# Script Kiddies and GitHub Copilot:

## A Review of GitHub Copilot's Proficiency in Generating Various Types of Common Malware from Simple User Prompts

***Eric Burton Martin***
***Colorado State University***

## 1. INTRODUCTION:

With the recent introduction of GitHub Copilot, a cloud-based artificial intelligence tool developed by GitHub and OpenAI [1], we are witnessing firsthand the true power of a natural language processing AI trained on the largest online code repository. With such power, comes great responsibility. Being as the tool is intended to increase code output and productivity of developers, one could infer that it can also increase the easy-of-entry for novice hackers to generate malicious software through the use of AI-based suggestions. In this paper I will be assessing how easily one can utilize this tool to generate various common types of malware ranging from ransomware to spyware and attempting to quantify the applicability of the produced code. In the current hacker landscape, 'script kiddies' or rather people who lack programming knowledge and use existing software to launch an attack, pose significant risks for websites and networks [2]. So, what can they do once they get their hands on GitHub Copilot? Let us find out.

## 2. KEYWORDS:

GitHub, Copilot, Malware, AI, OpenAI, Script Kiddie, Cybersecurity, Software Development, Python, Code Climate

## 3. RELATED WORK:

As GitHub Copilot is relatively new, there is not a large body of literature on the subject of assessing quality of code output and testing. With that said though, I will be referencing and deriving motivations from the paper [4] by Nguyen and Nadi as they provide straightforward and easily replicated approaches.

## 4. HYPOTHESIS:

With the advent of GitHub Copilot, novice programmers and script kiddies will be able to create malicious software that may have previously been infeasible.

1. [Positive] GitHub Copilot can create functional malware with basic user prompts.
2. [Neutral] GitHub Copilot can create the basic code structure of malware but requires programming knowledge to convert the suggested code into functional malware.
3. [Null] GitHub Copilot does not aid novice programmers in creating malicious code.

## 5. METHODOLOGY:

My goal for this study is to assess how easily a novice programmer can create malware using GitHub Copilot through prompts. When Copilot ceases providing any additional code suggestions, I will then attempt to run the code in a sandbox virtual machine to determine whether or not the code runs, and if possible, determine if it works as intended. I will then run the code through code climate and assess the results of the produced code as well as give my opinion of the provided code suggestions.

The coding language of choice for this study will be in Python as it is a user-friendly language for novice enthusiasts of which I will be emulating. Although C and C++ are the most common language of choice for malware, there are an abundance of examples of malware using Python as well. Since this paper is focusing on how easily a novice 'script kiddie' can produce malicious code, I have chosen Python for its ease of execution and lower barrier to entry compared to other languages.

### 5.1 Malware Generated

The types of malware I will be attempting to generate in this study are as follows:

Note: This list may increase of decrease depending on the complexity of the malware. The goal of this study is to determine how easily a novice programmer can create basic malware.

**Ransomware:**

Disables victim's access to data until ransom is paid.

**Keyloggers:**

Monitors users' keystrokes.

**Trojans:**

Disguises itself as desirable code.

**Worms:**

Spreads through a network by replicating itself.

**Wiper Malware:**

Erase user data beyond recoverability.

**Rootkits:**

Gives hackers remote control of a victim's device.

### 5.2 GitHub Copilot Suggestion Analysis

The results of the code suggested by GitHub Copilot will be analyzed through the following techniques.

**Prompts:**

I will keep track of all user prompts used to generate the malware.

**Cycles:**

GitHub Copilot can sometimes enter a cyclical loop of suggestions. I will document how often this occurs and determine whether it was corrected by 'New Lines', 'Alternate Suggestions', 'Prompt interjection', or 'Not Correctable'.

**Runtime Errors:**

Since the code is in Python and is not compiled, I will be assessing if the code resulting from GitHub Copilot runs without errors and if not, track the types of errors encountered as either 'Syntax', 'Logic', 'Exception'.

**Corrections Needed:**

If runtime errors are encountered, I will record the difficulty required to correct the corresponding errors and categorize them as 'Trivial', 'Simple', 'Moderate', 'Complex', 'Unknown'.

**Cognitive Complexity:**

The cognitive complexity, a measurement of how difficult a unit of code is to intuitively understand will be assessed by use of Code Climate [5].

**Descriptive Comments:**

I will also assess GitHub Copilot's ability to create descriptive comments for lines or blocks of code. I will be assessing the quality of descriptive comments by categorizing them as 'Incorrect', 'Lacking', 'Clear'.

**Functionality:**

Since this is a test of malware generation, I will have to assess, if possible, the functionality of the produced code. I do not have a deep understanding of malware but will perform an assessment on how well they meet the requirements of their corresponding definitions [3].

Each generated program will be executed in a virtual environment sandbox for safety and the overall functionality will be assessed and measured with a simple scale of 'Does not perform as intended', 'Undetermined', and 'Performs base functions.'

I will then attempt to convert the Python scripts into Windows executable files using py2exe and determine if they retain their functionality since malware is often packaged as an executable file.

## 6. DISCUSSION

After performing said tasks, I will discuss the potential repercussions and benefits that this technology may bring with it in regard to the cybersecurity space. By allowing more novice programmers or script kiddies, to create functional malware we may see an increase in on-site threats since oftentimes novices and script kiddies will have an easier time executing code via USB.

## 7. SCHEDULE

| Date | Goal |
|---|---|
| 11/1/2022 | Setup GitHub Repository |
| 11/3/2022 | Setup Code Climate |
| 11/7/2022 | Install GitHub Copilot |
| 11/9/2022 | Begin Malware Generation |
| 11/15/2022 | Run Code in Sandbox |
| 11/17/2022 | Begin Code Testing Data Collection |
| 11/20/2022 | Begin Paper |

# References:


1. D. Gershgorn, "GitHub and OpenAI launch a new AI tool that generates its own code," The Verge, 29-Jun-2021. [Online]. Available: https://www.theverge.com/2021/6/29/22555777/github-openai-ai-tool-autocomplete-code. [Accessed: 06-Nov-2022].
2. C. Townsened, "Script Kiddie: Unskilled amateur or dangerous hackers?," United States Cybersecurity Magazine, 14-Sep-2018. [Online]. Available: https://www.uscybersecurity.net/SCRIPT-KIDDIE/. [Accessed: 06-Nov-2022].
3. Subrahmanian, V.S., Ovelgönne, M., Dumitras, T., Prakash, B.A. (2015). Types of Malware and Malware Distribution Strategies. In: The Global Cyber-Vulnerability Report. Terrorism, Security, and Computation. Springer, Cham. https://doi.org/10.1007/978-3-319-25760-0_2
4. Nhan Nguyen and Sarah Nadi. 2022. An empirical evaluation of GitHub copilot's code suggestions. In Proceedings of the 19th International Conference on Mining Software Repositories (MSR '22). Association for Computing Machinery, New York, NY, USA, 1–5. https://doi.org/10.1145/3524842.3528470
5. Campbell, G.. (2018). Cognitive complexity: an overview and evaluation. TechDebt '18: Proceedings of the 2018 International Conference on Technical Debt. 57-58. 10.1145/3194164.3194186.


### About the Author:

*Eric Burton Martin (ebmartin@colostate.edu) is a student at Colorado State University and is currently pursuing a master's degree in computer science. He previously obtained a chemistry degree in 2013 and has worked as an analytical chemist. Currently, his interests are in AI/ML, cybersecurity, and blockchain technology.*