Medusa

A disassembler and something more...

Angelin Njakasoa BOOZ

LSE Summer Week 2016

Quarkslab

Presentation

- Whoami?
- Where do I work?
- What do I do?

Reverse Engineering

What is Reverse Engineering?

Reverse engineering, also called back engineering, is the processes of extracting knowledge or design information from anything man-made and re-producing it or re-producing anything based on the extracted information. The process often involves disassembling something and analyzing its components and workings in detail.

Why using reverse engineering?

- Analyze goodware for security reinforcement.
- Analyze malware to identify it easier and develop counter-measure.

Financial impact of malware

- At rate, ransomware is on pace to be a $1 billion a year crime this year.
- The recent cyber attack on Bangladesh's central bank that let hackers stole over $80 Million from the institutes' Federal Reserve bank account was reportedly caused due to the Malware installed on the Bank's computer systems.
- Although the malware type has not been identified, the malicious software likely included spying programs that let the group learn how money was processed, sent and received.

What is Medusa?

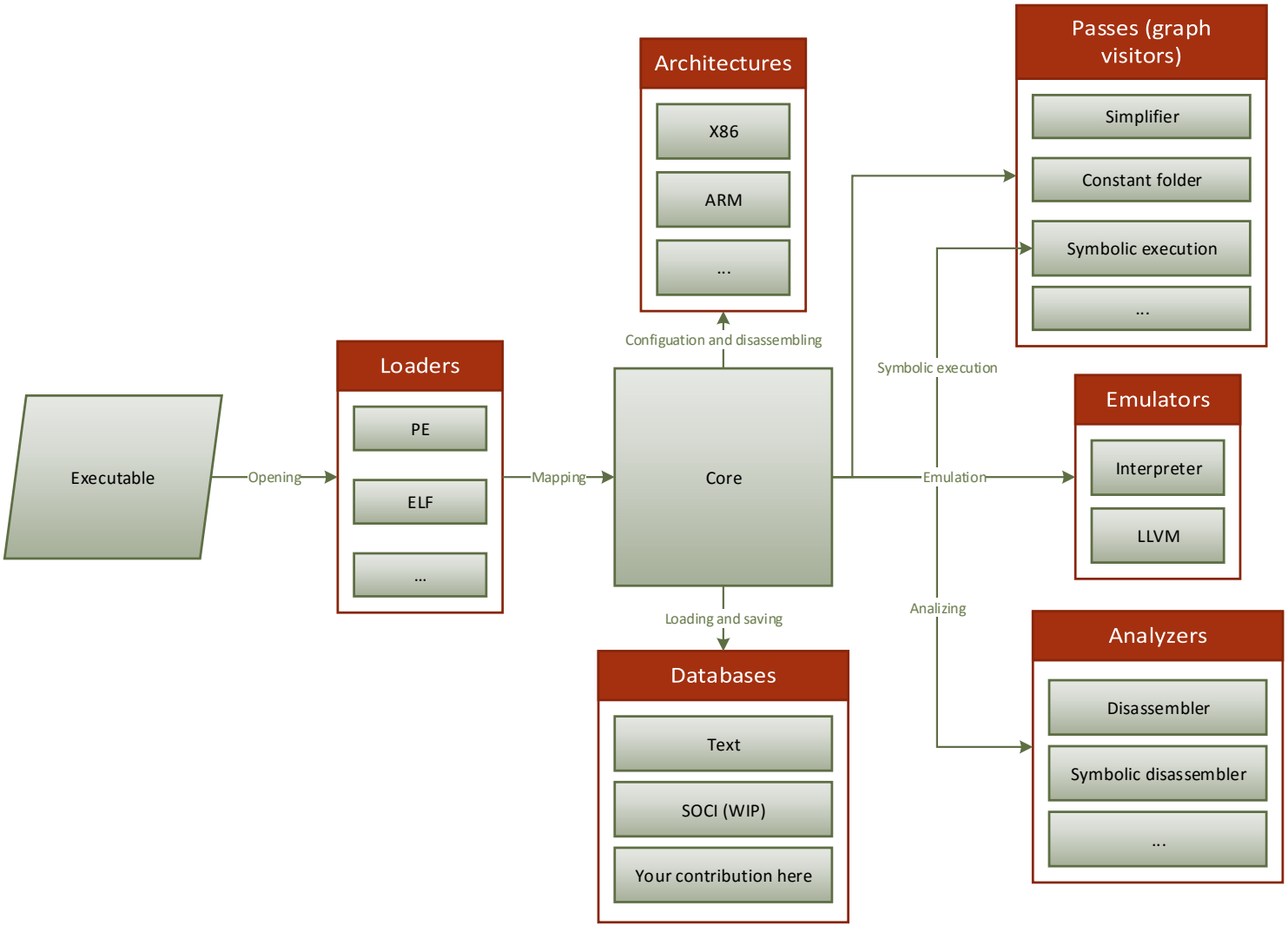
Medusa

A disassembler with semantic, emulation and symbolic execution. It was made to have a more detailed analysis of binaries.

Medusa is composed by:

- Loaders
- Architectures
- Passes
- Databases
- Analyzers
- Disassembler
- Emulator
- Symbolic Execution Engine

Design


Architectures
X86
ARM
...
Passes (graph visitors)
Simplifier
Constant folder
Symbolic execution
...
Configuation and disassembling
Symbolic execution
Loaders
PE
ELF
...
Executable
Opening
Mapping
Core
Emulation
Emulators
Interpreter
LLVM
Loading and saving
Analizing
Databases
Text
SOCI (WIP)
Your contribution here
Analyzers
Disassembler
Symbolic disassembler
...

Emulator

- CPU: Medusa relies on YAML files to describe each instructions, most of them also contain a specific field name semantic.
- Memory: Create a memory context to execute a program
- OS: We emulate function's behavior in python

Why emulation?

- Control what the target can access by managing memory, API, etc;
- Modify the execution on the fly
- Monitoring the context of the program

Semantic

Architectures:

- arm: .yaml 11485 loc - .py of 681 loc
- x86: .yaml 14121 - .py 794 loc
- z80: .yaml 4151 loc - .py 187 loc
- st62: .yaml 589 loc - .py 348 loc

Semantic

Into yaml file:

- opcode 0x00
- mnemonic add
- operand Eb, Gb
- update_flags: cf, pf, af, zf, sf, of
- semantic add

The generator is written in python because it's easier to parse.

How does it works?

Demo

Obfuscation

Obfuscation

Obfuscation is the obscuring of intended meaning in communication, making the message confusing, willfully ambiguous, or harder to understand.

Symbolic Execution

Definition

Symbolic execution (also symbolic evaluation) is a means of analyzing a program to determine what inputs cause each part of a program to execute.

Obfuscation

Some methods of obfuscations:

- Constant unfolding
- Obfuscated pattern
- Data flattening
- Code flattening

Symbolic execution on Constant unfolding

$x = 0xf9cbe47a + 0x6341b86$

Symbolic execution on Pattern of obfuscation

Demo

Conclusion

Questions

Github

https://github.com/wisk/medusa

Big Thanks!

Thanks to Wisk, Quarkslab and the LSE!