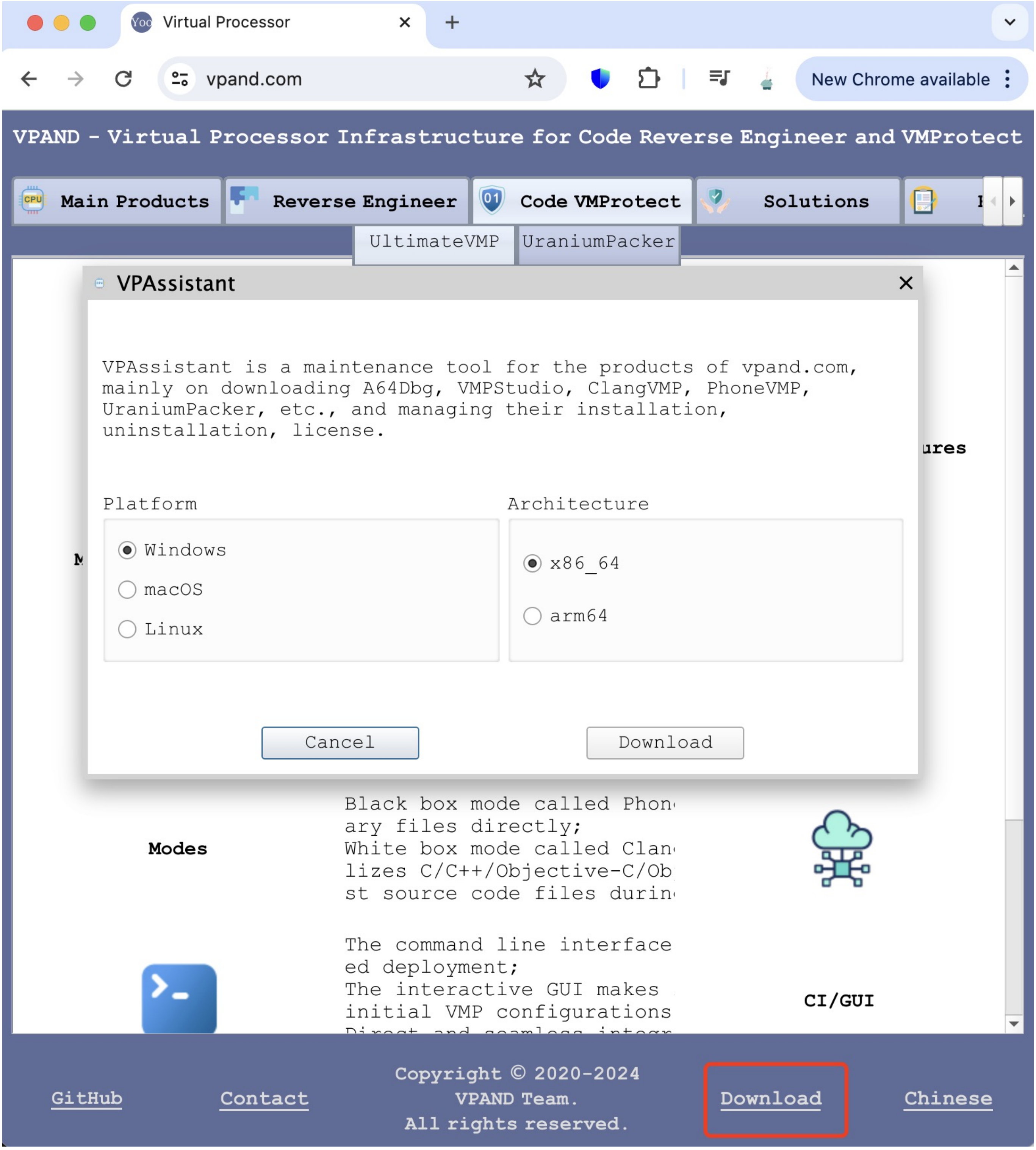
PhoneVMP is a LLVM based code virtualized tool. It encodes **Assembly Instructions** into its private VMP data. The VMP data will be directly interpreted by PhoneVMP runtime library without restoring the original instructions. Because of this mechanism, reverse engineering on this kind of code will be much more harder than raw machine code.

Till now, PhoneVMP supports running on **macOS, Windows and Linux**, and supported virtualized architectures including **x86, x86_64, arm and arm64**. We recommend **ClangVMP** in source code mode first, if you DO love vm protection in binary file mode, then PhoneVMP is still OK.

Download

All of our products are hosted in vpand.com, you can download the assistant tool called **VPAssistant** to fetch your interested product like PhoneVMP.

Please note that you should download **the exact platform and architecture** which matches your current running OS, all the real product download through VPAssistant highly depends on the architecture that it's running on. For the native performance, you'd better not download x86_64 version on arm64 macOS even though it can directly run on it with Rosetta 2.



VPAssistant

VPAssistant is a maintenance tool for the products of vpand.com, mainly on downloading A64Dbg, VMPStudio, ClangVMP, PhoneVMP, UraniumPacker, etc., and managing their installation, uninstallation, license.

Platform

● Windows
○ macOS
○ Linux

Architecture

● x86_64
○ arm64

Cancel Download

Install

PhoneVMP

As the VPAssistant on different platform(like Windows, Linux and macOS) is **absolutely the same**, unless some feature is just available on that specified platform, otherwise all the generic feature screenshots are from macOS. Here we go.

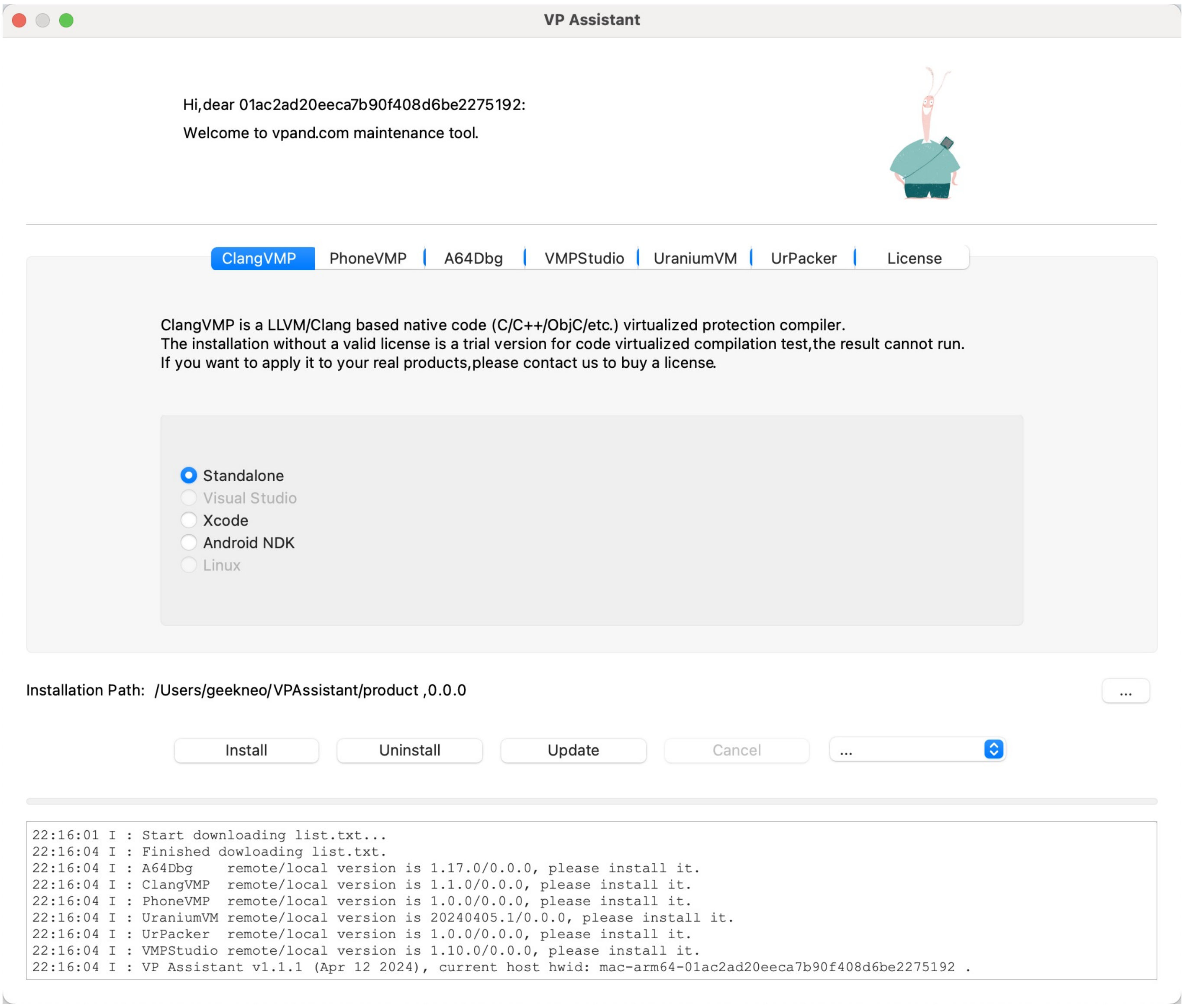
After download, unzip and launch VPAssistant, we're gonna be in the ClangVMP tab widget, then click the **PhoneVMP** tab to activate the assistant for PhoneVMP. The default installation path is(You can **change it with the "..." button** on the right):

```
macOS/Linux : ~/VPAssistant/product

Windows     : SysDrive:\Users\user-name\VPAssistant\product
```

