**Spring 2020 Cryptography and Network Security**

# Homework 3

*Release Date: 2020/5/21*

*Due Date: 2020/6/17, 23:59*

## Instruction

- **Submission Guide:** Please submit all your codes and report to NTU COOL. You need to put all of them in a folder named by your student id, compress it to hw3_{student_id}.zip. For example, hw3_r04922456.zip. The report must be in **PDF** format, and named report.pdf.
- You may encounter new concepts that haven't been taught in class, and thus you're encouraged to discuss with your classmates, search online, ask TAs, etc. However, you must write your own answer and code. Violation of this policy leads to serious consequence.
- You may need to write programs in the Capture The Flag (CTF) problems. Since you can use any programming language you like, we will use a pseudo extension code**.ext** (e.g., code.py, code.c) when referring to the file name in the problem descriptions.
- This homework set are worthy of 150 points.
- You are recommended to provide a brief usage of your code in **readme.txt** (e.g., how to compile, if needed, and execute). You may lose points if TAs can't run your code.
- In each of the Capture The Flag (CTF) problems, you need to find out a flag, which is in `CNS{...}` format, to prove that you have succeeded in solving the problem.
- Besides the flag, you also need to submit the code you used and a short write-up in the report to get full points. The code should be named **code{problem_number}.ext**. For example, code3.py.
- In some CTF problems, you need to connect to a given service to get the flag. These services only allow connections from 140.112.0.0/16, 140.118.0.0/16 and 140.122.0.0/16.
- The TAs in charge of each problem are listed below.
  - TA 1: Problem 5
  - TA 2: Problem 4
  - TA 3: Problem 1, 3
  - TA 4: Problem 2

  If you have problems, please come to the right TA hour. You can also mail to us.

# Fun Hands-on Projects

## 1. Packet Analysis (12%)

In this problem, you are asked to analyze the packet traces under `pa/` and answer the following questions.

Recommend tools: Wireshark or TShark. Both of them are network protocol analyzers and use the same format. Wireshark has a powerful display filter, by which you can filter packets by protocol / protocol fields and combine with logical operators. TShark is useful for extracting specific data via Command-line interface (CLI).

1. (5%) An attacker often does port scanning to discover potential vulnerabilities. Please observe the traffic in `traffic_1.pcapng` and identify which port (a) is open but not listening, and (b) is open and listening.
2. (7%) The TAs intercepted some USB communication from a mouse and a keyboard and stored it in `traffic_2.pcap`. Please find out the flag.

## 2. IoT Security (40%)

IoTGoat is a deliberately insecure firmware. You are asked to examine common vulnerabilities in this virtual IoT device.

For static analysis, please download the firmware image here. Kali Linux is recommended for doing the static analysis as Kali comes with many built-in tools including `binwalk`, `john`, `medusa`, `nmap`.

For runtime dynamic analysis, please create a virtual machine using the IoTGoat disk image, which can be downloaded either in the vdi or vmdk format, and loaded with virtualBox or VMWare, respectively.
Note that you need to enable the PAE/NX option if using VirtualBox.

1. In this subproblem, you need to find out the Hard-coded user credentials compiled into firmware. First, download the firmware image. Second, use `binwalk` to get the *passwd* and shadow file, and then you can use `john` (or `medusa`) with the dictionary file of the mirai botnet to get the password.
   (a) What is *iotgoatuser*'s password? Write down the command you use. (5 points)
   (b) Why Hard-coded user credentials is insecure? (3 points)
2. In this subproblem, you will investigate insecure network services.
   (a) Use `nmap` to scan for open ports and screenshot your result. (Hint: scan the ports in the range of 0-6000 only) (5 points)
   (b) What is the port number of the UPNP service? What is the version of OpenWRT, UPnP an MiniUPnPd? (3 points)

(c) Is UPNP an insecure network services? Why or why not? (4 points)

(d) What is the port number of backdoor service? Why is it vulnerable and how do you exploit it? (3 points)

3. In this subproblem, you will investigate Cross-Site Scripting (XSS) vulnerabilities. Now navigate to IoTGoat's website using its IP address. After logging in as *root*, go to the *Network* menu, select *Firewall* and select the *Traffic Rules* tab. You can find XSS vulnerability on this page.

   (a) How did you login the website as root? Write down your process. (You should make connect from kali vm. Please don't use your IoTGoat console.) (5 points)

   (b) Show your student ID in an alert message by conducting an XSS attack on this page. Screenshot your result. (5 points)

4. IoTGoat has lots of vulnerabilities. Please list at least four vulnerabilities in OWASP-IoT-Top-10-2018. (7 points)

## 3. SSL Stripping (18%)

In this problem, you are asked to setup two virtual machines—an attacker VM and a victim VM, and the attacker VM will launch an SSL stripping attack against the victim VM. As we learned in class, the attacker can see all the traffic content if the victim accidentally uses the HTTP protocol to connect to an HTTPS-required website.

You can use any OS on your virtual machines. However, Kali Linux is recommended as it comes with many built-in tools including `arpspoof`, `sslstrip`, and `wireshark`.

1. (9%) First of all, you need to reroute all the traffic of your victim VM to the attacker VM. In this experiment, you are asked to perform *ARP spoofing attack* using `arpspoof`. By performing ARP spoofing, the attacker can intercept all the victim's traffic and see the content of unencrypted packets.

   Demonstrate your work by recording the intercepted packets with `wireshark` on the attacker VM when the victim machine is visiting `http://linux8.csie.org:8888`. Write down the steps and commands you use, and save the captured packets as `packet1.pcapng`. Remember to describe your environment settings, including the IP address and virtual machine network setting (e.g., NAT/Internal/etc.) of each VM you use. You might need to enable IP forwarding on the attacker VM so that the victim can still access the Internet as usual.

2. (9%) However, with only `arpspoof`, the attacker cannot get the content if the packets are encrypted, such as by HTTPS. Therefore, you are asked to perform an SSL Stripping attack using `sslstrip`.

   Demonstrate your work by recording the intercepted packets (of the victim) using `wireshark` on the attacker VM when the victim visits a login page protected by HTTPS but not HSTS (e.g., myNTU or CEIBA). Write down the steps and commands you use, and save the captured packets as `packet2.pcapng`. The packets captured should include the username (student name) and password entered by the victim. Please do not enter any real password when you are conducting the experiment.

## 4. (D)DoS (40%)

The end of this semester is approaching! The CNS TAs came up with an online service helping them grade students' homeworks. However, they have no time to check whether the service is *secure* enough. You, as a professional security researcher, are asked to help them find out potential risks in their project.

1. (21%) The source code of the server is included in `hw3/dos`. The server is written in Python 2.7.18. Can you point out **3 bugs** which may cause DoS attacks? If the server hangs more than 1 second, the server is under a DoS attack. Please explain what the bugs are, and how to fix them in your report. You should also save your malicious inputs as `input1.txt`, `input2.txt` and `input3.txt` respectively.
2. (10%) Before starting up the server, TAs want to ensure the server works fine under SYN flood attacks. The testing VM can be downloaded from here. Here are some details about the VM:
   - The IP of the VM is `192.168.0.187`.
   - You can access the VM with account `ta2020`, password `ta2020`.
   - `run.sh` in the home directory is used to start `server.py` with IP `192.168.0.187` and port `1234`, which means clients can connect to the server by `nc 192.168.0.187 1234`.
   - `record.sh` in the home directory is used to intercept packets from the network interface.

   Your goal is to launch a SYN flood attack to the server. You are allowed to use any tool to achieve this task. Please describe your work step by step including environment settings and commands you used and save intercepted packets as `flood.pcap` when the server is under attack.
3. (3%) After demostrating a successful SYN flood attack, please help TAs protect their server by SYN cookie. Write down the steps to enable SYN cookie.
4. (6%) How to ensure that SYN cookie works properly? Please compare the server status with and without SYN cookie when the server is attacked.

*Note: if the requested file size is over 1MB, please provide a shared link in your report.*

## 5. Hidden Maze (40%)

Do you like puzzles? Do you know that you don't have to use any algorithm to solve a puzzle? In fact, you don't have to fully understand how a puzzle works in order to solve it.

All you have to do is use *Symbolic Execution*!

To solve this challenge using Symbolic Execution, we highly recommend students to follow this tutorial on solving a maze puzzle using a Symbolic Execution Engine called **KLEE**.

If you follow the tutorial above, you may notice that it is super efficient to solve a maze puzzle without implementing an algorithm such as Breadth-First Search. All you have to do is modify three lines!

Now imagine the goal of the maze ('#') is a program bug, using Symbolic Execution can help a developer to trigger the bug effortlessly, and the developer can fix the bug immediately before deployment.

We design a new puzzle game called Hidden Maze. You can access the service by `nc cns-temp.csie.org 10240`. Can you solve 5 puzzles in a row within 8 minutes? We provide you with a script `hw3/maze/check_solution.c` to check if your solution for the puzzle is correct.

If you directly use a Symbolic Execution Engine to solve this puzzle, you may encounter two problems:

1. It ends immediately.
2. It may takes forever to get the result. This is one of the bottlenecks in Symbolic Execution called *Path Explosion*. One solution to mitigate the problem is to prune unnecessary paths. In this case, you have to trace the code and understand how the puzzle works, then try to modify the code so that the Symbolic Execution Engine can solve the puzzle.

Grading Policy:

1. ( 5%) Get the flag.
2. ( 5%) Explain how this puzzle works.
3. ( 5%) Explain why the Symbolic Execution Engine **KLEE** ends immediately if you only modify the three lines similar in the tutorial above. (Add KLEE's header, make a buffer symbolic and add a klee assertion)
4. ( 5%) Explain why this challenge suffers from *Path Explosion*.
5. (10%) Explain in details how you mitigate the problem in (3) and (4).
6. (10%) Reproducibility: Provide a fully automated script to solve this challenge using **KLEE**. You need to provide a `Makefile` to compile all your binaries. The TA will run your script by running `make run` on a **KLEE** docker image, so you have to make sure your script can successfully get the flag on that image. Note that the TA will only run `make run`, if you need to install any libraries, please provide a script `initialize.sh`.

*Note: You need to solve a PoW first when you connect to the server,* `hw3/maze/pow.py` *is provided to solve the PoW.*

*Hint: Modifying 6 lines (including the similar 3 lines in the tutorial) of the code* `hw3/maze/check_solution.c` *should work.*