Clickjacking

What is Clickjacking?

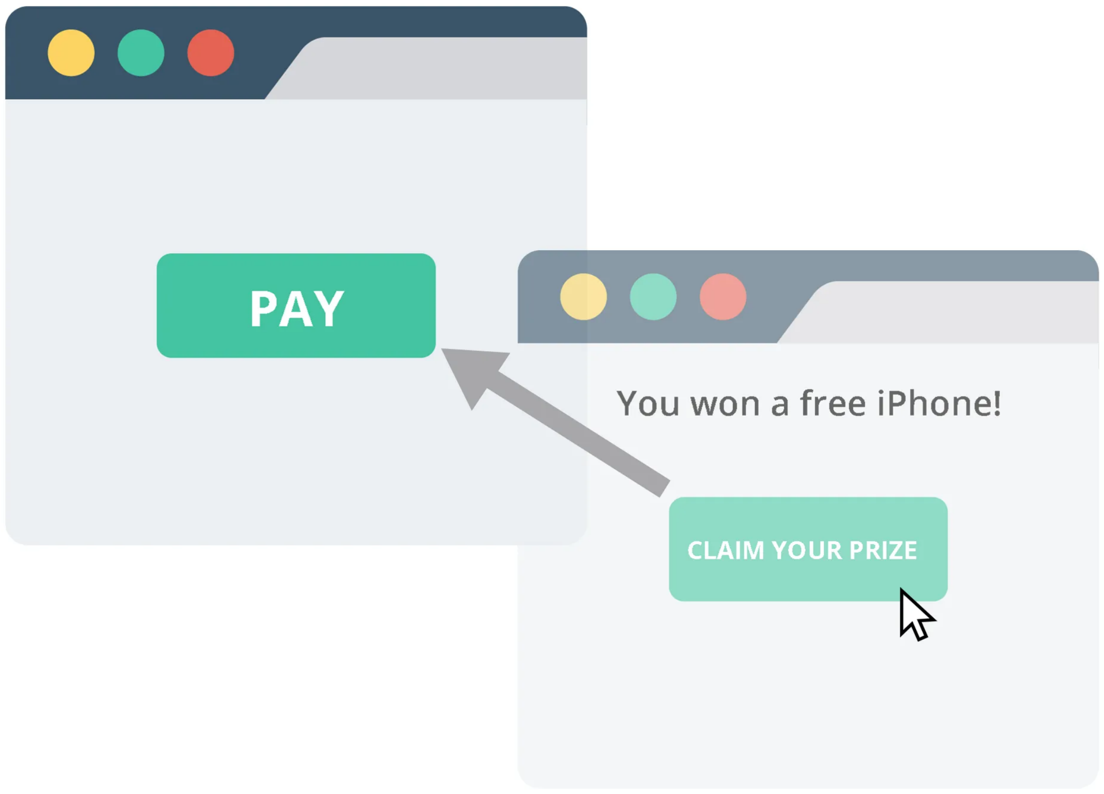
Clickjacking, also known as a “UI redress attack” is an interface-based attack in which a user is tricked into clicking on actionable content on a hidden website by clicking on some other content in a decoy website. In simple words, an attacker uses multiple transparent or opaque layers to trick a user 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.

Using a similar technique, keystrokes can also be hijacked. With a carefully crafted combination of stylesheets, iframes, and text boxes, a user can be led to believe they are typing in the password to their email or bank account, but are instead typing into an invisible frame controlled by the attacker.

How does Clickjacking work?

Clickjacking works by using an iframe to load a vulnerable website on top of an attacker's controlled domain.

Let's take an example



Over here, you can see there are two web pages let us call them A and B.

Website A : A website vulnerable to clickjacking which says that "You won a free iPhone!" so that the victim gets lured by "Claiming the Prize!"

Website B: It is an attacker controlled website wherein you have the option to "Pay"

This is the exact scenario of how Clickjacking works. Attackers create an attractive website, in our case website A, to lure the victim. Create an iframe and load it in the attacker controlled domain which is website B.

In this manner the attacker manages to fool the victim and make him pay while attracting him for a free prize!

So this is how Clickjacking attack is performed.

Severity

Clickjacking based vulnerabilities are one of the simple bugs to find and are classified into two types:

- Clickjacking on Non-Sensitive Pages
- Clickjacking on Sensitive Pages

Clickjacking on Non-Sensitive Pages are generally considered as Informational and categorized as P5 vulnerability where-as Clickjacking on Sensitive Pages are categorized as P4.

Clickjacking on sensitive pages can also increase the impact of account takeover and hence can sometimes go up to P3 category.

How to demonstrate Clickjacking vulnerabilities

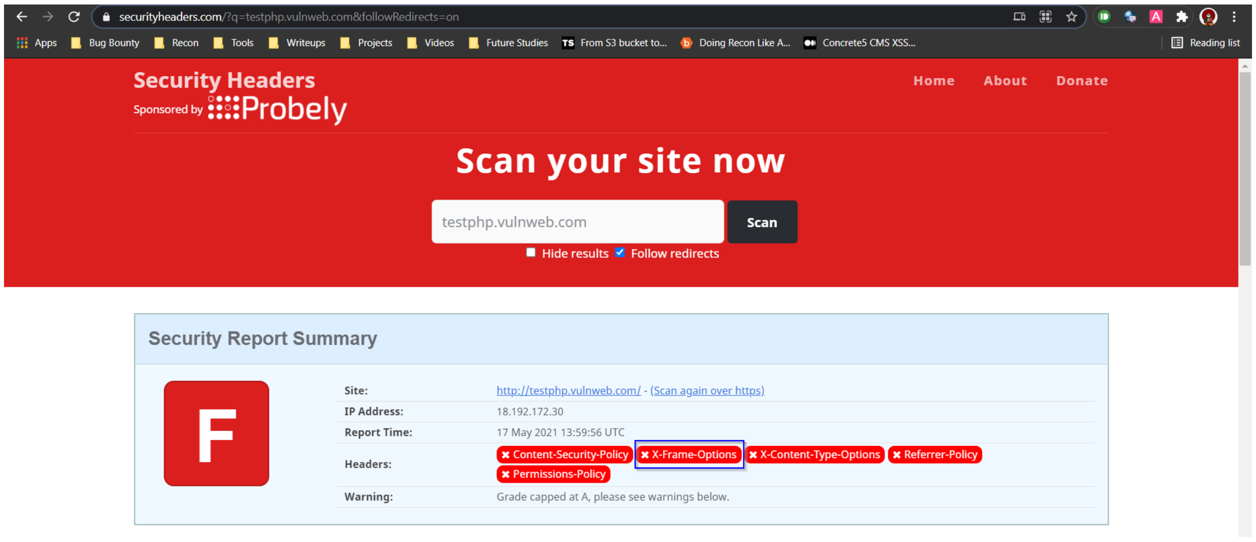
To demonstrate Clickjacking vulnerabilities, we first need to check for `X-Frame-Options`

`X-Frame-Options` tells the browser whether you want to allow your site to be framed or not. By preventing a browser from framing your site you can defend against attacks like clickjacking. Recommended value "`X-Frame-Options: SAMEORIGIN`".

To check if a website has `X-Frame-Options` set or not simply go to https://securityheaders.com/

Over here simply type-in the website you want to check for and hit `Scan`

Below is an example of a website vulnerable to clickjacking


securityheaders.com/?q=testphp.vulnweb.com&followRedirects=on

Security Headers
Sponsored by Probely

Home About Donate

Scan your site now

testphp.vulnweb.com Scan

Hide results [✓ Follow redirects]

Security Report Summary

F

Site:	http://testphp.vulnweb.com/ - (Scan again over https)
IP Address:	18.192.172.30
Report Time:	17 May 2021 13:59:56 UTC
Headers:	✖ Content-Security-Policy ✖ X-Frame-Options ✖ X-Content-Type-Options ✖ Referrer-Policy ✖ Permissions-Policy
Warning:	Grade capped at A, please see warnings below.

Clickjacking on non-sensitive pages

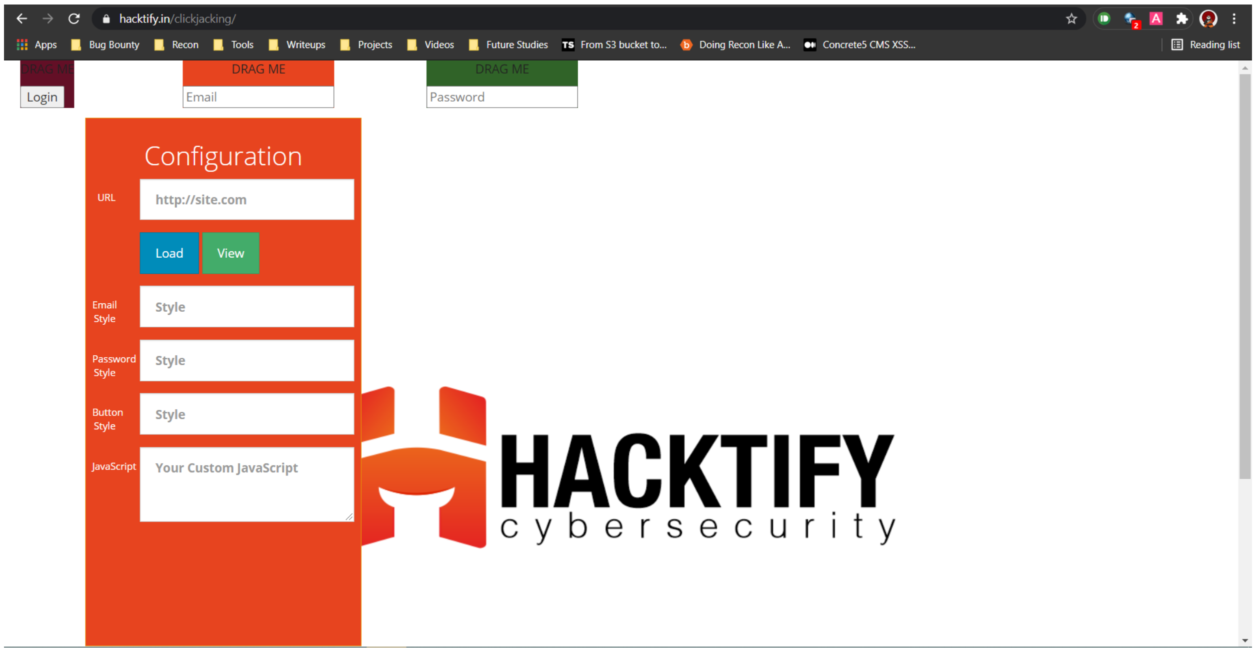
To demonstrate clickjacking on non-sensitive pages replace `{URL}` with the URL of the vulnerable target in the below HTML file, save and run in the web browser. Take a snapshot as your proof of concept (POC)

```
<html>
<head>
<title>Clickjack test page</title>
</head>
<body>
<p>Website is vulnerable to clickjacking!</p>
<iframe src="{URL}" width="500" height="500"></iframe>
</body>
</html>
```

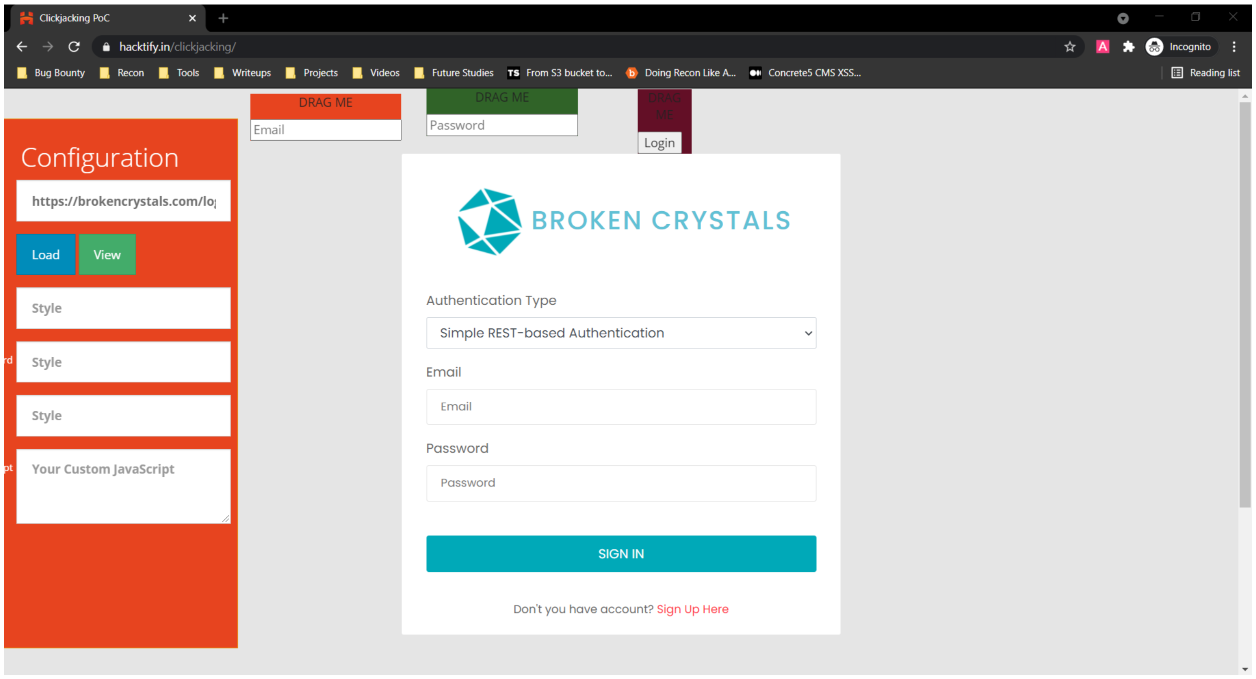
Clickjacking on sensitive pages

To demonstrate clickjacking on sensitive pages, we will be using a tool developed by Hacktify Cyber Security. The tool can be found at https://hacktify.in/clickjacking/

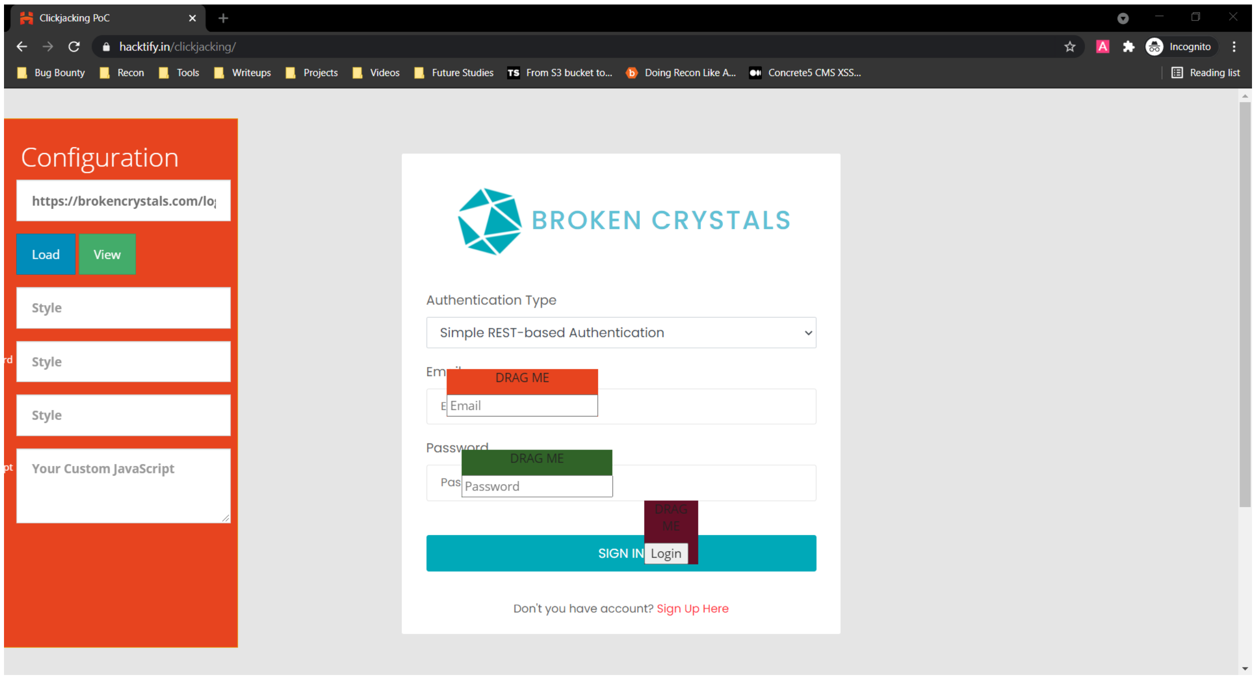
The website looks like this:



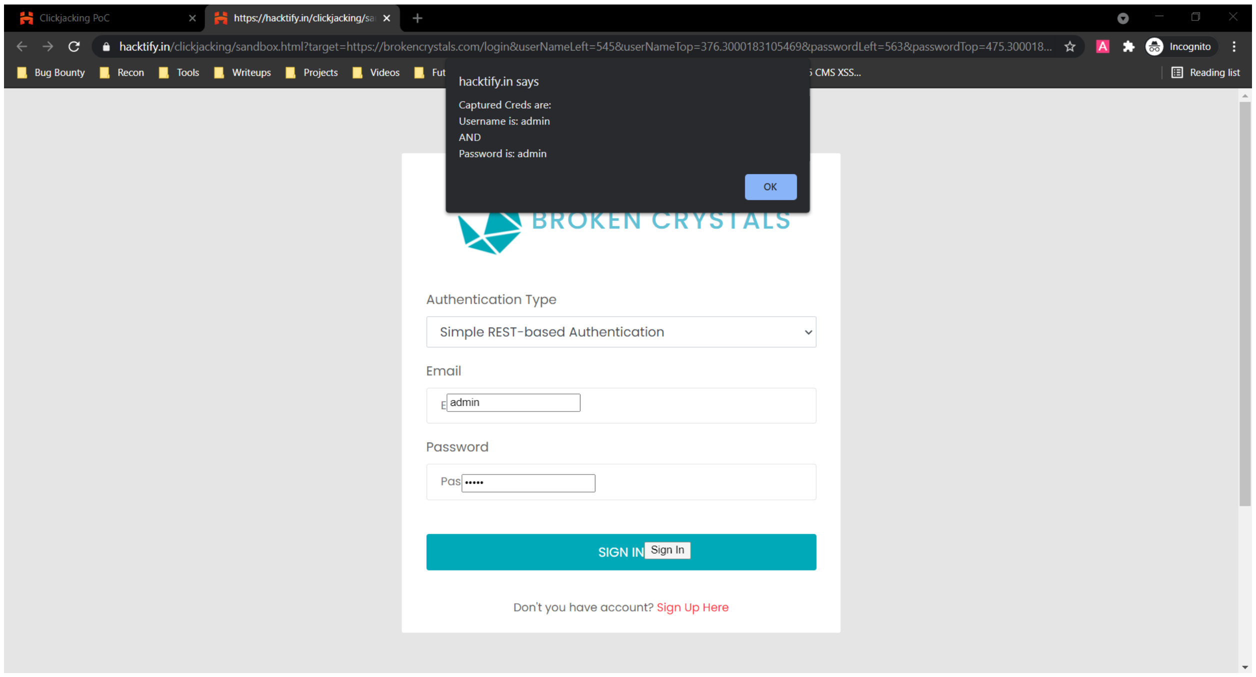
First in the `{URL}` field you have to add your URL and click on `Load`. For demonstration I will be using the vulnerable website: https://testphp.vulnweb.com



Once the webpage is loaded, drag the `Email` `Passowrd` and `Login` to match with the respective fields of the vulnerable website



Click on `View` . A new website must have opened. Enter all the details in the new input fields and click on Sign In Button



```
hacktify.in/clickjacking/sandbox.html?target=https://brokencrystals.com/login&userNameLeft=545&userNameTop=376.3000183105469&passwordLeft=563&passwordTop=475.300018...
```

hacktify.in says

Captured Creds are:
Username is: admin
AND
Password is: admin

Notice an alert box pops up which says credentials captured which leads to account takeover via Clickjacking.

An attacker can simply change the script to forward these credentials to his controlled website.

Impact of Clickjacking

Attackers may abuse clickjacking vulnerabilities for many different purposes:

- To gain followers on social media and then, possibly, sell the social media account/page for mass marketing.
- To gain email or RSS subscribers for the same purpose as social media followers.
- To use the fact that the user is logged into their e-commerce account and have them buy products on behalf of the attacker.
- To have the user unknowingly transfer funds to the attacker.
- To have the user download malware (e.g. a trojan).

In general, clickjacking uses depend only on the attacker’s imagination and on finding a vulnerable tool page to use for that purpose.

Clickjacking Prevention

Content Security Policy (CSP)

Content Security Policy (CSP) is a detection and prevention mechanism that provides mitigation against attacks such as XSS and clickjacking. CSP is usually implemented in the web server as a return header of the form:

`Content-Security-Policy: policy`

where policy is a string of policy directives separated by semicolons. The CSP provides the client browser with information about permitted sources of web resources that the browser can apply to the detection and interception of malicious behaviors.

The recommended clickjacking protection is to incorporate the `frame-ancestors` directive in the application's Content Security Policy. The `frame-ancestors 'none'` directive is similar in behavior to the X-Frame-Options `deny` directive. The `frame-ancestors 'self'` directive is broadly equivalent to the X-Frame-Options `sameorigin` directive.

X-Frame-Options Header Types

There are three possible values for the X-Frame-Options header:

- `DENY`, which prevents any domain from framing the content. The `DENY` setting is recommended unless a specific need has been identified for framing.
- `SAMEORIGIN`, which only allows the current site to frame the content.
- `ALLOW-FROM` **uri**, which permits the specified 'uri' to frame this page. (e.g., `ALLOW-FROM http://www.example.com`).

References

- Clickjacking by PortSwigger : https://portswigger.net/web-security/clickjacking
- OWASP Clickjacking: https://owasp.org/www-community/attacks/Clickjacking