Every time when VPAssistant finishes launching, at the end of logs there'll be a line for your running OS triple: os-arch-hwid. It's the key to authenticate your computer to fetch a valid PhoneVMP license, copy and send to us before you want to purchase and install a license.

```
current host hwid: mac-arm64-01ac2ad20eeca7b90f408d6be2275192
```



Hi,dear 01ac2ad20eeca7b90f408d6be2275192:

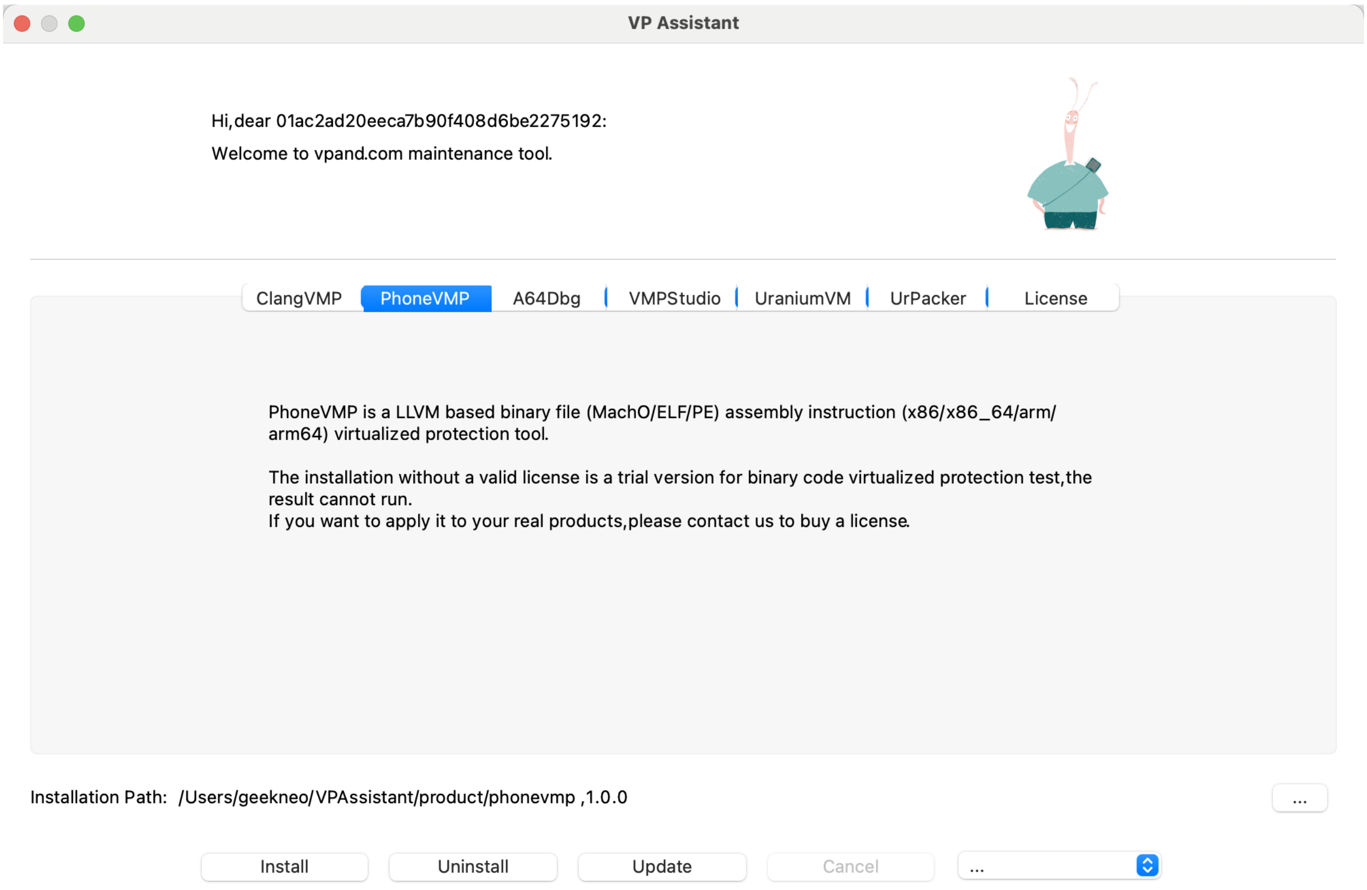
Welcome to vpand.com maintenance tool.

ClangVMP is a LLVM/Clang based native code (C/C++/ObjC/etc.) virtualized protection compiler.
The installation without a valid license is a trial version for code virtualized compilation test,the result cannot run.
If you want to apply it to your real products,please contact us to buy a license.

● Standalone
○ Visual Studio
○ Xcode
○ Android NDK
○ Linux

Installation Path: /Users/geekneo/VPAssistant/product ,0.0.0

```
22:16:01 I : Start downloading list.txt...
22:16:04 I : Finished dowloading list.txt.
22:16:04 I : A64Dbg    remote/local version is 1.17.0/0.0.0, please install it.
22:16:04 I : ClangVMP  remote/local version is 1.1.0/0.0.0, please install it.
22:16:04 I : PhoneVMP  remote/local version is 1.0.0/0.0.0, please install it.
22:16:04 I : UraniumVM remote/local version is 20240405.1/0.0.0, please install it.
22:16:04 I : UrPacker  remote/local version is 1.0.0/0.0.0, please install it.
22:16:04 I : VMPStudio remote/local version is 1.10.0/0.0.0, please install it.
22:16:04 I : VP Assistant v1.1.1 (Apr 12 2024), current host hwid: mac-arm64-01ac2ad20eeca7b90f408d6be2275192 .
```



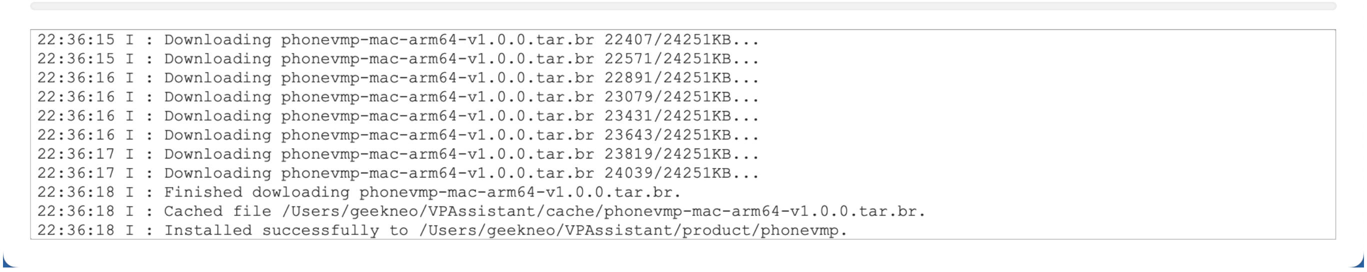
Hi,dear 01ac2ad20eeca7b90f408d6be2275192:

Welcome to vpand.com maintenance tool.

PhoneVMP is a LLVM based binary file (MachO/ELF/PE) assembly instruction (x86/x86_64/arm/arm64) virtualized protection tool.

The installation without a valid license is a trial version for binary code virtualized protection test,the result cannot run.
If you want to apply it to your real products,please contact us to buy a license.

