Deliverable 4

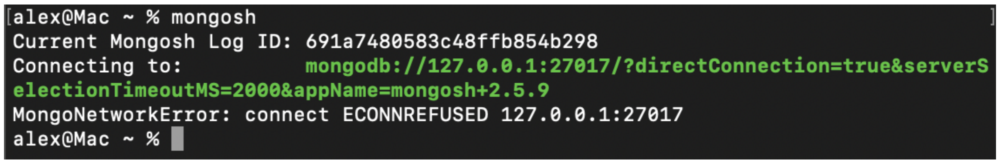
Team Name: DAAP-SecDov-518

MILESTONE REPORT

Fixes Implemented:

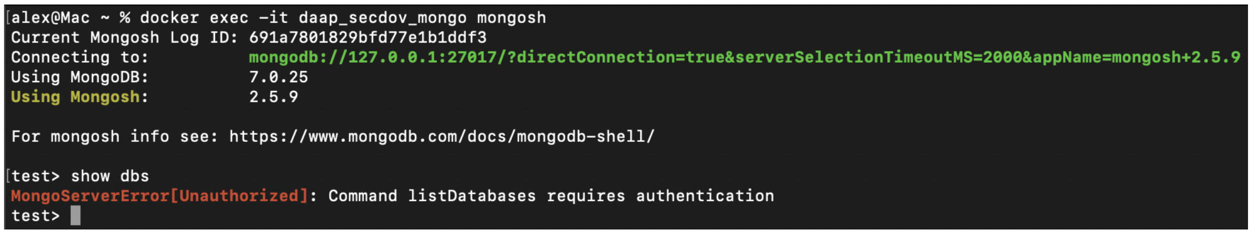
Securing the Database:

The application database was initially exposed on port 27017:27017, and lacked authentication controls. This allowed for easy inspection of the database during early development, however it exposed the database to anyone with network access to the host environment. Having removed this port mapping, access into the database container is no longer allowed by simply executing mongosh with no parameters.



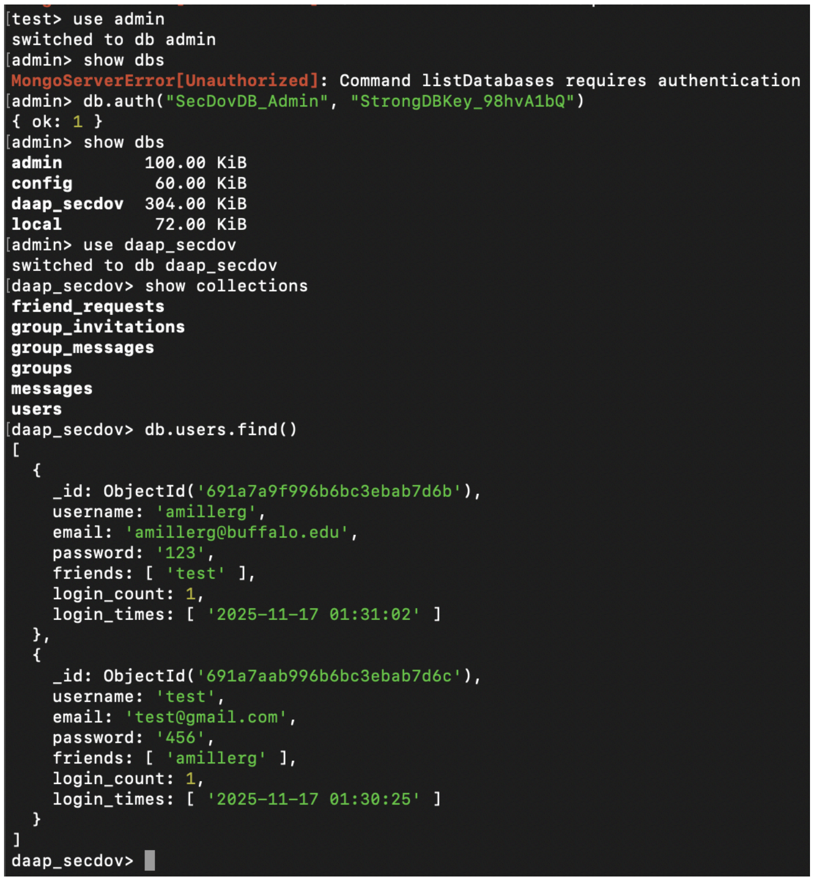
```
alex@Mac ~ % mongosh
Current Mongosh Log ID: 691a7480583c48ffb854b298
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverS
electionTimeoutMS=2000&appName=mongosh+2.5.9
MongoNetworkError: connect ECONNREFUSED 127.0.0.1:27017
alex@Mac ~ %
```

The database container can now be entered using docker exec -it daap_secdov_mongo mongosh, however authentication is now required to execute commands.



```
alex@Mac ~ % docker exec -it daap_secdov_mongo mongosh
Current Mongosh Log ID: 691a7801829bfd77e1b1ddf3
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.9
Using MongoDB:          7.0.25
Using Mongosh:          2.5.9

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

test> show dbs
MongoServerError[Unauthorized]: Command listDatabases requires authentication
test>
```

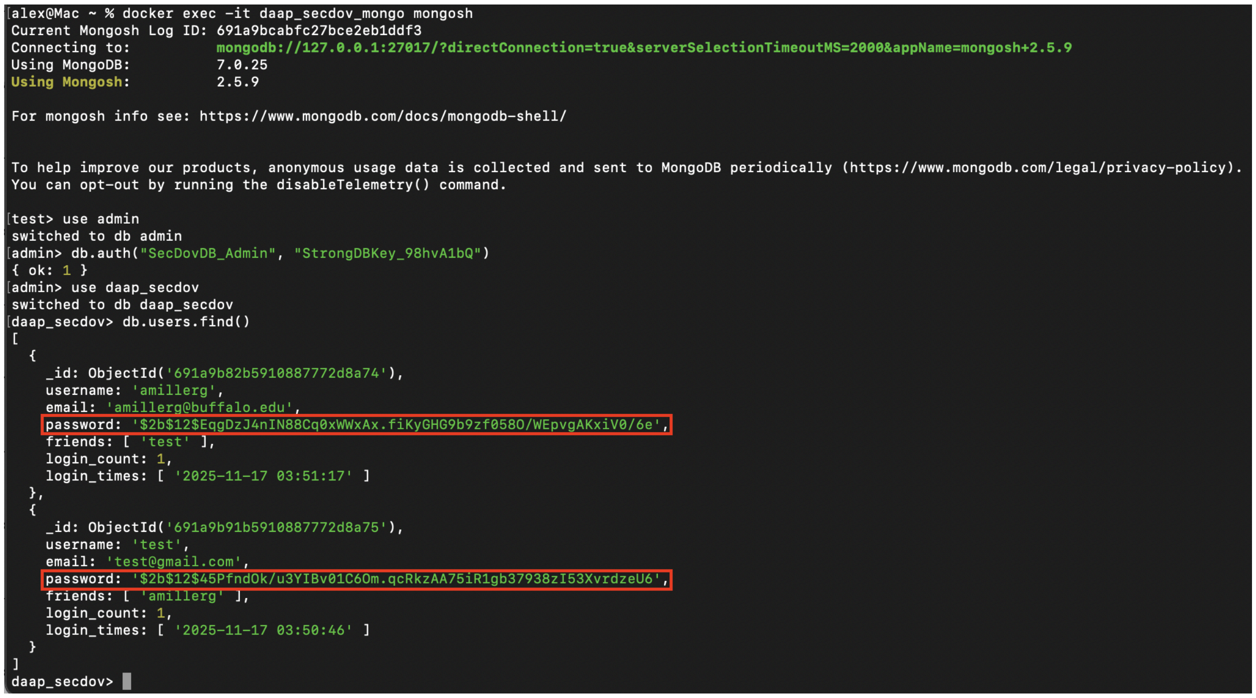
Access control was implemented by reconfiguring MongoDB to create a root administrative user upon initialization. Database operations now require authentication through prior input of defined username and password credentials. Unauthenticated operations are now rejected as shown in the image below:



```
[test> use admin
switched to db admin
[admin> show dbs
MongoServerError[Unauthorized]: Command listDatabases requires authentication
[admin> db.auth("SecDovDB_Admin", "StrongDBKey_98hvA1bQ")
{ ok: 1 }
[admin> show dbs
admin        100.00 KiB
config        60.00 KiB
daap_secdov  304.00 KiB
local         72.00 KiB
[admin> use daap_secdov
switched to db daap_secdov
[daap_secdov> show collections
friend_requests
group_invitations
group_messages
groups
messages
users
[daap_secdov> db.users.find()
[
  {
    _id: ObjectId('691a7a9f996b6bc3ebab7d6b'),
    username: 'amillerg',
    email: 'amillerg@buffalo.edu',
    password: '123',
    friends: [ 'test' ],
    login_count: 1,
    login_times: [ '2025-11-17 01:31:02' ]
  },
  {
    _id: ObjectId('691a7aab996b6bc3ebab7d6c'),
    username: 'test',
    email: 'test@gmail.com',
    password: '456',
    friends: [ 'amillerg' ],
    login_count: 1,
    login_times: [ '2025-11-17 01:30:25' ]
  }
]
daap_secdov>
```

Password Hashing and Brute Force Mitigation:

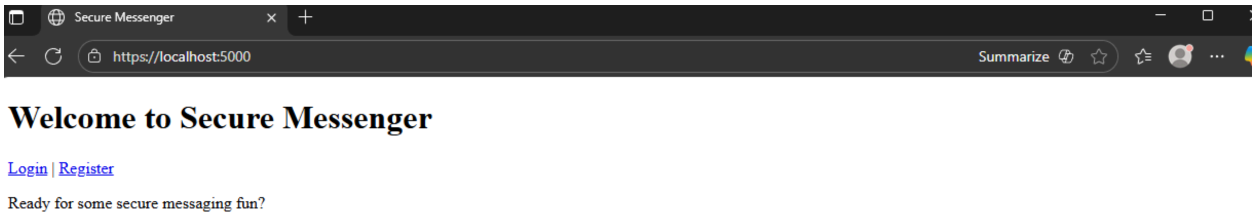
The application database originally stored the passwords of users in plaintext. This posed a significant security concern as anyone with access to the database could directly view and reuse user credentials. The password-hashing function Bcrypt was used to remedy this vulnerability, as recommended by the attacking team. During registration, users' passwords are now salted and hashed with Bcrypt. This hash is then stored in the application database, replacing the previously stored plaintext version. We also use the same process during password reset. To avoid timing differences between valid and invalid usernames, we hash all attempts, not just those of correct usernames (we use a dummy hash for invalid users). These hashes can be viewed in the database through the authenticated admin account as shown below:



```
alex@Mac ~ % docker exec -it daap_secdov_mongo mongosh
Current Mongosh Log ID: 691a9bcabfc27bce2eb1ddf3
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.9
Using MongoDB:          7.0.25
Using Mongosh:          2.5.9

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/


To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

test> use admin
switched to db admin
admin> db.auth("SecDovDB_Admin", "StrongDBKey_98hvA1bQ")
{ ok: 1 }
admin> use daap_secdov
switched to db daap_secdov
daap_secdov> db.users.find()
[
  {
    _id: ObjectId('691a9b82b5910887772d8a74'),
    username: 'amillerg',
    email: 'amillerg@buffalo.edu',
    password: '$2b$12$EqgDzJ4nIN88Cq0xWWxAx.fiKyGHG9b9zf058O/WEpvgAKxiV0/6e',
    friends: [ 'test' ],
    login_count: 1,
    login_times: [ '2025-11-17 03:51:17' ]
  },
  {
    _id: ObjectId('691a9b91b5910887772d8a75'),
    username: 'test',
    email: 'test@gmail.com',
    password: '$2b$12$45PfndOk/u3YIBv01C6Om.qcRkzAA75iR1gb37938zI53XvrdzeU6',
    friends: [ 'amillerg' ],
    login_count: 1,
    login_times: [ '2025-11-17 03:50:46' ]
  }
]
daap_secdov> 
```

Bcrypt includes a tunable work factor that determines how computationally expensive the hashing process is. This work factor is encoded into each Bcrypt hash string along with the salt and derived hash output. Our application defines a work factor of 12, meaning Bcrypt performs 2^{12} = 4096 rounds of its key derivation process for each password. This intentionally slows the password hashing process. Normal users are minimally affected by the slowdown, but large scale brute force cracking attempts now take significantly more time to execute.

HTTPS/TLS



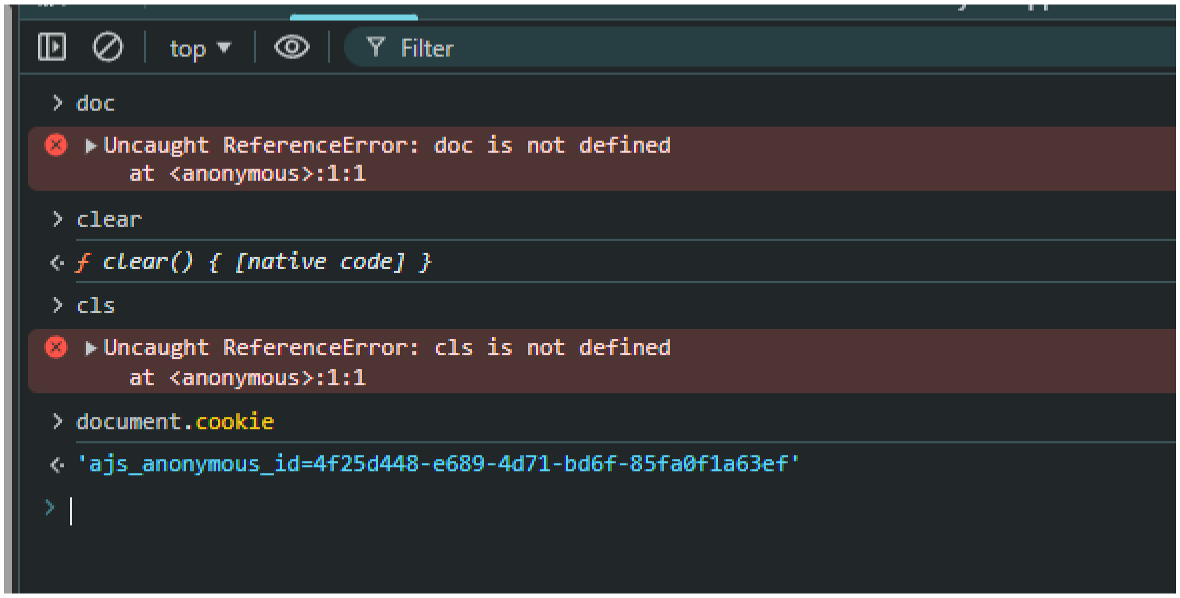
Welcome to Secure Messenger

Login | Register

Ready for some secure messaging fun?

To provide secure communication, HTTPS was implemented using Transport Layer Security (TLS). This ensures that all data transmitted between the user's browser and the server is encrypted in transit, preventing attackers from intercepting or reading sensitive information such as login credentials, messages, and session data. A TLS certificate was generated and configured inside the Dockerized Flask application, allowing the web server to serve content over HTTPS rather than plain HTTP. This significantly improves confidentiality and protects against man-in-the-middle (MITM) attacks by guaranteeing encrypted and authenticated connections.

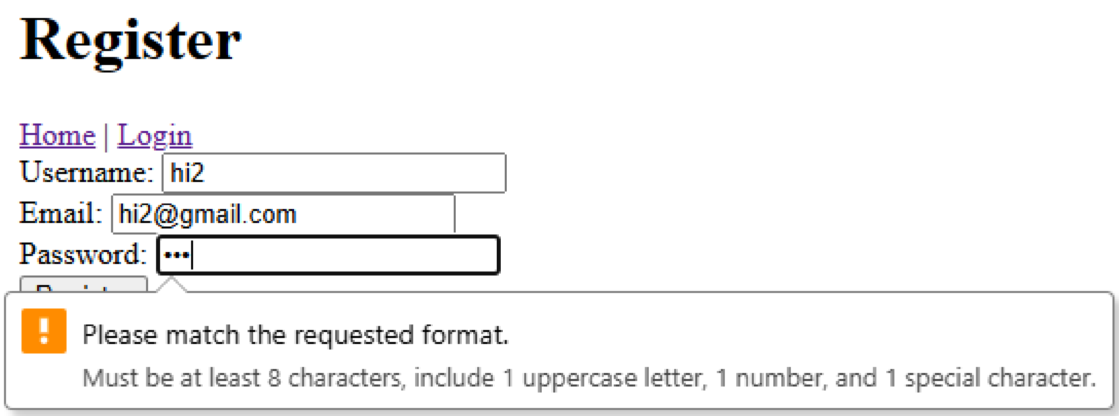
JS cookie protection



```
> doc
✖ ▶Uncaught ReferenceError: doc is not defined
     at <anonymous>:1:1
> clear
<· ƒ clear() { [native code] }
> cls
✖ ▶Uncaught ReferenceError: cls is not defined
     at <anonymous>:1:1
> document.cookie
<· 'ajs_anonymous_id=4f25d448-e689-4d71-bd6f-85fa0f1a63ef'
> |
```

To prevent session hijacking through JavaScript-based attacks, security flags were added to session cookies. Specifically, the **HttpOnly** flag was enabled to ensure that cookies cannot be accessed by client-side JavaScript, even if an attacker injects malicious scripts via XSS vulnerabilities. Additionally, the **Secure** flag was enabled to ensure cookies are only transmitted over encrypted HTTPS channels. Together, these settings significantly reduce the risk of cookie theft and unauthorized account access by protecting sensitive information on the client side.

Password Complexity for Users



Register

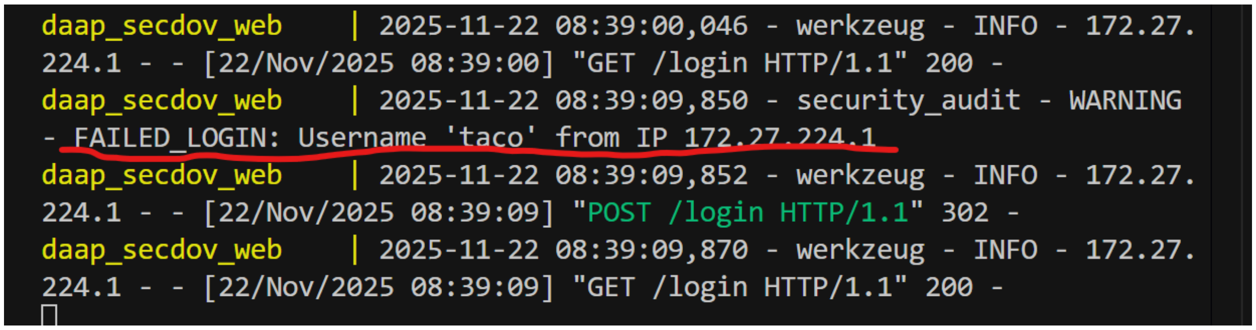
Home | Login
Username: hi2
Email: hi2@gmail.com
Password: •••

Please match the requested format.
Must be at least 8 characters, include 1 uppercase letter, 1 number, and 1 special character.

