OWASP
Open Web Application
Security Project

HTTP SECURITY HEADERS

(Protection For Browsers)

BIO

- Emmanuel JK Gbordzor

ISO 27001 LI, CISA, CCNA, CCNA-Security, ITILv3, …

11 years in IT – About 2 years In Security

Information Security Manager @ PaySwitch

Head, Network & Infrastructure @ PaySwitch

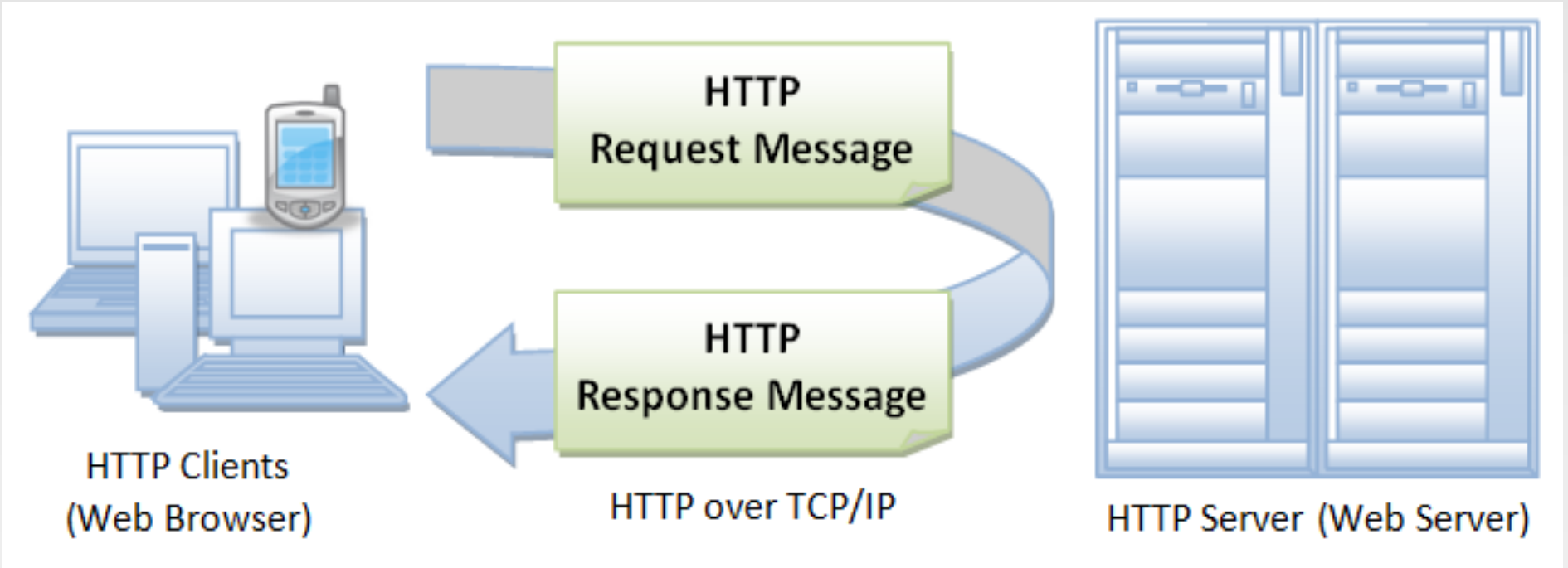
Head of IT @ Financial Institution

Bug bounty student by night – 1st Private Invite on Hackerone

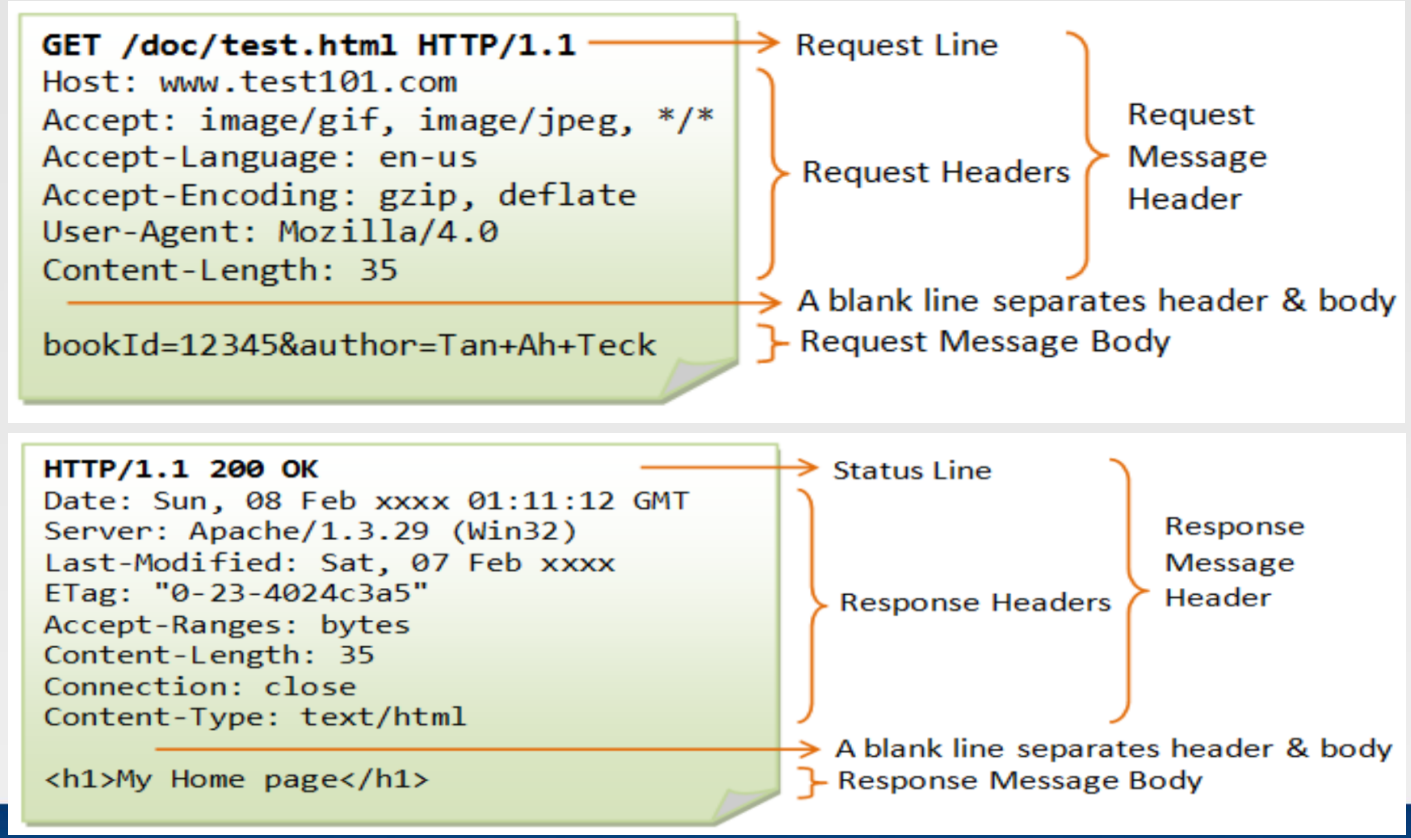
Introduction

- In this presentation, I will introduce you to HyperText Transfer Protocol (HTTP) response security headers.
- By specifying expected and allowable behaviors, we will see how security headers can prevent a number of attacks against websites.
- I'll explain some of the different HTTP response headers that a web server can include in a response, and what impact they can have on the security of the web browser.
- How web developers can implement these security headers to make user experience more secure

A Simple Look At Web Browsing



Snippet At The Request And Response Headers



Why Browser Security Headers?

Browser Security Headers help:

- to define whether a set of security precautions should be activated or deactivated on the web browser.
- to reinforce the security of your web browser to fend off attacks and to mitigate vulnerabilities.
- in fighting client side (browser) attacks such as clickjacking, injections, Multipurpose Internet Mail Extensions (MIME) sniffing, Cross-Site Scripting (XSS), etc.

