FoxDec User Manual

FoxDec

Decompilation based on Formal Methods

dr. Freek Verbeek
prof. Binoy Ravindran

freek@vt.edu

Supported by the DARPA project FALCON:
Formal Analysis of Legacy COde domaiNs

User Manual

March 2, 2022

FoxDec is a tool actively developped at Virginia Tech (US) and the Open University of the Netherlands. Its aim is to lift binaries to a higher level of abstraction, in such a way that formal guarantees can be provided that the lifted representation is sound with respect to the original binary. This document provides a user manual, further information on implementation and limitations, as well as references for further reading.

Remark: *FoxDec is evolving quickly, and new features and capabilities are actively being developped. Do not hesitate to contact us for questions, remarks and suggestions.*

1 User Manual with Example

1.1 Download & Installation

Up-to-date information on where to download FoxDec, and instructions for building and installation, can be found at:

`https://ssrg-vt.github.io/FoxDec/#build`

1.2 Running FoxDec to create `.report` file

Compile. As running example, we will consider the `wc` command. For sake of explanation, we consider a small and simple implementation instead of taking the binary as available in a standard Linux or Mac distribution[1]. First, we compile the example. Go to the directory for the running example `wc_small`. There, we compile the file `wc.c` to an executable `wc`.

Compile the running example

```
cd ./FoxDec/foxdec/examples/wc_small
gcc wc.c -o wc
```

Extract. Subsequently, we extract information from the generated binary. We use standard tools for this: for Linux these are `readelf` and `nm`, and for MacOs these are `otool` and `nm`. Two scripts are provided: `dump_elf.sh` for Linux ELF files, and `dump_macho.sh` MacOs MachO files. Their command-line usage is:

```
dump_elf.sh $BINARY $NAME
```

[1]The source code of the `wc` example can be found here:
`https://www.gnu.org/software/cflow/manual/html_node/Source-of-wc-command.html`

`$BINARY` The path to the binary, including its filename.
`$NAME` Any name that clearly identifies the binary, without extensions or dots.

Extract information from binary

```
../../scripts/dump_macho.sh ./wc wc
  ↪ Created wc.dump
    Created wc.data
    ...
```

Run FoxDec. The command-line usage for FoxDec is:

```
foxdec-exe $PDF $DIRNAME $NAME
```

`$PDF` Either `0` or `1`. Iff `1` then Graphviz is used to generate PDFs from .dot files. For larger examples we recommend `0`, as Graphviz may get stuck on large graphs.
`$DIRNAME` Name of directory where the files created above (e.g., `$NAME.dump`) are located.
`$NAME` Use the same name as previously used.

Run FoxDec

```
foxdec-exe 1 ./ wc
```

Observe Output. At this point, FoxDec will have generated output concerning the *control flow* of the program, the *function boundaries*, it will have generated *invariants* and *disassembled instructions*, etc. All of this information is stored in a `.report` file, which can be accessed through a Haskell interface (see Section **??**). For sake of convenience, some of this information is also outputted in humanly readable formats. First, in the file `./$NAME_calls.pdf` an extended call graph is generated. Section **??** contains information on all the results stored in this file. For each function entry `$f`, a subdirectory has been created, and a control flow graph is generated in the file `$f/$NAME.pdf`. An overview of all resolved indirections can be found in the file `$NAME.indirections`. Finally, for each function entry `$f` a log has been maintained providing information on the results per entry (file `$f/$NAME.log` and an overall log has been maintained in `$NAME.log`.

Observe output

```
less wc.log
less wc.indirections
open wc_calls.pdf
less 7c0/wc.log
open 7c0/wc.pdf
```

1.3 Accessing information from `.report` file

All information in the generated `.report` file can be accessed through an interface. Implementation details on that interface, providing the exact list of functions that can be used to access the `.report` file, can be found here:

`https://ssrg-vt.github.io/FoxDec/foxdec/docs/haddock/VerificationReportInterface.html`

We have created several applications that use this interface to extract information from a `.report` file and provide output. The greyed out applications are currently under development.

Application	Functionality
foxdec-disassembler-exe	Basic instruction disassembly
foxdec-functions-exe	Function Boundaries
foxdec-controlflow-exe	Control Flow
foxdec-invariants-exe	Invariants
foxdec-isabelle-exe	Isabelle Code Generation
foxdec-symbolizer-exe	Position Independent NASM Generation

Basic Disassembly. Provides an enumeration of all instructions of all functions encountered while running FoxDec.

Basic Disassembly

```
foxdec-disassembler-exe wc.report
  ↪ 870: XOR EBP, EBP 2
    872: MOV R9, RDX 3
    875: POP RSI 1
    876: MOV RDX, RSP 3
    879: AND RSP, 18446744073709551600 4
    ...
```

Function Boundaries. Provides a coarse overview of the function boundaries of all functions encountered while running FoxDec. Splits the address ranges of the instructions belonging to the functions into chunks and shows their boundaries.

Function Boundaries

```
foxdec-functions-exe wc.report
  ↪ Function entry: 9ff
    9ff-->a9b
    Function entry: aa3
    aa3-->b3f
    ...
```

Control Flow. Given an instruction address, provides an overapproximative bound on the set of next instruction addresses. In the example below, address `0xada` may jump to two next addresses.

Control Flow

```
foxdec-controlflow-exe wc.report 0xada
  ↪ ada --> [adc,afc]
```

Invariants. Given an instruction address, produce the invariant. In the example below, some registers have not been modified wrt. their original value (e.g., `rcx` and `rdx`). The stack frame below the stack pointer stores certain values, e.g., the original value of register `rbp` and of the lower 32 bits of register `rdi`. The return address at the top of the stack frame has not been modified. Register `rax` holds an unknown value, returned by function `vfprintf`.

Invariants

```
foxdec-invariants-exe wc.report 0x9d1
  ↪ Invariant at address 9d1
     RIP := 0x9d1
     RAX := Bot[c|vfprintf@GLIBC_2.2.5|]
     RCX := RSI_0
     RDX := RDX_0
     RDI := Bot[m|[0x202080, 8]_0|]
     RSI := RSI_0
     RSP := (RSP_0 - 40)
     [(RSP_0 - 8), 8] := RBP_0
     [(RSP_0 - 12), 4] := b32(RDI_0)
     [(RSP_0 - 24), 8] := RSI_0
     [(RSP_0 - 32), 8] := RDX_0
     ...
     flags set by CMP(DWORD PTR [RBP - 4],0)
```