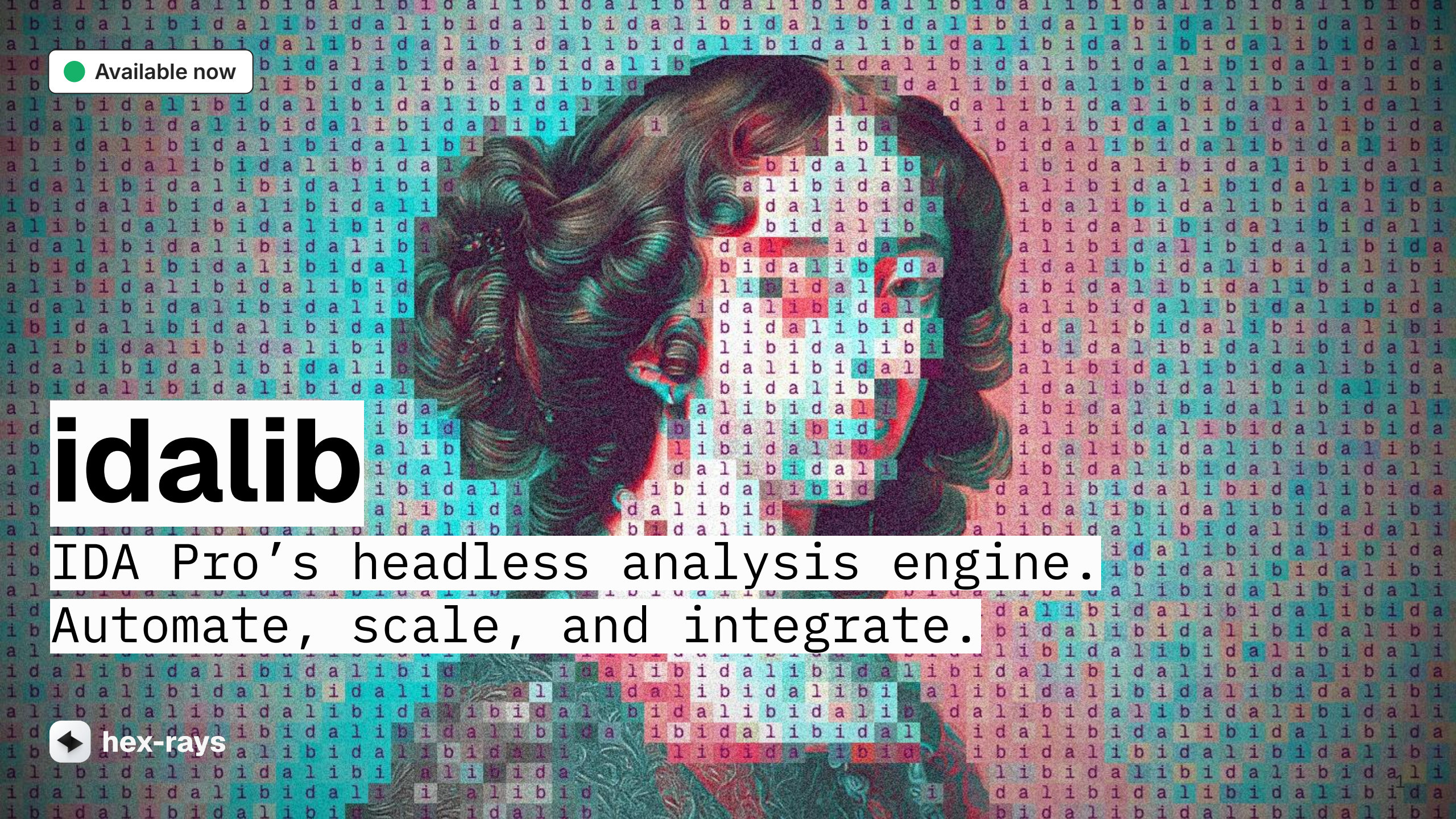
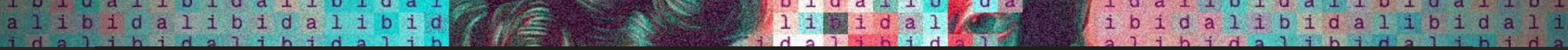

Available now
idalib
IDA Pro’s headless analysis engine.
Automate, scale, and integrate.
hex-rays



Claude Code Research preview

idawilli / Malware analysis with idalib

- New session
- Routines
- Customize

Recents

- Malware analysis with idalib
- FLAIR utilities download availability
- Recent CI failures triage
- Add Google Docs and Slides activity tr
- Index Obsidian vault files in daily note
- Explore data-driven test conversion fo
- Add snapshot testing for feature extra
- Reduce Crown job frequency to twice
- General coding session
- Design self-hosted web interface for P
- codemode chat
- Streamline plugin installation and distri

Try the Slack app Install

Willi · Hex-Rays

download https://github.com/mandiant/capa-testfiles/raw/refs/heads/master/Practical%20Malware%20Analysis%20Lab%2001-01.dll_ and then demonstrate you can analyze it with idalib, using the script

Initialized session ›

I'll use the idalib-analysis skill for this. Let me start by loading it and downloading the file.

Ran skill /idalib-analysis ›

Let me install IDA Pro and download the file in parallel.

Ran 2 commands

Ran Install IDA Pro idalib ›
Ran Download PMA Lab 01-01 DLL ›

The download worked. Let me find the install script — the env var wasn't expanded.

Ran Run IDA install script ›

IDA is installed and the file is downloaded. Let me write an analysis script.

Created a file, ran a command

Created analyze.py +35 -0 ›
Ran Run idalib analysis script

Bash

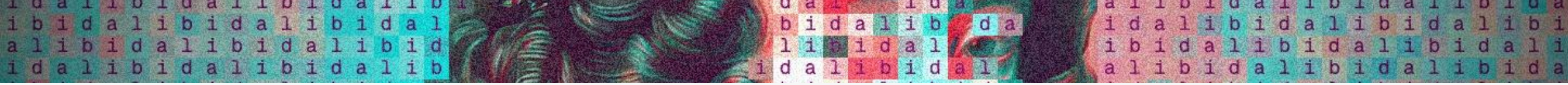
```
$ cd /tmp && python3 analyze.py 2>&1 | grep -v -E "^(Hex-Rays|IDA |Licensed|Copyright|.*flair|autoanalysis)"

=== Metadata ===
  architecture : metapc
  range        : 0x10001000 - 0x10027000
```

Claude Fable 5 is currently unavailable. Learn more

Type / for commands

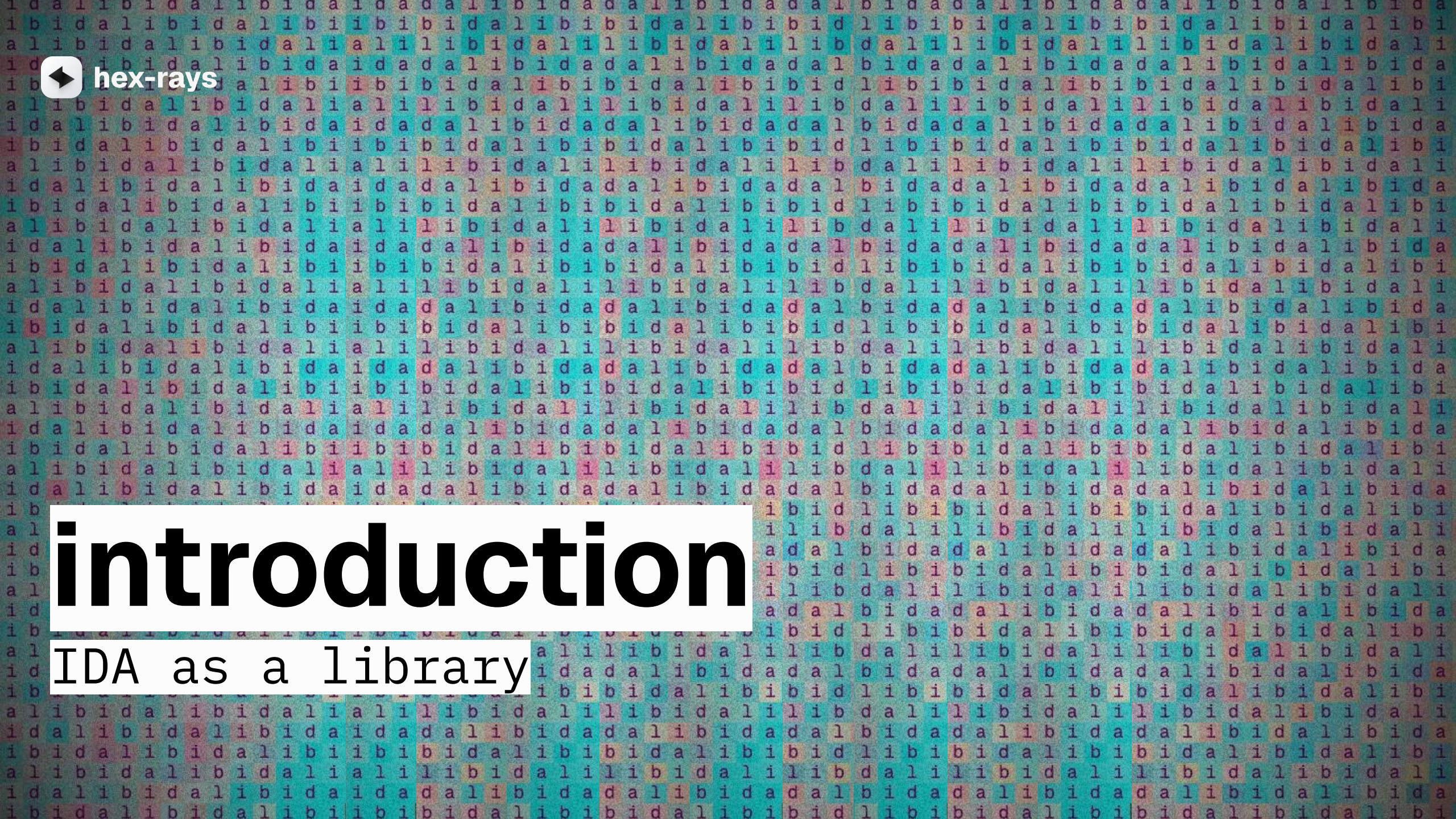
Accept edits

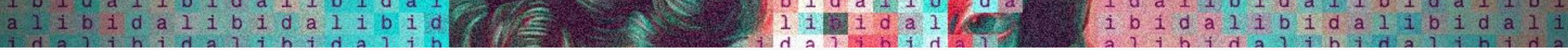
Opus 4.8 High



Agenda

- Intro
- Examples
- Installation
- Licensing
- Real World Tricks
- AI & MCP
- Q&A


hex-rays
introduction
IDA as a library



idalib

IDA Pro minus the Qt GUI

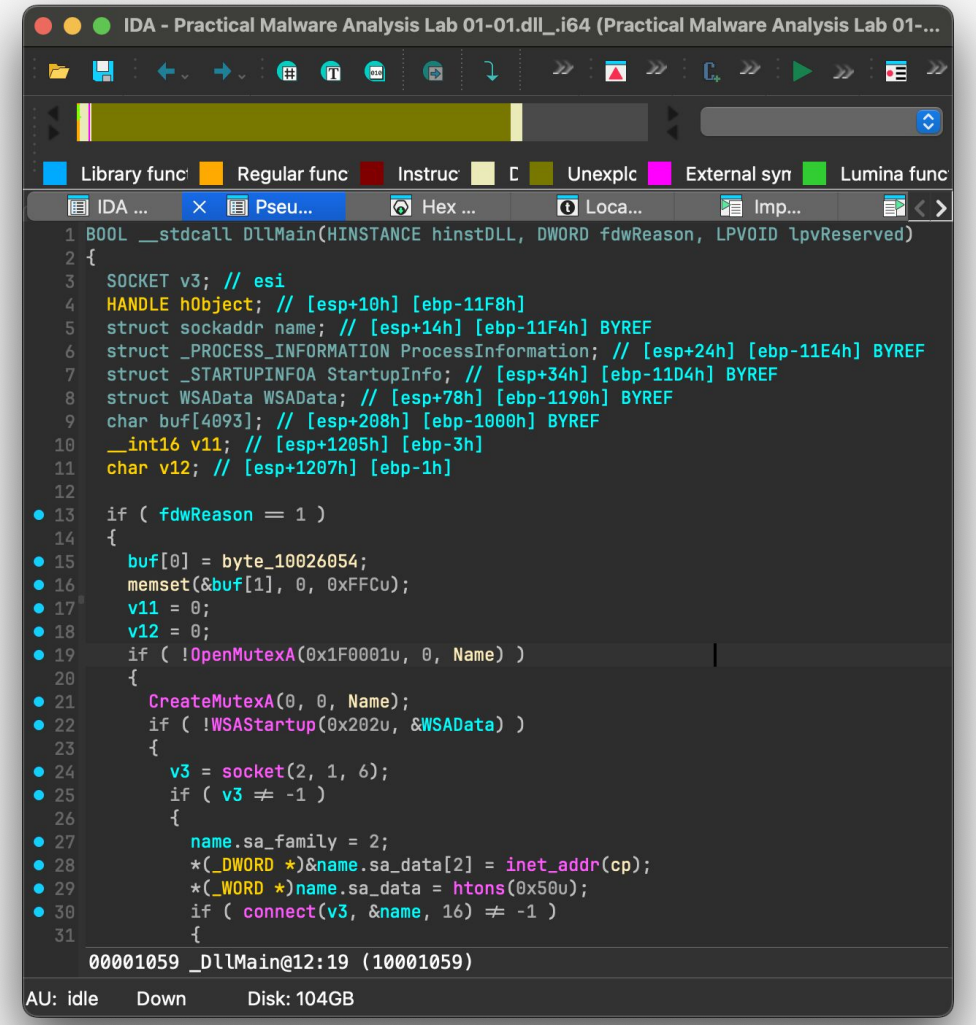
Use IDA as a library to automate, scale, and integrate binary analysis.

Disassembly, decompilation, names, cross references, strings, etc.

> …from Python script
>
> - for example: capa, idasql

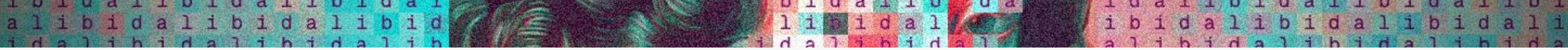
> …from C++ (or native) application
>
> - for example: 0xdea/Rhabdomancer vulnerability hunting

idalib: IDA Pro minus Qt GUI




```
❯ python foo.py Practical\ Malware\ Analysis\ Lab\ 01-01.dll_ | bat --language=cpp -P -
BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
{
  SOCKET v3; // esi
  HANDLE hObject; // [esp+10h] [ebp-11F8h]
  struct sockaddr name; // [esp+14h] [ebp-11F4h] BYREF
  struct _PROCESS_INFORMATION ProcessInformation; // [esp+24h] [ebp-11E4h] BYREF
  struct _STARTUPINFOA StartupInfo; // [esp+34h] [ebp-11D4h] BYREF
  struct WSAData WSAData; // [esp+78h] [ebp-1190h] BYREF
  char buf[4093]; // [esp+208h] [ebp-1000h] BYREF
  __int16 v11; // [esp+1205h] [ebp-3h]
  char v12; // [esp+1207h] [ebp-1h]

  if ( fdwReason == 1 )
  {
    buf[0] = byte_10026054;
    memset(&buf[1], 0, 0xFFCu);
    v11 = 0;
    v12 = 0;
    if ( !OpenMutexA(0x1F0001u, 0, Name) )
    {
      CreateMutexA(0, 0, Name);
      if ( !WSAStartup(0x202u, &WSAData) )
      {
        v3 = socket(2, 1, 6);
        if ( v3 != -1 )
        {
          name.sa_family = 2;
          *(_DWORD *)&name.sa_data[2] = inet_addr(cp);
          *(_WORD *)name.sa_data = htons(0x50u);
          if ( connect(v3, &name, 16) != -1 )
```

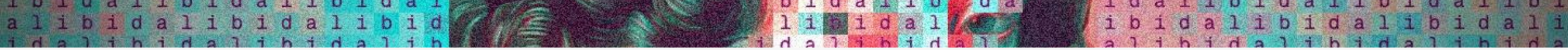
hex-rays



idalib: IDA Pro minus Qt GUI

```python
import sys

from ida_domain import Database
from ida_domain.database import IdaCommandOptions

ida_options = IdaCommandOptions(auto_analysis=True)
with Database.open(path=sys.argv[1], args=ida_options) as db:
    f: func_t | None = db.functions.get_function_by_name(name="_DllMain@12")
    assert f is not None
    for line in db.functions.get_pseudocode(func=f):
        print(line)
```

```
~/Downloads/2 via py v3.14.0 (.venv) ❄  took 11s
❯ python foo.py Practical\ Malware\ Analysis\ Lab\ 01-01.dll_ | bat --language=cpp -P -
BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
{
  SOCKET v3; // esi
  HANDLE hObject; // [esp+10h] [ebp-11F8h]
  struct sockaddr name; // [esp+14h] [ebp-11F4h] BYREF
  struct _PROCESS_INFORMATION ProcessInformation; // [esp+24h] [ebp-11E4h] BYREF
  struct _STARTUPINFOA StartupInfo; // [esp+34h] [ebp-11D4h] BYREF
  struct WSAData WSAData; // [esp+78h] [ebp-1190h] BYREF
  char buf[4093]; // [esp+208h] [ebp-1000h] BYREF
  __int16 v11; // [esp+1205h] [ebp-3h]
  char v12; // [esp+1207h] [ebp-1h]

  if ( fdwReason == 1 )
  {
    buf[0] = byte_10026054;
    memset(&buf[1], 0, 0xFFCu);
    v11 = 0;
    v12 = 0;
    if ( !OpenMutexA(0x1F0001u, 0, Name) )
    {
      CreateMutexA(0, 0, Name);
      if ( !WSAStartup(0x202u, &WSAData) )
      {
        v3 = socket(2, 1, 6);
        if ( v3 != -1 )
        {
          name.sa_family = 2;
          *(_DWORD *)&name.sa_data[2] = inet_addr(cp);
          *(_WORD *)name.sa_data = htons(0x50u);
          if ( connect(v3, &name, 16) != -1 )
```