Content / Context

HTTP STRICT TRANSPORT SECURITY (HSTS)

X-FRAME-OPTIONS

EXPECT-CT

CONTENT-SECURITY-POLICY

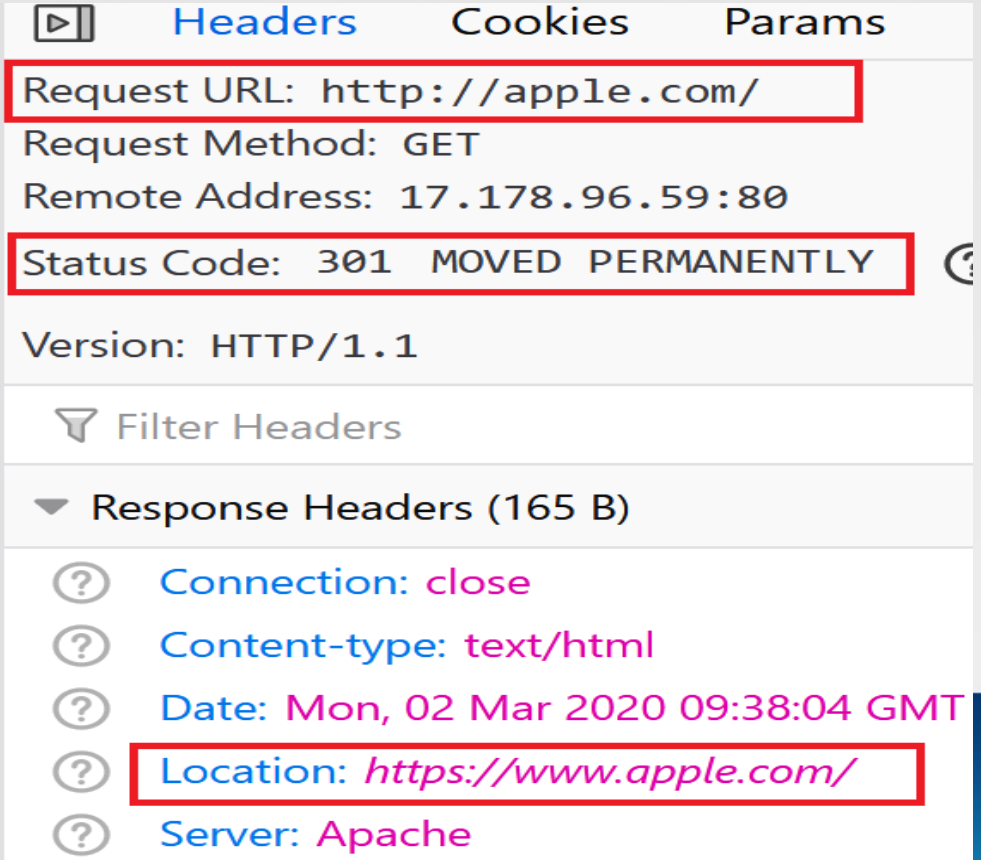
X-XSS-PROTECTION

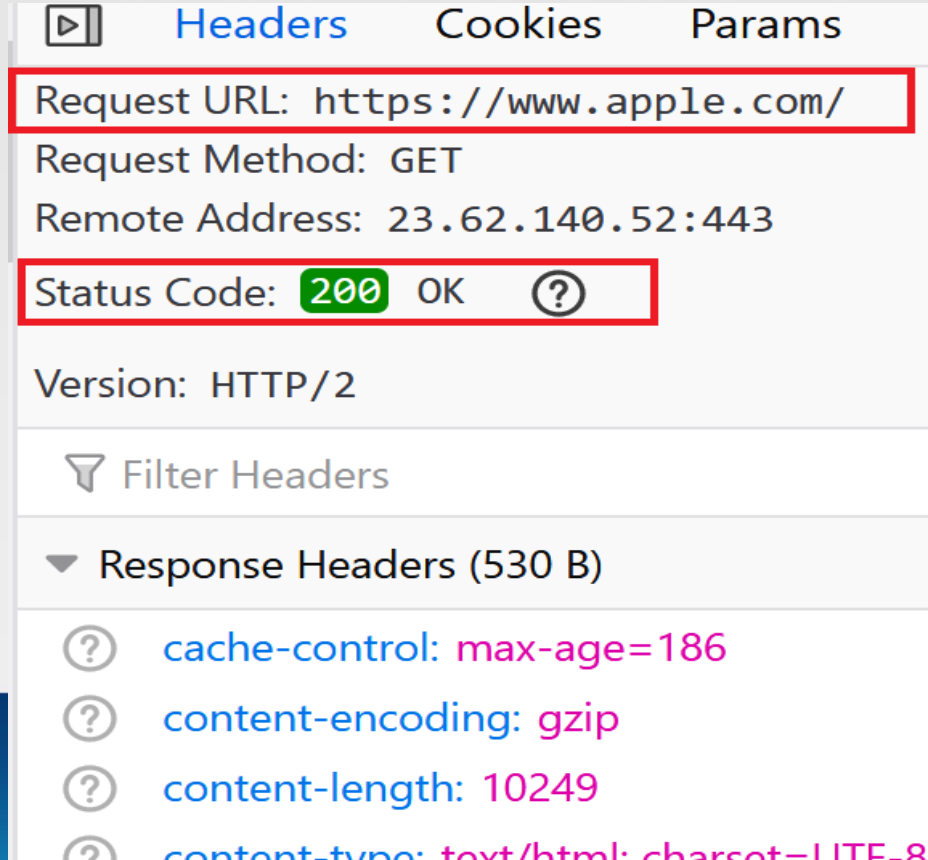
X-CONTENT-TYPE-OPTIONS

HTTP Strict Transport Security (HSTS)

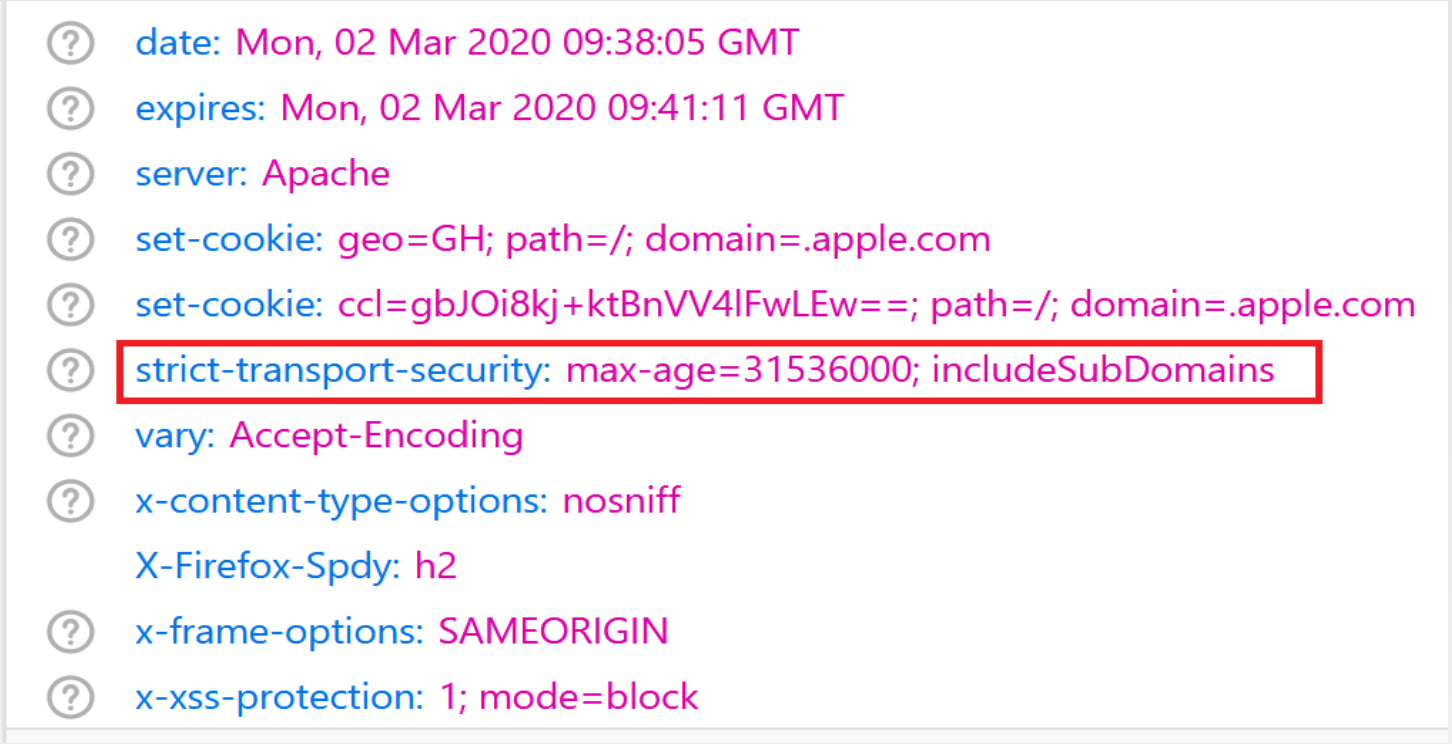
- HSTS header forces browsers to communicate using secure (HTTPS) connection.
- Protects against “downgrade attacks”
- When configured with the “Preload” option, it can prevent Man-In-The-Middle (MiTM) attack
- “Preload” - https://hstspreload.org/ - from google

