Twitter Penetration Test Report

BI-EHA Semestral work

Jan Koníř, Tadeáš Kouba

FIT CTU, 2023

Table of Contents

Introduction

Team Info

- Jan Konír
  - konirjan@fit.cvut.cz
  - Student of CTU, Faculty of Information Technology
  - Responsible for
    - 4.2 Configuration and Deployment Management Testing
    - 4.6 Session Management Testing
    - 4.7 Input Validation Testing
      - 4.7.10 Testing for IMAP SMTP Injection
      - 4.7.11 Testing for Code Injection
        - 4.7.11.1 Testing for Local File Inclusion
        - 4.7.11.2 Testing for Remote File Inclusion
      - 4.7.12 Testing for Command Injection
      - 4.7.13 Testing for Format String Injection
      - 4.7.14 Testing for Incubated Vulnerability
      - 4.7.15 Testing for HTTP Splitting Smuggling
      - 4.7.16 Testing for HTTP Incoming Requests
      - 4.7.17 Testing for Host Header Injection
      - 4.7.18 Testing for Server-side Template Injection
      - 4.7.19 Testing for Server-side Request Forgery
    - 4.9 Testing for Weak Cryptography
    - 4.10 Business Logic Testing

- Tadeáš Kouba
  - koubatad@fit.cvut.cz
  - Student of CTU, Faculty of Information Technology
  - Responsible for
    - 4.3 Identity Management Testing
    - 4.4 Authentication Testing
    - 4.5 Authorization Testing
    - 4.7 Input Validation Testing
      - 4.7.1 Testing for Reflected Cross Site Scripting
      - 4.7.2 Testing for Stored Cross Site Scripting
      - 4.7.3 Testing for HTTP Verb Tampering
      - 4.7.4 Testing for HTTP Parametr Pollution
      - 4.7.5 Testing for SQL Injection
      - 4.7.6 Testing for LDAP Injection
      - 4.7.7 Testing for XML Injection
      - 4.7.8 Testing for SSI Injection
      - 4.7.9 Testing for XPath Injection
    - 4.8 Testing for Error Handling

Project Overview

This report is part of semestral work for subject Ethical hacking of the Information Security program on FIT, CTU.
The focus is to assert skills gained during the semester as well as use them in real-world scenario. The website twitter.com has been chosen for this purpouse as it offers a bug bounty and is therefore eligible for ethical hackers to try and hack.

Scope Description

We will be focusing on the web application on domain twitter.com.
While the bug bounty posted by twitter puts some vulnerabilities out scope for financial reward, we will still be trying to perform these attacks as to deepen our ethical hacking skills.

Pentesting Methodology

As a pentesting methodology we have chosen the OWASP Web Application Security Testing guide - version 4.2, which is the latest stable version. To be more precise, we will be covering following topics:

- Configuration and Deployment Management Testing
- Identity Management Testing
- Authentication Testing
- Authorization Testing
- Session Management Testing
- Input Validation Testing
- Testing for Error Handling
- Testing for Weak Cryptography
- Business Logic Testing
- Client-side Testing

Scoring System

We will be using the widely used Common Vulnerability Scoring System (CVSS) standard for evaulating the threat levels of vulnerabilities found in this report. In particular we will use the CVSS 3.1 version. The standard defines multiple metrics (base, temporal, environmental), however we will use only the base metric since we are performing a one-time pentesting as external entity so the temporal and environmental metrics are not that important and available to us.
We will be scoring each vulnerability on several factors such as access complexity, confidentiality impact, integrity impact and others, to calculate the final score. This will determine the final rating of that vulnerability as seen in the following table.

Rating	CVSS Score
Low	0.1 - 3.9
Medium	4.0 - 6.9
High	7.0 - 8.9
Critical	9.0 - 10.0

Executive Summary

TBD

List of Findings

TBD

Pre-engagement

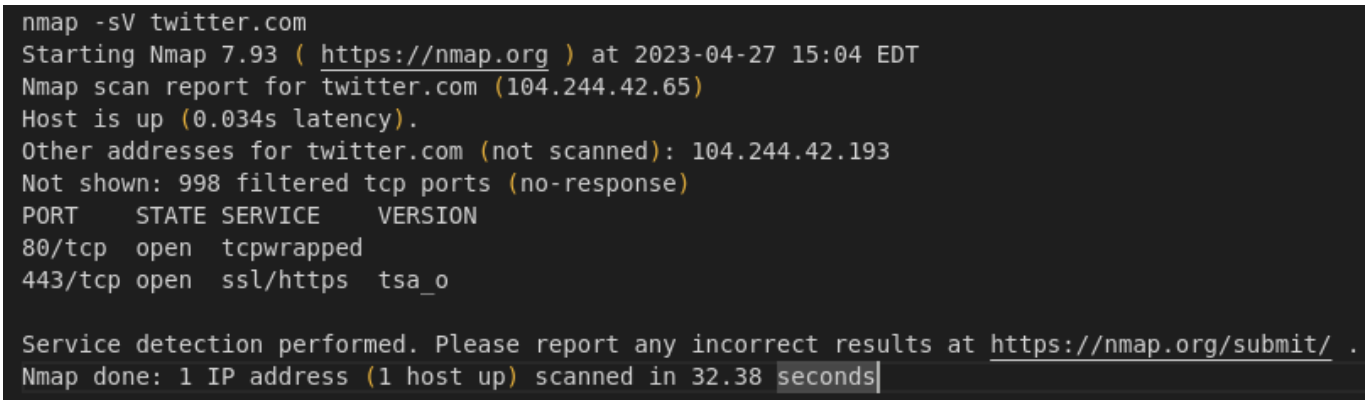
Threat Model

Twitter does not openly disclose how its backend communication works.
Since twitter is a web application it is obviously exposed to the internet which naturally puts it in proximity of multitude of threats. On top of that it is definitely an attractive target for hackers, since it stores personal information of about 350 million people.
Some vulnerable points might be the client-server-database communication for authentication or tweets being stored in database and showing to completely random users' feeds.

Intelligence-gathering Outcomes

Firstly we ran port scan with **nmap**. We used the following command to find the open ports and possibly the version of service running on them:

```
nmap -sV twitter.com
```



Full output can be seen in *outputs/nmap_version.txt*.
We can see that there are two open tcp ports (80, 443).

We then ran **nmap** with intention to determine the OS running, version and traceroute with:

```
nmap -A twitter.com
```

```
PORT    STATE SERVICE   VERSION
80/tcp  open  http      tsa_o
|_http-title: Did not follow redirect to https://twitter.com/
| fingerprint-strings:
|   DNSVersionBindReqTCP:
|     HTTP/1.1 400 Bad Request
|     content-length: 11
|     content-type: text/plain
|     x-connection-hash: e41ea0468e7b5be750935d042454e8711e3b3961c1737992a4cc017f6081eee2
|     date: Thu, 27 Apr 2023 19:07:40 GMT
|     server: tsa_o
|     connection: close
|     Request
|   FourOhFourRequest:
|     HTTP/1.0 400 Bad Request
|     x-connection-hash: 02d1cd22645d11f2bf49cdace64f2604c7e3a924a2cfc0bc51ef47d0e2cb5f6b
|     date: Thu, 27 Apr 2023 19:07:33 GMT
|     server: tsa_o
|     connection: close
|     content-length: 0
|   GetRequest:
|     HTTP/1.0 400 Bad Request
|     x-connection-hash: f18259468620a4e8ad658e4dc615da660dc1215ad1a001d84c1e96c4e3d25137
|     date: Thu, 27 Apr 2023 19:07:29 GMT
|     server: tsa_o
|     connection: close
|     content-length: 0
|   HTTPOptions:
|     HTTP/1.0 400 Bad Request
|     x-connection-hash: 53ab8f7fb60e3463465249882514a86891e613e9036689229e062378af384b46
|     date: Thu, 27 Apr 2023 19:07:30 GMT
|     server: tsa_o
|     connection: close
|     content-length: 0
```

```
443/tcp open  ssl/https tsa_o
| ssl-cert: Subject: commonName=twitter.com/organizationName=Twitter, Inc./stateOrProvinceName=
| Subject Alternative Name: DNS:twitter.com, DNS:www.twitter.com
| Not valid before: 2023-02-05T00:00:00
|_Not valid after:  2024-02-05T23:59:59
|_http-server-header: tsa_o
|_ssl-date: TLS randomness does not represent time
| http-robots.txt: 13 disallowed entries
| /search/realtime /search/users /search/*/grid /*?
| /*/followers /*/following /account/deactivated
|_/settings/deactivated /oauth /1/oauth /i/streams /i/hello /i/u
|_http-title: Site doesn't have a title (text/html; charset=utf-8).
| fingerprint-strings:
|   DNSVersionBindReqTCP:
|     HTTP/1.1 400 Bad Request
|     content-length: 11
|     content-type: text/plain
|     x-connection-hash: 4a220339f0d48a93266c291c45e2d2945f95f6e4c4d9e0be7a4b8c0e7af86e2a
|     date: Thu, 27 Apr 2023 19:07:47 GMT
|     server: tsa_o
|     connection: close
|     Request
|   FourOhFourRequest:
|     HTTP/1.0 400 Bad Request
|     x-connection-hash: 562463708a4194b26a6cd538525aeb5c4ed5db1011d43b9d221accfdab4b191d
|     date: Thu, 27 Apr 2023 19:07:37 GMT
|     server: tsa_o
|     connection: close
|     content-length: 0
|   GetRequest:
|     HTTP/1.0 400 Bad Request
|     x-connection-hash: 8627003e329bef15b9d472e46b19be7b09d9ed3834bdd984dde2be922899bd26
|     date: Thu, 27 Apr 2023 19:07:35 GMT
|     server: tsa_o
|     connection: close
|     content-length: 0
```

Full output can be seen in *outputs/nmap_A.txt*.
Here it is specified that on port 80 there is http running. Also all requests sent are returned with HTTP 400 so service as well as OS could not be recognized.

Then we ran **dnsenum** to try and gather some infromation about the domain:

```
dnsenum --enum twitter.com
```

```
Host's addresses:
__________________

twitter.com.                            1007    IN    A        104.244.42.193


Name Servers:
_____________

a.u06.twtrdns.net.                      91      IN    A        204.74.66.101
b.r06.twtrdns.net.                      97      IN    A        205.251.196.198
b.u06.twtrdns.net.                      10      IN    A        204.74.67.101
c.r06.twtrdns.net.                      103     IN    A        205.251.194.151
c.u06.twtrdns.net.                      55      IN    A        204.74.110.101
a.r06.twtrdns.net.                      91      IN    A        205.251.192.179
b.r06.twtrdns.net.                      97      IN    A        205.251.196.198
b.u06.twtrdns.net.                      10      IN    A        204.74.67.101
c.r06.twtrdns.net.                      103     IN    A        205.251.194.151
c.u06.twtrdns.net.                      55      IN    A        204.74.110.101
a.r06.twtrdns.net.                      91      IN    A        205.251.192.179
a.u06.twtrdns.net.                      91      IN    A        204.74.66.101
d.r06.twtrdns.net.                      11      IN    A        205.251.199.195
d.u06.twtrdns.net.                      12      IN    A        204.74.111.101


Mail (MX) Servers:
__________________

aspmx3.googlemail.com.                  224     IN    A        142.250.150.26
aspmx2.googlemail.com.                  221     IN    A        142.251.9.26
aspmx.l.google.com.                     71      IN    A        173.194.76.27
alt2.aspmx.l.google.com.                193     IN    A        142.251.9.26
alt1.aspmx.l.google.com.                194     IN    A        142.250.153.27
```

Full output can be seen in *outputs/dnsenum.txt*.
Which listed us a lot of subdomains and IP addresses of twitter.

As stated before the twitter developers don't share much information about twitter. However through developer blog they sometimes discuss what actually happens on twitter's backend. Through that we were able to find out that twitter used to run on MySQL database, but has switched to distributed database NoSQL system called Manhattan, that is custom developed for twitter. Manhattan itself utilizes multiple open-source technologies like Apache Cassandra or Hadoop. This information might prove useful for performing some SQL Injection attacks.

As for the front-end part of Twitter it mostly consists of HTML, CSS and Javascript, particularly React.js. Of those, React.js is obviously the one to be concentrated on when pentesting. Some of its most common vulnerabilities are Cross-site scripting, SQL injection, Cross-site request forgery, Vulnerability in packages and dependencies, Broken authentication, Zip slip or XML external entities.

Last but not least we tried so called Google Hacking/Dorking but it didn't disclose any important information.

Testing process

4.1 Information Gathering

This topic is covered in the Pre-engagement section.

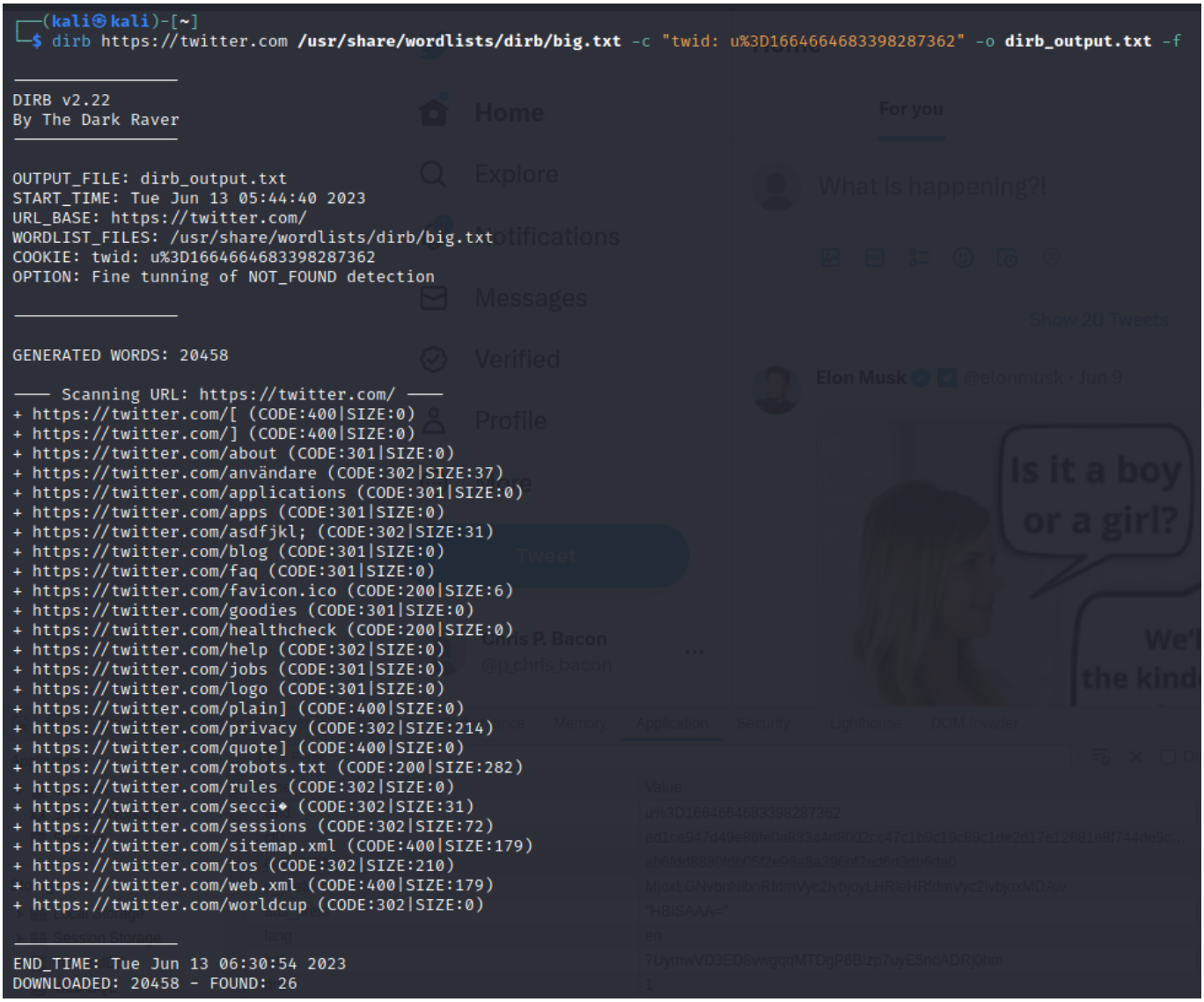
4.2 Configuration and Deployment Management Testing

4.2.1 Test Network Infrastructure Configuration

As we don't have access to neither used systems nor any configuration files, this test cannot be performed.

4.2.2 Test Application Platform Configuration

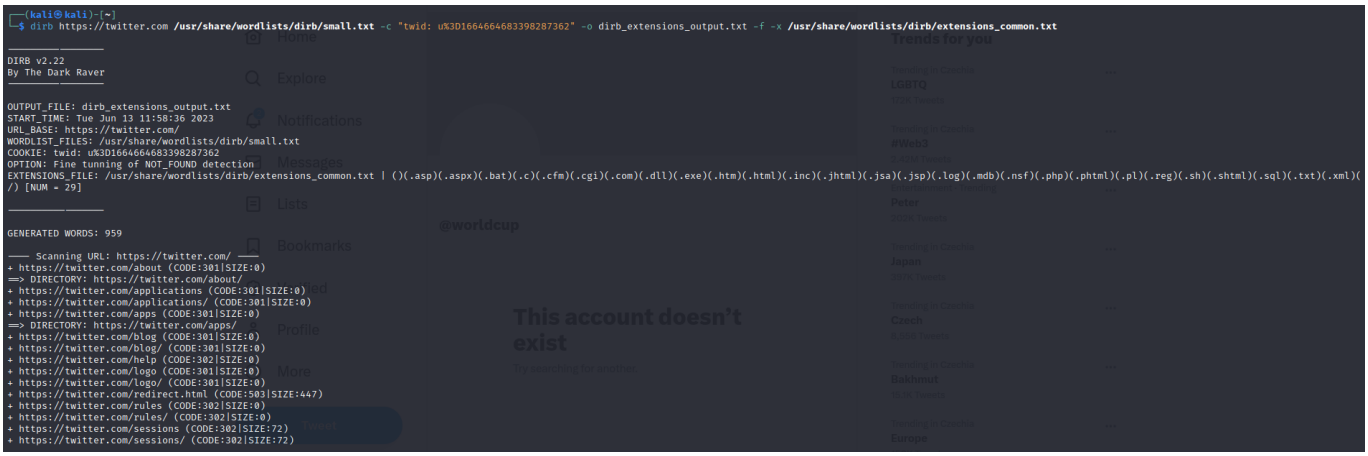
We ran `dirb` to scan for sample and known files and directories.



All of the requests were valid endpoints, redirections, account names or the Twitter's custom "Page was not found" page.

4.2.3 Test File Extensions Handling for Sensitive Information

For this test we ran `dirb` again, but this time with file extensions wordlist. It didn't disclose any new files or pages.



4.2.4 Review Old Backup and Unreferenced Files for Sensitive Information

We haven't found neither endpoints nor functions suitable for inference from the naming scheme.

When reviewing the front-end source code, no useful comments were found.

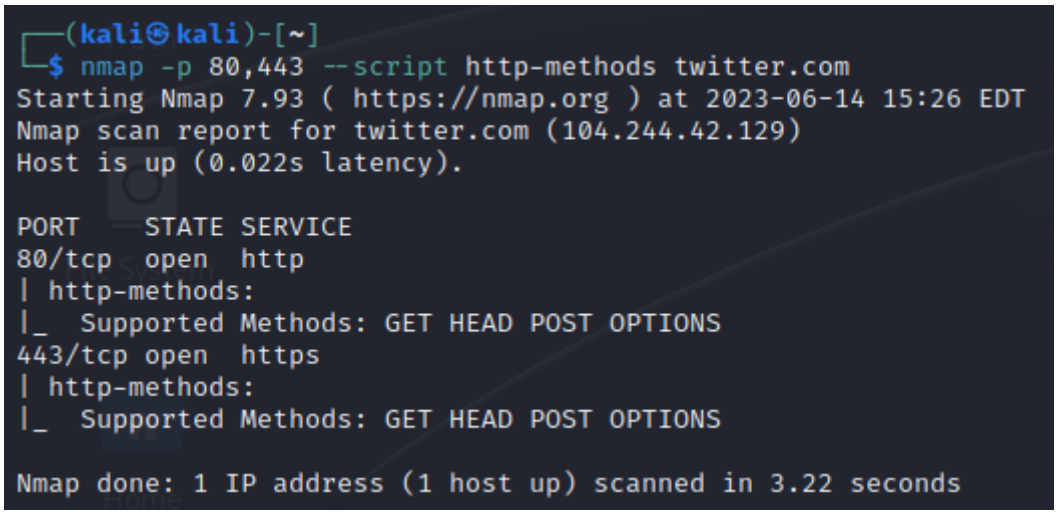
Blind guessing was performed in the previous sections with no success.

