

-Install Burp Suite and configure the necessary Certificate Authority.

I successfully installed Burp Suite and completed the required Certificate Authority setup as follows:

To begin, I executed the command `'apt-get install burpsuite'` in the CLI to download and install Burp Suite.

I obtained the necessary SSL certificates from PortSwigger's official website and proceeded to import them into my web browser. This step was crucial to ensure that my browser recognized Burp Suite as a trusted Certificate Authority (CA).

To focus my testing efforts, I added specific websites, www.instagram.com and www.google.com, to my **scope settings** within Burp.

I applied a rule in the **Request/Response Interception Rules** to exclusively intercept and include URLs within the target scope. This configuration prevented unwanted interference from other browser traffic during testing.

On my Firefox web browser, I manually adjusted the proxy settings to route all traffic through 127.0.0.1:8080, effectively channeling it through Burp Suite.

-Browse the internet with your proxy on and observe the traffic. Does anything surprise you? Try editing requests in the Repeater.

I observed an intriguing pattern while using Google's search functionality. When I initiated a search for "**instagram**," I noticed that for every letter I typed, a distinct query was being sent to the server. These queries look like "**q=i**", "**q=in**", "**q=ins**", and so forth, with each letter triggering a new query. This behavior is most likely associated with the way Google's search system operates and how it communicates with its servers. It is a fundamental component of Google's auto-suggest or autocomplete feature, which delivers real-time search suggestions as users type.

Here's a breakdown of how this process typically functions:

As I start typing a search query on Google (in this case, "instagram"), my web browser transmits a request to Google's servers.

Google's servers analyze each and every request and attempt to predict what I might be searching for based on the characters I've entered up to that point.

Google then responds with a list of suggested search queries that match my input. With each additional letter I type, a new request is sent to Google's servers to fetch updated suggestions.

One noteworthy detail that I noticed from examining the headers of these requests was the presence of the "**cp**" variable.

```
GET
/complete/search?q=i&cp=1&client=gws-wiz&xssi=t&gs_pcrt=undefined&hl=en&authuser=0&psi=8I0kZePYCu6q0PEPio2mwA8.1696894448830&dpr=1
GET
/complete/search?q=in&cp=2&client=gws-wiz&xssi=t&gs_pcrt=undefined&hl=en&authuser=0&psi=8I0kZePYCu6q0PEPio2mwA8.1696894448830&dpr=1
...
GET
/complete/search?q=instagram&cp=9&client=gws-wiz&xssi=t&gs_pcrt=undefined&hl=en&authuser=0&psi=8I0kZePYCu6q0PEPio2mwA8.1696894448830&dpr=1
```

It became apparent that this variable incremented in correspondence with the number of characters I provided in my search input. This mechanism likely serves to assist the search engine in sending partial queries to its servers with each keystroke. It may also signify a checkpoint at which Google's autocomplete suggestions should start appearing.

**-Research what the HTTP security headers and Cookie Flags mean and report what several do.
(at least 3)**

HTTP Security Headers:

1. **HTTP Strict Transport Security (HSTS):** HSTS header informs the browser to only access a website over HTTPS. This helps prevent SSL-stripping attacks.
2. **X-Content-Type-Options:** This header safeguards against browsers interpreting files differently from how they are declared by the server, reducing the vulnerability to content-type attacks.
3. **Content Security Policy (CSP):** CSP defines the permissible sources of content for a web page, preventing cross-site scripting (XSS) attacks.

Cookie Flags:

1. **Secure:** The "Secure" flag signifies that the cookie should solely be transmitted over secure HTTPS connections, elevating the overall security of the cookie.
2. **HttpOnly:** The "HttpOnly" flag acts as a safeguard against JavaScript accessing the cookie, minimizing the risk of cross-site scripting (XSS) attacks.
3. **SameSite:** The "SameSite" flag regulates when a cookie should be included in cross-origin requests, helping mitigate cross-site request forgery (CSRF) attacks.

- Try logging into a site you normally visit, take screenshots of the auth process in Burp (hide your credentials) and explain how the web server is logging you into the app.

1. Initially, upon opening the Instagram page, I initiated a GET request from my web browser to retrieve the Instagram page. This GET request included a User-Agent header.

```
Request to https://www.instagram.com:443 [157.240.28.174]
Forward Drop Intercept is on Action Open browser

Pretty Raw Hex
1 GET / HTTP/1.1
2 Host: www.instagram.com
3 Cookie: csrftoken=R7IyFFEBmFz47G01ebFDgEdA4JkCz0Ed; mid=ZSS0ZgAEAGhbIJZtbAxCuu5g4z0; ig_did=573A29E1-7E98-4216-BF16-8393688C0788;
ig_nrcb=1; datr=0o4kZeu8UwMaDpk2yioYamNx; rur=
"LDC\0545415960636\0541728433802:01f77017debf213f99dbff4ea83b726de3fb00bba0c6bd36f9a4ff2e145253abad889851"; ds_user_id=5415960636; shbid=
"10032\0545415960636\0541728432943:01f7530792e302c927a3c9bf1dd48709568bc6f31cb0ecf2581cda310b876e9c0ad2a0a4"; shbts=
"1696896943\0545415960636\0541728432943:01f7af444d7d6da67069fe928d410ddb82f8a2cd7b4867fc180a1fb085c9435f1a8dd6dc"; sessionid=
54159606363Aug8tMeSS1ZkEn0%3A25%3AAyCeYAF1DjTdCKBv011Pn9DsBtf5cHUcgSLfInSKYQ
4 User-Agent: Mozilla/5.0 (X11; Linux aarch64; rv:109.0) Gecko/20100101 Firefox/115.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Upgrade-Insecure-Requests: 1
9 Sec-Fetch-Dest: document
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-Site: none
12 Sec-Fetch-User: ?1
13 Te: trailers
14 Connection: close
```

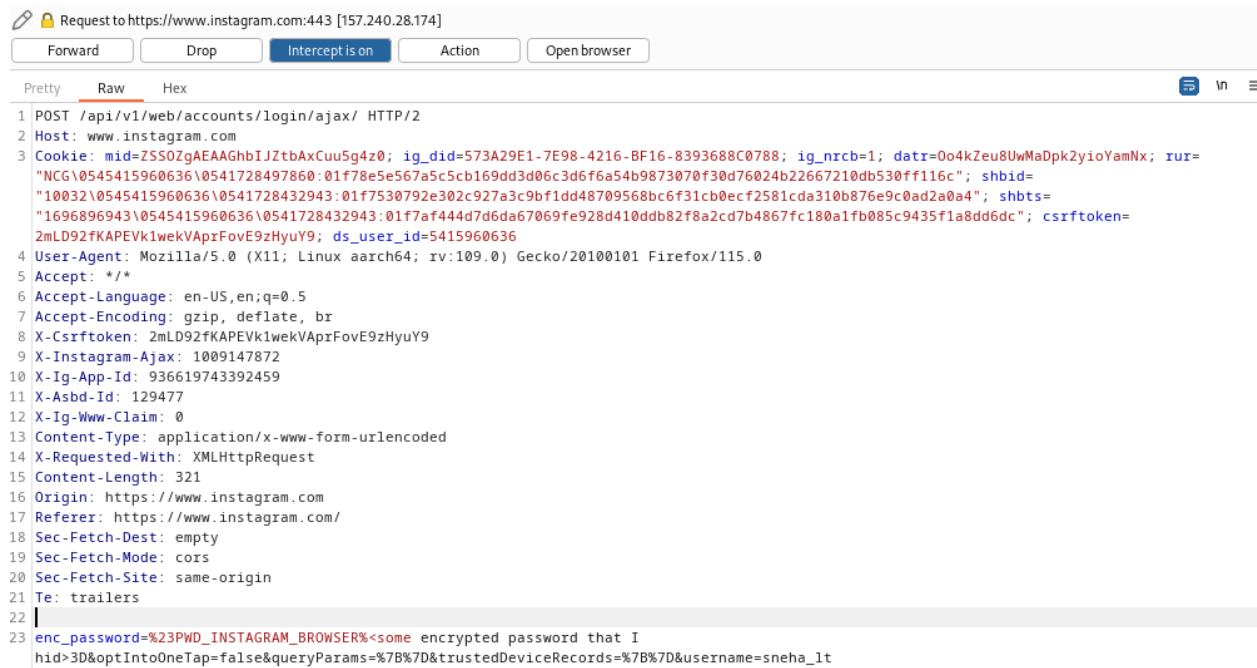
2. Another GET request was made to the URL `"/api/v1/public/landing_info/"` on the Instagram website. This request was aimed at fetching landing page information, which might include essential data required for rendering the Instagram login page. The URL path `"/api/v1/public/landing_info/"` suggests its purpose of obtaining landing page-related data (instagram's login page). Following this request, I received Instagram's login page.

```
Request to https://www.instagram.com:443 [157.240.28.174]
Forward Drop Intercept is on Action Open browser

Pretty Raw Hex
1 GET /api/v1/public/landing_info/ HTTP/2
2 Host: www.instagram.com
3 Cookie: mid=ZSS0ZgAEAGhbIJZtbAxCuu5g4z0; ig_did=573A29E1-7E98-4216-BF16-8393688C0788; ig_nrcb=1; datr=0o4kZeu8UwMaDpk2yioYamNx; rur=
"NCG\0545415960636\0541728485297:01f77eb66639f3f76948694d194e0b0738dbbfa4cf2660468ef1901b518d45a30eb5f818"; shbid=
"10032\0545415960636\0541728432943:01f7530792e302c927a3c9bf1dd48709568bc6f31cb0ecf2581cda310b876e9c0ad2a0a4"; shbts=
"1696896943\0545415960636\0541728432943:01f7af444d7d6da67069fe928d410ddb82f8a2cd7b4867fc180a1fb085c9435f1a8dd6dc"; csrftoken=
Q3egEahInyIK0v23yriRR8za10ILTtr
4 User-Agent: Mozilla/5.0 (X11; Linux aarch64; rv:109.0) Gecko/20100101 Firefox/115.0
5 Accept: */*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 X-Csrftoken: Q3egEahInyIK0v23yriRR8za10ILTtr
9 X-Ig-App-Id: 936619743392459
10 X-Asbd-Id: 129477
11 X-Ig-Www-Claim: 0
12 X-Requested-With: XMLHttpRequest
13 Referer: https://www.instagram.com/
14 Sec-Fetch-Dest: empty
15 Sec-Fetch-Mode: cors
16 Sec-Fetch-Site: same-origin
17 Te: trailers
--
```

Another thing to note here is the Sec-Fetch-* parameters. The "Sec-Fetch" headers are part of the browser's security mechanisms to provide information about the context and security of a request, especially in the context of cross-origin requests. The Sec-Fetch-Mode is set to "cors," which stands for Cross-Origin Resource Sharing. This means that the request is being made from one origin (firefox) to a different origin (Instagram's servers). It just is mainly used to determine the relationship between the client and the server and the server uses this information to decide if the request should be allowed or not. For example, same-origin requests would be allowed by default but might need more checking mechanisms if the request is cors.

4. Following this, I entered my username and password.



The screenshot shows a web browser's developer tools with the 'Network' tab selected. A request to `https://www.instagram.com:443 [157.240.28.174]` is highlighted. The request is a POST to `/api/v1/web/accounts/login/ajax/` with HTTP/2. The 'Raw' tab is active, displaying the raw HTTP request. The request includes a long cookie string, a user agent, and various headers. The body of the request is a JSON object containing an encrypted password and a trusted device record.

```
1 POST /api/v1/web/accounts/login/ajax/ HTTP/2
2 Host: www.instagram.com
3 Cookie: mid=ZSS0ZgAEAGhbIJZtbAxCu5g4z0; ig_did=573A29E1-7E98-4216-BF16-8393688C0788; ig_nrcb=1; datr=0o4kZeu8UwMaDpk2yioYamNx; rur=
  "NCG\0545415960636\0541728497860:01f78e5e567a5c5cb169dd3d06c3d6f6a54b9873070f30d76024b22667210db530ff116c"; shbid=
  "10032\0545415960636\0541728432943:01f7530792e302c927a3c9bf1dd48709568bc6f31cb0ecf2581cda310b876e9c0ad2a0a4"; shbts=
  "1696896943\0545415960636\0541728432943:01f7af444d7d6da67069fe928d410ddb82f8a2cd7b4867fc180a1fb085c9435f1a8dd6dc"; csrftoken=
  2mLD92fKAPEVklwekVAprFovE9zHyyY9; ds_user_id=5415960636
4 User-Agent: Mozilla/5.0 (X11; Linux aarch64; rv:109.0) Gecko/20100101 Firefox/115.0
5 Accept: */*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 X-Csrftoken: 2mLD92fKAPEVklwekVAprFovE9zHyyY9
9 X-Instagram-Ajax: 1009147872
10 X-Ig-App-Id: 936619743392459
11 X-Asbd-Id: 129477
12 X-Ig-Www-Claim: 0
13 Content-Type: application/x-www-form-urlencoded
14 X-Requested-With: XMLHttpRequest
15 Content-Length: 321
16 Origin: https://www.instagram.com
17 Referer: https://www.instagram.com/
18 Sec-Fetch-Dest: empty
19 Sec-Fetch-Mode: cors
20 Sec-Fetch-Site: same-origin
21 Te: trailers
22
23 enc_password=%23PWD_INSTAGRAM_BROWSER%<some encrypted password that I
hid>3D&optIntoOneTap=false&queryParams=%7B%7D&trustedDeviceRecords=%7B%7D&username=sneha_lt
```

5. The Instagram servers on the backend proceeded to authenticate my credentials within their database. If my credentials proved valid, the server, generated an access token to signify my authenticated status. This access token was then stored as a cookie within my web browser. Subsequently, Instagram's server responded by sending an access token along with a login_success token. My web browser subsequently transmitted this access_token to request the Instagram home page.

```
Request to https://www.instagram.com:443 [157.240.28.174]
Forward Drop Intercept is on Action Open browser
Pretty Raw Hex
1 POST /logging/falco HTTP/2
2 Host: www.instagram.com
3 Cookie: mid=ZSS0ZgAEAGhbIJZtbAxCuu5g4z0; ig_did=573A29E1-7E98-4216-BF16-8393688C0788; ig_nrcb=1; datr=0o4kZeu8UwMaDpk2yioYamNx; rur=
  "FRC\0545415960636\0541728498514:01f77007d25cc315c36e132b4497ec8d9891353e5c4feacedae9a6645344423d34ff67e4"; shbid=
  "10032\0545415960636\0541728432943:01f7530792e302c927a3c9bf1dd48709568bc6f31cb0ecf2581cda310b876e9c0ad2a0a4"; shbts=
  "1696896943\0545415960636\0541728432943:01f7af444d7d6da67069fe928d410ddb82f8a2cd7b4867fc180a1fb085c9435f1a8dd6dc"; csrftoken=
  K23MPA2e6MkftDepUqXQzK0GI1k5Ivny; ds_user_id=5415960636; sessionid=
  54159606363AzokuyJdDZy8LCq%3A14%3AAydcEbJWdhMZDp5tK8LYEx5R2ocZqb8n7uaJi_ei-Q
4 User-Agent: Mozilla/5.0 (X11; Linux aarch64; rv:109.0) Gecko/20100101 Firefox/115.0
5 Accept: */*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 X-Csrftoken: K23MPA2e6MkftDepUqXQzK0GI1k5Ivny
9 X-Ig-App-Id: 936619743392459
10 X-Asbd-Id: 129477
11 X-Ig-Www-Claim: hmac.AR2oIC5zpCX0ymaQf4bB6ejbFyxwc7Fhd5SpfV0xaltikeyU
12 Content-Type: application/x-www-form-urlencoded
13 X-Requested-With: XMLHttpRequest
14 Content-Length: 1014
15 Origin: https://www.instagram.com
16 Referer: https://www.instagram.com/
17 Sec-Fetch-Dest: empty
18 Sec-Fetch-Mode: cors
19 Sec-Fetch-Site: same-origin
20 Te: trailers
21
22 access_token=936619743392459%7C3cdb3f896252a1db29679cb4554db266&message=
  %7B%22app_id%22%3A%22936619743392459%22%2C%22app_uid%22%3A%220%22%2C%22app_ver%22%3A%221.0.0%22%2C%22claims%22%3A%5B%22hmac.AR2oIC5zpCX0yma
  Qf4bB6ejbFyxwc7Fhd5SpfV0xaltikeyU%22%5D%22%2C%22data%22%3A%5B%7B%22extra%22%3A%7B%22event_name%22%3A%22login_success%22%2C%22fbconnect_status%
  22%3A%22logged_out%22%2C%22frontend_env%22%3A%22C3%22%2C%22ig_userid%22%3A%225415960636%22%2C%22login_identifier%22%3A%22sneha_1t%22%2C%22li
  ogin_identifier_type%22%3A%22username%22%2C%22login_source%22%3A%22anul1%22%2C%22login_type%22%3A%22password%22%2C%22nav_chain%22%3A%22PolarisHome
```

6. Afterward, my browser sent a GET request to retrieve Instagram's home page (which includes the page asking if I want to enable onetap). Following the forwarding, I successfully received Instagram's home page. It's worth noting that the server continues to verify the access token with each subsequent request to ensure I have the necessary permissions.

```
Request to https://www.instagram.com:443 [157.240.28.174]
Forward Drop Intercept is on Action Open browser
Pretty Raw Hex
1 GET /accounts/onetap/?next=%2F HTTP/2
2 Host: www.instagram.com
3 Cookie: mid=ZSS0ZgAEAGhbIJZtbAxCuu5g4z0; ig_did=573A29E1-7E98-4216-BF16-8393688C0788; ig_nrcb=1; datr=0o4kZeu8UwMaDpk2yioYamNx; rur=
  "FRC\0545415960636\0541728497170:01f7e0b0a6ba64a11f1d842efc61fc547e812bd2c350024c2cf12dd7e7586c96c12a41df"; shbid=
  "10032\0545415960636\0541728432943:01f7530792e302c927a3c9bf1dd48709568bc6f31cb0ecf2581cda310b876e9c0ad2a0a4"; shbts=
  "1696896943\0545415960636\0541728432943:01f7af444d7d6da67069fe928d410ddb82f8a2cd7b4867fc180a1fb085c9435f1a8dd6dc"; csrftoken=
  2mLD92fKAPEVkiwekVApFovE9zHyuY9; ds_user_id=5415960636; sessionid=
  54159606363A50Ljwkth7bJT2%3A25%3AAyF254jufQ-Z54BP1uTiCSWK29EsjdS8x-qZ0V3aIA
4 User-Agent: Mozilla/5.0 (X11; Linux aarch64; rv:109.0) Gecko/20100101 Firefox/115.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://www.instagram.com/
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Te: trailers
```