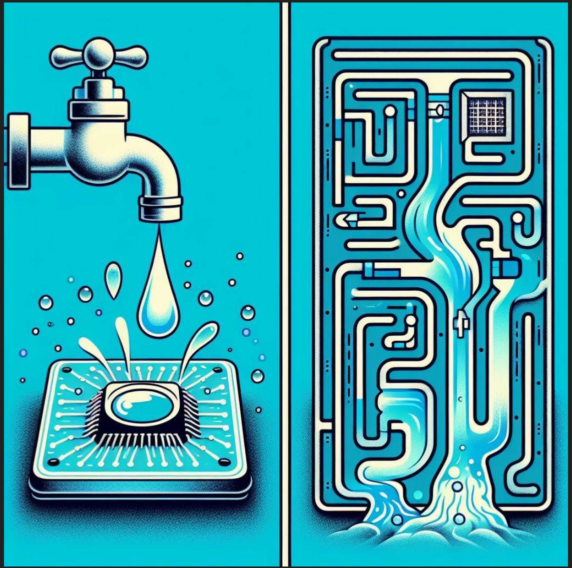
Reverse Engineering: Where to Start

Joshua Reynolds
NMSU Reverse Engineering
SP 2024

Four Ways to Get Started

1. Identify the source and sink - where does execution start and where you want to get to?
2. Work backwards from the sink towards the source
3. Read functions from the end back to the beginning
4. Use C standard library functions to figure out what variables contain

1. Sources and Sinks



Programs execution flow can be modeled with a **control flow graph**. Even for normal applications, this control flow graph is very large.

One RE technique to reduce the complexity is to focus only on a particular part of the larger graph. This is done by choosing a particular **source** (beginning point) and **sink** (ending point).

Source and Sink Example: Crackmes

What is the “source” we should choose?

What is the “sink” we should choose?

How could we find those in IDA or Ghidra?

Symbolic Execution

Tools exist to automatically trace a path from source to sink, keeping track of what inputs led to the right combination of IF-statements to arrive at the intended sync.

Rather than running code on inputs, a **symbolic execution engine** keeps track of the set of all possible inputs leading from a particular source to a particular sink.

This grows exponentially, so symbolic execution is limited to specific sources and sinks that are relatively close to one another.

Concolic Execution

To speed up symbolic execution, you can choose not to keep track of every input, and to let the program run normally until a specific analysis point. This speeds up execution.

The combination of **concrete execution** and **symbolic execution** is called “**concolic execution**”

2. Work backwards from sink to source

Often, diving into nested functions as in a depth-first-search allows you to figure out what is happening – again by starting at the end.

3. Reverse functions from End to Beginning

When reversing a function or subroutine, I find it helpful to work backwards from the end towards the beginning.

Which variable is being returned?

What other inputs affect that variable?

What other functions are called to set that variable?

4. Use library functions to figure out what variables contain

Most binaries link to library functions to perform basic functions.

In the Windows context, this is required because Windows makes breaking changes to its system calls, and provides compatibility through standard DLLs

Look for DLL or SO paths, and library function names in strings

If attackers are “living off the land”, they are also giving away what they are doing based on the tools they choose to load.

Activity 1 - Reverse basic crackme 1 from assignment

Activity 2 - Reverse first control-flow crackme from assignment