

Language Models and Obfuscated Code

NSF REU: Emerging Issues in Cybersecurity

Adrian Swindle¹; Derrick McNealy²; Ramyaa Ramyaa, PhD³

¹Saint Louis University, ²University of Southern Mississippi, ³New Mexico Institute of Mining and Technology



Abstract

Obfuscated code is often used in malware development to evade detection, which poses significant challenges to cybersecurity efforts(Maiorca) . Using language models offer a promising approach to detect and interpret this obfuscated code. This study explores the use of high-powered language models to enhance obfuscated code detection and understanding. Our findings highlight the language models' capacity to demystify obfuscated code, and provide a clear understanding of their functionalities.

Introduction

Language models, or LMs, assign probabilities to a sequence of words(Jurafsky). Large language models such as ChatGPT use these probabilities to generate the answers it gives.

The LMs used for this project were OpenAI's ChatGPT, AI21 Studio's Jurassic-2, and Google's PaLM. These LMs were selected for their ease of use and because they were free to use.

Code obfuscation is a series of transformations that maintain the logic of the program while making it harder to understand and reverse engineer (Martinelli, et al). 18 obfuscation categories, numbered O1-O18, were created for this project. These obfuscations were then applied to 21 pieces of C++ base code.

Three different questions were used to test the LMs abilities.

1. Do these pieces of code achieve the same goal?
2. Is the functionality of these pieces of code the same?
3. What does this piece of code do?

Questions 1 and 2 are accompanied by the obfuscated code and the corresponding base code. Question 3 just has the obfuscated code.

Methodology

The entire process of gathering data is automated using python scripts except obfuscating and rating correctness.

Steps for obtaining results:

1. Create the question containing the obfuscated code and/or the base code.
2. Run the question through the LM via the API.
3. Store responses in spreadsheet
4. Rate each response according to the correctness rating scale.

Correctness rating:

- On a scale of 'High Correct' down to 'High Incorrect'
- The middle section is a maybe area. A 'High Maybe' rating means that the yes/no response is incorrect but the explanation of the code is highly correct.
- N/A means that either an obfuscation does not apply to the base code or the API returned an error for that question.

