Homework 4
Web

huytseng, jschou

國立陽明交通大學資工系資訊中心

Computer Center of Department of Computer Science, NYCU

Outline

- `preflight`: HTTP Preflight Checks
- `4-1`: HTTP File Server (50%)
- `4-2`: Reverse Proxy (20%)
- `4-3`: Health Check API (15%)
- `4-5`: POST Data (15%)
- `4-4`: Web Crawler (Bonus 20%)

Requirement (1/2)

- For each stage, the **`$BASE_DIR`** is defined as **`/home/judge/hw4/$STAGE_NAME`**, where **`$STAGE_NAME`** is in format **`/4-\d/`**.
  - **`$SRC`** is defined as **`$BASE_DIR/src`**.
  - **`$DATA`** is defined as **`$BASE_DIR/data`**.
- Your hand written source codes, configurations and executables should be placed in **`$SRC`**, no other files allowed there.
- For any metadatas, templates or static data, place it in **`$DATA`**.
- No any executables outside **`$SRC`** or **`$DATA`** should be called, unless it is:
  - Generated by programs inside these directories.
  - Installed with package manager.
  - Can be auto-installed with a script.
  - Explicitly required by the stage.

Requirement (2/2)

- If a executable is automatically installable, or it is install with a package manager, an installation script should be provided in **`$BASE_DIR/init.sh`**
- Do not put auto-generated contents (e.g. **`node_modules`**) inside these pre-defined (that is, **`$SRC`** and **`$DATA`**) directories.
- Please modify your hosts file, make **`ca.nasa.nycu`** resolve to **`10.113.200.1`**

```
$ tree /home/judge/hw4/preflight/
/home/judge/hw4/preflight/
├── data
├── down
├── src
└── up
```

Example

```
#!/bin/sh

DOMAIN="$1"

cd /home/judge/hw4/preflight
./src/get-certs.sh "$DOMAIN"
cp ./data/config.template "./data/config"
./src/set-conf.sh "$DOMAIN" "./data/config"
./src/server-up.sh "./data/config"

exit 0
```

```
# /home/judge/hw4/preflight/up
```

HTTP Preflight Checks

Requirement

Given a **`$DOMAIN`** in format **`[a-zA-Z0-9\-]+\.$ID\.nasa\.nycu`**,
you should provide a HTTP server which meets the following requirements:

- Using a valid cert acquired from the ACME server
  - Appliable for the following stages, too.
  - Server: [**`https://ca.nasa.nycu:9000/acme/acme/directory`**](https://ca.nasa.nycu:9000/acme/acme/directory)
  - Root CA
- Auto redirect **`80`** port HTTP to **`443`** port HTTPS with HTTP **`302`**
- We will run
  - **`$BASE_DIR/up "$DOMAIN"`** when test begins.
  - **`$BASE_DIR/down`** when test ends.

HW 4-1: HTTP File Server (50%)

HW 4-1: Requirement (1/3)

In this stage, you should provide a HTTP File Server with the following requirements:

- Automatic HTTPS redirection
  - As the preflight requested
- Permission
  - **`$DATA/static/$USERNAME`** is only available for the user
  - **`$DATA/static/public`** is available for everyone
  - We will push files to these folders when the test start
  - We will clean it up when the test ends
  - No frontend preview / dir list is needed, we will access the file directly
  - Give HTTP **403** if no permission
  - Give HTTP **404** if the file does not exist

HW 4-1: Requirement (2/3)

- Login endpoint
  - **`/login`** should be provided
  - **`username`**, **`password`** will be provided in POST data
    - Both **`JSON`**、**`x-www-form-urlencoded`** should be supported
  - Return JWT token with HTTP **`200`** if success
  - Return HTTP **`403`** if username/password does not match
- The endpoint should allow both HTTP Basic auth and JWT auth
  - HTTP Basic: Using **`Authorization: Basic $SECRET`**
  - JWT: Using **`Authorization: Bearer $TOKEN`**
- File endpoint
  - **`fs:$DATA/static/*`** is mapped to **`hfs:/*`**
    e.g. **`$DATA/static/leko/my_file`** -> **`https://test.69.nasa.nycu/leko/my_file`**

HW 4-1: Requirement (3/3)

- Basic Auth
  - Return HTTP 403 if username/password does not match
- JWT Token
  - Sign the token with HS256
  - Use the secret assigned by server
  - payload should be **`{"user": "$USERNAME"}`**
  - Should validate the token
- We will run
  - **`$BASE_DIR/up [-d "$DOMAIN"] [-p "$JWT_SECRET"] [-u "$USER:$PASSWORD_HASHED_IN_BCRYPT"] ...`** when test begins.
  - **`$BASE_DIR/down -d "$DOMAIN"`** when test ends.

HW 4-2: Reverse Proxy (20%)

國立陽明交通大學資工系資訊中心

Computer Center of Department of Computer Science, NYCU

HW 4-2: Requirement (1/2)

In this stage, you are required to provide a reverse proxy which proxies our HTTP API Server.

- Server: **`http://ca.nasa.nycu:4442`**
- Domain: Varient domain provided by server
  - We will run **`$BASE_DIR/up "$DOMAIN"`** when test begins.
  - **`$BASE_DIR/down`** when test ends.
- Check CORS
  - Dynamic allow domain depends on **`origin`** header
  - **`'*'`** is not allowed
  - Preflight request should be handled

HW 4-2: Requirement (2/2)

- Header
  - Remove to X-Powered-By header from response
  - Pass X-Forwarded-For header correctly
- Path mapping
  - replace all apis from **`{path}`** to **`/api/1.0/{path}`**
  - **`/query?k={k}&v={v}`** -> **`/api/1.0/query/{k}/{v}`**

HW 4-3: Health Check API (15%)

HW 4-3: Requirement

In this stage, you are required to write a simple health-check API

- Write a simple health-check API
  - **`$DOMAIN`** in format **`[a-zA-Z0-9\-]+\.$ID\.nasa\.nycu`**
  - **`GET /`** should return json response:

```
{
    "disk": <free space / total space> (float, 0~1, error=+-0.01),
    "uptime": <system uptime> (int, in second, error=+-10),
    "time": <current time> (int, epoch in second, error=+-10)
}
```

- We will run
  - **`$BASE_DIR/up "$DOMAIN"`** when test begins.
  - **`$BASE_DIR/`**down when test ends.

HW 4-5: POST Data (15%)

HW 4-5: Requirement (1/2)

In this stage, you are required to post data using three different **`Content-type`**.

We provide three endpoint for you to send request.

We will run **`$BASE_DIR/up "$SOURCE_URL"`** and check the result.

- HTTP POST to **`$SOURCE_URL/json`** and get **`secretKey`**
  - Using **`application/json`**
  - Body: **`{"keyword: "give_me_secret_key"}`**

HW 4-5: Requirement (2/2)

- HTTP POST with **`secretKey`** from previous step to **`$SOURCE_URL/urlencoded`** and get **`secretFile`**.
  - Using **`application/x-www-form-urlencoded`**
  - Body: **`{"secretKey": secretKey}`**
- HTTP POST with **`secretFile`** from previous step to **`$SOURCE_URL/multipart`**
  - Using **`multipart/form-data`**
  - Body: **`{"secretFile": secretFile}`**

HW 4-4: Web Crawler (Bonus 20%)

HW 4-4: Requirement

In this stage, you are required to write a web crawler which transforms provided XML data into a JSON.

The XML source checks if user is a human, also, it only allows its LAN (**`192.168.69.0/24`**) to access it.

Hint: How does HTTP Proxy work?

HW 4-4: Requirement (1/2)

We will run `$BASE_DIR/up "$SOURCE_URL"` and judge your answer.

- HTTP GET `$SOURCE_URL` for the question
- HTTP POST your answer to `$SOURCE_URL`
  - Using `application/json`
  - Body is your whole JSON object

HW 4-4: Requirement (2/2)

- An example server is provided at **`http://ca.nasa.nycu:4444`**
    - HTTP GET **`/`** for the exmaple question
    - HTTP POST your answer to **`/`**
        - Using **`application/json`**
        - Body: **`{"q": "$QUESTION_XML", "a": $ANSWER_JSON_OBJECT}`**
        - HTTP **`200`** for correct answer
        - HTTP **`400`** for wrong answer
        - HTTP **`500`** for wrong input
        - We provide three set of example XML and expected JSON for reference

Attention!

- Your work will be tested by Online Judge system.
  - You can submit multiple judge requests. However, OJ will cool down for several minutes after each judge.
  - **We will take the last submitted score instead of the highest score.**
  - Late submissions will not be accepted.
- BACKUP your server before judge EVERY TIME
  - We may do something bad when judging.
- Make sure everything is fine after reboot.

Attention!

- TAs reserve the right of final explanations. Specs and the points of each sub-judges are subject to change in any time.
- We might randomly pick some student to demo after end of HW4.
- **Start from 2022/12/1 21:00**
- **Deadline 2022/12/21 23:59**

Help me!

Questions about this homework

- Ask them on https://groups.google.com/g/nctunasa
- We MIGHT give out hints on google group
    - Be sure to join the group :D
    - When posting a question, be sure to include all information you think others would need
        - including but not limiting to your ID, setups, configurations and / or what you have done to trace the error / problem
- Do not email us
- Do not use e3 to email us

Recommended Tools & Tips (1/2)

- HTTP Server
  - caddy
  - nginx
- ACME Client
  - acme.sh
  - certbot
- API Server
  - Express
  - Flask
  - Gin
- HTTP Client
  - axios
  - curl
  - requests
- Other
  - gsed
  - nohup
  - pkill

Recommended Tools & Tips (2/2)

- Check **`net.inet.ip.portrange.reservedhigh`** out if your ACME client complains about insufficient permission to open port.
- In most cases you do not want to block the **`up`** script unless you are the client. That is, your server should run in background instead of blocking the script. In this case, preventing server being killed after exiting the script is required. Check **`nohup(1)`** out for further information.
- Saving the PID of background-running processes after executed and detached it can be helpful for the **`down`** script to clean up.

Good Luck!