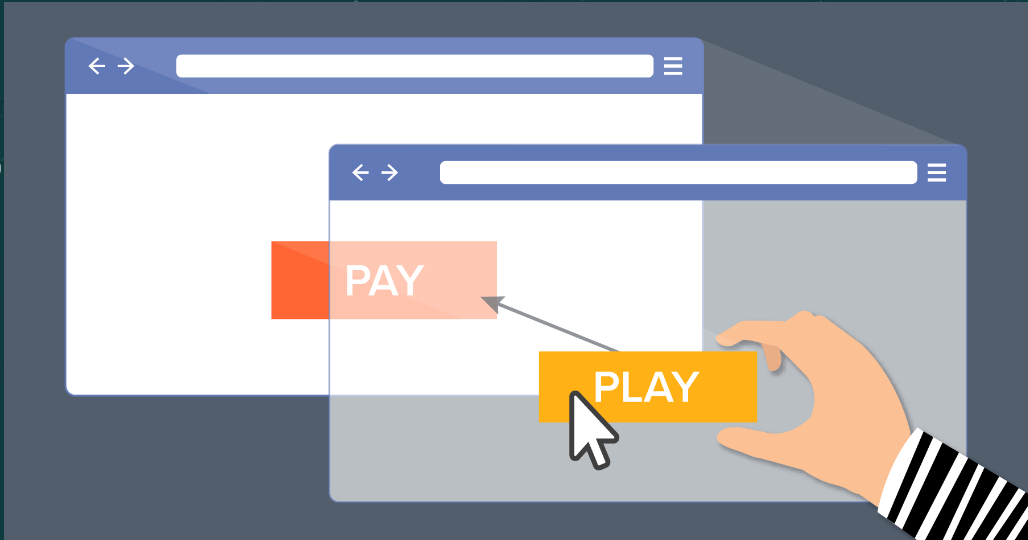
Web Application Security - Clickjacking

Information Security – Lecture 13

Aadil Zia Khan

Clickjacking

- Clickjacking / User Interface redress attack / UI redress attack / UI redressing
  - Malicious technique of tricking a user into clicking on something different from what the user believes he is clicking
  - Could reveal confidential information or allow others to take control of their computer



Clickjacking - Types

- Classic clickjacking
    - Attacker uses hidden layers on web pages to manipulate the actions a user's cursor does, resulting in misleading the user about what truly is being clicked on

- Example
    - Assume your browser settings are such that you stay signed-in to your Amazon account
    - A malicious attacker can render Amazon’s single click purchase webpage in an iframe (set its opacity to zero so that it becomes invisible) inside his own webpage, and place his own button (with message like “click to play”) on the exact location of Amazon’s button so that they overlap
    - If the victim clicks on the button, in reality he would be clicking on the purchase button

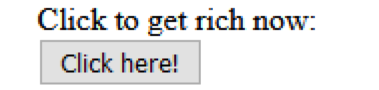
Clickjacking - Types

Contains a single click purchase button – hidden due to opacity=0, and under the overlapping button due to z-index=1

```
<style> iframe {
            width: 400px;
            height: 100px;
            position: absolute;
            top:0; left:-20px;
            opacity: 0;
            z-index: 1;
}
</style>
<div>Click to get rich now:</div>
<!-- The url from the victim site -->
<iframe src="amazon.com/xyz.htm"></iframe>
<button>Click here!</button>
```

Easy to clickjack mouse clicks and mobile screen taps

Difficult with Keyboard input - everything that the visitor types will be hidden, because the iframe is not visible


Click to get rich now:

Clickjacking - Types

- Likejacking
  - Clickjacking which aims to trick users viewing a website into "liking" a Facebook page or other social media posts/accounts that they did not intentionally mean to "like"

Clickjacking - Types

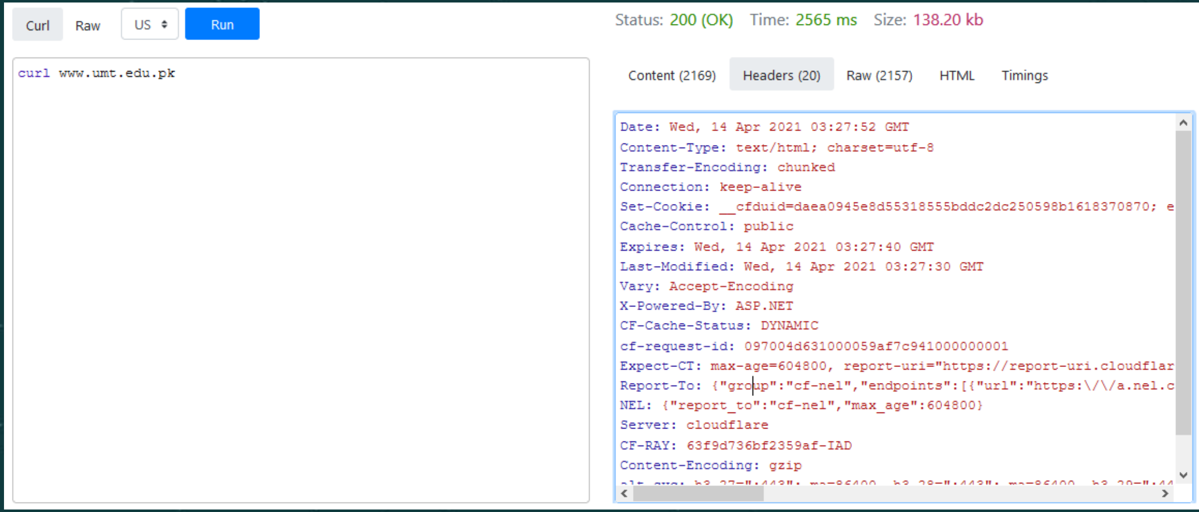
- Cookiejacking / Filejacking
  - User is tricked into granting access to computer files or cookies

Clickjacking Prevention using X-Frame-Options

- Web server (e.g., Amazon's) can respond to a browser's request with an X-Frame-Options HTTP header with three possible values to control how it's pages can be rendered/embeded inside webpages belonging to other domains
  - DENY - the website can never be embeded in an iframe, frame, or HTML object (highest level of protection)
  - SAMEORIGIN - the webpage can only be rendered if it is embedded inside an iframe, frame, or HTML object on a page belonging to the same domain (allows developers of a particular website to embed their own pages anywhere in their own applications/sites)
  - ALLOW-FROM <URI> - the webpage can be embedded inside pages belonging to the provided URI (allows a third party to embed your content through an iframe)
    - Note that you can only specify one URI

cURL-ing UMT

- curl www.umt.edu.pk
  - X-Frame-Options missing
  - UMT's webpage can be embedded inside other websites

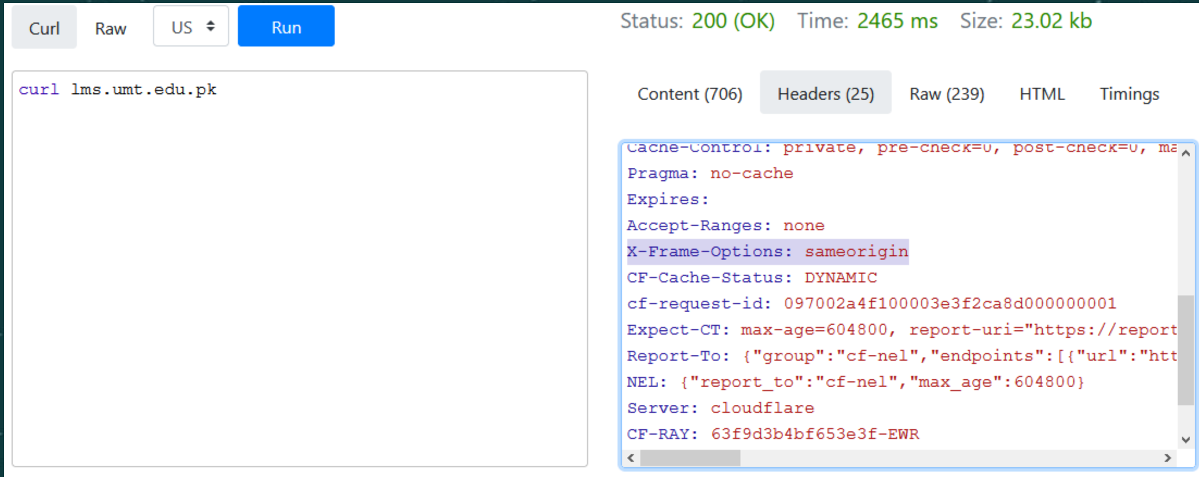


```
curl www.umt.edu.pk
```

Status: 200 (OK) Time: 2565 ms Size: 138.20 kb

```
Date: Wed, 14 Apr 2021 03:27:52 GMT
Content-Type: text/html; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive
Set-Cookie: __cfduid=daea0945e8d55318555bddc2dc250598b1618370870; e
Cache-Control: public
Expires: Wed, 14 Apr 2021 03:27:40 GMT
Last-Modified: Wed, 14 Apr 2021 03:27:30 GMT
Vary: Accept-Encoding
X-Powered-By: ASP.NET
CF-Cache-Status: DYNAMIC
cf-request-id: 097004d631000059af7c941000000001
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflar
Report-To: {"group":"cf-nel","endpoints":[{"url":"https:\/\/a.nel.c
NEL: {"report_to":"cf-nel","max_age":604800}
Server: cloudflare
CF-RAY: 63f9d736bf2359af-IAD
Content-Encoding: gzip
alt-svc: h3-27=":443"; ma=86400, h3-28=":443"; ma=86400, h3-29=":44
```

- curl lms.umt.edu.pk
  - X-Frame-Options set to sameorigin
  - LMS webpage can not be embedded inside other websites – but can be embedded inside UMT's websites



```
curl lms.umt.edu.pk
```

Status: 200 (OK) Time: 2465 ms Size: 23.02 kb

```
Cache-Control: private, pre-check=0, post-check=0, ma
Pragma: no-cache
Expires:
Accept-Ranges: none
X-Frame-Options: sameorigin
CF-Cache-Status: DYNAMIC
cf-request-id: 097002a4f100003e3f2ca8d000000001
Expect-CT: max-age=604800, report-uri="https://report
Report-To: {"group":"cf-nel","endpoints":[{"url":"htt
NEL: {"report_to":"cf-nel","max_age":604800}
Server: cloudflare
CF-RAY: 63f9d3b4bf653e3f-EWR
```

Problem with Web Proxies

- Web proxies/caches are often used to save webpages so that they may be accessed quickly and also to reduce the access link usage

- They perform header manipulation before saving the page and in doing so may strip off security related headers from the response

Clickjacking not Prevented by X-Frame-Options

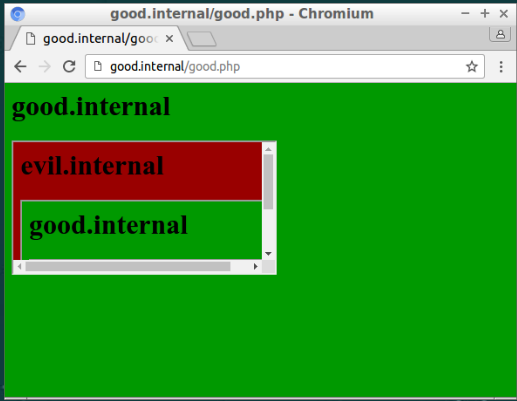
- Double Clickjacking
  - Open target page (e.g., Amazon) as a popup window behind the current browser window
  - Trick the user into double clicking on the current browser window
  - First click hits the current browser window and shifts focus to the popup window
  - Second click hits the button/link on the popup window, triggering some action (e.g., purchase)
- This was used to attack Google's OAuth authentication popup

Clickjacking not Prevented by X-Frame-Options

- Clickjacking via History Navigation
  - Open target page (e.g., Amazon) as a popup window
    - Browser will cache it
  - Immediately navigate the popup to a page belonging to the attacker
  - Trick the user into clicking on the popup page (e.g. make a shooting game)
  - Just before the user clicks, call the history.back() function
    - The target page (e.g., Amazon) would open in the popup just before the user clicks
    - Close the popup after the click

Clickjacking not Prevented by X-Frame-Options

- Nested Clickjacking
  - Suppose attacker embeds Google’s page inside his own webpage
  - When I load his webpage in my browser, the frame on his webpage tries to load Google’s page - the X-Frame-Options:SAMEORIGIN would not allow the Google page to load in my browser
  - NOTE: when checking for X-Frame-Options:SAMEORIGIN, the browser only checks the top level origin – it makes the following attack possible:
    - Attacker embeds Google’s page inside his own webpage
    - Suppose attacker’s webpage shows up in a Google search
    - Now the top level origin is Google
    - Browser will only check the top level origin, since it is Google, the frame inside attacker’s webpage will be able to load Google’s page


good.internal/good.php - Chromium
good.internal/goo
good.internal/good.php
good.internal
evil.internal
good.internal