Introduction to Computer Security

Project 3: Capture The Flag

Chi-Yu Li (2019 Spring)
Computer Science Department
National Chiao Tung University

Goals

- Understand the exploitation of basic programming bugs, Linux system knowledge, and reverse-engineering

- You will learn about
  - Solving basic CTF problems
  - Investigating C/Linux functions deeply instead of simply using them
  - Learning what the buggy codes are and how they can be exploited

What is CTF?

From Wikipedia

- A traditional outdoor game
  - Two teams each have a flag
  - Objective: to capture the other team’s flag
- In computer security, it is a type of cryptosport: a computer security competition
  - Giving participants experience in securing a machine
  - Required skills: reverse-engineering, network sniffing, protocol analysis, system administration, programming, etc.
  - How?
    - A set of challenges is given to competitors
    - Each challenge is designed to give a “Flag” when it is countered

CTF Example

- A toy CTF

```
$ python -c 'v = input(); print("flag:foobar") if v == "1" else print("failed")'
```

- You should enter “1” to pass the *if* statement and get the flag (flag:foobar)
- Otherwise, “failed” is obtained

Requirements

- Being able to connect to our provided servers via SSH
- Linux/Unix environment is required for Task II-2 (flagbin)
- You are NOT allowed to team up: one student one team
  - Discussions are allowed between teams, but any collaboration is prohibited
- TA: Louie Lu (nctu+ics2019spring@louie.lu)

How to Proceed?

- SSH into each CTF server, except for Task II-2
- On the CTF server
  - A normal Linux environment: you can use whatever you are allowed to use
  - Two text editors: vim and nano
  - pytho3 and bash shell are available
  - You are allowed to use /tmp to play around toy codes
    - "ls" is not allowed, but you can create and run your own programs there

How to Proceed? (Cont.)

- For each CTF problem
  - There are three files: a flag program, an executable file, a source code file
    - e.g., Task I-1: flag, fildes, fildes.c
  - The executable file was compiled from the source code
  - Your mission is to get the content inside the flag file

What If Get Stuck?

- Learn to use “man” in UNIX system
  - If you don’t know something in UNIX, ask “man”
  - e.g., what is man?
    - $ man man

- Learn to find answers with FIRST-HAND INFORMATION/REFERENCE
  - Using ENGLISH KEYWORDS!!
  - First-hand information: Wikipedia / cppreference.com / devel mailing-list ...etc
  - First-hand reference: Paper / Standard / Spec / man / Source Code ...etc
  - Second-hand information: A Blog / Medium / ptt / reddit / stackoverflow post ..etc

Two Tasks

- Task I: Basic CTF programs (60%)
- Task II: CTF beginners (35%)
- Demo Q&A (15%)

Task I: Basic CTF Problems

- Task I-1: Fildes (20%)
- Task I-2: You-should-read-manual (20%)
- Task I-3: Nasty-rules

Task I-1: Fildes

- Goal: learn about Linux fd & standard I/O streams
- Server: ssh fildes@nctuics.louie.lu -p 20000 (password: guest)
- Hints
  - $ man stdin
  - $ man 2 read
  - $ man 2 atoi
  - Take time to read the code

Task I-2: You-should-read-manual

- Goal: learn to read the manual carefully
- Server: ssh manual@nctuics.louie.lu -p 20001 (password: guest)
- Hints
  - Do you really know how a C function works?
    - Don't lie to yourself. If you don't know, ask "man" as usual
    - $ man 3 rand
  - Read the manual carefully
    - You an test your thought by writing toy-code at /tmp
    - Manual/Documentation/Source code are important
    - Do you really read it carefully?

Task I-3: Nasty-rules

- Goal: learn about the sense of language details
- Server: ssh nasty@nctuics.louie.lu -p 20002 (password: guest)
- Hints
    - Operator precedence

Task II: CTF Beginner

- Task II-1: Lucky-pot (15%)
- Task II-2: Flagbin (10%)
- Task II-3: Random-pass-auth (10%)

Task II-1: Lucky-pot

- Goal: learn to identify basic logic flaw in source codes
- Server: ssh lucky@nctuics.louie.lu -p 20003 (password: guest)

Task II-2: Flagbin

- Goal: learn to use tools to inspect binary file
- Download “flagbin” to your Linux system
  - http://nctuics.louie.lu:20008/flagbin
- You should not use any GUI tools

Task II-2: Flagbin (Cont.)

- Hints
  - Your target is to find the flag inside the binary executable file
    - Try to run the “flagbin”
    - $ chmod +x flagbin
    - $ ./flagbin
  - You can use the following command line tools
    - file: determine file type
    - strings: print the strings of printable characters in files
    - objdump: display information from object files
    - gdb: debugger

Task II-3: Random-pass-auth

- Goal: learn about shell resource limitation and C error handling
- Server: ssh rpa@nctuics.louie.lu -p 20004 (password: guest)
- Hints
  - How to limit the resource?
    - Why should we do this?
    - Check the code and find its design flaw
  - Signal handling in Linux
    - How do you know which signal is given out? gdb

Project Submission

- Due date: 6/12 11:55 p.m.
- Submission rules
  - Please put your flags in a text file
    - First line: your ID number
    - Next lines: “problem_name:flag”
    - For example
      - 123456789
      - fildes:FLAG_1
      - You-should-read-manual:FLAG_2
  - Submit this text file to new E3
    - Filename: ONLY your id without “.txt”

Project Submission (Cont.)

- We will grade the text file by a script
  - Any submission that fails the script will get **NO POINTS**
  - Remember that no extension in the filename
- Grade script & an example of your submission file is on GitHub
  - mlouielu/nctuics-p3-grade-script
  - https://github.com/mlouielu/nctuics-p3-grade-script
- Make sure you have tested your file by the grade script **Before Submission**

Demo

- Date: 6/13 (9:30a.m.-5p.m.) @ EC315

- You will
  - be asked to solve one basic problem (a modified version) from Task I within 5 min
  - be asked how you solve the problems

Questions?