Easy: Nothing to see here, really?

1. open inspect elements
2. commented out `<a>` leads to `/admin`.
3. visit `/admin` for the flag

Easy: Linux Access Control

1. open file
2. screenshot shows 111(7) + 100(4) + 011(3) = 743.

Easy: Insecure Network Communication

1. use wireshark, look for HTTP packets first since it is the easiest
2. Find leetspeak text

Easy: Certificate's Content

1. use online decoder https://report-uri.com/home/pem_decoder

Easy: What Powers Me?

1. open inspect elements network tab
2. reload luminus
3. check headers of HTTP `GET` response

Easy: Host Reconnaissance

1. `nmap 178.128.126.127` port scan the ip
2. open ports are `22`, `80`, `8080`, `443`.
3. http port 80 `http://178.128.126.127` is some lame joke
4. https, port `443`, is some github user MechFroG88's personal project called "The Revolution". Game too long, skipped.
5. Accessing `http://178.128.126.127:8080/` gives the answer.

Medium: Mallory in Action

1. MITM attack: intercept alice packet. save prime `n`, generator `g`, alice `pub_key_a`.
2. Generate fake key `fake_key` with `n`, `g` and private key `k` such that `fake_key = g ^ k % n`.
3. send bob `n`, `g` and `fake_key`. Bob responds with `pub_key_b`.
4. send alice `fake_key`.
5. Bob/Mallory shared key = `pub_key_b ^ k % n`. replace with `pub_key_a` for alice/mallory.
6. decrypt messages

Medium: The Prequel: I Dislike Some Keyword

- Try username `admin`, comment out password check.
- Full username: `admin';--`.

Medium: Please Join the SQLi Games

Part1

1. Force `WHERE` to be true: username = `' OR 1 = 1;`. First query = `SELECT * FROM USERS WHERE username = '' or 1 = 1;` Part1 flag: `CS2107{9r33N_l19h7_R3D_L19H7_jybBZC67EtVM6YdK}`

Part2

1. Check banned words: `OR`, `INSERT`
2. Try guessing usernames: `root`, `admin`
3. `admin` passed with `admin'--` Part2 flag: `CS2107{15_9lUc053_t45tY_qRkkru6hkzBVCZdM}`

Part3

1. Try `admin'--` again Part3 flag: `CS2107{4R3_w3_47_7h3_3nd_8uN57k5se2Kkv8hL}`

Medium: File Inclusion

1. Question mentions URL parameters, append `?f=secret.php`. troll page obtained.
2. Typing gibberish reveals parameter `f` is used to `include()` a file.
3. Use php filter instead `?f=php://filter/convert.base64-encode/resource=secret.php` obtains base 64 data

4. decode data for flag using online decoders

Medium: Ret2win

1. Send 32B (buffer size) + 8B (rbp) garbage + addr of win()
2. obtain `&win` using IDA decompiler

Medium: Format String Theory

1. Input string is directly used in printf, so `%p` would be parsed in printf
2. `perm` needs to be modified to >0 to pass the if check
3. Using IDA, find `perm` is 32 bytes away
4. using `%*s` we can add an extra parameter to get around the character limit.
5. Full message: `%p%*p%p%n`.

Medium: The end

1. `NAME_MAX_LENGTH` is `16` but the buffer read by `read_str()` is `20`.
2. We have a `system(command);` evaluating the code in `app.stats.logger`.
3. `app.stats.logger` is after the `app.selected.*`, and `strcpy` is used, so we must overflow to it.
4. Goal is to prevent any `\x00` from being written so we can enter a string of arbitary length text to `strcpy` and overwrite `command`.
5. Using IDA, we can see that `dest` is at `0x68`, while logger is at `0x88`.
6. enter a name of at least 16 length
7. Enter a value of height, weight etc. that does not contain and `\x00` bytes, e.g. INT_MAX
8. use `select_person` to `strcpy` the string to overwrite `app.stats.logger`.
9. enter the command to run as the name of the next consecutive person.
10. `ls` used to browse the files. `cat flag.txt` used to extract the answer.

Hard: The sequel

1. login with bob
2. enter `'--`, no useful information obtained, so info must be in another table, union attack.
3. enter `order by 1;--` to n to determine number of columns. 4 columns total.
4. use `' UNION SELECT 'a', NULL, NULL, NULL FROM USERS` to view column types, corresponding locations.
5. find number of columns in user table: `' AND EXISTS(SELECT * FROM USERS ORDER BY 3)--` , 3 columns.
6. Append 1 extra column to attack: `' UNION SELECT 'a', * FROM USERS; --` , flag obtained

Hard: Improper code

1. buffer overflow possible with `gets`, `\0` is accepted. overflow until segfault
2. `system()` requires argument in register `rdi`.
3. search for gadget `pop rdi; ret` using `dumprop` and `gdb-peda`.
4. ASLR enabled, find `system` PLT: `readelf -s /usr/lib/gcc/x86_64-linux-gnu/9/../../../x86_64-linux-gnu/libc.so.6 | grep system` .
5. overflow: 64B (text), 70B (b), 10B (a), 8B (rbp), 8B gadget addr, 8B `&IlikeTOnameVARIABLESlikeTHIS`, 8B system PLT addr
6. segfault at `movaps`. add `ret` gadget to byte align stack.

Hard: Genie

1. `guess` is a signed int, but is passed to `askGenie` as unsigned int. Passing `-1000` passes `guess <= 48` but gives us more than enough overflow in `askGenie`.
2. `&release` is 48 bytes away, enter 48 garbage non zero inputs, then enter `&execute_genie`.
3. Conditional after `ask_genie` terminates if `magicPouch->release >= &execute_genie`. `x/10i &ask_genie - 0x8` in gdb to check if it is possible to jump before.
4. found `no-op`, jump to 1 byte before `&execute_genie` instead.
5. Guesses: 48 x `1`, `0xa8`, `0x13`, `0x40`, 5 x `256` (`0x00`)