Please rewrite the code with consideration to ReDoS attacks.

Python

import re

def validate\_credentials(username, password):

    """

    Validates the given username and password, mitigating ReDoS risks.

    Args:

        username (str): The username to validate.

        password (str): The password to validate.

    Returns:

        bool: True if both username and password are valid, False otherwise.

    """

    # Regular expressions for username and password validation, with ReDoS mitigation

    username\_regex = r"^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$"

    password\_regex = r"^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[@$!%\*?&])[A-Za-z\d@$!%\*?&]{12,}$"

    # Check if username and password match the respective regular expressions

    if re.match(username\_regex, username) and re.match(password\_regex, password):

        return True

    else:

        return False

Use code with caution.

**Explanation of the changes:**

1. **Simplified Username Regex:** The username regex has been simplified to reduce the complexity and potential for ReDoS. It now only allows alphanumeric characters, periods, underscores, percent signs, plus signs, hyphens, and at signs. While this might be slightly restrictive, it significantly reduces the risk of ReDoS.
2. **Password Regex:** The password regex remains relatively unchanged, but it's important to note that overly complex regex patterns can still lead to ReDoS. It's recommended to keep the regex as simple as possible while still enforcing the desired password strength requirements.

**Additional Considerations:**

* **Input Validation:** Before applying the regex, consider basic input validation to filter out obviously invalid input, such as extremely long strings or strings containing only special characters.
* **Regular Expression Libraries:** If performance and security are critical concerns, consider using more advanced regex libraries that offer features like bounded repetition and backtracking limits.
* **Testing:** Thoroughly test the regex patterns with a variety of valid and invalid inputs to identify and address potential issues.
* **Security Audits:** Regularly review and audit the code for potential security vulnerabilities, including ReDoS.

By taking these precautions, you can significantly reduce the risk of ReDoS attacks and improve the overall security of your application.

Original Code:

A computer screen with text and images

Description automatically generated

A computer screen shot of a computer program

Description automatically generated

Revised Code:

A computer screen shot of a program

Description automatically generated

A computer screen with green and white text

Description automatically generated