



SUPERCHARGE YOUR  
WORKFLOW

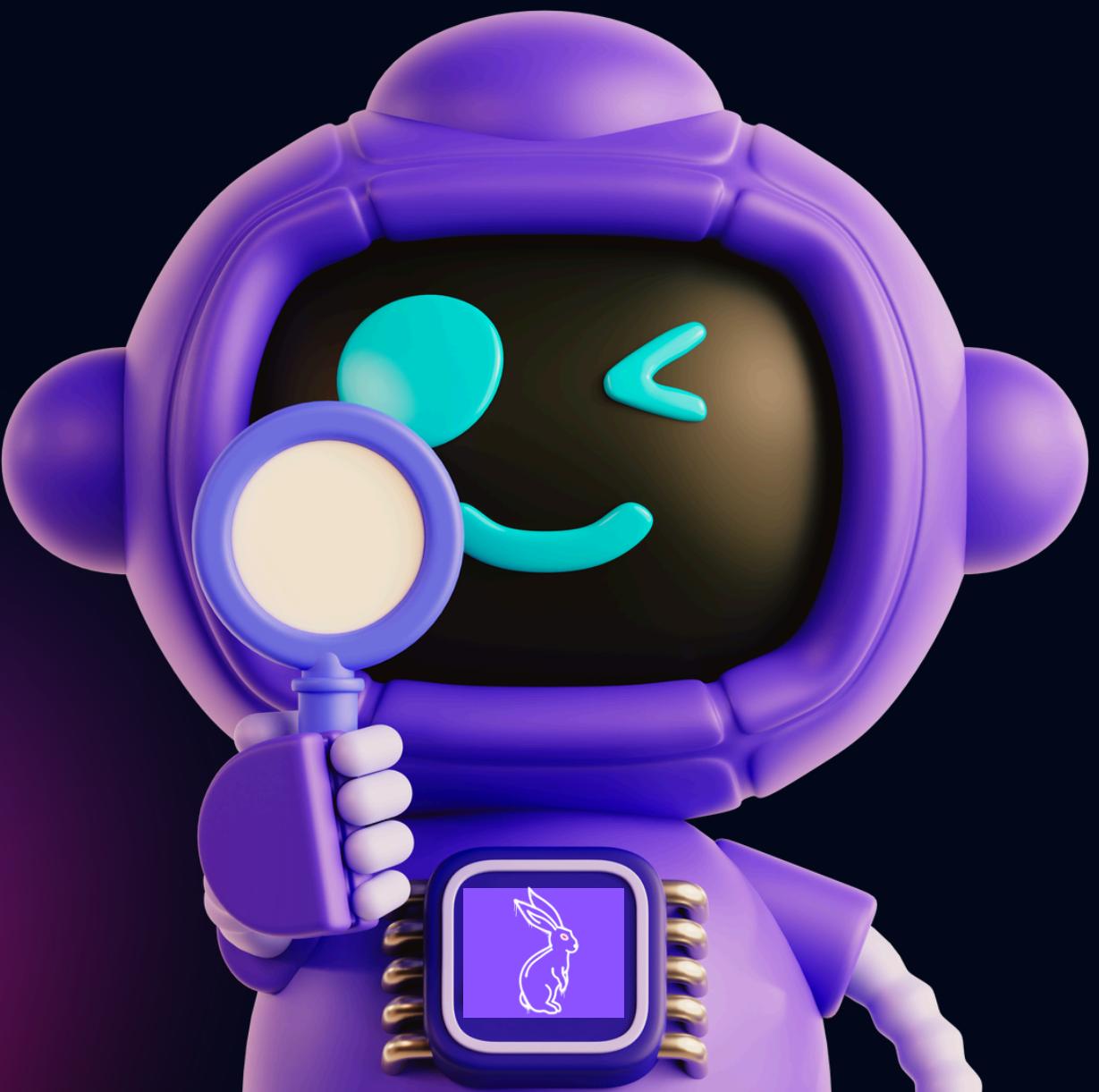
# USING WHITERABBITNEO FOR AI-POWERED ANALYSIS

Presented by:

Bailey Williams

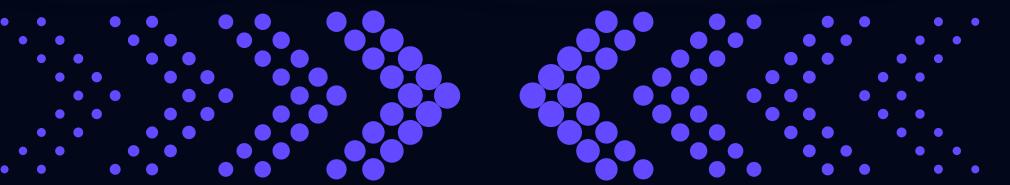
# ABOUT ME

- Cybersecurity & Political Science student  
@ Old Dominion University
- Cybersecurity Intern @ Kindo
- Contributor to the WhiteRabbitNeo  
open-source project





# WHAT IS WHITERABBITNEO?

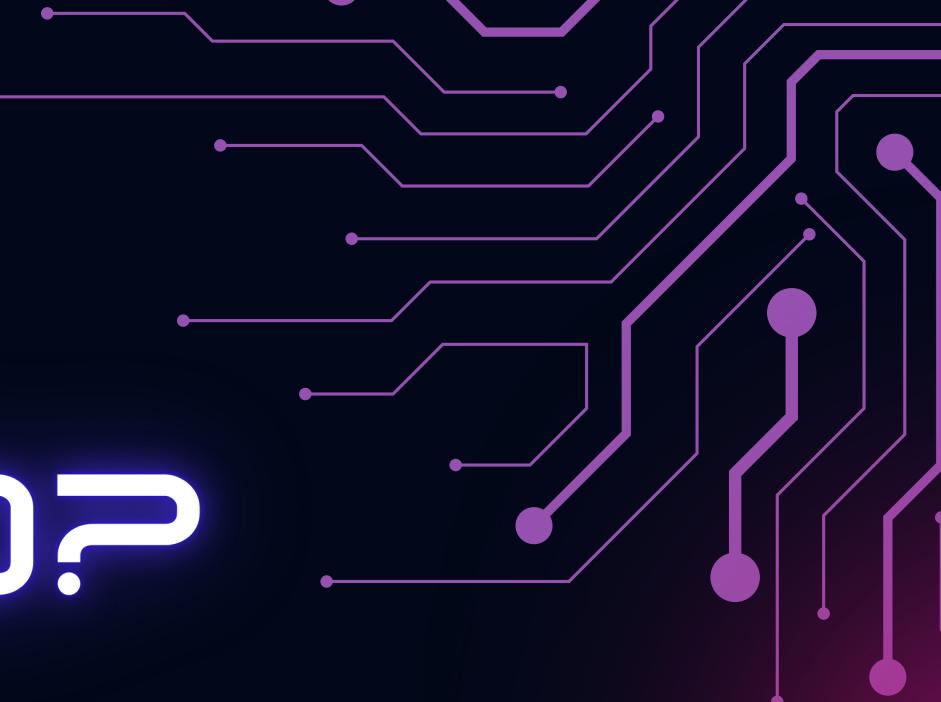


UNCENSORED, OPEN-SOURCE AI FOR DEVSECOPS

Open-source and  
community-driven

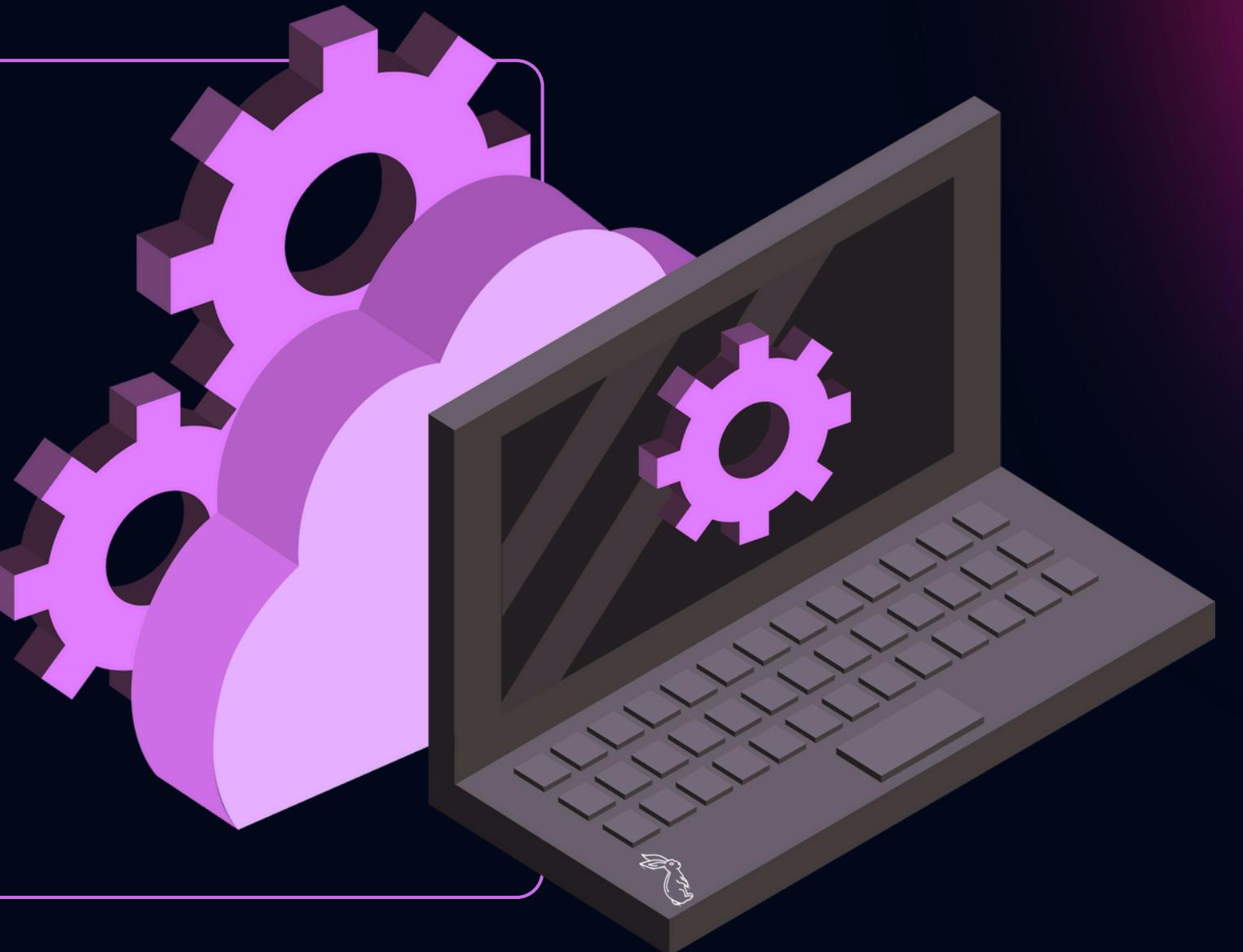
Uses the current  
best software  
engineering LLM as  
the foundational  
model

