2. What are nginx header security and its uses. Also implement in the test.conf file.

The HTTP headers are the additional information passed between the client (browsers) and web servers. The security HTTP headers are the response HTTP headers that the server can add in order to increase the security of HTTP exchange. Each security header serves its own purpose. Here are a few http security headers, with their uses:

- **HTTP Strict Transport Security (HSTS)** - It is a way to deal with potential vulnerability by instructing the browser that a domain can only be accessed using HTTPS. It can be implemented as:

  add_header Strict-Transport-Security "max-age=1000; includeSubDomains" always;

  The *max-age* is the time in seconds for which the information is cached. The *includeSubDomains* option tells that this rule is applied to all subdomains of the main domain.

- **X-Frame-Options** - It is used to defend our website from clickjacking attacks by disabling iframes on our site. There are three ways to configure X-Frame-Options:

  DENY **:** This will disable iframe features completely.

  SAMEORIGIN **:** iframe can be used only by someone of the same origin.

  ALLOW-FROM **:** This will allow pages to be put in iframes only from specific URLs.

  It can be implemented as:

  Add_header X-Frame-Options “DENY” always;

- **X-XSS-Protection** - It is used to defend against Cross-Site Scripting attacks. This header stops pages from loading when they detect reflected cross-site scripting (XSS) attacks. It can be implemented as: add_header X-XSS-Protection "1; mode=block";

  X-XSS-Protection: 0 : It will disable the filter entirely.

  X-XSS-Protection: 1 : It enables the filter but only sanitizes potentially malicious scripts.

X-XSS-Protection: 1; mode=block : It enables the filter and completely blocks the page.

- **X-Content-Type-Options** - It is used to prevent web browsers from sniffing a response away from the declared Content-Type. It can be implemented as:

  add_header X-Content-Type-Options nosniff;

- **Content-Security-Policy** - It is an improved version of the X-XSS-Protection header and provides an additional layer of security which is used to prevent XSS and data injection attacks. It is used to filter the file types that are accepted. It can be implemented as:

  add_header Content-Security-Policy "default-src 'self'; font-src *;img-src * data:; script-src *; style-src *";

- **Permissions-Policy** - It allows a site to control which APIs or features can be used in the browser. Like controls of individual hardwares like mic and webcam. It can be implemented as:

  add_header Permissions-Policy "geolocation=(), midi=(), sync-xhr=(), microphone=(), camera=(), fullscreen=(self), payment=()";

- **X-Permitted-Cross-Domain-Policies** - used to allow PDF and Flash documents for cross-domain requests.
- **Others** : There are many more security headers like Public Key Pinning Extension for HTTP (HPKP), Referrer policy and so on.

In our NginxTask.conf file, we applied following security headers:

Nano /etc/nginx/nginx.conf

```
server {
        listen 80;
        server_name www.bijaykandel37.com;
        root /var/www/html/NginxTask;
        }

        ##    Added security headers    ###
        add_header X-Frame-Options "SAMEORIGIN";
        add_header X-XSS-Protection "1; mode=block";
        add_header X-Content-Type-Options "nosniff" always;
        add_header Permissions-Policy "geolocation='none', microphone='no>
        add_header Content-Security-Policy "default-src 'self'; font-src >
```

To test the updated configurations:

Nginx -t

And to restart nginx:

Systemctl restart nginx

Now when the browser is opened with the same link, we can see the activated security headers by inspecting.