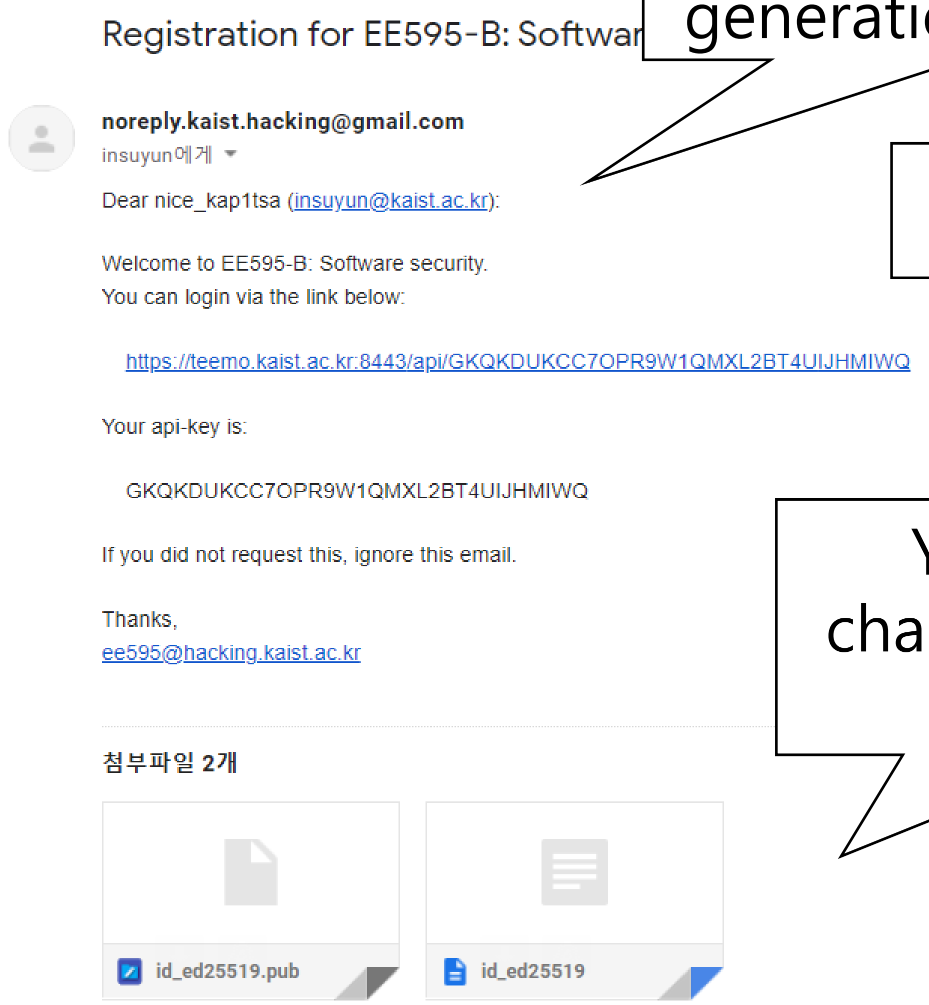
Lab00: Getting started

Insu Yun

Your account e



Registration for EE595-B: Softwar

noreply.kaist.hacking@gmail.com
insuyun에게

Dear nice_kap1tsa (insuyun@kaist.ac.kr):

Welcome to EE595-B: Software security.
You can login via the link below:

https://teemo.kaist.ac.kr:8443/api/GKQKDUKCC7OPR9W1QMXL2BT4UIJHMIWQ

Your api-key is:

GKQKDUKCC7OPR9W1QMXL2BT4UIJHMIWQ

If you did not request this, ignore this email.

