Wrappers

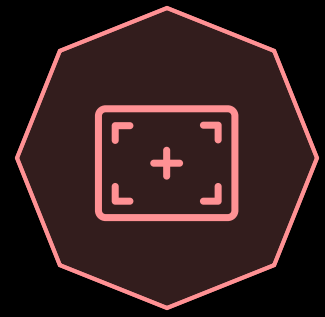
Code in Binaries

STATICALLY LINKED LIBRARIES

INCLUDED EXECUTION ENVIRONMENTS

- Wrappers
- Installers

INITIALIZATION CODE



ACTUAL MALWARE CODE

The code we are interested in

Code in Binaries

STATICALLY LINKED LIBRARIES

INCLUDED EXECUTION ENVIRONMENTS

- Wrappers
- Installers

INITIALIZATION CODE

ACTUAL MALWARE CODE

The code we are interested in

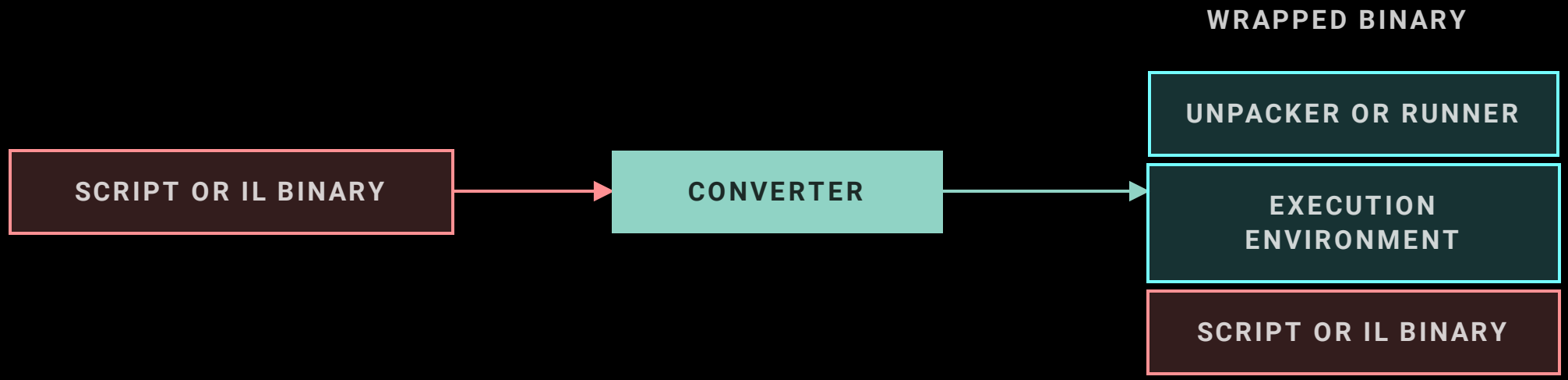
Wrappers

- scripts or intermediate language binaries need special environment to run in
- execution environment might not be installed on target system
- wrappers create a single executable from the script or IL binary that has everything to run it

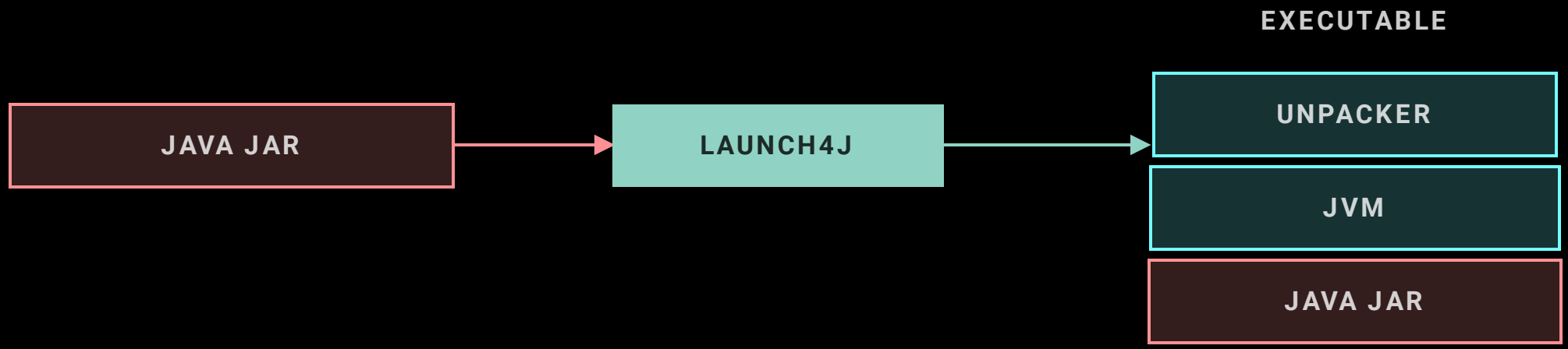
Wrapped files

- often big executable and tied to one operating system
- but works on that operating system regardless of installed programs

Wrappers

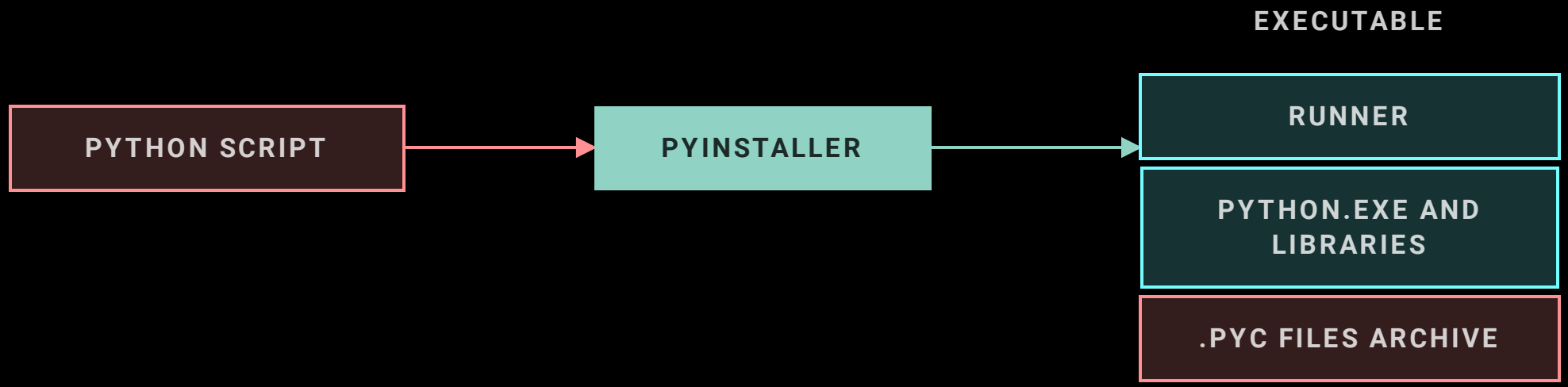

WRAPPED BINARY
SCRIPT OR IL BINARY
CONVERTER
UNPACKER OR RUNNER
EXECUTION ENVIRONMENT
SCRIPT OR IL BINARY

Wrappers



Wrappers

Wrappers


EXECUTABLE
PYTHON SCRIPT
PYINSTALLER
RUNNER
PYTHON.EXE AND LIBRARIES
.PYC FILES ARCHIVE

Obtaining the code from wrapped files

DYNAMICALLY

- often: unpacker drops everything into %TEMP%
- step 1: monitor file writing operations
- step 2: run executable
- step 3: copy the newly written files

STATICALLY - NOT ENCRYPTED

- search for embedded script or IL binary and extract with hex editor
- use magic numbers for IL binary search
- use strings.exe for script search

STATICALLY - ENCRYPTED

- identify wrapper with DiE
- search for extraction tool
- otherwise you need to analyze the unpacker for decryption procedure