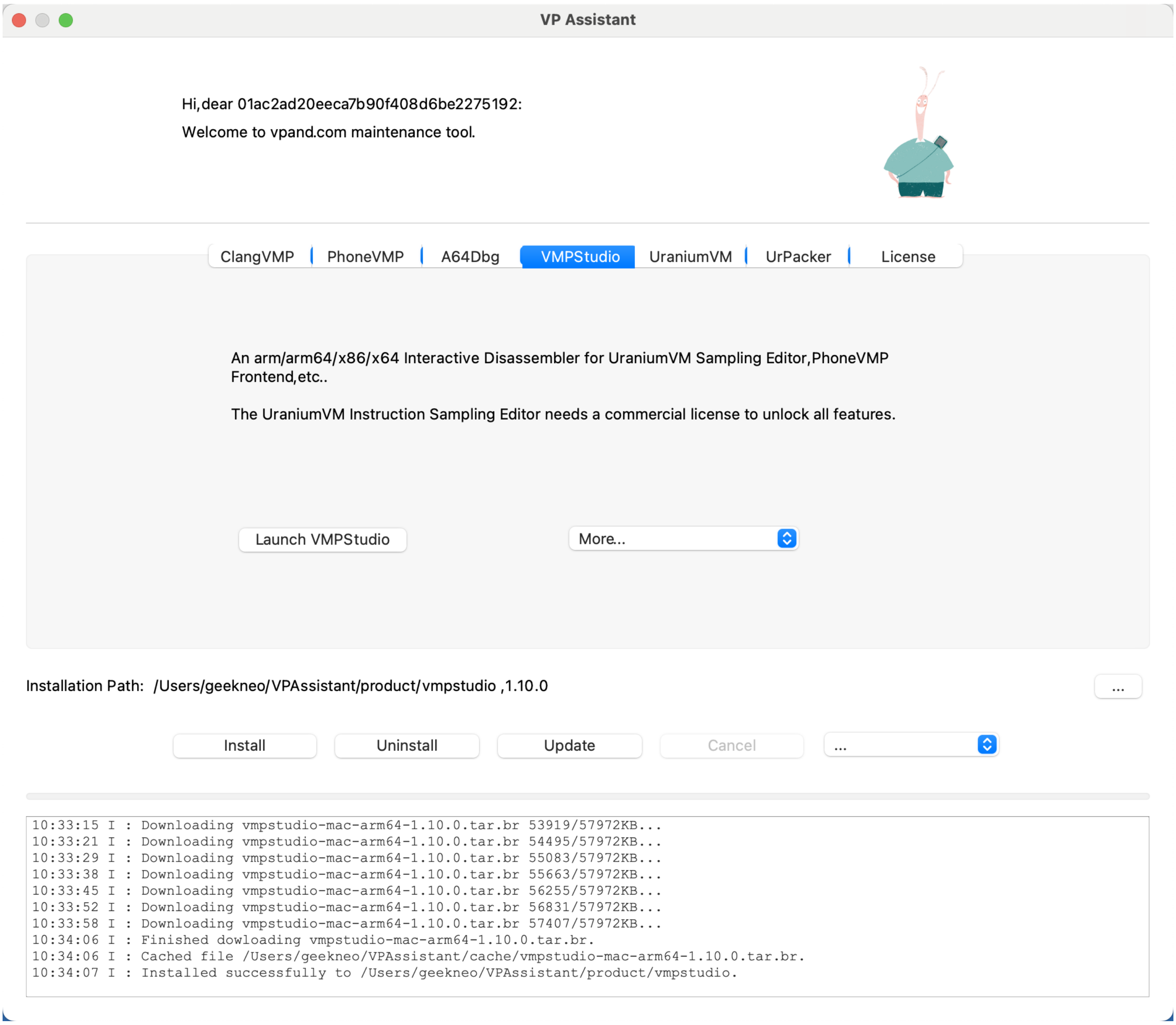
Installation Path: /Users/geekneo/VPAssistant/product/phonevmp ,1.0.0



```
22:36:15 I : Downloading phonevmp-mac-arm64-v1.0.0.tar.br 22407/24251KB...
22:36:15 I : Downloading phonevmp-mac-arm64-v1.0.0.tar.br 22571/24251KB...
22:36:16 I : Downloading phonevmp-mac-arm64-v1.0.0.tar.br 22891/24251KB...
22:36:16 I : Downloading phonevmp-mac-arm64-v1.0.0.tar.br 23079/24251KB...
22:36:16 I : Downloading phonevmp-mac-arm64-v1.0.0.tar.br 23431/24251KB...
22:36:16 I : Downloading phonevmp-mac-arm64-v1.0.0.tar.br 23643/24251KB...
22:36:17 I : Downloading phonevmp-mac-arm64-v1.0.0.tar.br 23819/24251KB...
22:36:17 I : Downloading phonevmp-mac-arm64-v1.0.0.tar.br 24039/24251KB...
22:36:18 I : Finished dowloading phonevmp-mac-arm64-v1.0.0.tar.br.
22:36:18 I : Cached file /Users/geekneo/VPAssistant/cache/phonevmp-mac-arm64-v1.0.0.tar.br.
22:36:18 I : Installed successfully to /Users/geekneo/VPAssistant/product/phonevmp.
```

VMPStudio (optional)

The easiest way to apply PhoneVMP to target file is using the GUI program VMPStudio, it's optional but highly recommended.



Hi,dear 01ac2ad20eeca7b90f408d6be2275192:
Welcome to vpand.com maintenance tool.

An arm/arm64/x86/x64 Interactive Disassembler for UraniumVM Sampling Editor,PhoneVMP Frontend,etc..

The UraniumVM Instruction Sampling Editor needs a commercial license to unlock all features.

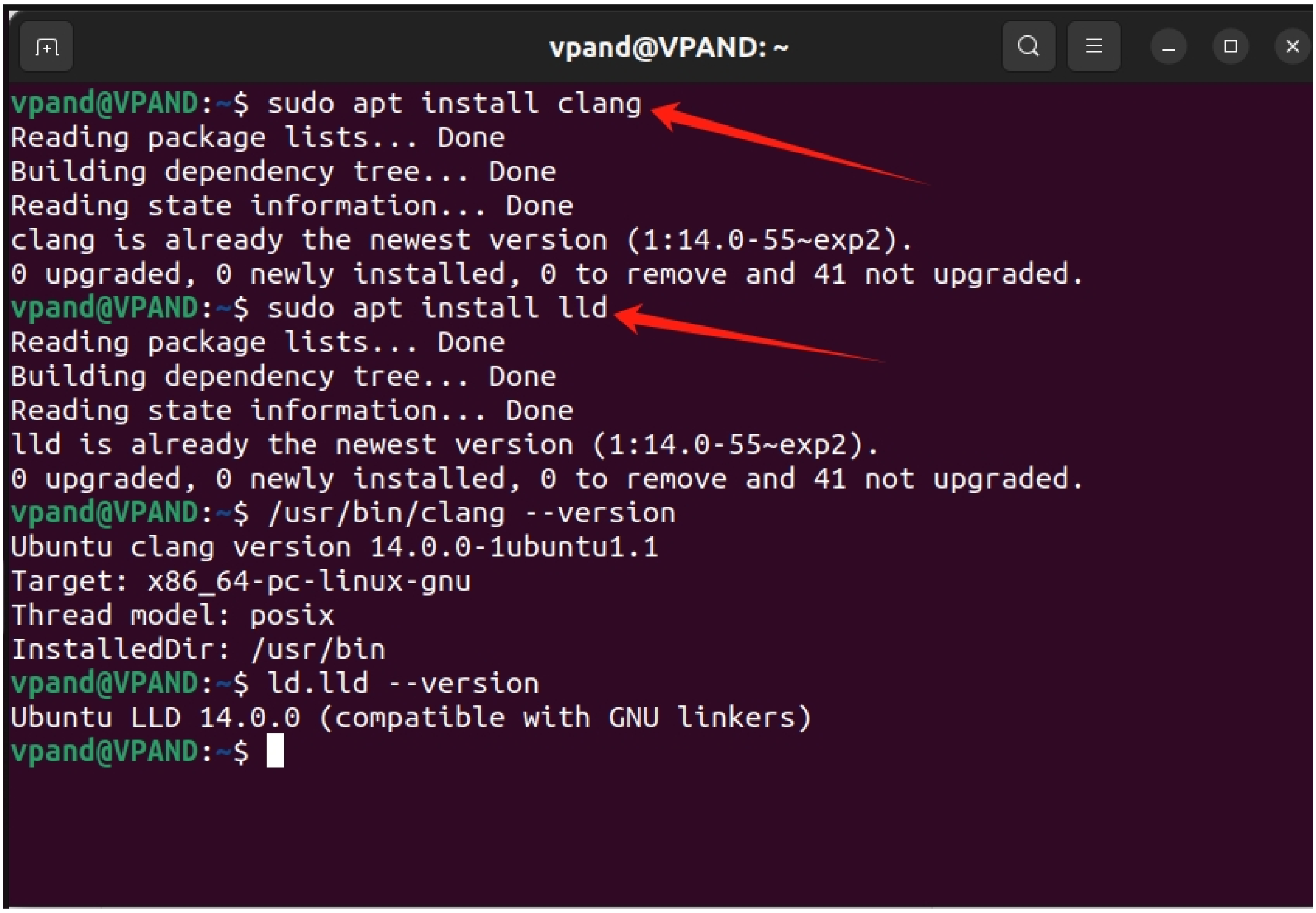
Installation Path: /Users/geekneo/VPAssistant/product/vmpstudio ,1.10.0

```
10:33:15 I : Downloading vmpstudio-mac-arm64-1.10.0.tar.br 53919/57972KB...
10:33:21 I : Downloading vmpstudio-mac-arm64-1.10.0.tar.br 54495/57972KB...
10:33:29 I : Downloading vmpstudio-mac-arm64-1.10.0.tar.br 55083/57972KB...
10:33:38 I : Downloading vmpstudio-mac-arm64-1.10.0.tar.br 55663/57972KB...
10:33:45 I : Downloading vmpstudio-mac-arm64-1.10.0.tar.br 56255/57972KB...
10:33:52 I : Downloading vmpstudio-mac-arm64-1.10.0.tar.br 56831/57972KB...
10:33:58 I : Downloading vmpstudio-mac-arm64-1.10.0.tar.br 57407/57972KB...
10:34:06 I : Finished dowloading vmpstudio-mac-arm64-1.10.0.tar.br.
10:34:06 I : Cached file /Users/geekneo/VPAssistant/cache/vmpstudio-mac-arm64-1.10.0.tar.br.
10:34:07 I : Installed successfully to /Users/geekneo/VPAssistant/product/vmpstudio.
```