Thanks,
ee595@hacking.kaist.ac.kr

첨부파일 2개

id_ed25519.pub

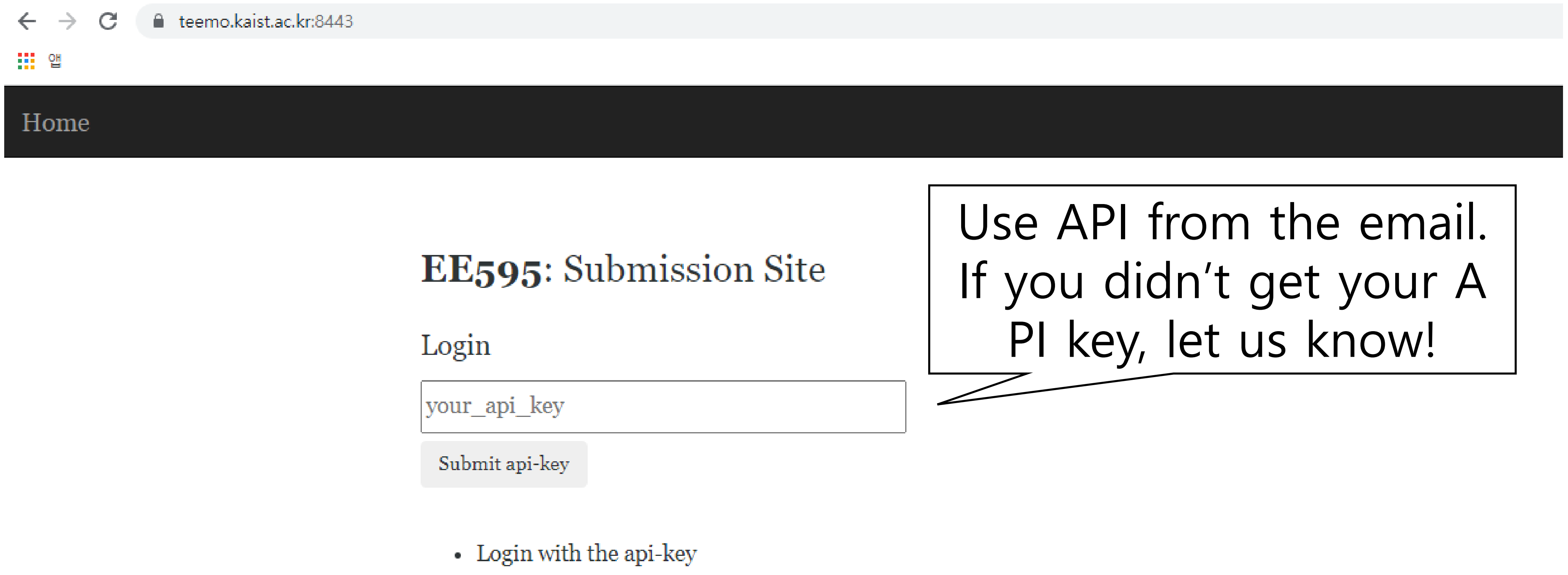
id_ed25519

Your ID: pseudonym (Using Docker's generation mechanism)

Your API Key

Your SSH key for challenge sever will be attached!

Submission : Login



teemo.kaist.ac.kr:8443

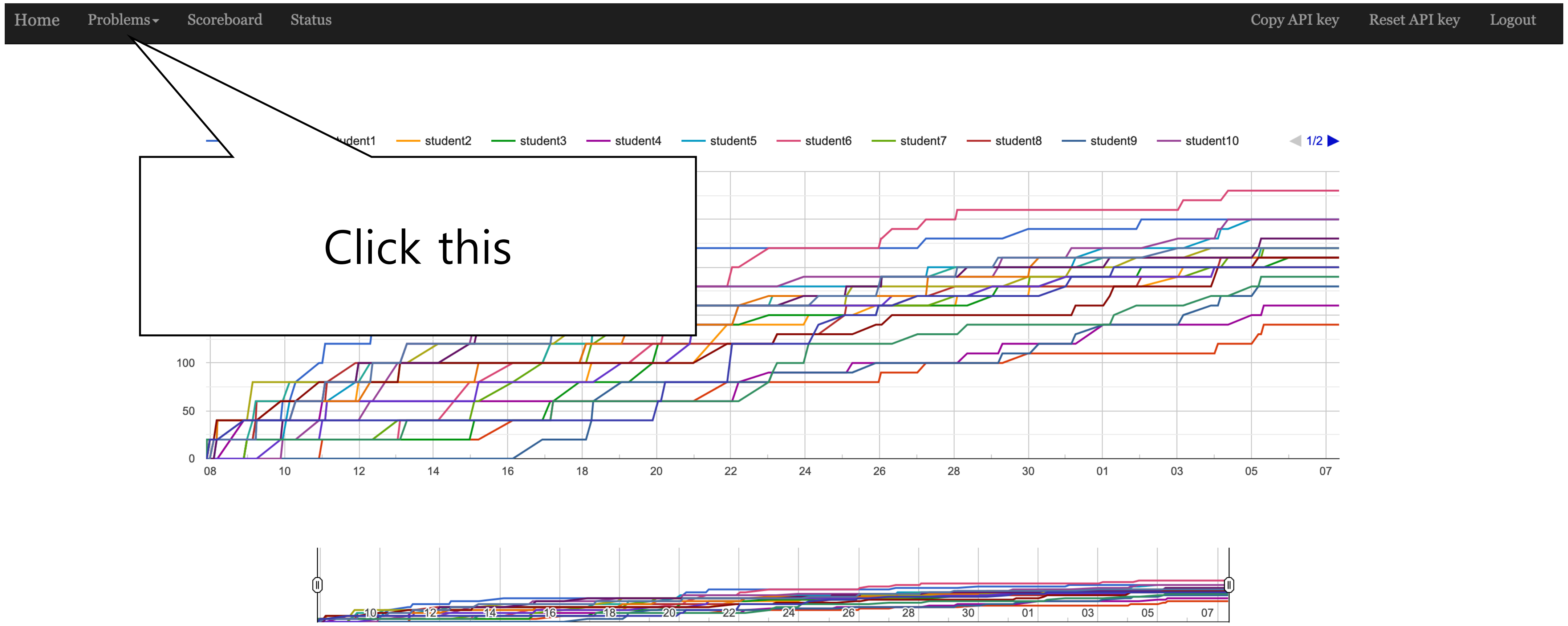
EE595: Submission Site

Login

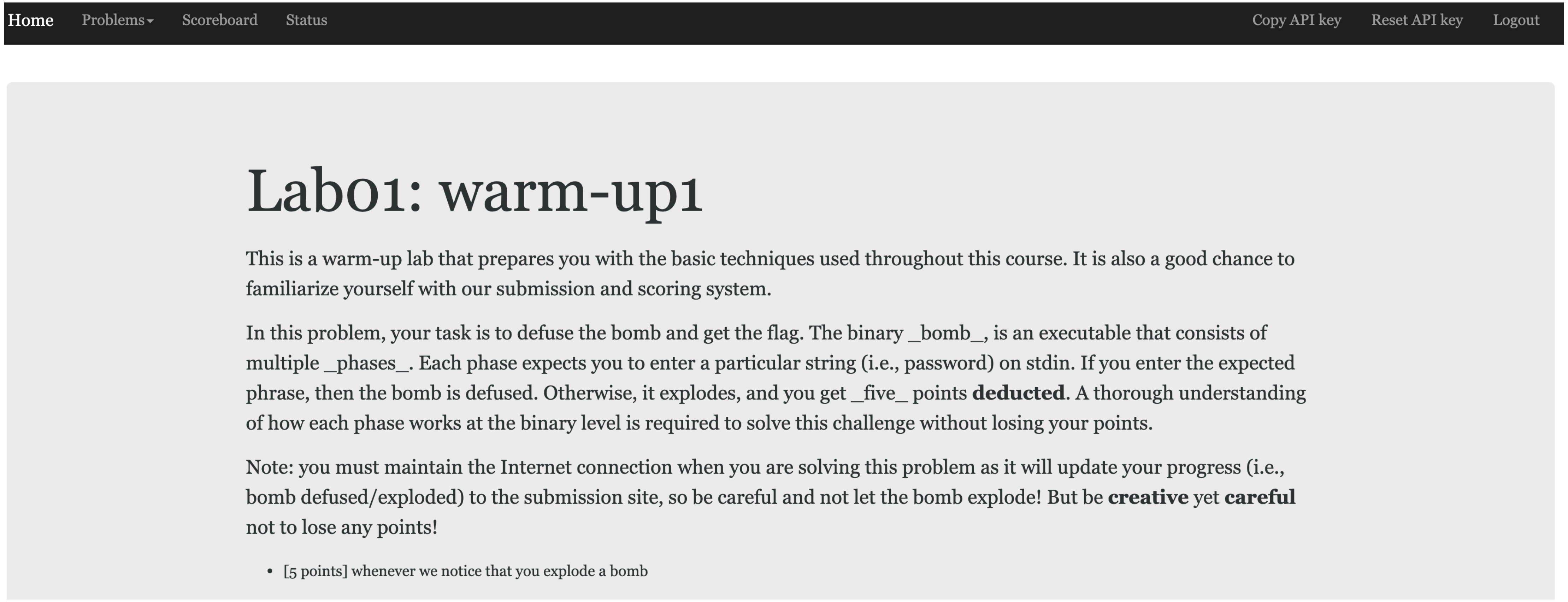
- Login with the api-key

Use API from the email.
If you didn’t get your A PI key, let us know!