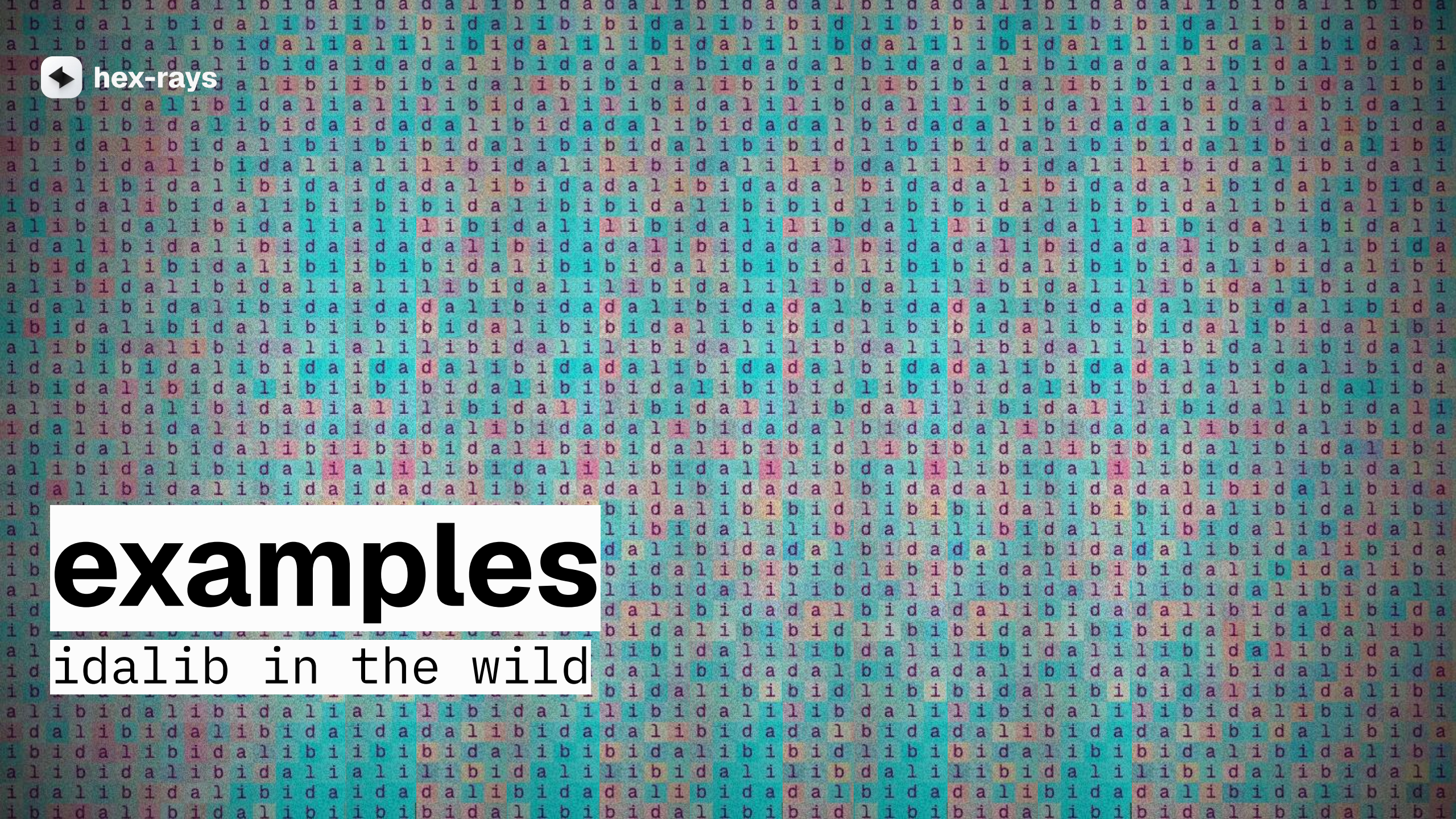


Domain API vs Python SDK

Idiomatic Python API for IDA

```python
from ida_domain import Database
from ida_domain.database import IdaCommandOptions

ida_options = IdaCommandOptions(auto_analysis=True)
with Database.open(sys.argv[1], ida_options) as db:
    f = db.functions.get_function_by_name("_DllMain@12")
    assert f is not None
    for line in db.functions.get_pseudocode(f):
        print(line)
```

```python
import idapro
import ida_auto
import ida_name
import ida_funcs
import ida_hexrays
import ida_lines
from ida_idaapi import BADADDR

idapro.enable_console_messages(False)
ret = idapro.open_database(sys.argv[1])
assert ret == 0
ida_auto.auto_wait()

func_ea = ida_name.get_name_ea(BADADDR, "_DllMain@12")
assert func_ea != BADADDR

func = ida_funcs.get_func(func_ea)
cfunc = ida_hexrays.decompile(func.start_ea)

for line in cfunc.get_pseudocode():
    print(ida_lines.tag_remove(line.line))

idapro.close_database()
```


hex-rays
examples
idalib in the wild

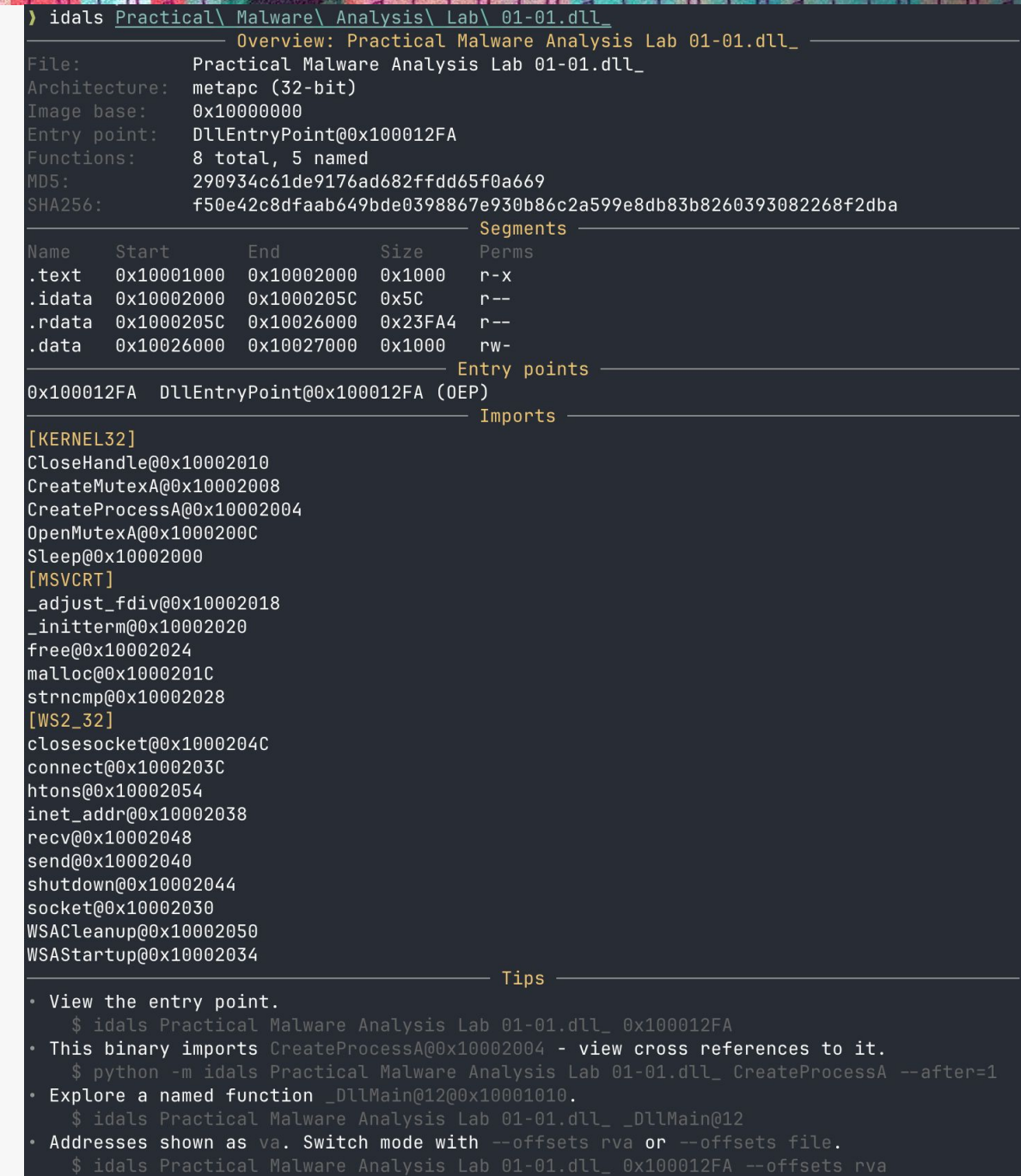
idals

personal CLI for IDA

```
idals /path/to/binary [<address or name>]
```

shows what IDA GUI shows at the address.

williballenthin/idawilli/idals



hex-rays

idals

personal CLI for IDA

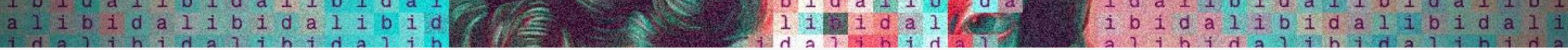
```
idals /path/to/binary [<address or name>]
```

shows what IDA GUI shows at the address.

[williballenthin/idawilli/idals]

hex-rays

```
❯ idals Practical\ Malware\ Analysis\ Lab\ 01-01.dll_ DllEntryPoint
; 0x100012FA
DllEntryPoint:
push    ebp  ; ⟵ target
mov     ebp, esp
push    ebx
mov     ebx, [ebp+hinstDLL]
push    esi
mov     esi, [ebp+fdwReason]
push    edi
mov     edi, [ebp+lpReserved]
test    esi, esi
jnz     short loc_10001316
cmp     dword_10026058, 0
jmp     short loc_1000133C
loc_10001316:
cmp     esi, 1
jz      short loc_10001320
cmp     esi, 2
jnz     short loc_10001342
; ...57 instructions skipped...
retn    0Ch
; 0x10001394
Pseudocode:
BOOL __stdcall DllEntryPoint(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpReserved)
{
  bool v4; // zf
  BOOL v6; // eax
  DWORD fdwReasona; // [esp+18h] [ebp+Ch]

  if ( fdwReason )
  {
    if ( fdwReason != 1 && fdwReason != 2 )
      goto LABEL_10;
    if ( dword_10026068 && !dword_10026068(hinstDLL, fdwReason, lpReserved) )
      return 0;
    v4 = _CRT_INIT(hinstDLL, fdwReason, lpReserved) == 0;
  }
  else
  {
    v4 = dword_10026058 == 0;
  }
  if ( v4 )
    return 0;
LABEL_10:
  v6 = DllMain(hinstDLL, fdwReason, lpReserved);
  fdwReasona = v6;
  if ( fdwReason != 1 )
  {
LABEL_13:
```



allthingsida/idasql

IDASQL is a SQL interface for IDA Pro databases

created by Elias Bachaalany

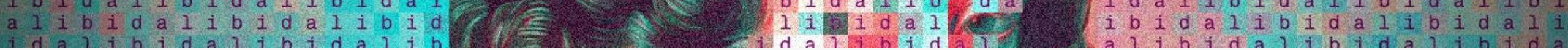
exposes 30+ virtual tables covering:

- functions
- cross-references
- strings
- types
- imports
- disassembly
- decompilation

```
$ idasql -s WerFaultTool.exe.i64 -q "SELECT * FROM funcs LIMIT 5"
Opening: WerFaultTool.exe.i64...
Database opened successfully.

+---------+-----------------------------------------------+------+--------+-------+
| address | name                                          | size | end_ea | flags |
+---------+-----------------------------------------------+------+--------+-------+
| 16      | WerFaultTool.AboutForm::.ctor                 | 13   | 29     | 4096  |
| 32      | WerFaultTool.AboutForm::Dispose               | 30   | 62     | 4096  |
| 64      | WerFaultTool.AboutForm::InitializeComponent   | 295  | 359    | 4096  |
| 400     | WerFaultTool.WerFaultGUI::.ctor               | 936  | 1336   | 4096  |
| 1344    | WerFaultTool.WerFaultGUI::CreateDynamicControls | 231  | 1575   | 4096  |
+---------+-----------------------------------------------+------+--------+-------+
5 row(s)
```

mandiant/capa/…/loader.py

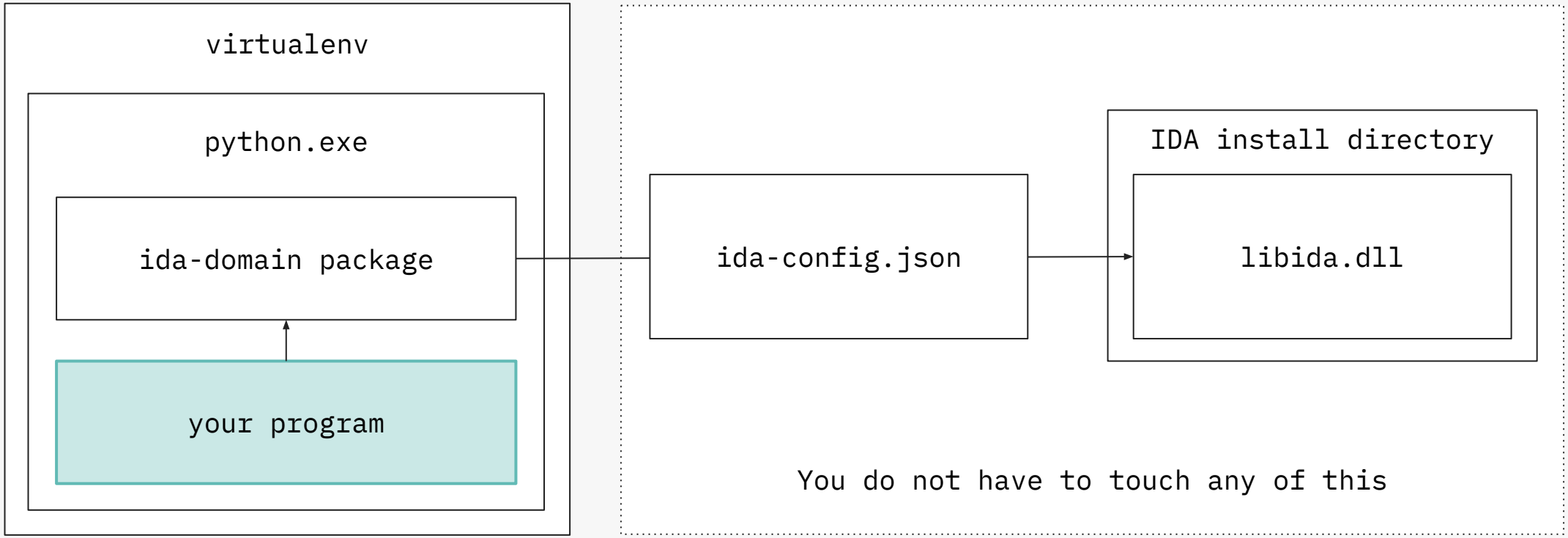
```
386         elif backend == BACKEND_IDA:
387             import capa.features.extractors.ida.idalib as idalib
388
389             if not idalib.is_idalib_installed():
390                 raise RuntimeError("idalib not available.")
391
392             import idapro
393             import ida_auto
394
395             import capa.features.extractors.ida.extractor
396
397             logger.debug("idalib: opening database...")
398             idapro.enable_console_messages(False)
399             with console.status("analyzing program...", spinner="dots"):
400                 # we set the primary and secondary Lumina servers to 0.0.0.0 to disable Lumina,
401                 # which sometimes provides bad names, including overwriting names from debug info.
402                 #
403                 # use -R to load resources, which can help us embedded PE files.
404                 #
405                 # return values from open_database:
406                 #   0 - Success
407                 #   2 - User cancelled or 32-64 bit conversion failed
408                 #   4 - Database initialization failed
409                 #   -1 - Generic errors (database already open, auto-analysis failed, etc.)
410                 #   -2 - User cancelled operation
411                 ret = idapro.open_database(
412                     str(input_path),
413                     run_auto_analysis=True,
414                     args="-Olumina:host=0.0.0.0 -Osecondary_lumina:host=0.0.0.0 -R",
415                 )
416                 if ret != 0:
417                     raise RuntimeError("failed to analyze input file")
418
419                 logger.debug("idalib: waiting for analysis...")
420                 ida_auto.auto_wait()
421                 logger.debug("idalib: opened database.")
422
423             return capa.features.extractors.ida.extractor.IdaFeatureExtractor()
```