Linux clang/lld

Before you can apply PhoneVMP to target file on Linux, clang and lld should be installed as the following steps:

```
sudo apt install clang
sudo apt install lld
```



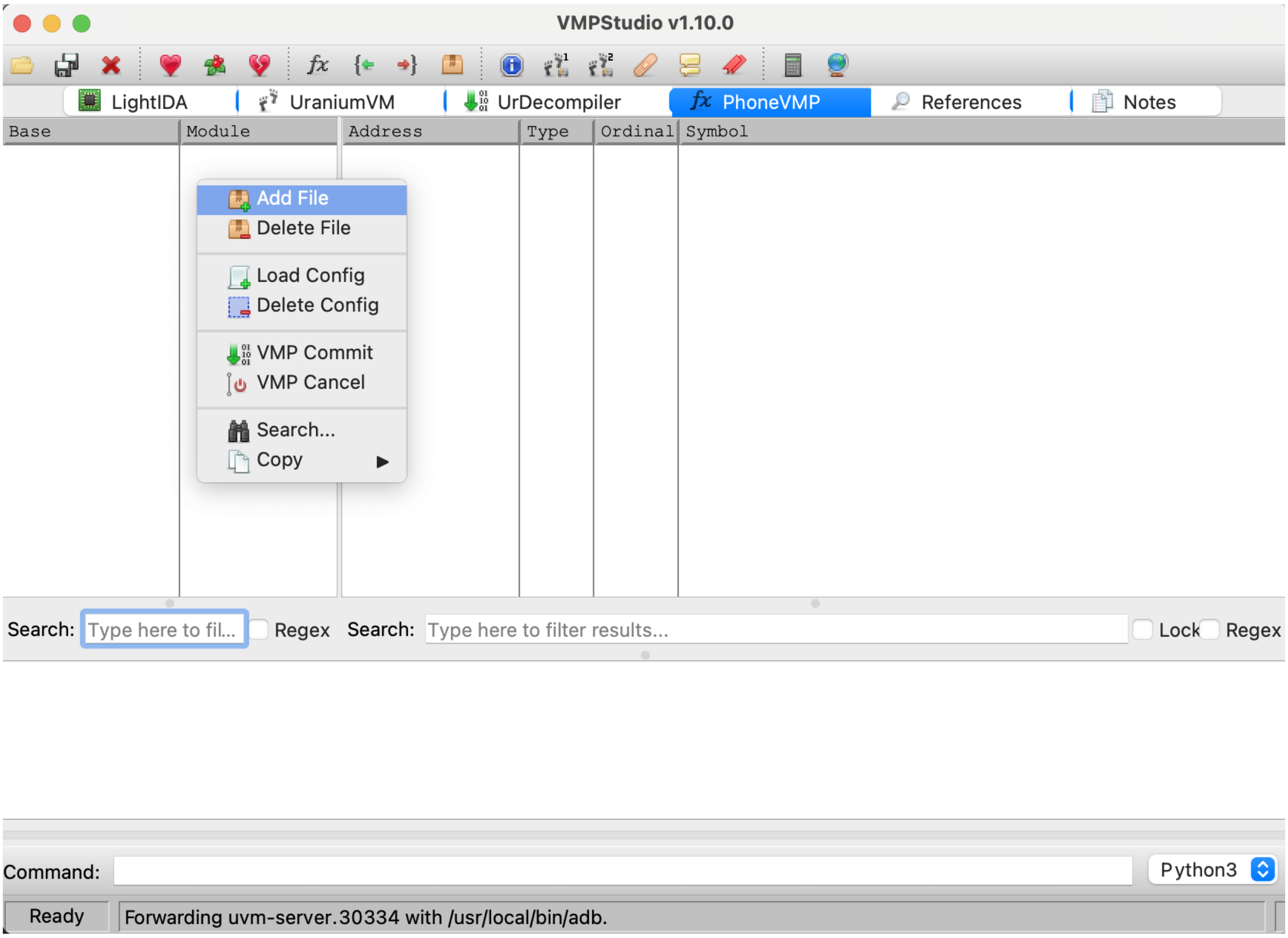
```
vpand@VPAND:~$ sudo apt install clang
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
clang is already the newest version (1:14.0-55~exp2).
0 upgraded, 0 newly installed, 0 to remove and 41 not upgraded.
vpand@VPAND:~$ sudo apt install lld
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
lld is already the newest version (1:14.0-55~exp2).
0 upgraded, 0 newly installed, 0 to remove and 41 not upgraded.
vpand@VPAND:~$ /usr/bin/clang --version
Ubuntu clang version 14.0.0-1ubuntu1.1
Target: x86_64-pc-linux-gnu
Thread model: posix
InstalledDir: /usr/bin
vpand@VPAND:~$ ld.lld --version
Ubuntu LLD 14.0.0 (compatible with GNU linkers)
vpand@VPAND:~$ 
```

VMP Configuration

As the GUI VMPStudio and CLI phonevmp on different platform(like Windows, Linux and macOS) is **absolutely the same**, unless some feature is just available on that specified platform, otherwise all the generic feature screenshots are from macOS.