Course homepage: Home



Course homepage: Lab



Lab01: warm-up1

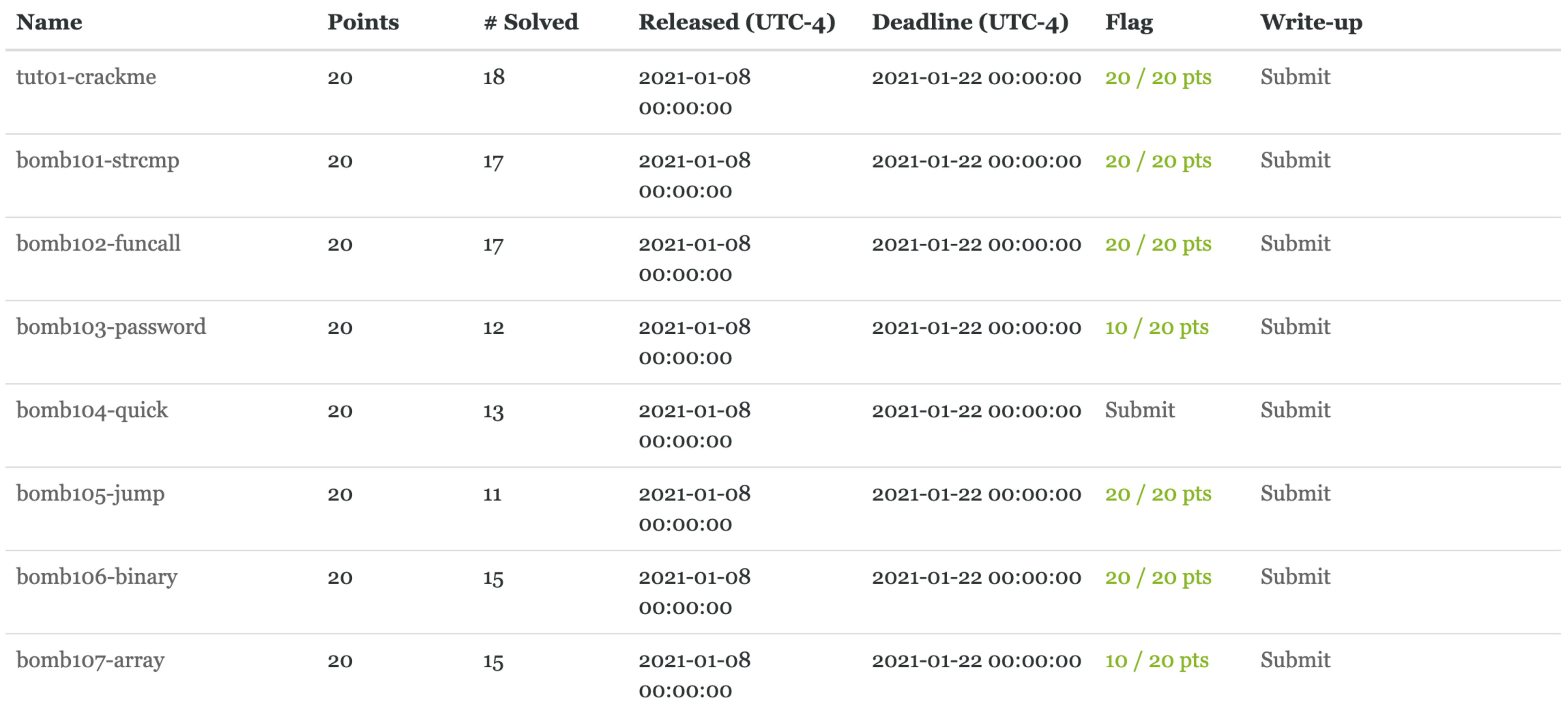
This is a warm-up lab that prepares you with the basic techniques used throughout this course. It is also a good chance to familiarize yourself with our submission and scoring system.

In this problem, your task is to defuse the bomb and get the flag. The binary _bomb_, is an executable that consists of multiple _phases_. Each phase expects you to enter a particular string (i.e., password) on stdin. If you enter the expected phrase, then the bomb is defused. Otherwise, it explodes, and you get _five_ points **deducted**. A thorough understanding of how each phase works at the binary level is required to solve this challenge without losing your points.

Note: you must maintain the Internet connection when you are solving this problem as it will update your progress (i.e., bomb defused/exploded) to the submission site, so be careful and not let the bomb explode! But be **creative** yet **careful** not to lose any points!

- [5 points] whenever we notice that you explode a bomb

Course homepage: Lab

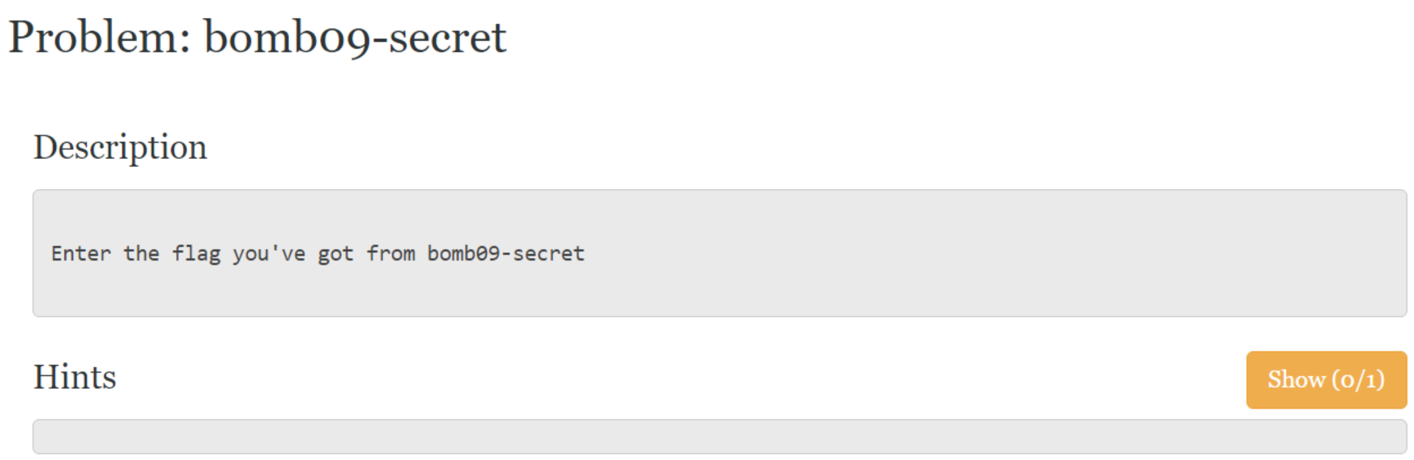


Name	Points	# Solved	Released (UTC-4)	Deadline (UTC-4)	Flag	Write-up
tut01-crackme	20	18	2021-01-08 00:00:00	2021-01-22 00:00:00	20 / 20 pts	Submit
bomb101-strcmp	20	17	2021-01-08 00:00:00	2021-01-22 00:00:00	20 / 20 pts	Submit
bomb102-funcall	20	17	2021-01-08 00:00:00	2021-01-22 00:00:00	20 / 20 pts	Submit
bomb103-password	20	12	2021-01-08 00:00:00	2021-01-22 00:00:00	10 / 20 pts	Submit
bomb104-quick	20	13	2021-01-08 00:00:00	2021-01-22 00:00:00	Submit	Submit
bomb105-jump	20	11	2021-01-08 00:00:00	2021-01-22 00:00:00	20 / 20 pts	Submit
bomb106-binary	20	15	2021-01-08 00:00:00	2021-01-22 00:00:00	20 / 20 pts	Submit
bomb107-array	20	15	2021-01-08 00:00:00	2021-01-22 00:00:00	10 / 20 pts	Submit

Note on flag

- Format: is521{...}
  - e.g., is521{thi5_i5_s4mple_fla9_f0r_y0u}
- Usually, locate at /is521/[lab_name]/[challenge_name]/flag
  - e.g., /is521/lab01/tut01-crackme/flag
- Sometimes, a binary embeds the flag itself
  - e.g., bomlab in lab01
- If you cannot find where the flag is, don't hesitate ask us

Hint system for you!


