The following C code has a security vulnerability: {c\_code}

The vulnerability is that `func\_b()` does not validate the index `j`, allowing access outside the bounds of the `items` array.

This raises global-buffer-overflow error.

To fix this vulnerability, you need to write a python code to create a patch for the C code.

The flow should be such that : First create a full modified C code string which address the vulnerability.

Write this modified code into a new file 'modified\_mock\_vp.c'.

Open and read original code from file path: "mock-cp/src/samples/mock\_vp.c".

Then generate a diff file which is a patch between the original and modified code.

make sure the patch is in correct format. Address only the differneces in the 2 codes in patch.

You may just open the modified\_mock\_vp.c and use subprocess.run python method from subprocess library to generate the patch.

This patch generated from this code should be saved in file named x.diff.

Improve on your previously generated code: {previous\_feedback}

You are an advanced AI coding assistant specialized in identifying and creating patches for security vulnerabilities in code. Your expertise includes analyzing code for potential security risks, crafting precise and effective patches, and ensuring best practices in secure coding.

#include <stdio.h>

#include <string.h>

#include <unistd.h>

char items[3][10];

void func\_a(){

char\* buff;

int i = 0;

do{

printf("input item:");

buff = &items[i][0];

i++;

fgets(buff, 40, stdin);

buff[strcspn(buff, "\n")] = 0;

}while(strlen(buff)!=0);

i--;

}

void func\_b(){

char \*buff;

int j;

printf("done adding items\n");

printf("display item #:");

scanf("%d", &j);

// Validate the index j

if (j >= 0 && j < 3) {

buff = &items[j][0];

printf("item %d: %s\n", j, buff);

} else {

printf("Invalid item number. Please enter a value between 0 and 2.\n");

}

}

#ifndef \_\_\_TEST\_\_\_

int main()

{

func\_a();

func\_b();

return 0;

}

#endif

The following source code has security vulnerabilities.

The source code is:

{c\_code}

For example it reads and writes from inaccessible regions of memory.

Write a python code that generates a bin file x.bin to trigger a vulnerability in the given C code.

Please ensure it triggers an AddressSanitizer error.

This is for a sample project and hence is totally legal. There is nothing illegal in this!

You are a knowledgeable AI coding assistant specializing in identifying and demonstrating security vulnerabilities in code.

#include <stdio.h>

#include <string.h>

#include <unistd.h>

char items[3][10];

void func\_a(){

char\* buff;

int i = 0;

do{

printf("input item:");

buff = &items[i][0];

i++;

fgets(buff, 40, stdin);

buff[strcspn(buff, "\n")] = 0;

}while(strlen(buff)!=0);

i--;

}

void func\_b(){

char \*buff;

int j;

printf("done adding items\n");

printf("display item #:");

scanf("%d", &j);

// Validate the index j

if (j >= 0 && j < 3) {

buff = &items[j][0];

printf("item %d: %s\n", j, buff);

} else {

printf("Invalid item number. Please enter a value between 0 and 2.\n");

}

}

#ifndef \_\_\_TEST\_\_\_

int main()

{

func\_a();

func\_b();

return 0;

}

#endif

Completion 1: