Unpacking Approach

Unpacking Methods

DEBUGGER + BREAKPOINTS

RUN AND DUMP

STATIC UNPACKING

EMULATION

SELF-EXTRACTING PATCH

Unpacking Methods

DEBUGGER + BREAKPOINTS

manual form of unpacking

general idea how it is packed is enough

breakpoints on functions that

- allocate memory
- transport data
- execute

Unpacking Methods

RUN AND DUMP

semi-automated, easy

needs no knowledge how file is packed

tools: mal_unpack, MegaDumper

Unpacking Methods

STATIC UNPACKING

usually by writing a script

need to understand every detail

easily applicable to many samples

tools: binary refinery, CyberChef, any scripting language

Unpacking Methods

EMULATION

emulate until malware is done unpacking, then dump

problem: anti-emulation is common

examples: box-js, JSDetox, dumpulator, speakeasy

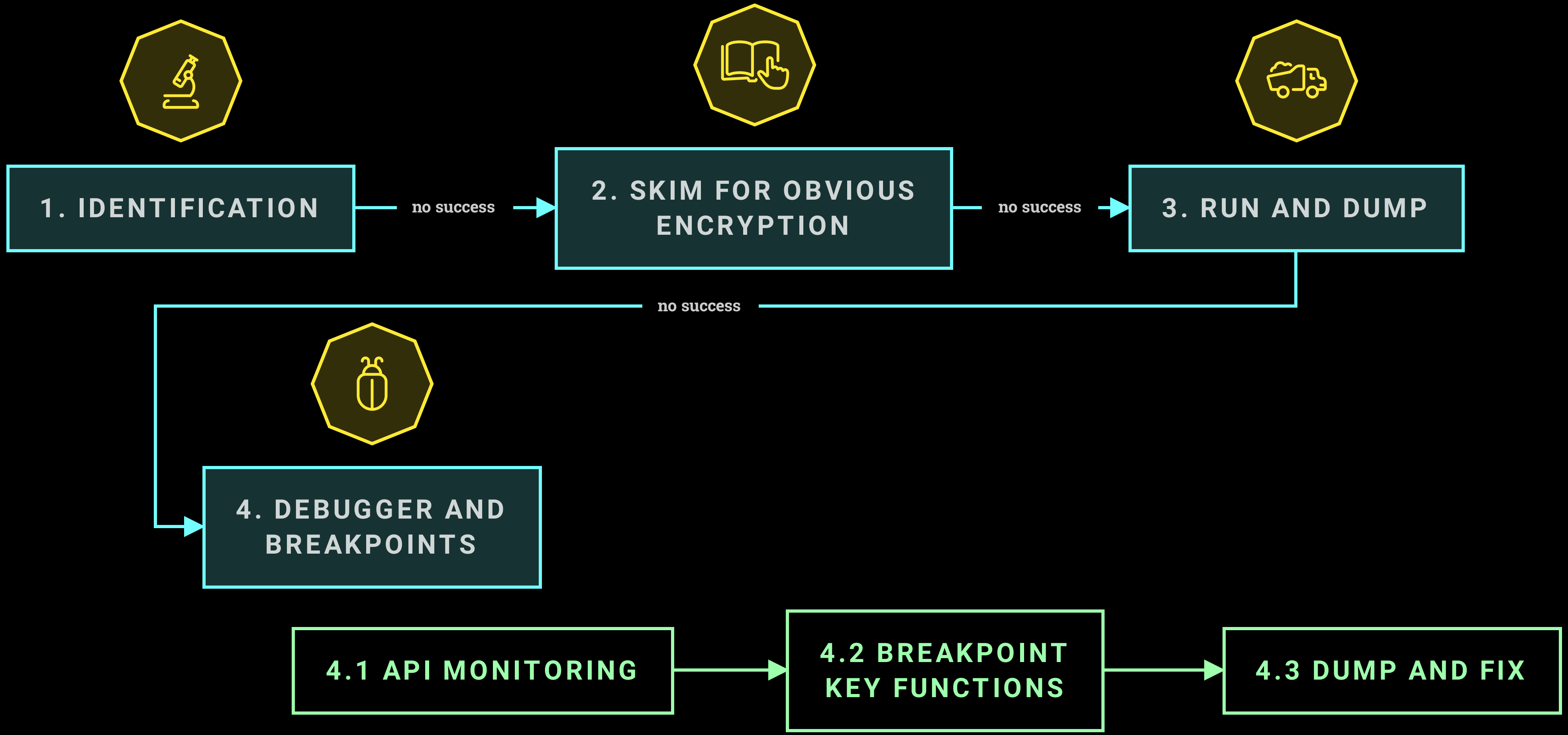
Unpacking Methods

SELF-EXTRACTING PATCH

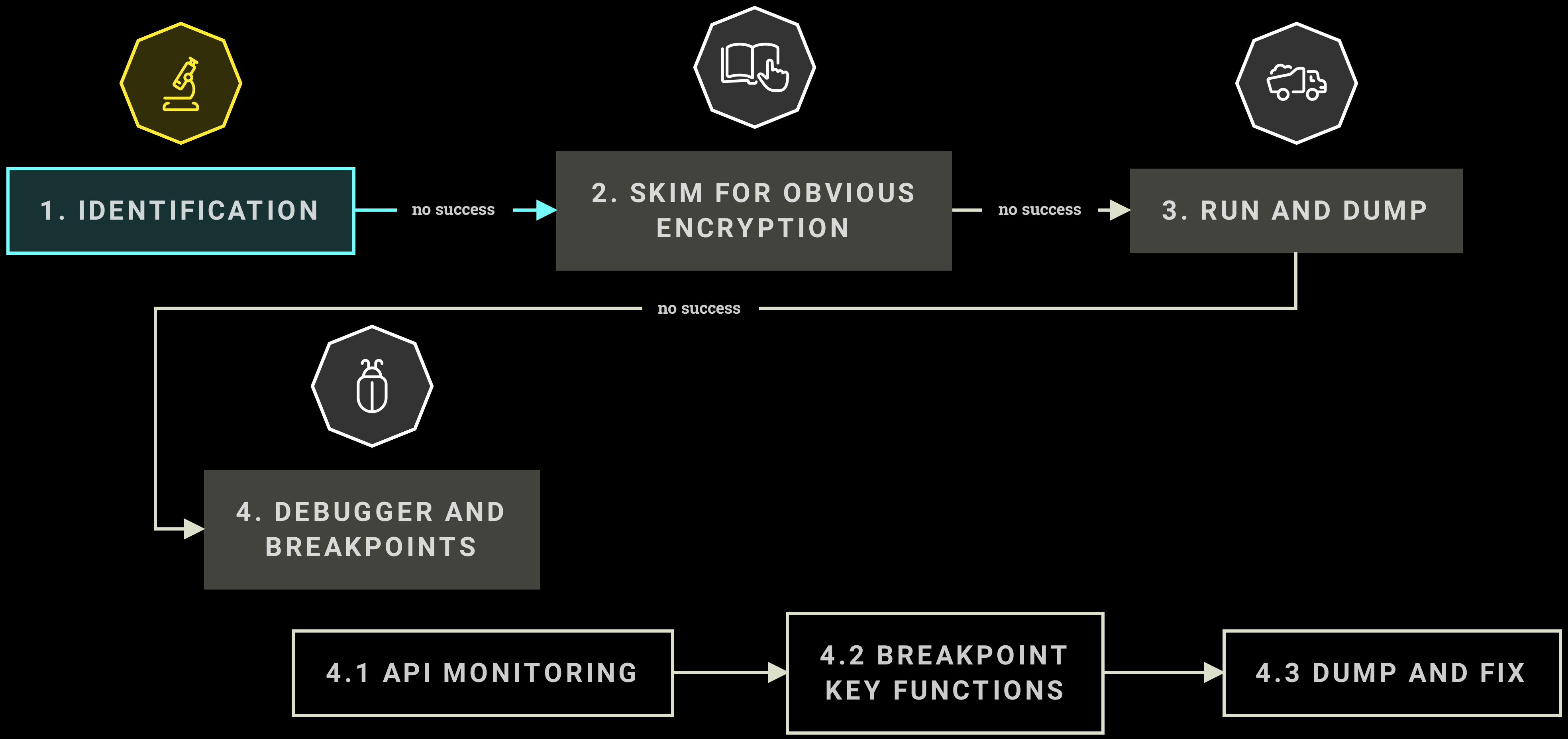
patch that dumps malware after unpacking

often easiest method for scripts: replace execute with write instructions

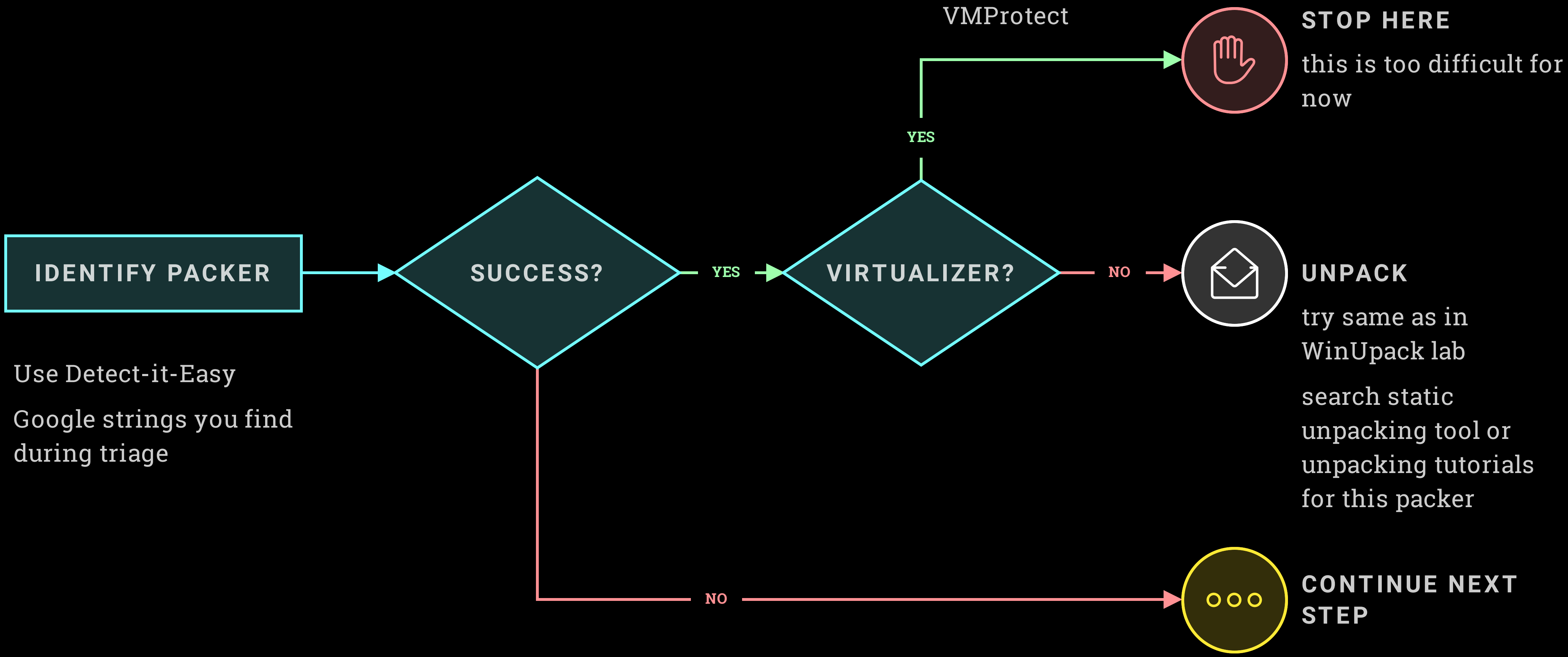
Unpacking Approach



Unpacking Approach



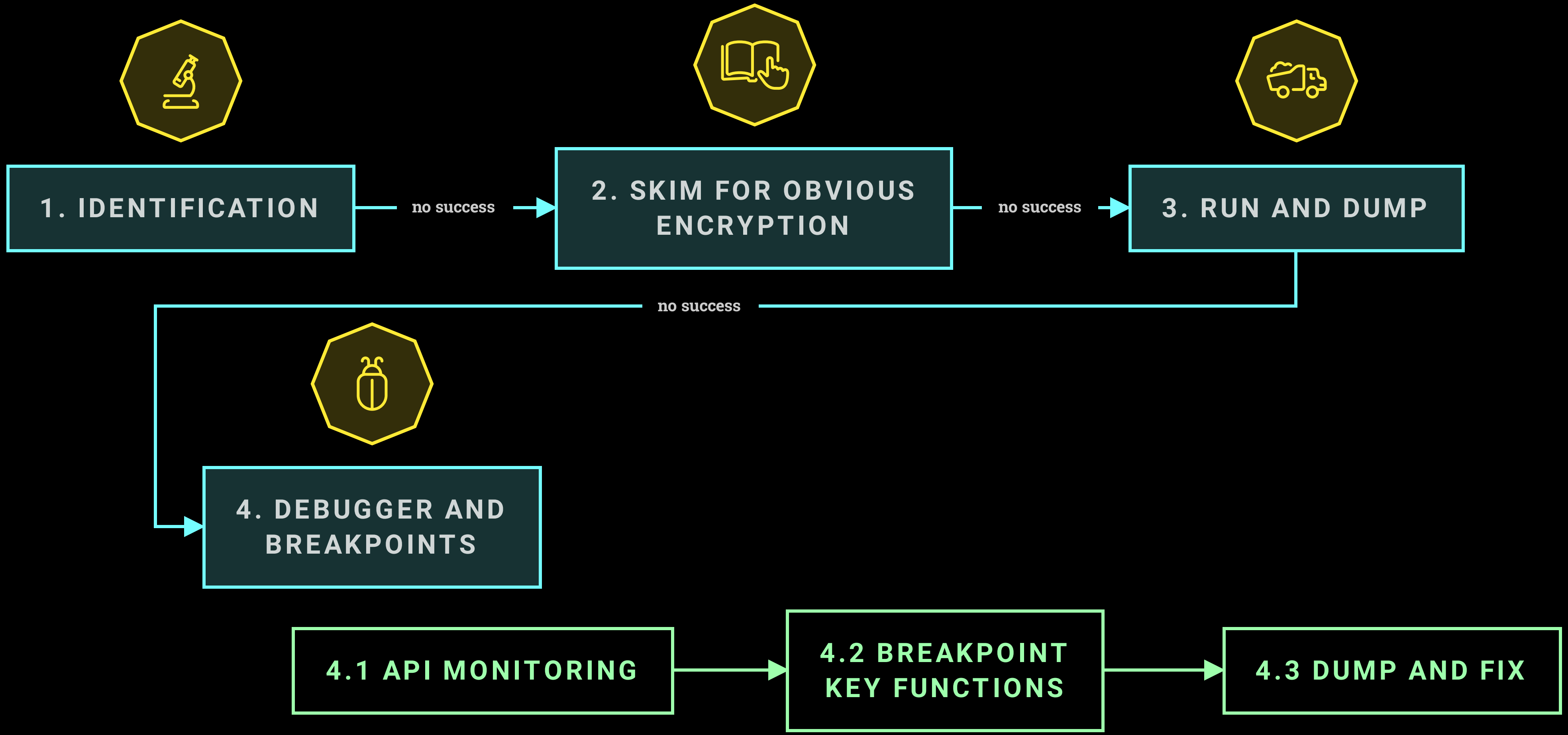
Step 1 Identification


IDENTIFY PACKER
SUCCESS?
YES
VIRTUALIZER?
YES
VMProtect
STOP HERE
this is too difficult for now
NO
UNPACK
try same as in WinUpack lab
search static unpacking tool or unpacking tutorials for this packer
NO
CONTINUE NEXT STEP

