Assembly Re-alignment Experiment Report

Mohamed Aziz Sghaier
Mohamed Radhi Halila

1 Introduction

The purpose of this experiment is to measure the effectiveness of a linear sweep disassembler in handling randomly chosen misaligned offsets in 32 and 64-bit executable code. We aimed to determine how often the disassembler realigns or encounters an invalid instruction when starting from these incorrect offsets. This analysis helps understand the resilience and accuracy of disassemblers in real-world scenarios where misaligned code might occur.

2 Methodology

1. **Executable Collection**: We collected a large number (153) of 32 and 64-bit executables from a github repository.
2. **Disassembly Setup**:
   - We used the Capstone disassembly framework to correctly disassemble the executables.
   - PE files were parsed to determine the architecture (32-bit or 64-bit).
3. **Random Offset Selection**:
   - For each executable, 1000 random offsets were chosen.
   - Disassembly was attempted from each offset.
4. **Data Collection**:
   - For each offset, we recorded the number of valid instructions encountered before hitting an invalid instruction.
   - Results were saved in a CSV file for further analysis.
5. **Analysis and Visualization**:
   - Data was analyzed to calculate average, maximum, and minimum numbers of valid instructions.
   - Plots were generated to visualize the distribution of valid instructions.

3 Results

3.1 Summary Statistics

- **Total Samples**: 153,261
- **Average Number of Valid Instructions**: 2,051.83
- **Maximum Number of Valid Instructions**: 217,626
- **Minimum Number of Valid Instructions**: 1

3.2 Visualizations

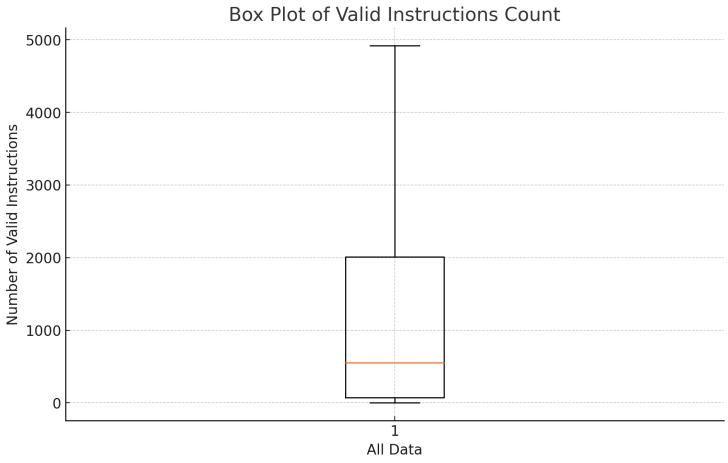

Figure 1: Box Plot of Valid Instructions Count

Observations:

- The median line is near the bottom of the box, suggesting that more than half of the offsets quickly lead to an invalid instruction.
- The interquartile range (IQR) is relatively small compared to the overall range, indicating that the majority of valid instruction counts are clustered close to the median.
- The whiskers extend far above the upper quartile, showing that there are a significant number of offsets that result in a much higher number of valid instructions before encountering an invalid one, although these are less common.

Observations:

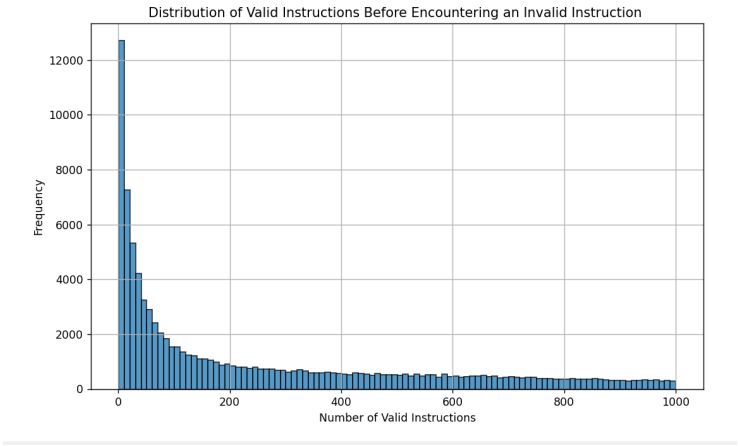

Figure 2: Histogram of Valid Instructions Count (Zoomed In)

- The histogram is heavily skewed to the left, with the highest frequency of valid instructions count being very low.
- The majority of offsets result in fewer than 100 valid instructions before encountering an invalid instruction.
- There is a rapid decrease in frequency as the number of valid instructions increases, with very few offsets resulting in more than 200 valid instructions.
- This plot confirms that most offsets quickly lead to an invalid instruction.

4 General Conclusions

1. **Predominant Early Invalid Instructions**:
   - The majority of the random offsets quickly lead to invalid instructions, as indicated by the heavy left skew in the histogram and the box plot's median being close to the lower quartile.
2. **Short Valid Instruction Sequences**:
   - For most offsets, the number of valid instructions encountered before hitting an invalid one is low. This is crucial for understanding how linear sweep disassemblers handle misaligned code.
3. **Rare Long Valid Instruction Sequences**:
   - Although rare, there are some offsets where the disassembler can process a high number of valid instructions. These outliers could be

due to sequences of data or padding that coincidentally align as valid instructions.

4. **Impact on Disassembler Design**:
    - These findings highlight the importance of incorporating mechanisms in disassemblers to detect and correct misalignment early, as misaligned offsets frequently lead to invalid instructions.

5 References

- Capstone disassembly framework: `http://www.capstone-engine.org/`
- GitHub repository for executables: `https://github.com/bormaa/Benign-NET`