Abstract

This study examines the possibility of using a neural network based system to translate compiled executable binaries back to a human readable format. The main goal of the project is to map out possible avenues of approach and pitfalls for successive research projects. As such, some limitations were placed on this project. The source is simple C programs, and attempts have been made to use existing programs and tools when possible. The system takes C source files, and creates tokens that can be understood by a neural network. The tokens are then used to teach the neural network how to translate between the tokens and the associated compiled binary. The neural network then attempts to translate new executables. The human code was drawn from simple programs written for analysis, and compiled using GCC. ANTLR4 was used to facilitate the parsing of the program, and TensorFlow was used to build the neural network. Many concerns and issues were documented by the effort. One major concern for future is the suitability of existing tools. Current machine translation techniques and tools are not applicable for decompiling machine code. New techniques and tools need to be developed to support further efforts.

Methods

- ANTLR4 Java implementation using the Eclipse plugin
- VMWare with Ubuntu 14.04 was used as the development environment, along with Eclipse
- The neural network was attempted on Python 3.6 using TensorFlow 1.4 and the associated NMT machine translation model
- The binary is assembly generated directly by gcc 7.3.0

Example Parse Tree

```c
int hello = 0; // The previous line yields the following parse tree
```

Results

There are several important issues that need to be addressed in future attempts:

- **Parser**
  - Construction of a new parser is advised. The ANTLR4 framework is too rigid during tree processing to be a suitable candidate for an in-depth study.
  - How to handle preprocessor commands and conditional includes.
  - How to handle libraries. One possible solution is to keep a database of commonly used libraries for help in parsing.
- **Binary**
  - Whether to use the compiler created assembly, create a dissembler, or translate directly from binary.
  - What selection of compilers to use to generate the binary code.

Conclusion

Given the issues that occurred during this project, successive ones will need to thoroughly revamped the approach and implementation of a neural network based decompiler. Nearly every single section of the translation system will need to undergo massive revisions. Given that the reviewed tools did not meet the requirements of this project, entirely new parsers, a neural network translation system, a larger code base, and an automated readability checker need to be created or found.

Acknowledgements

I would like to thank my advisor Dr. Thomas Daniels for his assistance and guidance in this project. I would also like to thank Chinar Kaul for her assistance in gathering a library of C programs to use for training and testing the neural network.