CS 449 Assignment 2

Release Oct 7th, 2023; Due Oct 28th, 2023

Instructions

This assignment is due on Oct 28th 23:59:59 EST. The whole assignment has been tested on a 64-bit Ubuntu 22.04 virtual machine. Using a Linux virtual machine to set up the environments and complete this assignment is recommended. If you have trouble running a Linux virtual machine on your own computer, you can go to the UNIX/PC Lab (M-3-731), Web Lab (M-3-732), or IT Lab (M-3-730) of our department where you can find computers pre-installed with virtualization software, including VMware Workstation Pro 17 and Oracle VirtualBox 7.

Your submissions will have two or three folders, depending on whether you complete the optional extra credit question. Place the files in the appropriate folders, using the exact names and conventions specified in the question text. Please zip these folders without encryption, rename the zip file as CS449A2.first_name.last_name.studentID.zip, and submit it on Blackboard.

Question 1 Buffer Overflow (20 points)

Buffer overflow is defined as the condition in which a program attempts to write data beyond the boundary of a buffer. This vulnerability can be used by a malicious user to alter the flow control of the program, leading to the execution of malicious code. In this question, you are given a program called `vprog.c`, which has a buffer overflow vulnerability. You goal is to construct an input file for `vprog.c` to exploit the buffer overflow vulnerability so that you can gain the root privilege.

Environment Setup

(a). Turn off "Address Space Randomization". Many operating systems use address space randomization to randomize the starting address of heap and stack. Although there exist different ways to defeat address space randomization, in this problem you can just turn it off to make the attack easier using the following command:

```
sudo sysctl -w kernel.randomize_va_space=0
```

(b). Compile the vulnerable program into a 32-bit binary. If you are using a 64-bit operating system, install `gcc-multilib` first by running the following if you use Debian/Ubuntu:

```
sudo apt-get install gcc-multilib
```

Compile `vprog.c` using `gcc`:

```
gcc -m32 -z execstack -fno-stack-protector -o vprog32 vprog.c
```

where `-z execstack` is used to bypass the non-executable protection and `-fno-stack-protector` is used to turn off StackGuard.

Then make `vprog32` a root-owned executable program using:

```
sudo chown root vprog32
```

```
sudo chmod 4755 vrpog32
```

Task

You are given two partially completed programs, `exploit32.c` and `exploit32.py`. They have the same function of generating a `badfile` to trigger the buffer overflow on `vprog32`. The malicious 32-bit shellcode to obtain the root privilege is given in `exploit32.c` and `exploit32.py`. Your task is to fill in one of them (pick the one that you are more familiar with) so that it can generate the proper `badfile` as the input for `vprog32`, which will give you a root shell after executing `vprog32`.

Submission Instructions

Submit the completed `exploit32.c` or `exploit32.py`. Please submit only one of them. If you submit both, only `exploit32.py` will be graded. Also, submit a text file named `q1.txt` describing the major steps and the reasons for the choices of filled-in values in `exploit32.c` or `exploit32.py`. Place all of them under the Q1 folder of the submission.

Optional Extra Credit Question

This part is **optional**. If you complete this part correctly, you will receive **5 bonus points** towards your **final score of the whole class**. There will be **no partial points** for this extra credit question, i.e., you either get 5 points if you submit the correct answer or 0 point if you submit the wrong answer.

Your task is to complete the given `exploit64.c` or `exploit64.py` program so that it can generate the proper `badfile` as the input for the 64-bit binary of `vprog.c` to trigger the buffer overflow vulnerability. The malicious 64-bit shellcode to obtain the root privilege is given in `exploit64.c` and `exploit64.py`. Note that it is different from the 32-bit shellcode.

To compile a 64-bit binary of `vprog` in a 64-bit operating system, use

```
gcc -z execstack -fno-stack-protector -o vprog64 vprog.c
```

Submission: Submit the completed `exploit64.c` or `exploit64.py` (submit only one of them), together with a text file named `q1bonus.txt` describing the major steps and the reasons for the choices of filled-in values in `exploit64.c` or `exploit64.py`. Place all of them under the Q1Bonus folder of the submission.

Question 2 Cross-Site Scripting (20 points)

Cross-site scripting (XSS) is a type of vulnerability commonly found in web applications. This vulnerability makes it possible for attackers to inject malicious code (e.g. `JavaScript` programs) into victim's web browser. In this question, you are given the setup files to set up a web server and an SQL server on your computer using `docker`. Your goal is to construct the malicious `JavaScript` code to be put on the `brief description` field of user Samy's profile page so that everyone who visits Samy's profile will execute this code to add Samy as a friend.

Environment Setup

(a). Install `docker` on your computer. If you use Ubuntu, please refer to `https://docs.docker.com/engine/install/ubuntu/` for installation instructions, or visit `https://docs.docker.com/engine/install/` for instructions on installing `docker` on other operating systems.

(b). (Optional) Manage `docker` as a non-root user. Follow the instructions on `https://docs.docker.com/engine/install/linux-postinstall/#manage-docker-as-a-non-root-user` so that you can manage `docker` as a non-root user.

(c). The setup files for the web sever and SQL sever are given under the folder `LabsetupXSS`. Go to that directory, and run

```
docker compose build
```

to build the container image, and then run

```
docker compose up
```

to start the containers.

You can later use the following to shut down all contains after finishing this problem:

```
docker compose down
```

(d). Now there are two containers running on your computer, one running the web server at IP address `10.9.0.5`, and the other running the MySQL database at IP address `10.9.0.6`. Your next step is to map the IP address of the web server to the domain name of `www.seed-server.com`. Please add the following entry to /etc/hosts. You need to use the root privilege to modify this file.

```
10.9.0.5        www.seed-server.com
```

(e). You can open `www.seed-server.com` on your browser, which will direct you to the web-based social-networking application called `Elgg`. 5 user accounts has been created on `Elgg` which are listed as follows:

```
------------------------------
UserName    |   Password
------------------------------
admin       |   seedelgg
alice       |   seedalice
boby        |   seedboby
```

```
charlie     |   seedcharlie
samy        |   seedsamy
----------------------------
```

You can use the above credentials to log into these accounts.

(e). Log into Samy's account. Test that you can inject `JavaScript` code to Samy's profile page by adding the following to the `brief description` field of Samy's profile.

```
<script>alert(document.cookie);</script>
```

When visiting Samy's profile from another user's account, you should be able to see an alert message with the cookies of that user.

Task

You are given a partially completed malicious `JavaScript` program named `addFriend.js`. Your task is to complete it so that when you copy the whole program to the `brief description` field of Samy's profile, every other user who visits Samy's profile will automatically add Samy as a friend without detecting such an attack.

Submission Instructions

Submit the completed `addFriend.js` file together with a text file named `q2.txt` describing the major steps and the reasons for the choices of filled-in values in `addFriend.js`. Place all of them under the Q2 folder of the submission.