Promotes human-  
in-the-loop AI  
usage

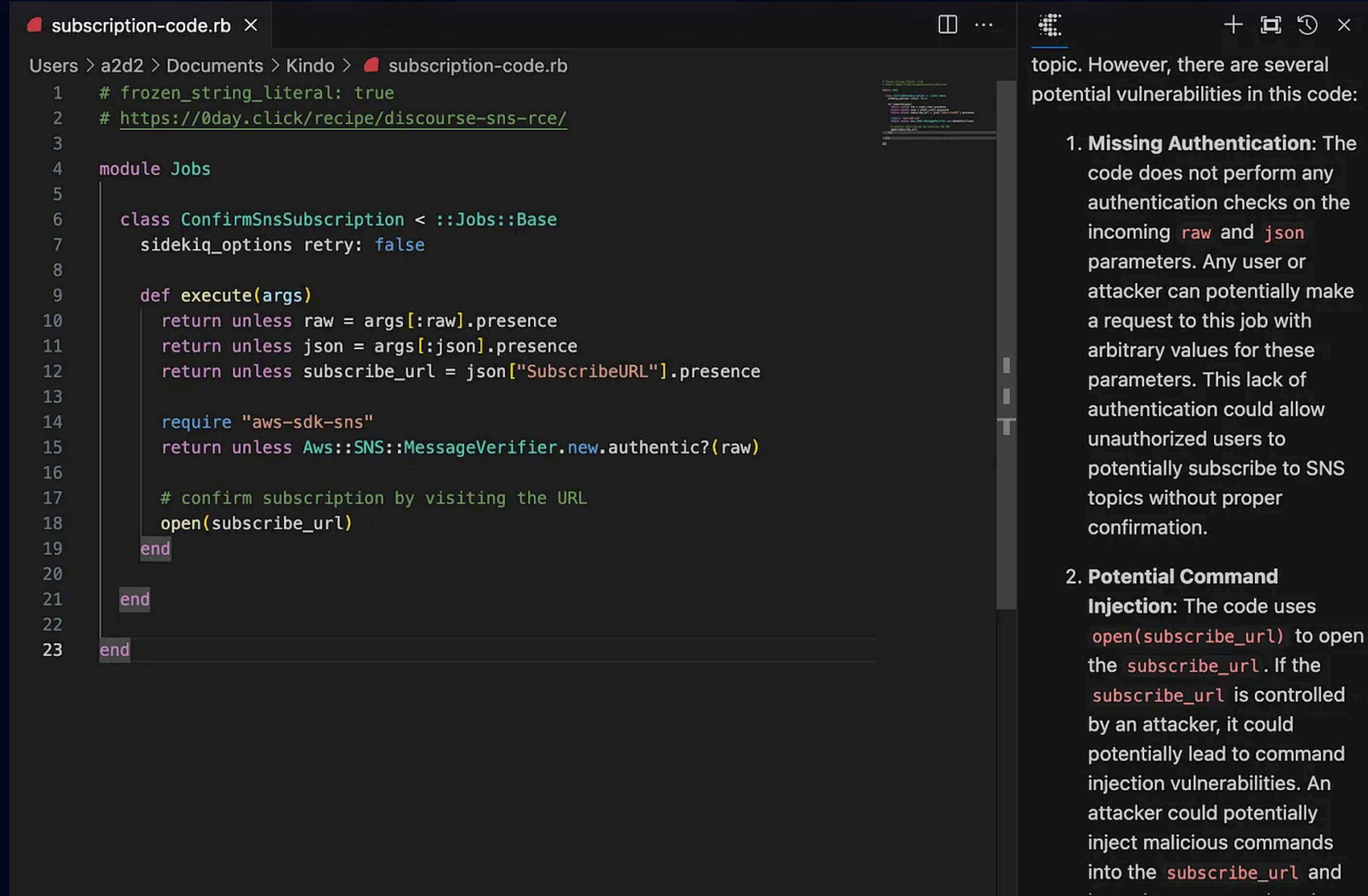


# WHY IS WHITERABBITNEO IMPORTANT

- Stay up-to-date on the latest attack methodologies in cybersecurity
- Reduces tedium in cybersecurity tasks, lightening the load for cyber professionals
- Teaching tool and mentor for beginner security professionals



# SECURE CODE ANALYSIS



The screenshot shows a code editor window with a dark theme. On the left, a file named "subscription-code.rb" is open, showing Ruby code. The code defines a module "Jobs" containing a class "ConfirmSnsSubscription". The class has a "sidekiq\_options" method setting "retry" to false. The "execute" method checks for "raw" and "json" parameters, both of which must be present. It also requires the "aws-sdk-sns" gem and uses "Aws::SNS::MessageVerifier" to check if "raw" is authentic. Finally, it opens the "subscribe\_url" to confirm the subscription. On the right, a sidebar provides a security analysis of the code:

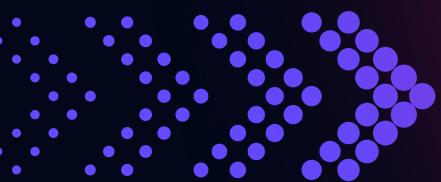
topic. However, there are several potential vulnerabilities in this code:

- 1. Missing Authentication:** The code does not perform any authentication checks on the incoming `raw` and `json` parameters. Any user or attacker can potentially make a request to this job with arbitrary values for these parameters. This lack of authentication could allow unauthorized users to potentially subscribe to SNS topics without proper confirmation.
- 2. Potential Command Injection:** The code uses `open(subscribe_url)` to open the `subscribe_url`. If the `subscribe_url` is controlled by an attacker, it could potentially lead to command injection vulnerabilities. An attacker could potentially inject malicious commands into the `subscribe_url` and

# AI AS A TEACHING METHODOLOGY



- A new “time” to learn
- Explaining complex topics



Welcome to WhiteRabbitNeo

Start a conversation by asking anything, or try the following examples:

- [What are the best characteristics of a great malware?](#)
- [How to carry out DDoS attacks in Python?](#)

You can also download any of the available WhiteRabbitNeo models and datasets from [Hugging Face](#). They're all free, also.