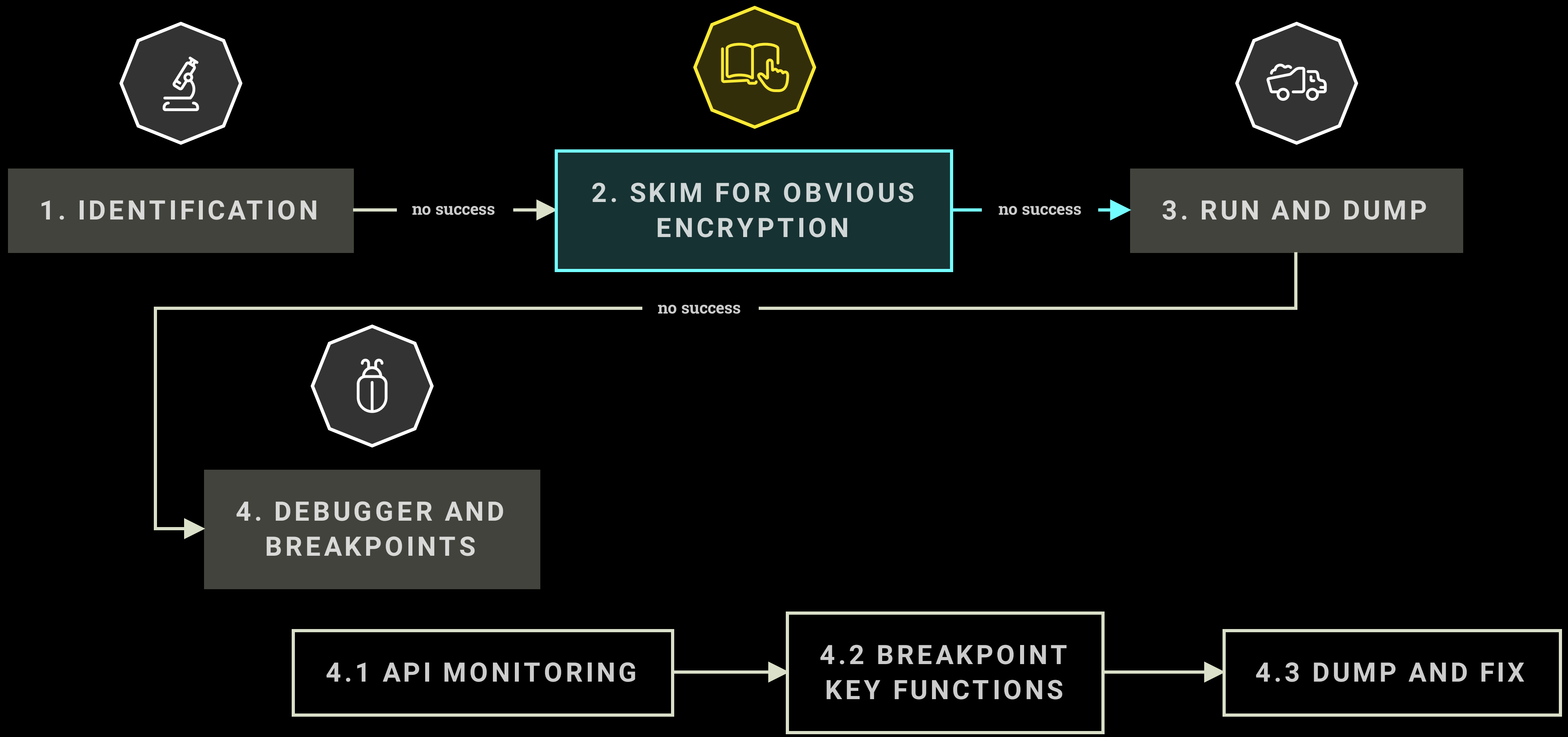
Use Detect-it-Easy

Google strings you find during triage

Unpacking Approach



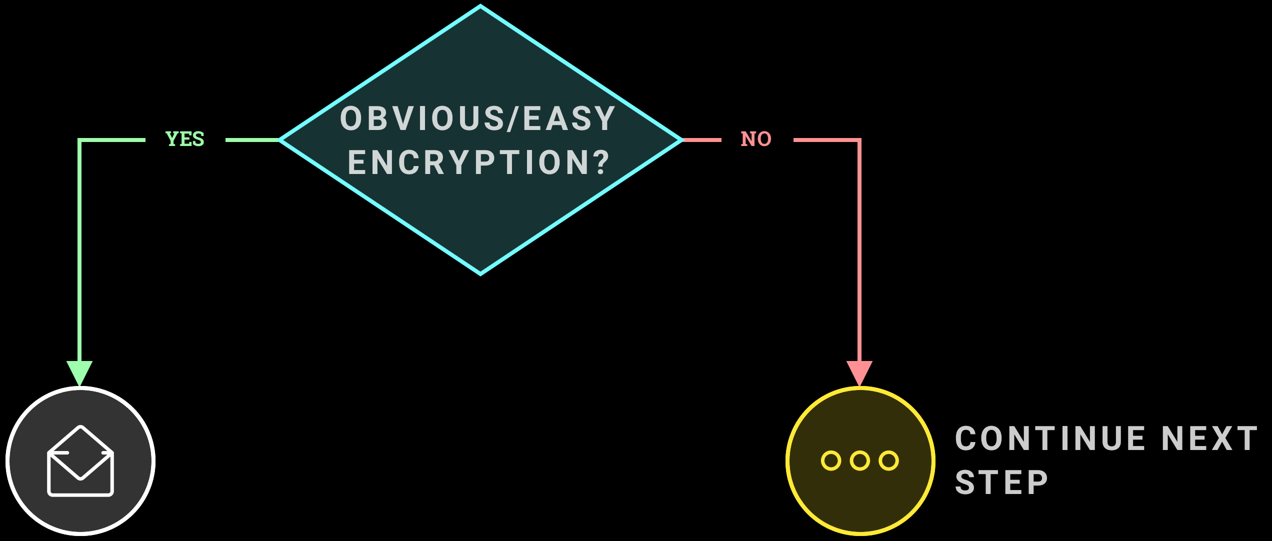
Unpacking Approach



Step 2 Skim for Obvious Encryption / Encoding

- **LARGE BASE64 STRINGS**
  decode them
- **XORED AREAS**
  XOR with one byte visible to naked eye in hex editor
  use XOR bruteforcing
- **LARGE INTEGER ARRAYS**
  often in managed assemblies and scripts
- **CHECK SPECIFIC AREAS**
  resources: PE, .NET
  overlay
  end-of-file
  last section
  strings

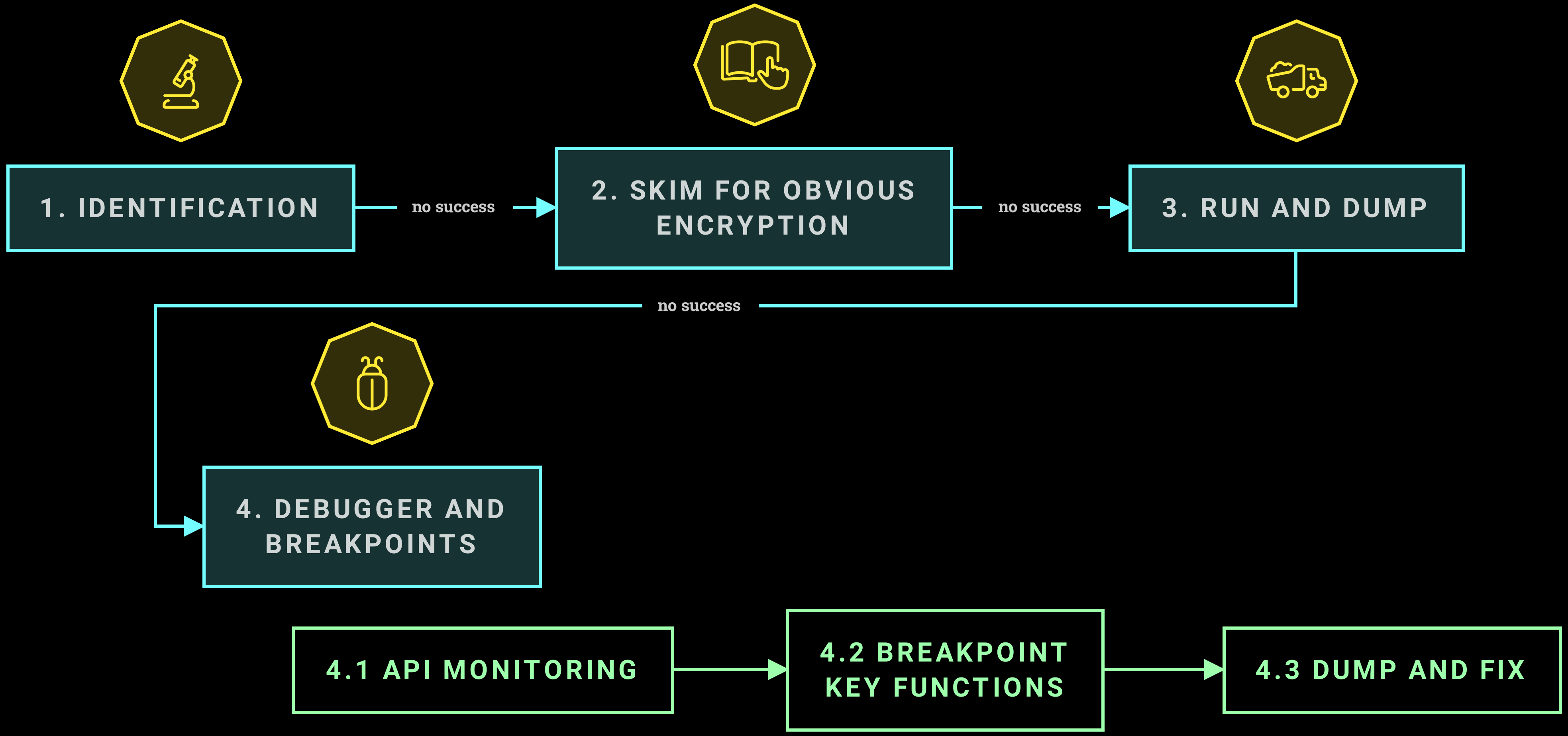
Step 2 Skim for Obvious Encryption / Encoding