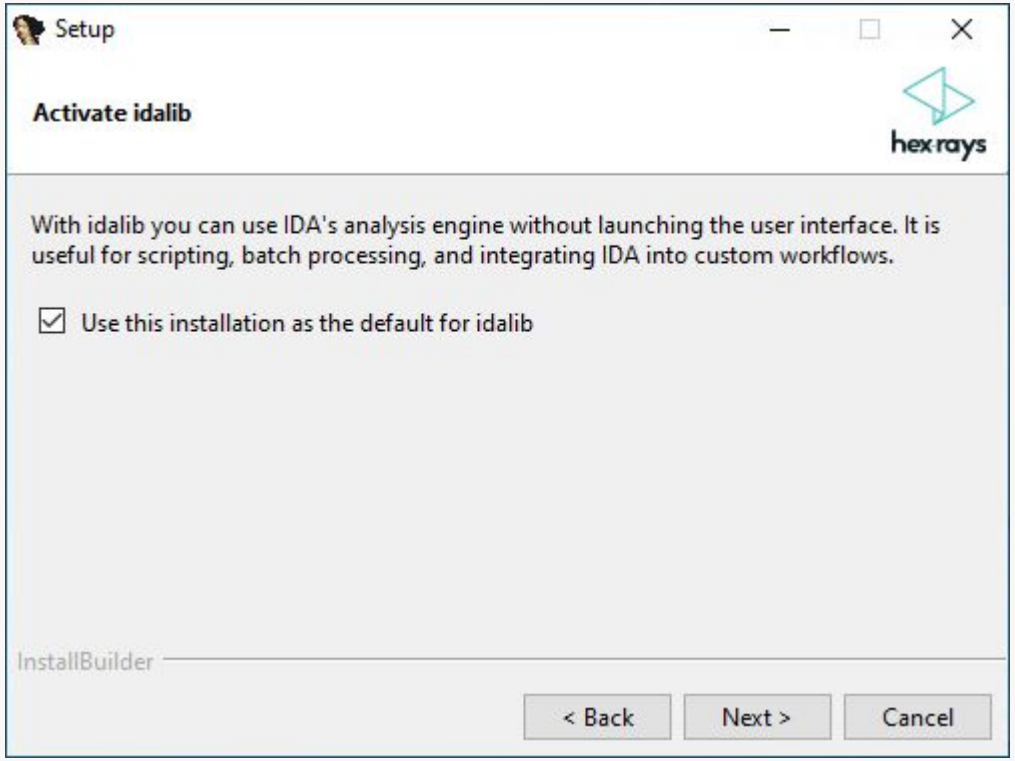
What do *you* want to do with idalib?


hex-rays
installation

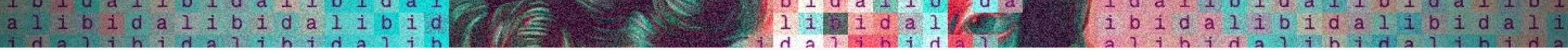
How idalib links with IDA



IDA 9.4 Installer



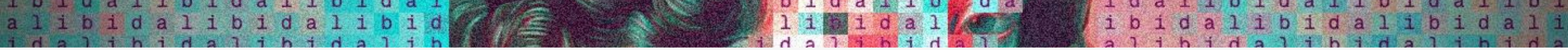
<=9.3: **<IDA installation path>**/idalib/python/py-activate-idalib.py



Install IDA(lib) via HCLI

https://hcli.docs.hex-rays.com

```
❯ hcli ida install \
      --license-id 96-345D-1F3F-12 \
      --download-id ida-pro:latest \
      --set-default \
      --accept-eula \
      --dry-run
Resolving tag: ida-pro:latest...
Normalized to: ida-pro:latest:armmac (current platform)
Tag resolved to: release/9.2/ida-pro/ida-pro_92_armmac.app.zip
Getting download URL for: release/9.2/ida-pro/ida-pro_92_armmac.app.zip
Starting download of release/9.2/ida-pro/ida-pro_92_armmac.app.zip...
Downloading ida-pro_92_armmac.app.zip    0% 0.7/565.0 MB 113.1 kB/s 1:23:07
```

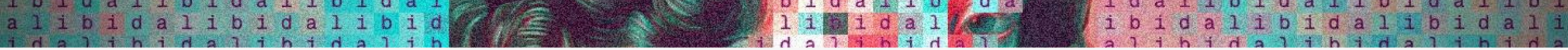


Install Domain API

http://domain-api.docs.hex-rays.com

```
~/Projects/idalib-slides % uv init
Initialized project `idalib-slides`
~/Projects/idalib-slides % uv add ida-domain
Using CPython 3.14.2
Creating virtual environment at: .venv
Resolved 5 packages in 571ms
Prepared 2 packages in 494ms
Installed 4 packages in 12ms
 + ida-domain==0.5.0
 + idapro==0.0.9
 + packaging==26.2
 + typing-extensions==4.15.0
```

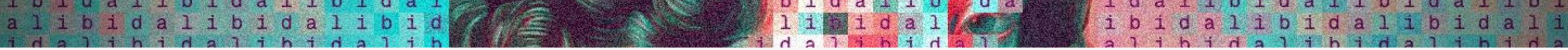

hex-rays
licensing



OEM license vs. Standard IDA Pro license

Personal automation Experimentation Solo workflows	any IDA **Pro** license (Essential, Expert, Ultimate or Classroom)
Team workflows Server-side analysis Production integration	IDA Pro OEM license **(Free during dev)**

> *OEM stands for **O**riginal **E**quipment **M**anufacturer. It simply means you're licensed to run IDA as part of another system. Like a video conferencing app embedding third-party noise-reduction tech into its platform.*



