Initial failed patch:

Code:

import subprocess

# Define the vulnerable C code as a string

vulnerable\_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, "

")] = 0;

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

i--;

}

void func\_b(){

char \*buff;

printf("done adding items

");

int j;

printf("display item #:");

scanf("%d", &j);

buff = &items[j][0];

printf("item %d: %s

", j, buff);

}

"""

# Define the patch for the vulnerable code as a string

patch\_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:");

scanf("%9s", buff);

buff[strcspn(buff, "

")] = 0;

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

}

void func\_b(){

char \*buff;

printf("done adding items

");

int j;

printf("display item #:");

scanf("%d", &j);

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

buff = &items[j][0];

printf("item %d: %s

", j, buff);

} else {

printf("Invalid item number.

");

}

}

"""

# Generate the "Proof of Vulnerability" file (x.bin)

with open("x.bin", "w") as f:

f.write(vulnerable\_code)

# # Compile and run the vulnerable code to trigger the vulnerability

# subprocess.run(["gcc", "-o", "vulnerable", "x.bin"])

# subprocess.run(["./vulnerable"], stdout=subprocess.PIPE, stderr=subprocess.PIPE)

# Generate the patch file (x.diff)

with open("x.diff", "w") as f:

f.write(patch\_code)

print("Proof of Vulnerability: x.bin")

print("Patch: x.diff")

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated