Bypass Security Checking With Frida

Hi!

I am Satria Ady Pradana

Cyber Security
Consultant
@
Mitra Integrasi
Informatika

xathrya.sabertooth

@xathrya

Reversing.ID
Komunitas Reverse Engineering Indonsia

My problem as Reverser (And Pentester)

- I want to analyze some binary, application, or whatever
  - How it works?
  - What's this functions for?
  - Why they use this solution?
- Sometimes I want to know specific part of function.
- And sometimes the binary is protected with certain kinds of "magic"
  - Bypass it? Of course
- Btw, I am lazy…

Solution (of OldSchool Technique)

- Code Injection to alter the behavior
    - DLL Injection, modify IAT, hook function, etc …

Problem:

- Too much works!
- Need to recreate the DLL for each iteration.

Enter Frida!

What is Frida?

- **Dynamic instrumentation** toolkit
  - Inspect and instrument live process
  - Execute instrumentation scripts inside other processes
- Multiplatform
  - Windows, Linux, Mac, iOS, Android, QNX
- Open source

DBI?

- Dynamic Binary Instrumentation
- Method of analyzing the behavior of a binary application at runtime through the injection of instrumentation code.
  - gain insight into the state of an application at various points in execution.
- Instrumentation code executes as part of the normal instruction stream after being injected.

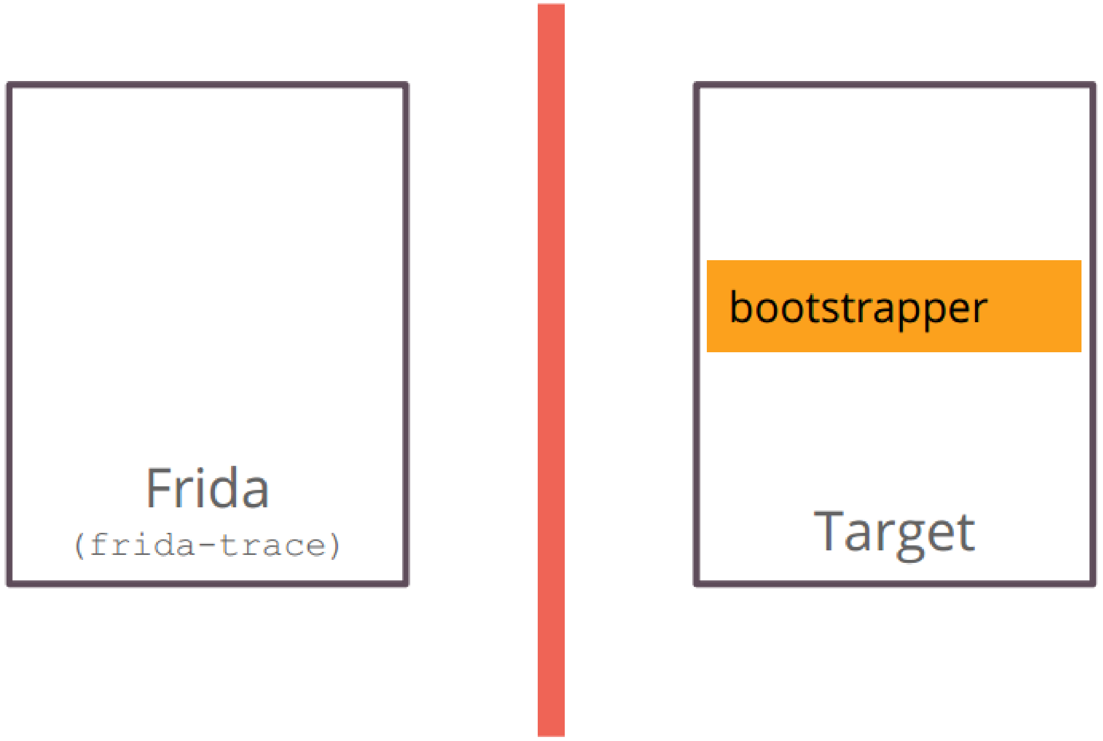
What can be done in DBI?

- Access process memory
- Overwrite functions while the application is running
- Call functions from imported classes
- Find object instance on the heap and use them
- Hook, trace, and intercept function.

Frida’s Dynamic Nature

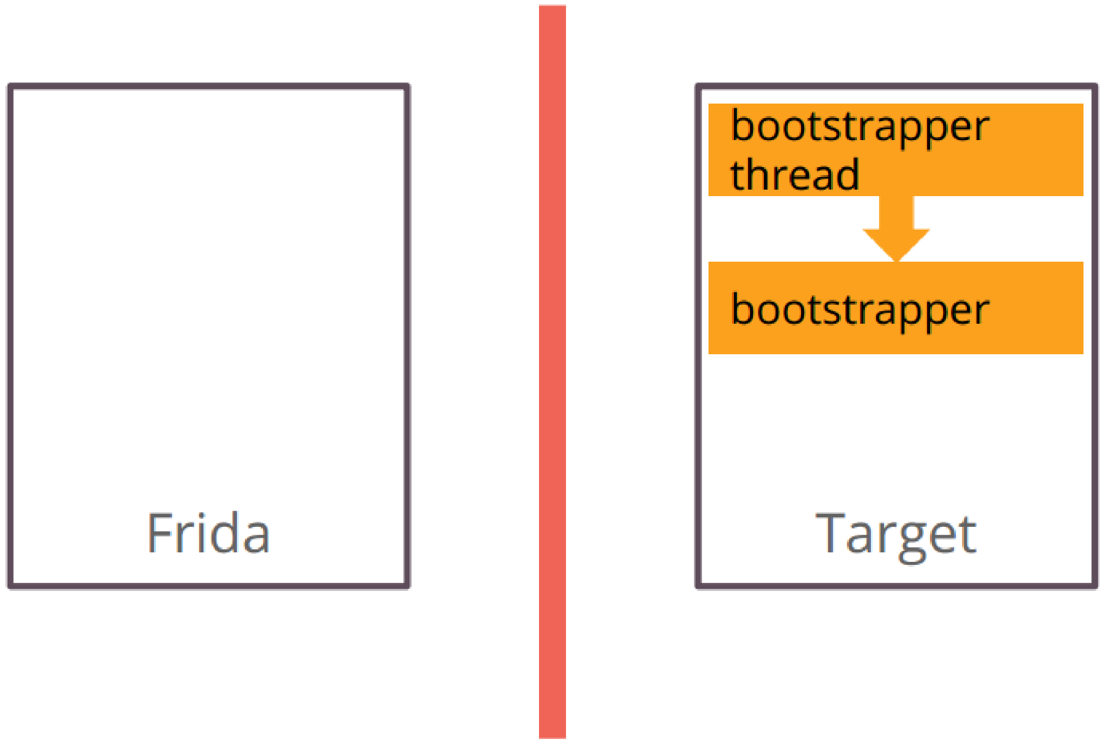
- JavaScript API for instrumentation script (debugging logic)
- With various bindings:
  - Python
  - .NET
  - JavaScript (Node.js)
  - Qt/Qml
  - etc

How does Frida work?



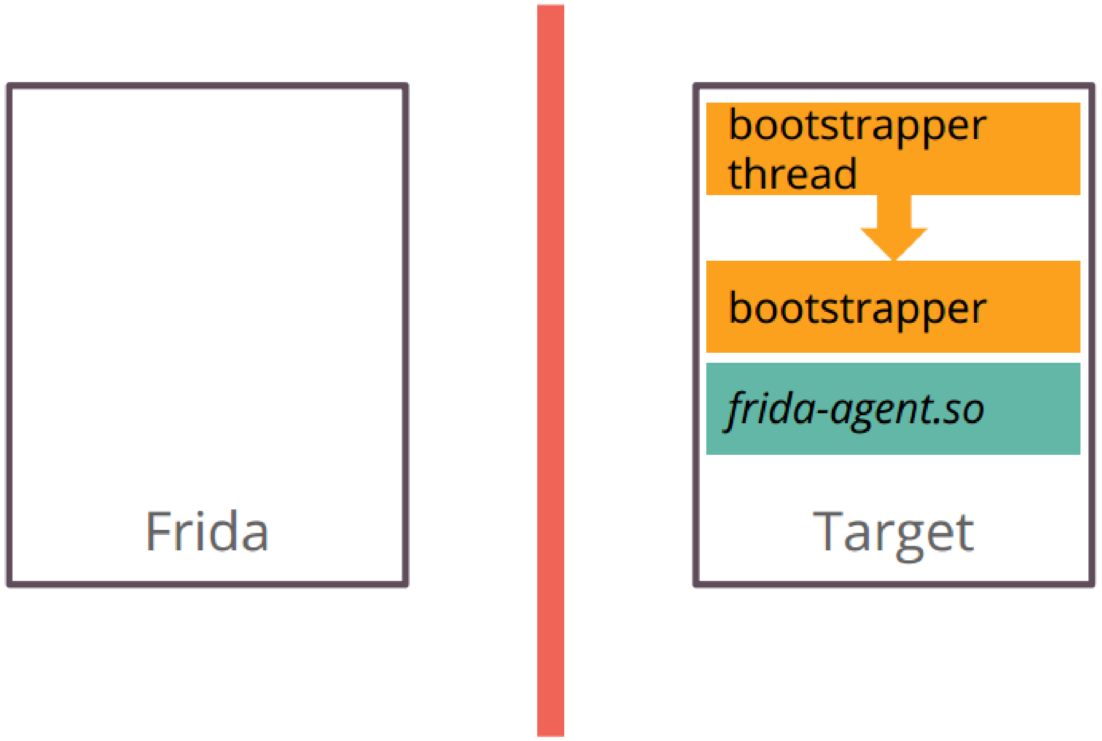
Frida process writes *bootstrapper* code into memory of *Target* process