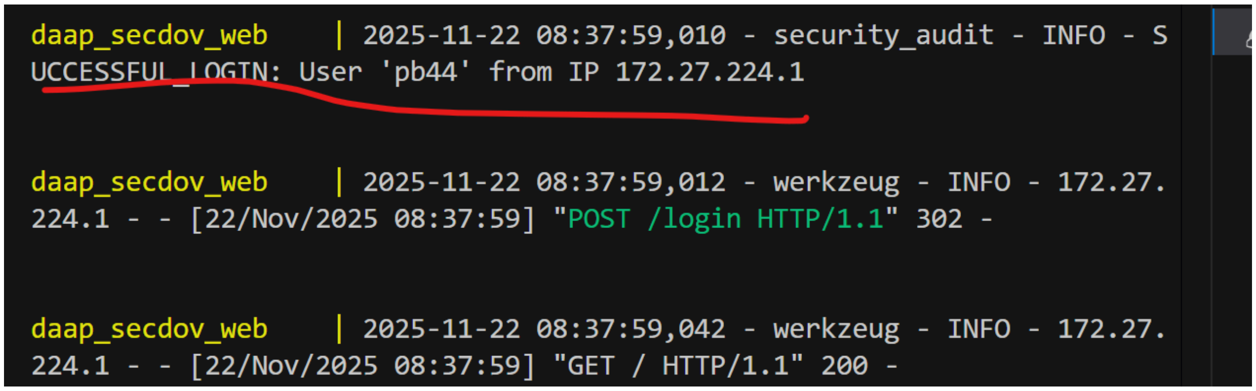
The registration system was enhanced by implementing strong password complexity requirements. Users are now required to create passwords that are at least eight characters long and include at least one uppercase letter, one numeric digit, and one special character. This validation is enforced on the backend to prevent bypass attacks and on the client side for better user experience. These controls reduce the risk of successful brute-force and credential stuffing attacks by making passwords significantly harder to guess or crack. Even if there is a database leak and the hashed passwords are found they will be significantly harder to crack since they will not be found in something like a rainbow table.

We also increased the complexity of the password reset codes from being 6 digits to being 8 digits and, per the TA suggestion from our original suggestion, they now last for only 5 minutes. We are also using the secrets module instead of random to make our codes more secure.

Security Audit Logs



```
daap_secdov_web    | 2025-11-22 08:39:00,046 - werkzeug - INFO - 172.27.
224.1 - - [22/Nov/2025 08:39:00] "GET /login HTTP/1.1" 200 -
daap_secdov_web    | 2025-11-22 08:39:09,850 - security_audit - WARNING
- FAILED_LOGIN: Username 'taco' from IP 172.27.224.1
daap_secdov_web    | 2025-11-22 08:39:09,852 - werkzeug - INFO - 172.27.
224.1 - - [22/Nov/2025 08:39:09] "POST /login HTTP/1.1" 302 -
daap_secdov_web    | 2025-11-22 08:39:09,870 - werkzeug - INFO - 172.27.
224.1 - - [22/Nov/2025 08:39:09] "GET /login HTTP/1.1" 200 -
```

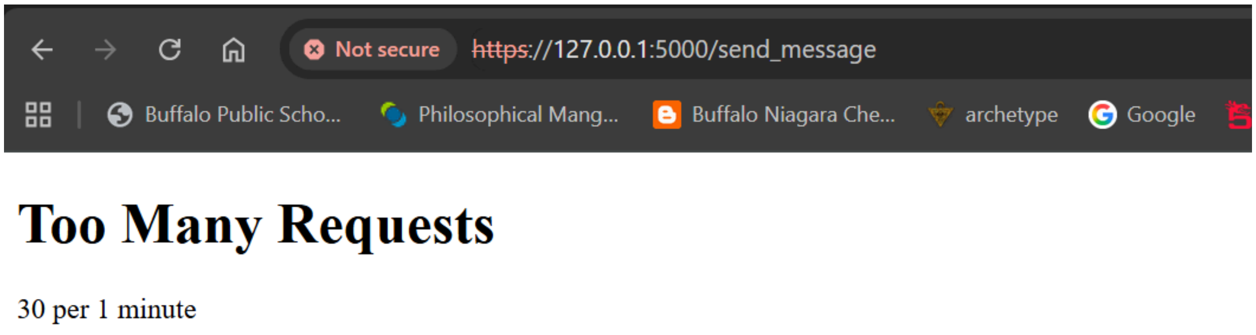

```
daap_secdov_web    | 2025-11-22 08:37:59,010 - security_audit - INFO - S
UCCESSFUL_LOGIN: User 'pb44' from IP 172.27.224.1

daap_secdov_web    | 2025-11-22 08:37:59,012 - werkzeug - INFO - 172.27.
224.1 - - [22/Nov/2025 08:37:59] "POST /login HTTP/1.1" 302 -

daap_secdov_web    | 2025-11-22 08:37:59,042 - werkzeug - INFO - 172.27.
224.1 - - [22/Nov/2025 08:37:59] "GET / HTTP/1.1" 200 -
```

New error auditing logs were added to record audit things like logins and failed logins. Currently these are visible in system logs and on the browser console but in practice in the live system, it would only be visible to admins in the server logs.

User Constraints On Functionality For Security and Sanitization

We restricted a lot of different features in our system to help protect users while still maximizing regular usability as much as possible. This ranged from limiting where/how javascript code could be executed on the site to constraints on how many messages a user could send or how big the files they could send are. We also sanitized user inputs to prevent possible avenues for user code. Here is a list of changes to carry this out:

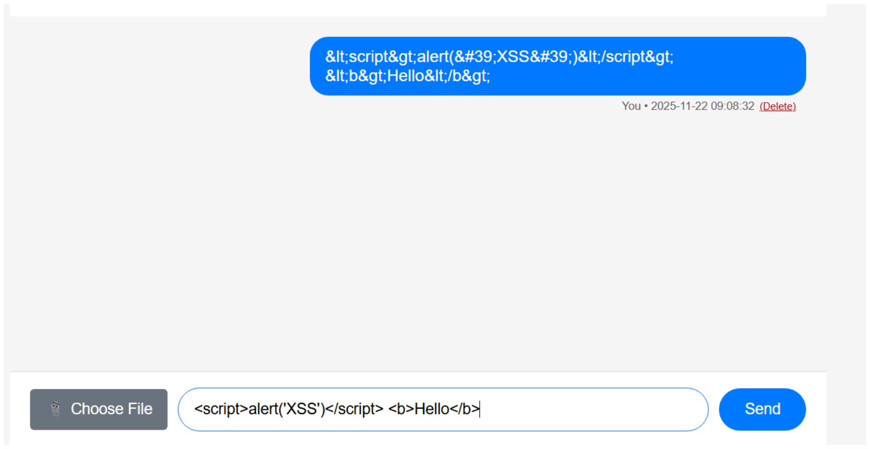
- Limited file uploads to be 16mb in size to help prevent the system from being overloaded from large files (potentially from a DoS attack)
- Set specific limits over a certain period of time for each of the POST routes. For example, messages are limited to 30/minute as that is pre-defined by the route. See below example:



Too Many Requests

30 per 1 minute

- We added security headers to restrict the browser to only use the file type specified, we restricted the resources that a browser was able to pre-load to be a small set (allowing things the necessary components for google's recaptcha, prevented most referral information from being sent to other sites (if navigating to our site) outside our own site, and specifically prevented api calls to the camera, microphone, and geolocation.
- Escaped both upload file names and all messages inputted by the user to prevent malicious access to internal files and prevent users from coming in contact with malicious code.


&lt;script&gt;alert(&#39;XSS&#39;)&lt;/script&gt; &lt;b&gt;Hello&lt;/b&gt;

You • 2025-11-22 09:08:32 (Delete)

<script>alert('XSS')</script> <b>Hello</b>

Above: Example of escaped Text as part of sanitization

Security Enhancement with Testing summary

Based on the security testing undertaken in Deliverable 3, we discovered a number of major security issues with our messaging application and developed a comprehensive plan to handle them.

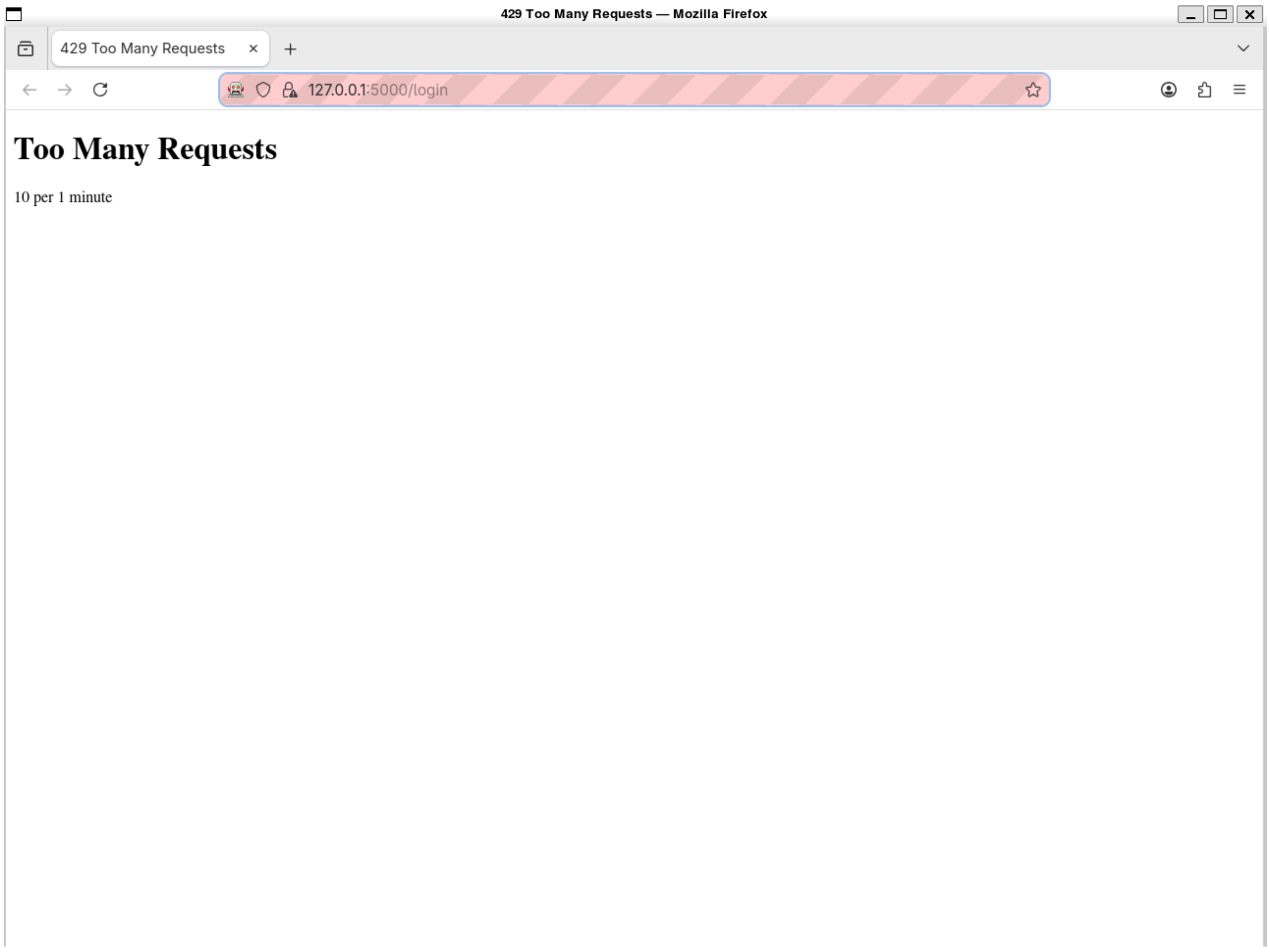
We conducted thorough testing to validate that all security improvements work correctly against the original attack methods.

We tried accessing the database without credentials and found that it indeed requires authentication now. Testing password security revealed passwords are hashed appropriately in the database and cannot be accessed in plain text. Brute force attacks that were possible earlier are prevented by our rate limiting implementation. We confirmed that weak passwords, such as "123", are denied at both registration and password change time.

Session security testing confirmed cookies are now flagged with HttpOnly, which prohibits JavaScript from accessing them. All previous attacks that succeeded in Deliverable 3 no longer work against our secured implementation.

These security metrics have significantly improved in all aspects. Where previously confidentiality was completely compromised and all messages were exposed, our current implementation maintains proper message privacy. Integrity protection is restored, and unauthorized modification of messages can no longer take place. System availability has increased owing to the rate limiting protections provided, which maintain service during attempts to attack.

To confirm these results, we reran the exploits and analysis conducted by the partner team. The code we ran and the analysis we performed can be found in the Attacks_Faced_And_Analysis folder in our repo. To start with, we ran the same BruteForce script that the partner group ran (just updating the url to include the https).

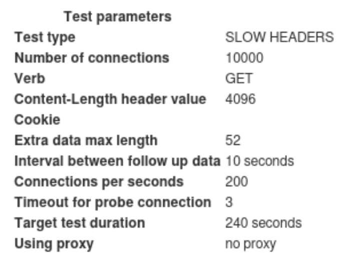


Too Many Requests

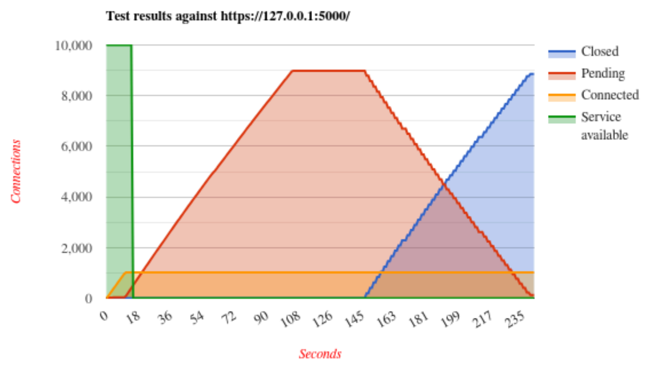
10 per 1 minute

As you can see above, due to our limits on routes as explained previously, this code quickly ran into this “too many requests” block out. This prevented the brute force attack from being able to guess the password due to the block out.

We then also ran the same slowhttptest code that the partner group ran (updating again for the URL). This is a simulation of a DoS attack, simulating 10000 connections to our application over a 240 second period. This is a very extreme attack. The results are shown below:

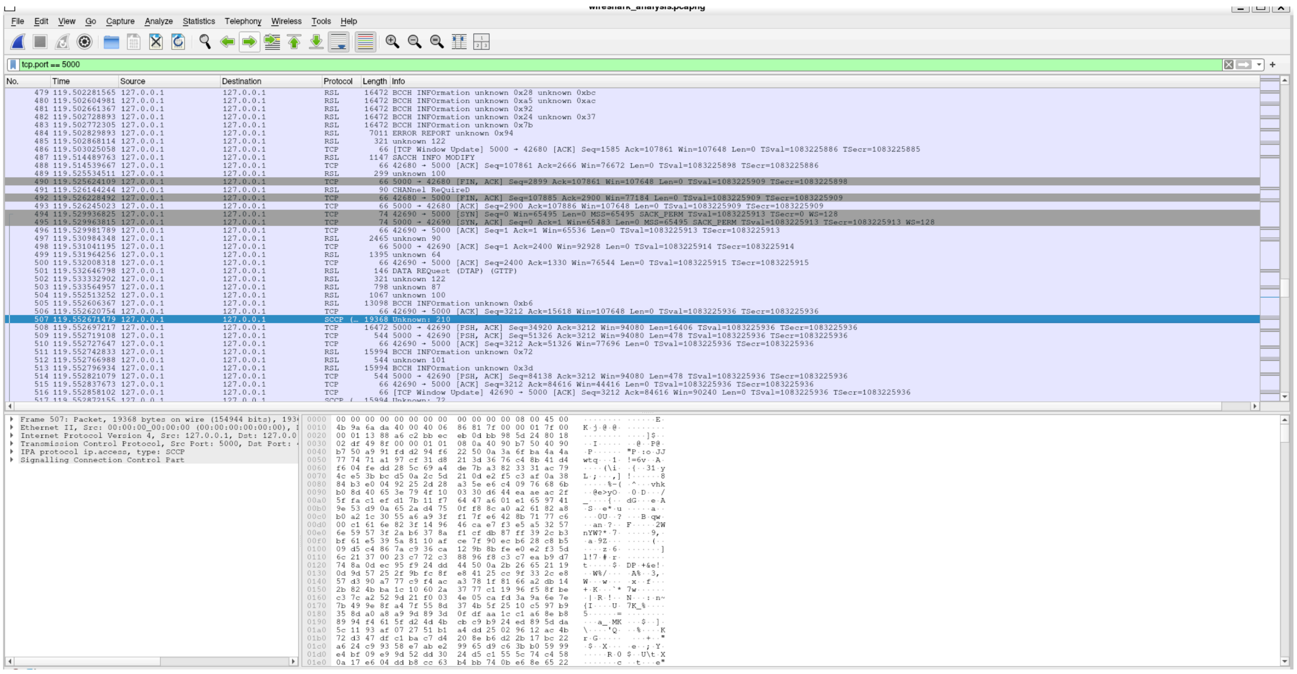


Test parameters	
Test type	SLOW HEADERS
Number of connections	10000
Verb	GET
Content-Length header value	4096
Cookie	
Extra data max length	52
Interval between follow up data	10 seconds
Connections per seconds	200
Timeout for probe connection	3
Target test duration	240 seconds
Using proxy	no proxy



Per the partner team, our previous system did not seem to properly close connections and never fully recovered even well after the test. As you can see, while the service was still taken down during the attack, it did show a marked improvement compared to the partner team's testing. Our system successfully recovered and closed out connection successfully as it slowly caught up after the attack completed. Roughly 20-30 seconds after the end of this test, the application was fully functional again (tested through curl).

For completeness, we still did a wireshark analysis on the site as we used the site to login, send messages, send attachments, and logout. A sample of what information was visible from this analysis can be seen below:


wireshark_analysis.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
tcp.port == 5000

No.  Time           Source     Destination  Protocol  Length Info
 479 119.502281565 127.0.0.1  127.0.0.1    RSL       16472 BCCH INFOrmation unknown 0x28 unknown 0xbc
 480 119.502604981 127.0.0.1  127.0.0.1    RSL       16472 BCCH INFOrmation unknown 0xa5 unknown 0xac
 481 119.502661367 127.0.0.1  127.0.0.1    RSL       16472 BCCH INFOrmation unknown 0x92
 482 119.502728893 127.0.0.1  127.0.0.1    RSL       16472 BCCH INFOrmation unknown 0x24 unknown 0x37
 483 119.502772305 127.0.0.1  127.0.0.1    RSL       16472 BCCH INFOrmation unknown 0x7b
 484 119.502829893 127.0.0.1  127.0.0.1    RSL        7011 ERROR REPORT unknown 0x94
 485 119.502868114 127.0.0.1  127.0.0.1    RSL         321 unknown 122
 486 119.503025058 127.0.0.1  127.0.0.1    TCP          66 [TCP Window Update] 5000 → 42680 [ACK] Seq=1585 Ack=107861 Win=107648 Len=0 TSval=1083225886 TSecr=1083225885
 487 119.514489763 127.0.0.1  127.0.0.1    RSL        1147 SACCH INFO MODIFY
 488 119.514539667 127.0.0.1  127.0.0.1    TCP          66 42680 → 5000 [ACK] Seq=107861 Ack=2666 Win=76672 Len=0 TSval=1083225898 TSecr=1083225886
 489 119.525534511 127.0.0.1  127.0.0.1    RSL         299 unknown 100
 490 119.525624109 127.0.0.1  127.0.0.1    TCP          66 5000 → 42680 [FIN, ACK] Seq=2899 Ack=107861 Win=107648 Len=0 TSval=1083225909 TSecr=1083225898
 491 119.526144244 127.0.0.1  127.0.0.1    RSL          90 CHANnel ReQuireD
 492 119.526228492 127.0.0.1  127.0.0.1    TCP          66 42680 → 5000 [FIN, ACK] Seq=107885 Ack=2900 Win=77184 Len=0 TSval=1083225909 TSecr=1083225909
 493 119.526245023 127.0.0.1  127.0.0.1    TCP          66 5000 → 42680 [ACK] Seq=2900 Ack=107886 Win=107648 Len=0 TSval=1083225909 TSecr=1083225909
 494 119.529936825 127.0.0.1  127.0.0.1    TCP          74 42690 → 5000 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=1083225913 TSecr=0 WS=128
 495 119.529963815 127.0.0.1  127.0.0.1    TCP          74 5000 → 42690 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM TSval=1083225913 TSecr=1083225913 WS=128
 496 119.529981789 127.0.0.1  127.0.0.1    TCP          66 42690 → 5000 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=1083225913 TSecr=1083225913
 497 119.530984348 127.0.0.1  127.0.0.1    RSL        2465 unknown 90
 498 119.531041195 127.0.0.1  127.0.0.1    TCP          66 5000 → 42690 [ACK] Seq=1 Ack=2400 Win=92928 Len=0 TSval=1083225914 TSecr=1083225914
 499 119.531964256 127.0.0.1  127.0.0.1    RSL        1395 unknown 64
 500 119.532008318 127.0.0.1  127.0.0.1    TCP          66 42690 → 5000 [ACK] Seq=2400 Ack=1330 Win=76544 Len=0 TSval=1083225915 TSecr=1083225915
 501 119.532646798 127.0.0.1  127.0.0.1    RSL         146 DATA REQuest (DTAP) (GTTP)
 502 119.533332902 127.0.0.1  127.0.0.1    RSL         321 unknown 122
 503 119.533564957 127.0.0.1  127.0.0.1    RSL         798 unknown 87
 504 119.552513252 127.0.0.1  127.0.0.1    RSL        1067 unknown 100
 505 119.552606367 127.0.0.1  127.0.0.1    RSL       13098 BCCH INFOrmation unknown 0xb6
 506 119.552620754 127.0.0.1  127.0.0.1    TCP          66 42690 → 5000 [ACK] Seq=3212 Ack=15618 Win=107648 Len=0 TSval=1083225936 TSecr=1083225936
 507 119.552671479 127.0.0.1  127.0.0.1    SCCP (…   19368 Unknown: 210
 508 119.552697217 127.0.0.1  127.0.0.1    TCP       16472 5000 → 42690 [PSH, ACK] Seq=34920 Ack=3212 Win=94080 Len=16406 TSval=1083225936 TSecr=1083225936
 509 119.552719108 127.0.0.1  127.0.0.1    TCP         544 5000 → 42690 [PSH, ACK] Seq=51326 Ack=3212 Win=94080 Len=478 TSval=1083225936 TSecr=1083225936
 510 119.552727647 127.0.0.1  127.0.0.1    TCP          66 42690 → 5000 [ACK] Seq=3212 Ack=51326 Win=77696 Len=0 TSval=1083225936 TSecr=1083225936
 511 119.552742833 127.0.0.1  127.0.0.1    RSL       15994 BCCH INFOrmation unknown 0x72
 512 119.552766988 127.0.0.1  127.0.0.1    RSL         544 unknown 101
 513 119.552796934 127.0.0.1  127.0.0.1    RSL       15994 BCCH INFOrmation unknown 0x3d
 514 119.552821079 127.0.0.1  127.0.0.1    TCP         544 5000 → 42690 [PSH, ACK] Seq=84138 Ack=3212 Win=94080 Len=478 TSval=1083225936 TSecr=1083225936
 515 119.552837673 127.0.0.1  127.0.0.1    TCP          66 42690 → 5000 [ACK] Seq=3212 Ack=84616 Win=44416 Len=0 TSval=1083225936 TSecr=1083225936
 516 119.552858102 127.0.0.1  127.0.0.1    TCP          66 [TCP Window Update] 42690 → 5000 [ACK] Seq=3212 Ack=84616 Win=90240 Len=0 TSval=1083225936 TSecr=1083225936
 517 119.552872155 127.0.0.1  127.0.0.1    SCCP (…   15994 Unknown: 72
```

```
▸ Frame 507: Packet, 19368 bytes on wire (154944 bits), 193
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), I
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0
▸ Transmission Control Protocol, Src Port: 5000, Dst Port:
▸ IPA protocol ip.access, type: SCCP
▸ Signalling Connection Control Part
```