VMPStudio

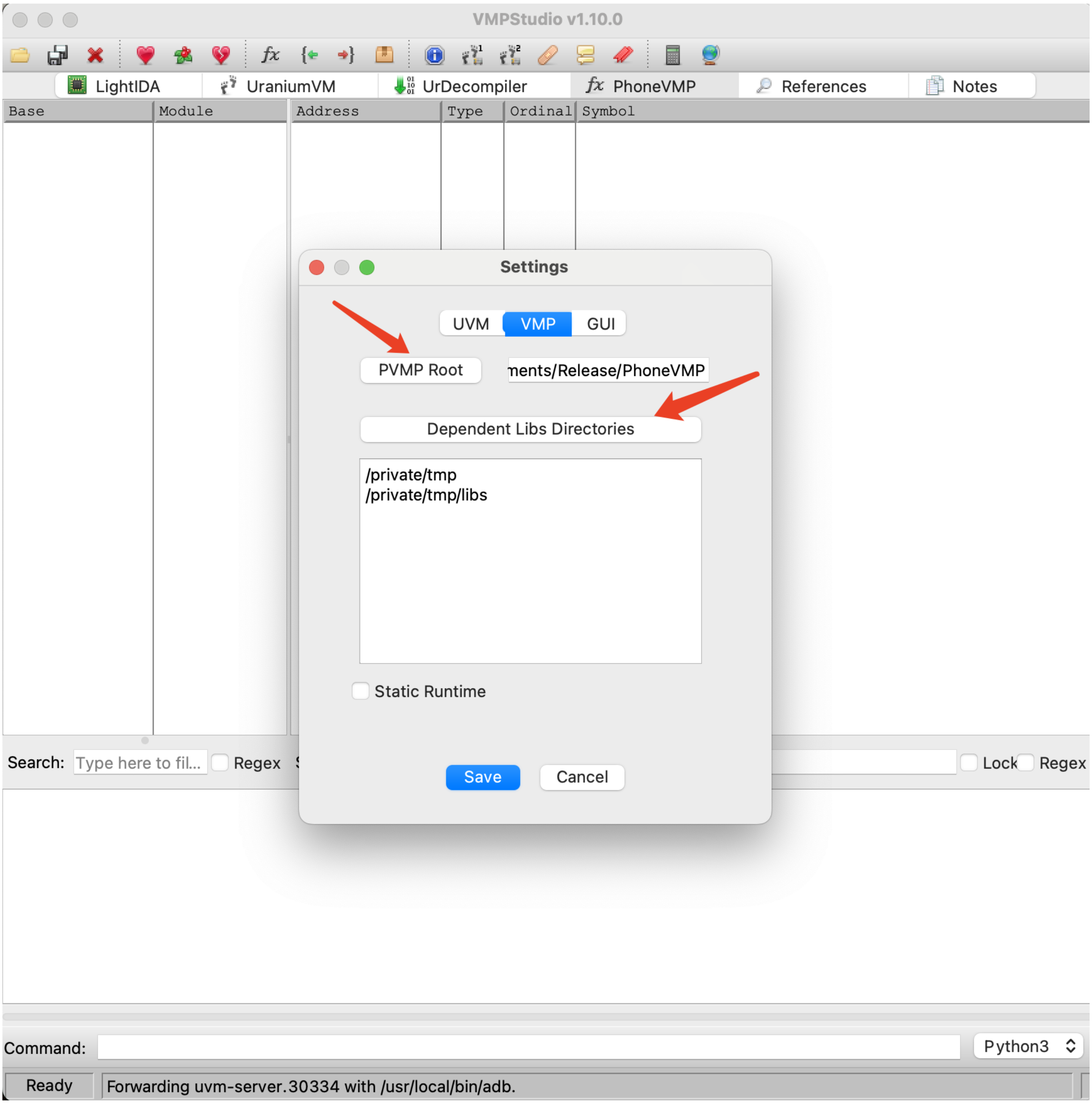
GUI mode is very convenient for PhoneVMP encryption of binaries, and it can also be used as a CLI mode configuration file generator, especially when PhoneVMP encryption of binaries is performed for the first time. All the features are integrated in the VMPStudio PhoneVMP tab widget:



Forwarding uvm-server.30334 with /usr/local/bin/adb.

- **Add File**: Add Macho/ELF/PE file to be virtualized protected.
- **Load Config**: Load an existing json configuration file to apply for current file.
- **VMP Commit**: Execute CLI phonevmp with current configuration.

But, before acting the real work, we have to specify the PhoneVMP product root directory for VMPStudio(don't forget apply the **Save** when you changed any configuration):



ments/Release/PhoneVMP

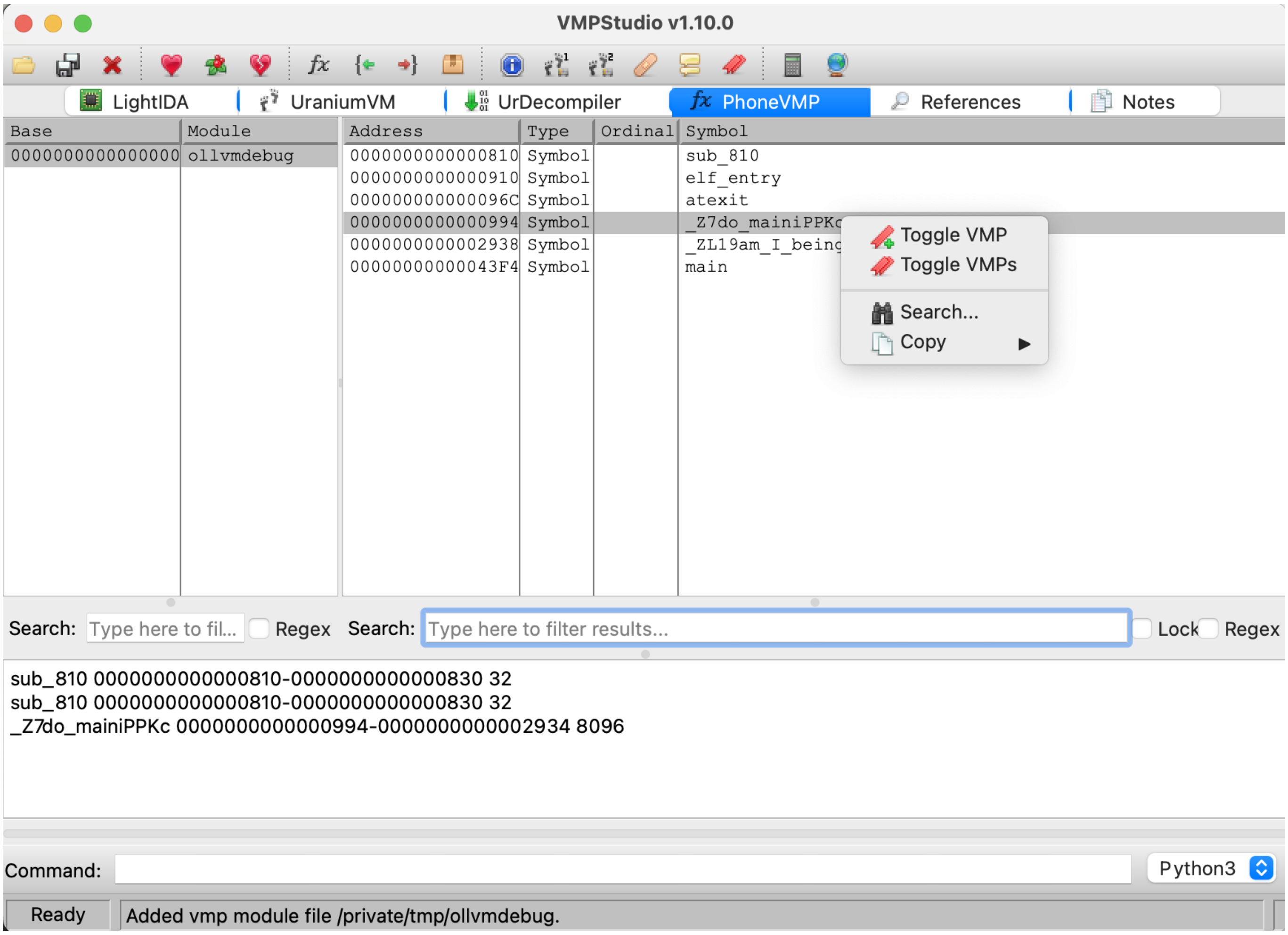
/private/tmp
/private/tmp/libs

□ Static Runtime

Forwarding uvm-server.30334 with /usr/local/bin/adb.

- **PVMP Root**: Select PhoneVMP product root directory.
- **Dependent Libs Directories**: Where to search the dependent library files when linking the final result by **UraniumPacker** after finishing VMP encoding process by **PhoneVMP**.

