In the name of God

Sharif University of Technology
Department of Computer Engineering

CE 874: Secure Software Systems

Mehdi Kharrazi *Aban 19th, 1399*

Homework 0[*]

1 Get Familiar with Git

You should submit all your HWs at S4lab's Git distributed revision control system named Tarasht. To access Tarasht, your account and personal repositories will be emailed.

1.1 Git Config

In order to use Git, we recommend using a Linux machine or using a Linux virtual machine. Git uses a username to associate commits with an identity, and your username is like **ce874-991-student-id**. Using following commands, you can specify Git configuration settings with the git config command. One of the first things you should do is to set up your name, username and email address:

```
git config –global user.name ”Your name”
git config –global user.username ”ce874-991-student-id”
git config –global user.email ”you@sharif.edu”
```

1.2 SSH Key

You can generate a new SSH key (or use an existing SSH key) for authentication, then add it to your Tarasht account to modify your repositories without entering passwords every time. Using following commands, you can generate a new pair of ssh key:

```
ssh-keygen -t rsa -b 4096 -C ”you@sharif.edu”
eval ”$(ssh-agent -s)”
ssh-add ~/.ssh/id_rsa
```

After key generation, log in at Tarasht, open your account's setting and add your public key. Your public key should start with **ssh-rsa**.

1.3 HW Submission

You can find your personal repo at Tarasht, and its name is exactly like your username; therefore your personal repo URL address is `https://tarasht.ce.sharif.ir/ce874-991-students/ce874-991-student-id`.

First clone your repository with the following command:

[*]Acknowledgement: This homework was developed by Iman Hosseini and Solmaz Salimi

```
git clone git@tarasht.ce.sharif.ir/ce874-991-students/ce874-991-student-id.git
```

For each HW, you should create a new folder with the name of HW id, i.e., **HW0**. We need a pdf file as HW's report, codes and related data to be added to this folder.

After adding your HW's directory, use the following command to submit your data:

```
git add HW0
gitt commit -m "Finished HW0"
git push
```

1.4 HW Handouts

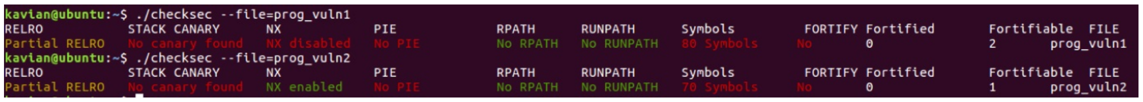
All handouts can be found at handouts repository `https://tarasht.ce.sharif.ir/ce874-991-students/ce874-991-handouts`. You should first clone this repository and before starting each HW, pull the new changes.

```
git pull git@tarasht.ce.sharif.ir/ce874-991-students/ce874-991-handouts.git
```

2 Simple BoF Attack

Your first assignment is easy Buffer Overflow and ROP attacks. We have a vulnerable binary program, which for the sake of simplicity, we have also released its C source code. You should first find the vulnerability and then write a simple script (bash, Python,Perl, etc.) to smash the stack and make it spawn a shell. The Aleph One tutorial, **Smashing The Stack For Fun And Profit**, is your friend.

The vulnerable program and its source code is available at handouts repository. Each program's security feature has shown in figure 1.



```
kavian@ubuntu:~$ ./checksec --file=prog_vuln1
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIFY Fortified       Fortifiable  FILE
Partial RELRO   No canary found   NX disabled   No PIE          No RPATH   No RUNPATH   80 Symbols      No      0               2       prog_vuln1
kavian@ubuntu:~$ ./checksec --file=prog_vuln2
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIFY Fortified       Fortifiable  FILE
Partial RELRO   No canary found   NX enabled    No PIE          No RPATH   No RUNPATH   70 Symbols      No      0               1       prog_vuln2
```

Figure 1: Security features of the programs

- **prog_vuln1:** this program is a 32bit executable that has no security feature, so with crafted input you can reach the shell.
- **prog_vuln2:** this program is a 64bit executable that has NX (Non-eXecutable) security feature on, so instead of putting the shell code into stack you should use ROP attack and find proper gadgets. Pay attention to the difference between 32 and 64 bit C calling conventions.

2.1 Delivery

You should submit a report, explaining your script, and the steps to find the return address. You should also submit your script and explain how to run it. We should note that you are not allowed to use libraries to find return address but you can use tools like pwntools and ROPgadget for automatically find gadgets and build ROP chain.

3 BAT: Binary Analysis Tool

3.1 Instructions

In this section of homework, you will implement a tool for binary analysis: *BAT*. This task is split across your first three assignments (this is the first of which) so be advised that you will be building upon the code you develop in each stage for the next parts thus having a neat implementation will benefit you later on.

The first task a binary analysis tool must achieve, is to be able to parse the header of an input binary file. An executable file (a '.exe' in windows) or an elf binary in linux is not just made up of machine instructions, there are various sections and metadata embedded into the file, you can learn more about the usage of these data and the layout of each file at the references. The specification for elf format can be found at reference 1 and for PE at reference 2. In this phase you will implement an elf parser.
There are tools which parse headers such as ELF parser (or just use readelf). You can use these to test your parser, but you cannot use additional libraries in your own implementation. Your program should take a file as input, and in the output determine the data in the header as detailed in the given samples. You can find information for every field using this reference Link. The output format does not need to be exactly as the sample outputs, the samples are used to show you the fields that you need to parse in the header. Your header parser will be tested on binaries similar to the examples.

3.2 Delivery

You should implement this only for elf files (32bit and 64bit) and submit a report, explaining your program. You should also submit your code and explain how to run it.