HTTP Redirection To HTTPS





HTTP Redirection To HTTPS - Continued



HTTP Strict Transport Security (HSTS) - Implementation

Syntax:

Strict-Transport-Security: max-age=<expire-time>
includeSubDomains
preload

Apache:

Header set Strict-Transport-Security "max-age=31536000; includeSubDomains; preload“

Nginx:

add_header Strict-Transport-Security 'max-age=31536000; includeSubDomains; preload';

Microsoft IIS:

Name: Strict-Transport-Security

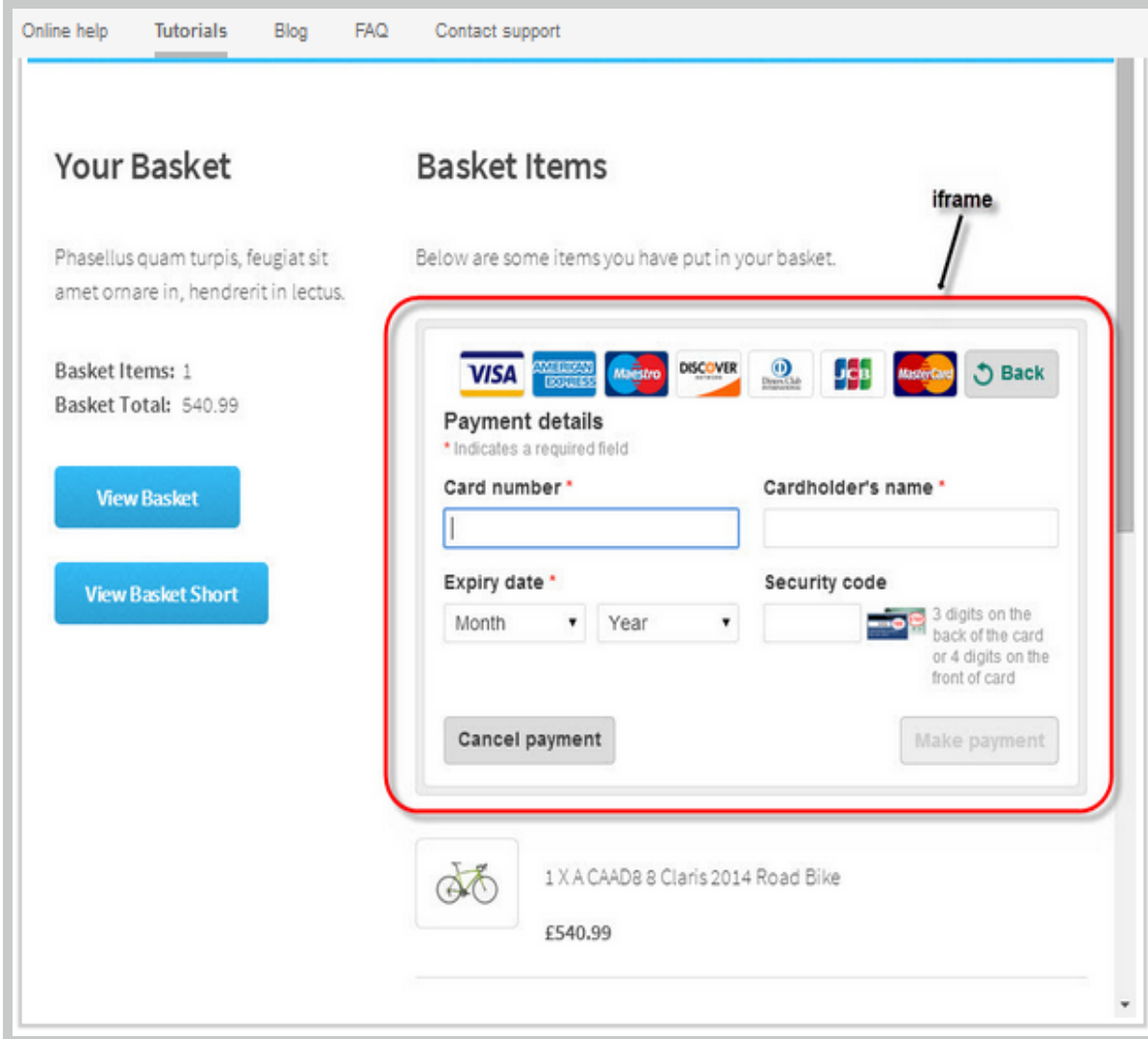
Value: max-age=31536000; includeSubDomains; preload

X-Frame-Options

- An iFrame is an element that allows a web app to be nested within a parent web app.
- Can be used maliciously for a clickjacking attack or loading a malicious website inside the frame

Prevention:

- Frame busting
- X-Frame-Option Header



X-Frame-Options - Implementation

Syntax:

X-Frame-Options: deny
sameorigin
allow-from url (deprecated)

Apache:

Header always set X-Frame-Options “deny”

Nginx:

add_header X-Frame-Options “DENY”;

WordPress:

header('X-Frame-Options: DENY);

Microsoft IIS:

Name: X-Frame-Options

Value: DENY

Expect-CT

- HTTP Public Key Pinning (HPKP) header is being deprecated to Expect-CT
- Expect-CT detects certificates issued by rogue Certificate Authorities (CA) or prevents them from doing so
- This header prevents MiTM attack against compromised Certificate Authority (CA) and rogue issued certificate

Expect-CT - Implementation

Syntax:

Expect-CT: max-age
enforce
report-uri

Apache:

Header set Expect-CT 'enforce, max-age=86400, report-uri="https://foo.example/report“’

Nginx:

add_header Expect-CT 'max-age=60, report-uri="https://mydomain.com/report"';

Content-Security-Policy (CSP)

This header helps you to whitelist sources of approved content into your browser hence, preventing the browser from loading malicious assets.

This helps prevents XSS, clickjacking, code injection, etc., attacks

When this header is well implemented, there is no need to implement “X-Frame-Options” and “*X-XSS-Protection”* headers

Content-Security-Policy - Directives

Keywords: *, none, self, hosts

Content-Security-Policy:

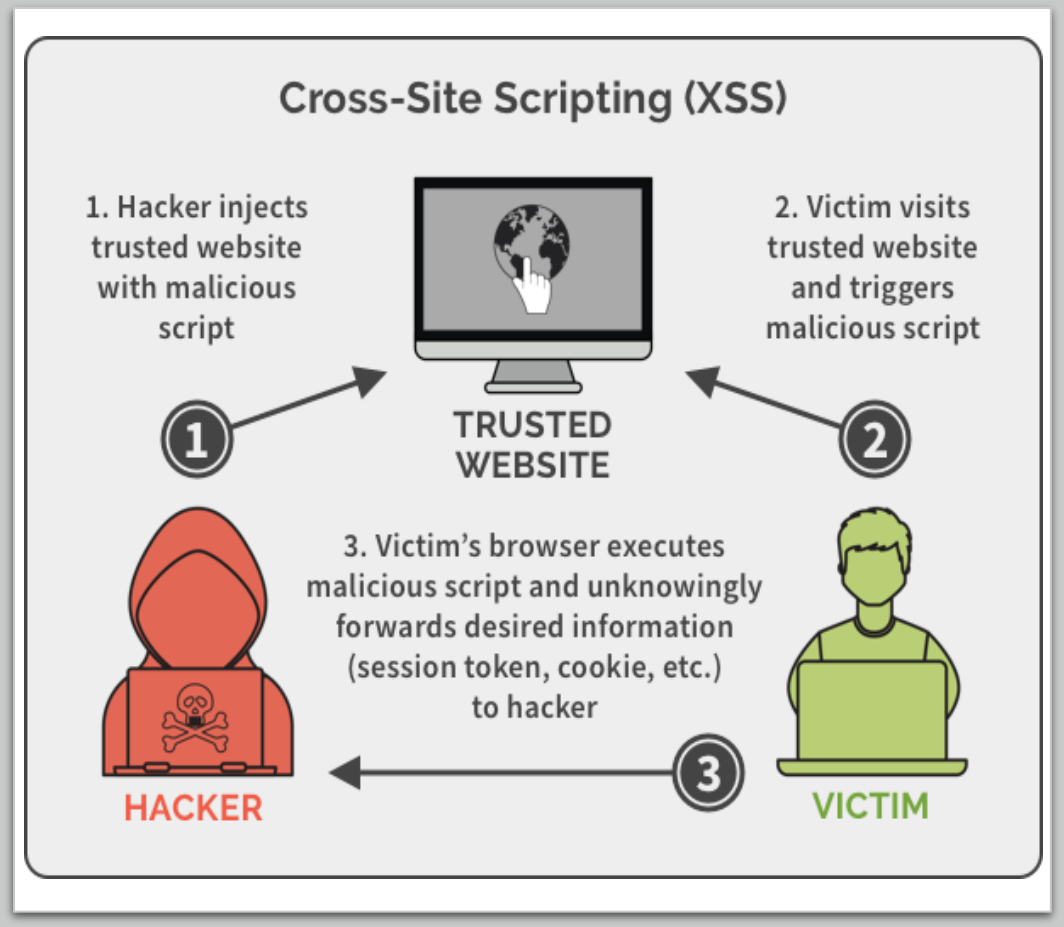
default-src	Serves as a fallback for the other fetch directives
font-src	Specifies valid sources for fonts loaded
frame-src	Sources for nested contexts such as <frame> and <iframe>
img-src	Sources of images and favicons
media-src	Valid sources for loading <audio>, <video> & <track>
object-src	Sources for the <object>, <embed> and <applet> elements
script-src	Specifies valid sources for JavaScript
style-src	Specifies valid sources for stylesheets
report-uri	Reports violations

CSP Sample - https://haveibeenpwned.com

content-security-policy: default-src 'none';script-src 'self' www.google-analytics.com www.google.com www.gstatic. js.stripe.com ajax.cloudflare.com;style-src 'self' 'unsafe-inline' cdnjs.cloudflare.com;img-src 'self' www.google-analytics.com stats.g.doubleclick.net www.gstatic.com;font-src 'self' cdnjs.cloudflare.com fonts.gstatic.com;base-uri 'self';child-src www.google.com js.stripe.com;frame-ancestors 'none';report-uri https://troyhunt.report-uri.com/r/d/csp/enforce.com/en_US/i/scr/pixel.gif;"

X-XSS-Protection

These header detect dangerous HTML input and either prevent the site from loading or remove potentially malicious scripts



X-XSS-Protection - Implementation

Syntax:

X-XSS-Protection: 0
1
mode=block

Apache:

Header set X-XSS-Protection "1; mode=block“

Nginx:

add_header X-XSS-Protection "1; mode=block";

Microsoft IIS:

Name: X-XSS-Protection

Value: 1; mode=block

X-Content-Type-Options

- For your seamless experience on the web, MIME sniffing of resource was introduced.
- Adversely, an attacker can introduce a malicious executable script such as an image. When acted on by MIME sniffing could have the script executed.

X-Content-Type-Options - Implementation

Syntax:

X-Content-Type-Options: nosniff

Apache:

Header set X-Content-Type-Options nosniff

Nginx:

add_header X-Content-Type-Options nosniff;

Microsoft IIS:

Name: X-Content-Type-Options

Value: nosniff

Demo Time

- Clickjacking
- iFrame injection
- Harlem shake

https://127.0.0.1/mutillidae/

Takeaways

- Enforce HTTPS using the Strict-Transport-Security header and add your domain to Chrome's preload list.
- Make your web app more robust against XSS by leveraging the X-XSS-Protection header.
- Block clickjacking using the X-Frame-Options header.
- Leverage Content-Security-Policy to whitelist specific sources and endpoints.
- Prevent MIME-sniffing attacks using the X-Content-Type-Options header.

Resources / Tools

- Check Website HTTP Response Header
  - https://gf.dev/http-headers-test
- Secure Headers Test
  - https://gf.dev/secure-headers-test
- Scott Helme – Security Header Scanner
  - https://securityheaders.com
- HTTP Headers Reference
  - https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers
- HTTP Compatibility Among Browsers
  - https://caniuse.com

References

- https://www.netsparker.com/whitepaper-http-security-headers
- https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html
- https://owasp.org/www-chapter-ghana/#div-pastevents
- https://www.keycdn.com/blog/http-security-headers

THANK YOU

Questions And Answers

Let’s Connect:

@egbordzor

linkedin.com/in/egbordzor

egbordzor@protonmail.com