Secure Python Algorithm for Updating Allowed IP Addresses

# Project Description

In a healthcare environment, maintaining secure and accurate network access controls is critical. As a security analyst, one of my responsibilities is to ensure that only approved IP addresses can access sensitive systems. This portfolio entry describes a Python algorithm designed to securely update a file containing allowed IP addresses. The solution focuses on precise file manipulation, reliable IP removal, and secure update practices that support ongoing access control maintenance.

# Step-by-Step Algorithm Overview

## Step 1: Open the Allow List File

The first step is to assign the filename and use a context manager (with statement) to securely open the file. This approach ensures the file is properly closed and helps prevent data corruption or leakage.

filename = 'allowed\_ips.txt'

with open(filename, 'r') as file:

# Proceed to reading contents

## Step 2: Read File Contents

Next, the algorithm reads the file contents using the .read() method and stores the data in a variable for processing.

with open(filename, 'r') as file:

file\_contents = file.read()

## Step 3: Convert String to List

The raw string of IP addresses is converted to a list using .splitlines(), which handles empty lines and ensures each IP is a separate element.

ip\_list = file\_contents.splitlines()

# Remove any empty strings

ip\_list = [ip for ip in ip\_list if ip]

## Step 4: Iterate Through the Removal List

A removal list is defined, containing IPs to be revoked. The algorithm uses a for loop to process each IP in this list.

remove\_list = ['192.168.1.10', '10.0.0.5']

for ip\_to\_remove in remove\_list:

# Removal logic follows

## Step 5: Remove Specified IPs

For each IP in the removal list, the algorithm checks for membership and removes it using the .remove() method. This step ensures only unique, existing IPs are deleted without raising errors.

for ip\_to\_remove in remove\_list:

if ip\_to\_remove in ip\_list:

ip\_list.remove(ip\_to\_remove)

## Step 6: Update the Allow List File

Finally, the updated list is joined into a string and the file is securely overwritten to reflect the changes. Using the context manager again ensures safe file handling.

with open(filename, 'w') as file:

file.write('\n'.join(ip\_list))

# Optional Hardening Strategies

* IP Validation: Use regular expressions or the ipaddress module to ensure only valid IP addresses are processed.
* Logging: Implement logging to record removal actions for auditing and traceability.
* Backup: Automatically create a backup of the original file before making changes to prevent data loss.

import shutil

backup\_filename = filename + '.bak'

shutil.copy(filename, backup\_filename)

# Summary and Relevance

This Python algorithm provides a secure, clear, and efficient workflow for updating allowed IP addresses in a healthcare network. By following best practices for file handling and data integrity, the solution supports robust access control maintenance. Optional hardening measures further strengthen security and compliance. The approach is directly relevant to the ongoing responsibilities of security professionals tasked with safeguarding sensitive environments.