After you load a binary file with **Add File** menu, the right symbol window will list all the parsed functions:



Base	Module
0000000000000000	ollvmdebug

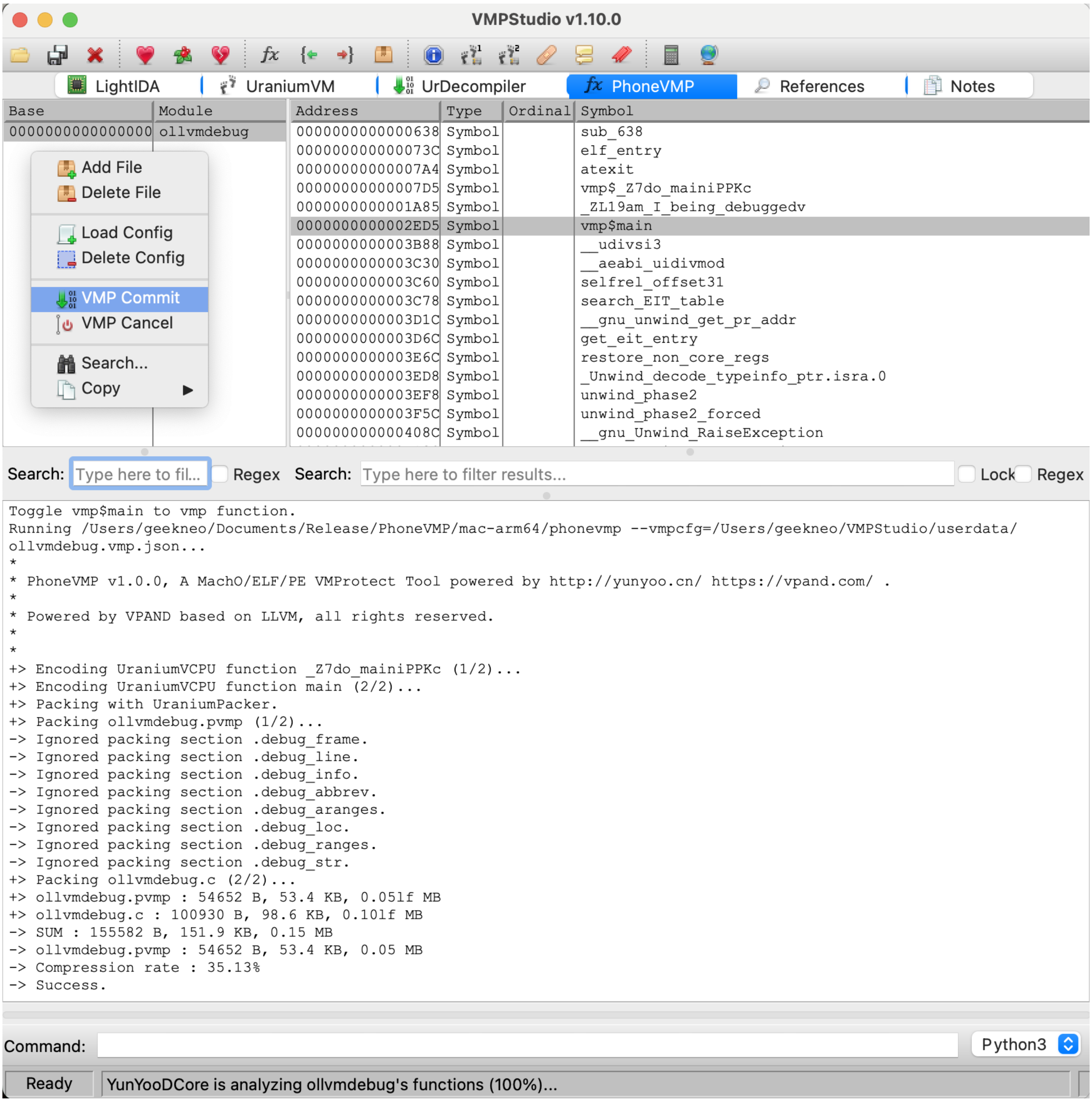
Address	Type	Ordinal	Symbol
0000000000000810	Symbol		sub_810
0000000000000910	Symbol		elf_entry
000000000000096C	Symbol		atexit
0000000000000994	Symbol		_Z7do_mainiPPKc
0000000000002938	Symbol		_ZL19am_I_being
00000000000043F4	Symbol		main

```
sub_810 0000000000000810-0000000000000830 32
sub_810 0000000000000810-0000000000000830 32
_Z7do_mainiPPKc 0000000000000994-0000000000002934 8096
```

Added vmp module file /private/tmp/ollvmdebug.

- **Toggle VMP**: Turn VMP on or off for current function.
- **Toggle VMPs**: Turn VMP on or off for current multiple selected functions.

When you finish the VMP configuration on all of the functions you want to apply virtualized protection, execute the menu **VMP Commit**:



Base	Module
0000000000000000	ollvmdebug

Address	Type	Ordinal	Symbol
0000000000000638	Symbol		sub_638
000000000000073C	Symbol		elf_entry
00000000000007A4	Symbol		atexit
00000000000007D5	Symbol		vmp$_Z7do_mainiPPKc
0000000000001A85	Symbol		_ZL19am_I_being_debuggedv
0000000000002ED5	Symbol		vmp$main
0000000000003B88	Symbol		__udivsi3
0000000000003C30	Symbol		__aeabi_uidivmod
0000000000003C60	Symbol		selfrel_offset31
0000000000003C78	Symbol		search_EIT_table
0000000000003D1C	Symbol		__gnu_unwind_get_pr_addr
0000000000003D6C	Symbol		get_eit_entry
0000000000003E6C	Symbol		restore_non_core_regs
0000000000003ED8	Symbol		_Unwind_decode_typeinfo_ptr.isra.0
0000000000003EF8	Symbol		unwind_phase2
0000000000003F5C	Symbol		unwind_phase2_forced
000000000000408C	Symbol		__gnu_Unwind_RaiseException

```
Toggle vmp$main to vmp function.
Running /Users/geekneo/Documents/Release/PhoneVMP/mac-arm64/phonevmp --vmpcfg=/Users/geekneo/VMPStudio/userdata/
ollvmdebug.vmp.json...
*
* PhoneVMP v1.0.0, A MachO/ELF/PE VMProtect Tool powered by http://yunyoo.cn/ https://vpand.com/ .
*
* Powered by VPAND based on LLVM, all rights reserved.
*
*
+> Encoding UraniumVCPU function _Z7do_mainiPPKc (1/2)...
+> Encoding UraniumVCPU function main (2/2)...
+> Packing with UraniumPacker.
+> Packing ollvmdebug.pvmp (1/2)...
-> Ignored packing section .debug_frame.
-> Ignored packing section .debug_line.
-> Ignored packing section .debug_info.
-> Ignored packing section .debug_abbrev.
-> Ignored packing section .debug_aranges.
-> Ignored packing section .debug_loc.
-> Ignored packing section .debug_ranges.
-> Ignored packing section .debug_str.
+> Packing ollvmdebug.c (2/2)...
+> ollvmdebug.pvmp : 54652 B, 53.4 KB, 0.051f MB
+> ollvmdebug.c : 100930 B, 98.6 KB, 0.101f MB
-> SUM : 155582 B, 151.9 KB, 0.15 MB
-> ollvmdebug.pvmp : 54652 B, 53.4 KB, 0.05 MB
-> Compression rate : 35.13%
-> Success.
```

YunYooDCore is analyzing ollvmdebug's functions (100%)...

Done. The **vmpcfg** and **pvmp** file are both ready for you.

CLI with JSON

CLI command line mode can be easily integrated into the existing packaging process, and its command line input parameters are as follows:

```
PhoneVMP@vpand.com mac-arm64 % ./phonevmp --help
OVERVIEW: PhoneVMP v1.0.0, A MachO/ELF/PE VMProtect Tool powered by
http://yunyoo.cn/ https://vpand.com/ .

USAGE: phonevmp [options] <input file>

OPTIONS:
```

```
Generic Options:

  --help              - Display available options (--help-hidden for more)
  --help-list         - Display list of available options (--help-list-hidden for more)
  --version           - Display the version of this program

PhoneVMP Options:

  --vmpcfg=<string> - PhoneVMP configuration json file
```

There is only one command line input parameter **--vmpcfg**, which accepts a json configuration file as the following format:

```
{
    "file": "/path/to/file",
    "level": "usrdef",
    "libdirs": [
        "/path/to/libs"
    ],
    "symbols": [
        "symbol_name"
    ]
}
```

- **file**: full path of the binary file which will be applied to PhoneVMP.
- **level**: always be **usrdef**.
- **libdirs**: list of library directories which contain the dependencies of the input file, it's used to search the dependent library files when linking the final result by **UraniumPacker**.
- **symbols**: list of function names which will be virtualized by PhoneVMP.

