CS253 Project 1

Due: 23:59 October 8

The goal of this assignment is to gain hands-on experience finding vulnerabilities in code and mounting buffer overflow attacks. In this project, you are given the source code for two exploitable programs which are to be installed with setuid root in a virtual machine we provide. For each target, you'll have first identify a vulnerability (buffer overflow, integer overflow, etc) in each program. Next, you'll write an exploit that takes advantage of that vulnerability. Running a successful exploit as an unprivileged user, will yield a root shell even though we didn't run our xploit as root!

The Environment

You'll run your exploits in a virtual machine (VM) emulated using QEMU. This serves two purposes. First, the vulnerable programs contain real, exploitable vulnerabilities and we strongly advise against installing them with setuid root on your machine. Second, everything from the particular compiler version, to the operating system and installed library versions will affect the exact location of code on the stack. The VM provides an identical environment. The VM is configured with Debian GNU/Linux 11 (bullseye), with ASLR (address randomization) turned off. It has a single user account "user" with password "cs253", but you can temporarily become the root user using sudo. The exploits will be run as "user" and should yield a command line shell (/bin/sh) running as "root".

1 Setting up your VM

1. You will first need to install QEMU. Follow the instructions on QEMU's website here (`https://www.qemu.org/download/`). MacOS users will need to first install Homebrew. You can find instructions on doing that here (`https://brew.sh/`).
2. Download the VM from the Shanghaitech epan `https://epan.shanghaitech.edu.cn/l/vF3iKt` (it's called "VM box").
3. Extract the tarball. You can usually do this by simply trying to open it. If you're on windows, you might have to right click on the download and select extract from there. The tarball contains a file disk.img, which is an QEMU QCOW Disk Image, and a couple helpful scripts (`run_qemu.sh` and `ssh_to_qemu.sh`).
4. Launch the machine using the included run qemu.sh script. The script just invokes the following line

```
qemu-system-x86_64 disk.img -m 2G -nic user,hostfwd=tcp::5555-:22 -nographic
```

   - The `-m 2G` option launches the VM with 2 Gigabytes of memory.
   - `-nic user, hostfwd=tcp::5555-:22` option forwards internet traffic on your computer's port 5555 to the VM's port 22. This allows you to ssh into your VM.
   - The `-nographic` option makes QEMU forward the output of your VM to your terminal. You can launch your VM without this option but we don't recommend it. If you do, QEMu will create a new window for your VM. See QEMU's documentation for more information on this site. `https://www.qemu.org/docs/master/`
5. Once the VM has booted, login with username "user" and password "cs253". It may take a minute or two for your VM to launch.
   - Your home directory contain the folder "proj1" which has the targets and starter code for the project. You can also find copies of the code on piazza.

6. The VM comes with a set of tools pre-installed (curl, wget, openssh, gcc, vim etc), but feel free to install additional software. For example, to install the emacs editor, you can run:

```
$ sudo apt-get install emacs
```

2 Using your VM

After opening and logging into your VM, you can directly edit the files with your text editor of choice (ex. nano, vim, emacs, etc). However, you may have a smoother terminal experience using your VM over ssh because qemu doesn't use your entire terminal window. The run qemu.sh script starts the vm and forwards traffic on your machine's port 5555 to the VM's port 22. We can ssh into the vm with the following:

1. Before moving on, make sure your vm is running and you've logged into it.
2. Open a new terminal window. This new window needs to be running on your local machine (it should not be running on the VM).
3. Now you can run `ssh_to_qemu.sh` and enter in "cs253" as the password. You should be logged in to the VM from your second terminal window.
4. Now we need to build and install the targets:

```
$ cd proj1/targets
$ make && sudo make install
password: cs253
```

   This will compile all of the target programs, set the executable stack flag on each of the resulting executables, and install them with setuid root in `/tmp`.
   Note: When you reboot your machine, files in the `/tmp` directory will be automatically deleted. This means you need to redo this everytime you restart the vm.
5. Write, build and test your exploits:

```
$ cd ../xploits
...edit,test...
$ make
$ ./xploit1
```

   The header file `shellcode.h`, provides a shellcode in the static variable `static const char* shellcode`.
6. When you're done, properly shutoff the vm with

```
$ sudo systemctl poweroff
```

   Not properly shutting down may cause problems the next time you open the vm.

3 Attack

Both exploits will take an in depth understanding of how functions get called and returned from and how that interacts with the rsp (stack pointer), rbp (base pointer), and rip (instruction pointer). You will also need to use **gdb** to find a variety of memory addresses on the stack (see Stanford CS 107's guide to gdb).

Successful Exploit

Successful exploits will result in a shell with root access:

```
user@cs253:~/proj1$ ./xploits/xploit1
# whoami
root
```

4 Submission

1. Open up `proj1/ID.csv` and fill in comma-separated line with your student number and Name (for example 2025233000, SanSi Zhang).
2. Navigate to `proj1/` directory
3. Run `make submission`. This will create a gzipped tarball (`.tar.gz`) that contains the contents of the xploits/ directory and ID.csv. Make sure that if you extract your submission tarball:
   - In the **extracted** `xploits/` directory, running `make` with no arguments should yield `xploit1` and `xploit2` executables in the same directory.
   - The tarball must include the file `ID.csv` filled out.
4. In order to move your tarball from your vm to your local computer, open up a local terminal (not one on the vm!), and run

   ```
   scp -P 5555 user@localhost:~/proj1/submission.tar.gz
   ```

   and login if asked. `/proj1/submission.tar.gz` should be the path of where your tarball lives on your vm.

Note: *Due to the size of the class, the correctness of your submission will be graded primarily by script. As a result, following the the submission format is important. We really, really want to give you full credit! Help us help you!*

5 Troubleshooting

Windows Troubleshooting

- **Adding qemu to path:** You might need to add qemu to your path.
- **Run Project 1 through WSL:** Try running QEMU using the Windows Subsystem for Linux (WSL). Follow the instructions here to install WSL. Then install QEMU on your WSL instance using `sudo apt-get install qemu-system`. The package name is qemu-system, not qemu. The qemu package doesn't have all the required files.
- Try running qemu on myth (instructions below).

VSCode

You can use VSCode to SSH into the VM and edit your code; however, students have historically had trouble using VSCode over SSH with the VM. There's not a lot of support course staff can give since VSCode isn't required for this assignment and the issues stem from VSCode rather than our starter code. If you choose to work on this assignment in VSCode and run into VSCodespecific issues (such as "Could not establish connection to [localhost]"), here are some troubleshooting tips:

- **Restart VSCode server:** Open the command prompt window (ctrl/command + shift + P). Run `Remote-SSH: Kill VS Code Server on Host...`. Then try reconnecting.
- **Redownload VSCode server:** Inside of the VM, run `rm -r .vscode-server/..` Then try reconnecting. This will completely re-download the VSCode server on the VM.
- Make sure the VM is open and logged into in a different terminal window.
- Verify that you can ssh into the VM in a different terminal window (not with VSCode).
- Increase partition size: Make sure you've increased the space for the partition for the VM (See below)
- **Try different ssh version:** Change the version of your remote ssh extension - either make sure it's completely updated or you could also try changing your ssh extension to an older version.
- Close the VM with quit or ctrl+a X and reopen it. Try connecting again.

Ultimately, these problems are often problems with VSCode and not with our starter code, so there's not one overarching fix we can provide beyond troubleshooting suggestions. If VSCode becomes too troublesome, it might be easier to complete the assignment without it.

Increasing Space in Partition

If your VM is running out of space, here are some steps you can follow to increase the partition size.

1. Backup any files you've made/edited just in case this process fails and corrupts them.
2. Run `sudo fdisk /dev/sda`
3. Type `n` to create a new partition, then select the default partition type `(p)`. Enter the last sector of the disk to use for the new partition `(3)`, then select the default options for the first sector and partition size (hit enter twice). Type `a` to toggle the bootable flag, then select the new partition number `3`. Type `w` to save and exit.
4. Now, you should be able to see a new partition if you run `lsblk`:

```
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
fd0      2:0    1    4K  0 disk
sda      8:0    0   14G  0 disk
  sda1   8:1    0    3G  0 part /
  sda2   8:2    0    1K  0 part
  sda3   8:3    0   10G  0 part
  sda5   8:5    0  975M  0 part [SWAP]
sr0     11:0    1 1024M  0 rom
```

5. Now copy everything on /dev/sda1 to dev/sda3: `sudo dd if=/dev/sda1 of=/dev/sda3 bs=4M`
6. Next, open the `/etc/fstab` file in a text editor: `sudo nano /etc/fstab`
7. Add this line to the bottom: `/dev/sda3 / ext4 defaults 0 1`
8. Now reboot, and you should see the home dir mounted on `/dev/sda3` under the `Welcome to GRUB!` line. To reboot, use `sudo reboot`.
9. Resize to the new partition using `sudo resize2fs /dev/sda3`. If you get the error `Please run 'e2fsck -f /dev/sda3' first`, add in the `-f` flag and run again.