CS 551: Assignment 2 Ishaan Roychowdhury

Part 2: Dockerfiles

My Dockerfile was structured like this:

```
FROM alpine

RUN adduser -D iroycho-clc

RUN apk update && apk add htop

USER iroycho-clc

ENTRYPOINT ["htop"]
```

When I ran "docker build .", I got a hash when the build was complete. Then I ran the command "docker run -it (hash)". At this point, I could use the arrow keys to move the columns left and right and see the values in the TIME+ column increasing.

Attached is a screenshot, of the same:

Part 3: Running a RESTful Go Microservice

Curl commands:

```
-/Desktop/cloud-docker — -zsh

(base) ishaanrc@ishaan cloud-docker % curl -d "username=ProfComer&email=comer@cs.purdue.edu&address=West
Lafayette, IN" -H "Content-Type: application/x-www-form-urlencoded" -X POST http://localhost:8080/adduser

{"message":"Created user ProfComer (4) with email comer@cs.purdue.edu and address West Lafayette, IN", "stl
atus":"success"}

(base) ishaanrc@ishaan cloud-docker %

(base) ishaanrc@ishaan cloud-docker % curl -d "username=ProfComer&email=comer@cs.purdue.edu&address=West
Lafayette, IN" -H "Content-Type: application/x-www-form-urlencoded" -X POST http://localhost:8080/adduser

{"message":"Created user ProfComer (5) with email comer@cs.purdue.edu and address West Lafayette, IN", "stl
atus":"success")

(base) ishaanrc@ishaan cloud-docker %

(base) ishaanrc@ishaan cloud-docker % curl -d "username=ProfComer&email=comer@cs.purdue.edu and address West Lafayette, IN", "stl
atus":"success")

(base) ishaanrc@ishaan cloud-docker % curl -d "username=ProfComer&email=comer@cs.purdue.edu and address West Lafayette, IN", "stl
atus":"success")

(base) ishaanrc@ishaan cloud-docker % curl -d "username=ProfComer&email=comer@cs.purdue.edu and address West Lafayette, IN", "stl
atus":"success")

(base) ishaanrc@ishaan cloud-docker % curl -d "username=ProfComer&email=comer@cs.purdue.edu and address West Lafayette, IN", "stl
atus":"success")

(base) ishaanrc@ishaan cloud-docker % curl -d "username=ProfComer&email=comer@cs.purdue.edu and address West Lafayette, IN", "stl
atus":"success")
```

Web server logs:

```
(base) ishaanrc@ishaan user
CONTAINER ID IMAGE
                                   -service % docker ps
                                      COMMAND
                                                    CREATED
                                                                         STATUS
                                                                                            PORTS
                                                                                                                            NAMES
                   5c7749c22848
398fddbfb66f
                                      "./main"
                                                    2 minutes ago
                                                                        Up 2 minutes
                                                                                           0.0.0.0:8080->8080/tcn
                                                                                                                           hopeful
bhaskara
(base) ishaanrc@ishaan user-service % curl http://localhost:8080/randomuser {"user":{"Address":"123 Main St","Email":"john.doe@example.com","Id":1,"Username":"johndoe"}}¾ (base) ishaanrc@ishaan user-service %
(base) ishaanrc@ishaan user-service
(base) ishaanrc@ishaan user-service %
(base) ishaanrc@ishaan user-service % curl -X POST -d "username=johndoe1&email=john1.doe@example.com&addr
ess=12345 Main St" http://localhost:8080/adduser
{"message":"Created user johndoe1 (2) with email john1.doe@example.com and address 12345 Main St","status
 :"success"}%
(base) ishaanrc@ishaan user-service %
(base) ishaanrc@ishaan user-service % curl http://localhost:8080/getuser/1
{"user":{"Address":"123 Main St","Email":"john.doe@example.com","Id":1,"Username":"johndoe"}}
(base) ishaanrc@ishaan user-service % curl http://localhost:8080/getuser/2
{"user":{"Address":"12345 Main St","Email":"john1.doe@example.com","Id":2,"Username":"johndoe1"}}
(base) ishaanrc@ishaan user-service % ■
```

Here we can see 200 status for GET and POST requests that we are trying to handle. [GIN-debug] POST /adduser --> main.addUser (3 handlers) [GIN-debug] [WARNING] You trusted all proxies, this is NOT safe. We recommend you to set a value. Please check https://pkg.go.dev/github.com/gin-gonic/gin#readme-don-t-trust-all-proxies for details. [GIN-debug] Listening and serving HTTP on :8080 [GIN] 2023/09/15 - 21:27:16 "/randomuser" 400 88.291µs 172.17.0.1 | **GET** [GIN] 2023/09/15 - 21:27:33 200 465.25µs 172.17.0.1 **POST** "/adduser" "/randomuser" [GIN] 2023/09/15 - 21:27:41 200 254.167µs 172.17.0.1 **GET** "/randomuser" [GIN] 2023/09/15 - 21:27:56 200 211.791µs 172.17.0.1 **GET** "/getuser/2" [GIN] 2023/09/15 - 21:28:17 400 400.792µs 172.17.0.1 GET 200 "/adduser" [GIN] 2023/09/15 - 21:28:34 676.417µs 172.17.0.1 **POST** [GIN] 2023/09/15 - 21:29:25 200 650.708µs "/randomuser" 172.17.0.1 GET "/adduser" [GIN] 2023/09/15 - 21:29:53 200 268.917μs 172.17.0.1 **POST** "/getuser/1" [GIN] 2023/09/15 - 21:30:14 200 317.167µs 172.17.0.1 GET [GIN] 2023/09/15 - 21:30:19 200 194.25µs 172.17.0.1 **GET** "/getuser/2"

Part 4: Security of Public Containers Analysis

Some of my public containers had the following vulnerabilities:

Container 1:

1. CVE-2023-24540

Severity: 9.8 Critical

- **Description:** Some JavaScript whitespaces aren't recognized as such, potentially allowing unsanitized template execution.
- **Mitigation:** Update the affected software to the latest version where the issue is fixed or sanitize JavaScript inputs thoroughly to consider all whitespace characters.

2. CVE-2022-23806

Severity: 9.1 Critical

- **Description:** In certain Go crypto operations, an invalid big.Int value might be wrongly considered valid.
- **Mitigation:** Upgrade Go to versions 1.16.14 or 1.17.7 and later to patch this vulnerability.

Container 2:

1. CVE-2023-31047

Severity: 9.8 Critical

- **Description:** Django versions mentioned could allow multiple file uploads without validating all files.
- **Mitigation:** Upgrade Django to versions 3.2.19, 4.1.9, or 4.2.1 and later to ensure proper file upload validation.

2. CVE-2023-36053

Severity: 7.5 High

- **Description:** Django's email and URL validation can be exploited to cause denial of service using specially crafted domain names.
- **Mitigation:** Update Django to versions 3.2.20, 4.1.10, or 4.2.3 and later to prevent this potential ReDoS attack.

3. CVE-2023-39417

Severity: 8.8 High

• **Description:** PostgreSQL has a SQL injection flaw when using certain extension script constructs.

• **Mitigation:** Avoid using vulnerable, trusted, non-bundled extensions or ensure PostgreSQL and its extensions are updated to patched versions.

Container 3:

1. CVE-2022-23219

Severity: 9.8 Critical

- **Description:** The sunrpc module in the GNU C Library (glibc) can have a buffer overflow due to the unchecked copying of its hostname argument.
- **Mitigation:** Upgrade the GNU C Library to a version beyond 2.34. Ensure applications using this library have stack protection enabled to prevent potential arbitrary code execution.

2. CVE-2021-33574

Severity: 9.8 Critical

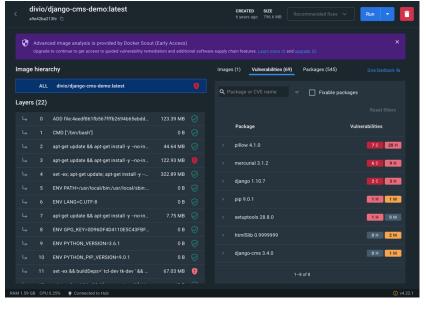
- **Description:** The **mq_notify** function in glibc might access a thread attributes object after its memory has been freed, causing potential crashes or other unspecified issues.
- **Mitigation:** Update the GNU C Library to a version later than 2.33 to ensure this use-after-free vulnerability is patched.

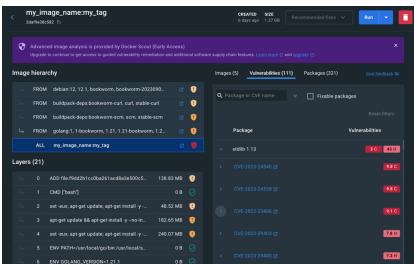
3. CVE-2022-41903

Severity: 9.8 Critical

- **Description:** Git has an integer overflow issue during commit formatting, potentially leading to arbitrary heap writes and code execution when using certain commands or attributes.
- Mitigation: Update Git to versions released on or after 2023-01-17. If updating isn't immediately possible, disable git archive in untrusted repositories and ensure daemon.uploadArch is set to false if git archive is exposed via git daemon.

Screenshots of the vulnerabilities:





Part 5: Securing Your Container

So, I re-wrote my Dockerfile in part 2 making it more secure.

```
FROM alpine:3.14

RUN adduser -D -h /nonexistent -s /sbin/nologin iroycho-clc

RUN apk update && apk add --no-cache htop && rm -rf /var/cache/apk/*

USER iroycho-clc

ENTRYPOINT ["htop"]
```

I decided to add the following security additions:

- 1) **Base Image Version added**: To avoid potential bugs or vulnerabilities that exist in using "latest" or just "alpine", we can use the exact version number to remove any possibility of buggy software.
- 2) Clean up after installation: To minimize the image size and reduce our attack surface, we clean up all temporary files, caches, etc. so attackers can't manipulate it later.
- 3) No home directory: To minimize the attack surface, we don't give the user a valid home directory. This removes the potential for attackers to gain access to temp files, configuration files, etc.
- 4) Deny shell access: This means even if a user uses "su" or similar command to switch users, he will not be given shell access. This prevents malicious actions that can be taken by an attacker.