Here's a simple PhoneVMP json configuration:

```
{
    "file": "/private/tmp/ollvmdebug",
    "level": "usrdef",
    "libdirs": [
        "/private/tmp",
        "/private/tmp/libs"
    ],
    "symbols": [
        "_Z7do_mainiPPKc",
        "main"
    ]
}
```

```
PhoneVMP@vpand.com mac-arm64 % ./phonevmp --
vmpcfg=/.../ollvmdebug.vmp.json
*
* PhoneVMP v1.0.0, A MachO/ELF/PE VMProtect Tool powered by
http://yunyoo.cn/ https://vpand.com/ .
*
* Powered by VPAND based on LLVM, all rights reserved.
*
*
+> Encoding UraniumVCPU function _Z7do_mainiPPKc (1/1)...
+> Packing with UraniumPacker.
+> Packing ollvmdebug.pvmp (1/2)...
-> Ignored packing section .debug_frame.
-> Ignored packing section .debug_line.
-> Ignored packing section .debug_info.
-> Ignored packing section .debug_abbrev.
-> Ignored packing section .debug_aranges.
-> Ignored packing section .debug_loc.
-> Ignored packing section .debug_ranges.
-> Ignored packing section .debug_str.
+> Packing ollvmdebug.c (2/2)...
+> ollvmdebug.pvmp : 41308 B, 40.3 KB, 0.04lf MB
+> ollvmdebug.c : 61218 B, 59.8 KB, 0.06lf MB
-> SUM : 102526 B, 100.1 KB, 0.10 MB
-> ollvmdebug.pvmp : 41308 B, 40.3 KB, 0.04 MB
-> Compression rate : 40.29%
-> Success.
```

Raw vs PhoneVMP

Here's the simple C/C++ code, let's see how PhoneVMP is going to virtualized protect the do_main function.

```
static bool am_I_being_debugged(void) {
  FILE *fp = fopen("/proc/self/status", "r");
  char status[128];
  size_t rd = fread(status, 1, sizeof(status) - 1, fp);
  status[rd] = 0;
  fclose(fp);
  return strstr(status, "TracerPid:\t0") == nullptr;
}

int __attribute__((noinline)) do_main(int argc, const char *argv[]) {
  if (!getenv("NO_ANTIDBG")) {
    anti_debug();
  }

  // modify register to continue
  for (int i = 0; i < argc; i++) {
    printf(
      "You should change the register to quit this infinite loop %d.\n"
      "I'm being debugged(%s).\n",
      i,
      am_I_being_debugged() ? "true" : "false");
    sleep(2);
  }

  // what a nice day...
  puts("Have fun with A64Dbg UraniumVM virtualization debug mode~");
  return 0;
}
```

Raw

After compiled with clang compiler, we'll get the following assembly function:

```
do_main:0004 ; int do_main(int argc, const unsigned __int8 **argv)
do_main:0004                 EXPORT _Z7do_mainiPPKc
do_main:0004 _Z7do_mainiPPKc                         ; CODE XREF:
main+24↓p
do_main:0004
do_main:0004 ; __unwind {
do_main:0004                 SUB             SP, SP, #0xF0
do_main:0008                 STP             X29, X30, [SP,#0x90+var_s0]
do_main:000C                 STP             X28, X27, [SP,#0x90+var_s10]
do_main:0010                 STP             X26, X25, [SP,#0x90+var_s20]
do_main:0014                 STP             X24, X23, [SP,#0x90+var_s30]
do_main:0018                 STP             X22, X21, [SP,#0x90+var_s40]
do_main:001C                 STP             X20, X19, [SP,#0x90+var_s50]
do_main:0020                 ADD             X29, SP, #0x90
do_main:0024                 MRS             X8, #3, c13, c0, #2
do_main:0028 argc = X19                              ; int
do_main:0028                 MOV             W19, W0
do_main:002C                 STR             X8, [SP,#0x90+var_90]
do_main:0030                 ADRP            X0, #byte_180@PAGE
do_main:0034                 ADD             X0, X0, #byte_180@PAGEOFF ;
name
do_main:0038                 LDR             X8, [X8,#0x28]
do_main:003C                 STUR            X8, [X29,#var_8]
do_main:0040                 BL              getenv
do_main:0044                 CMP             W19, #1
do_main:0048                 B.LT            loc_E8
do_main:004C                 MOV             W20, WZR
do_main:0050                 ADD             X27, SP, #0x90+ptr
do_main:0054                 ADRP            X21, #byte_227@PAGE
do_main:0058                 ADD             X21, X21, #byte_227@PAGEOFF
do_main:005C                 ADRP            X22, #byte_239@PAGE
do_main:0060                 ADD             X22, X22, #byte_239@PAGEOFF
do_main:0064                 ADRP            X23, #byte_23B@PAGE
do_main:0068                 ADD             X23, X23, #byte_23B@PAGEOFF
do_main:006C                 ADRP            X28, #byte_1E7@PAGE
```

```
do_main:0070                 ADD             X28, X28, #byte_1E7@PAGEOFF
do_main:0074                 ADRP            X26, #byte_1E2@PAGE
do_main:0078                 ADD             X26, X26, #byte_1E2@PAGEOFF
do_main:007C                 ADRP            X24, #byte_18B@PAGE
do_main:0080 i = X20                                 ; int
do_main:0080                 ADD             X24, X24, #byte_18B@PAGEOFF
do_main:0084
do_main:0084 loc_84                                  ; CODE XREF:
do_main(int,char const**)+E0↓j
do_main:0084                 MOV             X0, X21 ; filename
do_main:0088                 MOV             X1, X22 ; modes
do_main:008C                 BL              fopen
do_main:0090                 MOV             X25, X0
do_main:0094                 ADD             X0, SP, #0x90+ptr ; ptr
do_main:0098                 MOV             W1, #1  ; size
do_main:009C                 MOV             W2, #0x7F ; n
do_main:00A0                 MOV             X3, X25 ; stream
do_main:00A4                 BL              fread
do_main:00A8                 STRB            WZR, [X27,X0]
do_main:00AC                 MOV             X0, X25 ; stream
do_main:00B0                 BL              fclose
do_main:00B4                 ADD             X0, SP, #0x90+ptr ; haystack
do_main:00B8                 MOV             X1, X23 ; needle
do_main:00BC                 BL              strstr
do_main:00C0                 CMP             X0, #0
do_main:00C4                 MOV             X0, X24 ; format
...
```

If we decompile it in IDA or any other decompiler tool, we'll get a perfect pseudo C source code implementation like this, nearly the original source code:

```
__int64 __fastcall do_main(int argc, const unsigned __int8 **argv)
{
  int v2; // w19
  unsigned int v3; // w20
  FILE *v4; // x25
  const unsigned __int8 *v5; // x2
  __int64 result; // x0
  unsigned __int64 v7; // [xsp+0h] [xbp-90h]
  char ptr[128]; // [xsp+8h] [xbp-88h]
  __int64 v9; // [xsp+88h] [xbp-8h]

  v2 = argc;
  v7 = _ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2));
  v9 = *(_QWORD *)(v7 + 40);
  getenv((const char *)byte_180);
  if ( v2 >= 1 )
  {
    v3 = 0;
    do
    {
      v4 = fopen((const char *)byte_227, (const char *)byte_239);
      ptr[fread(ptr, 1uLL, 0x7FuLL, v4)] = 0;
      fclose(v4);
      if ( strstr(ptr, (const char *)byte_23B) )
        v5 = byte_1E7;
      else
        v5 = byte_1E2;
      printf((const char *)byte_18B, v3, v5);
      sleep(2u);
      ++v3;
    }
    while ( v2 != v3 );
  }
  result = puts((const char *)byte_1ED);
  if ( *(_QWORD *)(v7 + 40) == v9 )
    result = 0LL;
  return result;
}
```

PhoneVMP

The virtualized protection with phonevmp tool, we'll get the following assembly function:

```
.text:0000000000034E28 ; int __cdecl _Z7do_mainiPPKc(int argc, const char **argv, const char **envp)
.text:0000000000034E28 _Z7do_mainiPPKc
; DATA XREF: .got:_Z7do_mainiPPKc_ptr↓o
.text:0000000000034E28 ; __unwind {
.text:0000000000034E28                 MOV             X2, X1
.text:0000000000034E2C                 MOV             W1, W0
.text:0000000000034E30                 ADRP            X0, #aAntidebugPvmp@PAGE ; "antidebug.pvmp"
.text:0000000000034E34                 ADD             X0, X0, #aAntidebugPvmp@PAGEOFF ; "antidebug.pvmp"
.text:0000000000034E38                 B               .upacker_main
```

If we decompile it in IDA or any other decompiler tool, we'll get a fixed pseudo C source code implementation like this, which will always keeps this kind of instruction sequence, entering UraniumPacker and PhoneVMP virtual machine runtime library:

```
int __cdecl do_main(int argc, const char **argv, const char **envp)
{
  return upacker_main("antidebug.pvmp", (unsigned int)argc, argv);
}
```

Oops, we can see nothing about this function, all of the raw instructions are encoded as UraniumPacker and PhoneVMP runtime data. **As so, reverse engineering will be extremely hard. This is the core value of**

PhoneVMP.