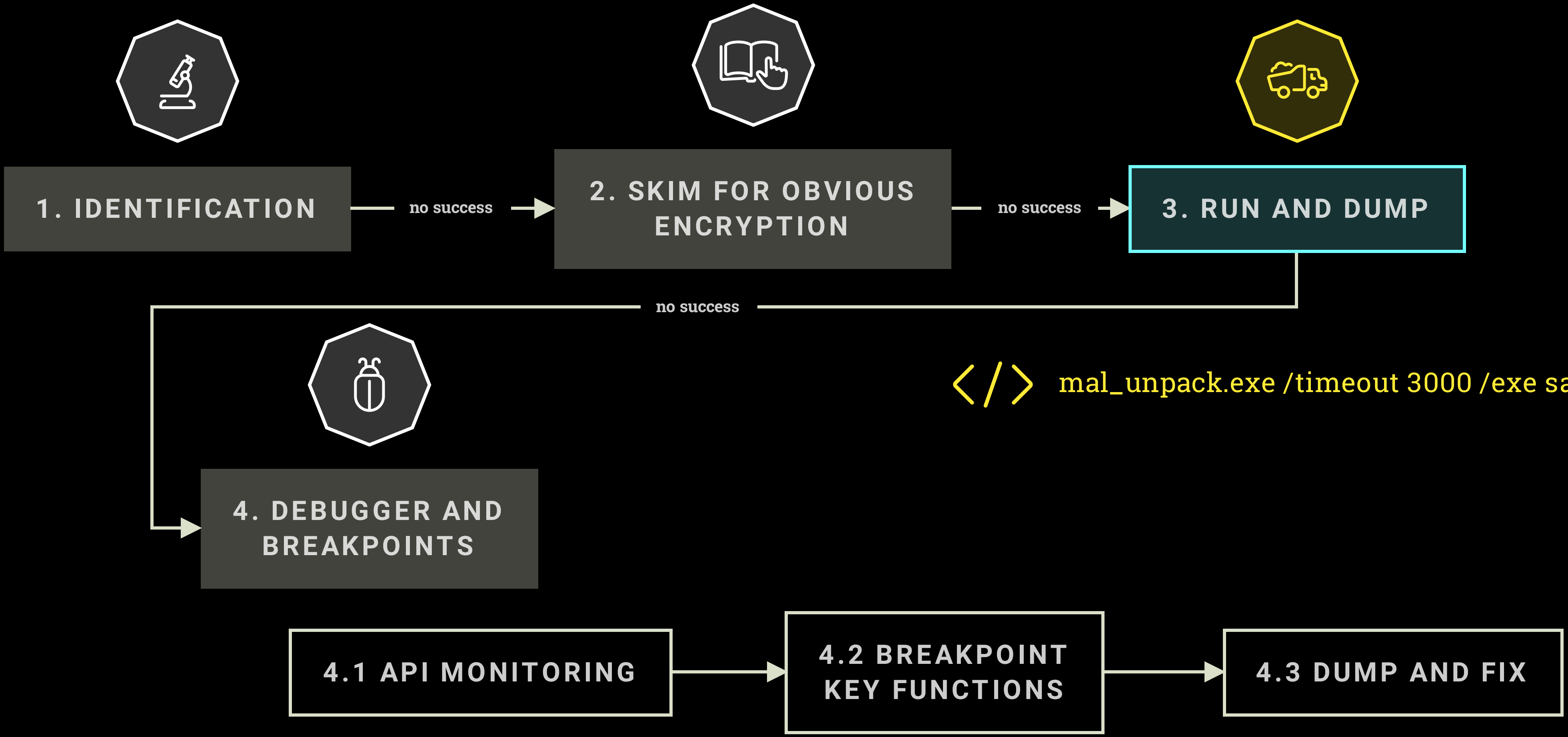
STATIC UNPACKING

- binary refinery
- Python script
- CyberChef
- ...

