Browser Header Security

-> There is some security issues that server can not control, so we use header to controls them
-> headers give instruction to the client to tell them how they should behave
-> headers are client side control, so it is up to the browser to implement them.
-> browser does not recognize a specific header, browser will ignore it
-> known header to the browser, browser has to implement them.
-> caniuse.com to see what header + tags browsers supports
-> headers are additional layer of defense to server side controls
-> Developer most often do not care about headers

Non-Standard and browser prefixed Headers

X-Content-Security-Policy and it is replace by SOP -> X prefix indicates a non-standard header
X-WebKit-CSP and it is replace by CSP
X-XSS-Protection header it block browser XSS filter -> it is not standard + firefox does not support

Security Header Usage by the top 1 M Sites

-> 93% of the site do not force https and from 7%, there is 6% are using the HSTS header

HTTP Strict Transport Security (HSTS)

-> force the browser to send requests only through https
-> very easy to implement
//the http request will be translated to https request

```
Strict-Transport-Security: max-age:123123; includeSubdomains; preload
```

1. Declare that a client should only interact with a site over https. even if user set the schema to http, change it to https. just change everything to https
2. HSTS protects against "downgrade attacks". Downgrade attack, where the attack try to make the client to request a resource over a less secure protocol. HSTS stop this attack
3. relies on "trust on first use" TOFU -> what does that mean ?
   the first request ever will be like -> first request will be 301 redirected and then 200 https -> here the first request 301 and the 200 https is seen by attacker
   the requests after that will be like -> first request will be 307 internal redirected and then 200 https -> here only the 200 https request will be seen by the attack, the first request 307 will be no seen by attack !!!!
   this is very effective, because I do not have to be risked with sending each request to server and be vulnerable to the MiTM attack

problem here is that in the very first request, we may be vulnerable to MiTM attack.....

the max-age keyword

1. declare the period for which insecure request can not be made.
2. the units are in seconds
3. the duration is reset on every receipt of the response header

the includeSubdomains keyword

1. the Scope of HSTS can be extended to all subdomains, this will force all subdomains to request HTTPS. will save the first request in subdomains from MiTM attack
2. it is required for the preload attribute

the preload attribute

1. Trust on first use is not foolproof, so there is an opportunity to MiTM
2. Preloading HSTS hard-coded it into the browser

you need to make sure that you want people to request website securely. because the browser will force users to go over https but the website is not configured properly for https. the content will not be loaded to users

HTTP Public Key Pinning (HPKP)

this header make sure that the website serve a particular certificate, that has specifc creteria
HTTPS is great, but what if CA get compromised.
I go to a website with https and the site return a valid certificate but the certificate is not the one, that the website is meant to have. attacker has created the certificate and is able to intercept / modify the traffice (MiTM)
certificates: each certificate has an thumbprint identifier.
to avoid such attack like MiTM attack, we need to tell the browser that this is the certificate for this website, if you found something else reject it. HPKP does that.
-> HPKP relies on "Trust on first use" TOFU
-> IE does not support HPKP

Content Security Policy (CSP)

it can be understood as a policy that decides which scripts, images, iframes can be called or executed on a particular page from different locations. also a protection from XSS

```
default-src 'none';
img-src 'self';
script-src 'self' https://code.jquery.com;
```

```
style-src 'self';
report-uri /__cspreport__
font-src 'self' https://addons.cdn.mozilla.net;
frame-src 'self' https://ic.paypal.com https://paypal.com;
media-src https://videos.cdn.mozilla.net;
object-src 'none';
```

-> about 1500 website from 1 Million website use CSP

-> while testing check always the console in the browser

-> Declaring Content Sources by Keywords or Hosts
keywords : `*`, `none`, `self`
hosts: `example.com`,`*.example.com`
-> default-src
allow sources to load scripts on website

```
default-src 'self' https://ww.google.com
```

-> style-src
allow sources to load style files on website

```
style-src 'self' https://cdnjs.cloudflare.com
```

-> img-src
allow sources to load images on website

```
img-src 'self' https://google.com
```

-> font-src
allow sources to load fonts style on website

```
font-src 'self' https://google.com
```

-> object-src
allow sources to load objects on website

```
object-src 'none'
```

does not serve any requests for objects on the website

-> media-src
allow sources to load media on website

```
media-src 'none'
```

I disallow all media on website

-> child-src
allow sources to load iframes on website

```
child-src https://google.com
```

-> connect-src
allow sources to load fonts style on website

```
connect-src 'self'
```

-> unsafe inline & unsafe eval ??
csp gonna block any inline scripts like google analyses, so we use the keyword `'unsafe inline'`

Tools

https://securityheaders.com/

https://report-uri.com/home/analyse

https://report-uri.com/home/generate