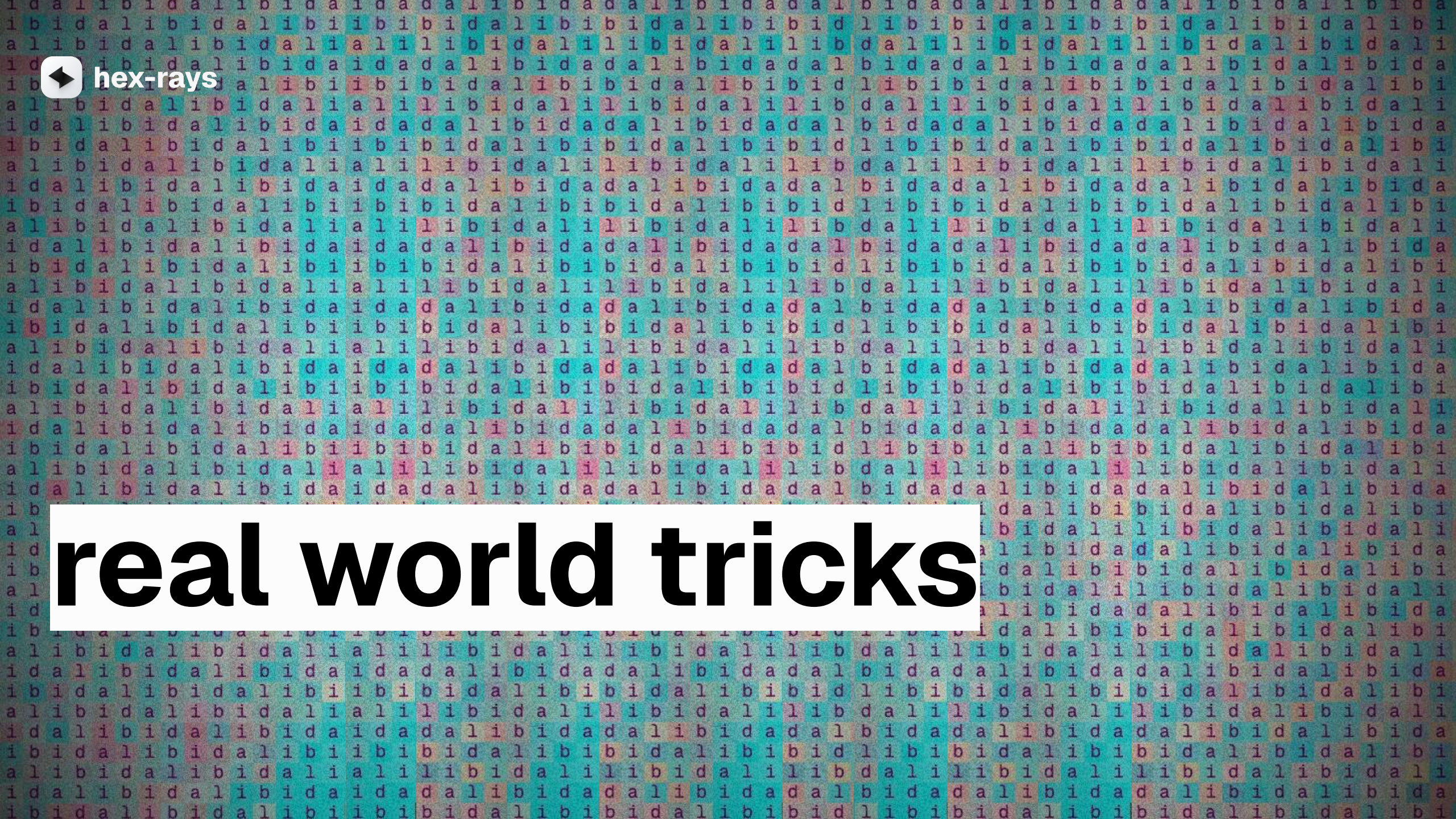
OEM license vs. Standard IDA Pro license

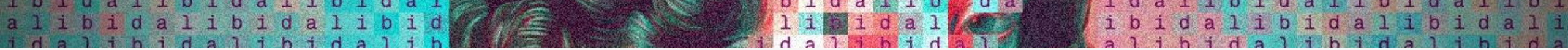
> *Under the hood, an OEM deployment is essentially:*
>
> ➥ ***idalib + a standard IDA Pro***
>
> *configured to operate legally and efficiently on servers or inside products.*

The OEM license also includes:

- scalability for headless/server workflows
- flexible deployment (containers, VMs, cloud)
- embedding IDA's decompiler engine in your tooling
- permission for team use and multi-user access
- the ability to integrate IDA functionality into commercial offerings

- **access to all decompilers** (quite an upgrade if you don't already have them!)
- direct access to our **technical experts for architecture guidance**
- **accelerated support** for setup, integration, and ideation
- **collaborative product feedback** and feature-proposal pathways


hex-rays
real world tricks



Deployment ideas

CI/CD

HCLI to install IDA(lib)

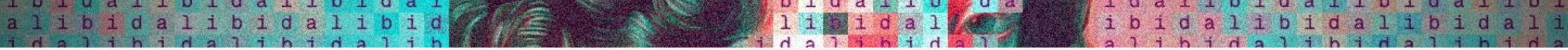
- Totally headless
- Depends on secrets: token + license ID, provide via env vars
- Two shell commands

GitHub Actions (or similar) to test many configurations (OS/Arch/version)

- e.g., HCLI runs full test suite against matrix with 34 combinations
- https://github.com/HexRaysSA/ida-hcli-actions

Docker: same idea, build image with idalib, push to private registry

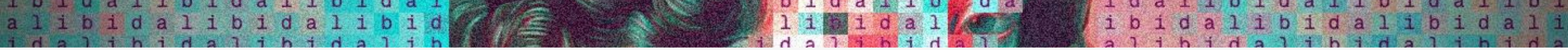
- IDA is single threaded, so allocate 1 vCPU per container
- Example: Dockerfile



Tips & Tricks

- Common pattern: follow xrefs to gather context
- IDA is single threaded - must access from the primary thread
- IDA uses a FS-backed database - operate in $TMP for transient analysis
- IDA avoids a complete decompilation unless requested
  - Therefore its very fast, and you can tune interactions
  - Target your decompilation requests for best performance
  - Decompile all for most thorough analysis

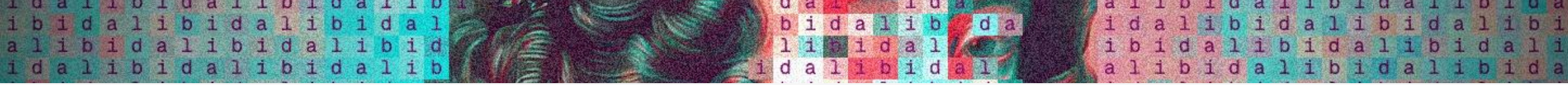

hex-rays
AI and MCP



Agentic IDA

- MCP server: https://github.com/mrexodia/ida-pro-mcp
  - or write your own
- “Code is all you need”: teach Agent to use the Domain API and encourage temporary/inline script evaluation
- Skills: https://github.com/HexRaysSA/claude-marketplace
- ida-domain llms.txt


hex-rays
q & a



Contact

- support@hex-rays.com (General questions & support)
- msobon@hex-rays.com (Michal, IDA Domain)
- dogilvie@hex-rays.com (Duncan, IDA Malware)
- wballenthin@hex-rays.com (Willi, IDA Malware)