The Art of Executing Javascript

About

- Akhil Mahendra
- Web application security enthusiast
- CTFer{@teambi0s}
- @Akhil_Mahendra

Agenda

- ➢ Introduction - XSS
- ➢ Types of XSS and different context
- ➢ Same Origin Policy
- ➢ Content Security Policy
- ➢ XSS via Angular JS

Introduction

Attack with a wrong name ?

Introduction

- ➢ Still exists after 18 years !
- ➢ NO.7 in OWASP top 10 2017
- ➢ Most commonly reported security vulnerability

Introduction - Impact

- Stealing user cookies
- Keylogger
- Deface website
- Redirect users

Types of XSS

- ➢ Reflected XSS
- ➢ Stored XSS
- ➢ DOM based XSS

Different Context

- HTML
- Attribute
- Script
- Style
- Url

Different Context -html context

- ➢ User input comes inside HTML elements
  - ○ `<p>Injection</p>`
- ➢ POC
  - ○ `<script>alert(1)</script>`

Different Context -attribute context

- User input comes inside HTML attributes
  - <p class = ” Injection ”> </p>
  - <p Injection = ” test123 ”> </p>
- POC
  - “ onload=alert(1)//
  - onload=alert(1)//

Different Context -script context

- User input comes inside <script> tags
  - <script> var a = ‘ Injection ‘; </script>
- POC
  - ‘;alert(1);//

Different Context -style context

- User input comes inside <script> tags
  - <p style " color: injection " > </p>
- POC
  - expression(alert(1));

Different Context -url context

- User input comes inside <script> tags
    - <a href = ” injection ” > click </a>
- POC
    - javascript:alert(1)

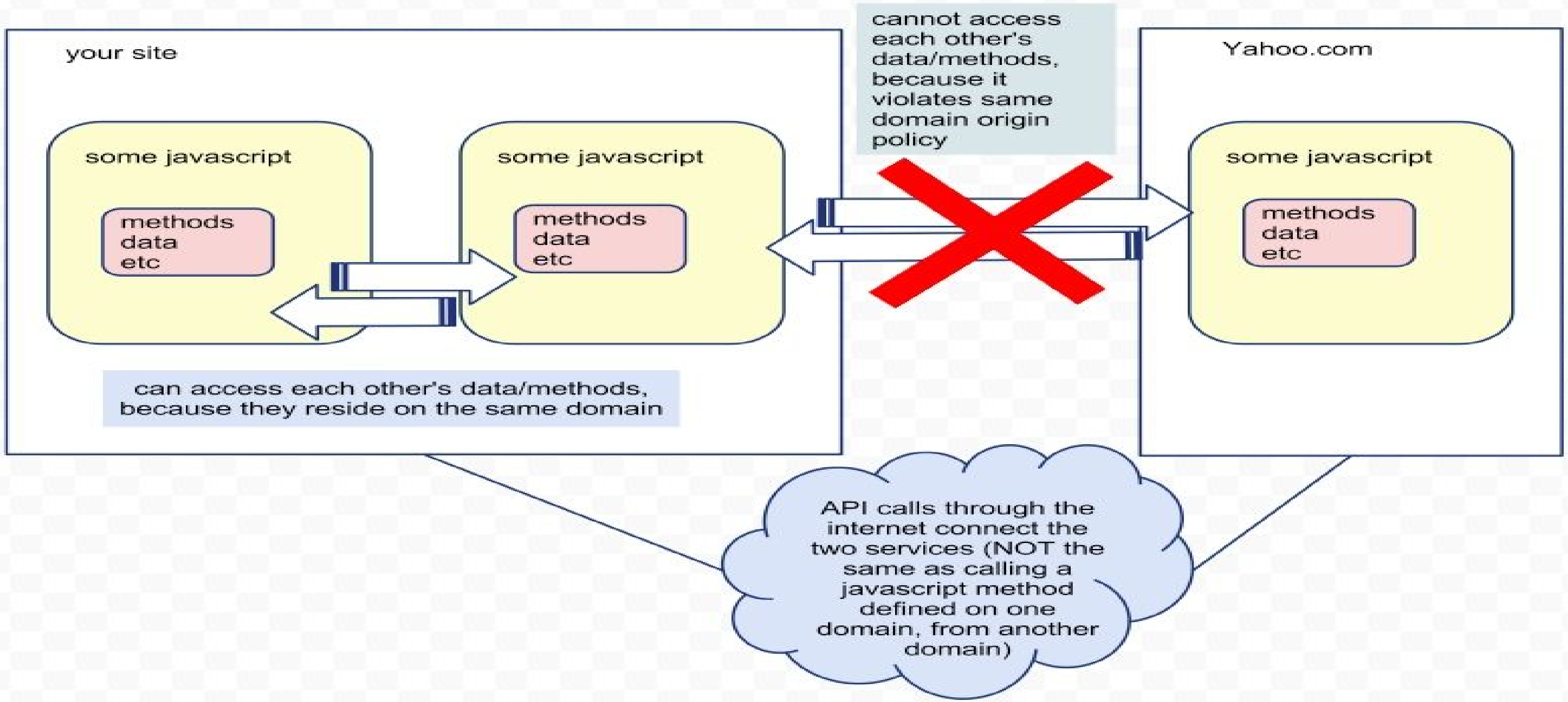
SOP

- Scripts on a page can make HTTP **request** and process **responses** between hosts that has the same:

Protocol, **Hostname**, **Port**

- An **IFRAME** loaded cannot **read** or **write** data into the page unless it’s in the same origin !

SOP


your site
some javascript
methods
data
etc
some javascript
methods
data
etc
can access each other's data/methods, because they reside on the same domain
cannot access each other's data/methods, because it violates same domain origin policy
Yahoo.com
some javascript
methods
data
etc
API calls through the internet connect the two services (NOT the same as calling a javascript method defined on one domain, from another domain)

CSP

- Introduced as a mechanism to mitigate code injection
- Directives defines:
  - From where and what content is allowed to load
  - In which context the content is allowed to execute
- It’s a mitigation not first line of defense!

CSP - Directives

- Directives:
  - default-src
  - script-src
  - object-src
  - style-src
  - image-src
  - frame-src

CSP - Keywords

- Keywords:
  - ‘*’
  - 'none'
  - 'self'
  - 'unsafe-inline'
  - 'unsafe-eval'

CSP

- HTTP Headers
  - <?php header('Content-Security-Policy: default-src https://cdn.example.net; child-src 'none'; object-src 'none'"');?>
- Meta tag in HTML
  - <meta http-equiv="Content-Security-Policy" content="default-src https://cdn.example.net; child-src 'none'; object-src 'none'">

CSP - Common mistakes

- unsafe-inline, unsafe-eval, data:
  - whole purpose of CSP is defeated
- Eg: default-src: ’self’;script-src: ‘unsafe-inline’
  - Bypass : <script>alert(1)</script>

CSP - Common mistakes

- Nonces:
  - Nonce must be a random string
  - Should not be reused
  - Should not be guessable

CSP - Common mistakes

- Examples of bad nonce
  - Request 1- D29162F1B99108DDA2406C697FFAC27586F42C7D021669F01F720CEEACBB06F5
  - Request 2- D29162F1B99108DDA2406C697FFAC27586F42C7D021669F01F720CEEACBB06F5
  - e10adc3949ba59abbe56e057f20f883e - md5(123456)
  - 1231441

Demo

CSP - bypass

CSP Bypass

XSS via Angular JS

Escaping the expression sandbox for XSS

Thanks

@Akhil_Mahendra