Problem: bomb09-secret

Description

```
Enter the flag you've got from bomb09-secret
```

Hints

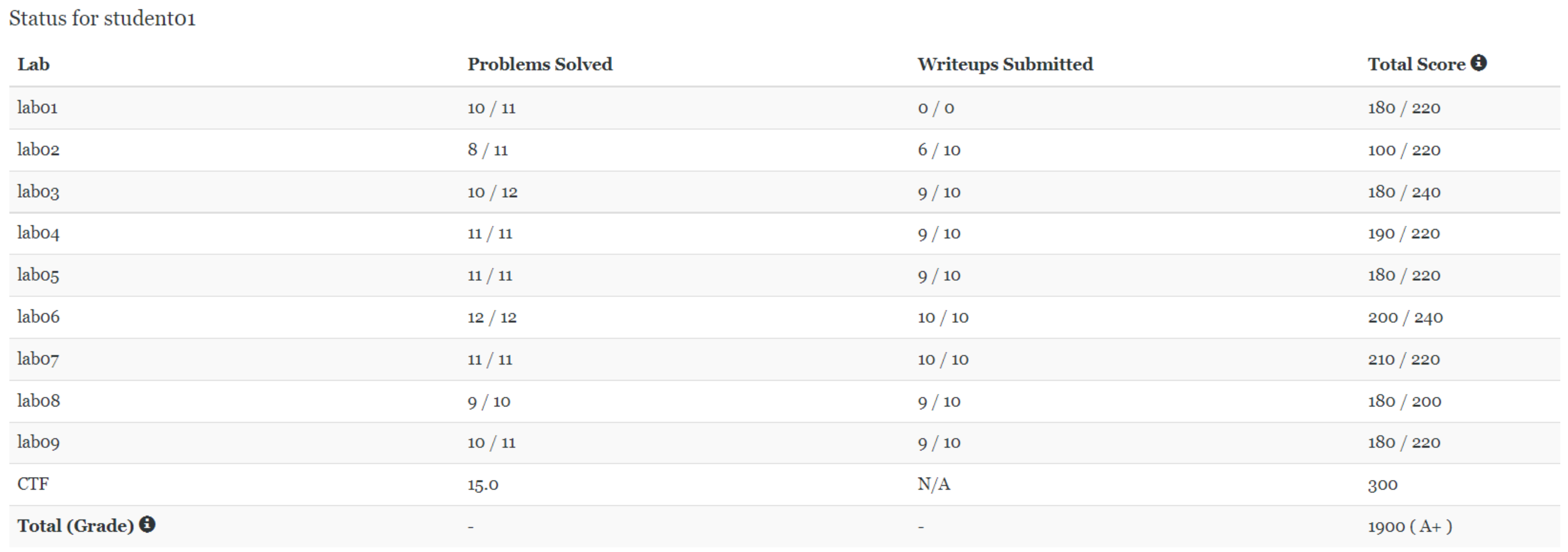
- Some challenges have hints for you.
- If you want, feel free to open it!

CTF server

- ssh YOUR_ID@teemo.kaist.ac.kr –p 9000 –i YOUR_PRIVATE_KEY
- cd /is521/lab01
- cat README

- If you are using Windows, please install WSL2 (https://docs.microsoft.com/en-us/windows/wsl/install) for using linux

Status



Status for student01

Lab	Problems Solved	Writeups Submitted	Total Score
lab01	10 / 11	0 / 0	180 / 220
lab02	8 / 11	6 / 10	100 / 220
lab03	10 / 12	9 / 10	180 / 240
lab04	11 / 11	9 / 10	190 / 220
lab05	11 / 11	9 / 10	180 / 220
lab06	12 / 12	10 / 10	200 / 240
lab07	11 / 11	10 / 10	210 / 220
lab08	9 / 10	9 / 10	180 / 200
lab09	10 / 11	9 / 10	180 / 220
CTF	15.0	N/A	300
Total (Grade)	-	-	1900 (A+)

- Total score reflects writeup status (i.e., no writeup == zero!)

Write-up

- You should submit a write-up for each challenge to get actual point!
- Be concise yet precise!
- You should use Markdown (https://www.markdownguide.org/) to write your writeup

- You don't need to submit writeups for tutorials and the first lab

Write-up sample

~~~
## Description

In this challenge, ebp and the return address are protected by stackshield.
By doing debugging, you can see all ebp and ret values are keep tracking and
storing somewhere. However, when you make an input large enough, you will see
that a function pointer will be overwritten. And the overwritten value will be
store in EAX and make it jump at <main+96>. I put my shellcode as env, get the
address, and put it. In my case, the function pointer(0x08048b0a at 0xbffff654)
was overwritten. So we could learn, we could jump using the weakpoint even
though the stackshield is working on.

## Exploit
```python
  #!/usr/bin/env python3

  import os
  import sys

  from pwn import *

  payload = cyclic(100) + p32(0xbffff654)
  p = process(["/ee595/lab02/func_ptr/target"])
  p.sendline(payload)
  p.interactive()
```

## Collaborator: Insu Yun
- I asked a question about how to get the core file from the server
~~~

Description

Exploit code

Collaborator