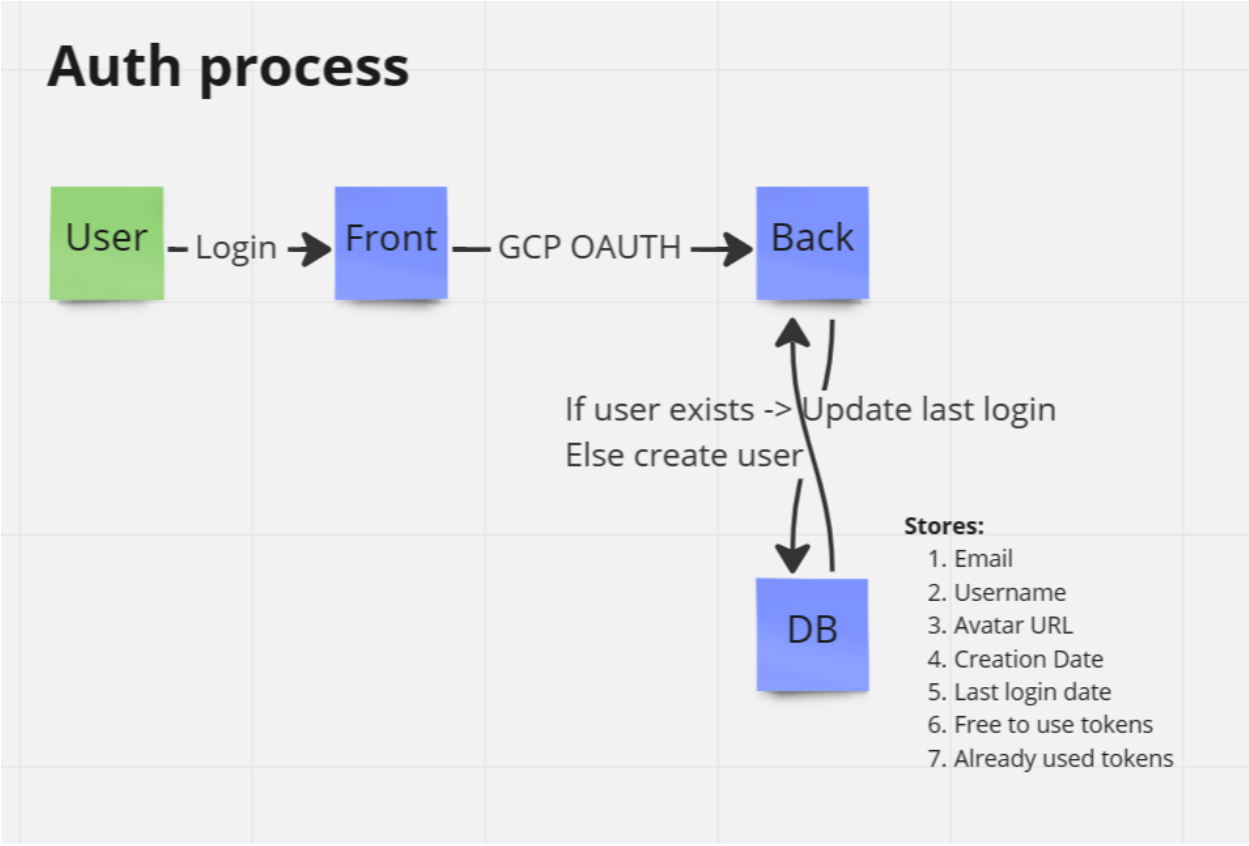
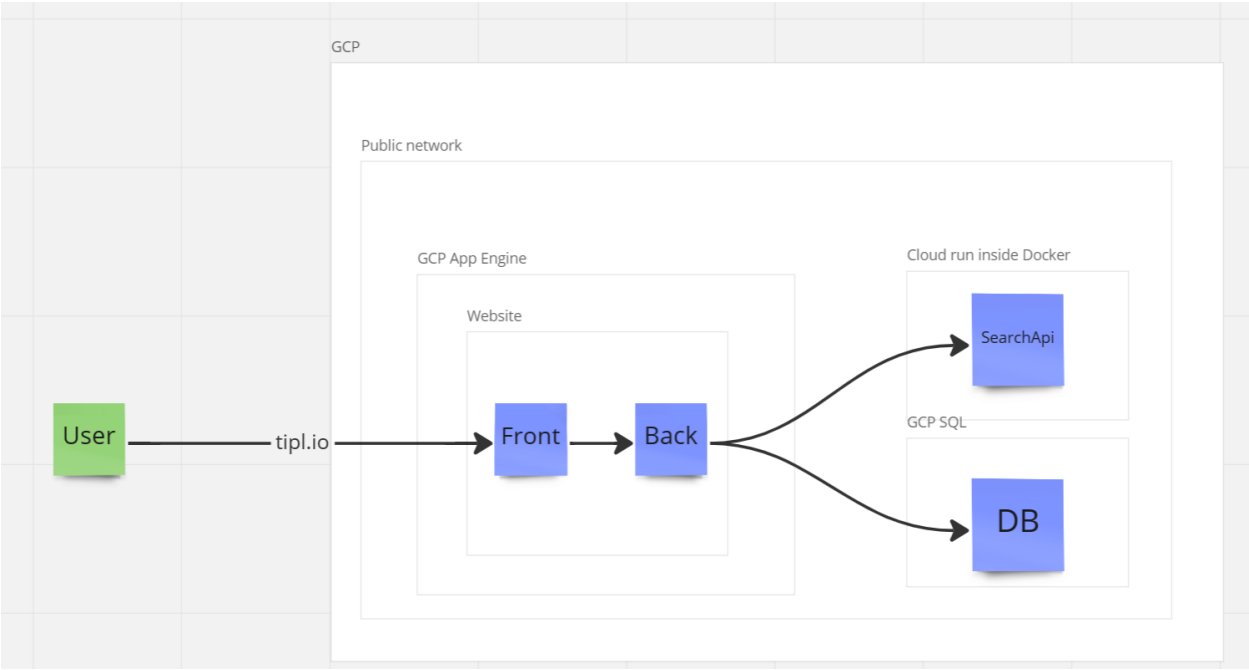
TIPL Security Assessment

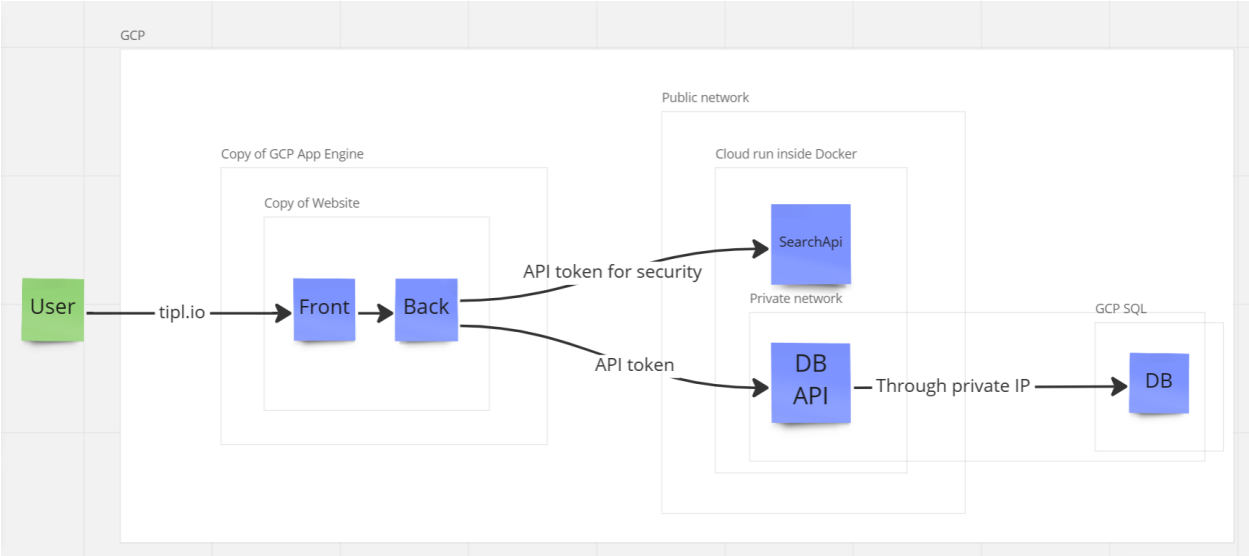
What is TIPL?

TIPL is a search engine powered by the powerful GPT AI. Its main feature is its ability to search a stock and provide the user with the most relevant information available, giving the user a deeper insight into the stock. It also lets the user know whether they should invest or not.

How does it work?




GCP
Public network
GCP App Engine
Website
Cloud run inside Docker
SearchApi
GCP SQL
DB
User
tipl.io
Front
Back


GCP
Public network
Copy of GCP App Engine
Copy of Website
Cloud run inside Docker
SearchApi
User
tipl.io
Front
Back
API token for security
Private network
GCP SQL
API token
DB
API
Through private IP
DB

Overview

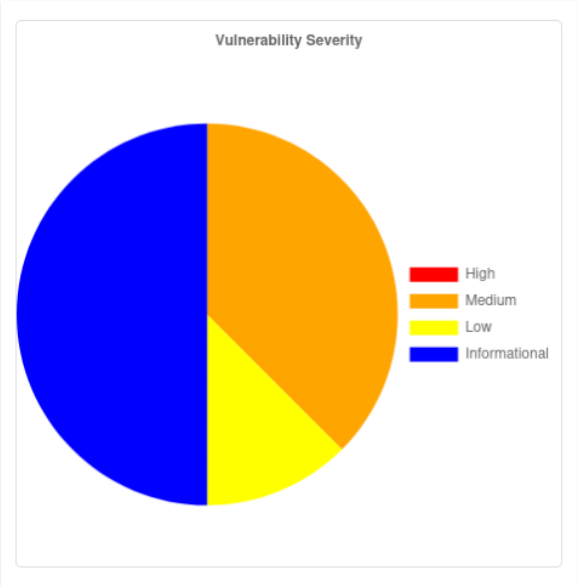
For my Capstone project, I did a complete security assessment for TIPL. The assessment is divided in the following steps:

1. Vulnerability Scan: I utilized a wide variety of tools to scan TIPL for any possible security flaws.
2. Penetration Testing: After running multiple scans, i attempted to exploit any vulnerabilities I could find
3. Secure Code Review: I went through all our lines of code and tried to identify any security flaws.
4. Google Cloud Platform: This is one of our most important assets, since everything is being hosted here. Any breaches to GCP could be fatal to our company.

Vulnerability Scan

OWASP ZAP

		Confidence				
		User Confirmed	High	Medium	Low	Total
Risk	High	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
	Medium	0 (0.0%)	2 (25.0%)	1 (12.5%)	0 (0.0%)	3 (37.5%)
	Low	0 (0.0%)	0 (0.0%)	1 (12.5%)	0 (0.0%)	1 (12.5%)
	Informational	0 (0.0%)	0 (0.0%)	3 (37.5%)	1 (12.5%)	4 (50.0%)
	Total	0 (0.0%)	2 (25.0%)	5 (62.5%)	1 (12.5%)	8 (100%)



MEDIUM LEVEL

Content Security Policy Header not set

1. Risk: Medium
2. Confidence: High
3. Description: Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.
4. Solution: In order to fix this vulnerability, I configured the web server to return the Contest Security Policy HTTP header.
5. Reference:
    a. https://developer.mozilla.org/en-US/docs/Web/Security/CSP/Introducing_Content_Security_Policy
    b. https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html
    c. http://www.w3.org/TR/CSP/
    d. http://w3c.github.io/webappsec/specs/content-security-policy/csp-specification.dev.html
    e. http://www.html5rocks.com/en/tutorials/security/content-security-policy/
    f. http://caniuse.com/#feat=contentsecuritypolicy
    g. http://content-security-policy.com/

Missing Anti-clickjacking Header

1. Risk Medium
2. Confidence Medium
3. Parameter: X Frame Options
4. Description: The response does not include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options to protect against 'ClickJacking' attacks.
5. Solution: This issue was resolved by setting the CSP header in the previous vulnerability.
6. Reference:
    a. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options

X-Content-Type-Options Header Missing

1. Risk: Low
2. Confidence: Medium
3. Description: The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform

MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.

4. Solution: I ensured that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages.
5. References:
    a. http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx
    b. https://owasp.org/www-community/Security_Headers

NMAP SCAN

IP:172.67.214.143

```
PORT      STATE SERVICE
53/tcp    open  domain
80/tcp    open  http
443/tcp   open  https
8080/tcp  open  http-proxy
```

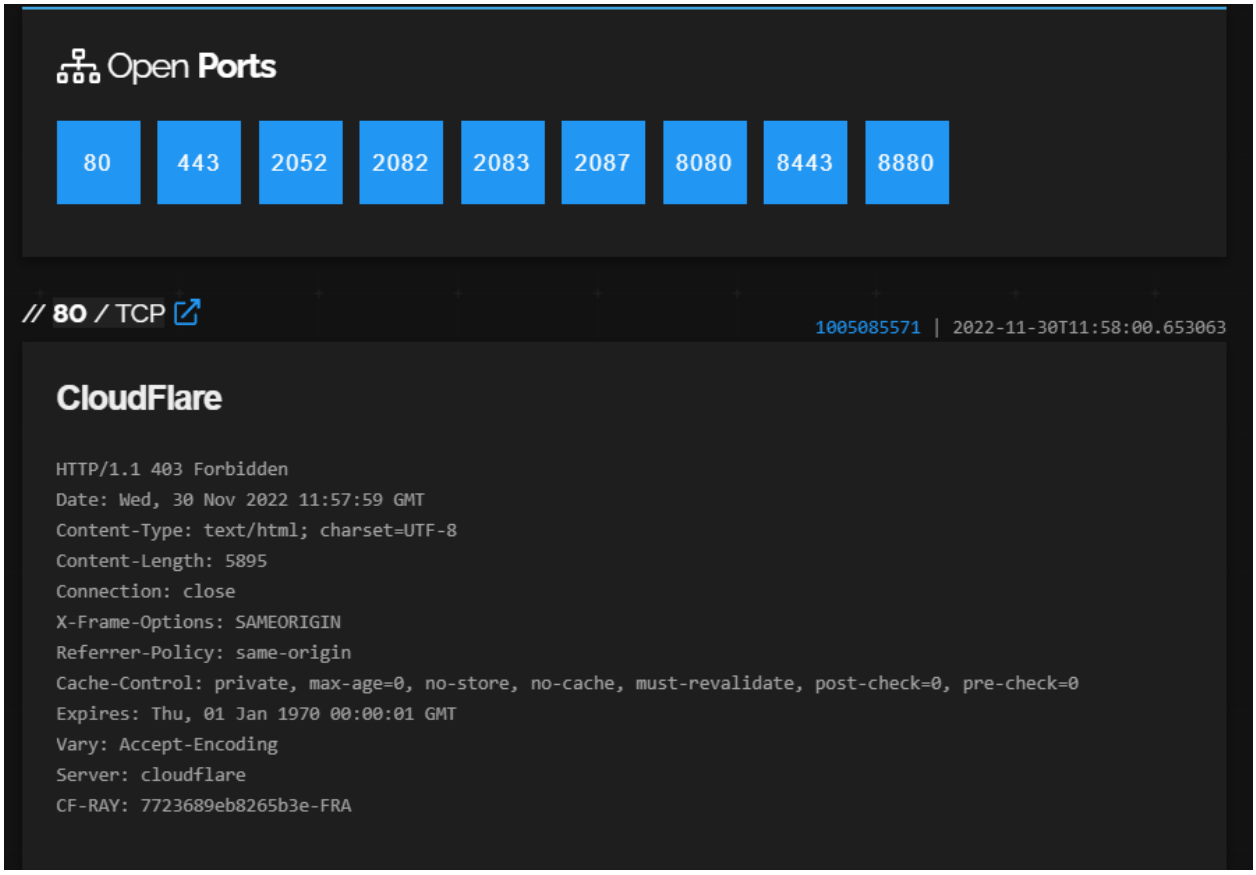
```
[*] Nmap: Completed Connect Scan at 21:46, 684.92s elapsed (1000 total ports)
[*] Nmap: Initiating Service scan at 21:46
[*] Nmap: Scanning 4 services on 172.67.214.143
[*] Nmap: Completed Service scan at 21:46, 17.18s elapsed (4 services on 1 host)
[*] Nmap: NSE: Script scanning 172.67.214.143.
[*] Nmap: Initiating NSE at 21:46
[*] Nmap: Completed NSE at 21:46, 5.13s elapsed
[*] Nmap: Initiating NSE at 21:46
[*] Nmap: Completed NSE at 21:46, 0.86s elapsed
[*] Nmap: Initiating NSE at 21:46
[*] Nmap: Completed NSE at 21:46, 0.00s elapsed
[*] Nmap: Nmap scan report for 172.67.214.143
[*] Nmap: Host is up (0.085s latency).
[*] Nmap: Not shown: 996 filtered tcp ports (no-response)
[*] Nmap: PORT     STATE SERVICE   VERSION
[*] Nmap: 53/tcp   open  domain    (generic dns response: NOTIMP)
[*] Nmap: 80/tcp   open  http      Cloudflare http proxy
[*] Nmap: |_http-title: Site doesn't have a title (text/plain; charset=UTF-8).
[*] Nmap: |_http-server-header: cloudflare
[*] Nmap: 443/tcp  open  ssl/https cloudflare
[*] Nmap: |_http-title: 400 The plain HTTP request was sent to HTTPS port
[*] Nmap: |_http-server-header: cloudflare
[*] Nmap: 8080/tcp open  http      Cloudflare http proxy
[*] Nmap: |_http-title: Site doesn't have a title (text/plain; charset=UTF-8).
[*] Nmap: |_http-server-header: cloudflare
[*] Nmap: 1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
[*] Nmap: SF-Port53-TCP:V=7.93%I=7%D=11/30%Time=63881576%P=x86_64-pc-linux-gnu%r(DNS
[*] Nmap: SF:StatusRequestTCP,E,"\0\x0c\0\0\x90\x84\0\0\0\0\0\0\0\0");
[*] Nmap: NSE: Script Post-scanning.
[*] Nmap: Initiating NSE at 21:46
[*] Nmap: Completed NSE at 21:46, 0.00s elapsed
[*] Nmap: Initiating NSE at 21:46
[*] Nmap: Completed NSE at 21:46, 0.00s elapsed
[*] Nmap: Initiating NSE at 21:46
[*] Nmap: Completed NSE at 21:46, 0.00s elapsed
[*] Nmap: Read data files from: /usr/bin/../share/nmap
[*] Nmap: Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
[*] Nmap: Nmap done: 1 IP address (1 host up) scanned in 708.92 seconds

msf6 >
```

Server Software and Technology Found

SOFTWARE / VERSION	CATEGORY
Next.js 12.3.1	Web frameworks, Web servers
Node.js	Programming languages
Cloudflare Network Error Logging	Issue trackers
Cloudflare	CDN
webpack	Miscellaneous
PWA	Miscellaneous
HTTP/3	Miscellaneous
React	JavaScript frameworks
Vercel	PaaS
core-js 2.6.12	JavaScript libraries
HSTS	Security

Shodan Network Scanner


Open Ports
80
443
2052
2082
2083
2087
8080
8443
8880
// 80 / TCP
1005085571 | 2022-11-30T11:58:00.653063
CloudFlare
HTTP/1.1 403 Forbidden
Date: Wed, 30 Nov 2022 11:57:59 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 5895
Connection: close
X-Frame-Options: SAMEORIGIN
Referrer-Policy: same-origin
Cache-Control: private, max-age=0, no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Expires: Thu, 01 Jan 1970 00:00:01 GMT
Vary: Accept-Encoding
Server: cloudflare
CF-RAY: 7723689eb8265b3e-FRA

B

Penetration Testing

ARMITAGE