```
0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ········ ······E·
0010  4b 9a 6a da 40 00 40 06  86 81 7f 00 00 01 7f 00   K·j·@·@· ········
0020  00 01 13 88 a6 c2 bb ec  eb 0d bb 98 5d 24 80 18   ········ ····]$··
0030  02 df 49 8f 00 00 01 01  08 0a 40 90 b7 50 40 90   ··I····· ··@··P@·
0040  b7 50 a9 91 fd d2 94 f6  22 50 0a 3a 6f ba 4a 4a   ·P······ "P·:o·JJ
0050  77 74 71 a1 97 cf 31 d8  21 3d 36 76 c4 8b 41 d4   wtq···1· !=6v··A·
0060  f6 04 fe dd 28 5c 69 a4  de 7b a3 82 33 31 ac 79   ····(\i· ·{··31·y
0070  4c e5 3b bc d5 0a 2c 5d  21 0d e2 f5 c3 af 0a 38   L·;···,] !······8
0080  84 b3 e0 04 92 25 2d 28  a3 5e e6 c4 09 76 68 6b   ·····%-( ·^···vhk
0090  b0 8d 40 65 3e 79 4f 10  03 30 d6 44 ea ae ac 2f   ··@e>yO· ·0·D···/
00a0  5f fa c1 ef d1 7b 11 f7  64 47 a6 01 e1 65 97 41   _····{·· dG···e·A
00b0  9e 53 d9 0a 65 2a d4 75  0f f8 8c a0 a2 61 82 a8   ·S··e*·u ·····a··
00c0  b0 a2 1c 30 55 a6 a9 3f  f1 7f e6 42 8b 71 77 c6   ···0U··? ···B·qw·
00d0  00 c1 61 6e 82 3f 14 96  46 ca e7 f3 e5 a5 32 57   ··an·?·· F·····2W
00e0  6e 59 57 3f 2a b6 37 8a  f1 cf db 87 ff 39 2c b3   nYW?*·7· ·····9,·
00f0  bf 61 e5 39 5a 81 10 af  ce 7f 90 ec b6 28 c8 b5   ·a·9Z··· ·····(··
0100  09 d5 c4 86 7a c9 36 ca  12 9b 8b fe e0 e2 f3 5d   ····z·6· ·······]
0110  6c 21 37 00 23 c7 72 c3  88 96 f8 c3 c7 ea b9 d7   l!7·#·r· ········
0120  74 8a 0d ec 95 f9 24 dd  44 50 0a 2b 26 65 21 19   t·····$· DP·+&e!·
0130  0d 9d 57 25 2f 9b fc 8f  e8 41 25 cc 9f 33 2c e8   ··W%/··· ·A%··3,·
0140  57 d3 90 a7 77 c9 f4 ac  a3 78 1f 81 66 a2 db 14   W···w··· ·x··f···
0150  2b 82 4b ba 1c 10 60 2a  37 77 c1 19 96 f5 8f be   +·K···`* 7w······
0160  c3 7c a2 52 9d 21 f0 03  4e 05 ca fd 3a 9a 6e 7e   ·|·R·!·· N···:·n~
0170  7b 49 9e 8f a4 7f 55 8d  37 4b 5f 25 10 c5 97 b9   {I····U· 7K_%····
0180  35 8d a0 a8 a9 9d 89 3d  0f df aa 1c c1 a6 8e b8   5······= ········
0190  89 94 f4 61 5f d2 4d 4b  cb c9 b9 24 ed 89 5d da   ···a_·MK ···$··]·
01a0  5c 11 93 af 07 27 51 b1  a4 dd 25 02 96 12 ac 4b   \····'Q· ··%····K
01b0  72 d3 47 df c1 ba c7 d4  20 8e b6 d2 2b 17 bc 22   r·G····· ····+··"
01c0  a6 24 c9 93 58 e7 ab e2  99 65 d9 c6 3b b0 59 99   ·$··X··· ·e··;·Y·
01d0  e4 bf 09 e9 9d 52 dd 30  24 d5 c1 55 5c 74 c4 58   ·····R·0 $··U\t·X
01e0  0a 17 e6 04 dd b8 cc 63  b4 bb 74 0b e6 8e 65 22   ·······c ··t···e"
```

The partner team was able to see what routes were being used as well as the plain text username and password through this analysis. However, currently our system is sending information through https and important information is being encrypted. Even the names were not discoverable through this analysis. The full analysis file can be found in our Attacks_Faced_And_Analysis folder.

We also did a bandit analysis. Our partner team previously found several issues with our code using this analysis including hard coded passwords, using the random module instead of the secrets module for more security, and having the debug flag set to true which could allow for the execution of arbitrary code. In our current analysis, these issues all resolved (gives a false positive error for hard coded passwords as it misinterprets regex). The only other issue the analysis came up with was “possible binding to all surfaces” which seems to be a result of running the application in the docker.

These validation results confirm that our security enhancements successfully addressed all the critical vulnerabilities identified during the previous testing phase.

Updated Security Goals

Confidentiality

***Updated*-information is now encrypted when passing through the application**

- Users should only be able to view the messages sent by them to others or messages sent to them by others.
    - Message history will show older messages sent to and from the user.

- Users should be able to upload and share small files with whom they want to, and the other users shouldn't be able to tell or see what the contents of the file are.
    - The file upload and sharing will be done safely with encryption so the contents are not seen by a third party.
- Users should only be able to see groups they are invited to and accepted into.
    - Groups will have a list of members that are in them that can't be seen by users outside of the group.
- Only group owners should be able to delete the groups, message owners should be able to delete their messages, and account owners should be able to delete their accounts.
    - Only authorized users can see and delete groups, messages and accounts that will be the owners of all.
- ***Updated*** Only users should be able to have access to their own passwords. ***Passwords are stored hashed inside the database. Brute force attempts get circumvented with route limiting.***

Integrity

- File upload will be checksumed and checked on arrival
    - Checksum will be calculated and sent with the file and will be checked on arrival to make sure that the file wasn't tampered with.
- Users won't be able to edit or delete messages sent by others.
    - Messages should not be editable or deletable by people who didn't send them.
- Users outside of the group should not be able to change the users in a group
    - Only users inside the group can leave or join a group.
    - Only the group owner can invite new people to the group.
    - Users on the outside of the group can't join or tamper with the users inside the group.

Availability

- Users should be able to send messages at all times to whoever they want
    - Messaging is available to users whenever they require it.
- Users should be able to upload and send files at all times to whoever they want.
    - File uploading and sharing are available to users whenever they require.
- Users outside of a group can't change group settings and members
    - Users outside of the group can't see the member inside, and the group is not available to them unless the group owner invites them
- ***Updated*** If the application does go down, it should be able to recover and return to full availability as the attack passes. We currently have route limiting implemented to prevent attackers from sending unlimited requests.

Security Metrics

Confidentiality -

- Percentage of messages that are viewed by a user other than the intended recipient or recipients. Note that this also includes messages in groups that are either viewable by users not in that group OR messages in groups that are viewable by users in that group, but who became active users (accepted invites) of that group after those messages had been sent (as users should not have access to messages from before they joined the group). Users should also not have access to group messages once they leave a group.
    - Target will be 0% and will be greater than 0% if any vulnerabilities are discovered
- ***Updated*** Percentage of passwords or login codes that attackers can access due to brute force or database leaks. Any passwords stored as plain text is unacceptable.
    - Target will be 0% and will be greater than 0% if any passwords are exposed (to attackers or even to admins)

Integrity -

- Percentage of messages that have their content modified in any way (either by accident, through malicious means, or through failures/bugs) after being sent by a sender
    - Target will be 0% and will be greater than 0% if any vulnerabilities are discovered
- Percentage of messages that have their sender field modified in any way (either by accident, through malicious means, or through failures/bugs) after being sent by a sender to appear as if they are coming from another user
    - Target will be 0%, and will be greater than 0% if any vulnerabilities are discovered

Availability

- Server downtime percentage
    - ***Updated*** Target will be **5%**. It must be able to recover on its own after a DoS attack and return to full functionality
- Percentage of server database updates that fail to complete
    - Target will be 0% and will be greater than 0% if the database does not properly update after applicable user actions (sending messages, uploading files, deleting messages, audit logging, etc.). for any reason
- Percentage of database deletions done by attackers
    - Target will be 0% and will be greater than 0% if an unauthorized user can exploit a vulnerability to delete elements from the database