4.2.5 Enumerate Infrastructure and Application Admin Interfaces

We weren't able to reveal any admin interface.

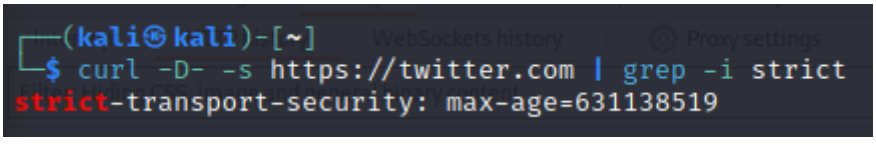
4.2.6 Test HTTP Methods

Running `nmap` with http-methods script disclosed that only reasonable HTTP methods are allowed.



4.2.7 Test HTTP Strict Transport Security

HTTP Strict Transport Security header is included in the response. However it lacks the includeSubDomains directive and since there are public subdomains such as blog.twitter.com or about.twitter.com and none of them probably require usage of unsecured http, the includeSubDomains directive should be included as well.



4.2.8 Test RIA Cross Domain Policy

Neither crossdomain.xml nor clientaccesspolicy.xml are accessible.

4.2.9 Test File Permission

We don't have access to any web server, so we cannot inspect the local file permissions. No online directories were found.

4.2.10 Test for Subdomain Takeover

`dnsrecon` found no records and therefore no records suitable for takeover.

```
┌──(kali㉿kali)-[~]
└─$ dnsrecon -d twitter.com
[*] std: Performing General Enumeration against: twitter.com...
[-] DNSSEC is not configured for twitter.com
[*]     SOA a.u06.twtrdns.net 204.74.66.101
[*]     NS c.u06.twtrdns.net 204.74.110.101
[*]     Bind Version for 204.74.110.101 Nameserver"
[*]     NS b.u06.twtrdns.net 204.74.67.101
[*]     Bind Version for 204.74.67.101 Nameserver"
[*]     NS d.r06.twtrdns.net 205.251.199.195
[*]     NS a.u06.twtrdns.net 204.74.66.101
[*]     Bind Version for 204.74.66.101 Nameserver"
[*]     NS a.r06.twtrdns.net 205.251.192.179
[*]     NS c.r06.twtrdns.net 205.251.194.151
[*]     NS d.u06.twtrdns.net 204.74.111.101
[*]     Bind Version for 204.74.111.101 Nameserver"
[*]     NS b.r06.twtrdns.net 205.251.196.198
[*]     MX alt1.aspmx.l.google.com 142.250.150.27
[*]     MX alt2.aspmx.l.google.com 74.125.200.27
[*]     MX aspmx2.googlemail.com 142.250.150.26
[*]     MX aspmx3.googlemail.com 74.125.200.26
[*]     MX aspmx.l.google.com 108.177.127.27
[*]     MX alt1.aspmx.l.google.com 2a00:1450:4010:c1c::1a
[*]     MX alt2.aspmx.l.google.com 2404:6800:4003:c00::1b
[*]     MX aspmx2.googlemail.com 2a00:1450:4010:c1c::1a
[*]     MX aspmx3.googlemail.com 2404:6800:4003:c00::1a
[*]     MX aspmx.l.google.com 2a00:1450:4013:c01::1b
[*]     A twitter.com 104.244.42.193
[*]     TXT twitter.com 0a8c0fc6-bfa5-4ea7-b09b-87f2989022d6
[*]     TXT twitter.com MS=BEE202D20C326867290BDEFA2DDDF4594B5D6860
[*]     TXT twitter.com adobe-idp-site-verification=a2ff8fc40c434d1d6f02f68b0b1a683e400572ab8c1f2c180c71c3d985b9270
a
[*]     TXT twitter.com apple-domain-verification=zd1iHoEO9LILEQIq
[*]     TXT twitter.com atlassian-domain-verification=j6u0o1PTkobCXC84uEF/sWpIPtaZURBVYqKzmTvT8wugLcHT1vvrzzA63iP1q
SLN
[*]     TXT twitter.com bj6sbt5xqs9hw9jrfvz7hplrg0l680sb
[*]     TXT twitter.com canva-site-verification=lMnZ3wMh7c1uqZqa-cxZTg
[*]     TXT twitter.com google-site-verification=TNhAkfLUeIbzzzSgPNxS5aEkKMf3aUcpPmCK1_kmIvU
[*]     TXT twitter.com google-site-verification=h6dJIv0HXjLOkGAotLAWEzvoi9SxqP4vjpx98vrCvvQ
[*]     TXT twitter.com loom-site-verification=638c6bc173b9458997f64d305bf42499
[*]     TXT twitter.com miro-verification=6e1ca9ad6d0c2cd2e4186141265f23ed618cfe37
[*]     TXT twitter.com mixpanel-domain-verify=164dda91-31f4-41e8-a816-0f59b38fea30
[*]     TXT twitter.com traction-guest=6882b04e-4188-4ff9-8bb4-bff5a3d358e6
[*]     TXT twitter.com traction-guest=a4d0248d-fe01-4222-8fcc-33f68323e667
[*]     TXT twitter.com v=spf1 ip4:199.16.156.0/22 ip4:199.59.148.0/22 ip4:8.25.194.0/23 ip4:8.25.196.0/23 ip4:204.
92.114.203 ip4:204.92.114.204/31 include:_spf.google.com include:_thirdparty.twitter.com include:_oerp.twitter.com i
nclude:spf.smtp2go.com -all
[*]     TXT twitter.com wrike-verification=MjU4MTA5MjoyN2UzNDc1MjU3MDZiZTY4NjBiNzliNDQ2OTUwNWY3NmM5NDgyMTBlYzFkNTcw
YTE2YWNmZDdkNTY2ZmE4Yzlh
[*]     TXT _dmarc.twitter.com v=DMARC1; p=reject; rua=mailto:d3omt-8484@rua.dmarc.emailanalyst.com; ruf=mailto:d3o
mt-8484@ruf.dmarc.emailanalyst.com; fo=1
[*] Enumerating SRV Records
[+] 0 Records Found
```

4.2.11 Test Cloud Storage

abs.twimg.com seems to be a cloud service but it only provides public content such as tweets, additional javascript code, emoji and CSS for the client web application so the authorization for reading is not required.

Updating the file was not successful.

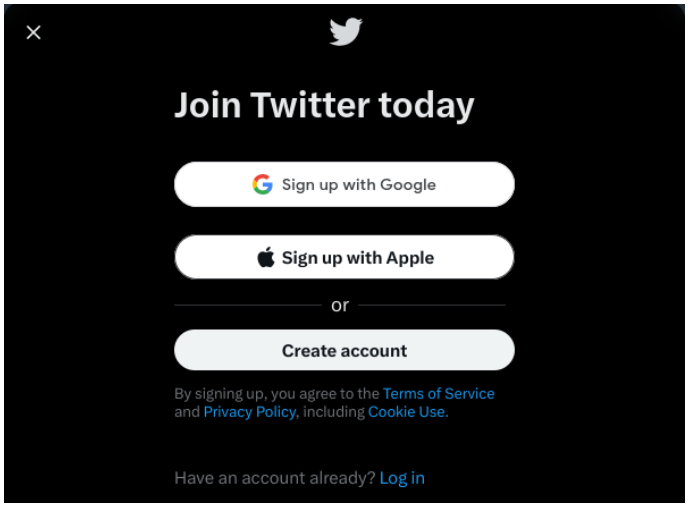
4.3 Identity Management Testing

4.3.1 Test Role Definitions

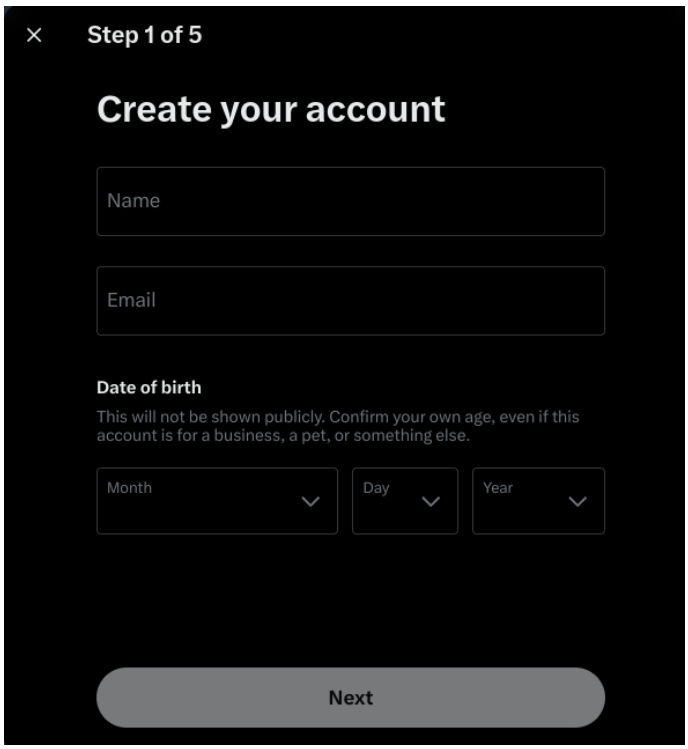
Since the application is being tested as a black-box from a normal user standpoint we don't have information about available roles. Even after looking throught HTTP communication any specific roles could not be identified. Therefore we are not able to test roles layout or permissions.

4.3.2 Test User Registration Process

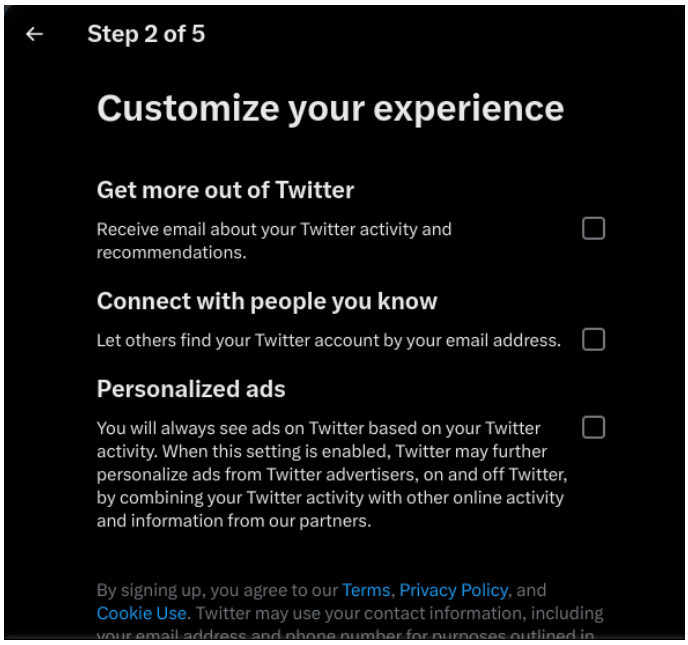
User can register using several different options (with Google/Apple account or native) we will test native registration process as the other two are using different applications and are out of scope.



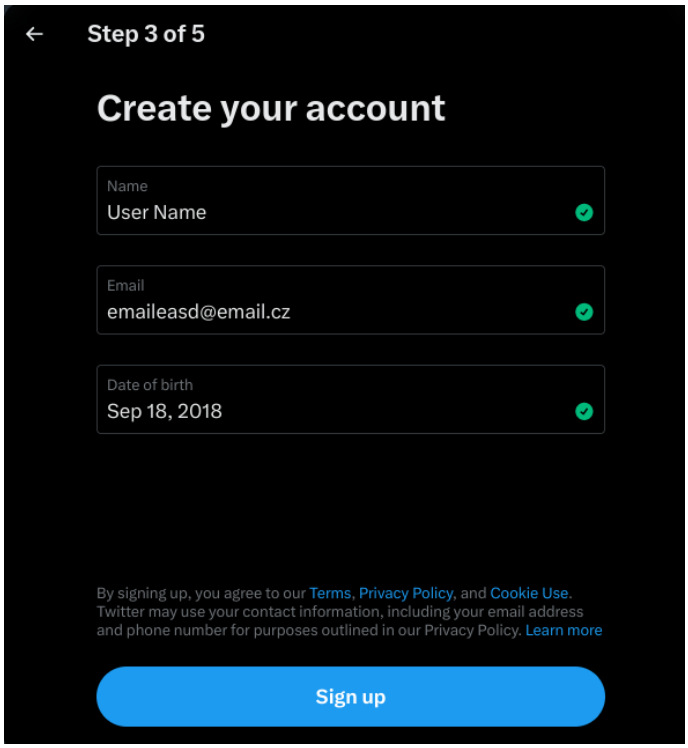
The requirments for registration are name, email and date of birth of user. Out of these options only email can be verified. Name can be anything and does not need to be unique, date of birth can be set to anything as it is not verified either.



In next step user is asked for different permissions which may be granted but are not required.



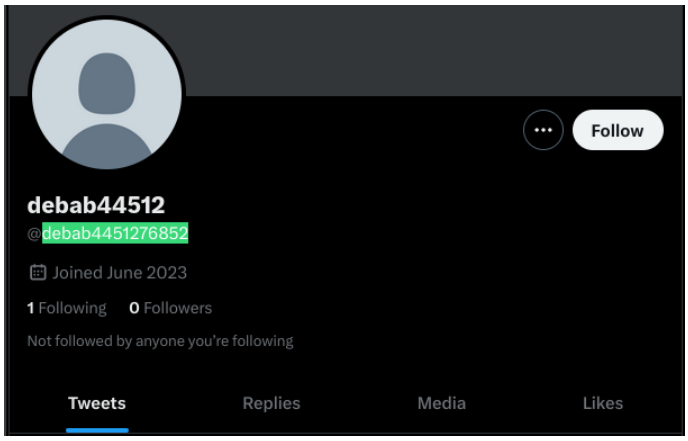
In next step user information is confirmed (Notice that date of birth really does not matter as in the example the user's age is 5 years and email is checked in prior stages).



Next user is prompted to input the code sent to him via email (6 digits). Last step is the creation of password which is described in detail in authentication testing
In summary anyone with access to an email can create twitter account. Registrations are granted automatically after confirmig email validity. As long as one has access to a valid email he can register as many times as he can (for purposes of testing we created 5+ accounts with throwaway emails). Users can only register with basic user permissions and the only proof of identity is email verification.

4.3.4 Testing for Account Enumeration and Guessable User Account

It is quite easy to enumerate emails of accounts either when registrating (already used emails inform the user that email is used) or when trying to log in - when using valid email user is redirected to input his password and when invalid email (not used in registration of any account yet) the following message is shown: `Sorry, we could not find your account.` also a HTTP response **400 Bad Request** is sent. When we have a valid email we are either redirected and can enter password or if suspicious activity is detected (Login from different location/multiple bad password/username guesses) we are redirected to input our username. This makes it more difficult to enumerate and brute force users since we need to know username+email to even be able to try passwords. However we still get 5-15 chances before lock-out to try usernames which can be bruteforced with a bit of educated guess like using name from email or that name + some numbers etc. When wrong username is input page shows message `Incorrect, please try again.` and **400 Bad Request** is sent, user is then redirected to the same page to try again on a correct username guess we are redirected to input password with **200 OK** response. Note that usernames can be seen by any logged in users - every twitter user has his username on his profile and under every tweet as `@<username>` - and everyone can become user and therefore see usernames. So we can actually get all information needed to try to brutforce passwords if we are able to connect an account to it's email. For example I used this method to get back access to a testing account I created and forgot the username of, because twitter appended some random numbers to it (as shown below). Another example is if we got the email used for registration of say Elon Musk (username @elonmusk) we could use this information to effectively lock him out of his twitter account or try to bruteforce his password. It is possible to enumerate user emails and potentially their usernames.



4.3.5 Testing for Weak or Unenforced Username Policy

When promted to set username, twitter by default fills the username field with first part of mail and if ambiguous apends some numbers. User can change his username to anything he pleases as long as it is unique across the application. So the usernames are implicitly structured but in reality can not be guessed with certainty as users can change them.

4.4 Authentication Testing

4.4.1 Testing for Credentials Transported over an Encrypted Channel

Twitter uses TLSv1.2 for data transfer encryption and HTTPS for all communication including login, user creation and password change, it can not be forced to use unsecure HTTP.

tcp.port == 443 && ip.dst == 104.244.42.130

No.	Time	Source	Destination	Protocol	Length	Info
88	5.400148045	10.0.2.15	104.244.42.130	TLSv1.2	206	Application Data
91	5.426801434	10.0.2.15	104.244.42.130	TCP	54	35558 → 443 [ACK] Seq=153 Ack=65 Win=63360 Len=0
92	5.427547926	10.0.2.15	104.244.42.130	TLSv1.2	261	Application Data
93	5.427579300	10.0.2.15	104.244.42.130	TLSv1.2	582	Application Data
97	5.562282422	10.0.2.15	104.244.42.130	TCP	54	35556 → 443 [ACK] Seq=736 Ack=270 Win=63360 Len=0
143	23.373504936	10.0.2.15	104.244.42.130	TLSv1.2	253	Application Data
144	23.373529254	10.0.2.15	104.244.42.130	TLSv1.2	562	Application Data
172	23.527780948	10.0.2.15	104.244.42.130	TCP	54	35556 → 443 [ACK] Seq=1443 Ack=403 Win=63360 Len=0
182	24.069939533	10.0.2.15	104.244.42.130	TLSv1.2	253	Application Data
183	24.070079215	10.0.2.15	104.244.42.130	TLSv1.2	231	Application Data
189	24.217180777	10.0.2.15	104.244.42.130	TCP	54	35556 → 443 [ACK] Seq=1819 Ack=536 Win=63360 Len=0
324	26.357784370	10.0.2.15	104.244.42.130	TLSv1.2	253	Application Data
325	26.357823423	10.0.2.15	104.244.42.130	TLSv1.2	1244	Application Data
326	26.357883746	10.0.2.15	104.244.42.130	TLSv1.2	4086	Application Data
333	26.532384005	10.0.2.15	104.244.42.130	TCP	54	35556 → 443 [ACK] Seq=7240 Ack=665 Win=63360 Len=0
345	27.573375301	10.0.2.15	104.244.42.130	TLSv1.2	141	Application Data
348	27.599338671	10.0.2.15	104.244.42.130	TCP	54	35558 → 443 [ACK] Seq=240 Ack=129 Win=63360 Len=0

Communication capture of login via Wireshark.

4.4.2 Testing for Default Credentials

While testing different credentials in the register page I found out there are users with admin@twitter.com and guest@twitter.com registered.

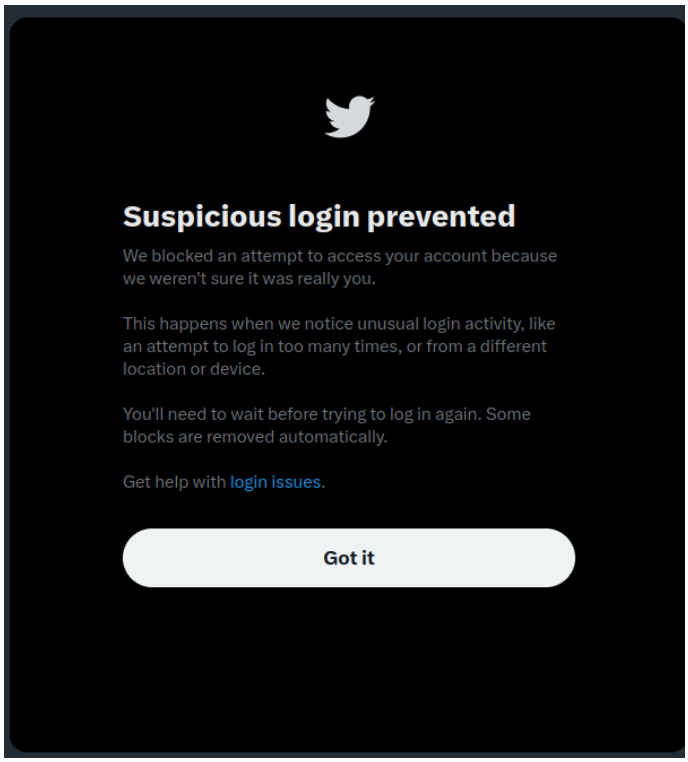
Twittter indeed uses @twitter.com address (for example registering verification mail is sent from verification@email.com) and they state it as official way of communication,

> Twitter will only send you emails from @twitter.com or @e.twitter.com. However, some people may receive fake or suspicious emails that look like they were sent by Twitter. These emails might include malicious attachments or links to spam or phishing websites. Please know that Twitter will never send emails with attachments or request your Twitter password by email.

however there is no way of knowing if they are actual twitter.com default/admin credentials, they can however be used to lock out these accounts as described below.