How does Frida work?



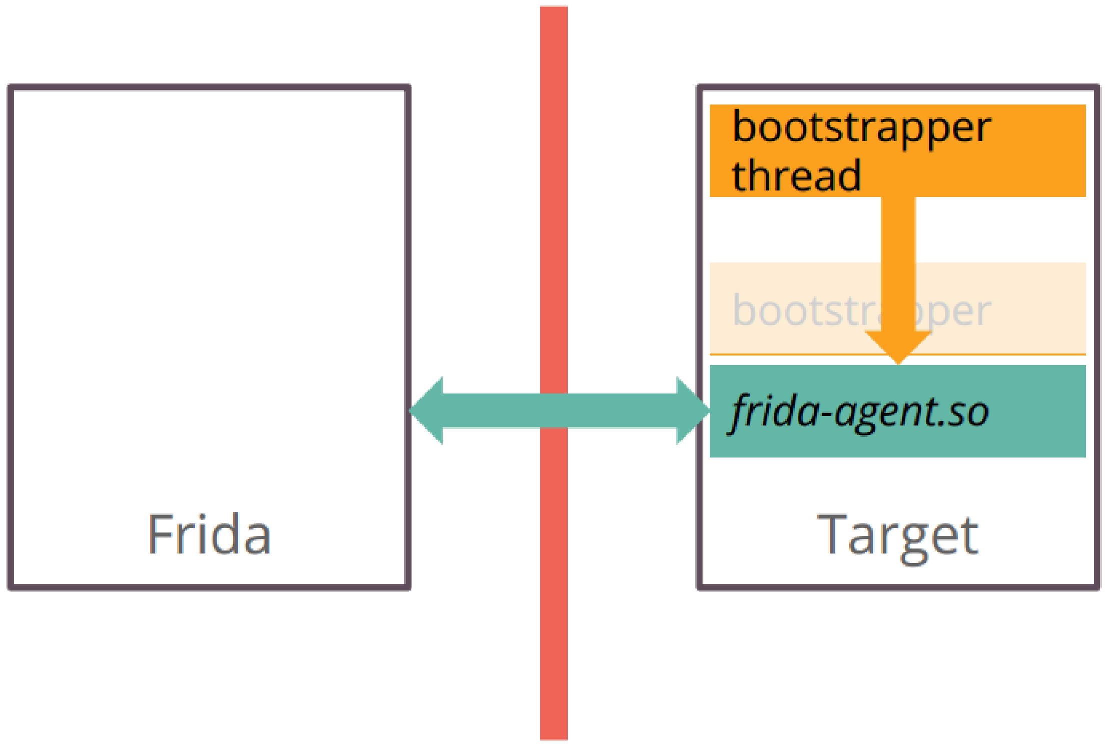
Frida hijacks an existing thread in *Target* and has it execute *bootstrapper*

How does Frida work?