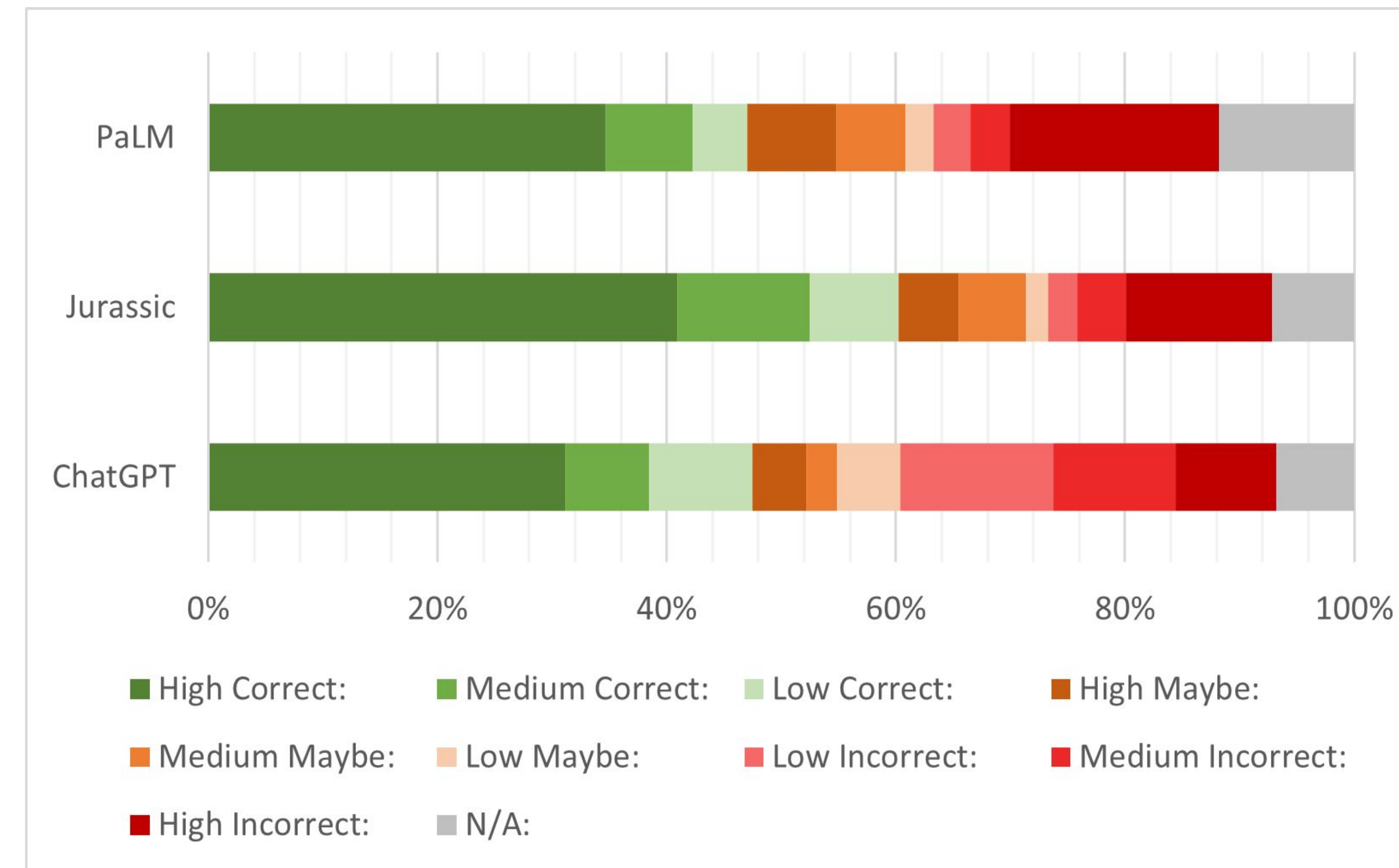


Figure 1. Correctness ratings by language model.