4.4.3 Testing for Weak Lock Out Mechanism

Twitter's lock out mechanism activates after 5-15 (scales with repeating tries) unsuccessful entries of password/username. Any following tries result in the following message.

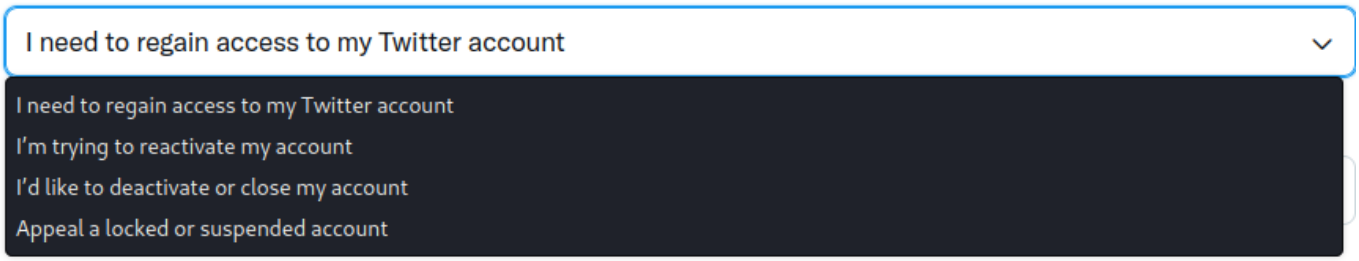


Unlock mechanism is time-based with time-out of 30+ minutes (Twitter officially states it is an hour lock-out, my first lock-out was shorter though).

What does it mean to be locked out?

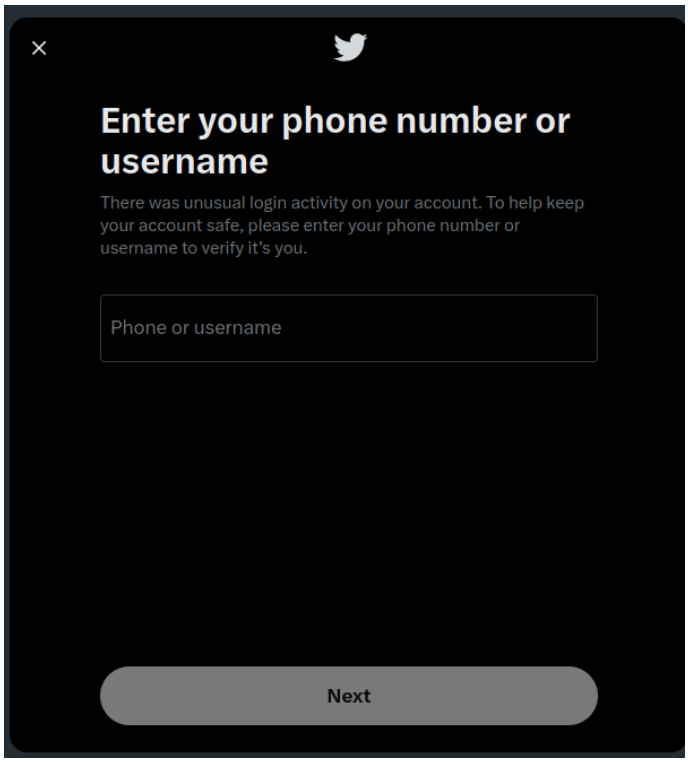
- After a limited number of failed attempts to sign in to Twitter, you will be temporarily locked out from trying to sign in. When your account is locked, you will not be able to sign in — even with the correct password.
- **This lock lasts about an hour and will then clear on its own.**

It also provides possibility to send a form with login issue, but none of the possibilities fit nor help with the lock out issue.



If twitter suspects an unusual login activity (lot of bad tries/new location) it prompts the user to input their

username or phone, this can also be used to lock-out an account, as it uses same policy as password lock-out mechanism.

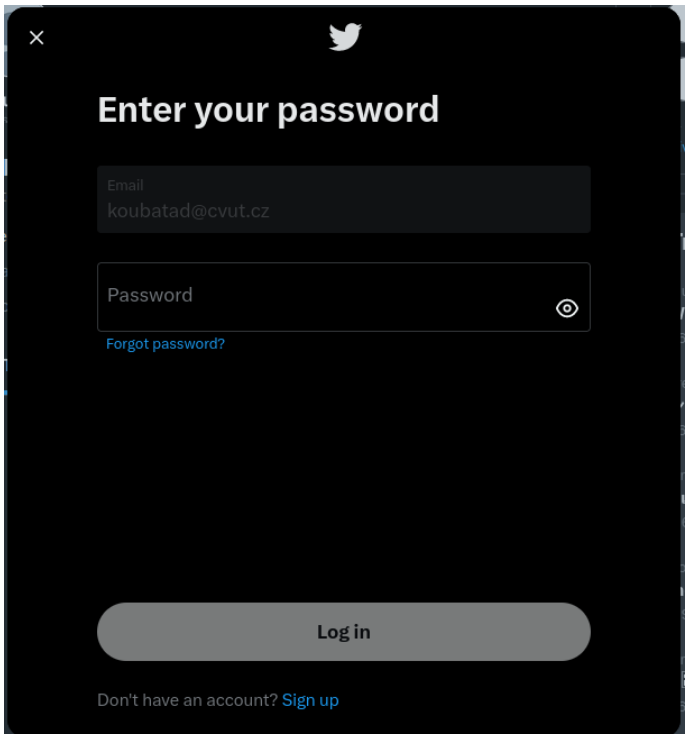


4.4.4 Testing for Bypassing Authentication Schema

It is not possible to bypass authentication through direct page request. It is possible to visit someones profile with `twitter.com/<username>`, however this does not disclose any personal/secret information as it shows someone's profile (what it shows depends on individual's account security settings).
Session IDs look random, they have some similar parts but their value changes unpredictably each login even in concurrent sessions.

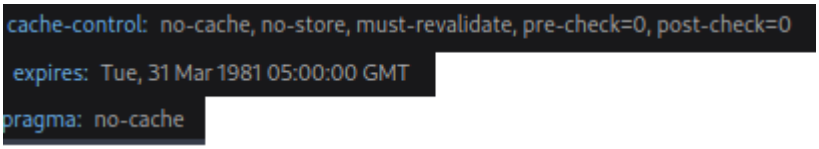
4.4.5 Testing for Vulnerable Remember Password

The login page does not offer 'remember password' functionality and it does not store user's credentials instead replacing them with secure tokens like sessionID.

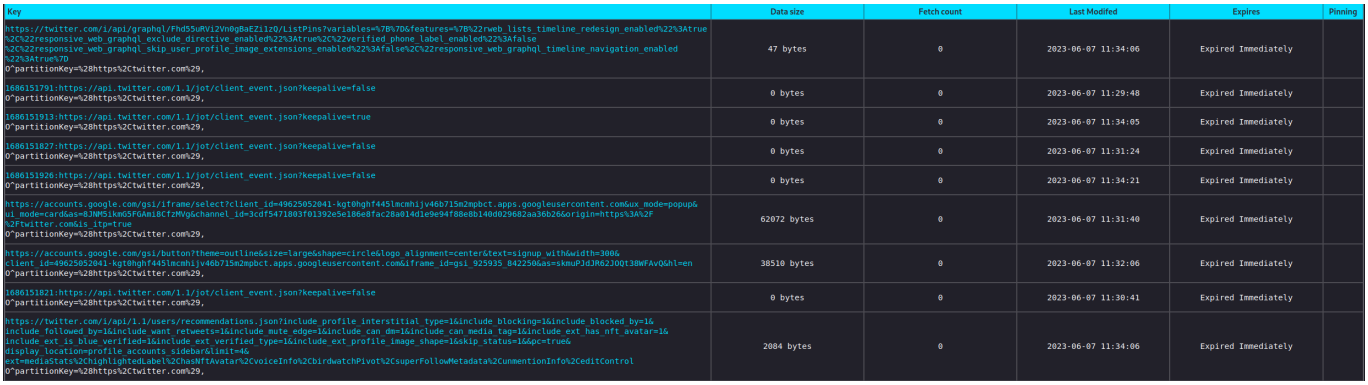


4.4.6 Testing for Browser Cache Weaknesses

Attacker can not access private information with the 'History' function and 'Back' button, he gets redirected to secure version of that page where he acts as a guest account or to login page to get authorized.
The website uses HTTP directives to secure cache correctly with the following settings (copied from response headers).



Twitter stores a lot of information in cache, it does so securely and no information can be gathered from the cache. The cache can be viewed by using URL `about:cache` on Firefox.

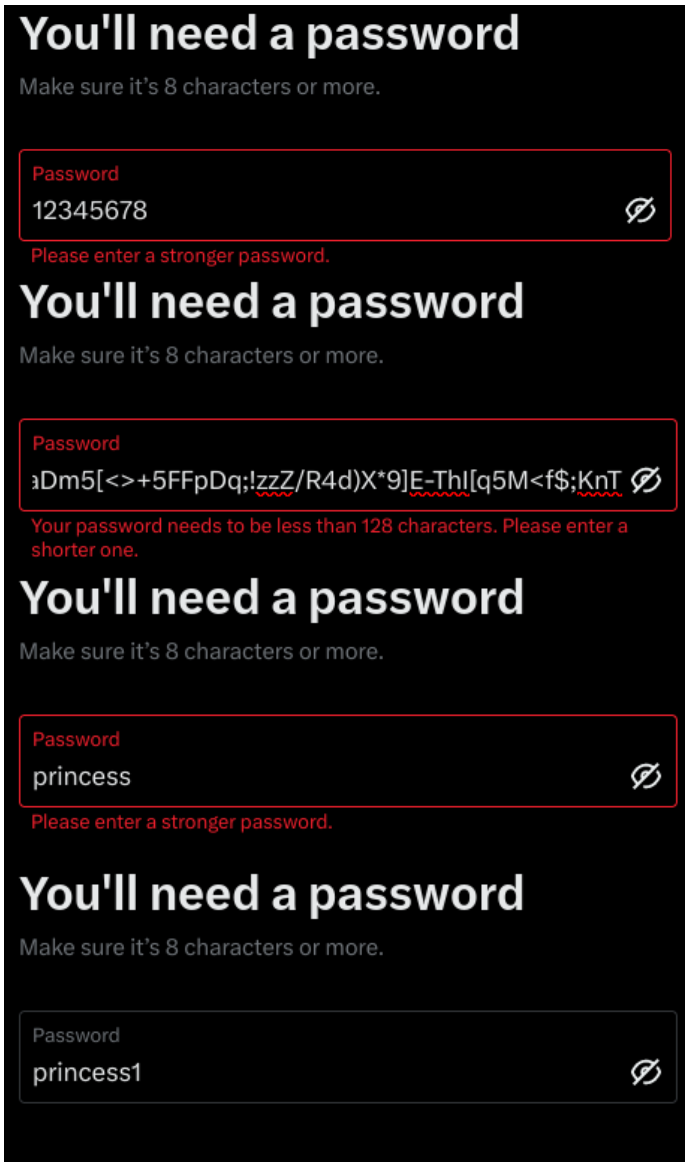


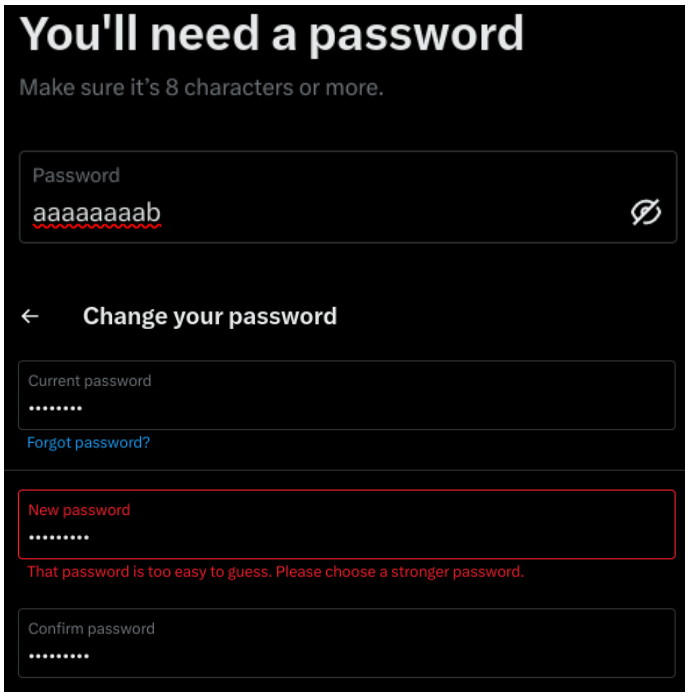
4.4.7 Testing for Weak Password Policy

Password policy allows any type of character found on standard keyboard. It requires at least 8 characters and accepts 128 most. It does not force numbers or special characters but a short password without those will be marked as weak and not accepted. The passwords are checked against common passwords so passwords like `princess` or `password` or `iloveyou` will not be allowed, however adding a number to them will 'fix' this so passwords like `princess1` or `password1234` work. General password entropy is not calculated as passwords like `aaaaaaaab` are also accepted. During registration user can choose password same as his chosen name but

if he tries to change a password to his username later via the 'Change your password' function as logged user password with his username will be rejected (part of it with numbers is okay though). Username is not rejected during registration before user picks his username after creating password. At least for consistency stake it would be best to choose username and then password so it can be checked like in change password functionality. It is not required to confirm password which made me accidentaly enter and submit password I was not yet intending to use and therefore did not know it when trying to log in again forcing me to reset password.

User is not forced to change his password after any period of time and he can change his password whenever he pleases with different methods. Change password function requires previous password knowledge. The changed passwords can be very similar to previous, it is not allowed to change to same password though. Also the warning messages are different from those seen during registration, if we use common password (princess/password) we are presented with `That password is too easy to guess. Please choose a stronger password.` while in registration the message was just `Please enter a stronger password`. Registration phase allows `show password` function while `change password` function does not.





As mentioned before twitter does not force pretty much any requirements for the password although they provide tips for password strenght in their Help center

Password strength

Create a strong and unique password for your Twitter account. You should also create an equally strong and unique password for the email address associated with your Twitter account.

Do's:

- **Do** create a password at least 10 characters long. Longer is better.
- **Do** use a mix of uppercase, lowercase, numbers, and symbols.
- **Do** use a different password for each website you visit.
- **Do** keep your password in a safe place. Consider using password management software to store all of your login information securely.

Don'ts:

- **Do not** use personal information in your password such as phone numbers, birthdays, etc.
- **Do not** use common dictionary words such as "password", "iloveyou", etc.
- **Do not** use sequences such as "abcd1234", or keyboard sequences like "qwerty."
- **Do not** reuse passwords across websites. Your Twitter account password should be unique to Twitter.

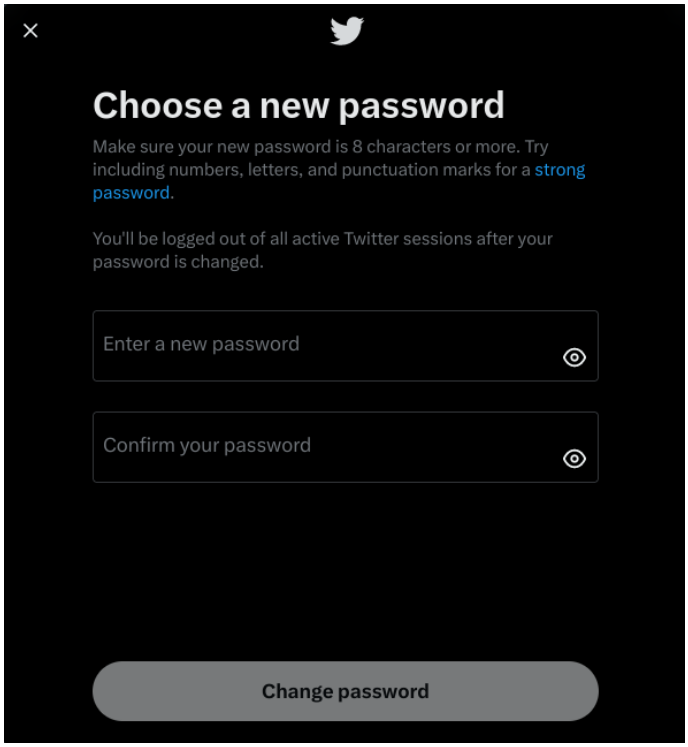
4.4.8 Testing for Weak Security Question Answer

Twitter does not use security question functionality.

4.4.9 Testing for Weak Password Change or Reset Functionalities

Logged in users can change their passwords with method described under Testing for Weak Password Policy using their current password.
If user is not logged in he can reset his password via the `Forgot password?` functionality. If unusual activity is detected user is prompted to input their username (As described in above subsections), meaning password of disclosed users (for example previously found admin@twitter.com) cannot be reset. Otherwise user is asked to select email to send verification code to (the code seems to be 8 lenght randomly generated digit-letter

sequence). After inputing the correct code user can change his password. The password policy to new password as well as warning messages are the same as when registrating with new password, this time however it is reqiuired to repeat the password. When the password is reset user is prompted to enter reason for changing password and is logged out of all active Twitter sessions, as stated in Twitter's Help Center.



4.4.10 Testing for Weaker Authentication in Alternative Channel

No alternative channels are in our scope (mobile/desktop app).

4.5 Authorization Testing

4.5.1 Testing Directory Traversal File Include

Twitter does not use GET arguments to load files, arguments are rather used to specify settings. Values in cookies are encoded and most likely are not used for dynamic page generation. Attack is therefore not possible.

```
GET /i/api/graphql/eB2SB0qYfesXNLf9joi-_w/HomeTimeline?variables=
%7B%22count%22%3A40%2C%22cursor%22%3A%22DAABCgABFypRl9DAJxEKAAIXKkvobtfAAAgAAwAAAAEAAA%22%2C%
22includePromotedContent%22%3Atrue%2C%22latestControlAvailable%22%3Atrue%2C%22withCommunity%2
2%3Atrue%7D&features=
%7B%22rweb_lists_timeline_redesign_enabled%22%3Atrue%2C%22responsive_web_graphql_exclude_dire
ctive_enabled%22%3Atrue%2C%22verified_phone_label_enabled%22%3Afalse%2C%22creator_subscriptio
ns_tweet_preview_api_enabled%22%3Atrue%2C%22responsive_web_graphql_timeline_navigation_enable
d%22%3Atrue%2C%22responsive_web_graphql_skip_user_profile_image_extensions_enabled%22%3Afalse
%2C%22tweetypie_unmention_optimization_enabled%22%3Atrue%2C%22responsive_web_edit_tweet_api_e
nabled%22%3Atrue%2C%22graphql_is_translatable_rweb_tweet_is_translatable_enabled%22%3Atrue%2C
%22view_counts_everywhere_api_enabled%22%3Atrue%2C%22longform_notetweets_consumption_enabled%
22%3Atrue%2C%22tweet_awards_web_tipping_enabled%22%3Afalse%2C%22freedom_of_speech_not_reach_f
etch_enabled%22%3Atrue%2C%22standardized_nudges_misinfo%22%3Atrue%2C%22tweet_with_visibility_
results_prefer_gql_limited_actions_policy_enabled%22%3Afalse%2C%22longform_notetweets_rich_te
xt_read_enabled%22%3Atrue%2C%22longform_notetweets_inline_media_enabled%22%3Afalse%2C%22respo
nsive_web_enhance_cards_enabled%22%3Afalse%7D HTTP/2
Host: twitter.com
Cookie: guest_id=v1%3A168675853719215783; _twitter_sess=
BAh7CSIKZmxhc2hJQzonQWN0aW9uQ29udHJvbGxlcjo6Rmxhc2g6OkZsYXNo%250ASGFzaHsABjoKQHVzZWR7ADoPY3Jl
YXRlZF9hdGwrCLgVpbqIAToMY3NyZl9p%250AZCIlODRkODZmNTJhODlmYzc4ZDYwOWZjMzQ0NDE3MTQ5OGY6B2lkIiU0
OWM3%250ANDkyYmFjNzdmZTc3NDJmYTQyYzcyNDlkYzM4NW%253D%253D--078e2d234a4f3f009ddb03c62f1b74db04
0a7afd; kdt=yKkQnd8awignI4FcE5SRKRDK6tRQNH4RwbZCX4J2; auth_token=
4d382f3986e21664398797e5aaebe05d88a610e7; ct0=
350abbc32a6506baac88efb25db53ca3259f201e2d585f23624c054e109358204c2e40cff0c0e7a5910573384fd2b
1dc6057d23fc82c622f21fa56fe4baadb4586c8f48e283deb8bab428e88ac3b07ef; att=
1-siqVRqIRfyJ79k8v3HAjLNPbWWD3MKauRWRS8GWx; lang=en; twid=u%3D1666898792648515584; d_prefs=
MjoxLGNvbnNlbnRfdmVyc2lvbjoyLHRleHRfdmVyc2lvbjoxMDAw
```

4.5.2 Testing for Bypassing Authorization Schema

As we are again testing in black-box setting, we can only test with user privileges/role. For the same reason we can only test for horizontal bypassing.
Twitter does not identify users by parameter but rather with combination of the right cookie + csrf token and header. So by changing the cookie of one users request with cookie from others gets the response 403 Forbidden and error message with code 353: `This request requires a matching csrf cookie and header.`
By changing both csrf token and cookie we can get the response of other user but by that we authorized ourselves as that user so that is correct behaviour (getting this information would require to catch communication and decode cookie and csrf token). Horizontal bypassing is not possible.

4.5.3 Testing for Privilege Escalation

Privileges on twitter website are most likely stored in the `auth_token` part of cookie as a hash. No other variables store or allow manipulation in HTTP communication.
No file with missing or bad authorization checking were found by URL traversal.

4.5.4 Testing for Insecure Direct Object References

Again twitter loads page contents via POST/GET graphQL requests (`POST /i/api/graphql/eB2SB0qYfesXNLf9joi-_w/HomeTimeline`) to api that return the needed content so there is no possible input for object/content manipulation.

4.6 Session Management Testing

4.6.1 Testing for Session Management Schema

Name	Domain	Expires in	HttpOnly	Secure	SameSite	Note
_gid	.twitter.com	28 hours	No	No		Google Analytics
IDE	.doubleclick.net	1 year	Yes	Yes	None	Doubleclick integration and/or analytics
_ga_	.twitter.com	1 year	No	No		Google Analytics
att	.twitter.com	1 day	Yes	Yes	None	Authentication and password reset
guest_id	.twitter.com	1 year	No	Yes	None	Advertising when logged out
twid	.twitter.com	1 year	No	Yes	None	Authentication
auth_token	.twitter.com	1 year	Yes	Yes	None	Account login and authentication
ct0	.twitter.com	1 year	No	Yes	Lax	Authentication
lang	twitter.com	Session	No	No		Language functionality
kdt	.twitter.com	1 year	Yes	Yes		Authentication of a known device
d_prefs	.twitter.com	6 months	No	Yes		Cookie preferences
_twitter_sess_	.twitter.com	Session	Yes	Yes		User session
guest_id_marketing	.twitter.com	1 year	No	Yes	None	Advertising when logged out
guest_id_ads	.twitter.com	1 year	No	Yes	None	Advertising when logged out

personalization_id	.twitter.com	1 year	No	Yes	None	Tracking activities on and off Twitter for personalization

4.6.2 Testing for Cookies Attributes

Attribute secure is used for all authentication related cookies. However few authentication cookies have the attribute SameSite set to None which may lead to cookie hijacking in case of CSRF/XSS vulnerability.

4.6.3 Testing for Session Fixation

The cookie _twitter_sess is set before the authentication and it remains the same after successful authentication. When that _twitter_sess value is set to another user it doesn't make them authenticated as the original one, probably because there are many cookies used in combination for the authentication.

4.6.4 Testing for Exposed Session Variables

Session cookie has its own different value for every session. Protection from eavesdropping is ensured by HTTPS protocol.

When the _twitter_sess cookie is set, it comes with cache-control directives (apart from others) `no-cache` and `no-store` which prevent it from being cached.

```
HTTP/2 200 OK
Date: Sat, 17 Jun 2023 21:30:04 GMT
Perf: 7626143928
Pragma: no-cache
Server: tsa_o
Status: 200 OK
Expires: Tue, 31 Mar 1981 05:00:00 GMT
Set-Cookie: fm=0; Max-Age=0; Expires=Sat, 17 Jun 2023 21:30:04 GMT; Path=/; Domain=.twitter.com; Secure; HTTPOnly
Set-Cookie: _twitter_sess=BAh7CSIKZmxhc2hJQzonQWN0aW9uQ29udHJvbGx1cjo6Rmxhc2g6OkZsYXNo%250ASGFzaHsABjoKQHVzZWR7ADoPY3J1YXR1ZF9hdGwrCG7mQ8uIAToMY3NyZ19p%250AZCI1ZmE0MmIyYTc4ZTd1NjJiNjc5M2UxNTEyYTQwOGFmNzk6B21kIiUyOD1i%250AODA1Y2M5Y2EwZDRiMWE5MjU4MTc3MWZiYzE4Ng%253D%253D--129fc5e110bbe1139ccfc8e38c20b86031af5857; Path=/; Domain=.twitter.com; Secure; HTTPOnly
Content-Type: text/javascript; charset=utf-8
Cache-Control: no-cache, no-store, must-revalidate, pre-check=0, post-check=0
```

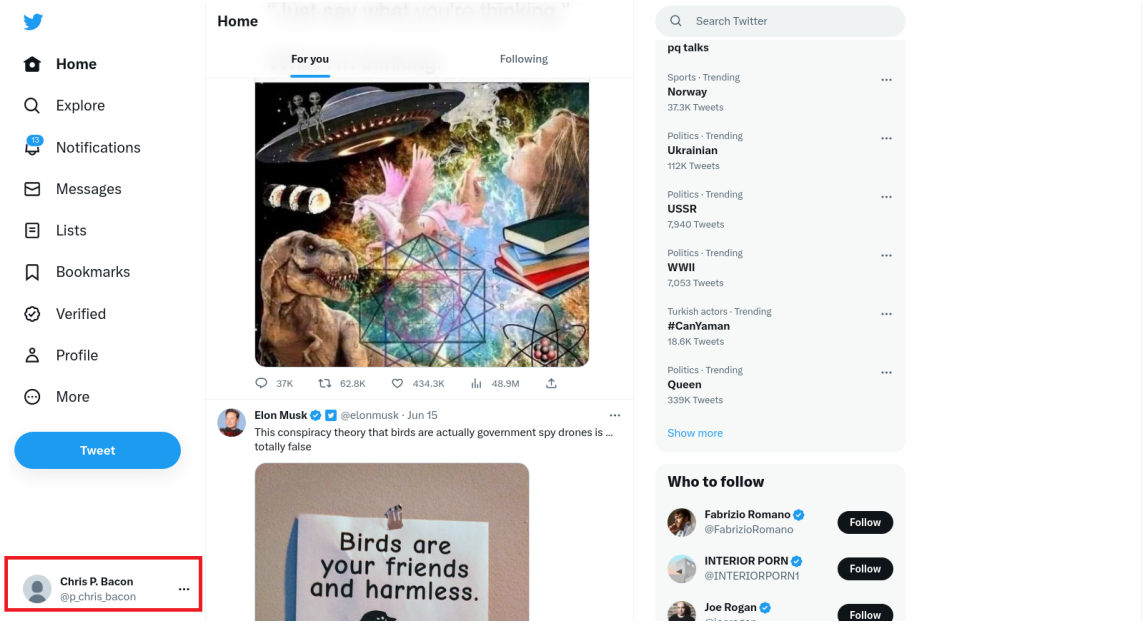
After reviewing the client-server communication the session cookie seems to be contained only in POST requests.

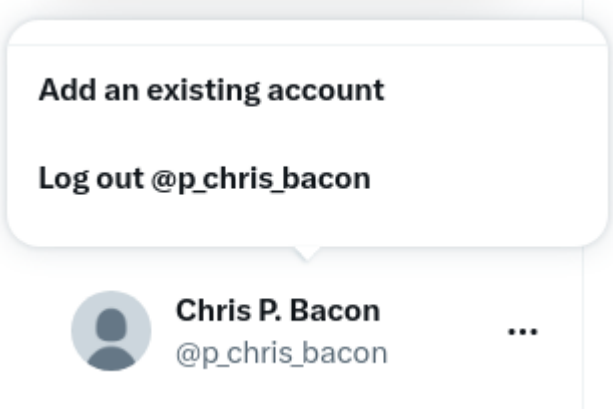
4.6.5 Testing for Cross Site Request Forgery

CSRF Token is included in the _twitter_sess cookie and also as a HTTP header X-Csrf-Token in requests, which prevents the application from being vulnerable to Cross Site Request Forgery.

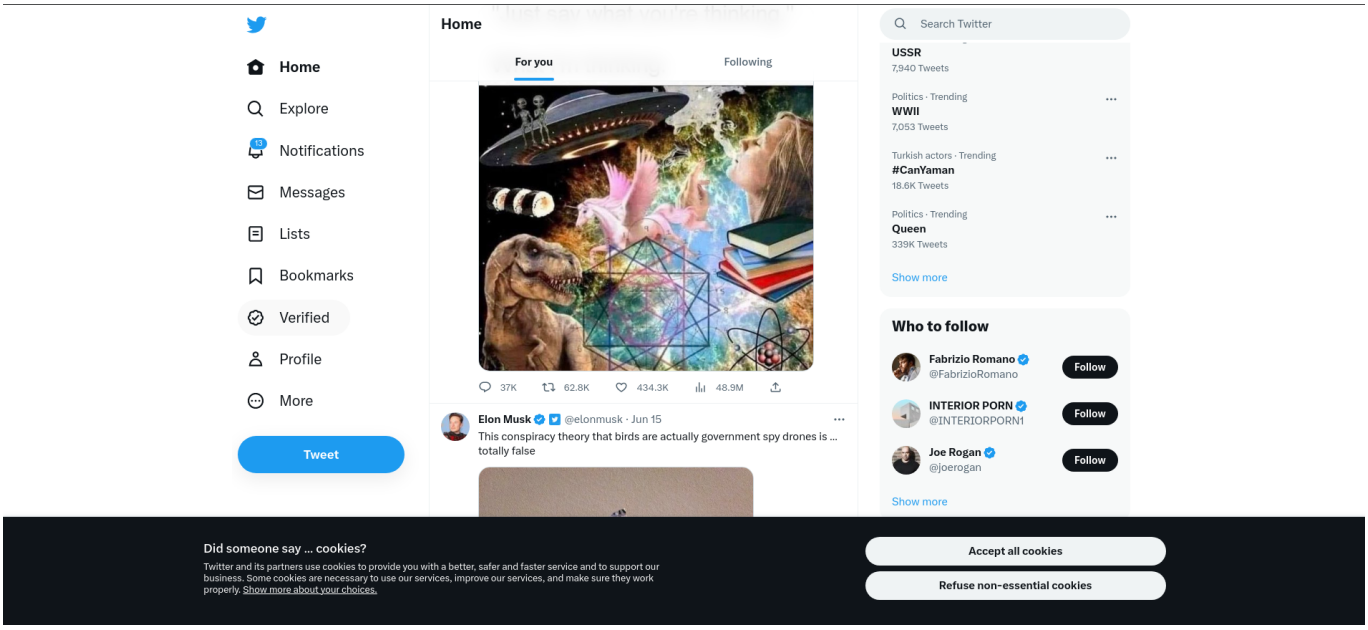
4.6.6 Testing for Logout Functionality

The logout button is placed in an area of the page that is fixed in the view port of the browser and isn't affected by scrolling of the content. However the logout button cannot be seen directly as user first has to click on the three dots next to his account name and then choose the logout button from the menu that pops out.

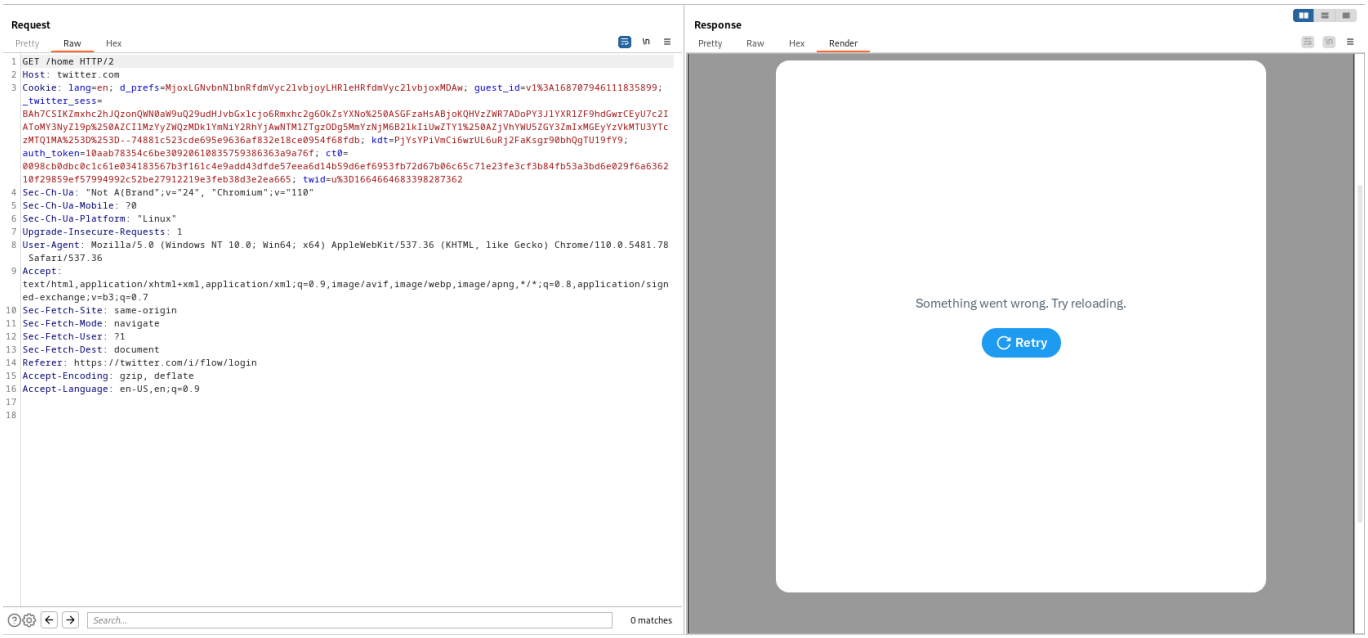




Note that user cannot log out within the user interface before they answer the cookie related dialog.



If one is logged out and tries to load a page that is only available to authenticated users with their old session cookies, the application sends a response that something went wrong.



4.6.7 Testing Session Timeout

During the testing process I haven't experienced a single timeout, therefore the only time limit appears to be the expiry date of cookies.

4.6.8 Testing for Session Puzzling

All session variables seem to be correctly regenerated for every session and no session puzzling was detected.

4.6.9 Testing for Session Hijacking

The application runs on HTTPS, uses HSTS and all cookies related to session or authentication are marked as secure.

4.7 Input Validation Testing

4.7.1 Testing for Reflected Cross Site Scripting

None of the input vectors (POST/GET parameters) are reflected in response or generate an output that would suggest XSS vulnerability.

4.7.2 Testing for Stored Cross Site Scripting

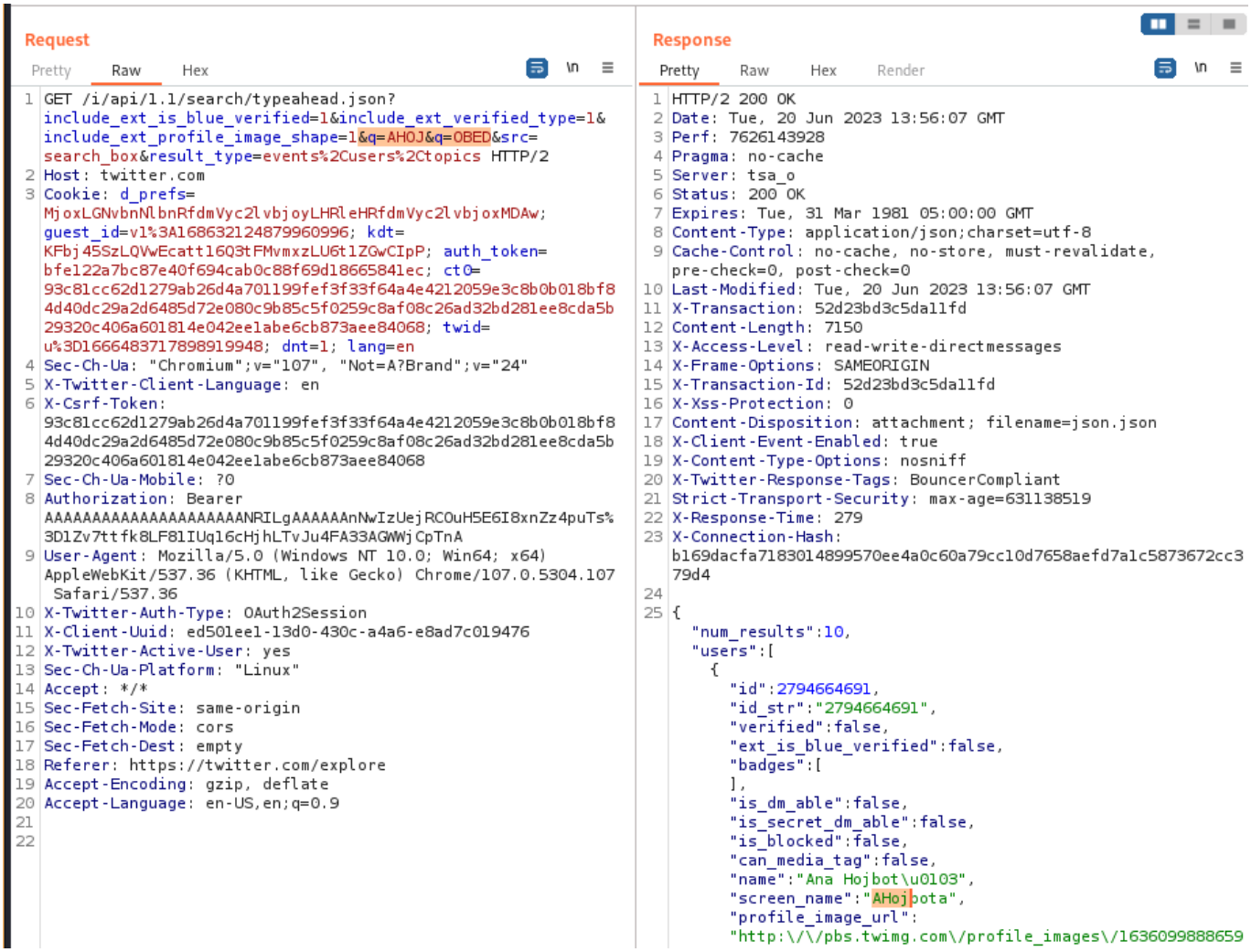
Here the input vector is obviously the "tweet" mechanism, where user can input anything and it is stored in the backend and can be displayed by any user. Another possible vector is comments below "tweets". After testing these vectors backend seems to validate and sanitize input very well and no issue was encountered on Twitter's part. Another problem is once again Twitter's obfuscation of its functionality.

4.7.3 Testing for HTTP Verb Tampering

Discussed in 4.2.6 Test HTTP Methods.

4.7.4 Testing for HTTP Parametr Pollution

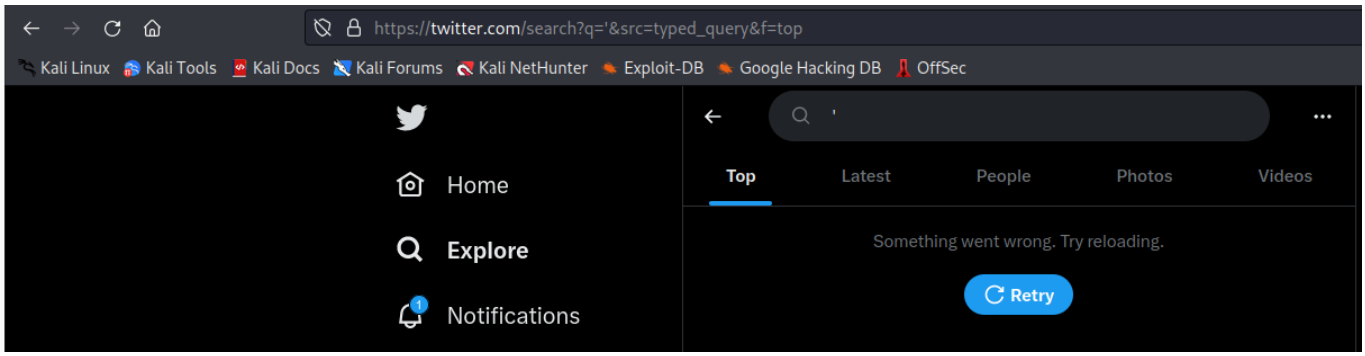
Search function uses GET parameters. Using multiple same name parameters returns results for only the first one.



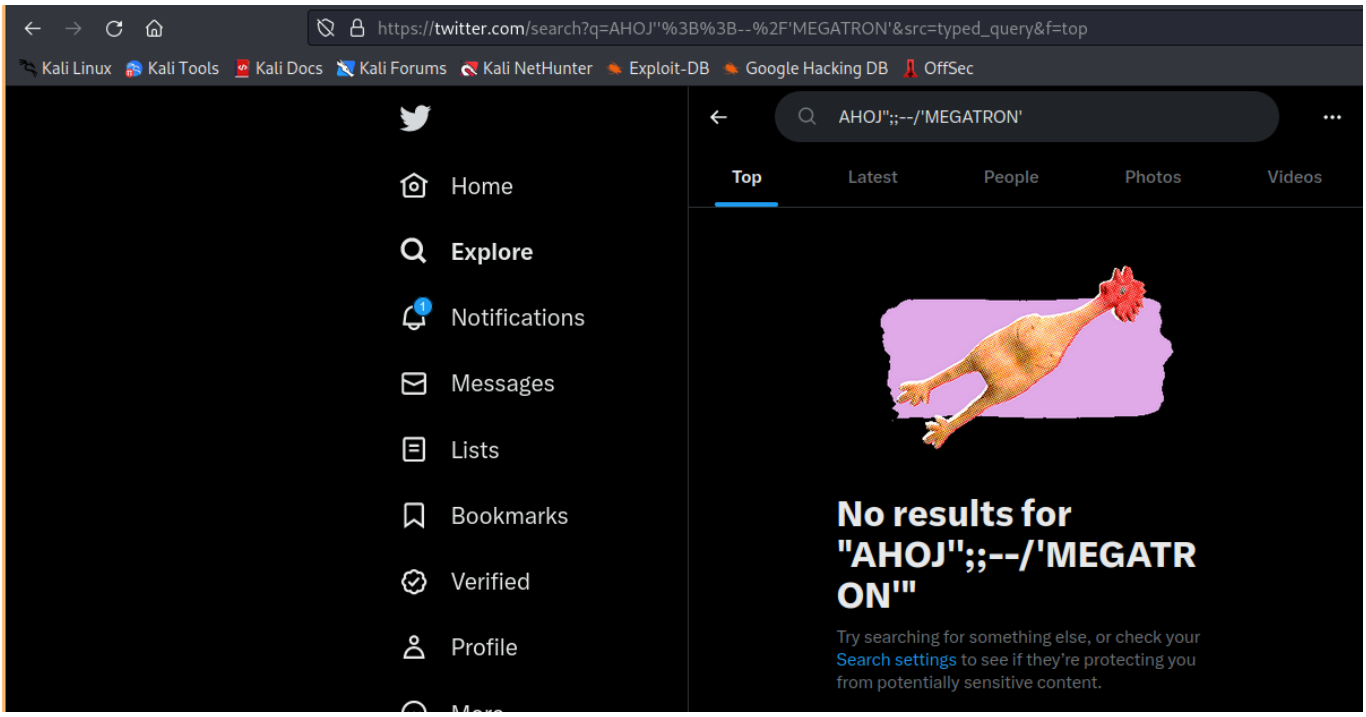
No vulnerability found.

4.7.5 Testing for SQL Injection

Input vector for SQL injection could again be the search function, that might communicate with database through API. Trying some classic SQL malicious code like different types of comments (since we don't know what database twitter is running on) did not yield any suspicious result. In fact testing right in the browser shows that any special symbol `@#$%^-'";:` on its own returns "Something went wrong".



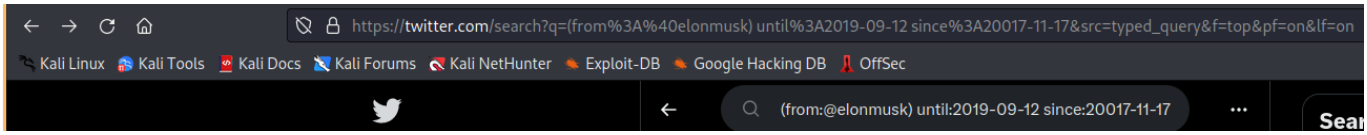
The following screenshot shows that the function sanitizes the input correctly and no unintended behaviour was found.



Once again testing in black-box setting with little to none information about how the back-end, front-end and API communication is realised and what exactly each layer does proves to be the biggest obstacle in finding possible vulnerabilities.

4.7.6 Testing for LDAP Injection

Twitter probably does not use LDAP, if so, it does so correctly. When applying search filters the request looks like this:



4.7.7 Testing for XML Injection

Twitter does not use XML style communication.

4.7.8 Testing for SSI Injection

We could not determine if SSI is enabled since we don't have access to web server and no `.shtml` pages were found.

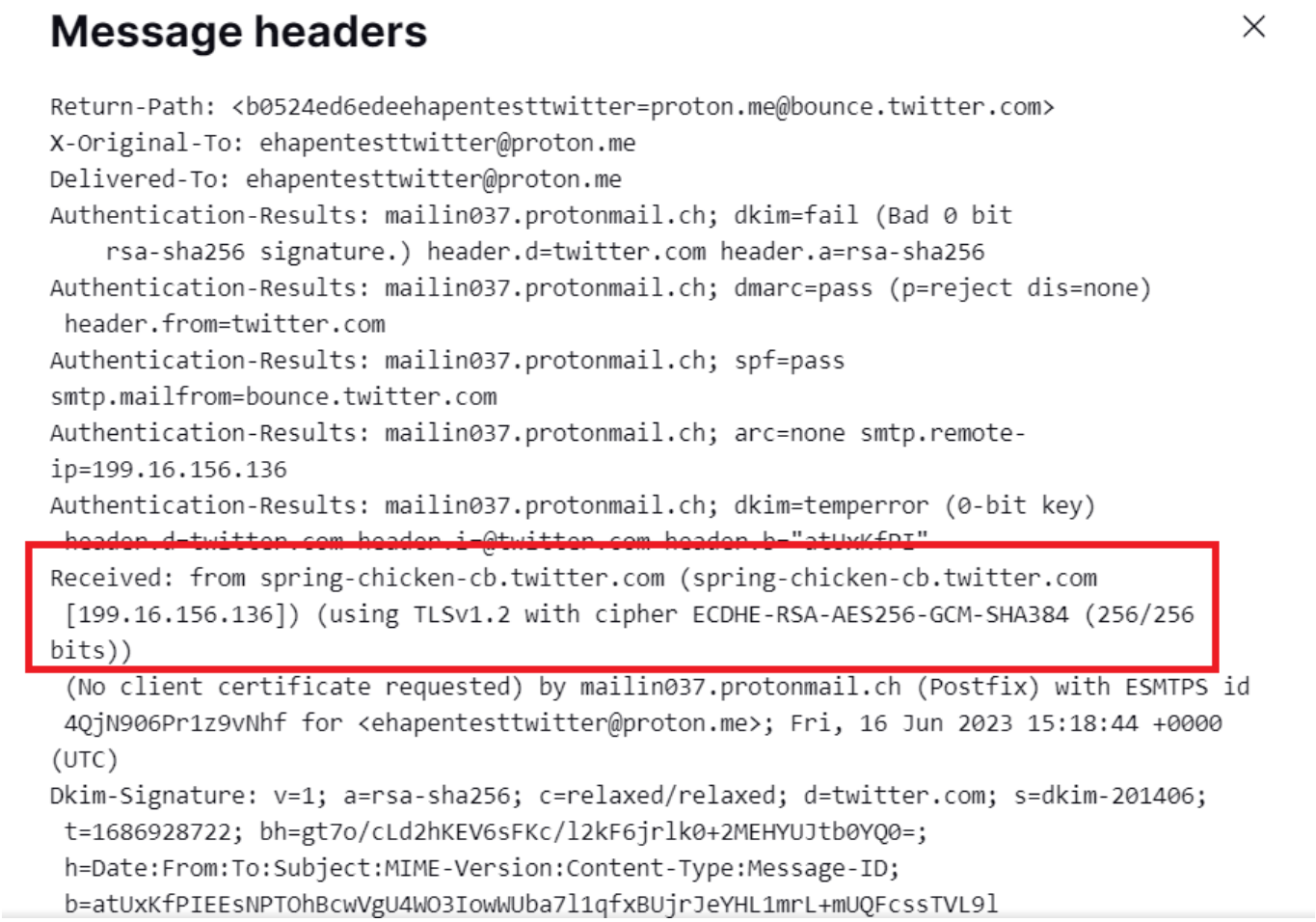
4.7.9 Testing for XPath Injection

Since Twitter does not use XML documents XPath injection is not possible as well.

4.7.10 Testing for IMAP SMTP Injection

Twitter does not provide any public mail service.

When creating a user account, we received an email with verification code from verify@twitter.com. In that email's headers was stated that it was received from spring-chicken-cb.twitter.com [199.16.156.136], but running `nmap` on it didn't reveal any open ports.



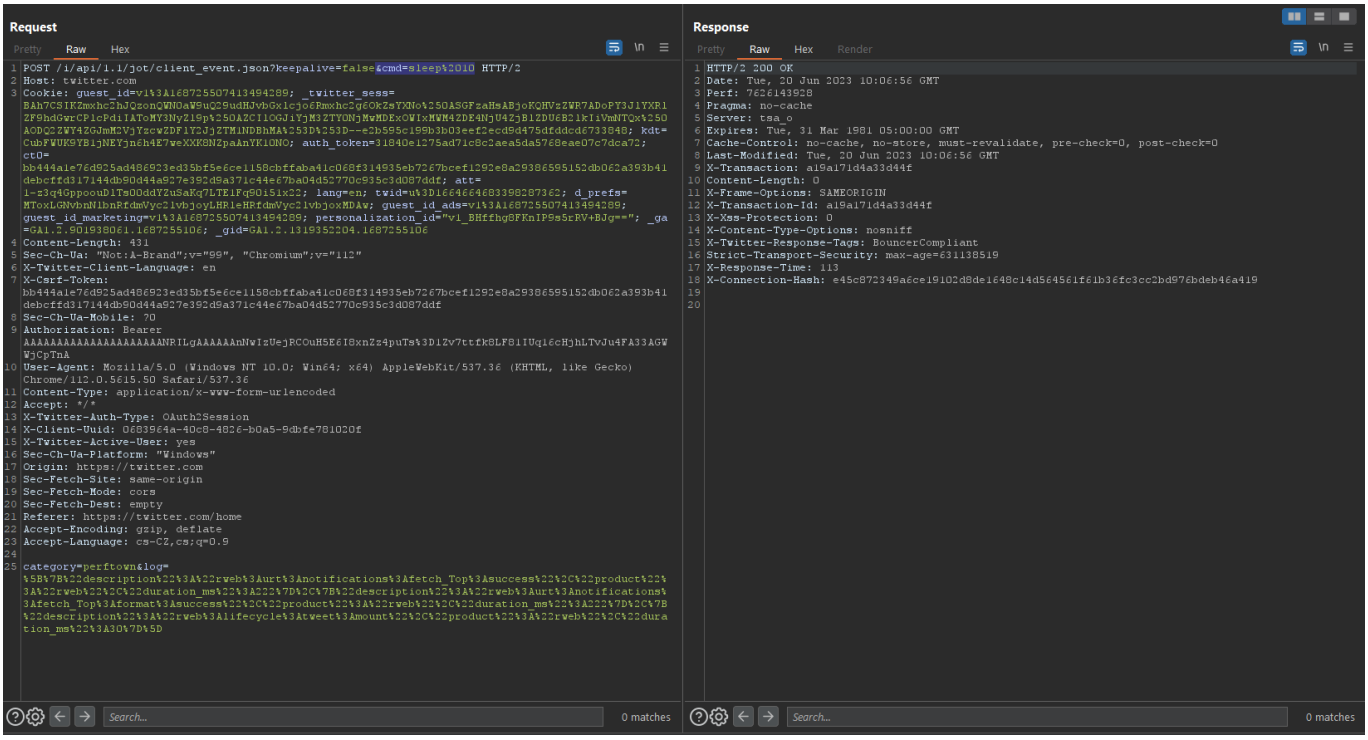
Message headers

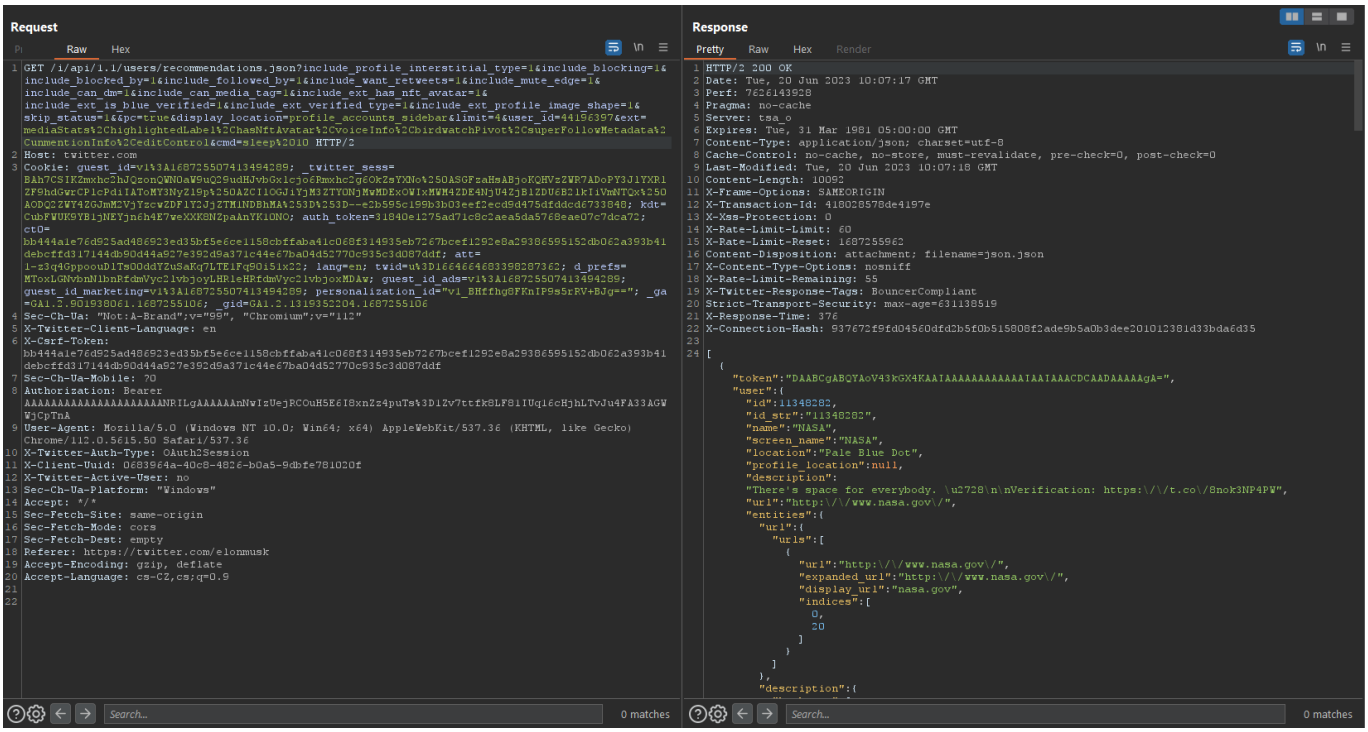
```
Return-Path: <b0524ed6edeehapentesttwitter=proton.me@bounce.twitter.com>
X-Original-To: ehapentesttwitter@proton.me
Delivered-To: ehapentesttwitter@proton.me
Authentication-Results: mailin037.protonmail.ch; dkim=fail (Bad 0 bit
    rsa-sha256 signature.) header.d=twitter.com header.a=rsa-sha256
Authentication-Results: mailin037.protonmail.ch; dmarc=pass (p=reject dis=none)
 header.from=twitter.com
Authentication-Results: mailin037.protonmail.ch; spf=pass
smtp.mailfrom=bounce.twitter.com
Authentication-Results: mailin037.protonmail.ch; arc=none smtp.remote-
ip=199.16.156.136
Authentication-Results: mailin037.protonmail.ch; dkim=temperror (0-bit key)
 header.d=twitter.com header.i=@twitter.com header.b="atUxKfPI"
Received: from spring-chicken-cb.twitter.com (spring-chicken-cb.twitter.com
 [199.16.156.136]) (using TLSv1.2 with cipher ECDHE-RSA-AES256-GCM-SHA384 (256/256
bits))
 (No client certificate requested) by mailin037.protonmail.ch (Postfix) with ESMTPS id
 4QjN906Pr1z9vNhf for <ehapentesttwitter@proton.me>; Fri, 16 Jun 2023 15:18:44 +0000
(UTC)
Dkim-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed; d=twitter.com; s=dkim-201406;
 t=1686928722; bh=gt7o/cLd2hKEV6sFKc/l2kF6jrlk0+2MEHYUJtb0YQ0=;
 h=Date:From:To:Subject:MIME-Version:Content-Type:Message-ID;
 b=atUxKfPIEEsNPTOhBcwVgU4WO3IowWUba7l1qfxBUjrJeYHL1mrL+mUQFcssTVL9l
```

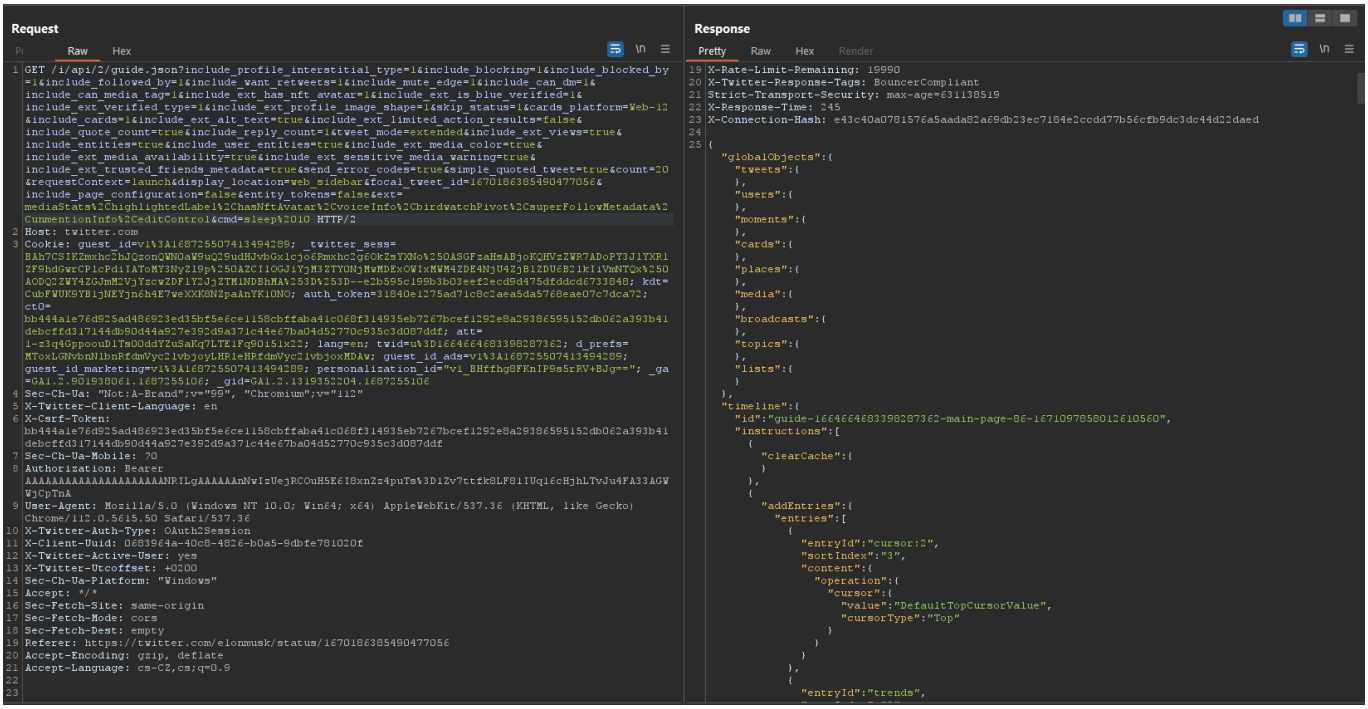
```
┌──(kali㉿kali)-[~]
└─$ nmap -sV -Pn --top-ports 1000 199.16.156.136
Starting Nmap 7.93 ( https://nmap.org ) at 2023-06-20 05:25 EDT
Nmap scan report for spring-chicken-cb.twitter.com (199.16.156.136)
Host is up.
All 1000 scanned ports on spring-chicken-cb.twitter.com (199.16.156.136) are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 202.40 seconds
```

4.7.11 Testing for Code Injection

Command parameters were injected into various URLs, but none of them worked.







4.7.11.1 Testing for Local File Inclusion

No requests or scripts that take filename as a parameter were found.

4.7.11.2 Testing for Remote File Inclusion

No requests or scripts that take filename as a parameter were found.

4.7.12 Testing for Command Injection

No place for command injection was found.

4.7.13 Testing for Format String Injection

We tried to insert formatting strings into many different places but none of them caused unexpected behaviour.

4.7.14 Testing for Incubated Vulnerability

We didn't find any vulnerability suitable for incubation.

4.7.15 Testing for HTTP Splitting Smuggling

We didn't find any data that were contained in the request and then would appear in the reponse header.

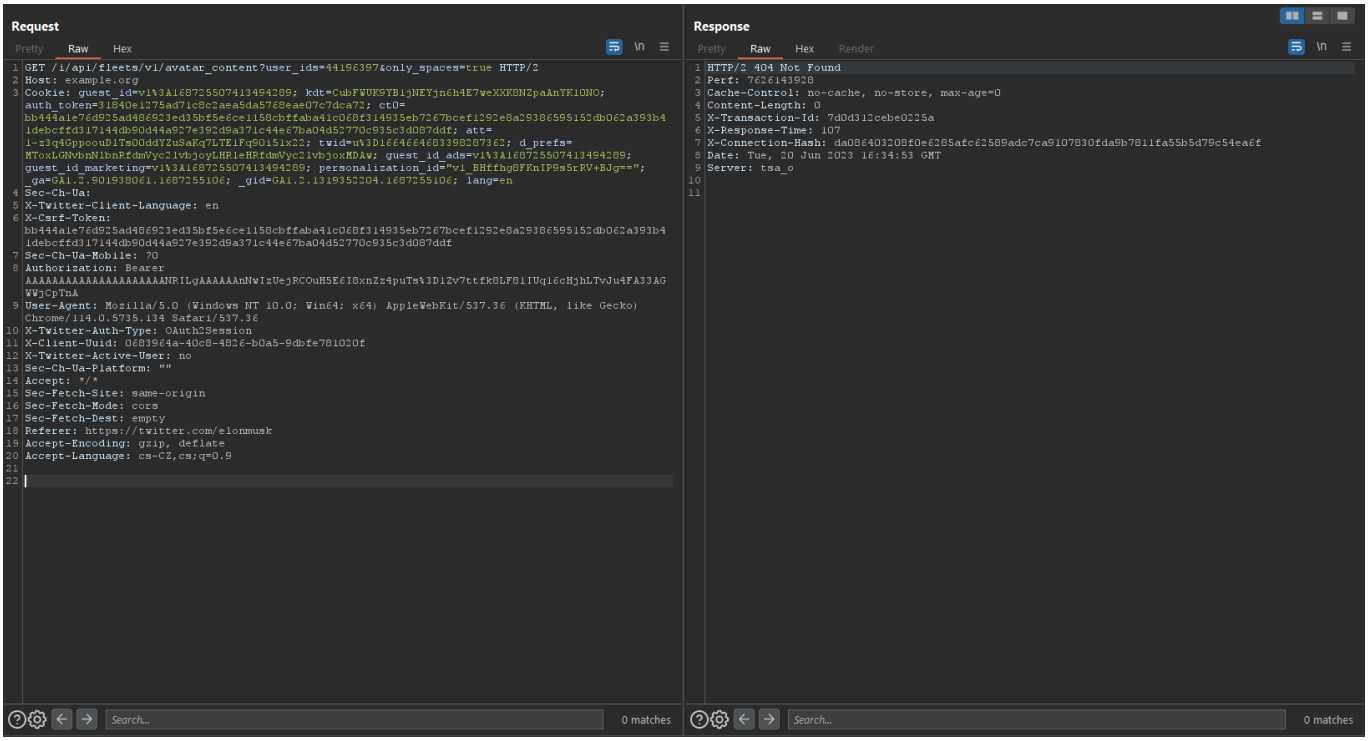
4.7.16 Testing for HTTP Incoming Requests

We don't have access to the server, hence we cannot setup a reverse proxy on the server side.

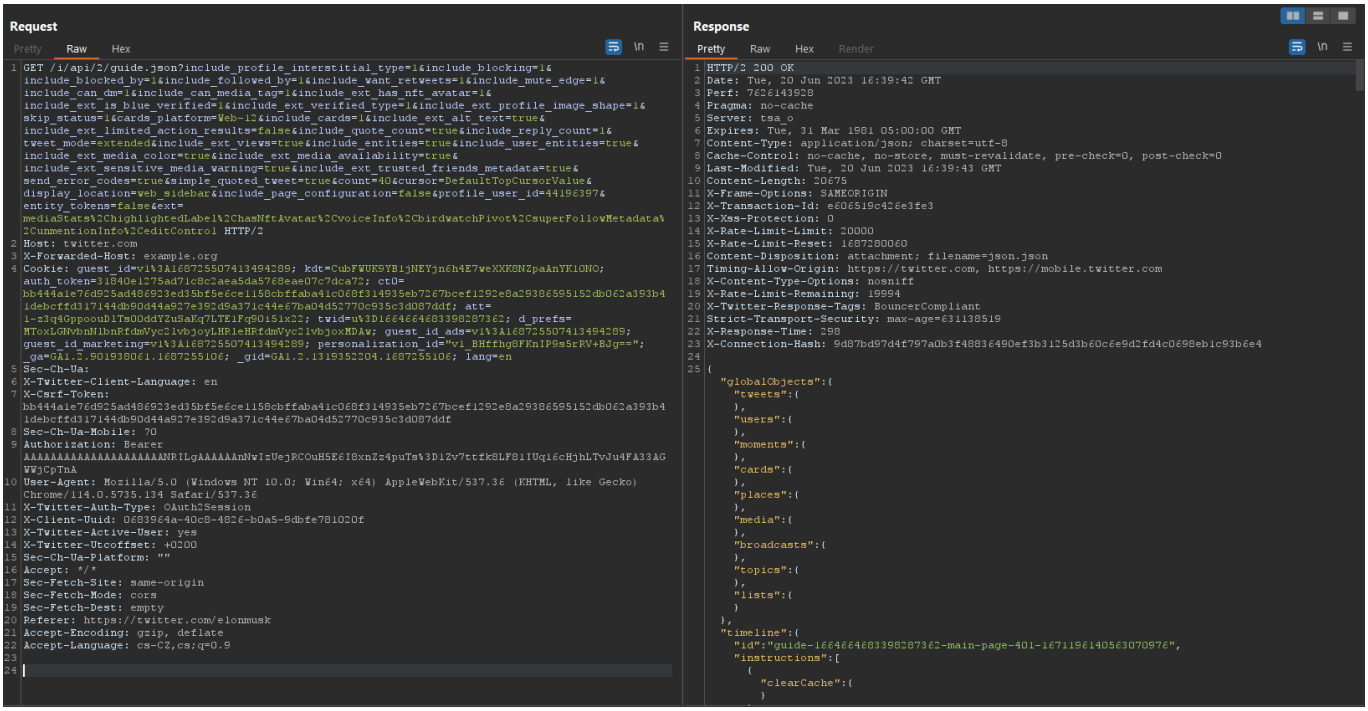
When reviewing requests and responses on the client side, none of them seemed suspicious.

4.7.17 Testing for Host Header Injection

When supplied another domain into the Host header field, the server responds with 404 Not Found.



We didn't find any request that would contain the X-Forwarded-Host header. When we added the header ourselves, the server seemed to not process that header at all.



4.7.18 Testing for Server-side Template Injection

No Server-side Template Injection vulnerabilities were found.

4.7.19 Testing for Server-side Request Forgery

No Server-side Request Forgery injection points were found.

4.8 Testing for Error Handling

4.8.1 Testing for Improper Error Handling

When testing for possible attack vectors we found out that the page responds correctly (with 404 (Not found) or 401 (Unauthorized)) when sending GET/POST requests to random files or files higher in the hierarchy. Trying different requests (HEAD/DELETE...) returns 405 (Method not allowed). When trying a really long path we recieve `RST_STREAM error 0x2 (Implementation fault)` which doesn't seem to be an issue on twitter's side but isn't really documented either.
Another testing was performed by messing with POST and GET parameters. The following GET request parameter `supports_ntab_urt` returns different responses:
`GET /i/api/2/badge_count/badge_count.json?supports_ntab_urt=1 HTTP/2`

- 1 / true -> OK 200
- 2 / abcd -> BAD REQUEST 400 (body of response: `{"errors":[{"code":358,"message":"Unable to parse supports_ntab_urt parameter."}]}`)
- 0 / false -> Internal Server Error 500 (body of response: `{"errors":[{"message":"Internal error","code":131}]}`)
  Which looks well defined and other tests did not find any unsuspected/improper behaviour.

4.9 Testing for Weak Cryptography

4.9.1 Testing for Weak Transport Layer Security

We ran `sslscan --show-sigs --verbose twitter.com`. You can see the full output in outputs/sslscan.txt.

The application only supports TLSv1.2 and TLSv1.3.

No ciphers that aren't considered secure nowadays are supported.

The SSL Certificate has signature ecdsa-with-SHA384 with key strength 128 bit, signed by DigiCert and is valid for 366 days.

```
  SSL/TLS Protocols:
SSLv2     disabled
SSLv3     disabled
TLSv1.0   disabled
TLSv1.1   disabled
TLSv1.2   enabled
TLSv1.3   enabled
```

```
  SSL Certificate:
Signature Algorithm: ecdsa-with-SHA384
ECC Curve Name:      prime256v1
ECC Key Strength:    128

Subject:  twitter.com
Altnames: DNS:twitter.com, DNS:www.twitter.com
Issuer:   DigiCert TLS Hybrid ECC SHA384 2020 CA1

Not valid before: Feb  5 00:00:00 2023 GMT
Not valid after:  Feb  5 23:59:59 2024 GMT
```

```
  Supported Server Cipher(s):
Preferred TLSv1.3  128 bits  TLS_AES_128_GCM_SHA256        Curve 25519 DHE 253
Accepted  TLSv1.3  256 bits  TLS_AES_256_GCM_SHA384        Curve 25519 DHE 253
Accepted  TLSv1.3  256 bits  TLS_CHACHA20_POLY1305_SHA256  Curve 25519 DHE 253
SSL_get_error(ssl, cipherStatus) said: 1
Preferred TLSv1.2  128 bits  ECDHE-ECDSA-AES128-GCM-SHA256 Curve 25519 DHE 253
Accepted  TLSv1.2  128 bits  ECDHE-ECDSA-AES128-SHA        Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-ECDSA-AES256-GCM-SHA384 Curve 25519 DHE 253
Accepted  TLSv1.2  256 bits  ECDHE-ECDSA-AES256-SHA        Curve 25519 DHE 253
SSL_get_error(ssl, cipherStatus) said: 1
```

4.9.2 Testing for Padding Oracle

Session and authentication related cookies are encrypted but aren't multiples of 8/16 which implies that a stream cipher was used, and therefore it doesn't meet the requirements for possibility of the Padding Oracle vulnerability.

4.9.3 Testing for Sensitive Information Sent via Unencrypted Channels

All data is sent via HTTPS.

4.9.4 Testing for Weak Encryption

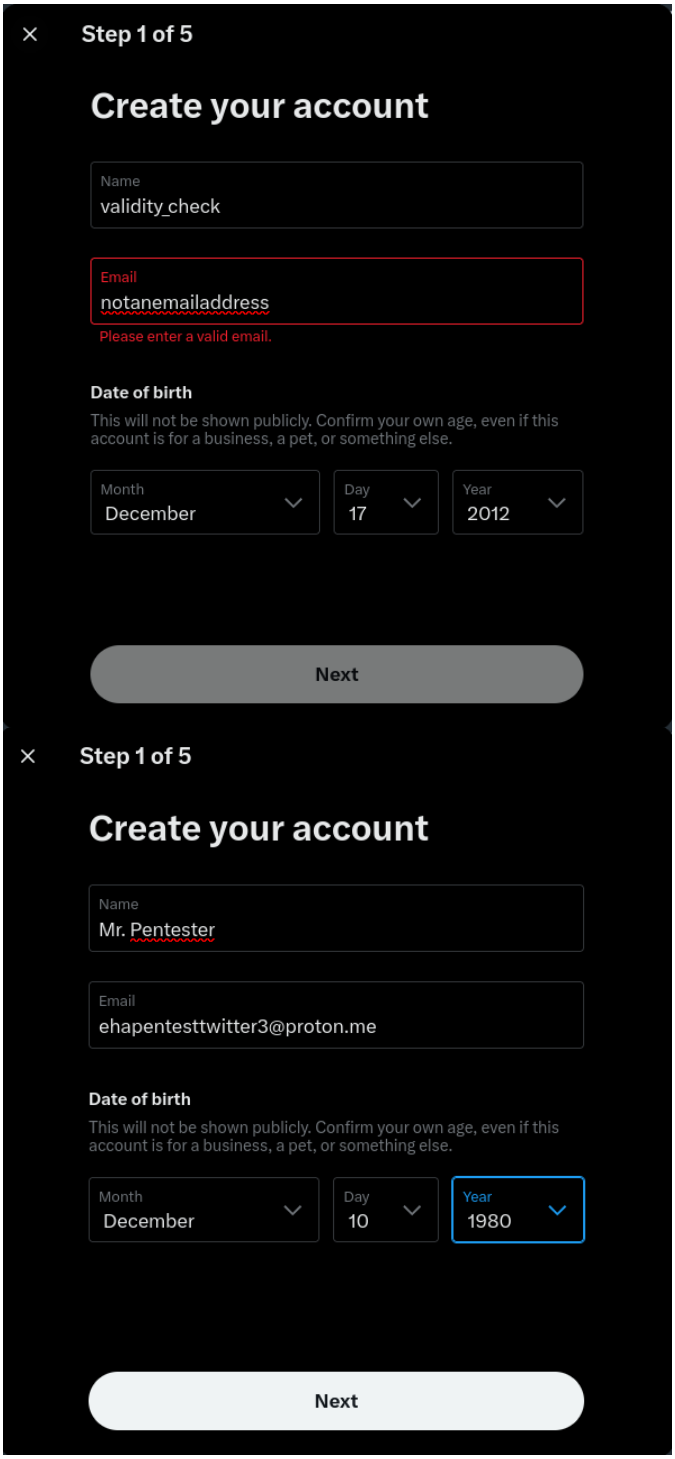
`sslscan` didn't find any weak encryption on the transport layer.

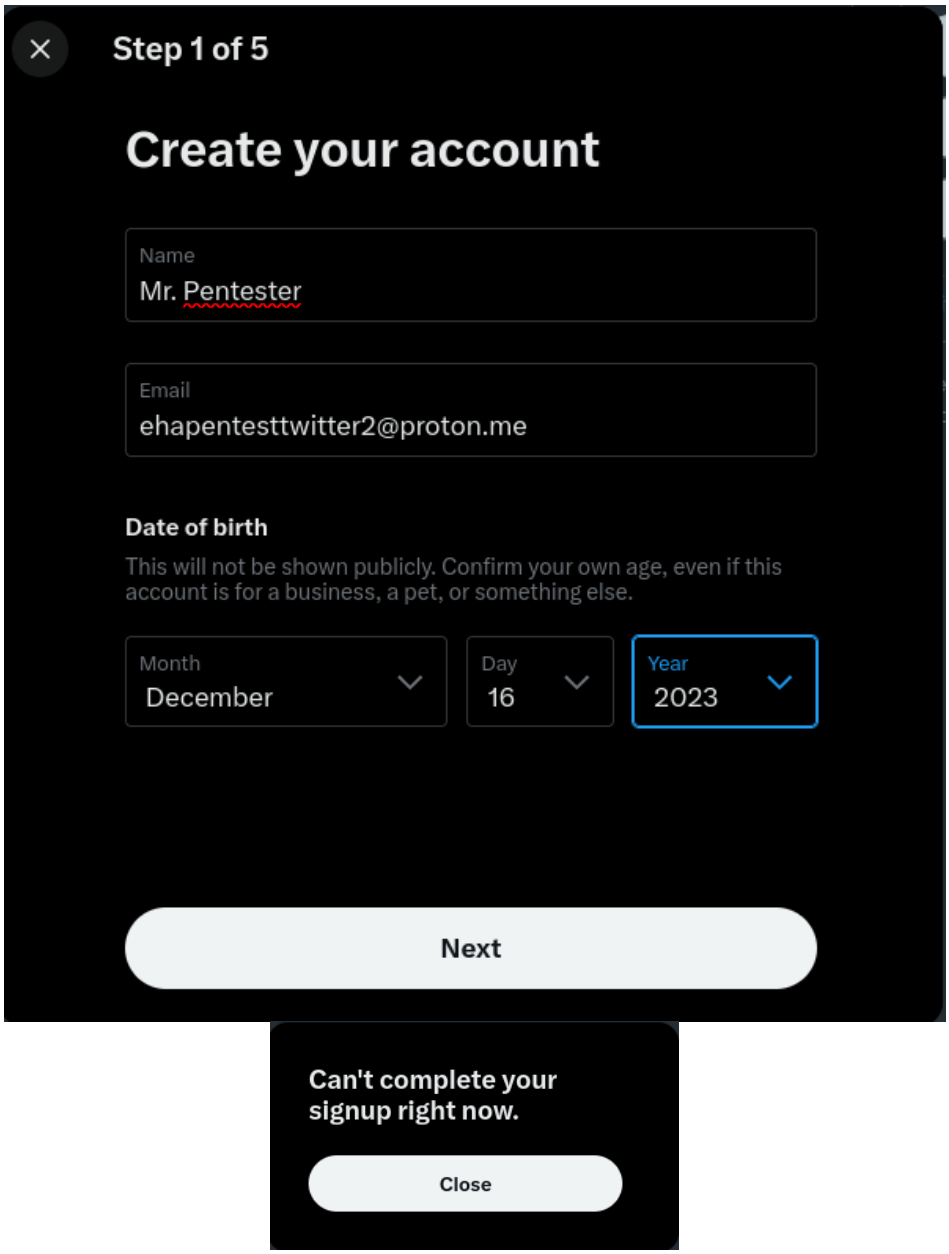
As we don't have access to the back-end source code we cannot determine which algorithms are used for the encryption that happens on the server side (password hashes, authentication cookies etc.).

4.10 Business Logic Testing

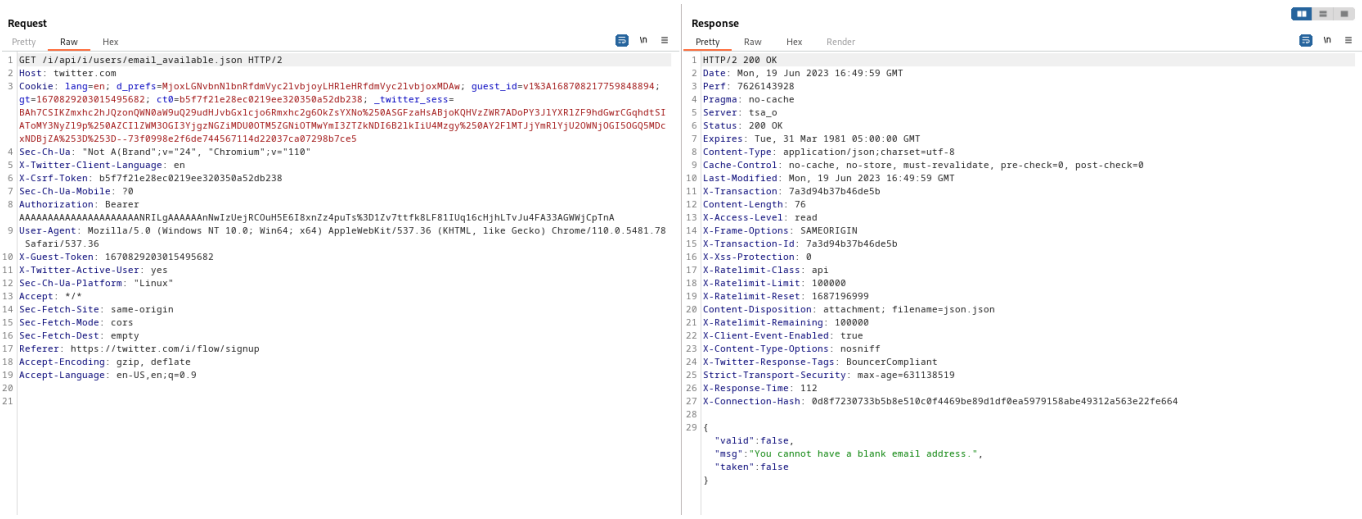
4.10.1 Test Business Logic Data Validation

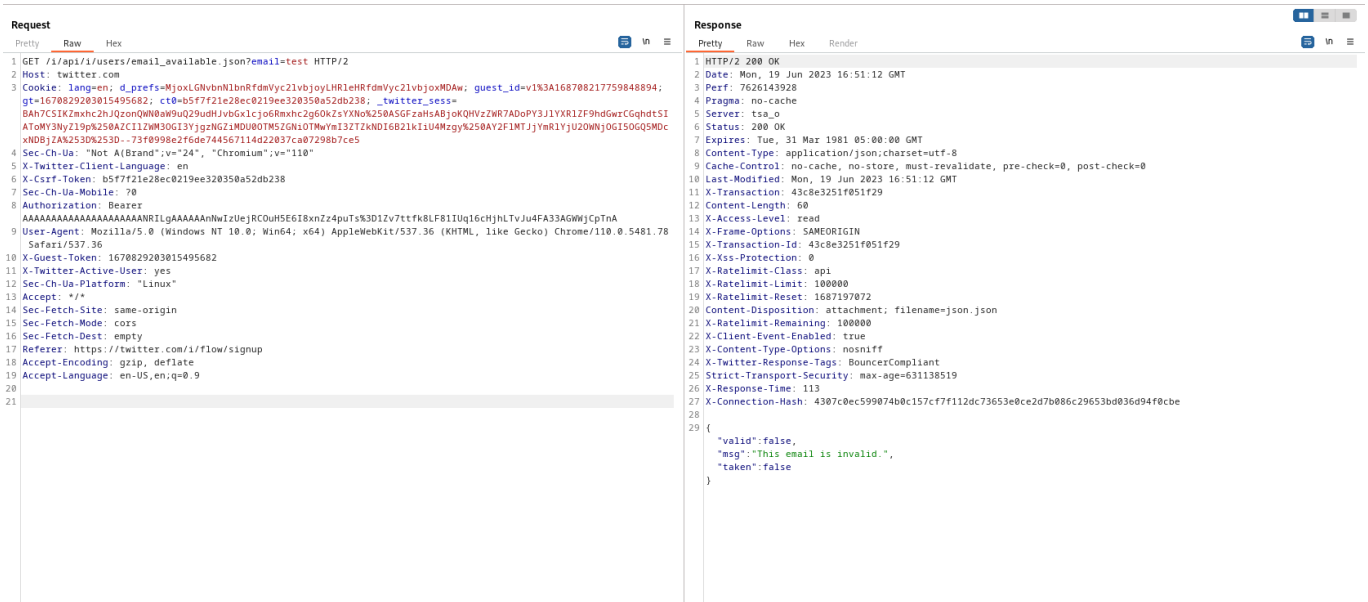
As for the GUI, the only input where validation is needed is the create account form. Other inputs such as tweet, searched value etc. may be pretty much anything. GUI checks the email meaning if it has form of an email address ({name}@{domain}). It doesn't check if the email actually exists, however the next step of account creation is entering the verification code which is sent to the email entered before. That implies that we cannot use neither non-existing email nor an email address that we don't have access to.
Date of birth is checked as well, account creation fails when date from future entered.





