# Reversing with Ghidra

2022-05-09

# Disclaimer

- Reverse Engineering is a very personalized process.
- The things we will cover…
  - … are by no means the best method,
  - … nor the most efficient,
  - … nor will they always work for every use case.

# Today

- Reversing / solving a challenge based on real Malware.
- Exploring commonly used features of Ghidra.
- Discussing various strategies in Reverse Engineering.

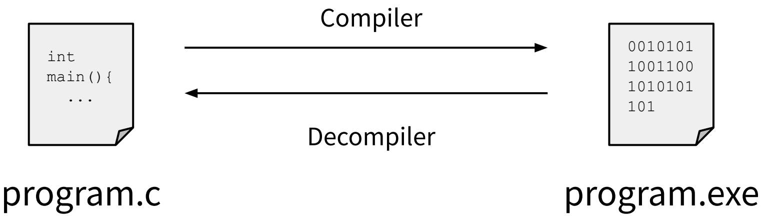
# Tools we will be using

- Wireshark (https://www.wireshark.org/)
- Ghidra (https://ghidra-sre.org/)
- Python (https://www.python.org/)

# What is Reverse Engineering?

# What is Reverse Engineering?

- Undoing what the compiler has done.

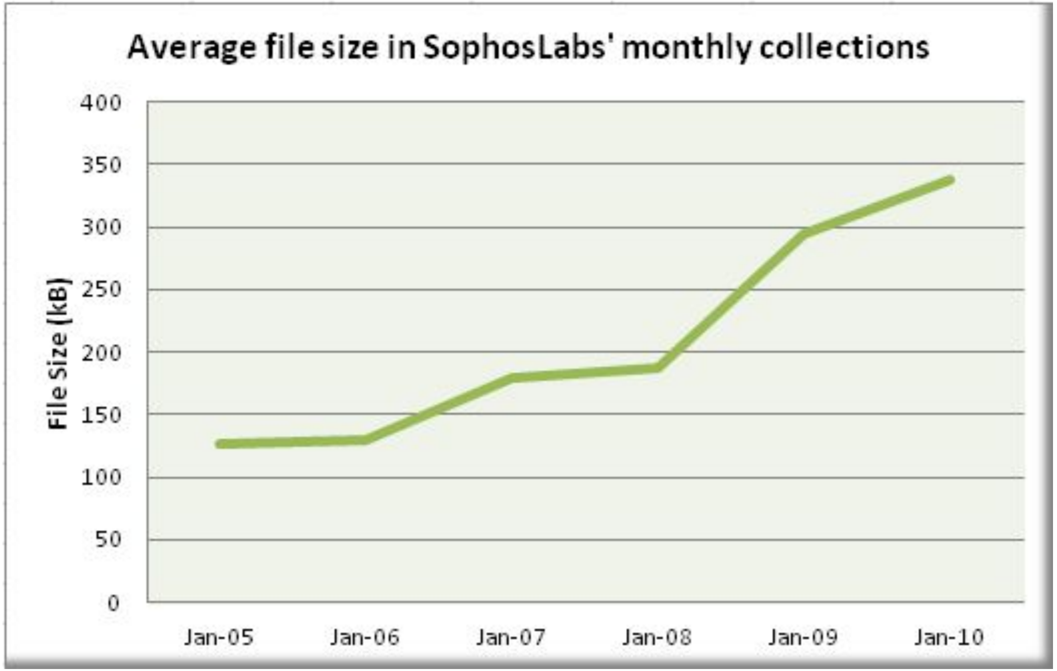


- Trying to understand what a program does (or has done).

# A problem

- Disassemblers and Decompilers are often wrong!
  - Compilers are better at applying optimizations than decompilers are at reverting them.
  - The process of decompilation approaches the Halting problem.
- Often decompilers need a little help …
  - Decompiler results often need to be cleaned up / corrected.

# A bigger problem



https://nakedsecurity.sophos.com/2010/07/27/large-piece-malware/

# A bigger problem

https://nakedsecurity.sophos.com/2010/07/27/large-piece-malware/

# Don't try to understand everything!

# Don’t try to understand everything!

- You won’t understand everything anyway.
- It is usually a waste of time to understand everything.
  - Majority of the code is standard library / boilerplate code.
  - Even the exact implementation of the relevant code is often irrelevant.
- High-level constructions are more important than implementation details.
- Focus on the end-goal (This can be difficult!)

# How do we know what to focus on?

# What is Reverse Engineering (really)?

- It’s all about pattern matching and making educated guesses!
    - Ask yourself: “What feature do I expect to be in this program?”
    - Imagine how it might be implemented (roughly).
    - Test your hypothesis by looking for evidence in the decompiler.
        - Strings?
        - Function calls?
        - Data structures?

# Let’s dive in