Unpacking Approach

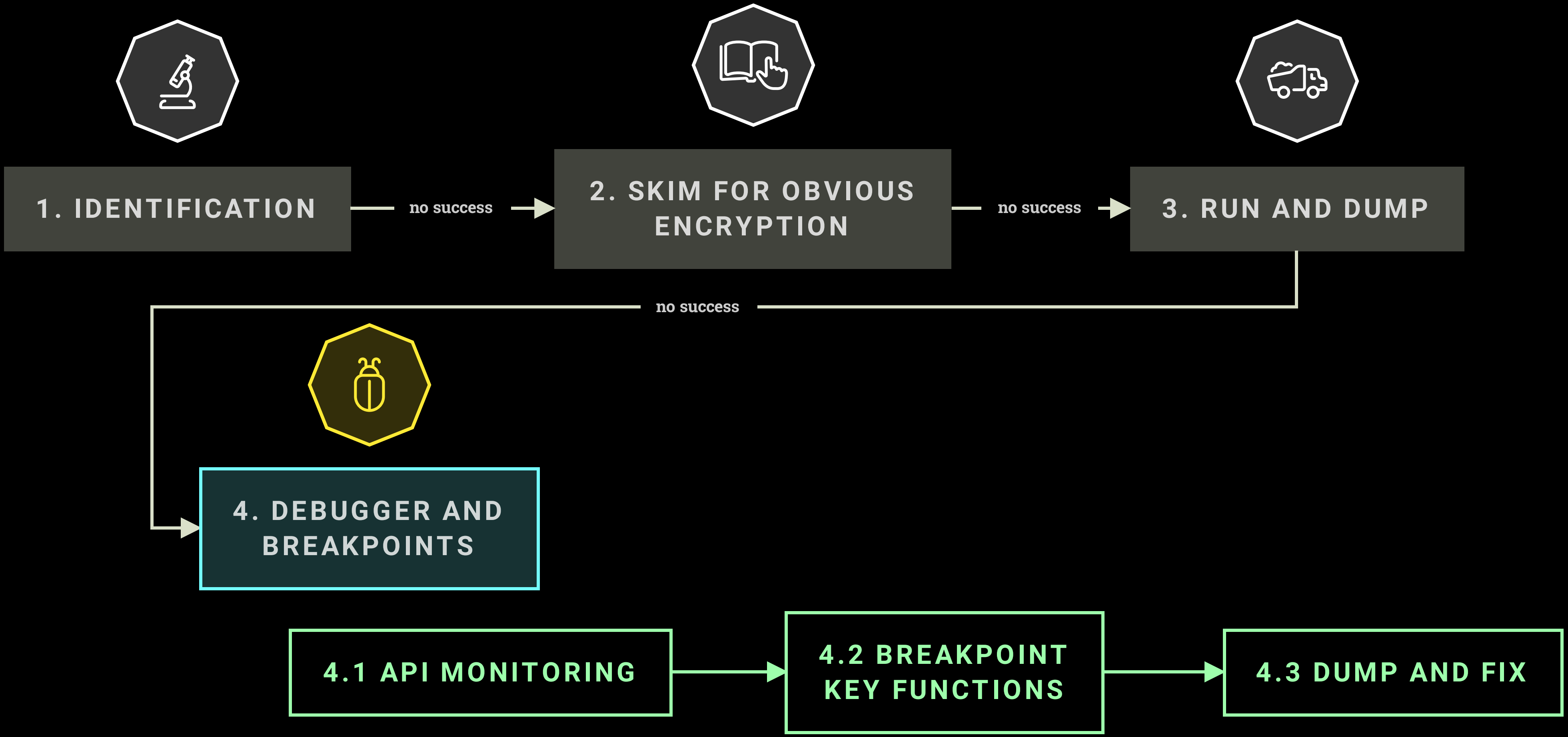


Unpacking Approach

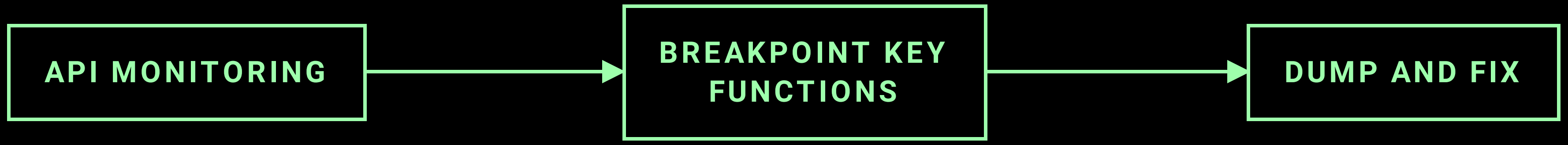


```
mal_unpack.exe /timeout 3000 /exe sample.exe
```

Unpacking Approach



Debugger and Breakpoints



log API calls, you can use:

* monitoring
* tracing
* emulation
* sandbox reports

purpose: get an idea how the stub unpacks

break on functions that probably:

* transport target file data
* execute target code
* allocate memory
* create processes
* decrypt target file data

fixes may include:

* PE unmapping
* set original entry point
* import fixing
* fixing erased header

Unpacking Methods

DEBUGGER + BREAKPOINTS

RUN AND DUMP

STATIC UNPACKING

EMULATION

SELF-EXTRACTING PATCH

We talked about these

DEBUGGER + BREAKPOINTS

RUN AND DUMP

STATIC UNPACKING

EMULATION

SELF-EXTRACTING PATCH

What about these?

DEBUGGER + BREAKPOINTS

RUN AND DUMP

STATIC UNPACKING

EMULATION

SELF-EXTRACTING PATCH

What about these?

EMULATION

- can use instead of debugger and breakpoints
- personal preference

SELF-EXTRACTING PATCH

- script unpacking