Bootstrapper loads *frida-agent.so* into *Target*'s memory space

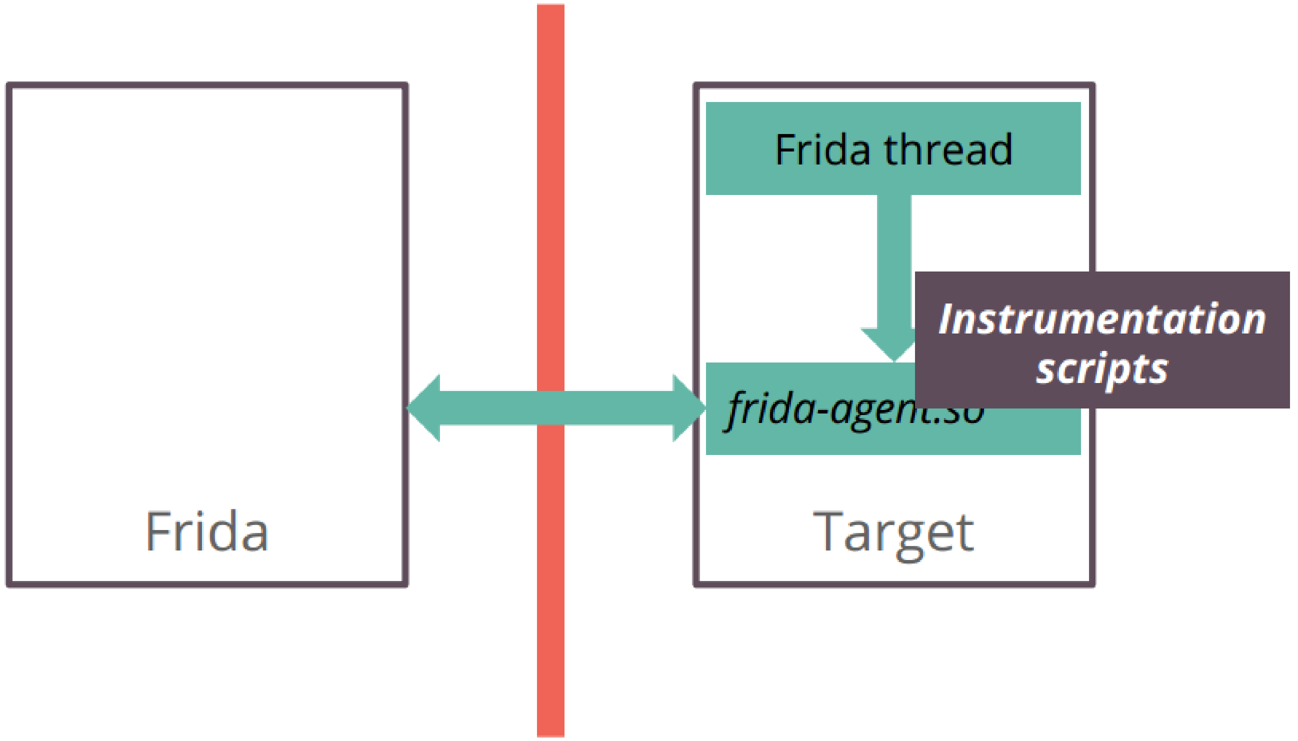
How does Frida work?



Frida-agent.so opens a bidirectional channel between *Frida* and *Target*

FRIDA

How does Frida work?



Frida-agent.so sets up its own *thread*, and accepts instrumentation scripts from *Frida*

Install Frida (Python)

- # python –m pip install Frida
- Or
- # pip install frida

Basics

- Attach to process (locally, remotely)
- Enumerate modules and threads
- Check function call’s arguments
- Modify arguments
- Modify function

Bypassing Security Check

In Our Example:

- Get Plaintext on Encryption Process
- Brute Force PIN
- Root Checking
- SSL Pinning

Get Plaintext on Encryption Process

- Concept: To produce a Ciphertext, an encryption process need a plaintext, key, and some other parameters such as Initialization vector.
- Bypass: Hook the encryption function and log the plaintext data.

Brute Force PIN

- Concept: A PIN or password protected binary need a function to compare user-input PIN with the correct one.
- Bypass: Hook the checking and force it to return true.

Root Checking (Android, iOS)

- Concept: Check the presence of several items as result or end of rooting process
  - Does binary SU exist?
  - Does apk Superuser exist?
- Bypass: Hook the checking and force it to return true.

SSL Pinning

- Concept: Use a local copy of x509 digital certificate to verify digital certificate provided by server.
- Bypass: Hook the checking and force it to return true, regardless the value of provided certificate.

Alternative?

- PIN:
  - https://software.intel.com/en-us/articles/pin-a-dynamic-binary-instrumentation-tool
- DynamoRIO:
  - http://www.dynamorio.org/

In both cases, you write a code on top of those framework, compile them, and inject the library into the target.