Network Administration and System Administration

Final Examination

Time: 2025/06/02 09:10 - 12:10

Instructions and Announcements

- 考試時間共三小時，三人一組考試。分組連結及簽到結果。
- 考試期間禁止使用手機、電話、任何通訊軟體等與同組成員外任何人聯繫，也禁止組與組之間**一切討論與合作**，如被發現視為作弊行為，**期末考 0 分**，並依校規懲處。
- 作答過程中請自行斟酌備份，避免電腦發生意外，損失過多進度，可考慮準備隨身碟或雲端空間備份進度。
- 為避免發生重大意外，請自行注意 VM 用量，同時開啟過多 VM 可能導致電腦當機，我們恕不負責。
- 題目檔案請至 雲端硬碟下載 (考試開始後公開) 。也可以在考試開始前先下載加密過的壓縮檔 (nasaws*:/tmp2/NASA_Final/2025-final-*.zip)，解壓縮密碼將會在考試開始後公佈。
- 公告在這：公告
- 完成題目時請至 Submission Form 上傳作答內容，每組每個 subtask **最多上傳 3 次**，請注意第八題的 reset 每組僅有 1 次額度。
- 計分板連結 Scoreboard。
- 各題後面黑色星號數目代表我們估計的難度。請參考，可用來決定解題順序。
- 題目可能難免有疏誤之處。若發現有解不開題目、題敘不清的狀況，請盡快跟助教或老師反應，或斟酌時間先解別的題目。
- 本次考試中，若有需要進行 vm to vm connection 的需求，推薦使用本機進行處理，若有必須在工作站上進行 vm to vm connection 的情況，請依照 hw9 所公布的 vde 與 ip 分配表進行使用：分配表，你所需要的 `vde*.ctl` 皆位於 `/tmp` 目錄底下。
- 滿分為 120 pts。

1 NFS ★ ∼ ★★★ (15 points)

Description

這題要求同學操作 NFS Server 與 Client，完成各種任務。

Resources

1. nasa-nfs-server.qcow2
   - username: `arona` (no sudo permission)
   - password: `nasa2025`
   - root password: `nasa2025`
2. nasa-nfs-client.qcow2
   - username: `plana` (no sudo permission)
   - password: `nasa2025`
   - root password: `nasa2025`
3. nasa-observer.qcow2
   - username: `spy`
   - password: `nasa2025`
   - root password: `nasa2025`

Tasks

0. **(0 points) 基礎設定**
   請在 192.168.[GroupID].0/24 的子網段下進行設定
   - nfs-server
     請自行安裝並啟用 nfs-kernel-server
     - IP：192.168.[GroupID].1
     - NFS 位置：/srv/momotalk
     - NFS 只對 192.168.[GroupID].2 開放
     - /etc/exports 中要開啟 `no_root_squash` 選項
   - nfs-client
     請自行安裝並啟用 nfs-client
     - IP：192.168.[GroupID].2
     - NFS 掛載位置：/mnt/momotalk
   - observer
     請自行安裝並啟用 nfs-client

1. **(3 points) ACL 設定 ★**
   (a) 在伺服器的 /srv/momotalk 資料夾下，為 arona 和 plana 兩個使用者分別建立一個專屬的子資料夾，並確保各自的子資料夾只有對應用戶可以訪問，其他用戶無法讀取或寫入。

(b) 在伺服器的 /srv/momotalk 資料夾下創立一個 chat.txt，允許兩個用戶讀寫。

(c) 請向助教 demo 以下 6 個指令：

i. `arona@nasa-nfs-server:momotalk$ ls arona`
ii. `arona@nasa-nfs-server:momotalk$ ls plana`
iii. `arona@nasa-nfs-server:momotalk$ echo "I'm arona." >> chat.txt`
iv. `plana@nasa-nfs-client:momotalk$ ls arona`
v. `plana@nasa-nfs-client:momotalk$ ls plana`
vi. `plana@nasa-nfs-client:momotalk$ echo "I'm plana." >> chat.txt`

2. **(2 points) IP Hijacking & NFS ★**

偷偷告訴你，arona 和 plana 目前使用的 NFS 系統存在著安全隱患，這種漏洞會給不懷好意的人（a.k.a. 地下生活者）有機可趁。他只要偽裝成合法用戶的 IP，就能繞過檢查、成功掛載並讀寫資料。

現在，請將 `nasa-client` 關機，使用 `nasa-observer` 並以 `root` 身份進行攻擊，完成以下任務：

(a) 將 `nasa-observer` 的 IP 手動設定為 `192.168.[GroupID].2`，偽裝成合法 NFS client。

(b) 嘗試使用 root 掛載 `192.168.[GroupID].1:/srv/momotalk` 至本機任一資料夾(如 `/mnt/fake`)。

(c) 若成功掛載，請寫入以下內容至 `chat.txt`：

```
spy: intercepted=[GroupID]_spoofed_access
```

(d) 向助教 demo 掛載過程與檔案內容。

3. **(7 points) 加密通信 ★★★**

在上一題中，地下生活者成功偽裝成 plana 入侵系統，繞過 NFS 權限控制。但這一次，arona 和 plana 決定升級系統安全，導入 Kerberos 身份驗證機制，防止未經授權的存取行為。

請幫助他們完成以下任務：

(a) 設定 NFS server 為 Kerberos 認證模式，掛載參數使用 `sec=krb5` 或 `sec=krb5p`，Realm 名稱請設定為 `TEAM[TeamID].EXAM`。

(b) 確保 arona 與 plana 能夠正常透過 Kerberos 掛載 NFS，並具備正確的資料夾存取權限。

(c) 嘗試再次讓 nasa-observer 偽裝成 `192.168.[GroupID].2` 掛載 NFS，觀察是否仍然能成功。

(d) **Demo：**

- 在 `nasa-nfs-server` 與 `nasa-nfs-client` 上分別使用 `klist` 查看 ticket 狀態。
- 實際示範 Kerberos 掛載成功（client）與 spoof 掛載失敗（observer）。

Note: 在啟用混雜模式的網路環境中，原始的 NFS 傳輸內容為明文，容易被封包擷取工具（如 Wireshark）截取。而 Kerberos 認證（特別是 `krb5p` 模式）會對資料進行加密與驗證，可有效防止封包竊聽與冒用。

Submission

- 請找 TA 並進行 demo。

2 Wireless ★ ∼ ★★★(20 points)

Description

請於雲端硬碟取得所需檔案。

- 2-1 與 2-2 的所需檔案在 `WEP/`。
- 2-3 的所需檔案在 `freeradius/`。

Tasks

1. (5 points) **The IV You Know ★★**
   我們在 lab 中有講到網路的協定，網路協定的演變正是因為被破解了才會持續進步。但是我們可不能去破解別人的網路，要剛好找到有漏洞的請況也不容易。在某個異世界的地方，他們還使用著類似 WEP 的協定。請你破解他，讓他們進步吧。
   lab.pcap 是一個模擬 WEP 的檔案，是用 WEP_gen.py 生成的。由於世界不同，他們的 WEP 跟我們的不完全一樣，必須從 WEP_gen.py 去看封包是如何產生的。以下是 lab.pcap 的**已知資訊**：
   - 檔案裡面混入了一些無關的封包，你的目標是破解 source MAC 為 `de:ad:be:ef:00:01` 的 traffic。
   - 無關的封包以及真實的封包各自有 100 組，但是由正確 keystream 所加密的訊息一定是長度為 47 bytes 的 plain.txt。

   雖然沒辦法直接用 aircrack-ng 去破解，但這個偽 WEP 跟原協定一樣本身就不安全，我們還是有辦法可以得到他的密碼的。在開始破解密碼前，我們要先收集些資訊，也就是我們要取得的是他的 IV 與其相對應的 keystream。

   請在蒐集完畢所有的 IV 後，回答當 IV=$8e44b2$，其所對應到的 keystream 是什麼？

   注意事項：
   - **請務必以 WEP_gen.py 當中的封包產生方式為解題主要參考，而非原本 WEP 的協定。**
   - 請於 Google Forms 以 hex 格式繳交答案，格式應為 `[0-9a-f]{94}`。
   - 在遵守考試規則的前提下，使用任何方法得到答案都是可以的。

   Hints：
   - 承題敘，這個協定與 WEP 十分相似，僅有如封包格式等細節上有所不同。故請你試著回想作業破解 WEP 的過程，IV/ciphertext/plaintext/keystream 之間有什麼關係？
   - 撰寫 Python script 會是一個不錯的解題方向，你可以參考 Scapy library。

2. (7 points) **Crack the Stream ★★★**
   在收集所需資訊後，我們就可以去破解密碼了。請你使用上一題蒐集的 IV 與 keystream，破解出這個偽 WEP 網路使用的密碼。

   注意事項：
   - 請於 Google Forms 繳交 (1) 此網路的密碼（WEP_gen.py 當中的 `KEY`）(2) 你得到答案所使用的方法（若有使用 scripts，請使用 Google drive 連結提供）
   - 在遵守考試規則的前提下，使用任何方法得到答案都是可以的。

Hints：

- 密碼格式為 `nasa2025[0-9a-z]{5}`
- 破解過程跑 30 分鐘是有可能的（By 驗題者：官解花了 10 分鐘，務必先用簡單的測資確認有正常運作再開始破解）

3. (4 points) **Wireless Group's Daily Job ★**
RADIUS 是無線網路系統中常見的 AAA 協議，而我們系上的 RADIUS 伺服器是使用 FreeRADIUS 這個軟體。在 `freeradius` 這個資料夾裡面包含某一台 FreeRADIUS 伺服器的完整設定檔。使用 RADIUS 進行 AAA 當中的驗證（Authentication）時，一種方法是到儲存使用者帳號密碼的資料庫確認正確性。請從提供的設定檔中找到答案並回答下列問題：

   (a) 請問使用的資料庫是什麼？請回答資料庫的名稱。

   (b) 用來登入這個資料庫軟體的帳號密碼分別是什麼？

   注意事項：

   - 設定檔所在伺服器的 FreeRADIUS 版本是 3.2.7，作業系統為 MacOs Ventura 13.7.6。
   - 設定檔在原伺服器的路徑為 `/usr/local/etc/raddb`。

4. (4 points) **CSIE Wireless Architecture ★**
我們上課有講過系上無線網路的 Authorization 架構圖。考慮當你使用你的手機（Client）連到系上的 csie-5G，到驗證成功能開始使用網路，這中間的驗證過程除了 Event log server 、DHCP server 與 Client，一共與四個系上的服務有關。請你回答以下內容：

   - 這四個系上服務的名稱，與其各自的功能。
   - 服務之間的溝通關係（你可以使用諸如 `Service 1 <-> Service 2` 的形式進行回答）

Submission

提交至 Google Forms。

---

3 Workstation ★ ∼ ★★★(20 points)

Description

請於雲端硬碟取得所需檔案。

- 包含 `Arch.qcow2` 以及 `Debian.qcow2`

Environment

- **LDAP Server**
  - OS: Debian GNU/Linux 12
  - 使用者名稱: `root`
  - LDAP admin dn: `cn=admin,dc=nasa,dc=csie,dc=ntu`

- **Workstation**
  - OS: Arch Linux
  - 使用者名稱: `root`

* 此題所有使用者密碼均為 `nasa2025`
* **請自行檢查網路設定。如果網路設定正常，則兩台虛擬機可以互相以 ssh 登入對方。**

Tasks

P.S 本題各個題目是獨立的，請按照任意順序作答。

1. (8 points) **Who Blocks Us? ★★★**

   你的 Workstation 無法使用 ldapsearch 連線到 LDAP Server，請嘗試尋找原因並修復（真人真事改編）。

   `hint`: LDAP Server 的機器是否有收到 Workstation 傳送過來的 LDAP 封包呢?
   `hint`: 觀察 LDAP Server 的 log，Server 本身是否有收到 LDAP 的請求呢?

2. (4 points) **The Mysterios Failure ★★**

   有一位使用者回報，在 Workstation 上安裝套件前，如果執行 `pacman -Syu` 會發生錯誤。請找出發生錯誤的原因並修復它，並展示成功執行 `pacman -S python-matplotlib` 的畫面。

3. (4 points) **Add User ★**

   為了簡化 LDAP 帳號建立的流程，我們希望撰寫一個 shell script，能夠自動根據輸入的資訊在 LDAP Server 上新增一個使用者帳號。請撰寫一個腳本，完成下列需求：

   - 執行腳本後，會有四行輸入，依序為:
     1. 中文姓氏 (例如: 王)
     2. 中文名字 (例如: 小明)
     3. 使用者帳號 (`uid`) 必須符合 `[0-9a-z]{1,10}`。這是代表使用者帳號只能夠使用數字和小寫英文字母，而長度為 1 到 10 個字元。
     4. 使用者密碼必須符合 `\w{1,10}`，這是代表密碼可以包含小寫和大寫英文字母、數字、底線，但不允許其他符號，且長度為 1 到 10 個字元。
   - 使用者會被加入到 `ou=people,dc=nasa,dc=csie,dc=ntu`
   - 請為使用者添加 `objectClass: inetOrgPerson`，使用者的 `sn` 與 `givenName` 必須分別為 base64 編碼後的姓氏與名字。
   - 依據使用者名稱來設定其家目錄的位置(也許你還會需要為使用者添加其他的 objectClass)。
   - 保證不會重複新增相同的使用者名稱。

4. (4 points) **User Lookup ★**

   使用者帳號建立後，我們也希望能快速查詢該帳號對應的「中文姓名」，因此請撰寫一個查詢用的腳本，讓 Workstation 上的使用者能夠根據在 LDAP Server 上的使用者帳號列出其姓名資訊。

   - 執行腳本後，應接受一行輸入（`uid`）。
   - 透過 LDAP 查詢該 uid 對應的 sn（姓）與 givenName（名），並將 base64 解碼後輸出為原本的中文姓名。
   - 保證查詢的 `uid` 在 LDAP Server 中。

Submission

請找 TA 進行 demo。

---

4 Security ★ ~ ★★★ (20 points)

Description

- 請不要用任何形式干擾其他人作答，或不是以解題為目的來攻擊本作業的各項設施。經舉發查證屬實者，將會受到非常嚴厲的懲罰。
- 請不要直接抄襲其他組別獲得的 Flag。
- 標有 (CTF) 的題目會需要你在表單繳交 Flag 作為答案。
- Flag format: `NASA2025{[a-zA-Z0-9_+-:]+}`
- 只要不是抄襲或作弊，非常歡迎你嘗試非預期解。

Environment

- **term1n4l 機器** IP/Port：`https://140.112.91.4:8780`
- **舊 FATCAT DNS**:
  - Source code: `server-a.py`
  - IP/Port: `140.112.91.4:48765` (TCP)
- **新 FATCAT DNS**:
  - Source code: `server-b.py`
  - IP/Port: `140.112.91.4:48766` (TCP)

Tasks

term1n4l ★ ~ ★★★ (10 points)

奉民主官的命令，身為絕地戰兵的你被派遣到遙遠的外星球關閉機器人陣營的非法廣播 (?

1. **(3 points) Hack the secret service ★**

   (a) 上述機器的 IP/Port 會 forward 到一台 VM 上的 nginx 服務，請你找出這個 nginx 服務中一個需要輸入帳號與密碼的隱藏頁面，並提供他相對於 / 的路徑。(1 points)
   Hint：有什麼樣的 Command line tool 可以幫你找到 nginx 有 host 什麼東西？

   (b) (CTF) 你的戰兵夥伴找到了這個 nginx 服務的部份設定檔，眼尖的你立刻就發現一個設定上的漏洞！請你利用這個設定檔的漏洞，嘗試破解上一題頁面的帳號密碼，並在登入服務後繳交你找到的 `flag1.txt`。(註：破解密碼時請使用 rockyou.txt 作為 wordlist) (2 points)

2. **(7 points) 🔪 🔪 🔪 is everyone's favorite ★★★**

   (a) 請你閱讀網頁中提供的 Flask source code，並簡單描述這個 Flask 服務的功能。(1 points)

   (b) `flag2.txt` 居然偷偷躲在外面兩層的目錄（`../../flag2.txt`）請你提供一個能把他抓出來的 Path。（註：這邊只要求你提供任何錯誤訊息不是 `Path traversal? Try harder...` 且包含 `flag2.txt` 的 Path）(2 points)
   Hint：怎麼叫 `curl` 一次幫我抓好幾個東西？這個功能怎麼幫我繞過服務的限制？

   (c) (CTF) 請你想辦法繞過 Flask 服務的限制偷出 `flag2.txt`。(4 points)
   Hint 1：這個人似乎把 `validate(key)` 寫壞了，導致 `key` 似乎不一定要是 `{...}` 的形式...
   Hint 2：看來你需要把 `NASA2025` 清掉，但 `flag2.txt` 本身沒有 `{...}` 可以做替換，所以你或許會需要用到上一題 `curl` 的功能，自己造一個 `{...}` 把 `NASA2025` 關進去？

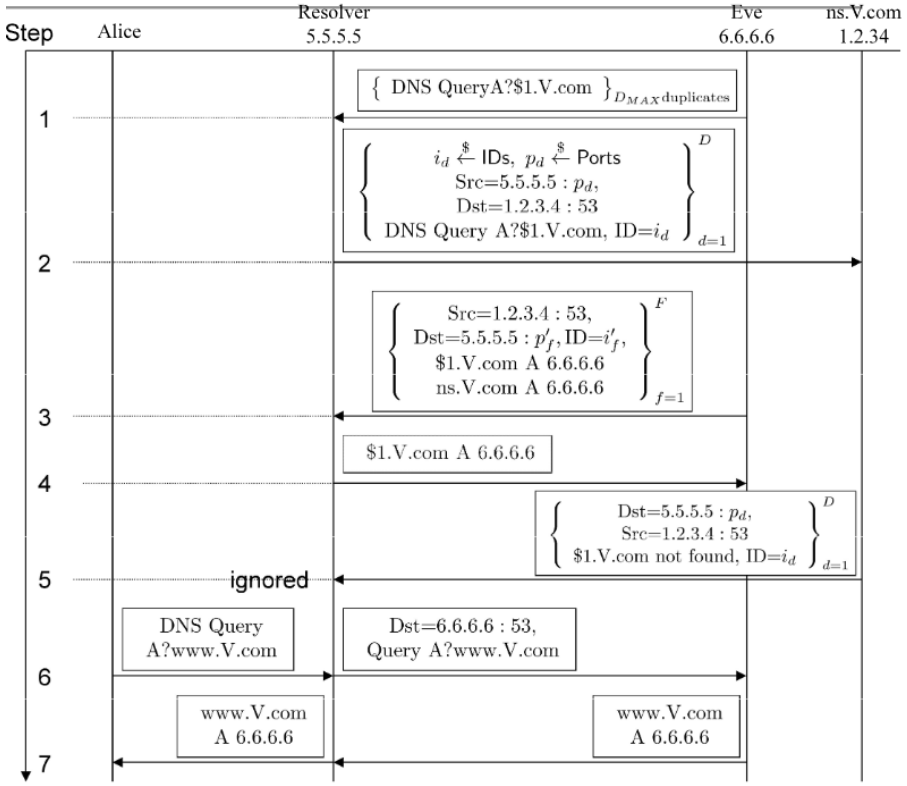
FATCAT DNS ★★ (10 points)

3. **(5 points) (CTF) Simple Cache Poisoning ★★**
   王肥貓同學寫了一個簡單的 DNS forwarder: FATCAT DNS (其 source code 為 `server-a.py`)，並把它架設在 `140.112.91.4:48765`。預設的 DNS resolver 在 `140.112.30.191`，當 FATCAT DNS 不知道 query 的答案，它便會轉而詢問之，而兩個 server 間使用簡化版的 DNS protocol 進行 query/response 的傳輸 (詳見 `utils.py`)。此外，FATCAT DNS 使用 cache 來儲存 DNS queries 的答案，以減少詢問它預設的 DNS resolver 的次數。
   作為肥貓向你發出的挑戰，只要你能污染 (poison) FATCAT DNS 的 cache，使得 cache 中 `www.google.com` 的 A record 變成 `11.4.5.14`，FATCAT DNS 就會輸出 `FLAG1`。

   - **Requirement:**
     - 修改提供的 skeleton code `solve-a.py` 來對 FATCAT DNS (位於 `140.112.91.4:48765`) 進行 cache poisoning attack，並取得 `FLAG1`。
       請在 Google form 提交你修改後的 `solve-a.py` 與 `FLAG1`。
   - **Restrictions:**
     - 你的 script 在執行時能使用 TCP 連線到 FATCAT DNS 至多一次，UDP 則不限次數。(這是為了避免你藉由重新連線到 FATCAT DNS 來清空 cache。)
     - 你的 script 可以使用任何你喜歡的 Python module。
   - **Hints:**
     - `utils.py` 定義了 FATCAT DNS 與它預設的 DNS resolver 間傳送的 DNS query/response packets 的格式，包含 record type (QTYPE) 與 response code (RCODE) 的種類。你無須看懂裡面的函數是如何實作的。
     - 要怎麼在短時間內送出大量封包？

4. **(5 points) (CTF) Kaminsky has come to the rescue! ★★**
   為了避免你再次成功冒充預設的 DNS resolver，肥貓將 transaction ID 的選取範圍由 0 ∼ 255 擴大至 0 ∼ 65535 (這也是真正的 DNS protocol 選取的範圍)。新的 FATCAT DNS 架設於 `140.112.91.4:48766`，且其 source code 為 `server-b.py`。
   這時你的好同學條絲向你透露了比直接污染目標 domain 更有效率的攻擊手段：Kaminsky's attack[1]。這個攻擊手段讓攻擊者不再需要等待 cache 過期，也能污染整個 zone。此攻擊手段的循序圖 (sequence diagram) 如下[2]：

[1] 詳見 http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html
[2] 圖片取自 https://github.com/ShlomiRex/kaminsky-attack



已知條絲在 `140.112.30.185` 架好一台用來冒充預設 resolver 的惡意 resolver ，該 resolver 會針對任何 parent domain 為 `nasa.csie.org` 的 DNS query 回覆惡意答案。請接著進行 Kaminsky's attack，讓 FATCAT DNS 向條絲的 resolver 詢問 `team{你的隊伍編號}.nasa.csie.org` 的 TXT record，條絲的 resolver 會將 `FLAG2` 放在 DNS response 的答案中。

- **Requirement:**
  - 修改提供的 skeleton code `solve-b.py` 來對新 FATCAT DNS (位於 `140.112.91.4:48766`) 進行 Kaminsky's attack，並取得 `FLAG2`。`FLAG2` 為條絲的 resolver (位於 `140.112.30.185`) 對 domain `team{你的隊伍編號}.nasa.csie.org` 回覆的 TXT record。假如你是第 5 隊，`FLAG2` 就是條絲的 resolver 回覆 `team5.nasa.csie.org TXT` 的答案。
    請在 Google form 提交你修改後的 `solve-b.py` 與 `FLAG2`。
- **Restrictions:**
  - 禁止透過直接向條絲的 resolver (位於 `140.112.30.185`) 發送 DNS query 來取得 `FLAG2`。
  - 禁止連線到舊 FATCAT DNS (位於 `140.112.91.4:48765`)。
  - (其他規定同前一小題。)
- **Hints:**
  - `server-a.py` 與 `server-b.py` 只差在選取 transaction ID 的範圍。
  - (其他提示同前一小題。)

Submission

提交 Google form。

5 Docker & Docker Compose ★ ∼ ★★★ (25 points)

Description

本題將引導同學撰寫 `Dockerfile` 與 `docker-compose.yml`，完成各項環境設定，並成功部署一個全端網站系統。

題目共分為兩大部分：

- 第 1∼2 小題為第一部分，著重於容器的建立與資源限制設定。
- 第 3∼5 小題為第二部分，聚焦於網站的容器化部署。

兩部分可分開作答。

Submission

請找 TA 進行 Demo。

Tasks

1. SSH Container 建立 ★ (5 points)

請撰寫一份 Dockerfile，並使用 `docker build` 建立一個 Ubuntu 22.04 的 SSH 容器，使我們可以在不使用 `-p` port 映射的情況下，從 host 主機直接透過以下指令登入 container。

- 使用 Dockerfile 建立映像檔，並使用 `docker build` 編譯。不得直接使用 `docker run -it ubuntu bash` 這種方式手動設定。
- 映像檔啟動後，應自動啟動 SSH server。
- 建立一個使用者帳號 `user`，密碼同為 `user`。
- 建立並執行一個名為 `myubuntu` 的 container。
- 在 host 主機上執行 `ssh myubuntu` 應能登入並進入 shell，相關的連線設定同學應自行處理。
- **禁止**使用 `-p`、`--publish` 等任何形式的 port 映射。
- **Hint:** 回想 lab 中教過的內容。建議在 Linux 系統下進行，可能需要 root 權限。在 Windows 或 macOS 上可能會遇到問題。

2. 限制 CPU 使用率 ★ (5 points)

承上題，為了避免單一 Container 佔滿主機的資源，我們可以透過 Docker Container 來限制資源。仿照上題的設定，但要讓你的 Container 在每個 1-second scheduling period 中，最多只能取得至多 500,000 µs 的 CPU 執行時間。換句話說，容器的理論峰值 CPU 使用率不得超過 50% of 1 vCPU。

測試方法

```
#!/bin/bash
x=1
while [ $x -le 1000000 ]
do
  x=$(( $x + 1 ))
done
```

請將上面的程式碼存檔成 `while.sh`，並對第一小題開的 container 和第二小題開的 container 進行測試，並 show 出實際的不同。

Hint: 可以對兩個容器分別執行 time bash while.sh 來測量程式碼跑了多久。

3. 前後端程式碼容器化 ★★★ (10 points)

在實際開發中，團隊常需將前後端服務容器化，使彼此可以獨立建構並部署。在本題中，請分別為提供的前端與後端程式碼撰寫 Dockerfile，並能成功使用 `docker build` 與 `docker run` 啟動對應容器。本小題不要求整個網站需成功運作，只需確保每一個容器可獨立啟動並提供正確的頁面或 API。

請下載 nasa-final-docker-app 資料夾，解壓縮後會看到以下內容：

- `frontend/`：React 製作的前端應用，使用 `npm run build` 產生靜態網站，再由 `serve` 提供服務。
- `backend/`：Node.js + Express 製作的後端 API 伺服器，使用 `npm start` 執行。
- `script.sql`：初始 SQL 指令，用於建立資料表與初始資料。

具體要求

- 分別在 `frontend` 和 `backend` 資料夾中撰寫 `Dockerfile`。
- 使用 `docker build` 成功編譯出映像檔。
- 使用 `docker run` 啟動容器後，能使用 `curl` 或瀏覽器造訪對應的 port，成功看到頁面或回傳資料，即視為通過。
- frontend 應開放 port 3001，backend 應開放 port 3000。
- 不要求此時前後端與資料庫完成整合。

Frontend（React）

- Base image：`node:20`
- 工作目錄：`/app`
- 執行命令：
  - `npm install`
  - `npm run build`
  - `npx serve -s build -l 3001`
- Port：3001

Backend（Node.js API）

- Base image：`node:20`
- 工作目錄：`/app`
- 執行命令：
  - `npm install`
  - `npm start`
- Port：3000

4. 前後端整合與資料驗證 ★★ (2.5 points)

請完成前後端與資料庫的串接，確保網站能正常運作，並能成功將表單資料寫入資料庫中。你應使用 `docker run` 啟動一個 MySQL 容器，並使用提供的 `script.sql` 自動建立資料表與資料內容。最後，請透過登入資料庫進行查詢，以驗證資料是否成功寫入。

Requirements

- 前後端需能與資料庫正確串接，使用者填寫表單後應能送出資料並寫入 DB。
- 使用 `mysql:8.0` 映像建構資料庫容器。
- 資料庫名稱為 `appdb`。
- root 密碼與一般使用者密碼設成相同（詳見程式碼中的環境變數）。
- 容器開放 port：3306。
- 啟動容器時自動執行 `script.sql`，初始化資料庫內容。
- 同學需登入資料庫，透過 SQL 查詢確認資料有成功寫入，並以 CLI 結果為驗證依據。

驗證方式

請執行下列指令並觀察是否可正確查詢出資料：

```
mysql -h localhost -P 3306 --protocol=TCP -u root -p
SHOW DATABASES;
USE appdb;
SHOW TABLES;
SELECT * FROM <TABLE_NAME> LIMIT 10;
```

5. 使用 docker-compose 完整建置服務 ★★ (2.5 points)

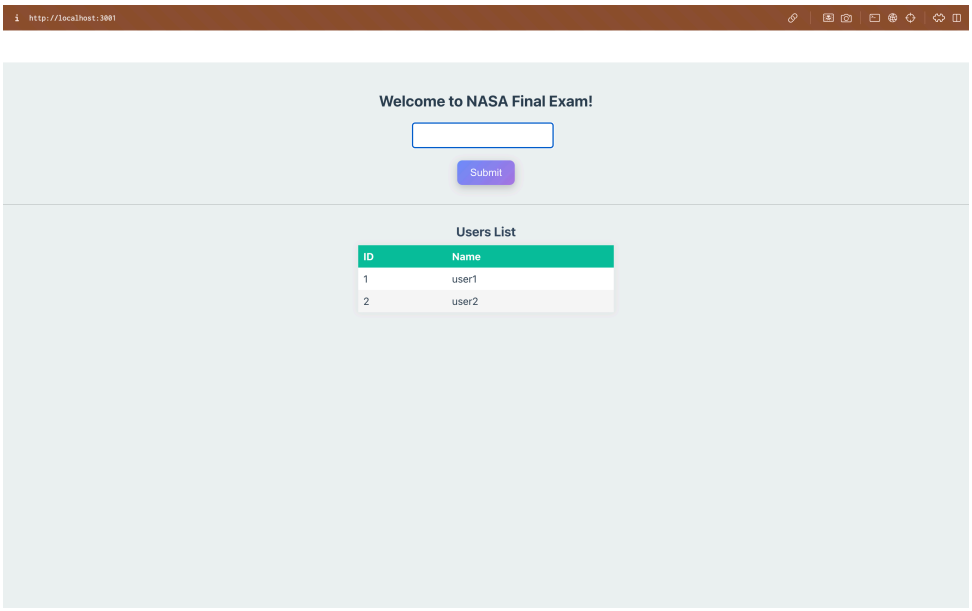
請撰寫一份 `docker-compose.yml`，讓整體架構可以透過單一指令自動建置與啟動，包括：

- frontend
- backend
- database（MySQL）

完成後，執行下列指令即可部署整組系統：

```
docker compose up -d --build
```

此時造訪 `http://localhost:3001`，網站應成功載入並正常運作（包含 API 串接與資料庫互動），如下圖所示：



---

6 LLM ★★ ∼ ★★★★(20 points)

Description

Open-LLM-VTuber 是一個有趣的 AI 系統。系統可以創造出一個可以客製化的語音互動式 AI 夥伴，不僅支援即時語音對話與視覺感知，還搭配生動的 Live2D 虛擬角色。所有功能皆可完全離線在您的電腦上運行！

您可以把它當作個人專屬的 AI 夥伴——無論您想要虛擬女友、男友、可愛寵物或其他任何角色，它都能滿足您的期待。此專案全面支援 Windows、macOS 和 Linux，並提供兩種使用模式：網頁版與桌面應用程式（特別支援透明背景桌寵模式，讓 AI 夥伴隨時陪伴在您螢幕的任一角落）。

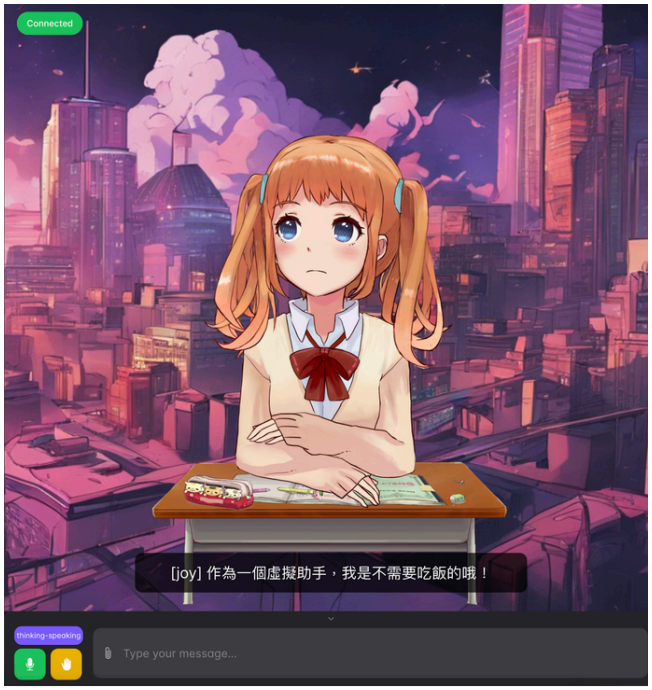
不過跟很多專案一樣，因為它持續在進行開發中，所以許多文件、程式碼等等可能不見得預設就可以運作得很好。另外，這個專案因為高度模組化，讓使用者在不同的功能上可以選擇不一樣的選擇方案，因此在安裝、設定上就相對比較複雜。例如在原本在 repository 中的 dockerfile 似乎無法產生。

專案的相關資源可見其 GitHub 頁面：https://github.com/Open-LLM-VTuber/Open-LLM-VTuber

在這個題目中，我們希望大家可以使用數個 docker container 完成一組完整的服務（題目的最後面有提示哦）。

1. 基本要求：(10 分) ★★

1. 安裝並使 v1.1.0 版本的 Open-LLM-VTuber 可以在 docker container 中正常運作，透過 http://localhost:12393 或者遠端網址連線到它的 Web GUI。https://github.com/Open-LLM-V



Tuber/Open-LLM-VTuber/releases/tag/v1.1.0

2. 使用 faster-whisper 作為語音辨識服務。請不要使用預設的 sherpa_onnx_asr。faster-whisper 請使用 base 或者更好的模型。(參見附註一)
3. 使用 ollama 作為大型語言模型推論引擎。請使用 gemma3:4b-it-qat 作為使用的語言模型（如果算力太低，可以允許使用 gemma3:1b-it-qat。如果算力充足，也可以使用更大的 12b-it-qat 或者 27b-it-qat 模型。）。基礎要求中只要求使用自己架設的 ollama、可使用本機安裝。
4. 使用 docker 中的 volume，使得 `conf.yaml` 設定檔以及 `models` 目錄可以對應到本地的目錄。這樣方便系統管理者更新設定、管理模型的儲存、並且使得在更新 container 時仍能保留這些檔案。

完成基本要求，在申請 demo 時需要繳交這些檔案/展示這些：

- 生成 open-llm-vtuber 的 docker container 的 dockerfile。這個 dockerfile 必須要能夠以 `docker build .` 完成 container 編譯。
- 請撰寫一隻 `docker-start.sh` 的 shell script，裡面包含正確的 `docker run` 指令以創造並啟動 open-vllm-vtuber container。
- **demo** 請展示你可以用語音與虛擬人物互動。
- **demo** 請展示本地的 `conf.yaml` 及 `models/` 目錄內容。
- **demo** 請展示 docker container 的 log，其中顯示你在使用正確的語音辨識模組以及大型語言模型。

2. 進階要求：(10 分) ★★★★

PS. 建議是可以直接三題一起寫會比較輕鬆一點（by 驗題者）

1. (3 分) 請將 ollama 也安裝在一個 container 中（而非本機），並使得前面的 open-llm-vtuber container 可以使用這個 container 中的 ollama 服務。這個 ollama container 也應該 docker volume 掛載本地的目錄，並存放模型檔案於其中。
   demo 請展示 open-llm-vtuber 的 `conf.yaml`，並解釋如何讓其可使用 ollama container 的服務
2. (4 分) 請設定一個 nginx reverse proxy 的 container，使得你可以透過 secure websocket 及 https 來連線 open-llm-vtuber 服務（遠端 IP 必須透過這樣的服務才能夠正常使用，見附註二）。注意你可能要自己生成可用的憑證。
   requirement 請繳交你的 `nginx.conf` 檔案。
   demo 請展示你的服務可以以遠端 IP（非 localhost）正常使用、與虛擬人物以語音互動。
3. (3 分) 請將上面的三個服務(open-llm-vtuber, ollama, nginx)結合成一個完整的 `docker-compose.yaml` 檔案來管理，並且可以透過 `docker-compose up -d` 來啟動所有服務。
   requirement 請繳交你的 `docker-compose.yaml` 檔案。
   demo 請展示你可以用 `docker-compose up -d` 啟用服務，並且在啟用後可以正常與虛擬人物以語音互動。

Submission

請找助教進行 demo

Hint

- `conf.yaml` 中的 `system_config:` 底下的 `host:` 可能要設定為 `0.0.0.0` 才能夠讓「docker container 自己以外其他機器」連線到網頁介面。
- 雖然 repository 中的 dockerfile 無法直接拿來運用，但是仍然是好的參考資料。
- 專案的文件「快速開始」中，有在一般環境中安裝的步驟。這些步驟是正確的。你是否可以將這些步驟轉化為 docker container 中的安裝步驟？
- https://github.com/astral-sh/uv 是一個這個專案用到的套件管理工具。要如何在 container 中安裝？

Note

- 附註一：`conf.yaml` 中關於 faster-whisper 的設定可參考下方設定。若已經正確設定，第一次會自動下載對應的模型檔案到下方 download_root 設定的目錄下。

```
faster_whisper:
    model_path: "base"
    download_root: "models/whisper" # models path
    language: "zh"
    device: "auto"
```

- 附註二：當使用非 `localhost` 的網址連線 open-llm-vtuber 的 web GUI 時，如果使用非 secure 的 protocol (`ws://`, `http://`) 連線，則無法使用麥克風。要正常可使用系統，請使用 `localhost` 或者用 `wss` / `https` 協定。
- 附註三：在網頁中，左上角的齒輪按開後，WebSocket URL 與 Base URL 這兩個欄位需要注意是否正確設定。注意如果是 Secure Web Socket 的話，網址是用 `wss` 開頭而不是 `ws` 開頭（注意有兩個 s）。另外當然如果是 Secure HTTP 的話，網址就是用 `https` 開頭。

- 附註四：`conf.yaml` 中 `character_config:` 底下的 `persona_prompt:` 的內容決定了角色回應的語言以及角色的個性等等。可以隨意地換成別的文字。