When performing single HTTP requests as such, containing invalid data, we received responses more or less describing why the request was invalid. Except when the JSON object contained two email elements, in that case the server responded with 500 Internal Server Error, meaning this edge case is not correctly handled (likely by the JSON parser).



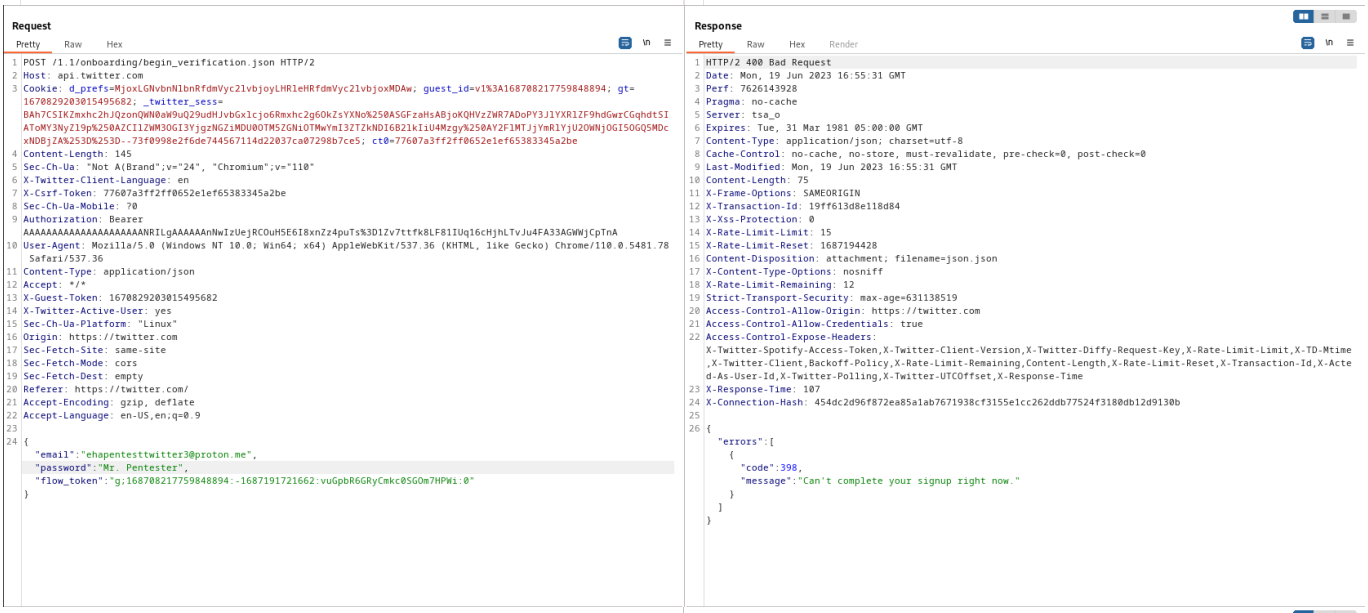


Request

```
GET /i/api/i/users/email_available.json?email=test HTTP/2
Host: twitter.com
Cookie: lang=en; d_prefs=MjoxLGNvbnN1bnRfdmVyc2lvbjoyLHR1eHRfdmVyc2lvbjoxMDAw; guest_id=v1%3A168708217759848894; gt=1670829203015495682; ct0=b5f7f21e28ec0219ee320350a52db238; _twitter_sess=BAh7CSIKZmxhc2hJQzonQWN0aW9uQ29udHJvbGxlcjo6Rmxhc2g6OkZsYXNo%250ASGFzaHsABjoKQHVzZWR7ADoPY3J1YXR1ZF9hdGwrCGqhdtSIAToMY3NyZl9p%250AZCI1ZWM3OGI3YjgzNGZiMDU0OTM5ZGNiOTMwYmI3ZTZkNDI6B21kIiU4Mzgy%250AY2F1MTJjYmRlYjU2OWNjOGI5OGQ5MDcxNDBjZA%253D%253D--73f0998e2f6de744567114d22037ca07298b7ce5
Sec-Ch-Ua: "Not A(Brand";v="24", "Chromium";v="110"
X-Twitter-Client-Language: en
X-Csrf-Token: b5f7f21e28ec0219ee320350a52db238
Sec-Ch-Ua-Mobile: ?0
Authorization: Bearer AAAAAAAAAAAAAAAAAAAAANRILgAAAAAAnNwIzUejRCOuH5E6I8xnZz4puTs%3D1Zv7ttfk8LF81IUq16cHjhLTvJu4FA33AGWWjCpTnA
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.5481.78 Safari/537.36
X-Guest-Token: 1670829203015495682
X-Twitter-Active-User: yes
Sec-Ch-Ua-Platform: "Linux"
Accept: */*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://twitter.com/i/flow/signup
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
```

Response

```
HTTP/2 200 OK
Date: Mon, 19 Jun 2023 16:51:12 GMT
Perf: 7626143928
Pragma: no-cache
Server: tsa_o
Status: 200 OK
Expires: Tue, 31 Mar 1981 05:00:00 GMT
Content-Type: application/json;charset=utf-8
Cache-Control: no-cache, no-store, must-revalidate, pre-check=0, post-check=0
Last-Modified: Mon, 19 Jun 2023 16:51:12 GMT
X-Transaction: 43c8e3251f051f29
Content-Length: 60
X-Access-Level: read
X-Frame-Options: SAMEORIGIN
X-Transaction-Id: 43c8e3251f051f29
X-Xss-Protection: 0
X-Ratelimit-Class: api
X-Ratelimit-Limit: 100000
X-Ratelimit-Reset: 1687197072
Content-Disposition: attachment; filename=json.json
X-Ratelimit-Remaining: 100000
X-Client-Event-Enabled: true
X-Content-Type-Options: nosniff
X-Twitter-Response-Tags: BouncerCompliant
Strict-Transport-Security: max-age=631138519
X-Response-Time: 113
X-Connection-Hash: 4307c0ec599074b0c157cf7f112dc73653e0ce2d7b086c29653bd036d94f0cbe

{
  "valid":false,
  "msg":"This email is invalid.",
  "taken":false
}
```

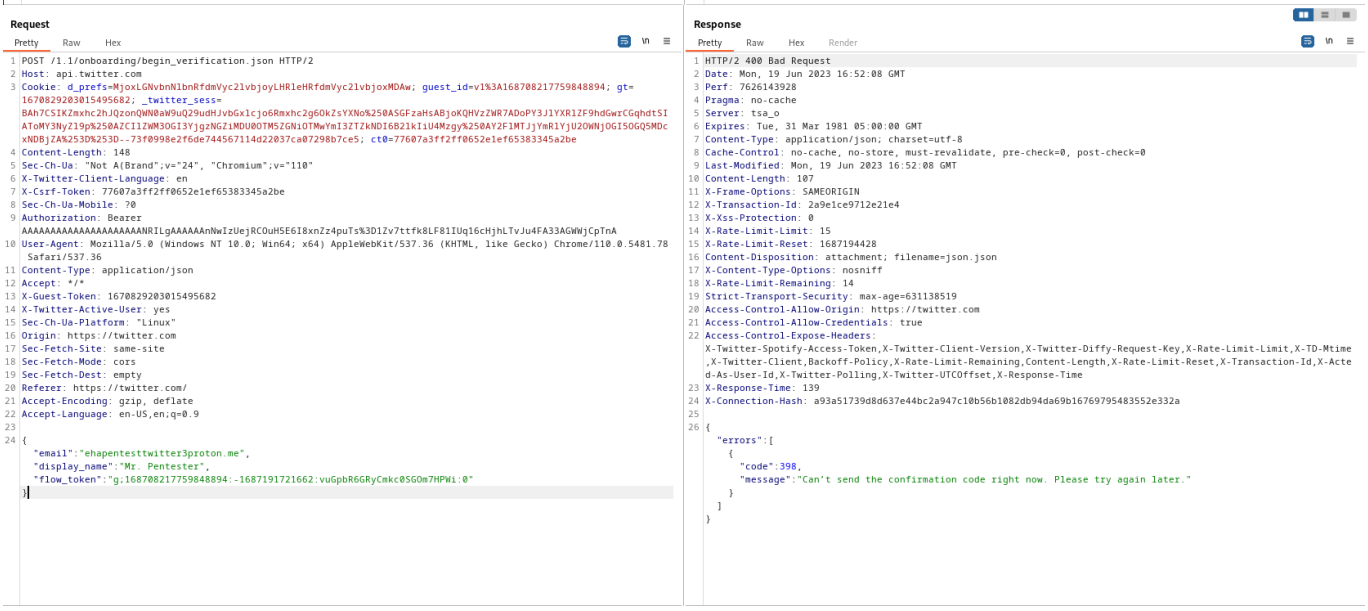


Request

```
POST /1.1/onboarding/begin_verification.json HTTP/2
Host: api.twitter.com
Cookie: d_prefs=MjoxLGNvbnN1bnRfdmVyc2lvbjoyLHR1eHRfdmVyc2lvbjoxMDAw; guest_id=v1%3A168708217759848894; gt=1670829203015495682; _twitter_sess=BAh7CSIKZmxhc2hJQzonQWN0aW9uQ29udHJvbGxlcjo6Rmxhc2g6OkZsYXNo%250ASGFzaHsABjoKQHVzZWR7ADoPY3J1YXR1ZF9hdGwrCGqhdtSIAToMY3NyZl9p%250AZCI1ZWM3OGI3YjgzNGZiMDU0OTM5ZGNiOTMwYmI3ZTZkNDI6B21kIiU4Mzgy%250AY2F1MTJjYmRlYjU2OWNjOGI5OGQ5MDcxNDBjZA%253D%253D--73f0998e2f6de744567114d22037ca07298b7ce5; ct0=77607a3ff2ff0652e1ef65383345a2be
Content-Length: 145
Sec-Ch-Ua: "Not A(Brand";v="24", "Chromium";v="110"
X-Twitter-Client-Language: en
X-Csrf-Token: 77607a3ff2ff0652e1ef65383345a2be
Sec-Ch-Ua-Mobile: ?0
Authorization: Bearer AAAAAAAAAAAAAAAAAAAANRILgAAAAAAnNwIzUejRCOuH5E6I8xnZz4puTs%3D1Zv7ttfk8LF81IUq16cHjhLTvJu4FA33AGWWjCpTnA
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.5481.78 Safari/537.36
Content-Type: application/json
Accept: */*
X-Guest-Token: 1670829203015495682
X-Twitter-Active-User: yes
Sec-Ch-Ua-Platform: "Linux"
Origin: https://twitter.com
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://twitter.com/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

{
  "email":"ehapentesttwitter3@proton.me",
  "password":"Mr. Pentester",
  "flow_token":"g;168708217759848894:-1687191721662:vuGpbR6GRyCmkc0SGOm7HPWi:0"
}
```

Response

```
HTTP/2 400 Bad Request
Date: Mon, 19 Jun 2023 16:55:31 GMT
Perf: 7626143928
Pragma: no-cache
Server: tsa_o
Expires: Tue, 31 Mar 1981 05:00:00 GMT
Content-Type: application/json; charset=utf-8
Cache-Control: no-cache, no-store, must-revalidate, pre-check=0, post-check=0
Last-Modified: Mon, 19 Jun 2023 16:55:31 GMT
Content-Length: 75
X-Frame-Options: SAMEORIGIN
X-Transaction-Id: 19ff613d8e118d84
X-Xss-Protection: 0
X-Rate-Limit-Limit: 15
X-Rate-Limit-Reset: 1687194428
Content-Disposition: attachment; filename=json.json
X-Content-Type-Options: nosniff
X-Rate-Limit-Remaining: 12
Strict-Transport-Security: max-age=631138519
Access-Control-Allow-Origin: https://twitter.com
Access-Control-Allow-Credentials: true
Access-Control-Expose-Headers: X-Twitter-Spotify-Access-Token,X-Twitter-Client-Version,X-Twitter-Diffy-Request-Key,X-Rate-Limit-Limit,X-TD-Mtime,X-Twitter-Client,Backoff-Policy,X-Rate-Limit-Remaining,Content-Length,X-Rate-Limit-Reset,X-Transaction-Id,X-Acted-As-User-Id,X-Twitter-Polling,X-Twitter-UTCOffset,X-Response-Time
X-Response-Time: 107
X-Connection-Hash: 454dc2d96f872ea85a1ab7671938cf3155e1cc262ddb77524f3180db12d9130b

{
  "errors":[
    {
      "code":398,
      "message":"Can't complete your signup right now."
    }
  ]
}
```

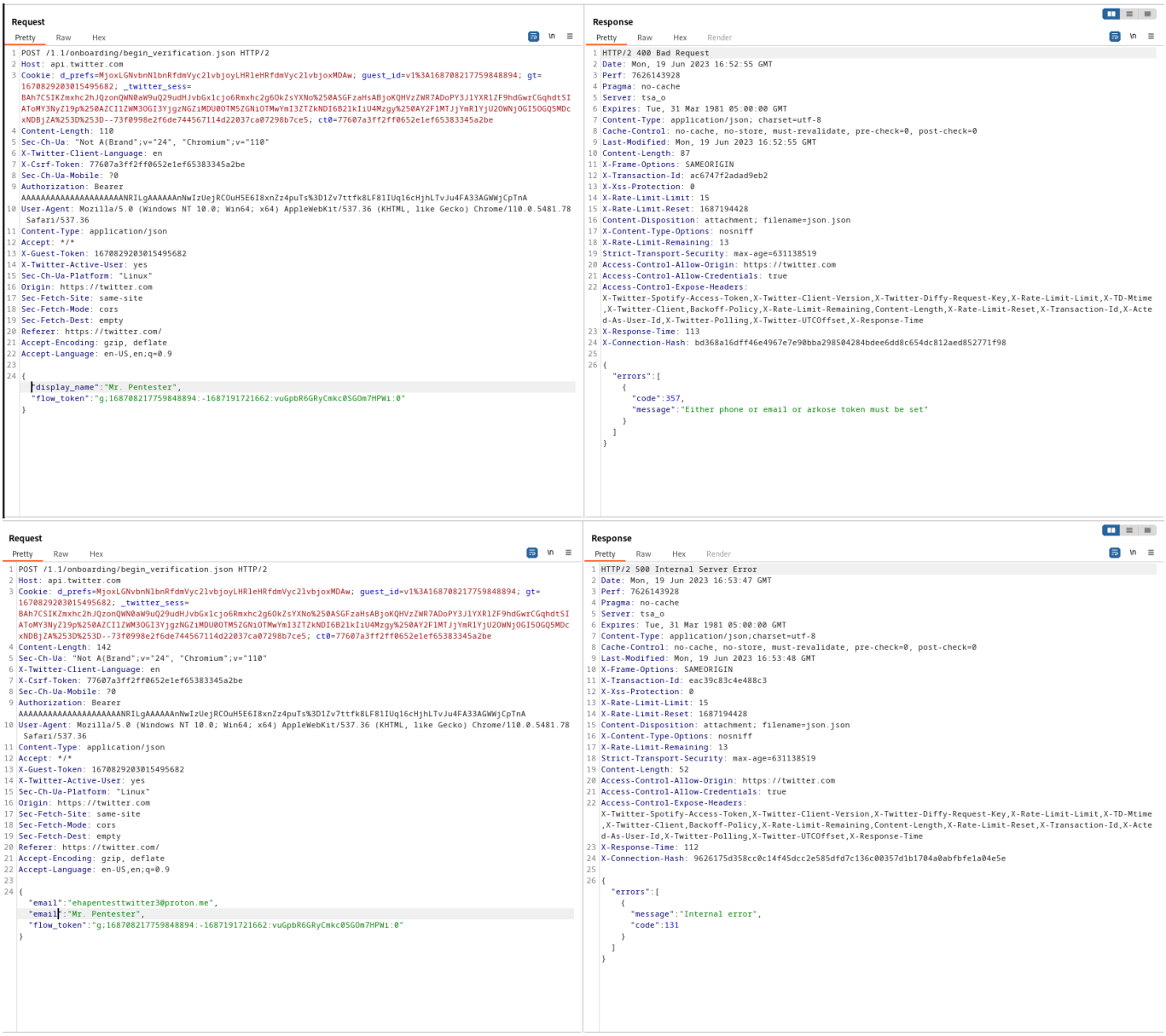


Request

```
POST /1.1/onboarding/begin_verification.json HTTP/2
Host: api.twitter.com
Cookie: d_prefs=MjoxLGNvbnN1bnRfdmVyc2lvbjoyLHR1eHRfdmVyc2lvbjoxMDAw; guest_id=v1%3A168708217759848894; gt=1670829203015495682; _twitter_sess=BAh7CSIKZmxhc2hJQzonQWN0aW9uQ29udHJvbGxlcjo6Rmxhc2g6OkZsYXNo%250ASGFzaHsABjoKQHVzZWR7ADoPY3J1YXR1ZF9hdGwrCGqhdtSIAToMY3NyZl9p%250AZCI1ZWM3OGI3YjgzNGZiMDU0OTM5ZGNiOTMwYmI3ZTZkNDI6B21kIiU4Mzgy%250AY2F1MTJjYmRlYjU2OWNjOGI5OGQ5MDcxNDBjZA%253D%253D--73f0998e2f6de744567114d22037ca07298b7ce5; ct0=77607a3ff2ff0652e1ef65383345a2be
Content-Length: 148
Sec-Ch-Ua: "Not A(Brand";v="24", "Chromium";v="110"
X-Twitter-Client-Language: en
X-Csrf-Token: 77607a3ff2ff0652e1ef65383345a2be
Sec-Ch-Ua-Mobile: ?0
Authorization: Bearer AAAAAAAAAAAAAAAAAAAANRILgAAAAAAnNwIzUejRCOuH5E6I8xnZz4puTs%3D1Zv7ttfk8LF81IUq16cHjhLTvJu4FA33AGWWjCpTnA
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.5481.78 Safari/537.36
Content-Type: application/json
Accept: */*
X-Guest-Token: 1670829203015495682
X-Twitter-Active-User: yes
Sec-Ch-Ua-Platform: "Linux"
Origin: https://twitter.com
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://twitter.com/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

{
  "email":"ehapentesttwitter3proton.me",
  "display_name":"Mr. Pentester",
  "flow_token":"g;168708217759848894:-1687191721662:vuGpbR6GRyCmkc0SGOm7HPWi:0"
}
```

Response

```
HTTP/2 400 Bad Request
Date: Mon, 19 Jun 2023 16:52:08 GMT
Perf: 7626143928
Pragma: no-cache
Server: tsa_o
Expires: Tue, 31 Mar 1981 05:00:00 GMT
Content-Type: application/json; charset=utf-8
Cache-Control: no-cache, no-store, must-revalidate, pre-check=0, post-check=0
Last-Modified: Mon, 19 Jun 2023 16:52:08 GMT
Content-Length: 107
X-Frame-Options: SAMEORIGIN
X-Transaction-Id: 2a9e1ce9712e21e4
X-Xss-Protection: 0
X-Rate-Limit-Limit: 15
X-Rate-Limit-Reset: 1687194428
Content-Disposition: attachment; filename=json.json
X-Content-Type-Options: nosniff
X-Rate-Limit-Remaining: 14
Strict-Transport-Security: max-age=631138519
Access-Control-Allow-Origin: https://twitter.com
Access-Control-Allow-Credentials: true
Access-Control-Expose-Headers: X-Twitter-Spotify-Access-Token,X-Twitter-Client-Version,X-Twitter-Diffy-Request-Key,X-Rate-Limit-Limit,X-TD-Mtime,X-Twitter-Client,Backoff-Policy,X-Rate-Limit-Remaining,Content-Length,X-Rate-Limit-Reset,X-Transaction-Id,X-Acted-As-User-Id,X-Twitter-Polling,X-Twitter-UTCOffset,X-Response-Time
X-Response-Time: 139
X-Connection-Hash: a93a51739d8d637e44bc2a947c10b56b1082db94da69b167697954835552e332a

{
  "errors":[
    {
      "code":398,
      "message":"Can't send the confirmation code right now. Please try again later."
    }
  ]
}
```



4.10.2 Test Ability to Forge Requests

When reviewing the common HTTP traffic, no guessable values nor hidden features were found. Data consist of user entered content and apparently randomly generated tokens and ids.

4.10.3 Test Integrity Checks

We didn't find neither hidden fields nor places to insert information into areas that are non-editable.

4.10.4 Test for Process Timing

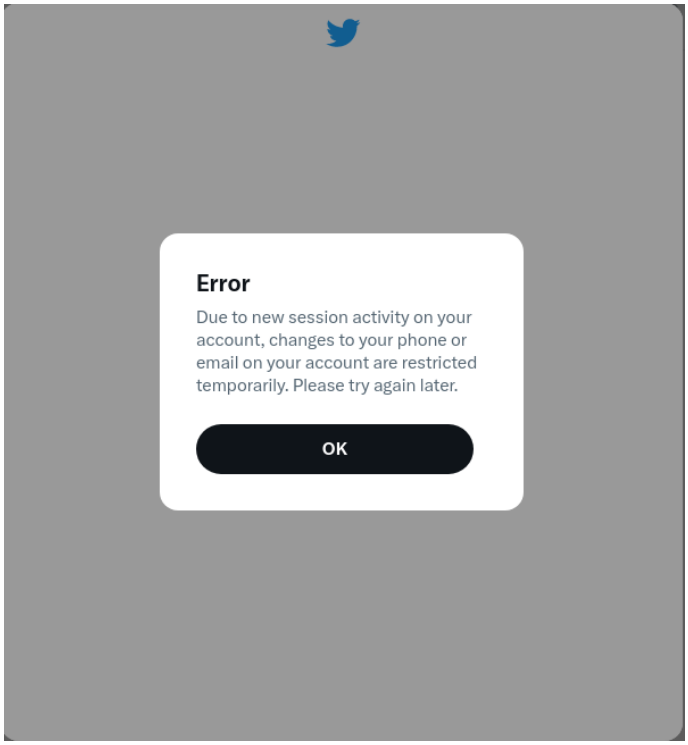
Entering both correct and incorrect usernames and passwords took the same time to process. We didn't find any other scenario in which process timing could be taken advantage of.

4.10.5 Test Number of Times a Function Can Be Used Limits

The application has several technical limits as follows:

Functionality	Limit
Direct Messages	500 messages sent per day
Tweets	2400 per day
Changes to account email	4 per hour
Following (daily)	400 per day
Following (account-based)	5000 per account

We haven't been able to test any of them because the limits are too high to perform manually, sending repeated requests through BurpSuite or curl didn't work and API for those tasks is a paid functionality. In case of Changes to account email it would be possible to perform manual test except the fact that probably due to our penetration testing Twitter wouldn't allow us to do so, even when we created a fresh account with no suspicious history.



4.10.6 Testing for the Circumvention of Work Flows

The only multistep activity is the account creation, which cannot be bypassed due to email verification as one of the steps. No other multistep activity was found.

4.10.7 Test Defenses Against Application Misuse

During the testing process we have found several defense mechanisms against application misuse:

- account lock out mechanism when repeatedly entered incorrect credentials
- impossibility of repeatedly posting tweets using the same HTTP request
- impossibility of repeatedly sending direct messages using the same HTTP request
- restrictions for account settings when suspicious activity is spotted
- front-end source code is well obfuscated with no comments in place and torn into pieces which makes it nearly impossible to understand

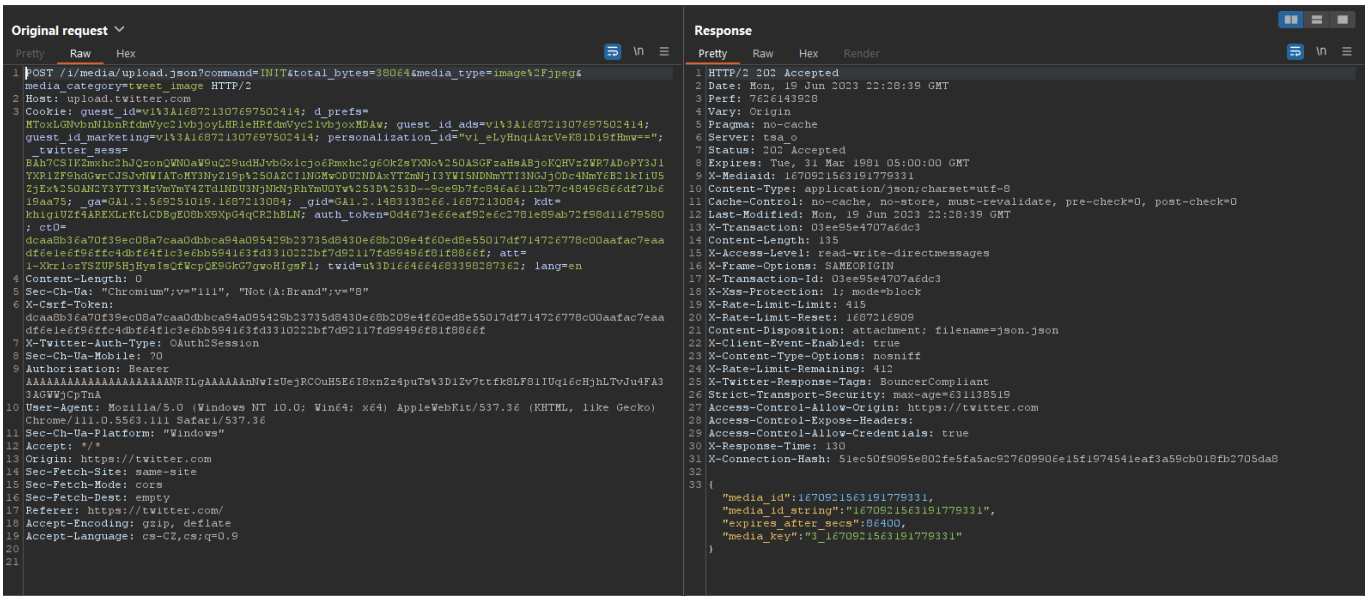
4.10.8 Test Upload of Unexpected File Types

The application supports uploading media when posting a tweet. Officially accepted file formats are GIF, JPEG and PNG.

GUI doesn't allow to upload any file format but these three. Even when an unsupported file is given a supported suffix, the page reports that upload failed.

When intercepting the request for upload the media and changing both file type and file data, the application reports that upload failed.

When media is uploaded successfuly, the aplication doesn't disclose the path but only reports the media id.



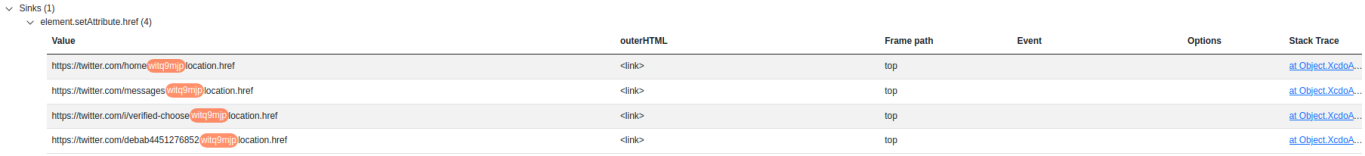
4.10.9 Test Upload of Malicious Files

As we weren't able to upload any unexpected file types, that leaves us with uploading a malicious image, but since the malicious code inserted into an image would be run by the browser, not the application, it is beyond the scope of this pentest.

4.11 Client-side Testing

4.11.1 Testing for DOM-Based Cross Site Scripting

Manual testing and searching for possible sources of DOM CSS did not yield any results. Testing with Burp Suite's DOM Invader gave a few sinks but these have proven to be false-positives. Application does not seem vulnerable to DOM-based CSS.



Twitter uses graphQL and its API to load dynamic page contents and store user tweets. All validation therefore happens in the backend, which we have no access to, so it is very difficult to guess what is happening there. However all tests indicate that the all the input validation/sanitization is well performed and no errors or suspicious artefacts were found.

4.11.2 Testing for JavaScript Execution

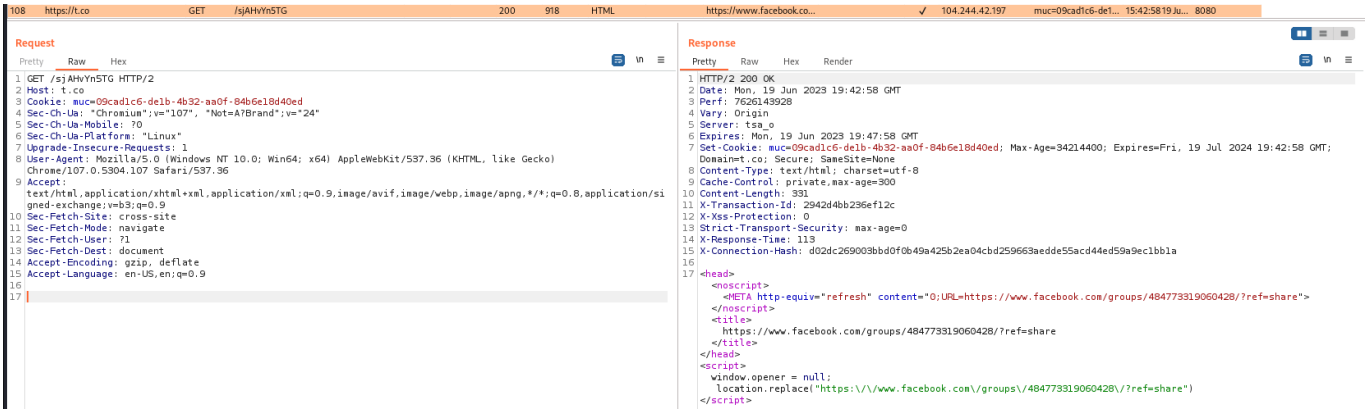
Similar to the previous paragraph, we did not find any possible attack vector.

4.11.3 Testing for HTML Injection

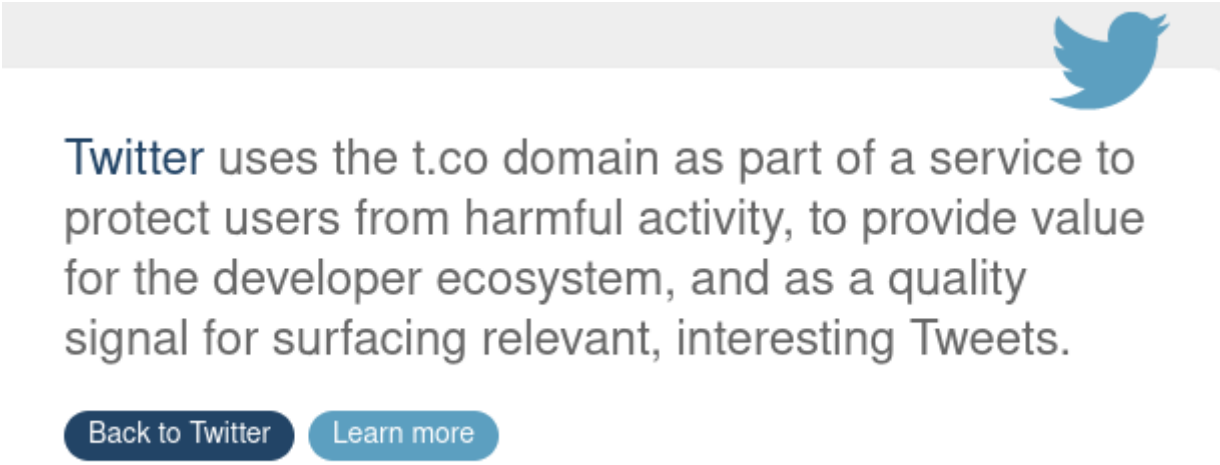
Again no possible input for HTML Injection found.

4.11.4 Testing for Client-side URL Redirect

URL redirect again happens through backend API and does not leave room for tampering.
When user clicks a link in a tweet from a different user he is redirected with the following GET request and response.



We can see that the request goes to the `t.co` webpage, which is twitter's own link shortener that, they claim, also protects users from various malicious activity.



4.11.5 Testing for CSS Injection

No CSS injection point found on the website.

4.11.6 Testing for Client-side Resource Manipulation

No vulnerability was found.

4.11.7 Testing Cross Origin Resource Sharing

Twitter uses `Access-Control-Allow-Origin` header safely only allowing `https://twitter.com` (no wildcard used). Other `Access-Control` headers are also used in a reasonable way.

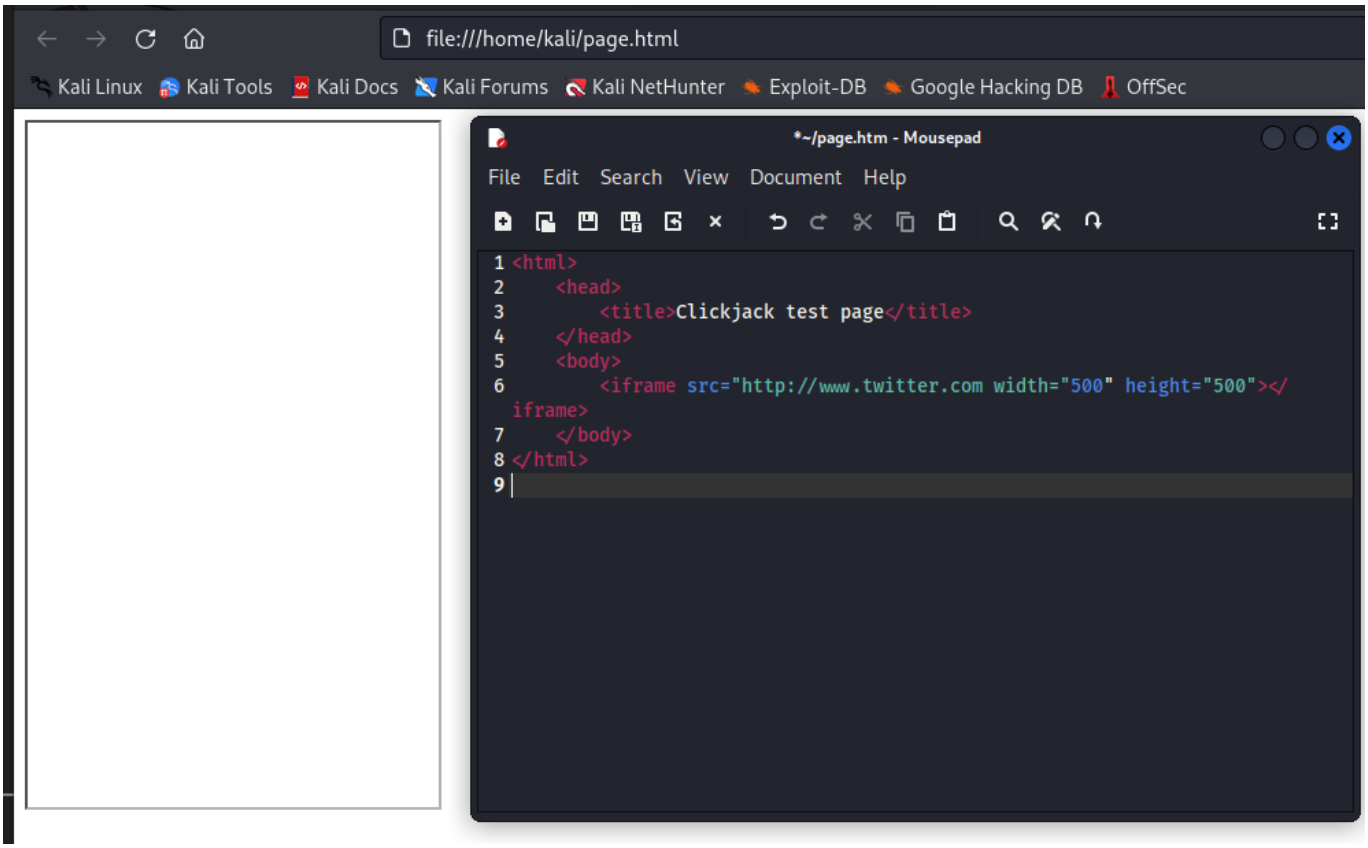
```
HTTP/2 200 OK
Access-Control-Allow-Origin: https://twitter.com
Access-Control-Allow-Credentials: true
Access-Control-Allow-Methods: HEAD,PUT,GET,POST,DELETE
Access-Control-Allow-Headers:
X-Contributor-Version,X-Twitter-Client-Version,Dtab-Local,X-Twitter-Client-Language,If-Modified-Since,X-Twitt
er-Client,X-TD-Iff-Mtime,X-Twitter-Auth-Type,X-B3-Flags,Content-Type,X-TD-Mtime-Check,X-CSRF-Token,X-Twitter-
Polling,X-Twitter-Active-User,X-Guest-Token,X-Twitter-UTCOffset,X-Act-As-User-Id,Authorization,X-Contribute-T
o-User-Id
Access-Control-Max-Age: 1728000
Access-Control-Expose-Headers:
X-Twitter-Client-Version,X-Rate-Limit-Limit,X-TD-Mtime,X-Twitter-Client,X-Rate-Limit-Remaining,X-Rate-Limit-R
eset,X-Acted-As-User-Id,X-Twitter-Polling,X-Twitter-UTCOffset
X-Connection-Hash: 0eb1a0e9806761a13db7e6b959e4feaa52009aaf9226d6d6a92496ac94c44c69
Date: Mon, 19 Jun 2023 19:41:35 GMT
Server: tsa_o
```

4.11.8 Testing for Cross Site Flashing

Attack not possible.

4.11.9 Testing for Clickjacking

Basic clickjacking attempt yields no result as the page does not show in our frame.



Advanced clickjacking attempts like double framing or using sandbox and security attributes are also unsuccessful.
Twitter uses header `X-Frame-Options: SAMEORIGIN` that ensures that page's contents can not be loaded into frames, iframes etc. that are from different origin than the page itself, effectively preventing Clickjacking attacks.

4.11.10 Testing WebSockets

No WebSockets signs were found in the client-side source code or communication.

4.11.11 Testing Web Messaging

Twitter does not use Web Messaging (Cross Document Messaging).

4.11.12 Testing Browser Storage

Twitter does not store any persistent values in `window.localStorage`, the only value output when trying to list the storage is `volume: 1` (note that the value changes based on browser used).
Listing `sessionStorage` which is non-persistent and resets after window closes yields some results but does not store any sensitive data.

Listing IndexedDB does not show any issues.
Cookies do not store any sensitive data neither.
Nothing of interest found stored as global window object.

4.11.13 Testing for Cross Site Script Inclusion

No global variables/function parameters or CSV values containging sensitive data were found.
There are no JS warnings or errors in the browser console.

Sources

- https://blog.twitter.com/engineering/en_us/topics/infrastructure/2017/the-infrastructure-behind-twitter-scale
- https://www.first.org/cvss
- https://blog.twitter.com/engineering/en_us/topics/
- http://www.pentest-standard.org
- https://www.thirdrocktechkno.com/blog/react-security-vulnerabilities/
- https://help.twitter.com/en/rules-and-policies/twitter-limits
- https://help.twitter.com/en/rules-and-policies/twitter-cookies