One more thing, when doing the real final packing, don't forget to pack one of the runtime library file into your product:

```
PhoneVMP@vpand.com phonevmp % find . -name "*PhoneVMP*"
./apple/ios/arm64/libPhoneVMP.dylib
./apple/mac/arm64/libPhoneVMP.dylib
./apple/mac/x86_64/libPhoneVMP.dylib
./android/armeabi-v7a/libPhoneVMP.so
./android/x86/libPhoneVMP.so
./android/arm64-v8a/libPhoneVMP.so
./android/x86_64/libPhoneVMP.so
```

Trial version

The default installation with a valid but restricted license is a trial version for **Android-arm/x86 virtualized protection test**, the result may **cannot run**. If you want to apply it to your real products and get technical support, please contact us to buy a license.

Licensed version

If you want to upgrade to a licensed version, here's the steps:

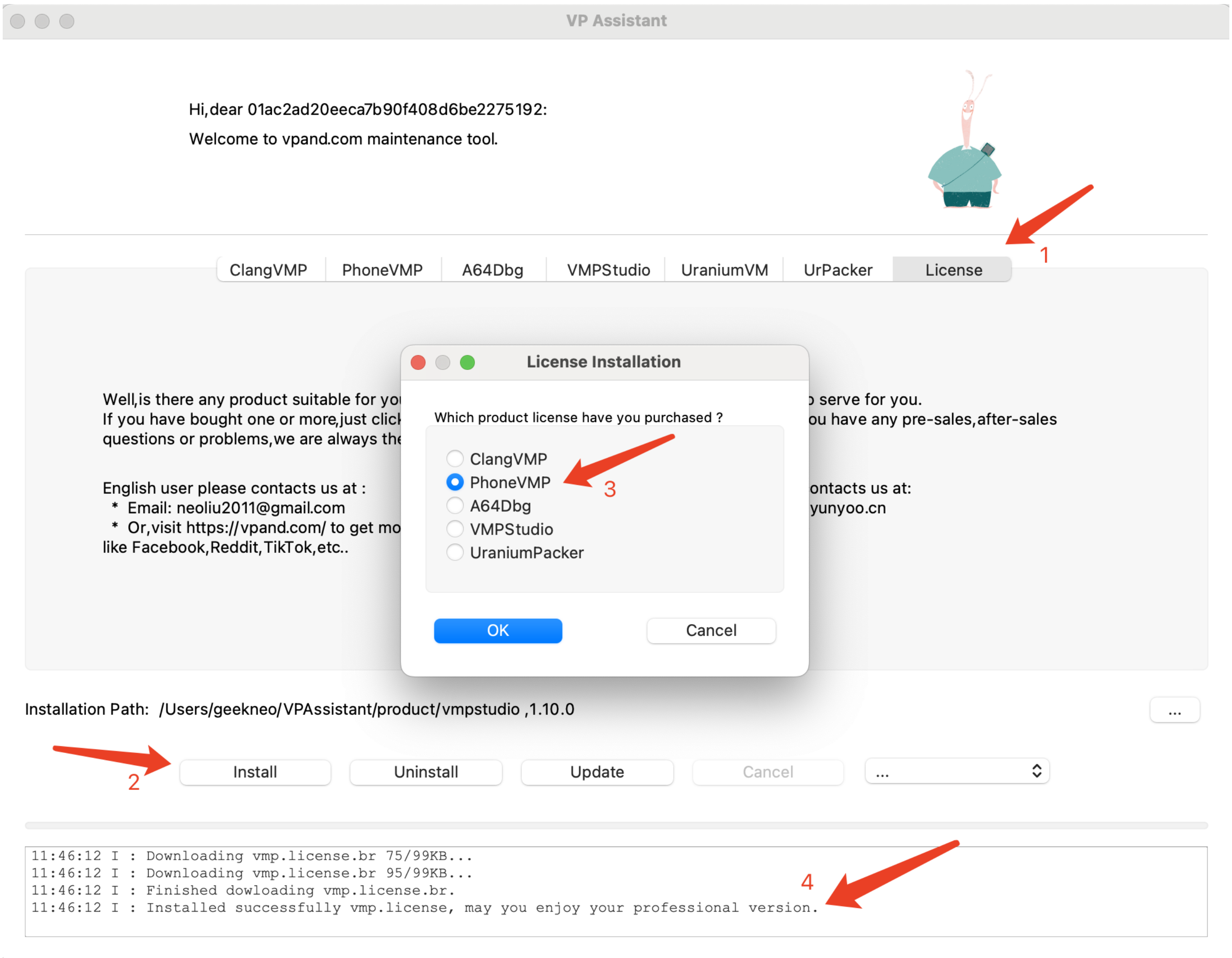
- 1. Contact us to pay for your license;

```
The price of PhoneVMP: 5000$/year/os-arch/license
```

- 2. Send your computer os-arch-hwid(copy from VPAssistant startup log) to us;

```
22:53:37 I : VP Assistant v1.1.2 (Apr 19 2024), current host hwid: mac-arm64-01ac2ad20eeca7b90f408d6be2275192 .
```

- 3. Switch to VPAssistant License tab widget, click Install button to download and install the license file;



Hi,dear 01ac2ad20eeca7b90f408d6be2275192:
Welcome to vpand.com maintenance tool.

Well,is there any product suitable for yo… serve for you.
If you have bought one or more,just click… ou have any pre-sales,after-sales
questions or problems,we are always the…

English user please contacts us at : … ontacts us at:
* Email: neoliu2011@gmail.com … yunyoo.cn
* Or,visit https://vpand.com/ to get mo…
like Facebook,Reddit,TikTok,etc..

Which product license have you purchased ?

○ ClangVMP
● PhoneVMP
○ A64Dbg
○ VMPStudio
○ UraniumPacker

Installation Path: /Users/geekneo/VPAssistant/product/vmpstudio ,1.10.0

```
11:46:12 I : Downloading vmp.license.br 75/99KB...
11:46:12 I : Downloading vmp.license.br 95/99KB...
11:46:12 I : Finished dowloading vmp.license.br.
11:46:12 I : Installed successfully vmp.license, may you enjoy your professional version.
```

Contact us

Email

If you have any questions or problems on our products or services, feel free to contact us via email at anytime:

- **neoliu2011@gmail.com**

We-Media

Till now, we-media is our main operation method, you can also contact us via the following platforms:

- Facebook
- YouTube
- Reddit
- X
- Instagram

FAQ

Does PhoneVMP support Windows or Linux?

In fact, we have finished the development of Windows-PE and Linux-ELF supporting. But we think ClangVMP is much better for desktop platform, so we haven't released it to current PhoneVMP version. If you do need this kind of feature, please let us know.

What's the price of PhoneVMP license ? Could the license be purchased by quarter or month?

The price of PhoneVMP is **5000$/year/os-arch/license**, os means **Andoid/iOS/macOS**, arch means **x86/x86_64/arm/arm64**.You can visit vpand.com to check the latest price policy, all of our products are sold at expressly marked price, and the license term is by year. So till now, quarterly or monthly purchasing is not supported.

When the license is expired, can the code virtualized before run normally?

Yes, of course, the license is for PhoneVMP tool itself, it has no effect on the virtualized code.

Can I update the license to rebind a new computer?

In principle, this is not supported. Because all of our license are offline type, it'll never connect to our server to check the license status. So if a license is released, it can not be repealed. This is good for client, as many clients are worried about data leakage. In order to avoid data leakage, offline license is the one and only choice, because we'll never read 1 byte data for our server to check what we need.