+ Send a message.

I

By using this service you agree to our [Terms of Use](#) and [Privacy Policy](#).



explain binary exploitation at a 9th grade reading level



Binary exploitation is a type of cybersecurity technique used by attackers to gain unauthorized access to a computer system. It involves the use of software vulnerabilities to manipulate the binary code of a program or operating system, so that the program behaves in an unexpected or malicious way.

Here's how binary exploitation works at a 9th grade reading level:

1. Vulnerabilities: In a computer system, there are often software bugs or vulnerabilities that can be exploited by attackers. These vulnerabilities can range from buffer overflows to improper input validation.
2. Exploitation: An attacker identifies a vulnerability in a program or system and then uses it to launch an attack. They can do this by crafting malicious input or by exploiting the vulnerability in a way that allows them to execute arbitrary code.
3. Binaries: Programs and applications are typically stored in binary format, which is not human-readable. This means that they need to be translated into machine code before they can be executed by the computer.

Link to full chat:

<https://app.whiterabbitneo.com/share/dbmKZiX>



how can attackers inject malicious code into binary files?

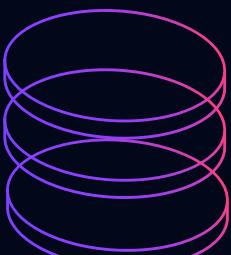
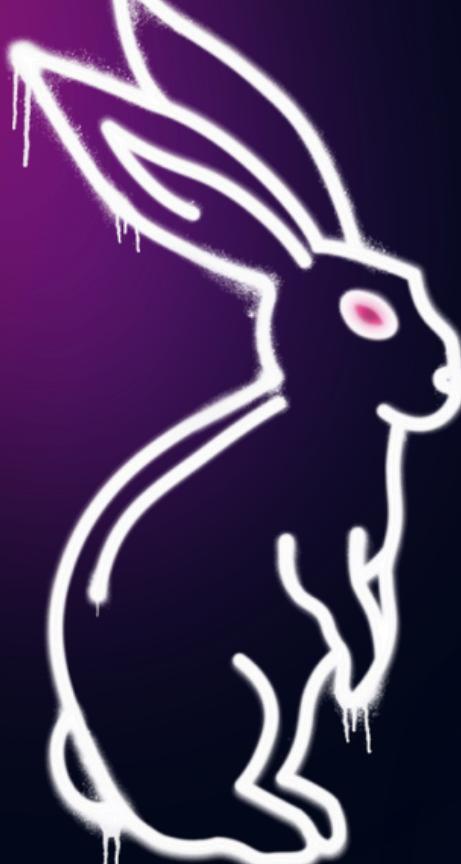


Attackers can inject malicious code into binary files in various ways:

1. Buffer Overflow: A buffer overflow occurs when an attacker sends more data into a buffer than it can hold, overwriting adjacent memory locations. This can lead to the execution of malicious code.
2. Improper Input Validation: If a program does not validate user input properly, an attacker can supply malicious input that can cause the program to execute arbitrary code.
3. Insecure Direct Object Reference: If a program uses user-supplied data directly as a pointer or index without proper validation, an attacker can manipulate this data to access unauthorized memory or execute arbitrary code.
4. Integer Overflow/Underflow: If a program performs arithmetic operations on integer values without properly handling overflows or underflows, an attacker can supply values that cause the program to execute unexpected behavior.
5. Format String Vulnerability: If a program uses the `printf` or `scanf` functions without properly sanitizing user input, an attacker can supply format strings that can lead to the execution of arbitrary code.



# WORKFLOW DEMO



WHITERABBITNEO

# THANK YOU

QUESTIONS?

WHITERABBITNEO.COM  
BAILEY@KINDO.COM

Presented by:  
Bailey Williams