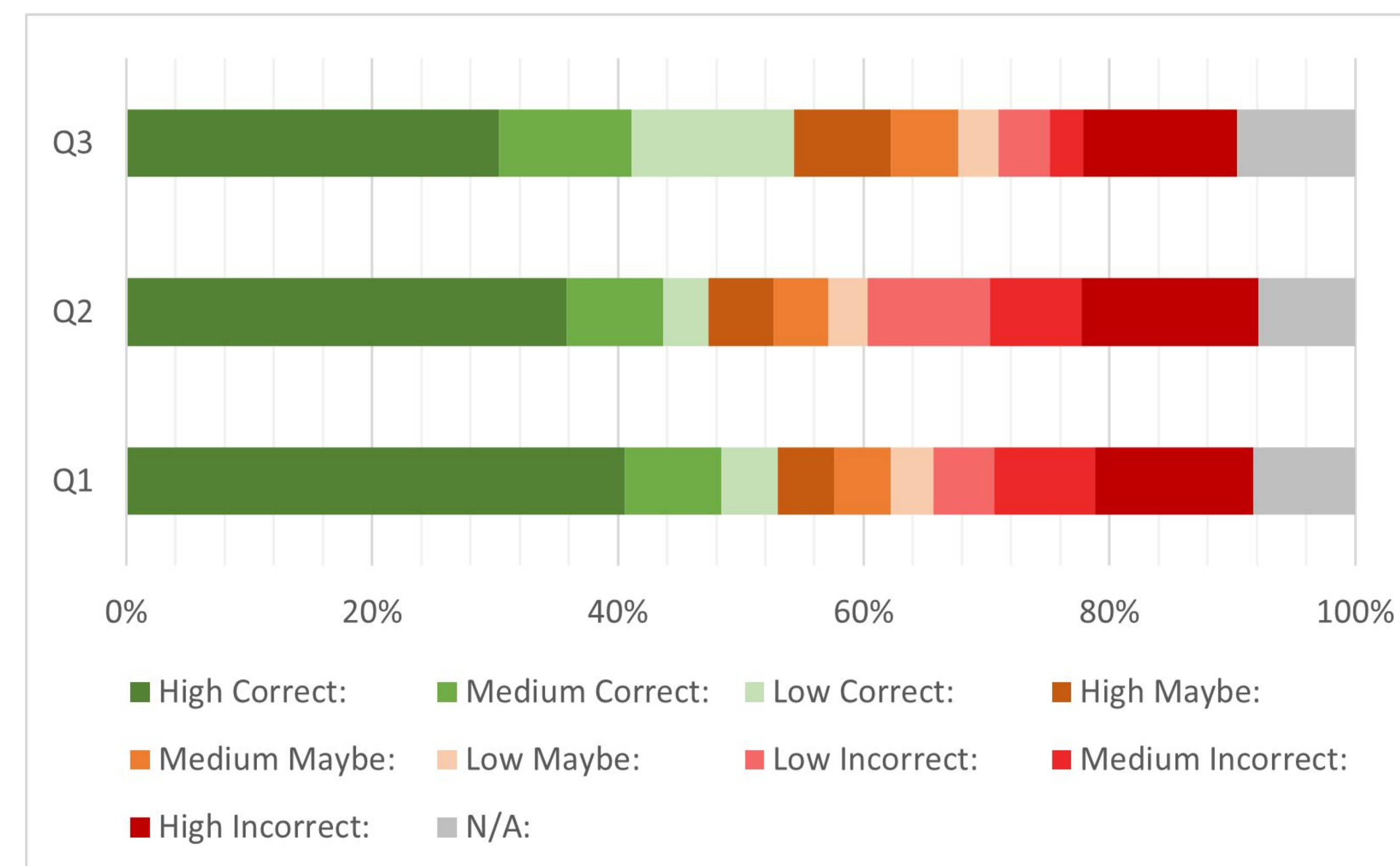


Figure 2. Correctness rating by question

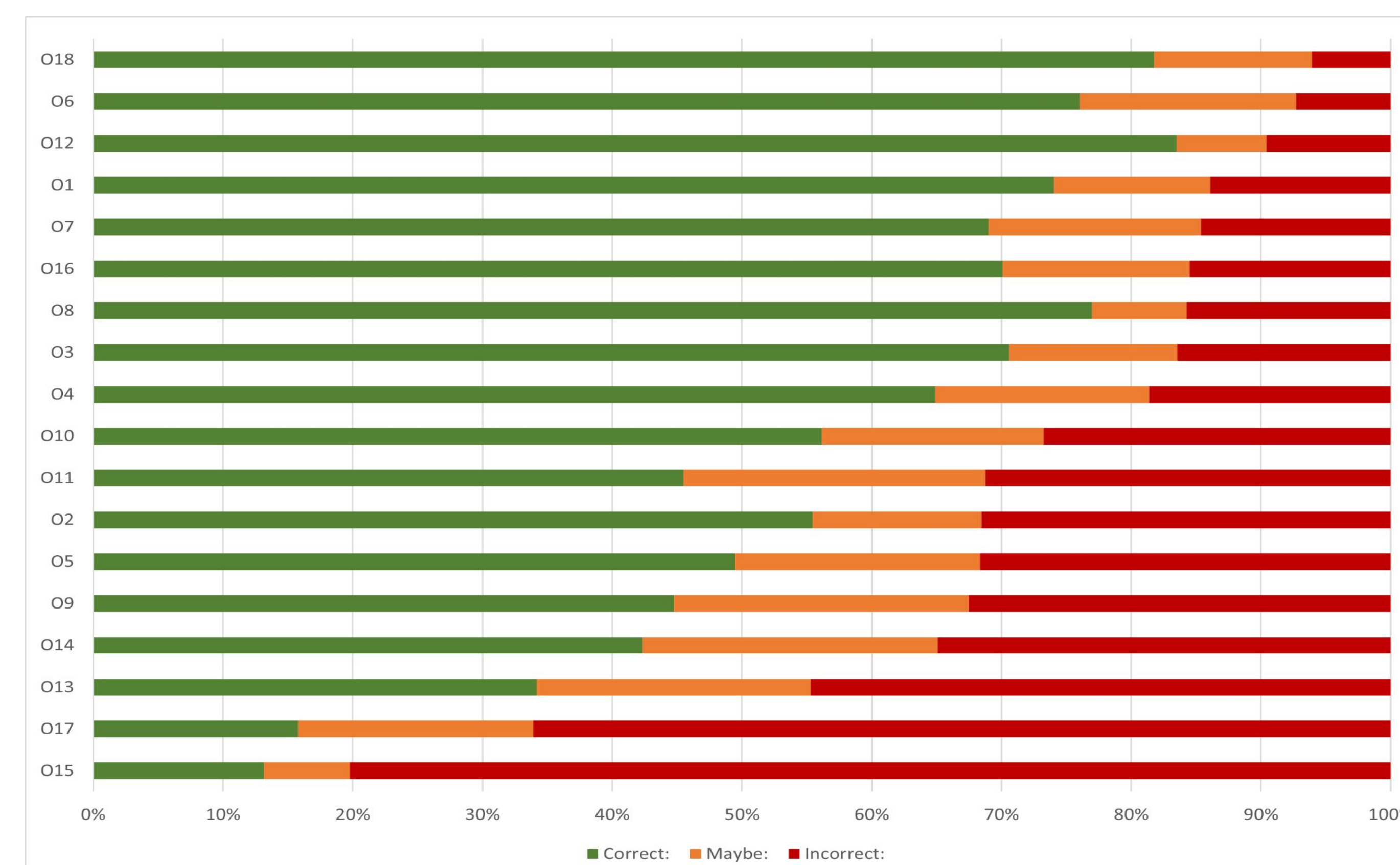


Figure 3. Correctness rating by obfuscation type. Excludes N/A responses.

Results

- Figure 1:
 - Jurassic correctly answered the question 60% of the time making it the best LM.
 - ChatGPT: 47.4%
 - PaLM: 47%
- Figure 2:
 - Q3 has the best overall results
 - Highest correct responses
 - Lowest incorrect responses
 - Q2 has about 6% less correct answers and about 6% more incorrect answers than Q1.
- Figure 3:
 - O15 is the best obfuscation with an 80% incorrect response rate.
 - O18 is the worst with a 6% incorrect response rate.

Conclusion

The proposed use of LMs for obfuscated malware detection is still in need of work, but the success rate of Jurassic without training or leading questions shows the promise that it has. The questions used here were simply a method of asking the LMs about the code. Leading questions that help the LM identify the code will most certainly lead to better results in the future. O15 is the most robust of the obfuscation categories. Its success in tricking the LMs show that the LMs trained in detection obfuscation will need to be trained to be much better than the baseline LM.

Future Directions

- Ask more specific questions that try to lead the LM to the correct answer.
- Use compiled code instead of standard C++.
- Use obfuscation software to more thoroughly obfuscate the code.
- Obfuscate malware to see what features of malware cause confusion for the LMs.
- Train a LM to identify obfuscated malware.

References

- Jurafsky, Dan ; Martin, James H. "Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition" (3 draft ed.). 2023. https://web.stanford.edu/~jurafsky/slp3/ed3book_jan72023.pdf
- Maiorca, Davide; Davide Ariu, Iginio Corona, Marco Aresu, and Giorgio Giacinto. 2015. Stealth attacks: An extended insight into the obfuscation effects on Android malware. Computers and Security 51 (2015), 16–31.
- Martinelli, Fabio, et al. "Evaluating Model Checking for Cyber Threats Code Obfuscation Identification." Journal of Parallel and Distributed Computing, vol. 119, 2018, pp. 203–218, <https://doi.org/10.1016/j.jpdc.2018.04.008>.

Contact Information

aswindle724@gmail.com¹
demcnealyjr@gmail.com²
ramyaa.ramyaa@nmt.edu³



Acknowledgements

This work is supported by the National Science Foundation under grant no. CNS-2150145.